KIRKLARELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

YAZ20411
DERİN ÖĞRENME

Hazırlayan: Dr. Öğr. Üyesi Fatih BAL

Kasım 2025

İÇİNDEKİLER

BÖLÜM - I

1. YAPAY ZEKÂ NEDİR?

Yapay Zekâ (YZ-AI) kavramının bir tanımını yapmadan önce Zihin, Zekâ ve Akıl tanımlamalarını yapmak gerekir. Bu sayede, yapacağımız tanımlama akıllarda daha iyi oturmuş olacaktır. ***Zihin*** ya da ***bilinç***; düşünme, algılama, hatırlama, öğrenme ve problem çözme gibi bilişsel süreçleri içeren karmaşık bir kavramdır. Zihin, insanların ve diğer organizmaların bilgiyi işlemelerine ve çevreleriyle etkileşimde bulunmalarına yardımcı olan bir süreçler topluluğunu ifade etmektedir. Zihin, düşünme yeteneklerini, duygusal tepkileri, belleği ve farkındalığı içerir. ***Zekâ***, insanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama ve sonuç çıkarma yeteneklerinin tümüdür. Zihnin öğrenme, öğrenilenden yararlanabilme, yeni durumlara uyabilme ve yeni çözüm yolları bulabilme yeteneğini ifade etmektedir. En geniş anlamıyla, genel zihin gücü olarak da tanımlanabilir. Zihnin algılama, bellek, düşünme, öğrenme gibi birçok işlevini içerir. Doğruyla yanlışı, yalan ile gerçeği ayırt edebilmemizi sağlayan bir düşünme ve kavrama yetisidir. ***Akıl***, aklımız sayesinde olayları değerlendirebilir ve görüşlerimizi ifade edebiliriz. Ve yine aklımız sayesinde verdiğimiz kararların sonuçlarını öngörür ve sorumluluğunu alabiliriz.

Zihin, Zekâ ve Akıl kavramını şöyle örneklendirebiliriz. Yeni doğmuş ve sağlıklı bir bebek düşünün. Bebek, yenidoğan sürecini tamamladıktan sonra bir şeyleri keşfetmeye başlar. Tanımaya başlar. Gördüğü yeni nesneleri algılamaya başlar ve hafızasına kaydeder. Bu süreç, Zihin sürecini oluşturur. Algılama süreci bittikten sonra, algıladığı nesnelerden sonuçlar üretmeye başlar. Koltuk üzerinden koltuk nesnesini tanır, kavrar. Ancak ilk zamanlarda koltuktan inemez. Zamanla algıladığı koltuktan nesnesinden nasıl ineceğini öğrenmek için Zekâsını devreye sokar. Zamanla inip inemeyeceğini kavrar. Aklını kullanarak tüm bu süreci kontrol eder. Koltuktan inmemenin tehlikeli olduğunu fark eder.

Yapay Zekâ, insanlara ait tüm bu sürecin makineye kazandırılması sürecidir. Sadece insanlara değil, tüm canlı varlıklara ait özelliklere kazandırma sürecini de kapsar. Ancak, insanı diğer canlılardan ayıran en belirgin özelliğin, yani düşünme ve karar verme yeteneğinin makineye kazandırılmasına yapay zekâ olarak tanımlayabiliriz. Teknik bir ifade ile tanımlaması yapılırsa, Yapay Zekâ, insan davranışını taklit edebilen, akıllı makineler oluşturmayı hedefleyen bir bilgisayar bilimi teknolojisidir. *Farklı bir ifadeyle*, Yapay Zekâ, kendisine verilen görevleri yerine getirmek için insan zekâsını taklit eden ve topladıkları bilgiye göre kendilerini iyileştiren sistemler veya makineler anlamına gelir. Yapay Zekâ, herhangi bir özel biçim veya işlevden ziyade süper güçlendirilmiş düşünce ve veri analizi yeteneği ve süreciyle ilgilidir. Yapay Zekâ

dendiğinde zihinlerde dünyayı ele geçiren çok fonksiyonel, insan benzeri robotlar canlansa da yapay zekâ insanların yerine geçmek üzere tasarlanmamıştır. İnsan yeteneklerini ve katkılarını önemli ölçüde geliştirmek üzere tasarlanmıştır. Bu nedenle oldukça değerli bir ticari varlıktır.

2. NEDEN YAPAY ZEKÂ?

Peki, neden Yapay Zekâya ihtiyaç duyuldu? Maddeler halinde açıklamak gerekirse,

1. Yapay Zekâ, tekrarlayan öğrenme ve verisel keşifleri otomatik hale getirmektedir. Yapay Zekâ, manuel görevleri otomatikleştirmek yerine sık, yüksek hacimli, bilgisayarlı görevleri güvenilir bir şekilde ve yorulmadan gerçekleştirir. Bu tür bir otomasyon için, sistemi kurmak ve doğru soruları sormak adına insan gücü hala gereklidir.
2. Yapay Zekâ, hâlihazırda var olan ürünlere zekâ eklemektedir. Çoğu durumda, yapay zekâ bireysel bir uygulama olarak satılmayacaktır. Bunun yerine, hâlihazırda kullandığınız ürünler, Siri'nin yeni nesil Apple ürünlerine bir özellik olarak eklenmesi gibi, AI yetenekleriyle geliştirilecektir. Otomasyon, konuşma platformları, botlar ve akıllı makineler, güvenlik istihbaratından yatırım analizine kadar evde ve işyerinde birçok teknolojiyi iyileştirmek için büyük miktarda veriyle birleştirilebilir.
3. Yapay Zekâ, verilerde yapı ve düzenlilik bulur, böylece algoritma bir beceri kazanmaktadır: Sınıflandırma veya Tahminleme. Yani, algoritma nasıl satranç oynanacağını kendi kendine öğretebildiği gibi, bir sonraki ziyaretinde kişiye hangi ürünü önereceğini kendi kendine öğretebilir ve modeller yeni veriler geldiğinde de buna uyum sağlar. Geri yayılma, modelin, ilk yanıt tam olarak doğru olmadığında, eğitim ve eklenmiş veriler yoluyla ayarlamasını sağlayan bir yapay zekâ tekniğidir.
4. Yapay Zekâ, birçok gizli katmana sahip sinir ağlarını kullanarak daha fazla ve daha derin verileri analiz eder. Beş gizli katmana sahip bir sahtekârlık tespit sistemi kurmak birkaç yıl önce neredeyse imkânsızdı. Tüm bunlar inanılmaz bir bilgisayar gücü ve büyük veri Doğrudan veriden öğrendiklerinden, derin öğrenme modellerini eğitmek için çok sayıda veriye ihtiyacınız var. Onları ne kadar çok veriyle beslerseniz, o kadar doğru olurlar.
5. Yapay Zekâ, derin sinir ağları sayesinde önceden neredeyse imkânsız olan bir doğruluk ile çalışmaktadır. Google Aramaları ve Google Fotoğraflar ile etkileşimlerinizin tümü derin öğrenmeye dayalıdır- ve biz onları kullandıkça daha doğru olmaya devam ederler. Tıp alanında, derin öğrenme, görüntü sınıflandırma ve nesne tanıma gibi yapay zekâ

teknikleri, artık yüksek eğitimli radyologlarla aynı doğrulukla MRI'larda kanseri bulmak için kullanılabilir.

NOT:
4. ve 5. Maddelerde yer alan Geri Yayılma, Gizli Katman, Derin Veri, Sinir Ağları ve Derin Sinir Ağları gibi kavramların tanımları ilerleyen bölümlerde daha detaylı anlatılacaktır.

3. YAPAY ZEKÂ TARİHÇESİ

Yapay zekânın fikir babası Alan Turing'dir.. İngiliz matematikçi Alan Turing'in 1936'da geliştirdiği Turing Makinesi ile bağlantılıdır. Bu teorik model, hesaplama ve mantıksal düşünme konularına temel oluşturmuştur. 1950'lerde «***Makineler Düşünebilir mi***?» sorusuna yanıt aramıştır. «Yapay Zekâ» terimini ilk olarak 1956 yılında İngiltere'de New Hampshire'da bulunan Darmouth College'da yapılan çalıştayda John McCarthy tarafından kullanılmıştır.

Türkiye'de 1959 yılında Erzurum Atatürk Üniversitesinde düzenlenen konferansta **Ord. Prof. Dr. Cahit ARF** tarafından «***Makine Düşünebilir Mi ve Nasıl Düşünebilir***» isimli çalışma sunmuştur. Yapay Zekânın tarihçesini derinlemesine incelersek;

Temeller (II. Dünya Savaşı Dönemi) (1940'lar): Yapay Zekâ çalışmaları, savaş sırasında şifre kırma ve stratejik planlama gibi askeri uygulamalar için kullanılmıştır. Bu dönemde, ilk dijital bilgisayarlar geliştirilmiştir. Bu dönem yapay zekânın felsefi ve matematiksel temellerinin atıldığı dönemdir.

i. 1943 – McCulloch & Pitts, yapay sinir ağlarının matematiksel modelini geliştirdi. Bu, "nöronların" bilgisayarda temsil edilebileceğini gösterdi.
ii. 1950 – Alan Turing, "Computing Machinery and Intelligence" makalesinde "Makineler düşünebilir mi?" sorusunu ortaya attı ve Turing Testi'ni önerdi.
iii. 1951 – İlk öğrenen programlar yazıldı (dama ve satranç oynayan basit yazılımlar).

Doğuş (1956-1960'lar): Yapay Zekâ terimi ilk kez John McCarthy, Marvin Minsky, Nathaniel Rochester ve Claude Shannon gibi bilim adamları tarafından kullanıldı. Dartmouth Konferansı, Yapay Zekâ alanının resmi olarak kurulduğu yer olarak kabul edilmektedir. Frank Rosenblatt'ın Perceptron adlı basit bir sinir ağı modeli geliştirmesi, YZ'ye büyük bir ilgi getirmiştir. Ancak, Perceptron'un sınırlamaları ve basitlikleri nedeniyle YZ çalışmaları bir durgunluğa girmiştir. Bu dönem, yapay zekânın resmi bir bilim dalı olarak ortaya çıktığı dönemdir.

i. 1956 – Dartmouth Konferansı'nda John McCarthy, Marvin Minsky, Allen Newell, Herbert Simon gibi öncüler "Artificial Intelligence" (Yapay Zekâ) terimini ilk kez kullandı.
ii. 1958 – John McCarthy, LISP programlama dilini geliştirdi (yapay zekâ araştırmalarında uzun yıllar standart oldu).
iii. 1960'lar – İlk uzman sistemler (satranç programları, mantıksal çıkarım yapan yazılımlar) geliştirildi.

İlk Umutlar ve Sınırlar (Yapay Zekâ Krizi) (1970'ler - 1980'ler): Bu dönemde, uzman sistemler gibi yazılım tabanlı Yapay Zekâ uygulamaları geliştirilmeye başlanmıştır. Uzman sistemler, belirli bilgi alanlarında insan uzmanları gibi kararlar alabilen bilgisayar programlarına verilen genel bir isim olmuştur. Bu dönem (70'ler) **"ilk yapay zekâ kışı"** olarak bilinir; fonlar azaldı, çalışmalar yavaşladı.

1970'ler

i. İlk heyecan yerini sınırlamalara bıraktı.
ii. Bilgisayarların işlem gücü yetersizdi, veri eksikliği büyüktü.
iii. Beklentiler çok yüksekti ama sonuçlar sınırlıydı.

1980'ler

iv. Uzman Sistemler: İnsan uzmanların bilgilerini kurallar şeklinde bilgisayarlara aktaran sistemler (ör. MYCIN – tıp alanında).
v. Japonya'da **Beşinci Nesil Bilgisayar Projesi** başlatıldı (1982).
vi. Donanım gelişmeleriyle birlikte yapay zekâ tekrar yükselişe geçti.
vii. Yapay zekâ, **endüstride ilk kez somut uygulamalarla** kullanılmaya başlandı.

İkinci Yapay Zekâ Kışı (1990'lar başları): Uzman sistemler pahalıydı, bakımı zordu, ölçeklenemedi. Beklenen devrim gerçekleşmedi ve birçok proje rafa kaldırıldı. Yapay zekâ tekrar gözden düştü.

Öğrenmenin Yükselişi (1990'lar 2000'ler başları): Makine Öğrenmesi öne çıktı: İstatistiksel yöntemler ve algoritmalarla bilgisayarların verilerden öğrenmesi sağlandı.

i. 1997 – IBM'in Deep Blue satranç bilgisayarı, dünya şampiyonu Garry Kasparov'u yendi.
ii. İnternet ve dijitalleşme sayesinde büyük veri (Big Data) ortaya çıktı.
iii. Yapay zekâ, veri temelli öğrenme ve istatistiksel modellemeye yöneldi.

Derin Öğrenme (2010'lar):

i. 2012 – AlexNet adlı derin öğrenme modeli, ImageNet yarışmasında büyük farkla birinci oldu. Bu, derin sinir ağlarının gücünü gösterdi.
ii. 2016 – Google DeepMind'in AlphaGo sistemi, Go oyununda dünya şampiyonunu yendi. Go, satrançtan çok daha karmaşık kabul ediliyordu.
iii. Doğal dil işleme (NLP), bilgisayarla görme ve otonom araçlarda yapay zekâ devrim yarattı.
iv. GPU'lar, büyük veri ve derin öğrenme bu dönemin temelini oluşturdu.

Günümüz (2020'ler):

i. Transformers (2017 sonrası): BERT, GPT, ChatGPT gibi modeller, insan dilini anlama ve üretmede çığır açtı.
ii. Görüntü, ses, metin ve çoklu veriyi birleştiren çoklu-modelli (multimodal) yapay zekâ ortaya çıktı.
iii. Sağlık, finans, eğitim, tarım, güvenlik gibi birçok sektörde yapay zekâ aktif olarak kullanılıyor.
iv. Etik tartışmalar (önyargılar, veri gizliliği, iş kaybı, insan–makine ilişkisi) giderek önem kazandı.
v. Yapay zekâ artık sadece araştırma değil, gündelik yaşamın ayrılmaz bir parçası haline geldi.

4. YAPAY ZEKÂNIN ALANLARI/ALT KÜMELERİ NELERDİR?

Yapay Zekânın alt alanları ve alt kümeleri şunlardır:

Makine Öğrenimi: Açıkça programlanmadan kendi kendilerine birkaç numara öğrenebilmeleri için bilgisayarları veri besleyerek harekete geçirme bilimidir.

Derin öğrenme bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Yani en az bir adet yapay sinir ağının kullanıldığı ve birçok algoritma ile bilgisayarın eldeki verilerden yeni veriler elde etmesidir

Doğal Dil İşleme: Doğal Dil İşleme (NLP), sorunları çözmek için yararlı içgörüler elde etmek için doğal insan dilini analiz eden Yapay Zekâ yöntemini ifade eder.

Sinir Ağları: İnsan beynine göre modellenmiş bir dizi algoritma ve tekniktir. Sinir Ağları, karmaşık ve gelişmiş makine öğrenimi sorunlarını çözmek için tasarlanmıştır.

Bulanık Mantık Sistemleri: Bulanık mantık, modern bilgisayarın dayandığı olağan "doğru veya yanlış" (1 veya 0) boole mantığından ziyade "doğruluk derecelerine" dayalı bir hesaplama yaklaşımıdır. Bulanık mantık Sistemleri, kesin olmayan, bozuk, gürültülü giriş bilgilerini alabilir.

Uzman Sistemler: Bir uzman sistem, bir insanın karar verme yeteneğini taklit eden bir bilgisayar sistemidir. Belirli bir alanda uzman bilgi ve deneyime sahip bir insan veya kuruluşun yargı ve davranışlarını simüle etmek için yapay zekâ (YZ) teknolojilerini kullanan bir bilgisayar programıdır.

Robotik: Robotik, robotların farklı dallarını ve uygulamalarını içeren yapay zekânın bir alt kümesidir. Bu Robotlar, gerçek dünya ortamında hareket eden yapay etkenlerdir. Bir AI Robot, çevresindeki nesneleri algılayarak, hareket ettirerek ve ilgili eylemleri gerçekleştirerek manipüle ederek çalışır.

5. YAPAY ZEKÂ TEKNOLOJİ GEREKSİNİMLERİ NELERDİR?

Yapay Zekâ geliştirmesinde en önemli konu, Yapay Zekâ uygulamalarının hangi programlama dilleri ile geliştirileceğidir. YZ uygulamalarının geliştirilmesinde en çok ve en yaygın kullanılan programlama dilleri şunlardır.

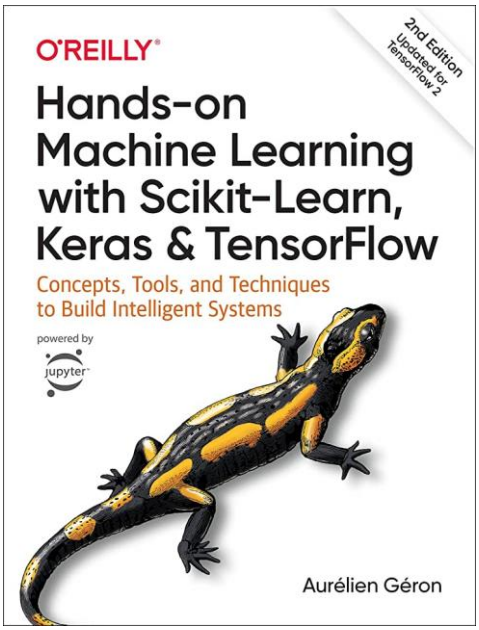
- Python Programlama
- R Programlama
- C++ Programlama
- Scala Programlama
- Julia Programlama
- Rust Programlama
- Java Programlama
- Lisp Programlama
- Haskell Programlama

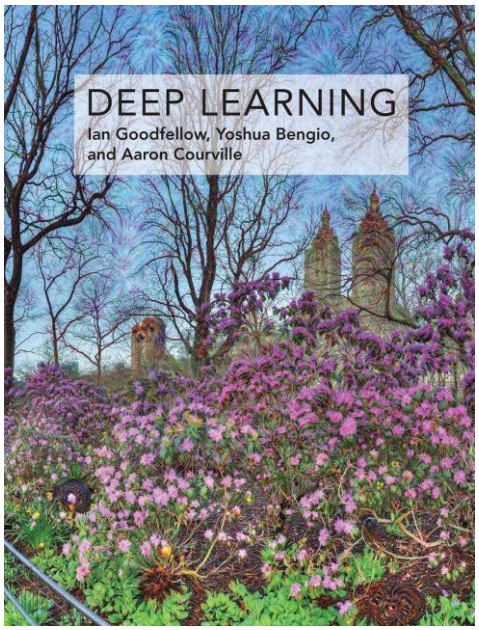
Bu programlama dilleri, kişisel bilgisayarlarda kullanıldığı gibi aşağıda yer alan Yazılım Geliştirme Platformlarında da kullanılabilmektedir.

