AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Zeynep ÜNAL

LİKERT TİPİ VERİLERDE BULANIK MANTIK VE DERİN ÖĞRENME ENTEGRASYONU

Ekonometri Ana Bilim Dalı
Doktora Tezi

Antalya, 2019

AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Zeynep ÜNAL

LİKERT TİPİ VERİLERDE BULANIK MANTIK VE DERİN ÖĞRENME ENTEGRASYONU

Danışman
Doç. Dr. Emre İPEKÇİ ÇETİN

Ekonometri Ana Bilim Dalı
Doktora Tezi

Antalya, 2019

Akdeniz Üniversitesi

Sosyal Bilimler Enstitüsü Müdürlüğüne,

Zeynep Ünal'ın bu çalışması, jürimiz tarafından Ekonometri Ana Bilim Dalı Doktora Programı tezi olarak kabul edilmiştir.

Başkan	:Prof. Dr. Ayşe KURUÜZÜM	(İmza)
Üye (Danışmanı)	:Doç. Dr. Emre İPEKÇİ ÇETİN	(İmza)
Üye	:Doç. Dr. Mehmet MERT	(İmza)
Üye	:Doç. Dr. Ömür TOSUN	(İmza)
Üye	:Doç. Dr. Hilal KAZAN	(İmza)

Tez Başlığı:	LİKERT TİPİ VERİLERDE BULANIK MANTIK VE DERİN ÖĞRENME ENTEGRASYONU

Onay : Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Tez Savunma Tarihi : 13/06/2019

Mezuniyet Tarihi : 27/06/2019

(İmza)

Prof. Dr. İhsan BULUT

Müdür

AKADEMİK BEYAN

Doktora Tezi olarak sunduğum “Likert Tipi Verilerde Bulanik Mantik ve Derin Öğrenme Entegrasyonu” adlı bu çalışmanın, akademik kural ve etik değerlere uygun bir biçimde tarafımca yazıldığını, yararlandığım bütün eserlerin kaynakçada gösterildiğini ve çalışma içerisinde bu eserlere atıf yapıldığını belirtir; bunu şerefimle doğrularım.

İmza

Zeynep ÜNAL

T.C.
AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
TEZ ÇALIŞMASI ORİJİNALLİK RAPORU
BEYAN BELGESİ

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ'NE

ÖĞRENCİ BİLGİLERİ	
Adı-Soyadı	**Zeynep ÜNAL**
Öğrenci Numarası	**20155245003**
Enstitü Ana Bilim Dalı	**Ekonometri**
Programı	**Doktora**
Programın Türü	**( ) Tezli Yüksek Lisans (X) Doktora ( ) Tezsiz Yüksek Lisans**
Danışmanının Unvanı, Adı-Soyadı	**Doç. Dr. Emre İPEKÇİ ÇETİN**
Tez Başlığı	**Likert Tipi Verilerde Bulanik Mantik ve Derin Öğrenme Entegrasyonu**
Turnitin Ödev Numarası	**1131958142**

Yukarıda başlığı belirtilen tez çalışmasının a) Kapak sayfası, b) Giriş, c) Ana Bölümler ve d) Sonuç kısımlarından oluşan toplam 169. sayfalık kısmına ilişkin olarak, 17/05/2019 tarihinde tarafımdan Turnitin adlı intihal tespit programından Sosyal Bilimler Enstitüsü Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Uygulama Esasları'nda belirlenen filtrelemeler uygulanarak alınmış olan ve ekte sunulan rapora göre, tezin/dönem projesinin benzerlik oranı;

alıntılar hariç % 2

alıntılar dahil % 3 'tür.

Danışman tarafından uygun olan seçenek işaretlenmelidir:

(x) Benzerlik oranları belirlenen limitleri aşmıyor ise;

Yukarıda yer alan beyanın ve ekte sunulan Tez Çalışması Orijinallik Raporu'nun doğruluğunu onaylarım.

( ) Benzerlik oranları belirlenen limitleri aşıyor, ancak tez/dönem projesi danışmanı intihal yapılmadığı kanısında ise;

Yukarıda yer alan beyanın ve ekte sunulan Tez Çalışması Orijinallik Raporu'nun doğruluğunu onaylar ve Uygulama Esasları'nda öngörülen yüzdelik sınırlarının aşılmasına karşın, aşağıda belirtilen gerekçe ile intihal yapılmadığı kanısında olduğumu beyan ederim.

Gerekçe:

Benzerlik taraması yukarıda verilen ölçütlerin ışığı altında tarafımca yapılmıştır. İlgili tezin orijinallik raporunun uygun olduğunu beyan ederim.

17/05/2019

(imzası)
Danışmanın Unvanı-Adı-Soyadı

İ Ç İ N D E K İ L E R

İKİNCİ BÖLÜM
DERİN ÖĞRENME

ÜÇÜNCÜ BÖLÜM
BULANIK MANTIK VE DERİN ÖĞRENME ENTEGRASYONU

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR LİSTESİ

RELU	:Rektifiye Edilmiş Lineer Birim (Rectified Linear Unit)
ELU	:Üstel Lineer Birim (Exponential Linear Units)
PReLU	:Parametrik Lineer Birim (Parametrik Lineer Birim)
OKH	:Ortalama Karesel Hata (Mean Square Error - MSE)
MHO	:Mutlak Hatalar Ortalaması (Mean Absolute Error – MAE)
OYH	:Ortalama Yanlılık Hatası
SVM	:Destek Vektör Makineleri (Support Vector Machines)
FOM	:Birinci Derece Metotları (First order methods)
SOM	:İkinci Derece Metotları (Second order methods)
ADAGRAD	:Uyarlanmış Gradyan Algoritması (ADAptive GRADient algorithm)
ADAM	:Uyarlanmış Momentum Tahmini (ADAptive Moment estimation)
NDAM	:Nesterov ile hızlandırılmış ADAM (Nesterov-accelerated ADAM)
GPU	:Grafik İşleme Ünitesi (Graphics Processing Unit)
CPU	:Merkezi İşleme Ünitesi (Central Processing Unit)
ROC	:Alıcı Çalışma Karakteristiği (Receiver Operating Characteristic)
AUC	:ROC Eğrisi Altındaki Alan (Area Under the Curve)

ÖZET

Son yıllarda, yapay zeka tekniklerinden insan beyninin çalışma şeklinden esinlenerek geliştirilen ve yapay sinir ağları ilkeleri üzerine inşa edilen derin öğrenme yaklaşımları, verilerin tanınması ve sınıflandırılmasında etkin yöntem olması sebebiyle büyük önem kazanmıştır.

Bu tez çalışmasının amacı, 5'li Likert tipi ölçeğiyle üretilen yapay veri setlerinin üçgensel ya da yamuk bulanık sayılar kullanılarak bulanık forma dönüştürülmesi ve bu yolla verilerin çoğalması durumunda derin öğrenme tekniklerinin performansının analiz edilmesidir. Çalışmada derin öğrenme ve bulanık mantık tekniklerinin entegrasyonu sonucunda önerilen modelin performansının test edilmesi için belirsizlik ve karmaşık ilişkiler içeren memnuniyet tahmin problemi seçilmiştir. Bulanık sayılarla oluşturulan veri setleri ile normal veri setinden en az 3 ya da 4 kat daha fazla parametre sayısına ulaşılmakta ve büyük veri ile optimizasyon çalışmalarında global optimizasyona ulaşmak yerine yerel optimuma tuzağına düşme sorunu ortadan kalkmaktadır. Bu çalışmada verilerin analizinde öncelikle birçok sınıflandırma tekniğinin temelini oluşturan lojistik regresyon gibi klasik bir yöntemle pilot çalışma yapılmıştır. Ardından yapay zeka teknikleri içerisinde literatürde sağladığı avantajlardan ötürü ilgi odağı olan derin öğrenme tekniğinin performansı bulanık mantık çerçevesinde değerlendirilip karşılaştırılmıştır. Derin öğrenme ile yapılan analizlerde de hem literatürdeki bulanıklaştırmaya örnek teşkil eden tepe, maksimum ve minimum değerler için ayrı ayrı sonuçlarla durulaştırmaya gidilmiş hem de literatürden farklı olarak bulanık sayıların tek sonuç dizisi üretmesi önerilerek derin öğrenme modelinin performansları araştırılmıştır. Derin öğrenme konusunda Türkçe literatürün sınırlı olması sebebiyle bu çalışmanın Türkçe literatüre katkı sağlayacağı düşünülmektedir.

Anahtar Kelimeleri: Yapay Zeka, Derin Öğrenme, Bulanık Mantık, Likert Ölçeği

SUMMARY

In recent years, the deep learning approaches, which have been developed from the Artificial Intelligence techniques based on the way that the human brain works and which are built on the principles of artificial neural networks, have gained great importance because it is an effective method in the recognition and classification of data.

The aim of this thesis is to analyze the performance of deep learning techniques in the form of a 5-point Likert-type scale by converting the artificial data sets into a fuzzy form using triangular or trapezium fuzzy numbers. In order to test the performance of the proposed model which is the integration of deep learning and fuzzy logic techniques, the satisfaction estimation problem involving uncertainty and complex relations was chosen. The data sets generated by fuzzy numbers are at least 3 or 4 times parameters more than the normal data set. In the optimization studies with this data, the problem of falling into the local optimum is eliminated and it reaches to the global optimization point. In this study, first of all pilot study was done by using a classical method such as logistic regression which forms the basis of many classification techniques. Then, the performance of deep learning technique which is in the center of attention due to the advantages, has been evaluated in the framework of fuzzy logic. In the analysis conducted with deep learning, it has been clarified with separate results for peak, maximum and minimum values which constitute an example of blurring in literature.

In contrast to the literature, it was suggested that fuzzy numbers produce a single result sequence and the performances of the deep learning model were investigated. Also, it is thought that this study will contribute to Turkish literature because of the limited literature in Turkish about deep learning.

Keywords: Artificial Intelligence, Deep Learning, Fuzzy Logic, Likert Scale

ÖNSÖZ

Tez konusunun seçimi sırasında ve çalışmanın her aşamasında değerli katkıları ile destek olan danışman hocam Doç. Dr. Emre İPEKÇİ ÇETİN'e teşekkürlerimi sunarım.

Ayrıca Doktora eğitimim esnasında desteklerini hiçbir şekilde esirgemeyen ve öğrendiklerimi uygulamaya dönüştürmek için beni teşvik eden tüm Ekonometri Bölümü öğretim üyelerine teşekkür ederim.

Araştırmalarıma katkıda bulunan ve destek veren yakın çevrem, dostlarım ve manevi desteğini hep yanımda hissettiğim sevgili eşim ve çocuklarıma teşekkürü bir borç bilirim.

Zeynep Ünal

GİRİŞ

Son yıllarda, derin öğrenme yaklaşımları, etiketlenmemiş veriden hiyerarşik gösterimler oluşturmaya yarayan bir yöntem olarak önem kazanmıştır. Karmaşık, yüksek boyutlu verilere anlam kazandırmak ihtiyaç haline gelmiştir. Derin makine öğrenme, denetimli veya denetimsiz öğrenme sürecinde verilerin tanınması ve sınıflandırılmasında etkin ve doğru bir yöntemdir (robotomy.eu, 2016). Daha hızlı bilgisayar işlemcilerinin üretilmesiyle, belleklerin ucuzlamasıyla ve yeni veri biçimlerinin ortaya çıkmasıyla, derin öğrenme tekniklerinin her ölçekteki işletme için gerçek zamanlı veri analizinde kullanılması sağlanmıştır(Lewis, 2016: 6).