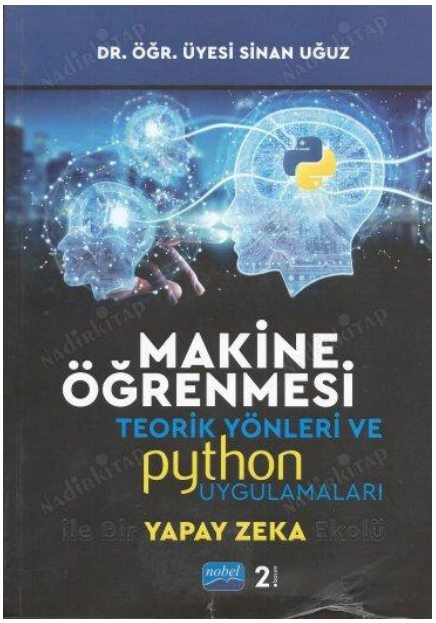
- Google Cloud AI Platfromu
- Microsoft Azure AI Platformu
- Amazon Cloud Computing Services
- Rainbird
- IBM Watson
- Google Colaboratory
- Jupyter
- Spyder
- PyCharm

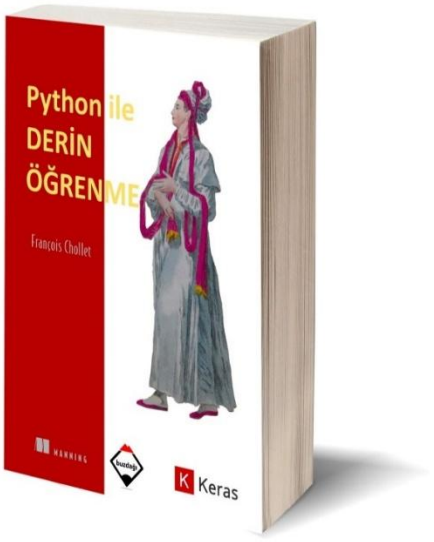
NOT:
Derin Öğrenme dersi süresi boyunca, yapay zekâ uygulamalarını «**Python Programlama**» dili kullanarak «**Google Colaboratoy**» platformu üzerinde geliştireceğiz.

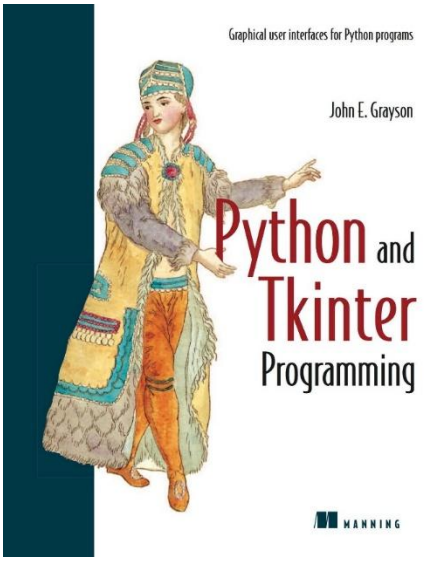
Derin Öğrenme dersi boyunca, faydalanacağımız kitaplar ise şunlardır:

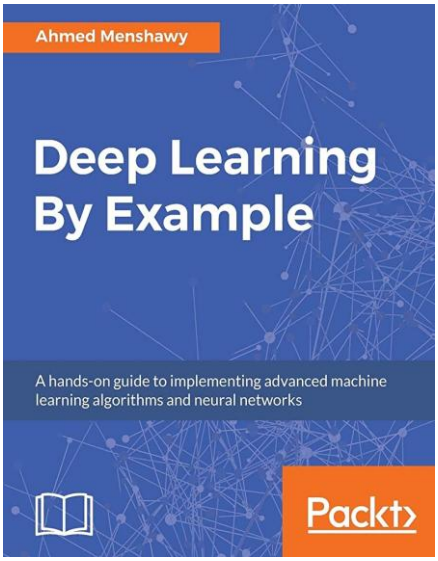


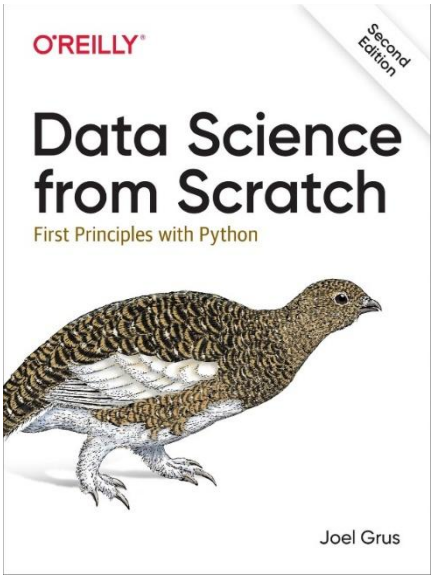



DR. ÖĞR. ÜYESİ SİNAN UĞUZ
MAKİNE ÖĞRENMESİ
TEORİK YÖNLERİ VE
python
UYGULAMALARI
YAPAY ZEKA
nobel
2


Python ile
DERİN
ÖĞRENME
François Chollet
MANNING
Keras


Graphical user interfaces for Python programs
John E. Grayson
Python and
Tkinter
Programming
MANNING


Ahmed Menshawy
Deep Learning
By Example
A hands-on guide to implementing advanced machine learning algorithms and neural networks
Packt


O'REILLY
Second Edition
Data Science
from Scratch
First Principles with Python
Joel Grus

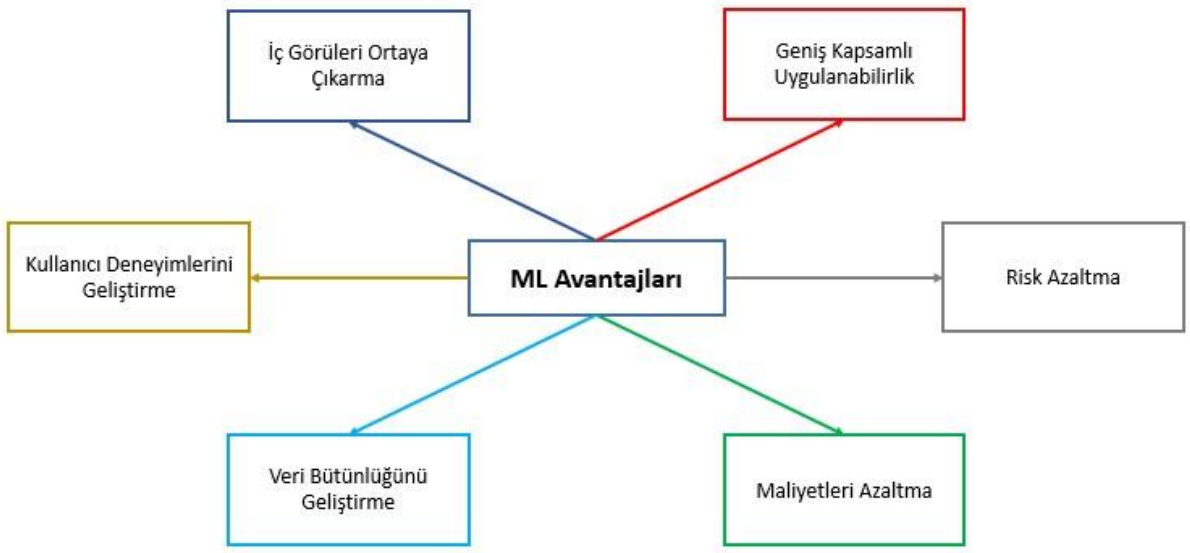
BÖLÜM - II

1. MAKİNE ÖĞRENMESİ NEDİR?

Makine Öğrenimi (ML), bilgisayar sistemlerinin verileri analiz etmeyi, öğrenmeyi ve kararlar almaya yetenek kazanmasını sağlayan bir yapay zekâ alt alanıdır. Diğer bir ifadeyle, bilgisayarların belirli görevleri, verileri ve deneyimleri kullanarak öğrenme yeteneğine sahip olduğu bir yapay zekâ dalıdır. Makine öğrenmesi, bir algoritma veya modelin belirli bir görevi, veriye dayalı deneyimlerle öğrenmesini sağlar ve bu deneyimlere dayalı olarak gelecekteki verilere yanıt verebilir. Daha basit bir ifadeyle makine öğrenimi, bir algoritma veya yöntem kullanarak ham verilerden örüntüler çıkaran bir yapay zekâ türüdür.

Makine öğrenmesinin temel amacı, verilere dayalı olarak kalıpları tanımak ve bu kalıpları kullanarak veri analizi, tahmin, sınıflandırma ve öneri gibi görevleri otomatikleştirmektir. Bilgisayar sistemlerinin açıkça programlanmadan veya insan müdahalesi olmadan deneyimlerden öğrenmesini sağlamaktır.

1. 1. MAKİNE ÖĞRENMESİNİN AVANTAJLARI NELERDİR?

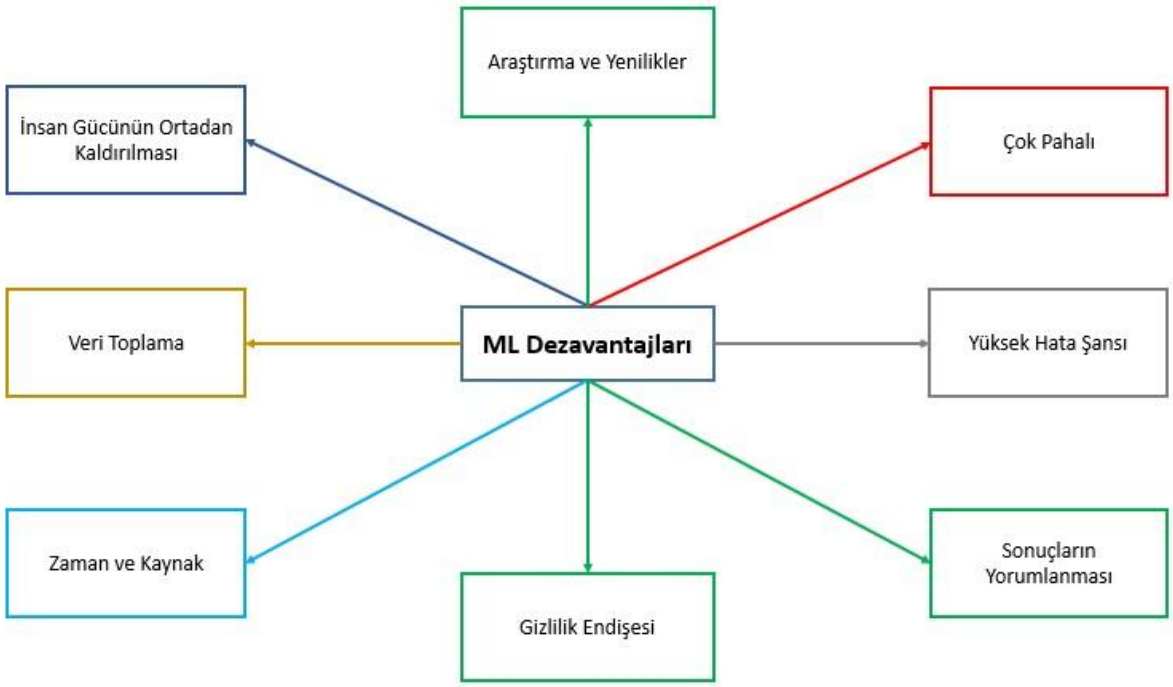


Şekil 1. Makine Öğrenmesinin Avantajları

Şekil 1, Makine Öğrenmesinin sağladığı avantajları göstermektedir. Bu avantajları daha detaylı inceleyecek olursak;

1. **Kullanıcı Deneyimini Geliştirme**: Uyarlamalı arabirimler, hedefli içerikler, sohbet botları ve sesli sanal yardımcılar, makine öğrenmesinin müşteri deneyimini iyileştirmeye nasıl yardımcı olabileceğini gösteren örneklerdir.
2. **İçgörüleri Ortaya Çıkarma:** Makine öğrenmesi, hem yapılandırılmış hem de yapılandırılmamış verilerde desenin tanımlanmasına ve verilerin anlattığı hikâyelerin belirlenmesine yardımcı olabilmektedir.
3. **Veri Bütünlüğünü Geliştirme:** Veri madenciliği konusunda oldukça iyi olan makine öğrenmesi, bunu bir adım ileriye taşıyarak zamanla özelliklerini geliştirebilmektedir.
4. **Maliyetleri Azaltma:** Bir makine öğrenmesi uygulaması, zamandan ve kaynaklardan tasarruf sağlayarak ekibinizin en önemli şeylere odaklanmasına olanak tanıyan süreç otomasyonudur.
5. **Risk Azaltma:** Dolandırıcılık taktikleri sürekli olarak değiştikçe, makine öğrenmesi de buna ayak uydurmaktadır. Makine öğrenmesi, dolandırıcılık denemeleri başarılı olmadan önce bunları yakalamak için yeni desenleri izlemekte ve belirlemektedir.
6. **Geniş Kapsamlı Uygulanabilirlik:** Bu teknoloji çok geniş bir uygulama yelpazesine sahiptir. Makine öğrenimi konaklama, eğitim teknolojisi, tıp, bilim, bankacılık ve işletme gibi hemen hemen her alanda rol oynamaktadır. Daha fazla fırsat yaratmaktadır.

1. 2. MAKİNE ÖĞRENMESİNİN DEZAVANTAJLARI NELERDİR?



Şekil 2. Makine Öğrenmesinin Dezavantajları

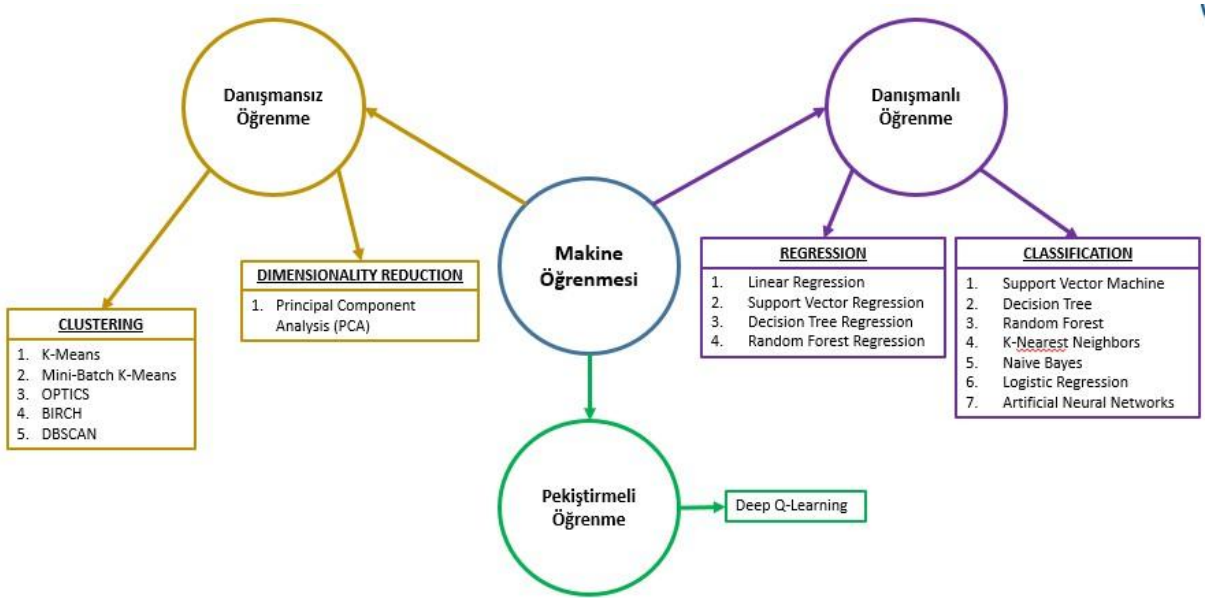
Şekil 2, Makine Öğrenmesinin sağladığı dezavantajları göstermektedir. Bu avantajları daha detaylı inceleyecek olursak;

1. **Veri Toplama:** Tüm makine öğrenimi kavramı, yararlı verileri tanımlamakla ilgilidir. Güvenilir bir veri kaynağı sağlanmazsa sonuç yanlış olacaktır. Verilerin kalitesi de önemlidir. Kullanıcı veya kurumun daha kaliteli veriye ihtiyacı varsa bekleyin. Çıktının sağlanmasında gecikmelere neden olacaktır. Bu nedenle, makine öğrenimi önemli ölçüde verilere ve kalitesine bağlıdır.
2. **Zaman ve Kaynaklar:** Makinelerin işlediği veriler, miktar olarak çok büyük kalır ve büyük ölçüde farklılık gösterir. Makineler, algoritmalarının ortama uyum sağlayabilmesi ve onu öğrenebilmesi için zamana ihtiyaç duyar. Makinenin doğruluğunu ve güvenilirliğini kontrol etmek için deneme çalışmaları yapılır. Bu kalitede bir altyapı oluşturmak için çok büyük ve pahalı kaynaklar ile yüksek kaliteli uzmanlık gerekir. Deneme çalıştırmaları, zaman ve masraflar açısından maliyetli olacağı için maliyetlidir.
3. **Sonuçların Yorumlanması:** Makine öğreniminin en büyük avantajlarından biri, makineden aldığımız yorumlanmış verilerin yüzde yüz doğru olamamasıdır. Bir

dereceye kadar yanlışlık olacaktır. Yüksek doğruluk derecesi için, güvenilir sonuçlar verecek algoritmalar geliştirilmelidir.

4. **Yüksek Hata Şansı:** İlk aşamalarda yapılan hata çok büyük ve o anda düzeltilmezse ortalığı kasıp kavuruyor. Taraflılık ve yanlışlık ayrı ayrı ele alınmalıdır; birbirine bağlı değiller. Makine öğrenimi, veri ve algoritma olmak üzere iki faktöre bağlıdır. Tüm hatalar iki değişkene bağlıdır. Herhangi bir değişkendeki herhangi bir yanlışlığın çıktı üzerinde büyük etkileri olacaktır.
5. **İnsan Gücünün Ortadan Kaldırılması:** Otomasyon, Yapay Zeka ve Makine Öğrenimi bazı işlerde insan arayüzünü ortadan kaldırdı. İstihdam olanaklarını ortadan kaldırdı. Artık tüm bu çalışmalar yapay zeka ve makine öğrenimi yardımıyla yapılıyor.
6. **Çok Pahalı:** Bu yazılım oldukça pahalıdır ve herkes buna sahip olamaz. Devlet kurumları, büyük özel firmalar ve işletmeler çoğunlukla sahibidir. Geniş kullanım için herkesin erişimine açık hale getirilmesi gerekiyor.
7. **Gizlilik Endişesi:** Bildiğimiz gibi, makine öğreniminin temel direklerinden biri veridir. Verilerin toplanması temel gizlilik sorununu gündeme getirdi. Verilerin toplanma ve ticari amaçlarla kullanılma şekli her zaman tartışmalı bir konu olmuştur. Hindistan'da, Hindistan Yüksek Mahkemesi mahremiyetin Hintliler için temel bir hak olduğunu ilan etti. Kullanıcının izni olmadan veriler toplanamaz, kullanılamaz veya saklanamaz. Ancak, büyük firmaların kullanıcının bilgisi olmadan verileri topladığı ve ticari kazançları için kullandığı birçok durum ortaya çıkmıştır.
8. **Araştırma ve Yenilikler:** Makine öğrenimi gelişen bir kavramdır. Bu alan henüz herhangi bir ekonomik sektörde tamamen devrim yaratan herhangi bir büyük gelişme görmedi. Alan sürekli araştırma ve yenilik gerektirir.

1. 3. MAKİNE ÖĞRENMESİ TÜRLERİ NELERDİR?



Şekil 3. Makine Öğrenmesinin Türleri

Şekil 3, Makine Öğrenmesinin türlerini içeren bir gösterimdir. Detaylı inceleyecek olursak;

- **Denetimli Öğrenme (Supervised Learning):** Denetimli öğrenmede, bir modelin eğitimi için etiketli (labelled) veriler kullanılmaktadır. Model, girdi verileri ile bu verilere karşılık gelen çıktıları (etiketleri) ilişkilendirmeyi öğrenmektedir. Örneğin, bir spam filtresi eğitimi, e-posta verileriyle "spam" ve "spam değil" etiketleri içerir. Denetimli öğrenme modelleri, sınıflandırma (classification) ve regresyon (regression) gibi görevlerde kullanılmaktadır.
- **Denetimsiz Öğrenme (Unsupervised Learning):** Denetimsiz öğrenmede, verilerin etiketlenmemiş olduğu bir senaryo söz konusudur. Model, verilerdeki gizli kalıpları veya yapıları bulmaya çalışır. Örnek olarak kümeleme (clustering) ve boyut indirgeme (dimensionality reduction) işlemleri verilebilir.
- **Pekiştirmeli Öğrenme (Reinforcement Learning):** Pekiştirmeli öğrenme, amaca yönelik ne yapılması gerektiğini öğrenen bir makine öğrenmesi yaklaşımıdır. Pekiştirmeli öğrenmede ajan (agent) adı verilen öğrenen makinemiz karşılaştığı durumlara bir tepki verir ve bunun karşılığında da sayısal bir ödül sinyali almaktadır. Ajan/öğrenen makine aldığı bu ödül puanını maksimuma çıkartmak için çalışır. Bu şekilde çalışan deneme yanılma yöntemi, pekiştirmeli öğrenmenin en ayırt edici özelliğidir.

BÖLÜM - III

1. LINEAR REGRESSION

Linear Regression (Doğrusal regresyon veya **basit doğrusal regresyon),** iki değişken arasındaki doğrusal ilişkiyi inceleyen istatistiksel yöntemlerdir. Farklı bir ifadeyle, iki değişken arasındaki ilişkiye dayanarak birinin değerini diğerinden tahmin etmeyi sağlayan bir denklem oluşturup tahmin yapmayı sağlar. Farklı bir ifadeyle, Lineer Regresyon, bağımlı bir değişkenin (hedef değişken) bir veya daha fazla bağımsız değişkenle ilişkisini modellemek için kullanılmaktadır. Lineer Regresyon, bu ilişkinin doğrusal bir denklemle ifade edilebileceği varsayımına dayanmaktadır.

Lineer regresyon üç ana türde karşımıza çıkmaktadır:

1. **Simple Linear Regression (Basit Lineer Regresyon):** Bu tür, bir bağımlı değişkenin sadece bir bağımsız değişkenle ilişkisini modellemektedir. Denklemi genellikle şu şekildedir:

$$y = \beta_0 + \beta_1 * x \qquad \text{Denklem (1)}$$

2. **Multiple Linear Regression (Çoklu Lineer Regresyon):** Bu tür, bir bağımlı değişkenin birden fazla bağımsız değişkenle ilişkisini modellemek için kullanılmaktadır. Denklemi genellikle şu şekildedir:

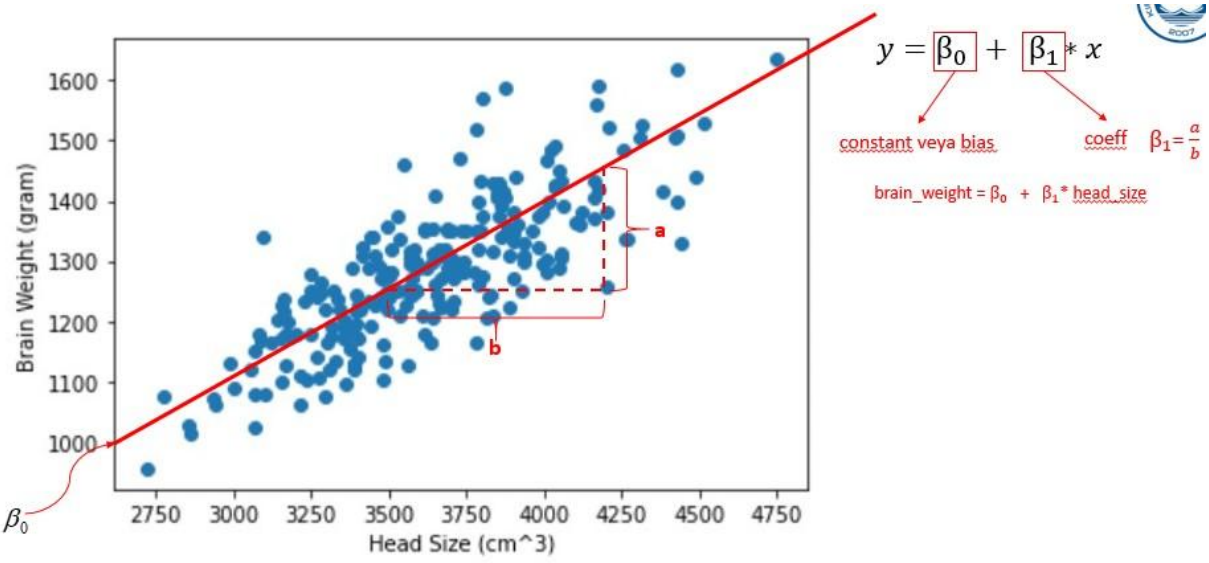
$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n \qquad \text{Denklem (2)}$$

3. **Polynomial Linear Regression (Polinom Lineer Regresyon):** Lineer regresyonun bir türüdür ve bağımlı değişken ile bağımsız değişkenler arasındaki ilişkiyi modellemek için kullanılır. Ancak, polinom lineer regresyon, bağımsız değişkenlerin ve bağımlı değişkenin arasındaki ilişkiyi doğrusal olmayan bir denklemle ifade etmek için polinomlar kullanmaktadır. Genellikle birinci veya ikinci dereceden polinomlar kullanılır, ancak daha yüksek dereceden polinomlar da kullanılabilir. Denklemi genellikle şu şekildedir:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2^2 + ... + \beta_n * x_n^n \qquad \text{Denklem (3)}$$

Denklemlerde yer alan ifadelerin anlamları şunlardır:

y	Bağımlı Değişken, Hedef Değişken
β_0	Bias Değeri, Constant (Sabit Değer), Intercept, Y-Eksenini Kestiği Nokta
x	Bağımsız Değişken(ler)
β_1	Bağımsız Değişken(ler)nin katsayı (Coefficient) değeri



Şekil 4. Lineer Regresyon Modeli

Şekil 4'te yer alan örnek, Head Size (Baş Ölçüsü) ve Brain Weight (Beyin Ağırlığı) verilerine içeren bir veri seti örneğidir. Bu örnekte, Head Size değerine göre Brain Weight değeri tahmin edilmeye çalışılmaktadır.

Yapılan tanımlamaya göre, Head Size verileri bağımsız bir değişkeni ifade ederken, Brain Weight değeri bağımlı bir değişkeni ifade etmektedir. Çünkü Brain Weight değeri, Head Size Değerine bağlıdır ve ona göre belirlenmektedir.

Lineer Regresyonda asıl amaç, her iki değişkeni en iyi şekilde ayarlayarak düz ve doğrusal bir çizgi çekmektir. Çizgi üzerinde karşılık gelinen noktaya göre tahmin yapmaktadır. β_0

değerinin ne anlama geldiğini Denklem (1)'de söylemiştik. β_0 değeri, bağımsız değişkenin eğimini ifade etmektedir.

Lineer Regresyon denklemine göre, bir veri değeri alarak uygulama yapalım: Head Size değerini 4250 cm^3 olarak alalım. Amacımız, bu değere göre Brain Weight değerini bulmaktır. Bu değeri seçtikten sonra karşılık gelen noktayı bulalım. Bu nokta, Lineer doğru üzerinde olmadığını varsayalım. β_0 değerini bilmediğimiz için 1000 olarak düşünelim. Karşılık gelen veri noktasının doğruya olan uzaklıklarını (a ve b) bulduktan sonra denklemde yerine yazarak sonuca ulaşabiliriz.

$$y = \beta_0 + \frac{a}{b} * x$$

$$1250 = 1000 + \frac{200}{750} * 4250$$

Yaptığımız işleme göre Head Size değeri 4250 cm^3 olduğunda, Brain Weight değerini 1250 gr. olarak bulduk.

Peki, bulunan bu değerler neyi ifade etmektedir? Modelimizin doğru sonuç bulup bulmadığına nasıl karar vereceğiz? Bu sorunun yanıtına bir sonraki bölümde yanıt arayacağız.

1. 1. LINEAR REGRESYON UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

2. REGRESYON MODELLERİ İÇİN PERFORMANS DEĞERLENDİRME ÖLÇÜTLERİ

Tasarladığımız bir regresyon modelinin doğru sonuç verip vermediğini ya da doğru sonuç vermeye ne kadar yaklaştığını ölçmek için birtakım performans değerlendirme metriklerine bakarız. Bu metriklere değinmeden evvel, bir Lineer Regresyon uygulamamıza bir bakalım.

Head Size ağırlığını rastgele bir değer alarak seçtik. Daha sonra, yine grafik üzerinden birtakım değerler bularak bir sonuca ulaştık ve sonuç olarak 1250 cm^3 değerini elde ettik. Bulduğumuz bu değerin sonucun doğruluğundan ya da ne kadar doğru olduğundan emin olmak için gerçek Head Size değerinden, tahmin ederek bulduğumuz Head Size değerini çıkararak bir değer elde ederiz. Bulduğumuz bu değer, artık (Residual - R) değeri olarak adlandırılmaktadır. Gerçek değer y ile ifade edilirken, tahmin ederek bulduğumuz değer y_head olarak ifade edilmektedir.

$$R = y - y\_head \qquad \text{Denklem (4)}$$

Ancak bu değeri bulmak da bize yetmeyecektir. Regresyon modellerinde, çizilen doğrunun altında kalan negatif değerleri ifade ederken, üstünde kalan alan pozitif değerler olarak ifade edilir. Şöyle düşünelim: Eğer tahmin değerlerimizin bir kısmı pozitif, bir kısmı negatif değerlerden oluşursa ve bu değerleri toplarsak sıfıra yaklaşmış oluruz. Yani, modelimizin iyi bir sonuç verdiği anlamını çıkarmış oluruz. Fakat, pozitif ve negatif değerleri birbirleriyle toplayarak elde ettiğimiz residual değerleri bizim modelimizin performansı açısından yeterli değildir. Çünkü biz pozitif ve negatif residual değerlerini toplayarak, negatif residual değerlerini kaybetmiş oluyoruz. Bu durumu yaşamamak için elde ettiğimiz tüm residual değerlerinin karesini alıp, negatif residual değerlerini kaybetmemek üzere modelimizin asıl performansını değerlendiriyoruz. Tüm residual değerlerinin karesini alıp toplayıp, veri setinde yer alan toplam örnek sayısına (n) bölersek, biz modelimizin performans değerlendirmesini yapabiliyoruz.

$$sum(R^2)/n \qquad \text{Denklem (5)}$$

Denklem (5)'te yer alan ifadenin sonucu bize **MSE (Mean Squared Error – Ortalama Kare Hatası)** sonucunu vermektedir. MSE, modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu ölçer. Her tahmin hatasının karesini alır ve bunların ortalamasını alır. Daha büyük

hataların daha fazla vurgulanmasını sağlar. MSE'nin düşük olması, daha iyi bir model performansını gösterir. Hesaplama formülü, Denklem (6)'da verilmiştir. Matematiksel olarak genel gösterimi Denklem (7)'deki gibidir.

$$MSE = sum(R^2)/n \qquad \text{Denklem (6)}$$

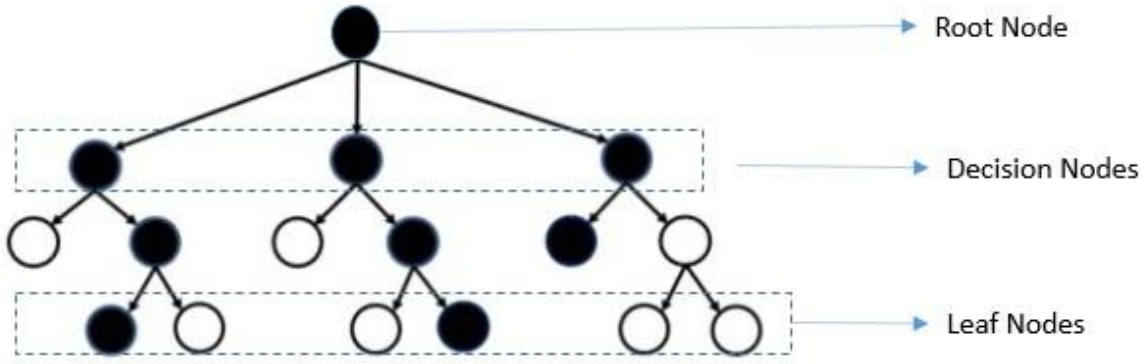
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y\_head_i)^2 \qquad \text{Denklem (7)}$$

Bir diğer performans değerlendirme metriği ise **MAPE (Mean Absolute Percentage Error - Ortalama Mutlak Yüzde Hata)** değeridir. MAPE, tahmin hatalarının yüzde cinsinden ortalama büyüklüğünü ölçer. Özellikle tahminlerin yüzdelik hata oranlarının önemli olduğu uygulamalarda kullanışlıdır. Denklem (8)'de hesaplanması gösterilmiştir.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{y_i - y\_head_i}{y_i} \qquad \text{Denklem (8)}$$

3. DECISION TREE

Decision Tree (Karar Ağacı) modelinin temeli 1986 yılında Ross Quinlann ID3 (Iterative Dichotomiser 3) algoritmasına dayanmaktadır. Yine Quinlann, 1993'te C4.5 (Classification and Regression Tree - CART) modelini çıkararak ID3 modelinin daha gelişmiş versiyonunu sunmuştur ve bugünkü Decision Tree modelinin temel yapısını oluşturmuştur. Şekil 5'te Decision Tree yapısı basit örnekte verilmiştir.



Şekil 5. Decision Tree Mimarisi

Kök Düğüm (Root Node): Karar ağacının başlangıç düğümüdür. Tüm veri kümesi bu düğümden başlar. Kök düğüm, tüm veri noktalarını içerir ve bu verileri belli özelliklere göre sınıflandırmaya başlamak için kullanılır.

Karar Düğümleri (Decision Nodes): Karar ağacı boyunca yer alan düğümlerdir. Her bir karar düğümü, bir veya daha fazla özelliği (veri setinin bir özelliğini veya özelliklerini) kullanarak veri noktalarını iki veya daha fazla alt küme veya dal (branch) arasında böler. Örnek olarak, bir karar düğümü cinsiyet özelliğine göre verileri iki kategoriye ayırabilir: "Erkek" ve "Kadın".

Yaprak Düğümleri (Leaf Nodes): Yaprak düğümleri, veri noktalarını son sınıflandırmalarına ayıran düğümlerdir. Yani, karar ağacı veri noktalarını bu düğümlerde sınıflandırır. Örneğin, bir yaprak düğümü "Erkek" veya "Kadın" olarak sınıflandırılmış veriler içerebilir.

Ağaç Yapısı: Karar ağacı, birçok karar düğümü ve yaprak düğümünün bir araya gelerek oluşturduğu bir ağaç yapısıdır. Her düğüm, bir özelliği ve bu özelliği kullanarak veri kümesini bölme kriterini temsil eder.

Decision Tree, veri kümesini analiz ederek hangi özelliklerin hedef değişkeni en iyi tahmin ettiğini veya sınıflandırdığını belirler. Bu analiz sonucunda ağaç yapısı oluşturulur ve bu yapı, yeni veri noktalarının hedef değişkenini tahmin etmek veya sınıflandırmak için kullanılabilir.

3. 1. DECISION TREE AVANTAJLARI

Decision Tree, birçok avantaja sahiptir ve bu nedenle birçok uygulama alanında kullanılırlar. Decision Tree modelinin avantajları:

Açıklanabilirlik: Karar ağaçları, veri madenciliği ve makine öğrenme modellerinin sonuçlarını anlamak için oldukça açık ve anlaşılırdır. Ağaç yapısını görselleştirmek ve anlamak kolaydır, bu nedenle modelin nasıl tahminlerde bulunduğunu anlamak için kullanıcılar ve paydaşlar için uygundur.

Kolay Yorumlanabilirlik: Karar ağaçları, hangi özelliklerin hedef değişkeni nasıl etkilediğini açıkça gösterir. Bu, veri analisti veya uzmanların verilere daha iyi bir anlayış kazanmalarına yardımcı olur.

Kategori ve Sayısal Değişkenlerle Uyum: Karar ağaçları, hem kategorik (karakteristik) hem de sayısal verilere uygun şekilde kullanılabilir. Bu, farklı türde verilerin bir arada kullanılabildiği çok çeşitli uygulama alanlarına olanak tanır.

Aşırı Uyum Kontrolü: Karar ağaçları, aşırı uyumu kontrol etmek için parametrelerle ayarlanabilir. Bu, modelin eğitim verilerine fazla uyum sağlayarak yeni veriler üzerinde kötü performans göstermesini engeller.

İşleme Maliyeti ve Hız: Karar ağaçları, büyük veri kümelerini hızlı bir şekilde işleyebilir ve model oluşturabilir. Ayrıca, eğitim ve tahmin süreçleri genellikle diğer daha karmaşık modellere göre daha hızlıdır.

İyi Başlangıç Noktası: Karar ağaçları, modelleme sürecinin başlangıcında veriye dair önemli görüşler sunabilir. Bu, daha karmaşık modellerin geliştirilmesi veya veri madenciliği süreçlerinin daha fazla analizine yol açabilir.

Sayısal ve Kategorik Sonuçlar: Karar ağaçları, regresyon (sayısal tahmin) ve sınıflandırma (kategorik tahmin) gibi farklı türde sonuçlar için kullanılabilir.

3. 2. DECISION TREE DEZAVANTAJLARI

Aşırı Uyum (Overfitting): Karar ağaçları, eğitim verilerine çok iyi uyum sağlayabilir ve bu, modelin yeni, görünmeyen verilere uygulandığında kötü performans gösterme eğilimindedir. Aşırı uyum, ağacın çok karmaşık hale gelmesi ve veriye özgü hataları modellemeye çalışması sonucu ortaya çıkar.