Düşünen ve karar verebilen makinelerin hayali Antik Yunan zamanlarına kadar dayanmaktadır. Pygmalion, Daedalus ve Hephaestus gibi mitolojik figürler Galatea, Talos ve Pandora gibi efsanevi karakterleri da yapay yaşam formu olarak yorumlamak mümkündür. Düşünen bilgisayarların fikri ise, bilgisayarların icat edilmesinin yüz yıl öncesine dayanmaktadır. Bugün, yapay zeka hala gelişmekte olan alan olarak görülse de birçok alanda konuşma veya görüntüleri anlamak, rutin işleyişi otomatize etmek, tıpta teşhis koymak ve temel bilimsel araştırmaları desteklemek gibi pratik uygulamalar halinde hayatımıza girmiştir (Goodfellow vd., 2016: 1).

Yapay zekanın literatüre girdiği ilk günlerde insanlar için entellektüel olarak zor gelen fakat matematiksel kuralların listesi halinde tanımlanabildiği sürece bilgisayarlar için basit olan problemler üzerine çalışmalar yapılmıştır. Yapay zekanın alanı genişledikçe ve geliştikçe bilgisayarların deneyimden faydalanması, sesi ve görüntüyü tanıması, sezgisel kararlar vermesi üzerine çalışmalar ilgi odağı olmuştur (Goodfellow vd., 2016: 1).

Yakın zamana kadar, çoğu makine öğrenimi ve sinyal işleme tekniğinde sığ yapılı mimarileri kullanılmıştır. Bu mimariler genellikle doğrusal olmayan dönüşüm içeren en fazla bir veya iki katmandan oluşmaktadır. Sığ mimarilere örnek olarak Gauss karışım modelleri, doğrusal veya doğrusal olmayan dinamik sistemler, maksimum entropi modelleri, lojistik regresyon, kernel regresyonu, tek bir gizli katmanı olan çok katmanlı perceptronlar verilebilir. Sığ mimariler birçok basit veya iyi sınırlanmış problemi çözmede etkili olmasına rağmen insan konuşması, doğal ses ve dil gibi doğal sinyaller içeren daha karmaşık gerçek dünya uygulamaları

konusunda yetersiz kalmıştır. Derin öğrenme ile beraber bu verinin işlenmesi mümkün olmuştur (Deng ve Yu, 2014: 205).

Piksel kümesinden oluşan obje görüntüsünün ve insan sesinin bilgisayarlar tarafından anlaşılması ve veri kümesinin bir nesne kimliği ile eşleşmesi çok zor bir işlemdir. Bu işlem, basit bir öğrenme modeli olarak kurgulanmaya çalışılırsa bu eşleştirme çok karmaşık hale geleceği için başarısız olması muhtemeldir. Derin öğrenme ise bu karmaşıklığı hiyerarşilere bölerek ele aldığı için bu zorluğu daha kolay aşmaktadır. İç içe basit eşleştirmeleri haritalar olarak modelledikten sonra, her bir model farklı bir katmanda değerlendirilmekte ve sonucu daha üst katmana ileterek bütün sonuca ulaşılmaya çalışılmaktadır (Goodfellow vd., 2016: 5).

1940'lardan beri sığ mimari olarak kullanılan tek katmanlı yapay sinir ağlarının da böyle veriyi işleme yeteneği mevcut değildir. Daha karmaşık verinin işlenmesi için daha derin mimarilere ihtiyaç duyulmuştur. 1980'lerde karmaşık sinir ağlarının başarılı bir şekilde eğitilmesinden sonra sinir ağlarının etkin bir şekilde kullanılması mümkün olmuştur. Bu gelişme de daha karmaşık ve daha derin mimarilerin tasarlanmasının yolunu açmıştır. Sinir ağların popülaritesinin artmasından itibaren çok fazla inişli çıkışlı dönemler geçirmiştir. Şu anda ise derin öğrenmeyi kullanan yapar sinir ağları oldukça büyük ilgi görmektedir (Heaton, 2015: 165).

Derin öğrenme, sağlık bakım endüstrisinde, tıbbi görüntü işleme, doğal dil işleme ve reklamcılıkta tıklama oranlarını artırmak için ticari olarak kullanılmaktadır. Microsoft, Google, IBM, Yahoo, Twitter, Baidu, Paypal ve Facebook gibi büyük şirketler hedeflenen hizmet ve ürünleri önerebilmeleri için kullanıcının tercihlerini derin öğrenmeyi kullanarak anlamaktadır. Sesli yardım teknolojisinin temelini oluşturduğu derin öğrenme akıllı telefonlarda bile karşımıza çıkmaktadır (Lewis, 2016: 10).

Bu bilgilerden hareketle son yıllarda hızlı gelişen teknolojiyle ve bu teknolojilerin kullanımı dolayısıyla oluşan verinin büyüklüğü nedeniyle analizi için ihtiyaç duyulan tekniklerin etkinliğinin araştırılması bu çalışmanın temel motivasyonunu oluşturmaktadır. Her geçen gün yeni tekniklerin literatüre girdiği yapay zeka alanında probleme ilişkin en uygun tekniğin seçilmesi veri analizinde verilmesi gereken en önemli kararlardan biridir.

Çalışmanın ilk bölümünde yapay zeka kavramı detaylı şekilde ele alınmıştır. Yapay zeka kavramı açıklanmaya çalışılırken tarihsel gelişiminde yer alan önemli

detaylara yer verilmiş, ilkel tekniklerden daha gelişmiş tekniklere doğru bir akış takip edilmiştir. Tekniklerin hangi ihtiyaçtan dolayı ortaya çıktığı, hangi probleme çözüm getirdiği konusu irdelenmiştir. Bazı geleneksel tekniklerin sağladığı avantajlardan dolayı günümüzde halen kullanıldığı ve gelişmiş teknikler için bir temel oluşturduğu vurgulanmıştır. Bu tekniklerden biri olan lojistik regresyon tekniği birçok sınıflandırma tekniğinin temelini oluşturması sebebiyle ayrıca ele alınmıştır. Diğer yapay zeka tekniklerinden kullanım şekliyle ayrılan bulanık mantık konusu detaylı şeklide açıklanmıştır. Bulanık mantığın, belirsizlik durumunda diğer yapay zeka tekniklerin etkinliğini artırma özelliğine dair örnekler verilmiştir.

Çalışmanın ikinci bölümünde yapa zeka araştırma alanının alt dalı olan Makine Öğrenmesi tekniklerinden literatürde yer alan en güncel ve gelişmiş olan teknikler araştırılmıştır. Makine öğrenme ilkelerinin büyük bölümünün uygulandığı Yapay Sinir ağları tekniğinin büyük veri kavramı ile beraber ön plana çıktığına değinilmiştir. Son yıllarda araştırmacıların bu alana gösterdiği ilgi yapay sinir ağları tekniğinin birçok eksiğinin giderilmesine sebep olmuştur. Daha önce tek gizli katmandan fazla eğitilemeyen yapay sinir ağları, yeni yaklaşımlar kullanılarak çok katmanlı ağlara dönüşme ve "Derin Öğrenme" kavramının ortaya çıkma süreci açıklanmıştır.

Derin öğrenme tekniğinde kullanılan güncel aktivasyon ve kayıp fonksiyonlarına, optimizasyon algoritmalarına, aşırı öğrenme probleminin çözüm yollarına, öğrenme sürecini verimli hale getirecek parametre ayarlarına ve derin öğrenme platformlarına yer verilmiştir. Derin öğrenme konusunda Türkçe literatürün sınırlı olması sebebiyle bu çalışmanın Türkçe literatüre katkı sağlayacağı düşünülmektedir.

Çalışmanın üçüncü ve son bölümünde derin öğrenme tekniğinin etkinliğinin bulanık mantık kullanılarak arttırma fikri genişletilmiştir. Bu iki tekniğin entegrasyonu sonucunda önerilen modelin performansının test edilmesi için memnuniyet tahmin problemi seçilmiştir. Memnuniyet tahmininde en yaygın kullanılan Likert ölçeği, bulanık sayılar kullanılarak bulanık Likert ölçeğine dönüştürülmüştür.

Deneysel çalışmanın tekrarlanabilir bulgular oluşturulması amaçlandığından, farklı koşullar altında sınıflandırma modellerinin performansının kapsamlı bir şekilde araştırılması yönünden avantajlı olan yapay veri üretme tekniklerinden yararlanılmıştır. Derin öğrenme mimarisinin etkinliği birçok parametreye bağlı

sağlanan başarının tesadüfî olmadığını göstermek amacıyla üretilen yapay veri geleneksel teknik olan ve birçok sınıflandırma tekniğinin temelini oluşturan lojistik regresyon tekniği ile test edilmiştir. Üretilen Likert tipi veri ile bulanık Likert tipi veri Lojistik Regresyon modeli kullanılarak sınıflandırılarak başarı ölçütleri incelenmiştir. Ayrıca Bulanık Lojistik Regresyon modeli kurularak sınıflandırma sonuçları karşılaştırılmıştır. Yapılan testler hem üçgensel hem de yamuk bulanık sayılar kullanılarak yapılmıştır.

Likert tipi verinin bulanık veriye dönüştürülmesinin modelin başarısını arttırdığı ve verinin derin öğrenme modelinin testi için uygun olduğu sonucuna varıldıktan sonra derin öğrenme mimarisinin tasarımı aşamasına geçilmiştir. Likert tipi veri ile bulanık Likert tipi verinin parametre sayılarında farklılık söz konusu olduğunda her mimari için ayrı ayrı mimari seçim prosedürü takip edilmiştir. Her veri tipi için uygun mimari seçildikten sonra modeller üretilen yapay veri ile test edilerek sonuçlar raporlanmıştır.

Farklı koşullar altında modellerin sınıflandırma performansını kapsamlı bir şekilde araştırmak için önce kuadratik denklem kullanılarak veri daha karmaşık veriye dönüştürülmüştür. Bu veri kümeleri kullanılarak modelleri test edilmiş ve yorumlanmıştır. Daha sonra veri sınıf dengesizliği ve gürültü içerecek şekilde üretilmiş veriyle modeller daha zorlu hale getirilerek analiz edilmiştir. Çalışmanın genelini kapsayan bulgular üzerine değerlendirmeler ve literatür bulguları ile karşılaştırmalar yapılmıştır.

BİRİNCİ BÖLÜM
YAPAY ZEKA

Yapay zeka basit anlamda, herhangi bir gözlemciye zeki görünecek şekilde davranma yeteneği olarak tanımlanmaktadır. Yapay zeka içeren sistemler, karmaşık problemleri çözmek için insanların ve diğer hayvanların davranışlarından esinlenerek geliştirilen metotları kullanmaktadır (Coppin, 2004: 4).

Dünyanın en karmaşık makinesi olarak kabul edilen insan beyninin, zor sayısal işlemlerin yapması dakikalarını alırken idrak etmeye yönelik işlemleri yapması çok daha kısa sürede gerçekleşmektedir. Çok karmaşık sayısal işlemleri saniyeler içinde yapabilen bilgisayarlar ise idrak etme konusunda oldukça yetersizdir. İnsan beynini taklit ederek düşünebilen ve belirsizlik ortamında mantıklı kararlar verebilen makineler yapma fikri yapa zeka kavramının ortaya çıkmasına neden olmuştur (Elmas, 2001: 21).

19. yüzyıldaki endüstri devrimi sırasında, insanın fiziksel iş gücünden tasarruf etme ve yapılan işleri hızlandırma için makinelerin yoğun kullanımı bilim ve teknolojinin önemini arttırmıştır. Yirminci yüzyılda bilişim teknolojilerinin ortaya çıkması yapay zekanın gelişimini doğurmuştur. Akıllı davranışlar sergileyebilen ve öğrenebilen makineler sadece fiziksel değil zihinsel işgücünü devralabilecek konuma gelmiştir. Bu akıllı davranışlar arasında algı, hafıza, duygu, yargılama, muhakeme, kanıtlama, tanımlama, anlama, iletişim, tasarım, düşünme ve öğrenme vb. şeklinde sıralanmaktadır (Li ve Du, 2008: 2).