Duyarlılık: Küçük değişiklikler veriye veya veri kümesinin yeniden örnekleme işlemlerine karar ağacının yapısını büyük ölçüde etkileyebilir. Bu, modelin kararlarının istikrarsız olabileceği anlamına gelir.

Başarı İlkesi: Karar ağacı, verileri özelliklere göre bölerek bir hiyerarşi oluşturur. Ancak bu bölünme, genellikle en yüksek bilgi kazancını elde etmek amacıyla yapılır ve bu nedenle bazı kararlar diğerlerine göre daha fazla öncelik kazanabilir.

Karmaşıklık: Karar ağaçları, çok büyük ve karmaşık veri kümeleri üzerinde çalışırken çok karmaşık ve büyük ağaçlar üretebilirler. Bu, modelin anlaşılması, yönetilmesi ve uygulanmasını zorlaştırabilir.

Bağımlı Veriye Duyarlılık: Karar ağacı, bağımsızlık varsayımı altında çalışır ve veri noktaları arasındaki bağımlılıkları genellikle göz ardı eder. Bu, zaman serileri veya mekansal veriler gibi bağımlı verilere uygulandığında modelin performansını etkileyebilir.

Kategorik Verilerle Uyum Sorunu: Bazı karar ağacı uygulamaları, kategorik verileri işlemekte zorlanabilir ve sayısal verilere göre daha düşük performans gösterebilir.

3. 3. DECISION TREE UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

4. RANDOM FOREST

2001 yılında Leo Breimann tarafından geliştirilmiştir. Random Forest (Rastgele Orman), makine öğrenme alanında yaygın olarak kullanılan bir ensemble (bir araya getirme) öğrenme yöntemidir. Bu yöntem, birden çok karar ağacını bir araya getirerek daha güçlü ve kararlı bir tahmin modeli oluşturur. Random Forest, sınıflandırma ve regresyon problemleri için kullanılabilir ve aşırı uyum (overfitting) eğilimini azaltır.

Random Forest'ın ana özellikleri şunlardır:

Karar Ağaçları: Random Forest, temel olarak karar ağaçlarından oluşur. Her bir karar ağacı, farklı alt veri örneklerine dayalı olarak bir hedef değişkeni tahminler.

Rastgele Veri Seçimi: Random Forest, her bir karar ağacının oluşturulması sırasında rastgele veri seçimini kullanır. Bu, her ağacın farklı alt kümelere sahip olmasını sağlar. Rastgele veri seçimi, aşırı uyumu önlemeye yardımcı olur ve modelin çeşitliliğini artırır.

Rastgele Özellik Seçimi: Aynı şekilde, her bir karar ağacı oluşturulurken rastgele özellik seçimi yapılır. Bu, ağaçlar arasındaki bağımlılığı azaltır ve modelin daha dengeli ve güçlü olmasını sağlar.

Oy Birliği (Voting): Random Forest, her bir karar ağacının tahminlerini bir araya getirir ve genellikle sınıflandırma durumunda çoğunluk oyu (majority vote) ilkesini uygular. Yani, her ağacın tahminleri toplanır ve en fazla oy alan sınıf tahmini sonucu olarak kabul edilir.

4. 1. RANDOM FOREST AVANTAJLARI

Yüksek Tahmin Doğruluğu: Random Forest, birden çok karar ağacını bir araya getirerek genellikle yüksek tahmin doğruluğu sağlar. Farklı ağaçlar farklı veri alt kümesi ve özelliklerle çalıştığı için bu çeşitlilik modelin daha güçlü olmasını sağlar.

Aşırı Uyumun Azaltılması: Random Forest, aşırı uyumu önlemeye yardımcı olur. Rastgele veri ve özellik seçimi, her ağacın farklı bir şekilde eğitilmesini sağlar ve bu, modelin genelleme yeteneğini artırır.

Kategorik ve Sayısal Değişkenlerle Uyum: Random Forest, hem kategorik (karakteristik) hem de sayısal verileri işlemek için uygundur. Bu, farklı türde verilerin bir arada kullanılabilmesine olanak tanır.

Verilere Duyarlılık: Random Forest, dengesiz veri kümeleriyle iyi çalışabilir ve az veri ile iyi sonuçlar elde edebilir.

Özellik Önem Sıralaması: Random Forest, her bir özelliğin tahminler üzerindeki etkisini sıralayan bir özellik önem sıralaması sunar. Bu, veri analisti veya uzmanların önemli özellikleri tanımlamasına ve modeli iyileştirmesine yardımcı olabilir.

Direnç: Outlier (aykırı) değerlere ve gürültülü verilere karşı dayanıklıdır ve bu nedenle dengesiz veya karmaşık veri kümeleri üzerinde iyi performans gösterebilir.

Paralelleştirme: Random Forest, çok sayıda ağaç oluşturmak ve tahmin yapmak için paralelleştirilebilir, bu da işlem hızını artırabilir.

4. 2. RANDOM FOREST DEZAVANTAJLARI

Model Karmaşıklığı: Random Forest, birçok karar ağacını bir araya getirdiği için model karmaşıklığı artabilir. Bu, modeli daha zor anlaşılır hale getirebilir ve eğitim süresini uzatabilir.

Bellek Kullanımı: Birden çok ağacın oluşturulması ve saklanması, bellek kullanımını artırabilir. Özellikle büyük veri kümeleri üzerinde çalışırken bu bir sorun olabilir.

Hız: Random Forest, tek bir karar ağacına göre daha fazla hesaplama gerektirir, bu nedenle eğitim ve tahmin süreleri daha uzun olabilir. Bununla birlikte, bu sorun, modelin paralelleştirilmesi veya donanım hızlandırıcılarının kullanılması ile giderilebilir.

İnterpretasyon Zorluğu: Karar ağaçlarının ve Random Forest'ın tahminleri ve sonuçları genellikle açıklanabilir olsa da, çok sayıda ağacın bir araya gelmesi, sonuçları yorumlamayı ve açıklamayı daha karmaşık hale getirebilir.

Özellik Seçimi: Random Forest, özellik önem sıralaması sağlasa da, bu sıralama bazen fazla özellik içeren veri kümeleri üzerinde çalışırken yetersiz olabilir. Ayrıca, bazı özelliklerin birçok ağaçta seçilmemesi durumunda önemsiz gibi görünmesi de bir dezavantaj olabilir.

Eğitim ve Ayar: Random Forest modelleri, karar ağacı sayısı ve diğer hiperparametrelerin ayarlanmasını gerektirebilir. Bu, modeli optimize etmek için ek çaba gerektirebilir.

Genelizasyon Sorunları: Random Forest, bazı problemlerde hala yanıltıcı sonuçlar verebilir veya doğrusal olmayan ilişkileri iyi yakalayamayabilir. Bu nedenle, bazı problemler için daha iyi performans gösterebilecek başka modeller de düşünülmelidir.

4. 3. RANDOM FOREST UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

5. SUPPORT VECTOR MACHINES

1995 yılında Vladimir Vapnik tarafından geliştirilmiştir. Support Vector Machines (SVM - Destek Vektör Makineleri), makine öğrenme alanında sınıflandırma ve regresyon problemlerini çözmek için kullanılan güçlü bir öğrenme algoritmasıdır. SVM, özellikle sınıflandırma problemlerinde başarılı bir şekilde kullanılır ve verileri iki veya daha fazla sınıfa ayırmak için tasarlanmıştır.

SVM'nin temel fikri, verileri daha iyi sınıflandırmak için bir hiper düzlem (hyperplane) seçmektir. Bu hiper düzlem, verilerin sınıflarını en iyi şekilde ayıran bir çizgi veya düzlem olarak düşünülebilir. SVM'nin ana hedefi, bu hiper düzlemi seçerken margin adı verilen bir boşluğu maksimize etmektir. Margin, hiper düzleme en yakın veri noktaları arasındaki mesafeyi ifade eder.

SVM, aşağıdaki ana prensiplere dayanır:

Destek Vektörler: SVM, sınıf sınırlarını veya hiper düzlemi belirleyen destek vektörleri kullanır. Destek vektörler, hiper düzleme en yakın olan ve margini belirleyen veri noktalarıdır.

Margin: SVM, iki sınıf arasındaki margini maksimize etmeye çalışır. Margin, hiper düzleme en yakın pozitif ve negatif sınıfa ait veri noktaları arasındaki mesafeyi ifade eder. Marginin maksimize edilmesi, modelin genelleme yeteneğini artırır ve aşırı uyumu azaltır.

Çekirdek Fonksiyonları: SVM, verileri yüksek boyutlu uzaylara dönüştüren ve daha karmaşık sınırların tanımlanmasına yardımcı olan çekirdek fonksiyonları kullanabilir. Özellikle veriler

doğrusal olarak ayrılamazsa, çekirdek fonksiyonları kullanarak SVM, verileri daha yüksek boyutlu uzaylarda ayrıştırabilir.

5. 1. SUPPORT VECTOR MACHINES AVANTAJLARI

Yüksek Sınıflandırma Başarısı: SVM, sınıflandırma problemlerinde yüksek başarı elde etme eğilimindedir. Özellikle margin maksimizasyonunu odak noktası olarak kullanarak, verileri çok iyi ayırmak için etkili bir yöntem sunar.

Aşırı Uyumun Azaltılması: SVM, margin maksimizasyonu ile aşırı uyumu azaltmaya yardımcı olur. Bu, modelin yeni verilere daha iyi genelleme yapmasını sağlar.

Kategorik ve Sayısal Verilere Uyum: SVM, hem kategorik (karakteristik) hem de sayısal verilerle çalışabilen bir esnekliğe sahiptir. Bu, farklı veri türlerinin aynı modelde birleştirilmesine olanak tanır.

İyi Genelleme Yeteneği: SVM, sınıf sınırlarını belirlemek için destek vektörleri kullanır ve bu, modelin iyi bir genelleme yeteneğine sahip olmasını sağlar.

Çekirdek Fonksiyonları: SVM, çekirdek fonksiyonlarını kullanarak verileri yüksek boyutlu uzaylara dönüştürebilir. Bu, karmaşık veri dağılımları ve doğrusal olarak ayrılamayan veriler üzerinde çalışmayı kolaylaştırır.

Destek Vektörler: SVM, modelin karmaşıklığını kontrol etmek ve veriye uygun bir model oluşturmak için destek vektörleri kullanır. Bu, modelin optimize edilmesini sağlar.

Genelleme Kabiliyeti: SVM, çok sayıda veri noktası ve çok sayıda özellik içeren veri kümeleri üzerinde bile iyi bir performans gösterebilir.

5. 2. SUPPORT VECTOR MACHINES DEZAVANTAJLARI

Hesaplama Maliyeti: SVM'nin hesaplama maliyeti, büyük veri kümeleri veya çok sayıda özellik içeren veri kümeleri üzerinde yüksek olabilir. SVM, hesaplama yoğun bir algoritmadır ve eğitim süresi uzun olabilir.

Modelin Hassas Hiperparametre Ayarı: SVM'nin performansı, hiperparametrelerin (örneğin, C ve kernel tipi gibi) hassas bir şekilde ayarlanmasına bağlıdır. Bu, kullanıcıların modelin en iyi performansını elde etmek için deneme yanılma ile hiperparametreleri ayarlamak zorunda kalmasına neden olabilir.

Büyük Veri Kümeleri Üzerinde Zorluklar: Büyük veri kümeleri üzerinde SVM'nin eğitilmesi ve tahmin yapılması, hesaplama maliyeti ve bellek gereksinimleri nedeniyle zor olabilir.

Tek Sınıf Sınıflandırması: SVM, genellikle iki sınıf (ikili sınıflandırma) problemleri için kullanılır. Çoklu sınıf sınıflandırma problemleri için SVM'nin uygulanması biraz karmaşık olabilir.

İnterpretasyon Zorluğu: SVM'nin sonuçları ve tahminleri genellikle açıklanması ve yorumlanması zor olabilir. Hiper düzlem ve çekirdek fonksiyonları kavramsal olarak anlaşılması gereken konseptlerdir.

Hassas Etiketleme: SVM, doğru etiketlemeye ihtiyaç duyar. Yanlış etiketlenmiş veya eksik veriler, modelin performansını olumsuz etkileyebilir.

Doğrusal Olmayan İlişkiler: SVM, doğrusal olarak ayrılamayan verileri işlemek için çekirdek fonksiyonlarını kullanabilir, ancak bu çekirdeklerin seçimi ve ayarlanması bazen karmaşık olabilir.

5. 3. SUPPORT VECTOR MACHINES ÇEKİRDEK FONKSİYONLARI

Destek Vektör Makineleri (SVM), verileri sınıflandırma veya regresyon yapmak için kullanırken çekirdek fonksiyonları (kernel functions) adı verilen matematiksel işlevleri kullanabilir. Çekirdek fonksiyonları, verileri daha yüksek boyutlu uzaylara dönüştürerek karmaşık sınırları tanımlamak ve verileri doğrusal olarak ayrılamayan hallerde sınıflandırmak için kullanılır. İşte bazı yaygın SVM çekirdek fonksiyonları:

Lineer Çekirdek (Linear Kernel): Lineer çekirdek, verileri yüksek boyutlu uzaylara dönüştürmeden doğrusal bir sınıf ayrımı oluşturur. Bu, verilerin doğrusal olarak ayrılabilir olduğu durumlar için uygundur.

Polinom Çekirdek (Polynomial Kernel): Polinom çekirdek, verileri yüksek boyutlu uzaylara dönüştürerek çoklu derecelerde polinomlarla sınıf ayrımı oluşturur. Derece parametresi, polinomun kaçıncı dereceden olduğunu belirler.

Radyal Temel Fonksiyon (Radial Basis Function - RBF): Radyal Temel Fonksiyon (RBF) çekirdeği, Gauss tipi bir fonksiyon kullanarak verileri yüksek boyutlu uzaylara dönüştürür. RBF, karmaşık ve doğrusal olarak ayrılamayan veriler için genellikle etkilidir. İki önemli parametresi vardır: gamma (γ) ve C.

Sigmoid Çekirdek (Sigmoid Kernel): Sigmoid çekirdek, verileri yüksek boyutlu uzaylara dönüştürmek için sigmoid (lojistik) bir fonksiyon kullanır. Bu, bazen belirli problemlerde kullanılan bir çekirdek türüdür.

Bu çekirdek fonksiyonları, verilerin farklı yapılarına ve sınıfların farklı dağılımlarına uyacak şekilde seçilebilir. SVM'nin başarısı, doğru çekirdek fonksiyonunun seçilmesi ve hiperparametrelerin iyi ayarlanmasına bağlıdır. Problem bağlamınıza ve veriye bağlı olarak en uygun çekirdek fonksiyonunu seçmek önemlidir.

5. 4. SUPPORT VECTOR MACHINES UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

6. NAIVE BAYES

Naive Bayes, makine öğrenme ve istatistik alanında kullanılan bir sınıflandırma ve olasılık temelli bir öğrenme algoritmasıdır. Bu algoritma, özellikle metin sınıflandırma, spam filtreleme, duygu analizi ve doküman sınıflandırma gibi uygulamalarda yaygın olarak kullanılır. Temel olarak Bayes Teoremi'ne dayanır ve bu nedenle "Bayes" adını taşır.

Naive Bayes, "naive" (saf) olarak adlandırılır, çünkü verilerdeki özelliklerin birbirinden bağımsız olduğunu varsayar. Bu bağımsızlık varsayımı, hesaplama ve model oluşturma açısından basitleştirmeye neden olur.

Naive Bayes sınıflandırma algoritmasının ana fikirleri şunlardır:

Bayes Teoremi: Bayes Teoremi, bir olayın olasılığını hesaplamak için kullanılan temel bir olasılık teoremidir. Sınıflandırma bağlamında, Bayes Teoremi, belirli bir sınıfa ait olasılığı hesaplamak için kullanılır.

Özelliklerin Bağımsızlığı: Naive Bayes, özelliklerin birbirinden bağımsız olduğunu varsayar. Yani, bir örneğin belirli bir sınıfa ait olma olasılığı, bu örneğin tüm özelliklerinin bu sınıfa ait olma olasılıklarının çarpımına eşittir.

Olasılık Dağılımları: Naive Bayes, her sınıf için özelliklerin olasılık dağılımlarını tahmin etmek için kullanılır. Bu dağılımlar, veri eğitim sırasında hesaplanır.

Naive Bayes, veri kümesindeki özellikler ve sınıflar arasındaki ilişkiyi kullanarak verileri sınıflandırır. Algoritma, her sınıfın özelliklerinin olasılık dağılımını öğrenir ve ardından yeni bir örnek geldiğinde, bu dağılımları kullanarak bu örneğin hangi sınıfa ait olduğunu tahmin eder. Tahminler, Bayes Teoremi kullanılarak hesaplanır ve en yüksek olasılığa sahip sınıf seçilir.

6. 1. NAIVE BAYES AVANTAJLARI

Basitlik ve Hız: Naive Bayes, hesaplama açısından basit bir algoritmadır ve eğitim ve tahmin süreleri genellikle hızlıdır. Bu, büyük veri kümeleri veya gerçek zamanlı uygulamalar için uygundur.

İyi Çalışma Performansı: Naive Bayes, metin sınıflandırma, spam filtreleme ve duygu analizi gibi pek çok uygulama alanında iyi çalışabilir. Özellikle metin verileri üzerinde etkilidir.

İyi Başlangıç Noktası: Naive Bayes, diğer sınıflandırma algoritmalarına kıyasla daha basit ve daha hızlıdır. Bu nedenle, yeni bir sınıflandırma problemini çözmeye başlarken bir başlangıç noktası olarak kullanılabilir.

Çok Sınıf Sınıflandırma: Naive Bayes, çok sınıflı sınıflandırma problemlerinde başarılı bir şekilde kullanılabilir.

Metin Verileri ile Uyum: Naive Bayes, metin madenciliği ve doğal dil işleme uygulamaları için çok uygundur. Özellikle kelime sayıları veya kelime frekansları gibi özelliklerin kullanıldığı metin sınıflandırma problemlerinde etkilidir.

Hiperparametre Ayarı Az: Naive Bayes, diğer bazı algoritmalara göre daha az hiperparametreye ihtiyaç duyar. Bu, hiperparametre ayarının daha basit olmasını sağlar.

İyi Başarı: Naive Bayes, veri setlerine ve problemlere bağlı olarak yüksek doğruluk elde edebilir.

Ancak Naive Bayes'ın bağımsızlık varsayımı bazı veri kümesi türlerinde geçerli olmayabilir ve bu nedenle yanıltıcı olabilir. Ayrıca, metin sınıflandırma gibi çok büyük özellik uzayları içeren veri kümeleri üzerinde bazı dezavantajlar yaşanabilir. Veri kümesine ve problem bağlamına bağlı olarak, Naive Bayes'in avantajları ve dezavantajları dikkate alınmalıdır.

6. 2. NAIVE BAYES DEZAVANTAJLARI

Bağımsızlık Varsayımı: Naive Bayes, her özelliğin birbirinden bağımsız olduğunu varsayar. Bu varsayım gerçek dünyadaki veriler için geçerli olmayabilir. Özellikler arasındaki bağımlılık, modelin yanıltıcı sonuçlar vermesine neden olabilir.

Eş Anlamlı Terimler: Metin sınıflandırma gibi uygulamalarda, eş anlamlı terimlerin kullanılması Naive Bayes'in performansını olumsuz etkileyebilir. Model, farklı terimleri farklı olarak ele alır ve bu, metindeki anlamı karmaşıklaştırabilir.

Hesaplama İşlemi: Özellikle çok büyük özellik uzaylarına veya büyük veri kümelerine sahip verilerle çalışırken, hesaplama maliyeti artabilir. Modelin eğitimi ve tahminleri hızlı olsa da, büyük verilerle çalışırken bellek ve hesaplama kaynakları gerekebilir.

Düşük Hassasiyet: Naive Bayes, bazı diğer sınıflandırma algoritmalarına kıyasla düşük hassasiyete sahip olabilir. Özellikle sınıfların dengesiz olduğu durumlarda yanıltıcı sonuçlar verebilir.

Değişken Olasılık Dağılımları: Naive Bayes, özelliklerin dağılımlarını varsayılan olarak kullanır ve bu dağılımlar gerçek veriye uymayabilir. Bu, özellikle verilerin heterojen olduğu durumlarda bir sorun olabilir.

Veri Eksikliği: Eğitim verilerinde eksik veya nadir sınıfların bulunduğu durumlarda, modelin performansı düşebilir. Bu tür verilerle çalışırken yeniden örneklemleme veya dengesiz veri işleme teknikleri gerekebilir.

Doğru Olmayan Olasılık Tahminleri: Naive Bayes, olasılıkları tahmin ederken doğru sonuçlar verme eğilimindedir, ancak bu olasılıklar kesin olmayabilir.

6. 3. NAIVE BAYES UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

7. K-NEAREST NEIGHBOR

K-En Yakın Komşu (K-Nearest Neighbors veya kısaca KNN), denetimli öğrenme içinde sınıflandırma ve regresyon problemlerini çözmek için kullanılan bir makine öğrenme algoritmasıdır. Temel fikir, yeni bir örneği etrafındaki K en yakın komşusuna (veri noktalarına)

dayanarak sınıflandırmak veya tahmin etmektir. KNN, basit ve anlaşılır bir algoritma olup, temel bir konsepti kullanır: benzer örnekler bir araya gelir.

KNN algoritması aşağıdaki şekilde çalışır:

Eğitim Verilerinin Alınması: KNN algoritması, sınıflandırmak veya tahmin etmek için kullanılacak eğitim verileri gerektirir. Bu veriler, girdi özelliklerini (feature) ve bu özelliklerle ilişkilendirilmiş sınıf etiketlerini içerir.

Benzerlik Ölçüsünün Seçilmesi: KNN'de, komşuluk ilişkilerini belirlemek için iki veri noktası arasındaki benzerliği ölçmek için bir benzerlik ölçüsü seçilir. Öklidyen mesafe (Euclidean distance) ve Manhattan mesafe (Manhattan distance) gibi yaygın olarak kullanılan metrikler mevcuttur.

Yeni Örneğin Tahmin Edilmesi: KNN, yeni bir örneği sınıflandırmak veya tahmin etmek için kullanılır. Bu örnek, eğitim verileri ile benzerlik ölçüsüne göre en yakın K veri noktasını bulur. Bu komşuların sınıfları (sınıflandırma için) veya değerleri (regresyon için) kullanılarak yeni örnek tahmin edilir. Örneğin, sınıflandırma durumunda, en yakın K komşunun çoğunluğunun sınıfı tahmin olarak kullanılabilir.

K Değerinin Seçilmesi: KNN algoritmasında K değeri, komşuluk sayısını belirler. K değeri, algoritmanın performansını etkileyen önemli bir hiperparametredir. Küçük K değerleri daha fazla gürültüye ve büyük K değerleri daha fazla düzgünleştirmeye yol açabilir. K'nin seçimi, veri kümesi ve probleme bağlı olarak ayarlanmalıdır.

7. 1. K-NEAREST NEIGHBOR AVANTAJLARI

Basitlik: KNN, basit bir algoritmadır ve kolayca anlaşılabilir. Bu nedenle, makine öğrenmeye yeni başlayanlar için iyi bir başlangıç noktası olabilir.

Eğitim Süresi Yok: KNN'nin eğitim süresi yoktur, çünkü veri noktalarını sadece saklar ve yeni bir örnek tahmin edildiğinde gerçekleştirilen hesaplamaları kullanır.

Hem Sınıflandırma hem Regresyon İçin Kullanılabilir: KNN, sınıflandırma problemlerinin yanı sıra regresyon problemlerini çözmek için de kullanılabilir. K sınıf etiketi tahmin etmek yerine, KNN regresyon problemi için K en yakın komşunun değerlerini kullanabilir.

Karmaşık Model Ayarına Gerek Yok: KNN'nin hiperparametreleri nispeten azdır ve K değeri dışında fazla karmaşık bir yapıya ihtiyaç duymaz. K değeri, probleme ve veriye bağlı olarak ayarlanabilir.

Veri Değişikliği ile Başa Çıkabilir: KNN, veri noktaları veya sınıflar değiştiğinde, modeli hızlı bir şekilde güncelleyebilir. Bu, gerçek zamanlı uygulamalarda kullanışlı olabilir.

Genel Performans: KNN, veriye bağlı olarak oldukça iyi performans gösterebilir, özellikle veriler düşük boyutlu ve komşuluk yapısı belirginse.

Anomalileri Belirleme: KNN, anormallik tespiti (outlier detection) uygulamalarında kullanılarak veri kümelerindeki anormal veri noktalarını tanımlayabilir.

7. 2. K-NEAREST NEIGHBOR DEZAVANTAJLARI

Hesaplama Maliyeti: KNN, tahmin yaparken tüm veri noktalarını incelemesi gerektiği için büyük veri kümeleri üzerinde hesaplama maliyeti yüksek olabilir. Özellikle çok yüksek boyutlu veri kümeleri için uygun olmayabilir.

Veri Ölçeği Sorunları: KNN, özellikler arasındaki mesafeyi hesaplarken verilerin ölçeğine hassas olabilir. Özelliklerin ölçeklenmemesi, büyük değerlere sahip bir özelliğin, tahminler üzerinde baskın bir etkiye sahip olmasına neden olabilir.

K Değeri Seçimi: KNN'de K değeri (komşu sayısı), modelin performansını etkileyen kritik bir hiperparametredir. Doğru K değerini seçmek zor olabilir ve yanlış bir seçim modelin performansını olumsuz etkileyebilir.

Yavaş Tahmin Hızı: KNN, her tahmin yaparken tüm veri noktalarını değerlendirmesi gerektiği için tahmin süresi yavaş olabilir. Özellikle gerçek zamanlı uygulamalar için uygun olmayabilir.

Eşit Sayıda Sınıflar: Eğer veri kümesindeki sınıflar dengesizse, yani bir sınıf diğerlerine göre çok daha fazlaysa, KNN modeli bu dengesizliği yansıtabilir ve çok büyük sınıfların diğerlerini etkileyebilir.

Hassaslık Ayarı: KNN'nin hassaslık derecesi, benzerlik ölçüsüne ve K değerine bağlıdır. Bu ayarlar yapılmazsa modelin performansı düşebilir.

Boyutluluk Sorunları: Yüksek boyutlu veri kümeleri üzerinde KNN kullanırken boyutluluk sorunları ortaya çıkabilir. Bu, verilerin uzayda seyrek olmasına ve mesafelerin anlamsız hale gelmesine neden olabilir.

Küçük Veri Kümeleri: KNN, küçük veri kümeleri üzerinde iyi çalışsa da, çok az veri içeren veri kümeleri için uygun olmayabilir.

7. 3. K-NEAREST NEIGHBOR UZAKLIK ÖLÇME TEKNİKLERİ

K-En Yakın Komşu (KNN) algoritması, veri noktaları arasındaki benzerliği hesaplamak ve komşuluk ilişkilerini belirlemek için bir uzaklık ölçüsü kullanır. KNN'de yaygın olarak kullanılan uzaklık ölçüm teknikleri şunlar:

Öklidyen Mesafe (Euclidean Distance): Öklidyen mesafe, iki veri noktası arasındaki en yaygın kullanılan uzaklık ölçüsüdür.

Manhattan Mesafe (Manhattan Distance):Manhattan mesafe, iki veri noktası arasındaki mesafeyi, her bir boyutun mutlak farklarının toplamı olarak hesaplar. Bu mesafe adını New York City'deki Sokak ağlarından alır, çünkü iki nokta arasındaki yol, yatay ve dikey hareketlerle belirtilir.

Minkowski Mesafe: Minkowski mesafe, Öklidyen ve Manhattan mesafelerini genelleştiren bir uzaklık ölçüsüdür.

Chebyshev Mesafe: Chebyshev mesafesi, iki veri noktası arasındaki mesafeyi, her boyutta en büyük mutlak farkı kullanarak hesaplar. Formül şu şekildedir: Bu mesafe ölçümü, özellikler arasındaki maksimum farkı temsil eder.

Kosinüs Benzerliği (Cosine Similarity): KNN'de sınıflandırma veya kümeleme yaparken, özellik vektörleri arasındaki açıyı ölçmek için kosinüs benzerliği kullanılabilir. Kosinüs benzerliği, iki özellik vektörünün birbirine ne kadar benzediğini ölçer. Daha büyük bir kosinüs benzerliği, daha benzer vektörler anlamına gelir.

Pearson Korelasyon Katsayısı: Pearson korelasyon katsayısı, iki değişken arasındaki lineer ilişkiyi ölçer. KNN'de benzerlik ölçümü olarak kullanılabilir. Pearson korelasyon katsayısı, iki veri noktası arasındaki korelasyonu hesaplar.

7. 4. K-NEAREST NEIGHBOR UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

8. SINIFLANDIRMA MODELLERİ İÇİN PERFORMANS DEĞERLENDİRME ÖLÇÜTLERİ

Sınıflandırma modellerinin performansını değerlendirmek için bir dizi farklı kriter ve metrik kullanılabilir. Bu metrikler, modelin sınıflandırma yeteneği karmaşıklık matrisi kullanarak doğruluk, hassasiyet, duyarlılık, özgüllük, F1 puanı gibi farklı yönlerini ölçmeyi amaçlar.

8. 1. CONFUSION MATRIX

Karmaşıklık matrisi (Confusion Matrix), sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir araçtır. Bu matris, modelin sınıflandırma doğruluğunu ve hatalarını detaylı bir şekilde gösterir. Karmaşıklık matrisi, modelin tahminlerini gerçek sınıf etiketleriyle karşılaştırır.

Örneklendirecek olursak; aşağıdaki tabloda Hasta ve Sağlıklı hastalara ait bir problem üzerinde gerçek verilere karşı tahmin edilen sınıf çıktısını içersin. Sınıf karşılığı olarak Hasta olanları 1, Sağlıklı olanları 0 şeklinde belirleyecek olursak Tablo 1'deki sınıflandırma sonucu aldığımızı varsayalım:

1	Hasta
0	Sağlıklı

Tablo 1: Sınıflandırma Sonuç Tablosu

Gerçek Veri	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
Tahmin Veri	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**

Hastayı (1), Hasta (1) olarak tahminlemek bize True Positive (TP) oranını verir. Yani modelim doğru tahmin yapmıştır.

Sağlıklı olanı (0), Hasta (1) olarak tahminlemek bize False Positive (FP) oranını verir. Yani, modelim yanlış tahmin yapmıştır.

Hastayı (1), Sağlıklı (0) olarak tahminlemek bize False Negative (FN) oranını verir. Yani, modelim yanlış tahmin yapmıştır.

Sağlıklı olanı (0), Sağlıklı (0) olarak tahminlemek bize True Negative (TP) oranını verir. Yani modelim doğru tahmin yapmıştır.

Yapılan bu tahminlerden oluşan karmaşıklık matrisi aşağıdaki gibi olmuştur.

	Tahmin Edilen Veriler	
Gerçek Veriler	4 *True Positive*	3 *False Positive*
	1 *False Negatif*	2 *True Negative*

8. 2. ACCURACY

Doğruluk (Accuracy): Doğru sınıflandırılan örneklerin toplam örneklere oranını ifade eder. Bu, genel model başarısını ölçer. Ancak dengesiz sınıf dağılımlarına sahip veri kümeleri için yetersiz olabilir.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

8. 3. PRECISION

Hassasiyet (Precision): Hassasiyet, pozitif olarak tahmin edilen örneklerin gerçekten pozitif olan örneklerin oranını ifade eder. Yanlış pozitifleri minimize etmek istediğiniz durumlarda kullanışlıdır.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

8. 4. RECALL

Duyarlılık (Recall veya Sensitivity): Duyarlılık, gerçek pozitif olan örneklerin ne kadarının doğru bir şekilde pozitif olarak tahmin edildiğini ifade eder. Yanlış negatifleri minimize etmek istediğiniz durumlarda önemlidir.

$$\text{Recall} = \frac{TP}{TP + FN}$$

8. 5. F1 SCORE

F1 puanı, hassasiyet ve duyarlılığın harmonik ortalamasıdır. Dengesiz sınıf dağılımlarına sahip veri kümelerinde kullanışlıdır, çünkü hem yanlış pozitifleri hem de yanlış negatifleri hesaba katar.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

9. LABEL ENCODING

Label Encoding, kategorik (nominal veya ordinal) verileri sayısal değerlere dönüştürmek için kullanılan bir veri dönüştürme tekniğidir. Makine öğrenme algoritmaları genellikle sayısal verilerle çalışır, bu nedenle kategorik verilerin sayısal formata dönüştürülmesi gerekir. Label Encoding, bu dönüşüm işlemini basit bir şekilde gerçekleştirir.

Label Encoding işlemi şu adımları içerir:

1. Her farklı kategoriye (etikete) bir benzersiz sayısal değer atanır. Bu sayılar, genellikle tamsayılar olarak temsil edilir.
2. Veri kümesindeki kategorik değerler, bu sayısal eşlemelere göre değiştirilir.
3. Elde edilen sayısal değerler, makine öğrenme modeline beslemek için kullanılır.

Örnek olarak, bir veri kümesi içinde "Kırmızı," "Mavi," ve "Yeşil" gibi kategorik bir özellik bulunduğunu düşünelim. Label Encoding ile bu kategorik değerler aşağıdaki gibi sayısal değerlere dönüştürülebilir:

Kırmızı: 0

Mavi: 1

Yeşil: 2

BÖLÜM - IV

1. LOJİSTİK REGRESYON

Lojistik Regresyon (Logistic Regression), genellikle sınıflandırma problemleri için kullanılan bir istatistiksel modeldir. Bu model, bağımlı değişkenin (outcome veya hedef değişken) kategorik olduğu durumlarda kullanılır. İkili sınıflandırma (binary classification) örneklerinde sıklıkla kullanılır; yani, çıktı yalnızca iki sınıftan biri olabilir. Ancak, çok sınıflı sınıflandırma problemlerine de genişletilebilir.

Logistic Regression, adını matematikteki "*logit*" fonksiyonundan alır. Logit fonksiyonu, olasılıkları (-∞, ∞) aralığından (0, 1) aralığına dönüştürür. Bu dönüşüm sayesinde, olasılıkların log-odds'u lineer bir model ile ifade edilebilir. Yani, Logistic Regression, bağımsız değişkenlerin bir lineer kombinasyonunu kullanarak bir log-odds değeri oluşturur ve bu değeri bir logit fonksiyonuna geçirerek sonuç olarak 0 ile 1 arasında bir olasılık değeri elde eder.

Bu model, lojistik fonksiyonun içindeki lineer kombinasyonu alarak bir olasılık değeri üretir. Eğer bu olasılık belirli bir eşik değerinden büyükse, model çıktıyı 1 olarak tahmin eder, aksi takdirde 0 olarak tahmin eder.

Logistic Regression'ın eğitimi genellikle maksimum olabilirlik tahmini kullanılarak yapılır. Model, veri setindeki gözlemlerin sınıflarını en iyi şekilde tahmin etmek için katsayıları ayarlamaya çalışır.

1. 1. LOJİSTİK REGRESYON AVANTAJLARI

Yorumlanabilirlik: Logistic Regression modeli, modelin iç işleyişi hakkında geniş bir anlayış sağlar. Katsayılar, her bir bağımsız değişkenin bağımlı değişken üzerindeki etkisini doğrudan gösterir. Bu özellik, modelin sonuçlarını yorumlamayı ve anlamayı kolaylaştırır.

Hızlı Eğitim ve Tahmin: Logistic Regression, eğitim süreleri genellikle düşüktür, çünkü sadece lineer bir modeli öğrenir. Bu özellik, büyük veri setleri üzerinde hızlı eğitim ve tahmin yapma avantajı sağlar.

Düşük Hiperparametre Sayısı: Modelin ayarlanması için çok az hiperparametre vardır, bu da modeli kullanmayı ve uygulamayı kolaylaştırır. Bu özellik, özellikle basit sınıflandırma problemlerinde avantaj sağlar.

Çok Sınıflı Sınıflandırma: Logistic Regression, ikili sınıflandırmadan çok sınıflı sınıflandırmaya genişletilebilir. Bu, çoklu sınıf problemleri için kullanışlı bir özelliktir.

Bağımsız Değişkenler Arasındaki İlişkiyi İfade Etme: Logistic Regression, bağımsız değişkenler arasındaki ilişkiyi modelleme yeteneği ile bir değişkenin diğerleriyle etkileşimini değerlendirebilir.

1. 2. LOJİSTİK REGRESYON DEZAVANTAJLARI

Doğrusallık Sınırlaması: Logistic Regression, doğrusal bir karar sınırına dayanır. Bu nedenle, veri setindeki karmaşık, doğrusal olmayan ilişkileri ifade etmekte sınırlıdır. Eğer veri seti bu tür ilişkiler içeriyorsa, Logistic Regression modeli yetersiz kalabilir.

Aşırı Uyum (Overfitting) Sorunu: Logistic Regression, veri setindeki gürültüye ve aşırı öğrenmeye karşı hassas olabilir. Aşırı uyarlanma, modelin eğitim verilerine aşırı uyum sağlaması ve genelleme yeteneğini kaybetmesi durumunu ifade eder.

Bağımlı Değişkenin Doğrusallığı: Logistic Regression'ın doğrusallık varsayımı, bağımlı değişkenin doğrusal bir kombinasyon halinde olması gerektiği anlamına gelir. Eğer bu varsayım sağlanmazsa, modelin doğruluğu düşebilir.