Düşünebilen yani zekası olan makinelerin hayalinin milattan öncesine dayandığı kabul edilmesine rağmen fikrin gelişmesinde etkili olan önemli olaylar bulunmaktadır. Yapay zeka tarihindeki en büyük figürlerden biri Alan Turing'dir. II. Dünya Savaşı sırasında, Turing Bletchley Park'ta çalışarak Almanların kodlarını çözmüştür. Savaştan sonra, düşünebilecek bir bilgisayar kurma fikri üzerinde çalışmaya başlamıştır. 1950'de "Bilişim Makineleri ve Zeka" adlı makalesi, bu konuda yazılan ilk makalelerden biri olmuştur (Coppin, 2004: 4).

Yapay zekanın araştırma alanına dönüşmesi 1956 yılında John McCarthy, Marvin Minsky, Nathaniel Rochester ve Claude Shannon tarafından düzenlenen, "makineler kullanarak insan zekasını taklit etme" konulu Dartmouth sempozyumu sonrasında olmuştur. İki ay devam eden sempozyuma matematik, nörofizyoloji, psikiyatri, psikoloji, bilgi teorisi ve bilgisayar bilimi gibi çeşitli alanlardan onlarca

bilim adamı davet edilmiştir. Bu sempozyumda John McCarthy tarafından "Yapay Zeka" kavramı önerildiğinden, Dartmouth sempozyumu ilk yapay zeka sempozyumu ve John McCarthy Yapay Zekanın babası olarak kabul edilmektedir (Li ve Du, 2008: 3).

1950'lerden bu yana yapay zekanın başlangıçta kurgulanan amacının dışına çıkarak gerçekçilik derecesi eklenerek yeni şekil aldığı görülmektedir. Yapay zeka çalışmaların amacı artık insan kadar zeki bir robot yaratmak değil, insan beyninin problemleri çözme şeklini temel alan algoritmalar, sezgisel yöntemler ve metodolojiler geliştirmektir (Coppin, 2004: 9). Böylece insan beyninin problemleri çözme şekillerine odaklanan uzman sistemler, genetik algoritmalar, bulanık mantık ve makine öğrenmesi gibi alt dallar doğmuştur (Elmas, 2001: 21). Günümüzde yapay zeka akademik alanda oldukça yaygındır. Yapay zeka uygulamalarına her geçen gün yenileri eklenirken, çoklu sistem analizi, doğal dil işleme, örüntü tanıma, robotik kontrol ve bilgisayar görüşü iyi bilinen ve yaygın örneklerindendir (NRC, 1997).

1.1. Uzman Sistemler

Uzman sistemler kavramı, makinenin bir uzman gibi belirli bir alanda daha önce tanımlanmış bilgileri kullanarak düşünebilmesi ve karar vermesidir. Uzman sistemlerin karar verdiği alanda yeni bilgilerin ortaya çıkması durumunda uzman tarafından yeni kurallar şeklinde sisteme tanıtılması gerekmektedir. Bu nedenle bu sistemler, kural tabanlı sistemler veya bilgi tabanlı sistemler olarak da adlandırılmaktadır. İlk başarılı çalışma sistemlerinden birisi olan kan enfeksiyonlarını teşhis eden için MYCIN tıbbi sistem, yaklaşık 450 kuralı kullanarak pratisyen doktordan daha iyi ve bazı uzman doktorlar kadar iyi kararlar vermiştir (Warwick, 2013: 25).

Yapay zekanın ilk uygulaması olarak kabul edilen uzman sistemler, belli bir alanda uzmanlığa sahip bir kişinin bilgi ve karar alma becerilerini taklit eden bir programı olarak da tanımlanmaktadır. Bu programlar aynı zamanda insanların zor kararlar vermesi durumlarında ikinci bir görüş sunarak insanların kararlar almasına yardımcı olmak için tasarlanmaktadır. Uzman sistemler bilgi tabanı ve çıkarım motoru olmak üzere iki bölümden oluşur. Bilgi tabanı oluşturulurken birçok uzmanla görüşülerek bilgi toplanmakta ve sisteme tanımlanmaktadır. Uzman sistemin ikinci kısmı olan çıkarım motoru, yönergeleri yorumlayan ve bir karar vermek için bilgileri değerlendiren bir mantık programıdır. Bu program "Güneş parlıyorsa, o zaman

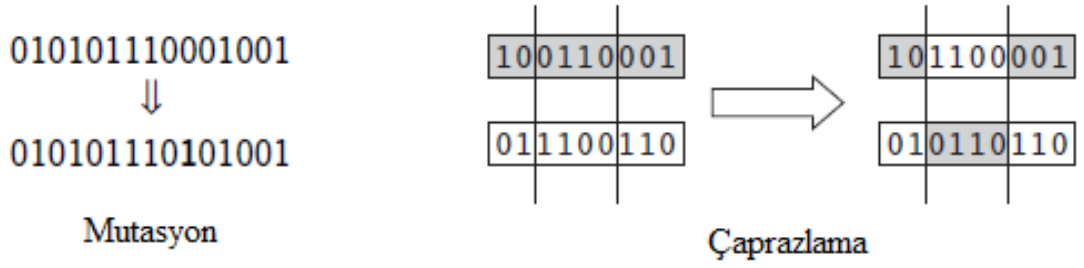
gündüz olmalı.” gibi önermeleri kullanarak sonuçlar üretmektedir. Her çıkarım motoru binlerce eğer komutu içermektedir. Günümüzde uzman sistemler halen borsa, bankacılık ve askeri savunma alanlarında kullanılmaktadır (Thomas, 2005: 26).

1.2. Genetik Algoritmalar

Uzman sistemler tek problemin çözümünde etkili bir tekniktir. Ancak hızlı değişen şartların olduğu durumlarda sürekli yeni kuralların tanımlama zorunluluğu araştırmacıları değişebilen kuralların olduğu sistemler tasarlamaya itmiştir (Thomas, 2005: 26). Biyolojik genetik biliminin mutasyon ve çaprazlama kavramlarından esinlenerek kural değiştirici algoritmalar kurgulanmış ve bu algoritmalar genetik algoritmalar olarak adlandırılmıştır (Boden, 1996: 286).

Genetik algoritmalar “en güçlülerin hayatta kalması” yaklaşımı kullanılarak geliştirildiğinden literatürde evrimsel algoritmalar olarak da adlandırılmaktadır. Genetik algoritmalar uygun şekilde kodlandığı takdirde gerçek dünyadaki sorunlara çözümler “geliştirebilme” yeteneği barındırmaktadır (Fulcher, 2006: 8).

Genetik algoritmalar biyolojik genetikte olduğu gibi mutasyon ve çaprazlama olmak üzere temel iki fonksiyonu kullanmaktadır. Mutasyon, bir kuralda rastgele bir değişiklik yapmaktadır. Çaprazlama ise iki kuraldan birinin sol kısmı diğerinin sağ kısmı ile birleştirilecek şekilde iki kuralı karıştırmaktadır (Boden, 1996: 286). Şekil 1.1’de mutasyon ve çaprazlama işlemi gösterilmiştir. Mutasyon işleminde ikili sayı içerisindeki 10. hanede bulunan 0 değeri mutasyon sonrası 1 değerini almaktadır. Çaprazlama işleminde ilk ikili sayıların her biri 3 parçaya bölünerek iki yeni ikili sayı oluşturmaktadır (Coppin, 2004: 392).



Şekil 1.1 Genetik Algoritmalarda Mutasyon ve Çaprazlama (Coppin, 2004: 392).

Kuralların kırılma noktaları rastgele seçilmekte veya sistemin en yararlı olan parçaları seçilerek birleştirilmektedir. Genetik algoritma sistemlerinin çoğu başarılı olan kuralları ve kural bölümlerini tanımlamak için mekanizmalar içermektedir. Bir kural başarı olarak tanımlandığında gelecek nesillerde "üreme" için seçilme

olasılıklarını artırmaktadır. Her yeni nesil bir öncekine göre daha başarılı olduğu için kurallar iyileştirilmektedir. Böylece bu algoritmalar göreve daha iyi adapte olmuş bir sistem oluşturmaktadır (Boden, 1996: 286).

1.3. Bulanık Mantık

Giriş ve çıkış arasındaki ilişkinin bir transfer fonksiyonu ile ifade edilmesinde doğru matematiksel model seçilmesi önemlidir. Ancak pratikte verinin çok karmaşık olması ve belirsizlik içermesi durumunda uygun model seçmek oldukça zordur. Verinin belirsizlik içermesi durumunda modeller bulanık mantıkla entegre edilerek performans artışı sağlanmaktadır (Li ve Du, 2008: 37).

1960'lı yıllarda Lotfi Zadeh, kümelerin kademeli üyeliğini tanımlayarak bulanık mantığın temelini atmıştır. Belirsizliklerle mücadele edilen gerçek dünyada birçok kavramın matematikten ziyade kelimelerle daha iyi tanımlanabileceğine inanan Zadeh, bulanık mantık ve bulanık kümeler matematiğin dili ve insan zekasını birbirine bağlama imkanı vereceğinin görüşündeydi. Geleneksel ölçekler yerine bulanık ölçeklerin ya da diğer ismiyle dilsel ölçeklerin kullanımıyla beraber insanın düşünce yapısını taklit ederek geliştirilen teknolojilerinin de önünü açmıştır (McNeill ve Thro, 2014).

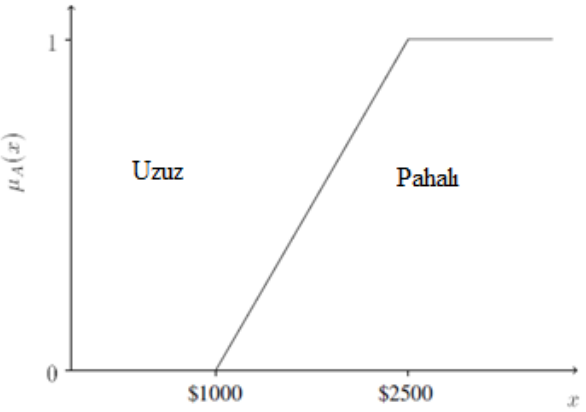
Bulanık kümeler kavramı ortaya çıktığından beri imalat, otomasyon ve diğer mühendislik alanlarında başarı sağlamıştır. Bu alanlarda başarı sağlamasının ardından sosyal bilimler araştırmalarında da yaygın olarak kullanılmaya başlanmıştır (Li, 2013).

Bulanık mantık konusunda tanımlamalar yapıldığında "küme teorisi" için "klasik küme teorisi" ifadesi kullanılmaktadır. Klasik küme teorisinde bir ***x*** sayısının A kümesinin elemanı olup olmadığı (1.1) şeklinde gösterilmektedir (Jager, 1995: 14).

$$\mu_A(x) = \begin{cases} 1, eğer\ x \in A \\ 0, eğer\ x \notin A \end{cases} \qquad (1.1)$$

Klasik küme teorisinde x sayısının A kümesinin elemanı olup olmadığı net olduğundan bu sayılar kesin sayılar (crisp numbers) olarak adlandırılmaktadır. Ancak birçok sınıflandırma problemlerinde kesinlik söz konusu değildir. Örneğin bir öğrenci bilgisayar satın alacağı zaman fiyatını sorduğu bilgisayarları "pahalı" ya da "ucuz" olarak sınıflandırmak istediğinde kesin bir eşik değeri belirlemesi mümkün olmamaktadır. Bu durumda "pahalı" ile "ucuz" arasındaki sınır bulanık olduğu için Şekil 1.2'de gösterildiği gibi bulanık küme ile tanımlanabilmektedir. Yani

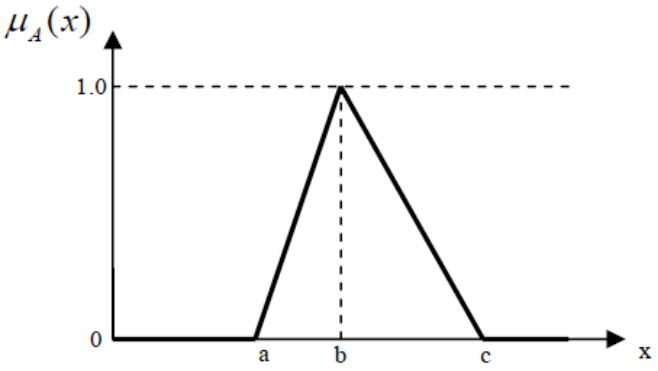
bilgisayarın pahalı olup olmama durumu 0 ile 1 arasında değişmektedir (Jager, 1995: 14).



Şekil 1.2 "Pahalı" ve "Ucuz" Kavramlarının Bulanık Küme ile Gösterimi (Jager, 1995: 14)

1.3.1. Üçgensel Bulanık Sayılar

Üyelik fonksiyonu a,b ve c gibi üç değerle aşağıdaki gibi gösterilir (Şen, 2009: 361).



Şekil 1.3 Üçgen Üyelik Fonksiyonu

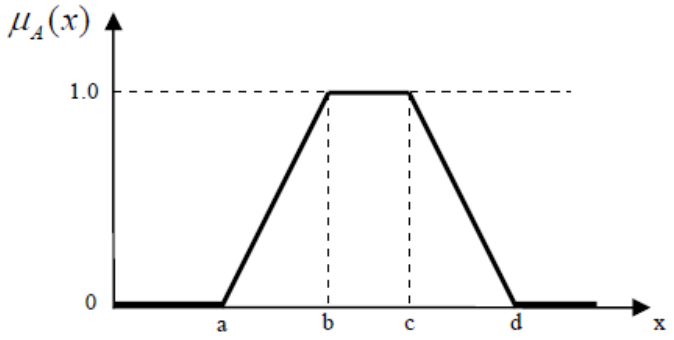
Üçgen üyelik fonksiyonu lineer segmentlerle (1.2)'de gösterildiği gibi ifade edilmektedir (Pedrycz vd., 2011: 26):

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \quad (1.2)$$

Burada a ve c parametreleri değişim aralığını b ise bulanık kümenin üyelik derecesinin 1'e eşit olduğu kısmını göstermektedir (Ersoylu, 2011: 39).

1.3.2. Yamuk Bulanık Sayılar

Üyelik fonksiyonu a, b, c ve d gibi dört değerle aşağıdaki gibi gösterilir (Şen, 2009: 361).



Şekil 1.4 Yamuk Üyelik Fonksiyonu

Yamuk üyelik fonksiyonu lineer segmentlerle (1.3)'de gösterildiği gibi ifade edilmektedir (Pedrycz vd., 2011: 27):

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases} \tag{1.3}$$

Burada a ve d parametreleri değişim aralığını c ve b ise bulanık kümenin üyelik derecesinin 1'e eşit olduğu kısmı göstermektedir (Ersoylu, 2011: 39).

Bulanık küme teorisi, diğer modern tekniklerle birleştirilerek, geleneksel istatistik modellerine kıyasla daha iyi performans elde edilmiştir. Bu modeller, nöro-bulanık modeller, bulanık destek vektör makinesi, bulanık genetik algoritma ve bulanık kural tabanlı k-en yakın komşular algoritması ve bulanık kümelenme şeklinde sıralanabilir (Sohn vd., 2016). Örneğin uzman sistemler, bilgilerin net mantıksal değerler kullanılarak tanımlandığı durumda oldukça başarılı bir tekniktir. Ancak çoğu zaman uzman kararları siyah ve beyaz değildir. Bir uzman hekim genellikle hastanın durumu ile ilgili kesinliğe sahip bir teşhisi yerine kanıtların ağırlığına dayanan bir teşhis koymaktadır. Uzman sistemlerde bulanık mantık uygulayarak kesin olmayan uzman bilgisinden yararlanılmasını sağlamaktadır. Bulanık uzman sistemi, soruna uygun bir dizi dilsel değişken seçilerek ve bu

değişkenler için üyelik fonksiyonlarını tanımlayarak oluşturulmaktadır (Coppin, 2004: 522).

Bulanık mantık sistemlerinin tercih edilmesinin iki nedeni vardır: bulanık sistemler yaklaşık muhakemeye ve kesin olmayan bilgileri kullanarak tahmini değerlerle karar verilmesine izin vermektedir. Belirsizlik söz konusu olduğunda bulanık öğrenmeye dayalı eğitim algoritması da geleneksel öğrenmeye göre çok daha etkilidir. Öğrenme parametrelerini ayarlamak için bulanık mantık kullanan bulanık öğrenme yöntemi kayıp fonksiyonun yerel minimum düzeyinden yani eyer noktasından çıkamama problemini önlemekte, böylece yakınsama hızını artırmakta ve hatayı en aza indirgemektedir (El Hatri ve Boumhidi, 2018).

Modellerin başarısının ölçülebilmesi için çıkan sonuçlara durulaştırma işlemi yapılması gerekmektedir. Durulaştırma işlemi üyelik fonksiyon kümesini muhtemelen en çok temsil edebilecek değere dönüştürme işlemidir. Literatürde ağırlık merkezi, ağırlıklı ortalama, toplamların merkezi, üyelik fonksiyonunun en yüksek noktası, en büyük alan merkezi yöntemi, üyelik fonksiyonunun en yüksek noktalarının ortalaması, ilk (ya da son) yükselti yöntemi gibi birçok durulaştırma yöntemi mevcuttur (Ross, 2010: 90).

Simetrik üçgen bulanık sayılar için yaygın olarak kullanılan ağırlık merkezi yönteminin matematiksel ifadesi (1.4)'te verilmiştir (Wang, 2009).

$$A(\tilde{x}_i) = \frac{1}{3}(a_i + b_i + c_i) \qquad (1.4)$$

1.4. Makine Öğrenmesi

Makine öğrenmesi, sunulan veriden kendi kendine öğrenebilen ve tahmin yapabilen algoritmalar olup, yapay zekanın alt dalı olarak gelişmiştir. İnsanlar tarafından büyük miktarda veriyi analiz ederek kuralları el ile oluşturmak yerine makine öğrenmesi, verideki bilgileri yakalayarak tahmin etme performansını aşamalı olarak geliştirmekte ve verilere dayalı kararlar verebilen daha etkili çözüm sunmaktadır. Makine öğrenimi bilgisayar bilimleri araştırmalarında giderek daha önemli konum edinirken, günlük yaşamımızda da büyük bir rol oynamaktadır. Makine öğrenimi ile e-posta spam filtrelerinin, uygun metin ve ses tanıma yazılımının, güvenilir web arama motorları gibi birçok uygulama geliştirilmiştir (Raschka ve Mirjalili, 2017: 2).

Makine Öğrenmesi veriden anlamlı ilişkiler çıkarmak için denetimli öğrenme, denetimsiz öğrenme, takviyeli öğrenme ve hibrit öğrenme gibi öğrenme kuralları kullanmaktadır (Fyfe, 2000).

1.4.1. Denetimli Öğrenme

Denetimli öğrenmede modele giriş verisinin yanında elde edilmek istenilen çıktı da verilmektedir. Giriş verisi çıkışa ulaşana kadar hep ileri yayılmaktadır. Çıkış biriminde elde edilen değerin istenilen değerle uyum sağlayıp sağlamadığı karşılaştırılmaktadır. Çıkış birimi ile istenilen veri arasındaki farkın kabul edilme sınırları içerisinde olmaması durumunda ağırlıklarda iyileştirme yapılmaktadır. Denetimli öğrenme aynı zamanda öğretmenli öğrenme olarak da adlandırılmaktadır (Fyfe, 2000).

Denetimli öğrenme, sadece girdilerin ve çıktıların önemli olduğu bir öğrenme algoritmasıdır. Dağılımın şeklinin tespiti, girdi ile çıktı arasındaki bağlantı şeklinin tespiti gibi orta kademe hedeflerin yer almadığı bir algoritmadır. Hatta bazen başlangıç ağırlıkların belirlenmesi bu öğrenme tekniğinde gereksiz görülmektedir, çünkü öğrenme sürecinde algoritma öğrenme performansına göre ağırlıklarda iyileştirmeler yapmaktadır. Bu nedenle denetimli veya ayrıştırıcı öğrenme, denetimsiz öğrenme gibi orta kademe hedefler için kaynakları harcamamaktadır (Jebara, 2012: 48).

Denetimli öğrenme yaklaşımını uygulayan en basit model regresyon modelleridir. Örneğin regresyon modelinde konum, oda sayısı, tesisler bir dizi parametre verilerek bir dairenin fiyatı tahmin etmesi sağlanmaktadır. Modele ilk önce parametreler fiyat bilgisi ile beraber verilerek, regresyon modelinin katsayıları hesaplanmaktadır. Daha sonra bu katsayılar ve modelin daha önce görmediği bir örneğin parametreleri kullanılarak fiyat tahmini yapılmaktadır. Literatürde yer alan en önemli denetimli öğrenme modelleri aşağıdaki verilmiştir (Russell, 2018: 14).

- Doğrusal regresyon
- Lojistik regresyon
- Yapay sinir ağlar
- K-en yakın komşular
- Vektör destek makineleri
- Karar ağaçları ve rastgele ormanlar

1.4.2. Takviyeli Öğrenme

Takviyeli öğrenmede giriş verisi modele ileri yayılım algoritması olarak verilmekte ve üretilen çıktının doğru olup olmadığı modele geri bildirim yapılmaktadır. Eğer çıktının verisi istenilen veriden farklı ise ağırlıklarda iyileştirme yapılmaktadır (Fyfe, 2000).

Eğitim psikolojisinde yer alan "Takviyeli Öğrenme" yaklaşımından esinlenerek geliştirilen bu makine öğrenmesi yaklaşımında amaç, çevre ile etkileşime dayalı olarak performansını artıran bir sistem geliştirmektir. Çevrenin mevcut durumu hakkındaki bilgiler sisteme ödül sinyali olarak verildiği öğrenme yaklaşımında deneme yanılma yoluyla bilgilerin keşfedilmesi beklenmektedir. Takviyeli öğreniminin popüler bir örneği bir satranç motorudur. Burada, oyuncu, tahtanın durumuna bağlı olarak bir dizi hamle yapmaya karar verir ve oyunun sonunda kazandığı takdirde ödül elde etmektedir (Raschka ve Mirjalili, 2017: 16).

Takviyeli öğrenme robotik alanında da kullanılmaktadır. Yürümeyi öğrenmeye çalışan robot, deneme yanılma yoluyla çevreyi keşfetmektedir. Olumsuz şartları sensörler yardımıyla algılayarak tecrübe etmesi durumunda bir sonraki hamlesinde bu adımdan kaçınma yolunu seçmektedir (Russell, 2018: 14).

1.4.3. Denetimsiz Öğrenme

Denetimsiz öğrenme, modele sadece giriş verileri sağlayarak gerçekleşmektedir. Model giriş verinin yapısına bağlı olarak kendini yeniden organize etmektedir. Böylece model kendi kendine öğrenme işlemini yapmaktadır. Denetimsiz öğrenme aynı zamanda öğretmensiz öğrenme olarak da adlandırılmaktadır. (Fyfe, 2000).