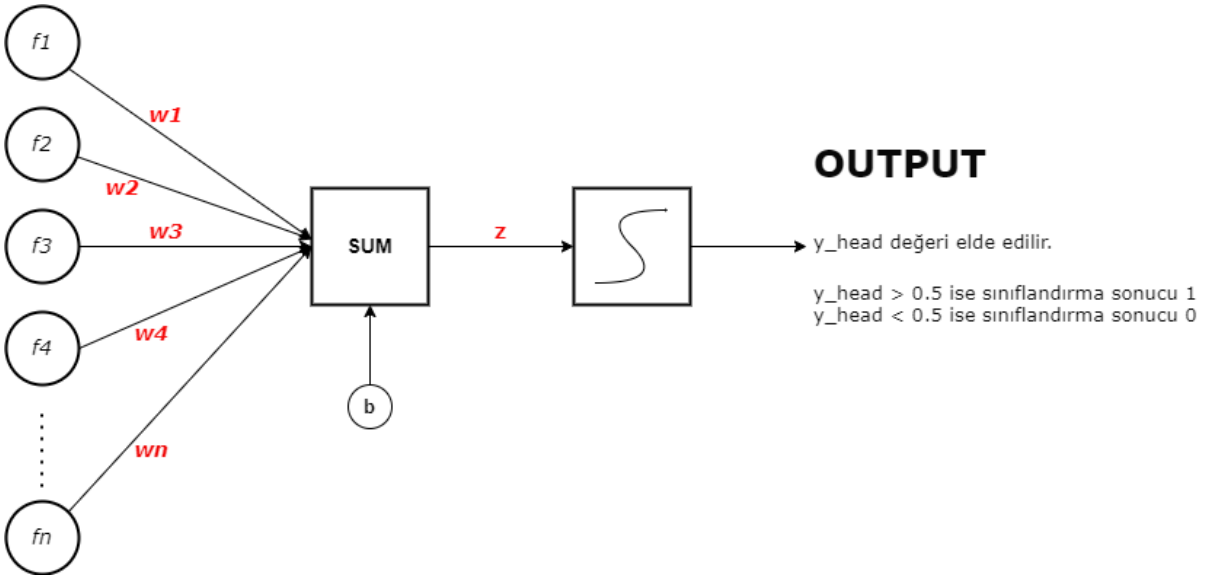
Aykırı Değerlere Duyarlılık: Logistic Regression, aykırı değerlere karşı dirençli olabilir, ancak çok sayıda aykırı değer içeren veri setlerinde performansı olumsuz etkilenebilir.

Çok Sayıda Bağımsız Değişken Sorunu: Eğer modelde çok sayıda bağımsız değişken varsa, aşırı öğrenme ve modelin karmaşıklığı artabilir. Bu durumda, regülerleştirme teknikleri kullanılabilir, ancak bu da modelin yorumlanabilirliğini azaltabilir.

Eksik Veri Sorunu: Logistic Regression, eksik veri sorunlarına duyarlı olabilir. Eğitim veri setinde eksik veri varsa, modelin performansı düşebilir.

Kesikli Sonuçlar: Logistic Regression'ın çıktısı sürekli değil, kesikli (0 veya 1) bir sınıf olasılığıdır. Bu durum, bazı durumlarda doğru olasılık tahminleri yapmak zor olabilir.

1. 3. LOJİSTİK REGRESYON MİMARİSİ



Şekil 6. Lojistik Regresyon Mimarisi

Kullanacağımız veri setimizde *n* tane özellik olduğunu varsayalım ve $f_1, f_2, f_3, f_4, \ldots f_n$ şeklinde gösterelim. Lojistik regresyonda bu *n* tane özellik, ağırlıklar (weights) adını verdiğimiz $w_1, w_2, w_3, w_4, \ldots w_n$ sayılarına karşılık gelecek şekilde çarpılır. Daha sonra bir *b* (bias) değeri eklenerek bir *z* değeri elde edilir. Elde edilen bu *z* değeri, bir aktivasyon fonksiyonu olan sigmoid aktivasyon fonksiyonunda işleme tabi tutulur. Daha sonra elde edilen *y_head* çıktısı 0 ile 1 arasında olasılıksal (probabilistic) bir sonuç döndürür ve sınıflandırma işlemi gerçekleştirilir.

$z = f_1 * w_1 + f_2 * w_2 + ... + f_n.w_n$	Denklem (9)
$y\_head = sigmoid(z)$	Denklem (10)

NOT:
Aktivasyon Fonksiyonlarına ait detaylı anlatım, ilerleyen konu başlıklarında ayrıca verilmiştir. İncelemeniz ve anlamanız önemlidir.

Lojistik regresyon, yapay sinir ağları modelinin atası ve en ilkel modeli olarak kabul edilmektedir. Bu modellerin, diğer makine öğrenmesi modellerine göre farkı ortaya çıkan Loss değerini azaltmaktır. Loss değeri, gerçek *y* değeri ile tahmin edilen *y_head* değerinin birbirinden çıkarılması ile elde edilir. Tüm *y* değerlerine karşılık ayrı ayrı elde edilen Loss

değerleri Loss Function'ı oluşturmaktadır. Tüm Loss Function değerlerinin toplamı ise Cost Function'ı vermektedir. Daha açıklayıcı bir şekilde özetlersek;

Loss Function (Kayıp Fonksiyonu): Loss function, bir modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu ölçen bir fonksiyondur. Modelin performansını değerlendirmek için kullanılır. Daha düşük bir kayıp, modelin daha iyi performans gösterdiği anlamına gelir. Modelin eğitimi sırasında kullanılarak, gerçek değerlerle karşılaştırılır ve ardından bu kayıp, geriye doğru yayılım (backpropagation) algoritması kullanılarak ağırlıkların güncellenmesinde bir hata sinyali olarak kullanılır.

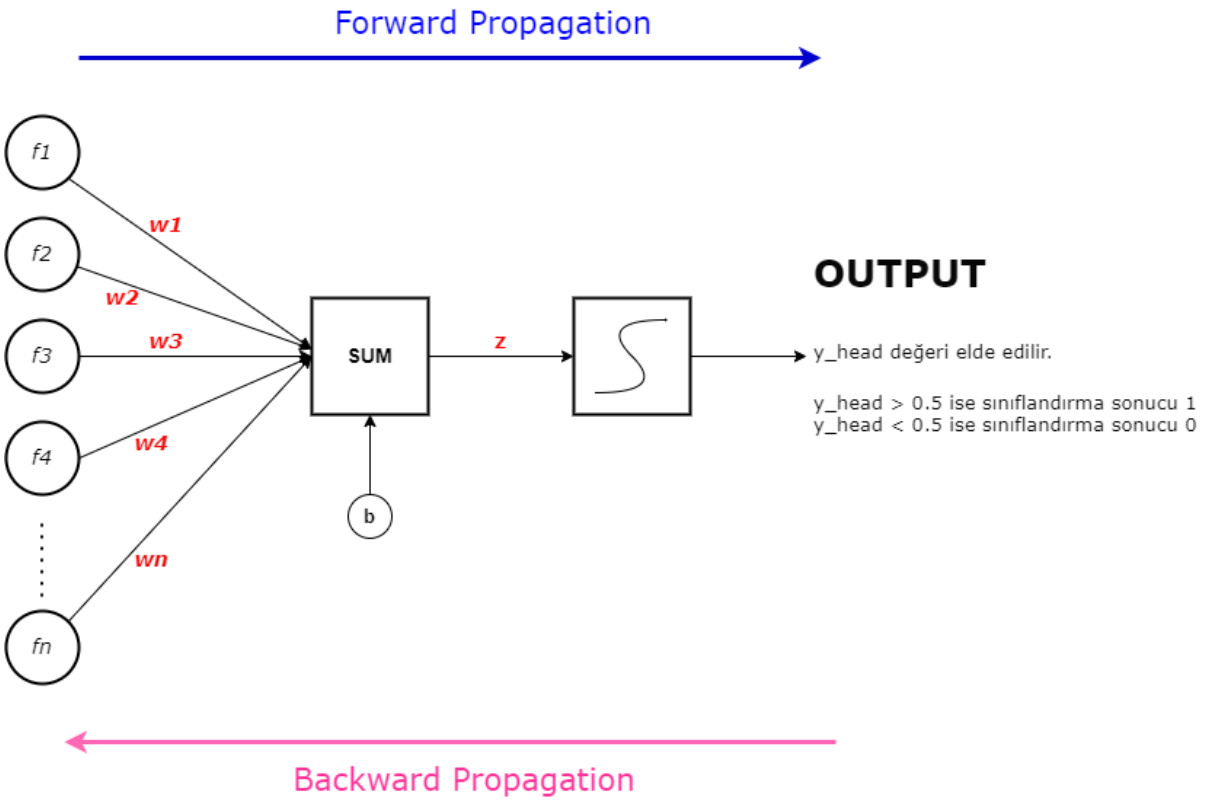
Cost Function (Maliyet Fonksiyonu): Cost function, genellikle bir veri kümesindeki tüm örneklerin kayıp fonksiyonlarının ortalamasıdır. Yani, cost function, bir modelin genel performansını değerlendirmek için kullanılan bir metrik sağlar. Eğitim sırasında, cost function, modelin ağırlıklarını güncelleme sürecinde optimize edilen hedefi temsil eder.

Lojistik regresyonun asıl amacı, Loss kaybını ve dolayısıyla Cost Function değerini azaltmak ve doğru tahmin değerini yüksek olasılıklı vermektedir. Bu yüzden, bu modelin tasarımında kullanılan aktivasyon fonksiyonlarının türevlenebilir fonksiyonlar olması önem arz etmektedir.

Lojistik regresyonda, modelin ilk olarak eğitilmesi ve ortaya çıkan ilk tahmin değeri sonucunda ortaya çıkan ilk Loss Function'un elde edilmesi süreci *Forward Propagation (İleri Yayılım)* adını alır. Loss Function değerinin azaltılması için, *w* (ağırlık), *b* (bias) değerlerinin güncellenmesi için aktivasyon fonksiyonun türevi alınır ve işlem tüm değerler güncellenerek başa döndürülür. Bu süreç ise *Backward Propagation (Geri Yayımlı)* ismini alır. Forward Propagation ve Backward Propagation işleminin kaç defa yapıldığına dair işlemin adı ise *iterasyon* adını almaktadır.

1. 4. LOGISTIC REGRESSION UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.



Şekil 7. Forward ve Backward Propagation

2. ARTIFICIAL NEURAL NETWORKS (ANNs)

Yapay Sinir Ağları (YSA), biyolojik sinir ağlarından esinlenerek tasarlanmış matematiksel modellerdir. Temel olarak, YSA'lar karmaşık işlemleri gerçekleştirmek üzere öğrenen ve adaptasyon yeteneğine sahip algoritmaları içerir. Yapay Sinir Ağları'nın temel bileşenleri ve çalışma prensipleri şunlardır:

Nöronlar: Yapay Sinir Ağları, yapay nöronlardan oluşan katmanlardan oluşur. Bu nöronlar, biyolojik sinir hücrelerine benzer şekilde bilgi işleme işlevini yerine getirir. Her nöron, girdi değerlerini (inputs) alır, bu girdileri belirli ağırlıklarla çarpar, bir toplam alır, bir aktivasyon fonksiyonuna tabi tutar ve ardından bir çıkış üretir.

Katmanlar: Yapay Sinir Ağları genellikle üç temel katmandan oluşur: giriş katmanı, gizli katmanlar ve çıkış katmanı. Giriş katmanı, modelin aldığı girdi değerlerini temsil eder. Gizli katmanlar, modelin öğrenme yeteneğini artırmak için kullanılan katmanlardır. Çıkış katmanı, modelin ürettiği sonuçları temsil eder.

Ağırlıklar ve Bağlantılar: Nöronlar arasındaki bağlantılar, ağırlıklar ile temsil edilir. Ağırlıklar, girdi değerleri ile çarpılarak nöronların aktivasyonlarını belirler. Eğitim süreci sırasında, bu ağırlıklar öğrenme algoritmaları kullanılarak güncellenir.

Aktivasyon Fonksiyonları: Her nöronun çıkışını belirleyen aktivasyon fonksiyonları, genellikle non-lineer fonksiyonlardır. Bu fonksiyonlar, YSA'nın karmaşıklığını artırmak ve çeşitli problem türlerine daha iyi uyan modeller oluşturmak için kullanılır.

Öğrenme Algoritmaları: Yapay Sinir Ağları, genellikle öğrenme algoritmaları kullanılarak eğitilir. Geriye doğru yayılım (backpropagation) en yaygın kullanılan öğrenme algoritmasıdır. Bu algoritma, modelin çıkışındaki hataları geriye doğru yayarak ve ağırlıkları bu hataları azaltacak şekilde güncelleyerek öğrenme sağlar.

Maliyet Fonksiyonları: YSA'nın performansını ölçmek için kullanılan maliyet fonksiyonları, modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu ölçer. Eğitim süreci sırasında bu fonksiyon, geriye doğru yayılım algoritması tarafından kullanılarak ağırlıkların güncellenmesinde bir hata sinyali olarak görev yapar.

2. 1. ANN AVANTAJLARI

Öğrenme Yeteneği: ANN'ler, büyük miktarda veri üzerinden öğrenme yeteneğine sahiptir. Karmaşık desenleri ve ilişkileri algılamak ve öğrenmek konusunda güçlüdürler.

Paralel İşleme: Yapay Sinir Ağları, birçok paralel işlemci üzerinde eşzamanlı olarak çalışabilir. Bu, büyük veri setleri üzerinde hızlı bir şekilde öğrenme yeteneğine katkıda bulunabilir.

Çeşitli Uygulama Alanları: ANN'ler, birçok farklı uygulama alanında başarıyla kullanılabilecek esneklik ve geniş bir uygulama yelpazesi sunar. Görüntü tanıma, doğal dil işleme, ses tanıma, sınıflandırma ve regresyon gibi birçok problem türüne uygundur.

Öznitelik Çıkarma: ANN'ler, veri setlerindeki önemli özellikleri çıkarma yeteneği ile özellikle öğrenme yeteneklerini artırabilirler. Bu, özellik mühendisliğine ihtiyaç duyulmadan otomatik olarak önemli özellikleri öğrenebilme anlamına gelir.

Duyarlılık ve Adaptasyon: Yapay Sinir Ağları, çevresel değişikliklere ve yeni veri örneklerine duyarlı bir şekilde adapte olabilirler. Bu, dinamik ve değişken veri setlerinde iyi performans göstermelerini sağlar.

Parçacık Bilimi (Fuzzy Logic) ve Belirsizlikle Başa Çıkma: Fuzzy logic ve belirsizlikle başa çıkma yetenekleri, ANN'leri belirsiz veya gürültülü veri setlerinde etkili kılar. Bu, gerçek dünya problemleriyle başa çıkarken avantaj sağlar.

Derin Öğrenme Yeteneği: Derin öğrenme (deep learning) alanı, çok katmanlı yapay sinir ağlarını içerir. Bu çok katmanlı yapılar, daha karmaşık özellikleri ve desenleri öğrenme yetenekleriyle bilinirler. Derin öğrenme, özellikle büyük veri setleri üzerinde etkili olduğu için bir avantaj sağlar.

Eğitilebilirlik ve Esneklik: ANN'ler, çeşitli eğitim algoritmaları ve aktivasyon fonksiyonları kullanılarak esnek bir şekilde yapılandırılabilirler. Bu, farklı problemlere uyacak şekilde özelleştirilebilmelerini sağlar.

Yüksek Boyutlu Veri İşleme Yeteneği: Yapay Sinir Ağları, yüksek boyutlu veri setleri üzerinde etkili bir şekilde çalışabilirler. Özellikle görüntü, metin veya ses gibi yüksek boyutlu verilerle çalışan modellerde başarılıdırlar.

2. 2. ANN DEZAVANTAJLARI

Veri İhtiyacı ve Büyük Veri Setleri: Yapay Sinir Ağları genellikle büyük miktarda veriye ihtiyaç duyar. Özellikle karmaşık modellerin eğitilmesi için büyük ve temsilci veri setlerine ihtiyaç vardır. Aksi takdirde, aşırı uydurma (overfitting) problemleri ortaya çıkabilir.

Hesaplama Gücü ve Kaynak İhtiyacı: Derin sinir ağları, çok sayıda parametre içerdiğinden hesaplama gücü ve kaynak ihtiyacı yüksektir. Bu, büyük modellerin eğitilmesi ve uygulanması için güçlü bilgi işlem kaynaklarını gerektirir.

Aşırı Uydurma (Overfitting): Yapay Sinir Ağları, aşırı uydurma eğiliminde olabilirler, yani eğitim verilerine çok iyi uyan ancak genelleme yeteneği zayıf olan modeller üretebilirler. Bu, özellikle küçük veri setlerinde ve çok karmaşık modellerde bir sorun olabilir.

İnsan Anlayışı Zorluğu: Yapay Sinir Ağları, genellikle "siyah kutu" modelleri olarak nitelendirilir, yani içsel işleyişleri tam olarak anlaşılamaz. Bu durum, modelin neden belirli bir tahminde bulunduğunu anlamak için zorluk yaratabilir.

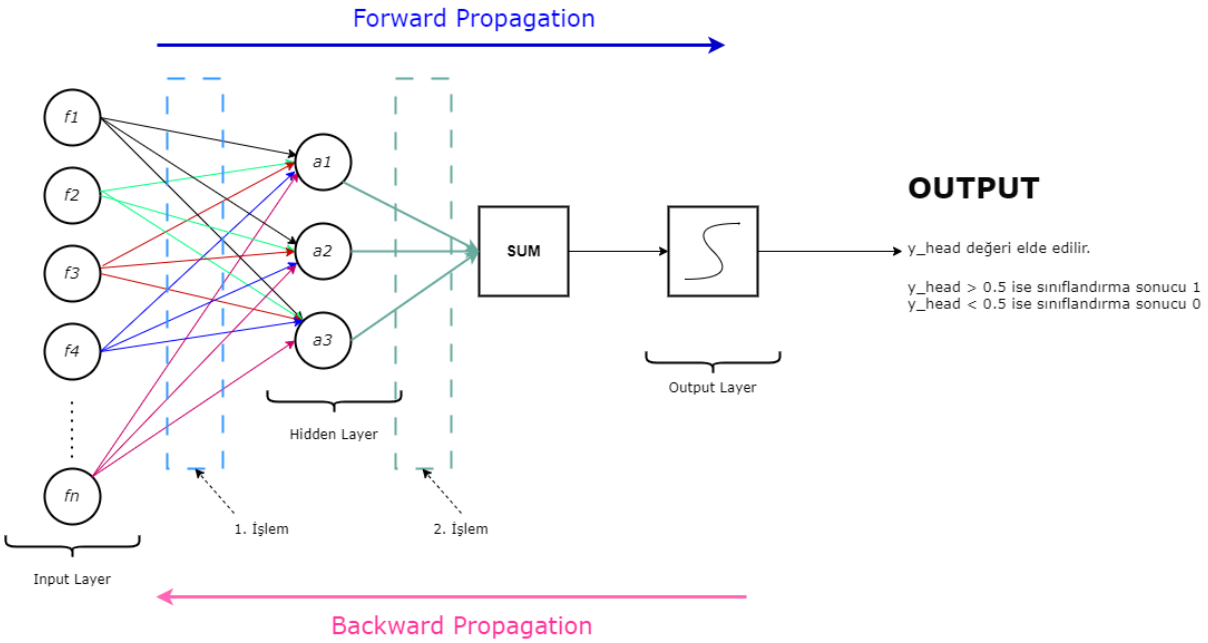
Hiperparametre Seçimi: Yapay Sinir Ağları, birçok hiperparametreye sahiptir ve bu parametrelerin optimal değerlerini bulmak zor olabilir. Doğru hiperparametre seçimi, modelin performansını belirleyebilir.