Denetimli öğrenmede örneklemdeki verileri kullanarak özellikleri bilinen sınıflar arasında kesin bir çizgi belirlemeye çalışmakta ve yeni veriyi hangi sınıfa ait olduğuna dair bir tahmin yapmaktadır. Denetimsiz öğrenmede ise sınıflar belli olmadığından öncelikle veriyi analiz ederek sınıflar oluşturmaya ve bu sınıflara dair özellikleri belirlemeye çalışmaktadır. Daha sonra gelen yeni veri için hangi sınıfa ait olduğuna dair bir tahmin yapmaktadır (Ng, 2017).

Denetimsiz öğrenme tekniğinin kullanıldığı modeller, hedef sınıfları hakkında hiçbir bilgi mevcut olmadığında, analiz amacıyla toplanan veri içerisinde sıralı bilgi keşfi amacıyla tasarlanmıştır. Modeller denetimsiz modda kullanıldığında, veriler içerisinde sınıfları ve sınıflar arasındaki ilişkileri istatistiksel dağılımları verinin bir

parçası haline getirerek veriye ek özellikler sağlamaktadır. Daha sonra elde edilen bu veriye denetimli öğrenme uygulayarak daha başarılı sonuçlar elde etmek mümkündür. Denetimsiz öğrenmenin bu haliyle kullanımı literatüre "ön öğrenme" olarak girmiştir (Deng ve Yu, 2014: 214).

Denetimsiz öğrenme amaç, hiçbir bilgiyi kaybetmeden çıktıyı mümkün olduğunca basit hale getirmektir. Bunu sağlamak için, birkaç özelliği tek parametrede birleştirmektedir. Örneğin bir otomobilin modeli ile verimini bir parametre alması gibi. Bu işlem özellik çıkarma olarak adlandırılmaktadır. Hiyerarşik küme analizi ve k-means gibi kümeleme algoritmaları, birliktelik kuralı öğrenme, görselleştirme ve boyut azaltılma algoritmaları denetimsiz öğrenme algoritmalarıdır. Örneğin görselleştirme algoritmalarında etiketlenmemiş veriler modele girdi olarak verilmekte çıktı olarak 2D veya 3D görselleştirme elde edilmektedir (Russell, 2018: 14).

1.4.4. Hibrit Öğrenme

Hibrit öğrenme hem denetimsiz, hem de denetimli öğrenme teknikleri barındıran tekniktir. Hibrit modellerin asıl hedefi sınıflandırmadır. Bu amacı yerine getirmek için literatürde hibrit mimaride genelde denetimsiz öğrenme denetimli öğrenmeye yardımcı olacak şekilde kullanılmaktadır. Doğrusal olmayan parametre tahmin problemlerinde denetimsiz öğrenme mükemmel başlatma noktaları tespit edererek modeli optimize etmektedir. Bu işlem literatürde ön eğitim olarak tanımlanmaktadır. Bunun yanında denetimsiz öğrenme genel modelin karmaşıklığını etkili bir şekilde kontrol ederek düzenleme görevini yerine getirmektedir (Deng, 2014).

1.5. Sınıflandırma Algoritmaları

Genel olarak, olasılıksal denetimsiz modeller için tahmin ve öğrenme süreci belirli bir görevi yerine getirmek için tasarlanmamıştır. Bu modellerde sınıflandırma, tahmin veya regresyon için maksimum olabilirlik gibi genel optimizasyon kriterleri kullanılmaktadır. Destek vektör makineleri gibi ayırıcı veya denetimli öğrenme modellerinde doğrudan ilgili göreve yönelik optimizasyon gerçekleştirildiğinden daha güçlü modeller olup birçok veri setinde daha iyi performans göstermektedir (Jebara, 2012: 12).

1.5.1. Vektör Destek Makineleri

Makine öğrenmesi algoritmalarını tasarlamak için otomatik bilgi işlem temellerinin kavranması gerekmektedir. "İkili sınıflandırma" otomatik bilgi işlemenin yer aldığı en temel sınıflandırmada problemidir. İkili sınıflandırmada sınıfları ayırma kriteri sınıflar arasındaki mesafe vektörüdür. Yani, aynı sınıflardaki nesnelerin mesafesi en aza indirilmeye çalışılırken, farklı sınıflardaki nesnelerin mesafesi çok yüksek düzeye çıkarılmaktadır. Düşük boyutlu uzayda doğrusal olmayan problemlerde bu mesafenin ölçülmesi mümkün olmayacağı için probleme doğrusal olmayan dönüşümler uygulayıp problem daha yüksek boyutlu bir uzaya taşınması gerekmektedir. Böylece problem doğrusal olarak ayrılabilir bir probleme dönüştürdüğü için mesafe vektörlerinin ölçülmesi mümkün hale gelmektedir. Bu problemlerde sınıflandırma işlevleri destek vektörlerine bağlıdır, bu nedenle modele "destek vektör makinesi" olarak adlandırılmaktadır (Li ve Du, 2008: 97).

1.5.2. Karar Ağaçları

Karar ağaçları, basit fakat güçlü çok değişkenli analiz biçimidir. Karar ağaçları, geleneksel istatistiksel analiz türlerini (çoklu doğrusal regresyon gibi), çeşitli veri madenciliği araçlarını (Yapay sinir ağları gibi), iş zekası alanında son zamanlarda geliştirilen çok boyutlu raporlama ve analiz biçimleri tamamlamak, değiştirmek ve yerine koymak üzere benzersiz yetenekler sağlamaktadır (Jensen, 2008).

Karar ağaçları, öğrenilen bilginin bir ağaç şeklinde gösterildiği, ayrık değerli hedef fonksiyonlara yaklaşma metodudur. Karar ağaçlarına gösterilmiş öğrenilen bilgi daha anlaşılır olması için "eğer" içerikli cümlelerle de ifade edilmektedir. Karar ağaçları kredi riski değerlendirme, tıbbi vakayı teşhis etme gibi birçok durumda başarıyla kullanılmaktadır. Karar ağaçları kökten itibaren başlayarak, verileri düğümlere, dallara ve yapraklara ayırarak sınıflandırmaktadır. Her düğüm bir kritere, her dal ise bir kriterin değerine karşılık gelmektedir. Karar ağaçları oluşturmak için birçok geliştirilmiş algoritma bulunmaktadır (Mitchell, 1997).

Karar ağacı oluşturmak için CHAID (Chi- Squared Automatic Interaction Detector), Exhaustive CHAID, CRT (Classification and Regression Trees), ID3, C4.5, MARS (Multivariate Adaptive Regression Splines), QUEST (Quick, Unbiased, Efficient Statistical Tree), C5.0, SLIQ (Supervised Learning in Quest), SPRINT

(Scalable Paralleizable Induction of Decision Trees) geliştirilen bu algoritmalar arasındadır (Albayrak ve Koltan Yılmaz, 2009).

1.5.3. Lojistik Regresyon

Lojistik regresyon, bağımlı değişkeninin ikili değişkenlerle ölçüldüğü problemleri analiz etmek için istatistiksel bir yöntemdir. Lojistik regresyonun amacı, sağlık sektörü için örneklendiğinde kişinin bir hastalığı olup olmadığını veya hastanın memnun olup olmadığını belirleyen faktörlerin arasındaki ilişkiyi tanımlamaktır (Salmani vd., 2017).

Lojistik regresyondan logit modeli (1.5)'te verilmiştir. Burada y bağımlı değişken, x_i bağımsız değişkenler, a sabit sayı, b_i bağımsız değişkenlerin katsayıları ve e hata terimidir (Rusli vd., 2008).

$$logit(y) = a + b_1x_1 + b_2x_2 + b_3x_3 + \ldots + e \qquad (1.5)$$

Bağımsız değişkeninin logit değeri olasılık oranlarının doğal logaritmasıdır. Matematiksel ifadesi (1.6)'da verilmiştir. Burada p olayın gerçekleşme olasılığı olup $(1-p)$ ise olayın gerçekleşmeme olasılığıdır.

$$logit(y) = ln[p/(1-p)] \qquad (1.6)$$

Lojisitk regresyonunun performansı diğer tekniklerde olduğu gibi Bulanık Mantık ile entegre edilerek arttırılabilir. Lojisitk regresyonunun faktörlerin bazen belirsizlik içermesi ya da bağımlı değişkenin "evet-hayır" durumu yerine "çok düşük, düşük, ortalama, yüksek, çok yüksek" gibi dilsel terimlerle ifade edilebilecek durumlar içermesi bağımlı değişkenin analizi için olasılık dağılımı anlamsız olacaktır. Analiz edilen veri kümesi yeterince büyükse ve temel varsayımlar karşılanıyorsa sıralı lojistik regresyon veri setini analiz etmede kullanılabilir. Ancak eğer veri seti gerçek sayı ile ifade edilemeyen belirsiz veri içeriyorsa, bulanık lojistik regresyon alternatif bir seçenek olabilir (Namdari vd., 2015).

Bunun yanında bağımsız değişkenlerin de dilsel ölçeklerle tanımlanmadığı durumlar da olabilir. Klinik araştırmalarda bağımsız değişken olan baş ağrısının ne kadar kötü olduğu tanımlanması durumu örnek verilebilir (Gao ve Lu, 2018).

Genellikle, tıbbi araştırmalarda birçok yanıt dilsel değişkenlerle ölçüldüğünden dilsel değişkenlerin kategorileri arasındaki sınırın kesin olmaması analizi zorlaştırmaktadır. Bu gibi durumlarda, önemli faktörlerin bağımlı değişken üzerindeki etkisini belirlemek için bulanık lojistik regresyon esnek ve uygun bir araçtır (Namdari vd., 2014).

Bulanık Lojistik regresyonun matematiksel ifadesi (1.7)'de verilmiştir. Burada x_i bağımsız tam sayılı parametreler yerine bulanık üçgen sayıları ifade eden $\tilde{x}_i$ parametresi ve toplama işlemi yerine bulanık toplama işlemi kullanılmaktadır (Sohn vd., 2016).

$$logit(\tilde{y}) = a \oplus b_1\tilde{x}_1 \oplus b_2\tilde{x}_2 \oplus b_3\tilde{x}_3 \oplus \ldots \oplus e \qquad (1.7)$$

Bulanık $\tilde{x}_i$ parametresi üçgensel bulanık sayı ifadesi için (1.8)'de verilmiştir. Üçgensel bulanık sayı için a_i sol sınırı değerini, b_i merkez değerini, c_i sağ sınır değerini temsil etmektedir (Sohn vd., 2016).

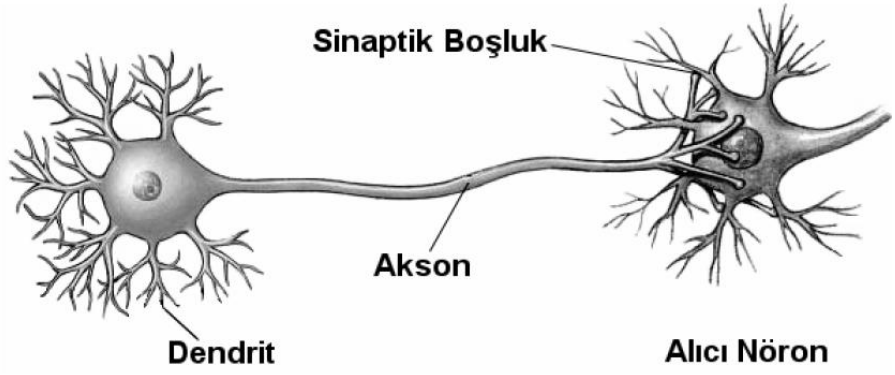
$$\tilde{x}_i = (a_i, b_i, c_i) \qquad (1.8)$$

1.5.4. Yapay Sinir Ağları

İsminden de anlaşıldığı gibi yapay sinir ağları, biyolojik (insan ya da hayvan), merkezi sinir sisteminde yer alan sinir hücrelerinin (nöronların) oluşturduğu ağları taklit eden bilişimsel ağlardır. Yapay sinir ağları tasarlanırken biyolojik nöron ağlarının nörofizyolojik bilgisinden faydalanılmıştır (Graupe, 2007: 1).

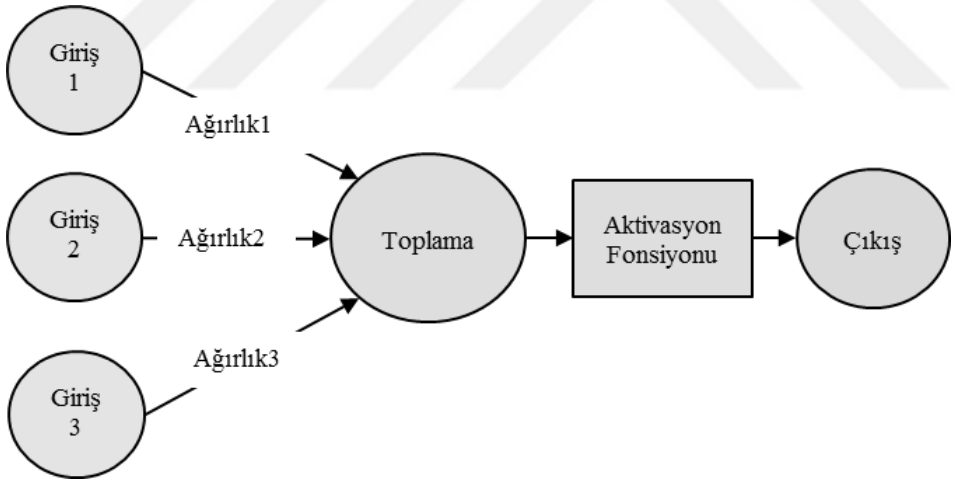
Bir yapay sinir ağı, çok sayıda bağlantı oluşturup birbiri ile sinyal alışverişinde bulunarak iletişim kuran basit işleme birimlerinden oluşmaktadır. Bir yapay sinir ağının çalışma prensibinin açıklanabilmesi için öncelikle nöronlar, birimler arasındaki bağlantı ağırlıkları, yayılma kuralı, toplama fonksiyonu, aktivasyon fonksiyonu, giriş birimi, çıkış birimi, öğrenme kuralı gibi kavramların açıklanması gerekmektedir (Kröse ve van der Smagt, 1996: 15).

Yapay sinir ağları tasarlanırken biyolojik nöron ağlarının nörofizyolojik bilgisinden faydalanarak biyolojik sinir hücresinin yapısı taklit edilmeye çalışılmıştır (Graupe, 2007: 1). Biyolojik sinir ağının temel birimi olan sinir hücrelerinin (nöron) birbirlerini uyarması sonucunda elektriksel sinyaller sinir ağı boyunca taşınmaktadır. Nöronların bilgi giriş noktaları olan dendritlere gelen işaret yeteri kadar güçlüyse, nöron bir çıkış işareti üretmekte ve üretilen bu çıkış işareti bir sonraki nöronun dendritine, aksonlar vasıtasıyla iletilmektedir. Sinyal, verici nöronun aksonundan, alıcı nöronun dendritlerine sinaptik boşluklar aracılığı ile iletilmektedir. Sinaptik boşluklara dolan nöro-transmitter sıvısı ise iletilen işaretin güçlendirilmesini veya zayıflatılmasını sağlamaktadır. Şekil 1.5'te biyolojik bir nöronun şematik gösterimi verilmiştir (Öğücü, 2006).



Şekil 1.5 Biyolojik Sinir Hücresinin Şematik Gösterimi (Öğücü, 2006)

Biyolojik sinir ağlarına benzer bir şekilde yapay sinir ağları da yapay nöronlardan oluşmaktadır. Yapay sinir ağları farklı çeşit ve mimarilerde olduğundan ve sürekli yeni çeşit sinir ağları denenmeye çalışıldığından hepsini kapsayacak yapay nöron modelini açıklamak mümkün olmamaktadır. Ancak, sinir ağı uygulamaları arasında bazı ortak noktalar bulunmaktadır. Yapay nöron, bir ya da birden fazla kaynaktan girdi aldıktan sonra ağırlıklarla çarpmakta ve toplanmaktadır. Elde edilen toplam, çıkışa iletilmek üzere aktivasyon fonksiyonuna aktarılmaktadır. Şekil 1.6'da yapay bir nöronun şematik gösterimi verilmiştir (Heaton, 2015: 31)



Şekil 1.6 Yapay Sinir Hücresinin Şematik Gösterimi (Heaton, 2015: 31)

Girişler, giriş katmanındaki hücreler için veri kaynağı iken diğer katmandaki hücreler için herhangi bir katmandaki hücrenin çıkışıdır. Girişler matematiksel ifade olarak $x_1, x_2, x_3, \ldots x_n$ şeklinde gösterilmektedir. (Öğücü, 2006).

Ağırlıklar, girişler aracılığı ile gelen verinin çıkışa ne kadar güçlü iletileceğini göstermektedir. Ağırlıklar matematiksel ifade olarak $w_1, w_2, w_3, \ldots w_n$ şeklinde gösterilmektedir (Freeman ve Skapura, 1991: 18).

Toplama fonksiyonu, her bir girdi değeri ile ilgili ağırlıkları ilişkilendirerek net girdi üretmektedir. En yaygın olarak kullanılan toplama fonksiyonu, her gelen girdinin kendi ağırlığı ile çarpılarak toplam elde edilme şeklindendir. Toplama fonksiyonun matematiksel şekli (1.9) nolu ifadede verilmiştir (Emir, 2013).

$$\text{net girdi} = \sum w_i x_i = w_1 x_1 + w_2 x_2 + \cdots + w_i x_i \qquad (1.9)$$

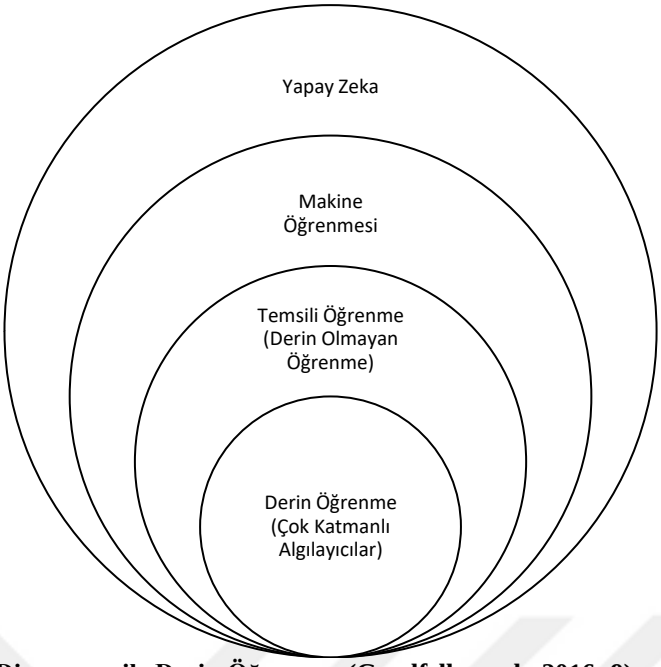
Aktivasyon fonksiyonu, yapay sinir hücresinin çıkışı için sınırlar belirlemektedir. Sinir ağlarında birçok farklı aktivasyon fonksiyonu kullanılmaktadır. En yaygın kullanılan aktivasyon fonksiyonları; lineer aktivasyon fonksiyonu, eşik değer aktivasyon fonksiyonu, sigmoid aktivasyon fonksiyonu, hiperbolik tanjant aktivasyon fonksiyonu, softmax aktivasyon fonksiyonu olarak sıralamak mümkündür. Aktivasyon fonksiyonunun seçimi çıkışa iletilen veriyi etkilediği için dikkatle ele alınması gereken bir husustur. Örneğin sadece pozitif çıkış elde edilmesi gerekiyorsa sigmoid fonksiyonu, hem pozitif hem de negatif çıkış elde edilmesi gerekiyorsa hiperbolik tanjant fonksiyonu seçilmesi gerekmektedir (Heaton, 2015: 48).

Çıkış, aktivasyon fonksiyonu tarafından belirlenen, başka bir hücreye yada dış dünyaya gönderilen bir değerdir. Geribeslemeli ağlarda çıkış değeri aynı zamanda girişe iletilerek geribesleme yapılmaktadır (Öğücü, 2006).

Yapay sinir ağlarında iki farklı yayılma kuralı kullanılmaktadır. İleri beslemeli ağlarda verilerin girdi birimlerinden çıktı birimlerine sadece ileri doğru yayıldığı bir kural tanımlanmaktadır. Geri beslemeli ağlarda ise veri akışının sadece ileriye doğru değil geriye doğru da olabileceğine dair bir kural tanımlanmaktadır. Bu ağ yapısında, ağ çıktısı aynı zamanda girdi olarak da kullanılabilmektedir (Hamzaçebi, 2011: 46).

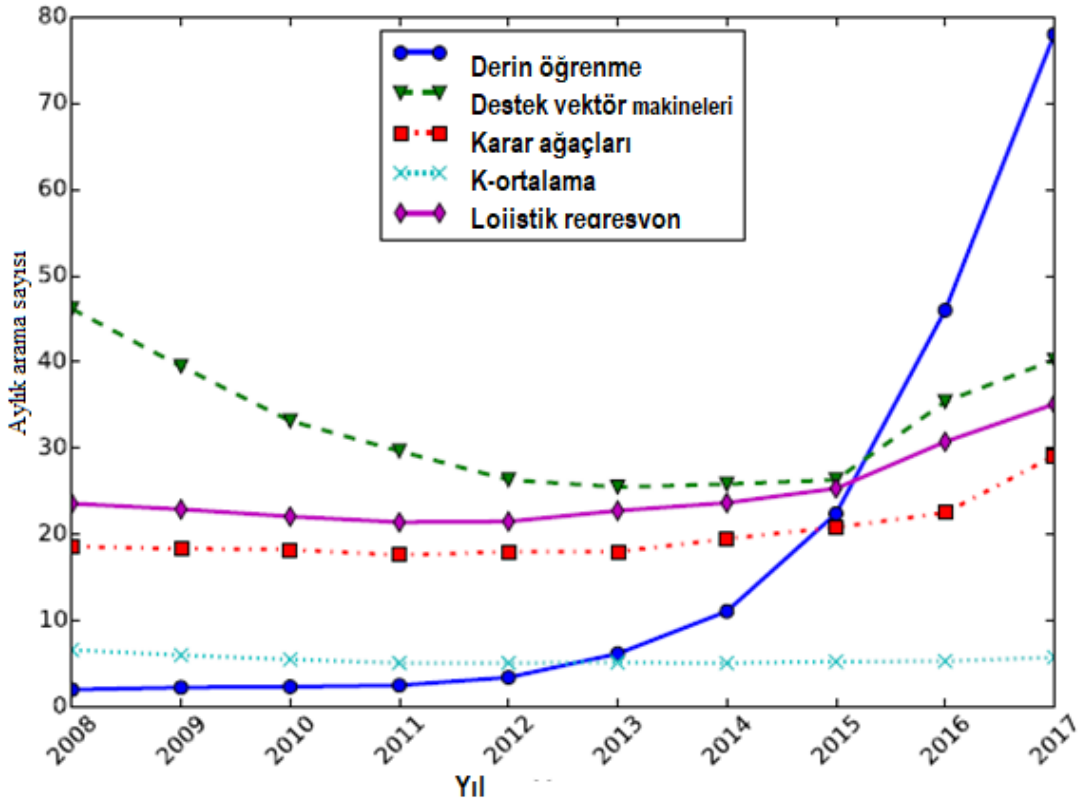
İnternetin gelişmesiyle beraber veriye ulaşmanın kolay hale gelmiş ve yapay sinir ağlarının daha büyük veri kümeleriyle eğitilmesi mümkün olmuştur. Veri miktarının artmasıyla beraber yapay sinir ağlarının öğrenme performansını arttıracak birçok yaklaşım geliştirilmiştir. Bu yaklaşımlardan biri de derin öğrenme yaklaşımıdır. Birden fazla gizli katman içeren yapay sinir ağı derin ağ olarak tanımlanmakta olup, sergilediği öğrenme şekli ise derin öğrenme olarak adlandırılmaktadır (Deng ve Yu, 2014: 206).