Eğitim Zorluğu: Büyük ve derin sinir ağları, eğitilmeleri zaman alabilir ve özellikle doğru bir şekilde ayarlanmadıklarında eğitim sırasında sorunlar yaşanabilir.

Bağımlılık ve İletişim Sorunları: Yapay Sinir Ağları, genellikle bağımsız gözlemler arasındaki bağımlılıkları modelleme eğilimindedir. Bu durum, doğrusal olmayan ilişkileri daha iyi ele almasına rağmen, bazı durumlarda gereksiz karmaşıklığa neden olabilir.

Güvenilirlik Sorunları: Yapay Sinir Ağları, gürültülü veri veya etiketleme hataları gibi güvenilirlik sorunlarına duyarlı olabilir. Bu durum, modelin yanlı veya hatalı tahminlere neden olabilir.

2. 3. ANN MİMARİSİ



Şekil 8. ANN Mimarisi

Yapay Sinir Ağlarının tasarımı temel farklarla birlikte Lojistik Regresyona benzmektedir. Şekil-8'de yer ANN mimarisine yakından göz atacak olursak; $f_1, f_2, f_3, f_4, \ldots f_n$ şeklinde belirtilen özellikleri modele girdi olarak verelim. ANN'de bu kısım *Giriş Katmanı (Input Layer)* olarak adlandırılır. Modele girdi olarak verilen özellikler $w_1, w_2, w_3, w_4, \ldots w_n$ ağırlıkları ile çarpılır. Bu işlem Şekil-8'de yer alan 1. İşlem kısmında yapılır. Bu kısımda elde edilen z değeri bir aktivasyon fonksiyonu olan *tanh* (Tangent Hyperbolic) ile aktivasyon işlemine tabi tutulur. *gizli katmanın (Hidden Layer)* girdisini oluşturur. Gizli Katman, dış dünyaya doğrudan gözlemlenemeyen, gizli olan özellikleri ve karmaşıklıkları öğrenmek için kullanılır. Bu katman, giriş ve çıkış katmanları arasında bulunur. Gizli katmana, girdi olarak verilecek z değerinin

aktivasyon işleminden sonra elde edilen yeni değerler, sırasıyla Denklem 11 ve Denklem 12'deki gibidir.

$z_1 = X * w_1 + b_1$	Denklem (11)
$A_1 = \tanh(z_1)$	Denklem (12)

Denklem 11 ve 12 incelendiğinde, gizli katmana gelmeden önce tüm özellikler, ağırlık değerleri ile çarpılır. Denklem 11 genelleştirilmiş gösterimdir. Tüm özellikler ve ağırlık birbirleri ile çarpıldıktan sonra elde edilen z değeri tanh aktivasyon işlemine tabi tutularak, gizli katmanın girdisini oluşturacak bir A değeri elde edilir. X değeri güncellendiği için, artık X değerinin yerini A değeri almaktadır. Denklemler yer alan 1 ifadesi 1. İşlem adımına vurgu yapmaktadır.

A değeri gizli katmana girdi olarak verildikten sonra aynı işlemler tekrarlanır. Çıkış Katmanına (Output Layer) girdi olarak verilecek değerler, sigmoid aktivasyon fonksiyonu ile işleme tabi tutulduktan sonra verilir. Bu işlem, Şekil-8'de 2. İşlem olarak gösterilmiştir. Bu katmandan elde edilen bir sonraki katmana girdi olarak verilecek veriler Denklem 13 ve Denklem 14'te gösterilmiştir.

$z_2 = A_1 * w_1 + b_2$	Denklem (11)
$A_2 = \text{sigmoid}(z_2)$	Denklem (12)

A_2 değeri, çıkış katmanında sigmoid fonksiyonu ile olasılıksal değeri hesaplanır ve lojistik regresyonda olduğu gibi sınıflama işlemini gerçekleştirir. Denklemler incelediğinde ANN mimarisinde işlemler, lojistik regresyon modelinde yer alan işlemlerle aynıdır. ANN mimarisindeki işlemleri, Lojistik regresyonun iki kere tekrar edilmesi şeklinde düşünebiliriz.

2. 4. ANN UYGULAMASI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

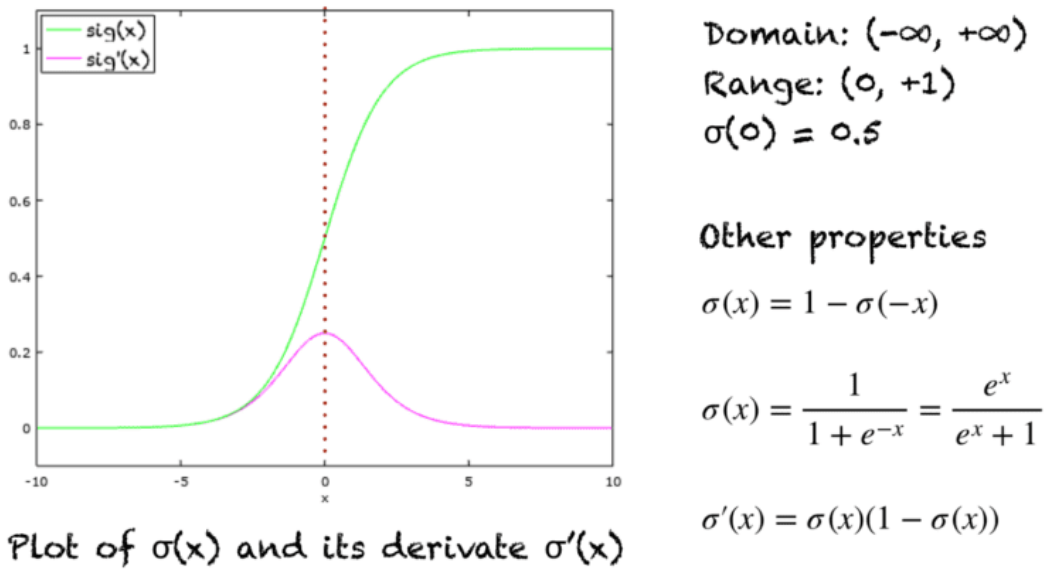
3. AKTİVASYON FONKSİYONLARI

NOT:

Bu kısım, aktivasyon fonksiyonları anlatıldıkça güncellenecektir.

3. 1. SIGMOID AKTİVASYON FONKSİYONU

Sigmoid aktivasyon fonksiyonu, yapay sinir ağlarında sıklıkla kullanılan bir aktivasyon fonksiyonudur. Bu fonksiyon, girdi değerini 0 ile 1 arasında bir çıkışa dönüştürür.



Şekil 9. Sigmoid Aktivasyon Fonksiyonu

Şekil-9 incelendiğinde, sigmoid fonksiyonuna ait grafik görünmektedir. Yeşil renk ile belirtilen çizgi sigmoid fonksiyonunun kendine ait grafiği belirtirken, pembe renk ile belirtilen çizgi sigmoid fonksiyonun türevine ait grafiği temsil eder.

Sigmoid fonksiyonunun önemli özellikleri şunlardır:

Sınırlı Çıkış Aralığı: Sigmoid fonksiyonu, çıkışını 0 ile 1 arasında sınırlar. Bu özellik, özellikle ikili sınıflandırma problemlerinde, çıkışı bir olasılık değeri olarak yorumlamak için uygundur.

Yatay Eğimi Vardır: Sigmoid fonksiyonunun türevi (eğimi) her noktada var olduğu için, geriye doğru yayılım algoritması gibi optimizasyon tekniklerinde kullanılabilecek bir sürekli ve diferansiyasyon yapılabilen bir fonksiyondur.

Sigmoid fonksiyonunun bazı dezavantajları vardır:

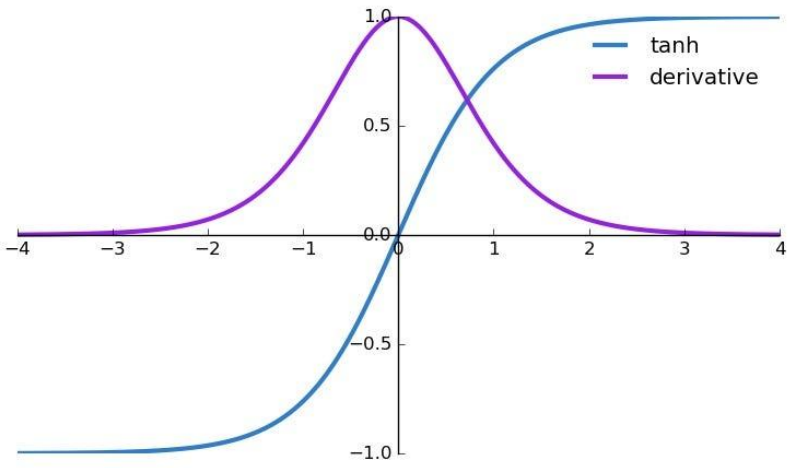
Vanishing Gradient Problem: Sigmoid fonksiyonu, giriş değerleri çok büyük veya çok küçük olduğunda, eğimi çok küçük olma eğilimindedir. Bu durum, geriye doğru yayılım sırasında ağırlıkların güncellenmesi sırasında çok küçük gradyanlarla çalışmayı gerektirir, bu da "vanishing gradient problem" olarak adlandırılan bir soruna yol açabilir.

Merkezi Eğilme Sorunu: Sigmoid fonksiyonunun çıkışı her zaman pozitiftir ve 0'dan uzaklaştıkça eğilimi 0'a yaklaşır. Bu, bir sinir ağının belirli durumlarda öğrenmesini yavaşlatabilir.

Sigmoid fonksiyonuna ait grafik görüntüsü bu web adresinden alınmıştır.

3. 2. TANH AKTİVASYON FONKSİYONU

Tanh (Hiperbolik Tanjant) aktivasyon fonksiyonu, yapay sinir ağlarında kullanılan bir başka popüler aktivasyon fonksiyonudur. Tanh fonksiyonu, sigmoid fonksiyonuna benzer, ancak çıkış aralığı -1 ile 1 arasındadır.



Şekil 10. Tanh Aktivasyon Fonksiyonu

$$tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Şekil-10 incelendiğinde, tanh fonksiyonuna ait grafik görünmektedir. Mavi renk ile belirtilen çizgi tanh fonksiyonunun kendine ait grafiği belirtirken, mor renk ile belirtilen çizgi sigmoid fonksiyonun türevine ait grafiği temsil eder.

Tanh fonksiyonunun özellikleri şunlardır:

Sınırlı Çıkış Aralığı: Tanh fonksiyonunun çıkışı -1 ile 1 arasında sınırlıdır. Bu, özellikle çıkışın normalize edilmiş bir aralıkta olması gerektiği durumlarda kullanışlıdır.

Ortalaması Sıfırdır: Tanh fonksiyonunun çıkış aralığı -1 ile 1 olduğu için, fonksiyonun ortalaması sıfırdır. Bu, bir sonraki katmana daha dengeli bir giriş sağlayabilir.

Vanishing Gradient Problem: Sigmoid fonksiyonunda olduğu gibi, tanh fonksiyonu da büyük giriş değerleri için doygunluk (saturation) eğilimindedir ve bu durum vanishing gradient problemine yol açabilir.

Tanh fonksiyonu, genellikle iç katmanlarda ve özellikle çıkış katmanında kullanılabilir.

Tanh fonksiyonuna ait grafik görüntüsü bu web adresinden alınmıştır.

BÖLÜM - V

1. MAKİNE ÖĞRENMESİ MODELLERİNİN HİPERPARAMETRE OPTİMİZASYONU

Makine öğrenimi modellerinin performansını artırmak ve genelleme yeteneklerini iyileştirmek için hiperparametre optimizasyonu önemli bir adımdır. Yaygın hiperparametre optimizasyon teknikleri aşağıdaki konu başlıklarında verilmiştir.

1. 1. GRID SEARCH

Grid Search (Grid Arama), makine öğrenimi modellerinde performansı artırmak için kullanılan bir hiperparametre optimizasyon yöntemidir. Bu yöntem, belirli bir hiperparametre uzayında önceden belirlenmiş bir alt küme içinde tüm kombinasyonları deneyerek en iyi hiperparametre setini bulmayı amaçlar. Grid Search, daha spesifik optimizasyon algoritmalarının aksine basit ve anlaşılır bir yaklaşım sunar.

Grid Search'in adımları ve çalışma prensibi şu şekildedir.

1. Hiperparametre Uzayının Belirlenmesi: İlk adım, optimize edilecek modelin hiperparametre uzayını belirlemektir. Örneğin, bir destek vektör makinesi (SVM) modeli düşünüldüğünde, C ve kernel tipi gibi hiperparametreler belirlenir.
2. Optimize Edilecek Hiperparametre Kombinasyonlarının Belirlenmesi: Hiperparametre uzayı belirlendikten sonra, bu hiperparametrelerin belirli değerlerini içeren bir alt küme seçilir. Her hiperparametre için bir veya birkaç değer belirlenir.
3. Modelin Eğitilmesi ve Performansın Değerlendirilmesi: Belirlenen hiperparametre kombinasyonlarıyla model eğitilir ve bir performans ölçütü (genellikle çapraz doğrulama ile belirlenen bir metrik) kullanılarak değerlendirilir.
4. Tüm Kombinasyonların Denenmesi: Belirlenen alt kümedeki tüm hiperparametre kombinasyonları sırayla denenecektir. Her kombinasyon için model eğitilir ve performans değerlendirilir.
5. En İyi Hiperparametre Kombinasyonunun Seçilmesi: Tüm kombinasyonlar denendikten sonra, en iyi performansa sahip olan hiperparametre kombinasyonu seçilir. Bu genellikle en yüksek doğruluk, en düşük hata veya başka bir performans metriği ile belirlenir.

Grid Search'in Avantajları:

- Basit ve anlaşılır bir optimizasyon yöntemidir.
- Sistemli bir arama yapar ve hiperparametre uzayının genelini kapsar.
- Kullanımı kolaydır ve birçok makine öğrenimi kütüphanesinde (örneğin, scikit-learn) standart bir fonksiyon olarak bulunur.

Grid Search'in Dezavantajları:

- Büyük hiperparametre uzaylarında hesaplama gücü ve zaman açısından maliyetli olabilir.
- Hiperparametreler arasındaki etkileşimleri dikkate almaz.
- En iyi kombinasyon, veri setine ve modelin karmaşıklığına bağlı olarak değişebilir.

Grid Search, özellikle daha küçük hiperparametre uzaylarında etkili olabilir, ancak daha büyük ve karmaşık uzaylarda diğer optimizasyon yöntemleri (örneğin, random search, bayesian optimization) daha verimli olabilir.

1. 2. RANDOM SEARCH

Random Search (Rastgele Arama), makine öğrenimi modellerinde hiperparametre optimizasyonu için kullanılan bir yöntemdir. Grid Search'in aksine, belirli bir hiperparametre uzayındaki kombinasyonları sıralı bir şekilde değil, rastgele seçilen kombinasyonlarla değerlendirir. Random Search, özellikle büyük hiperparametre uzaylarında daha etkili olabilir, çünkü daha az deneme ile genellikle iyi sonuçlar elde edebilir.

Random Search'in adımları ve çalışma prensibi şu şekildedir:

1. Hiperparametre Uzayının Belirlenmesi: İlk adım, optimize edilecek modelin hiperparametre uzayını belirlemektir. Bu hiperparametrelerin her biri için bir değer aralığı belirlenir.
2. Optimize Edilecek Hiperparametre Kombinasyonlarının Rastgele Seçilmesi: Hiperparametre uzayındaki her bir hiperparametre için belirlenen değer aralıklarından rastgele değerler seçilir. Bu rastgele seçilen değerlerle bir hiperparametre kombinasyonu oluşturulur.
3. Modelin Eğitilmesi ve Performansın Değerlendirilmesi: Rastgele seçilen hiperparametre kombinasyonları ile model eğitilir ve belirlenen bir performans metriği (örneğin, doğruluk, hata) kullanılarak değerlendirilir.

Belirli Bir Sayıda İterasyonun Tamamlanması: Belirli bir sayıda rastgele hiperparametre kombinasyonu oluşturulup modeller eğitildikten sonra, en iyi performansa sahip olan hiperparametre kombinasyonu seçilir.

Random Search'in Avantajları:

- Hiperparametre uzayındaki aralıklarda rastgele seçim yaparak sistemli bir şekilde keşif yapar.
- Grid Search'e kıyasla daha hesaplamalı açıdan verimli olabilir, çünkü her bir kombinasyonun değerlendirilmesi için daha az zaman ve kaynak harcar.
- Büyük hiperparametre uzaylarında daha etkili olabilir.

Random Search'in Dezavantajları:

- Grid Search gibi hiperparametreler arasındaki etkileşimleri dikkate almaz.
- Hiperparametre uzayındaki belirli bir bölgeye odaklanma şansı düşük olabilir.
- Random Search'in her çalıştırılmasında farklı sonuçlar alınabilir.
- Random Search, hiperparametre optimizasyonunda kullanılan popüler bir yöntemdir ve model performansını iyileştir.

Random Search, hiperparametre optimizasyonunda kullanılan popüler bir yöntemdir ve model performansını iyileştirmek için özellikle büyük ve karmaşık hiperparametre uzaylarında etkili olabilir.

1. 3. BAYESIAN OPTIMIZATION

Bayesian optimization, bir modelin performansını değerlendirip, bu bilgiyi kullanarak hiperparametre uzayında yeni değerler seçen bir yöntemdir. Bu, optimize edilmemiş alanlara odaklanarak daha verimli bir şekilde hiperparametre alanını keşfetmeye çalışır.

1. 4. GENETİK ALGORİTMALAR

Genetik algoritmalar, genetik süreçleri (seçilim, çaprazlama, mutasyon) kullanarak popülasyon içindeki hiperparametre kombinasyonlarını evrimleştiren bir optimizasyon yöntemidir.

1. 5. GRADIENT-BASED OPTIMIZATION

Hiperparametre optimizasyonu için gradyan tabanlı yöntemler de kullanılabilir. Bunlar, hiperparametre uzayındaki gradienti takip ederek en iyi kombinasyonu bulmaya çalışırlar.