Şekil 1.7'de derin öğrenme tekniğinin diğer yapay zeka teknikleri ile arasındaki ilişki görselleştirilmeye çalışılmıştır (Goodfellow vd., 2016: 9).



Şekil 1.7 Venn Diyagramı ile Derin Öğrenme (Goodfellow vd., 2016: 9)

Şekil 1.8'de verilen Google arama eğilimi grafiği son yıllarda derin öğrenme mimarilerinin diğer geleneksel makine öğrenim yaklaşımlarına göre daha fazla ilgi gördüğünü ve daha popüler hale geldiğini göstermektedir (Mohammadi vd., 2018).



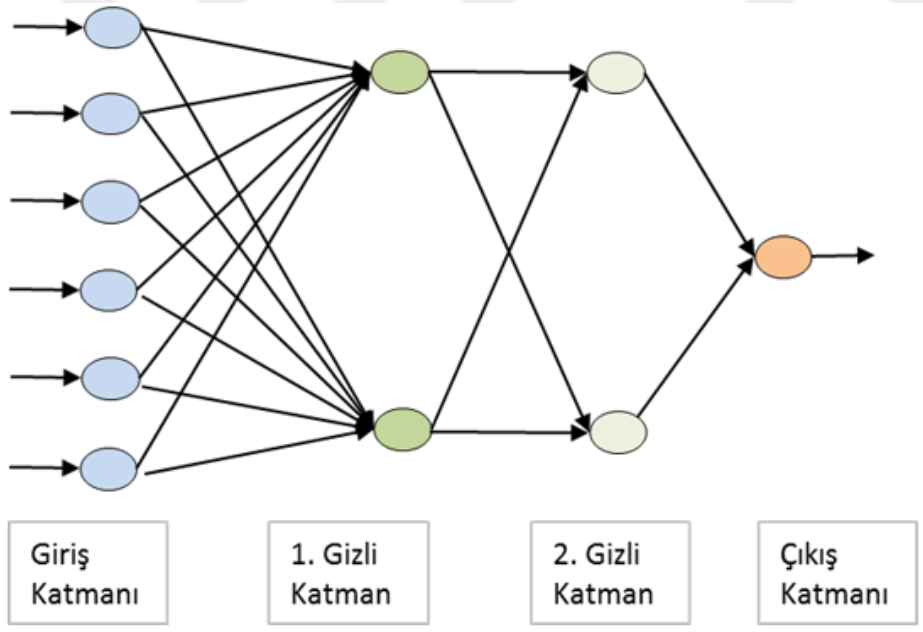
Şekil 1.8 Google Arama Sayıları (Mohammadi vd., 2018).

İKİNCİ BÖLÜM
DERİN ÖĞRENME

Derin öğrenme, bilgisayar sistemlerinin veride saklı olan deneyimden maksimum şekilde yararlanmaya imkan tanıyan, gerçek dünyanın karmaşık verileri ile çalışabilen, bu karmaşık verileri içi içe geçmiş hiyerarşiler ile ele alabilen ve bu hiyerarşileri basit ilişkilerle tanımlayarak daha başarılı sonuçlara elde eden makine öğrenmesi tekniğidir. Bir başka tanıma göre; derin öğrenme, yapay sinir ağları, yapay zeka, grafiksel modelleme, optimizasyon, model tanıma ve sinyal işlemenin bir araya gelmesiyle meydana gelen daha güçlü tahminler yapmak için kullanılabilecek makine öğrenimi tekniğidir (Lewis, 2016: 7).

2.1. Derin Ağlar

Çok katmanlı makine öğrenmesi modellerini kullanan derin öğrenme tekniğinde her katmadaki veriye doğrusal olmayan dönüşümler uygulanıp daha üst, daha soyut katmana iletilerek gözetimli veya gözetimsiz öğrenme gerçekleşmektedir. Derin öğrenmenin en basit hali Şekil 2.1'de verilmiştir (Lewis, 2016: 9).

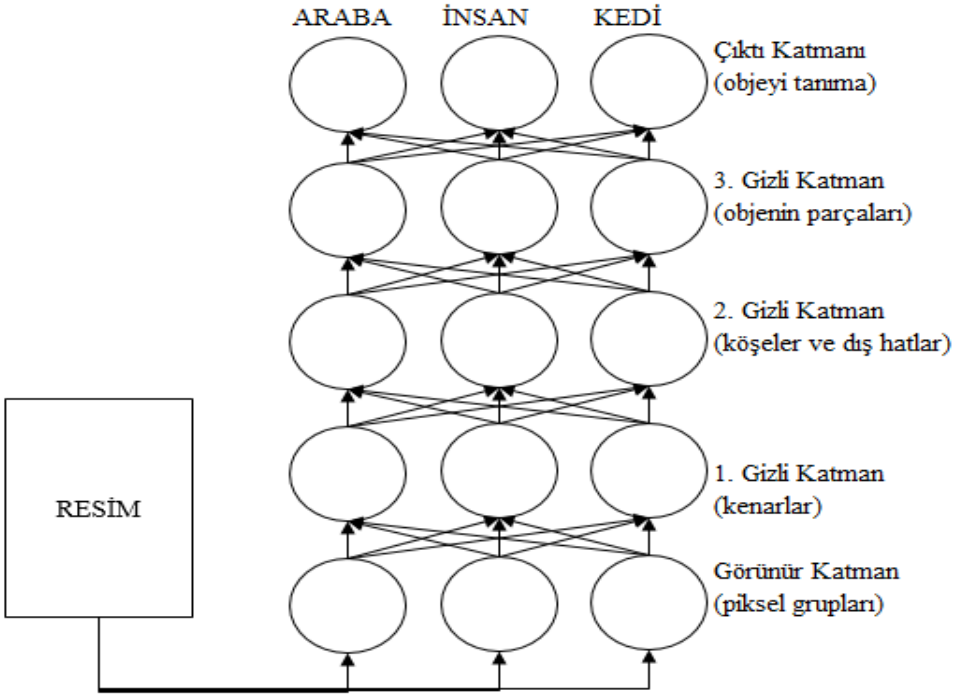


Şekil 2.1 İki Gizli Katmanlı Olan Derin Öğrenme Şeması (Lewis, 2016: 10)

Şekil 2.1'te görülen derin öğrenme modelinde ilk katman giriş katmanıdır. Bu katman bazı kaynaklarda "Görünür Katman" olarak da adlandırılmaktadır. Bu katmanın "Görünür" olarak adlandırılmasının sebebi gözlemlenebilir değişken ve

veriden oluşuyor olmasıdır. Örneğin resimdeki pikseller bu katmanda haritalanmakta ve resimden daha soyut özelliklerin elde edilmesi için gizli katmanlara iletilmektedir. Sonraki katmanların "Gizli Katman" olarak adlandırılmasının sebebi verideki değerler yerine modeli belirlemek için gerekli olan veri içindeki ilişkileri açıklamaya yarayan kavramlar içermeleridir (Goodfellow vd., 2016: 6).

Yine resim örneği verilecek olursa ilk gizli katmandaki gizli öğelerin her biri temsil edilen türün özelliklerini görselleştirmeye çalışmaktadır. Bu katmanda komşu piksellerin parlaklıkları karşılaştırılarak cismin kenarlarının elde edilmesi mümkündür. İlk gizli katmanda elde edilen kenarlar göz önünde tutularak ikinci gizli katmanda köşeler ve uzatılmış dış hatlar elde edilmektedir. İlk iki katmandan gelen bilgiler göz önünde tutularak cismin bölümleri elde edilmektedir. Son katmanda ise gizli katmanlarda elde edilen veriler bir bütün haline getirilerek resmi anlamlandırmada kullanılmaktadır. Görüntünün derin öğrenme tekniği ile sınıflanmasına örnek Şekil 2.2'de verilmiştir (Goodfellow vd., 2016: 6).



Şekil 2.2 Görüntü Tanımada Derin Öğrenme (Goodfellow vd., 2016: 6)

Bir makine öğrenim tekniğinin iyi performans elde edebilmesi büyük ölçüde girdi verilerinin iyi temsil edilmesine bağlıdır. Dolayısıyla, verilerin ön-işlenmesi öğrenen sistemlerin oluşturulması sürecinde kritik bir adımdır. Öznitelik çıkarımı sürecinde uzmanlardan yararlanarak girdi verilerinin özelliklerinin boyutunu azaltma çalışması yapılmaktadır. Destek vektör makineleri (SVM'ler) ve lojistik regresyon

gibi sığ öğrenme modellerinin performansı bu öznitelik çıkarma çalışmasına bağlıdır. Bu süreç önemli ama aynı zamanda çok zaman alıcı ve yorucudur. Bu sorunu işi kolaylaştıran algoritmalarla çözmek daha pratik bir yoldur. Derin öğrenme teknikleri, yüksek boyutlu verilerle başa çıkmak ve verilerden ayırt edici bilgi elde etmek için kullanılan en iyi çözümlerden biridir. Derin öğrenme algoritmaları, uzman bilgisine ihtiyaç duymadan öznitelik çıkarımını otomatik hale getirme yeteneğine sahiptir (Mousavi vd., 2016).

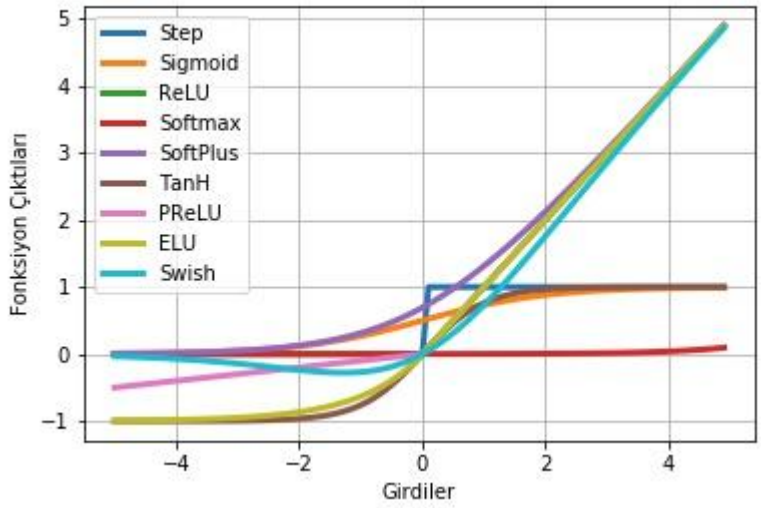
Derin öğrenme, bir tür makine öğrenimi olduğundan derin ağların gelişiminde geleneksel makine öğrenmenin temel prensipleri etkili olmuştur (Goodfellow vd, 2016: 98). Fakat deneysel çalışmalar derin mimarilerin sığ mimarilere göre eğitilmesinin daha zor olduğunu göstermiştir. Örneğin meyilli azalma algoritması kullanılan derin öğrenme ağında mimari derinleştikçe "yerel minimum" sorunu daha belirgin hal almaktadır. Derin mimarilerde bu tür problemleri aşmak için literatürde her geçen gün yeni çözümler önerilmektedir. Tüm bu çözümleri daha iyi kavramak için makine öğrenmenin temel prensiplerinin çok iyi kavranması gerekmektedir (Bengio, 2009: 34).