1. 6. HİPERPARAMETRE OPTİMİZASYON UYGULAMALARI

Çalışma için kullanılan veri seti ve uygulama kodları https://github.com/balfatih/YAZ20411_Deep_Learning_2025_Fall_Semester linkinden erişebilirsiniz.

BÖLÜM - VI

1. CONVOLUTIONAL NEURAL NETWORKS

Evrişimli Sinir Ağları (Convolutional Neural Networks-CNN) ızgara benzeri bir topolojiye sahip veriyi işlemek için kullanılan özel bir tür sinir ağıdır. Evrişim, özel bir tür doğrusal işlemdir. Daha basit tanımıyla evrişimsel ağlar, katmanlardan en az birinde genel matris çarpımı yerine evrişim kullanan sinir ağlarıdır. CNN modelleri genel yapı itibariyle temel olarak beş katmandan oluşmaktadır. Bunlar; giriş katmanı (input layer), evrişim katmanı (convolutional layer), ortaklama/havuzlama katmanı (pooling layer), tam bağlantı katmanı (fully connected layer) ve çıkış katmanıdır (output layer).

1. 1. GİRİŞ KATMANI

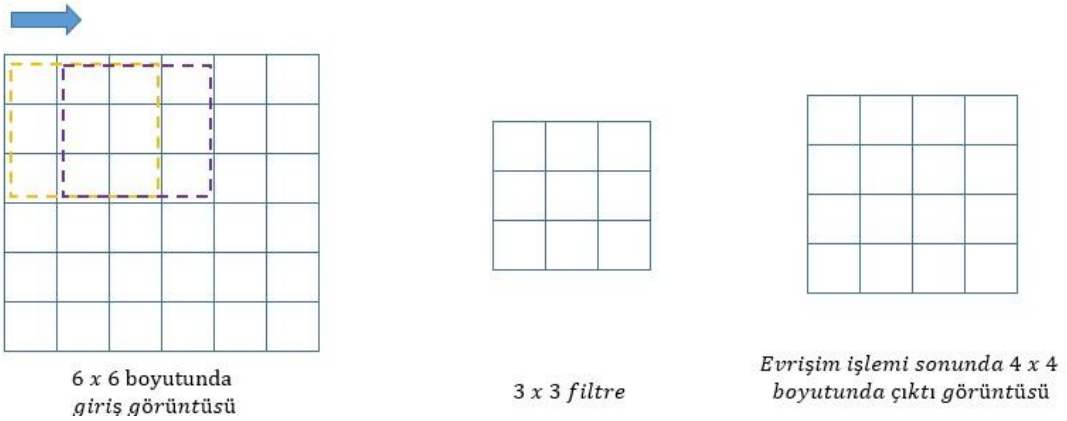
Evrişimli Sinir ağlarında ilk katman olan giriş katmanında (input layer) gözlemler sisteme aktarılır. Görüntüler, piksel değerleri şeklinde bir matris olarak temsil edilir ve bu matrisler evrişim katmanı için beslenmektedir.

$$giris\_boyutu = (goruntu\_genisligi, goruntu\_yuksekligi, kanal\_sayisi) \quad (1)$$

Görüntüler evrişim katmanına verilirken Denklem (1)'de yer alan ifade gibi verilmektedir. Sırasıyla görüntü genişliği, görüntü yüksekliği ve görüntünün kanal sayısı belirtilir. Kanal sayısı gri tonlamalı görüntüler için 1, renkli görüntüler (RBG) için 3 değeri almaktadır.

1. 2. CONVOLUTION (EVRİŞİM) KATMANI

CNN ağlarına adını veren evrişim katmanı (Convolutional Layer), nöron bağlantılarının evrişim işlemi ile sağlandığı ileri beslemeli bir ağ katmanıdır. Evrişim katmanı öznitelik çıkarıcı katman olarak da adlandırılır. Çünkü görüntüye ait tüm öznitelik çıkarımları bu katmandan elde edilmektedir. Evrişim işlemi yaparken görüntü üzerine bir filtre uygulanmaktadır. Filtre, girdi görüntüsü veya görüntüler üzerindeki bir dizi filtrenin eylemiyle tanımlanan bir katman yığınıdır. Görüntü veya görüntüye ait öznitelikler çekirdek boyutu belirtilerek evrişim katmanında yapılmaktadır. CNN'de uygulanan filtrenin biçimine veya türüne karar verilememektedir. Evrişim katmanında sadece filtre boyutu sağlanmaktadır. Çekirdek değerleri evrişim katmanında belirli ağırlıkları ifade etmektedir. Çekirdek boyutuna karar verildikten sonra eğitim süresince filtreler kendiliğinden öğrenilmektedir.



Şekil 1. Evrişim işlemi.

Şekil 1'de yer alan 6 x 6 görüntü üzerinde 3 x 3 boyutunda bir filtre uygulandığında evrişim işlemi filtrenin görüntü üzerinde dolaştırılmasıyla elde edilmektedir. Filtre görüntü üzerinde gezdirildikten sonra elde edilen yeni görüntünün boyutu Denklem (2)'deki gibi hesaplanmaktadır.

$$cikis\_boyutu = \frac{n - f + 2p}{s} + 1 \qquad (2)$$

Denklem (2)'de;

- n ifadesi giriş görüntüsünün genişliğini,
- s kaydırma (stride) değerini,
- f filtre boyutunu ve genişliğini
- p dolgulama (padding) ifade etmektedir.

CNN modellerinde aksi belirtilmedikçe dolgulama değeri varsayılan olarak 0 ve kaydırma değeri 1 olarak alınmaktadır.

1. 2. 1. DOLGULAMA (PADDING) İŞLEMİ

Şekil 1'de gösterildiği gibi bir giriş görüntüsünün üzerinde evrişim işlemi filtrenin kaydırılması sonucunda yapılmakta ve çakışan sayılar birbirleriyle çarpılmaktadır. Çarpım sonucunda elde edilen sayılar toplanarak öznitelik haritasına (feature map) aktarılmaktadır. Ancak bir görsele filtre uygulanıp evrişim işlemine tutulduğunda çıkış görüntüsünün boyutu, giriş görüntüsünün boyutuna göre eksik olmaktadır. Bu durum, piksel kaybını oluşturmaktadır. Piksel kaybını önlemek için dolgulama (padding) yöntemi kullanılmaktadır. Dolgulama işlemi sırasında giriş

görselinin etrafına bir çerçeve oluşturacak şekilde sıfır (0) değerleri eklenmektedir. Bu şekilde görüntü üzerinde filtre gezdirildiğinde piksel kaybı olmaz ve çıkış görüntüsünün boyutu, giriş görüntüsünün boyutu ile aynı olmaktadır.

1. 2. 2. KAYDIRMA (STRIDE) İŞLEMİ

Kaydırma (Stride) işlemi filtrenin görsel üzerindeki piksellerde kaç adım kayarak ilerleyeceğini belirlemektedir. Şekil 1'de kaydırma işlemi bir kaydırma değeri alarak devam etmiştir. CNN'de kaydırma değeri varsayılan değer olarak bir değerini almaktadır. İki veya daha yüksek bir kaydırma değeri verildiğinde kaydırma işlemi yapıldığında piksel kaybı ortaya çıkmaktadır.

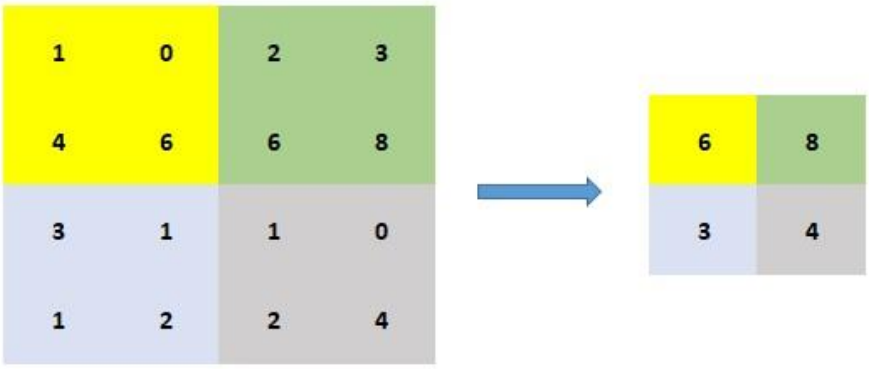
1. 3. TOPLU NORMALLEŞTİRME (BATCHNORMALIZATION) KATMAN

Toplu Normalleştirme (Batch Normalization-BN), derin sinir ağlarının ara katmanlarında aktivasyonları normalleştirmeye yönelik bir tekniktir. CNN modellerinde bir katmanın öğrenmeye başlaması için bir önceki katmanın öğrenmesini bitirmesi gerekmektedir. Dolayısıyla, bir önceki katmanın çıktısı bir sonraki katmanın giriş değerlerini oluşturmaktadır. Bir katmanda yapılan parametre değişiklikleri, kendinden sonraki gelen katmanlarda da değişikliğe yol açacağı için yapılan her parametre değişikliğinde hesaplama maliyeti ve zaman maliyeti artmaktadır. Bu maliyeti azaltmak için CNN modellerinde BN uygulanmaktadır. BN, tüm giriş değerlerinin ortalamasını sıfır ve standart sapmasını bir olacak şekilde düzenlemektedir. Değerler -1 ve +1 aralığında sıkıştırılmaktadır. BN sayesinde CNN'deki katmanlar, kendinden önceki katmanın öğrenmesini beklemek zorunda kalmaz ve eş zamanlı öğrenmeye olanak sağlayarak eğitim sürecini hızlandırılmaktadır.

1. 4. HAVUZLAMA/ORTAKLAMA (POOLING) KATMAN

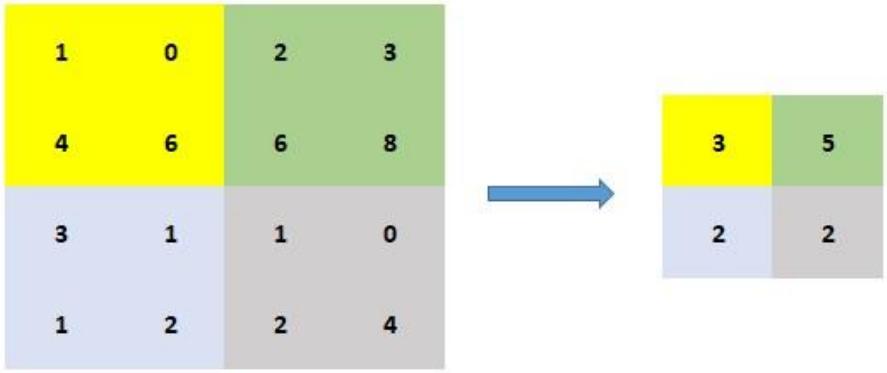
Havuzlama (Pooling) Katmanları, CNN modellerinin yapı taşlarından birini oluşturmaktadır. Bu katmanın amacı, evrişimli katmanların görüntülerden öznitelikleri çıkardığı yerde öğrenilen öznitelikleri birleştirerek ağdaki parametre ve hesaplama sayısını en aza indirgemektir. Bu işlem ile girdinin uzamsal boyutu kademeli olarak küçültülmektedir. Havuzlama işleminin maksimum havuzlama ve ortalama havuzlama olmak üzere iki farklı yöntemi bulunmaktadır.

Maksimum Havuzlama (Max Pooling) her havuzdan maksimum değeri seçerek çalışmaktadır. Öznitelik haritasının en belirgin özelliklerini korur ve elde edilen görüntü, giriş görüntüsünden daha keskindir. Şekil 2'de maksimum havuzlama örneği yer almaktadır.



Şekil 2. Maksimum havuzlama işlemi.

Ortalama Havuzlama (Average Pooling), her havuzdan ortalama değeri seçerek çalışmaktadır. Öznitelik haritasının ortalama değerini korumaktadır. Görüntünün öznitelik özünü korurken, görüntüyü pürüzsüzleştirmektedir. Şekil 3'te ortalama havuzlama örneği yer almaktadır.



Şekil 0. Ortalama havuzlama işlemi.

1. 5. DÜĞÜM SEYRELTME (DROPOUT) KATMAN

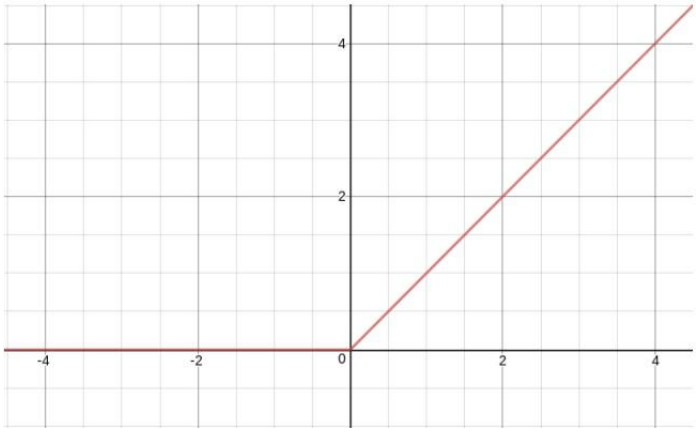
Düğüm Seyreltme (Dropout-DS), aşırı öğrenmeyi engellemek için uygulanan bir tekniktir. DS, gizli katmandan veya giriş katmanından rastgele düğümleri kaldırarak düğümleri daha sağlam olmaya zorlar ve öznitelikleri diğer düğümlere bağlı olmadan kendi başına öğrenmektedir. Kaldırılacak düğüm sayısı bir hiperparametre tarafından belirlenmektedir. DS tekniği ile düğümler arası bağlantılar koparılarak düğümlerin birbirlerinin sonuçlarından daha az etkilenmesi sağlanmaktadır.

1. 6. YOĞUN (DENSE) KATMAN

Dense Layer (Yoğun Katman-YK), bir önceki katmanla derinlemesine bağlı olan bir katmandır. Bu durum, katmanın nöronlarının önceki katmanın her bir nöronuna bağlı olduğu anlamına gelmektedir. Bir CNN modelinin sonunda yoğun katman (YK) kullanılmaktadır. Bir YK'nın girdisi, önceki evrişim ve havuzlama katmanları tarafından çıkarılmış olan özniteliklerin bir vektörüdür. Vektörde yer alan her öğe, girdinin belirli bir sınıfa ait olma olasılığını temsil etmektedir. YK, girdi vektörünün ağırlık matrisi ile matris çarpımını gerçekleştirmektedir. Daha sonra bir yanlılık terimi ekleyerek girdinin doğrusal bir dönüşümü ve ardından doğrusal olmayan bir aktivasyon işlemi gerçekleştirilmektedir.

1. 7. AKTİVASYON FONKSİYONLARI

Sigmoid ve Hiperbolik Tanjant fonksiyonlarından sonra belirli bir eşik değerinden sonra kaybolan gradyan problemi ortaya çıkmaktadır**.** Bu sebepten dolayı kaybolan gradyan problemini ortadan kaldırmak amacıyla Doğrutulmuş Doğrusal Birim (Rectified Linear Unit-ReLU) aktivasyon fonksiyonu geliştirilmiştir. ReLU, türev işlemini gerçekleştirebilen doğrusal olmayan bir aktivasyon fonksiyonudur. Şekil 4'te ReLU aktivasyon fonksiyonu gösterilmektedir.



Şekil 4. ReLU aktivasyon fonksiyonu.

ReLU, kaybolan gradyan değerlerinin üstesinden türev alarak gelebildiği için CNN modellerinde çok yaygın olarak kullanılmaktadır. Hesaplama yükünün diğer fonksiyonlara göre düşük olması, hızlı ve verimli çalışması ReLU'nun avantajı olarak görünürken, negatif değerlerin sıfır (0) olması dezavantajı olarak görülmektedir.

Denklem (0)'te yer alan x ifadesi negatif bir değer alırsa sonucu sıfır olarak döndürürken, pozitif bir değer alması durumunda $+\infty$ değerini almaktadır.

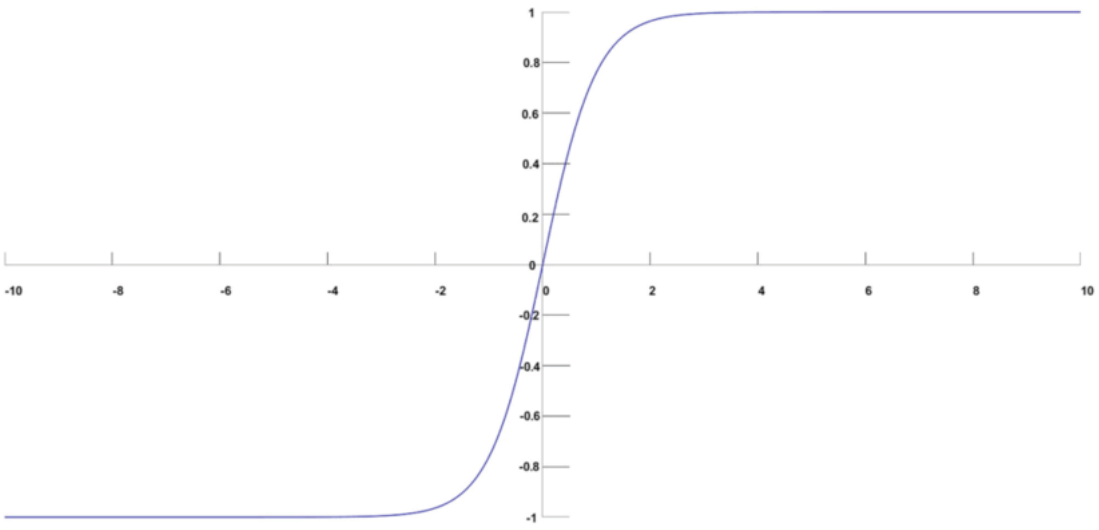
$$F(x) = \max(0, x) \tag{0}$$

Bir diğer aktivasyon fonksiyonu olan Softmax aktivasyon fonksiyonu, genellikle sigmoid fonksiyonların bir kombinasyonu olarak adlandırılmaktadır. Sigmoid ikili sınıflandırmalar için uygun bir aktivasyon fonksiyonu olarak kullanılırken, Softmax aktivasyon fonksiyonu çok sınıflı hedef çıktılar üreten CNN modelleri için uygun bir aktivasyon fonksiyonudur. Şekil 5'te Softmax aktivasyon fonksiyonu koordinat düzleminde gösterilmektedir.

Denklem (4)'te softmax girdi işlevi olarak K gerçek sayıdan oluşan bir z vektörünü alır ve bunu girdi sayılarının üstelleriyle orantılı K olasılıktan oluşan bir olasılık dağılımına normalleştirir. Softmax, her sınıfa ait bir olasılık sonucu üretir. Olasılık sonuçlarının değeri 0 ile 1 arasındadır. Bu aktivasyon fonksiyonu CNN modellerinin son katmanında kullanılmaktadır.

$$R^K \to (0,1)^K, \qquad K \geq 1$$

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \qquad için\ i = 1,2,..., K, \qquad z = (z_1, z_2, ..., z_K) \in R^K \tag{4}$$



Şekil 5. Softmax aktivasyon fonksiyonu.