Sinir ağının öğrenme sürecinde ağın istenen çıktısının y olduğu ve ağın çıkış ($\hat{y}$) ürettiği varsayılırsa, öngörülen çıktı ile istenen çıktı ($\hat{y}$ - y) arasındaki fark, kayıp fonksiyonu olarak bilinen bir metriktir. Nöral ağ çok fazla hata yaptığında kayıp yüksek, daha az hata yaptığında ise düşük olmaktadır. Eğitim sürecinin amacı, eğitim setindeki kayıp fonksiyonun en aza indiren ağırlık ve yanlılıkları bulmaktır (Sharma, 2017).

Yapay sinir ağ modelinin verimli ve etkili bir şekilde eğitilmesinde aktivasyon fonksiyonu, kayıp fonksiyonu ve optimizasyon algoritması gibi bileşenler doğru sonuçlar üretmede çok önemli bir rol oynamaktadır (Agrawal, 2017).

2.2. Aktivasyon Fonksiyonları

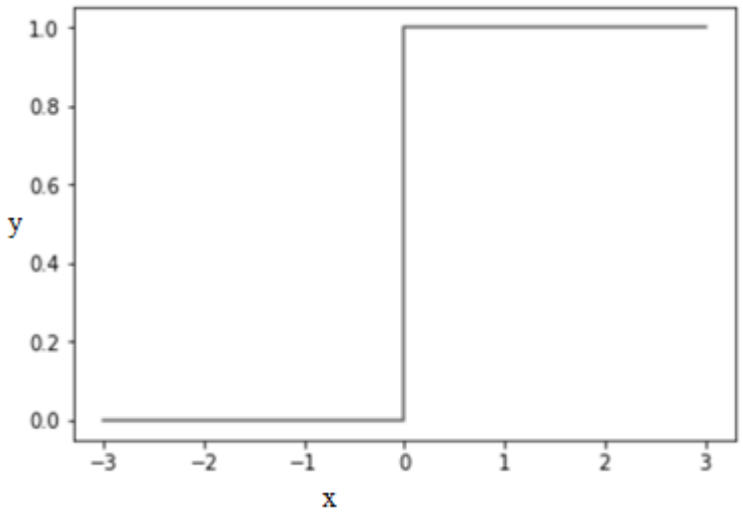
Biyoloji sinir hücrelerinde dendritlerden gelen sinyaller, hücre gövdesinde biriktirilir ve sonuçta ortaya çıkan sinyalin kuvveti belirli bir eşiğin üzerindeyse çıkışa yani aksona iletilir, aksi halde iletilmez. Yapay sinir hücresinde de buna benzer olarak bu görevi aktivasyon fonksiyonu yerine getirmektedir. Aktivasyon fonksiyonu sinyali iletip iletmeme kararı vermektedir (Sharma, 2017). Literatürde yaygın kullanılan aktivasyon fonksiyonları Şekil 2.3'te verilmiştir.



Şekil 2.3 Yaygın Kullanılan Aktivasyon Fonksiyonları (Karakuş, 2018)

2.2.1. Adım Aktivasyon Fonksiyonu

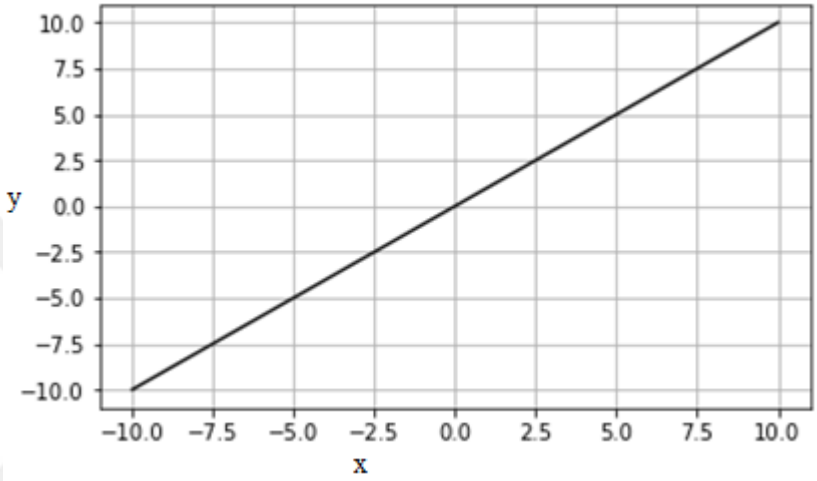
Tek nöron modeli için yaygın olarak kullanılan basit bir adım fonksiyonu olan adım aktivasyon fonksiyonu eşik değerine göre 1 veya 0 değerleri üretmektedir. Eğer girdi 0'da küçükse 0, diğer tüm durumlarda 1 değerini vermektedir. Şekil 2.4'ten de görüldüğü gibi girdi 0 değerini aldığı için fonksiyon ayırt ediciliğini kaybetmektedir. Ayrıca sadece 0 veya 1 değerini ürettiğinden ağırlıklarının güncellemede kullanılması mümkün olmamakta ve bu nedenle derin öğrenme algoritmalarında kullanılmamaktadır (Mehrotra vd., 1996).



Şekil 2.4 Adım Aktivasyon Fonskiyonu

2.2.2. Doğrusal Aktivasyon Fonksiyonu

Lineer ya da doğrusal aktivasyon fonksiyonu f (x) = Wx ya da $\hat{y} = W^T x + b$ şeklinde tanımlanmaktadır. Burada $\hat{y}$ ağın ürettiği çıkış, W^T ağırlık vektörü, x bağımsız değişkenler ya da ağ girişleri, b ise yanlılığı ifade etmektedir. Şekil 2.5'ten de görüldüğü gibi bağımlı değişken bağımsız değişkenle doğru orantılı olarak değişmektedir. Sığ mimarilerde etkin kullanımı görülen doğrusal aktivasyon fonksiyonunun, derin öğrenme problemlerinde kullanımı sınırlıdır (Patterson ve Gibson, 2017: 66).



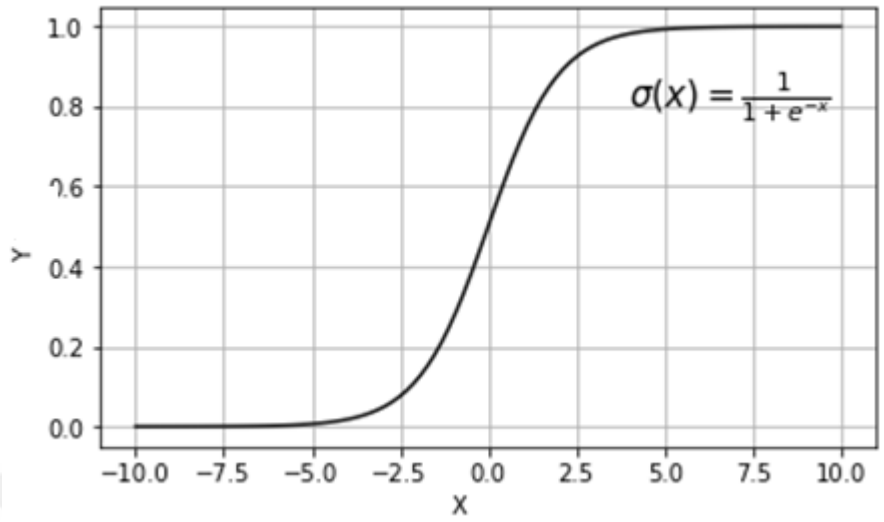
Şekil 2.5 Doğrusal Aktivasyon Fonksiyonu (Sharma, 2017)

2.2.3. Lojistik Aktivasyon Fonksiyonu

Lojistik aktivasyon fonksiyonu olarak da bilinen sigmoid aktivasyon fonksiyonu sonsuz aralığın bağımsız değişkenlerini 0 ile 1 arasındaki basit olasılıklara dönüştürmektedir (Patterson ve Gibson, 2017: 67). Lojistik aktivasyon fonksiyonu $\hat{y} = \sigma(W^T x + b)$ olarak ifade edilmektedir. Burada σ sigmoid fonksiyonunu ifade etmektedir. Sigmoid çıkış ünitesini iki bileşenli olarak düşünebiliriz. İlk olarak, $z = W^T x + b$ şeklinde doğrusal fonksiyon gibi hesaplanır. Daha sonra z değerini olasılık değerine dönüştürmek için sigmoid aktivasyonu kullanılmaktadır (Goodfellow vd., 2016: 183).

Yapay sinir ağlarında etkili bir şekilde kullanılmasına rağmen derin ağlarda çok fazla tercih edilmemektedir, çünkü çıkış değerleri 0 ve 1 değerlerine yaklaştıkça ağırlıkların güncelleme miktarları azalacağı için öğrenme süreci yavaşlamaktadır. Bu sorun literatürde kaybolan eğim sorunu olarak yer almaktadır. Ayrıca çıkışların sıfır merkezli olmaması, yani üretilen çıkışların ortalamadan farklarının toplamının sıfır

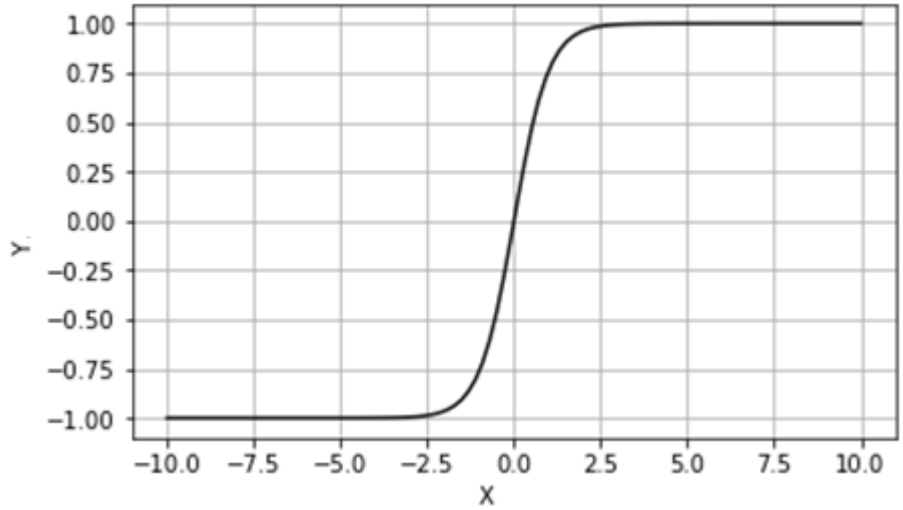
olmaması bir diğer dezavantajıdır. Sigmoid aktivasyon fonksiyonunun grafiği Şekil 2.6'da verilmiştir (Goodfellow vd., 2016: 183).



Şekil 2.6 Sigmoid Aktivasyon Fonsiyonu (Sharma, 2017)

2.2.4. Tanh Aktivasyon Fonksiyonu

Hiperbolik tanjant aktivasyon fonksiyonu, -1 ve 1 arasında değerler üretmesi gereken yapay sinir ağları için çok yaygın olarak kullanılan bir aktivasyon fonksiyonudur. Şekil 2.7'de görüldüğü gibi sigmoid fonksiyonuna benzer şekli vardır, sigmoid fonksiyonuna göre avantajı negatif sayıların kullanılabilmesidir, fakat sigmoid fonksiyonu gibi üretilen çıkışlar sıfır merkezli değildir ve kaybolan eğim sorunu ile karşı karşıya kalmaktadır (Heaton, 2015: 44).



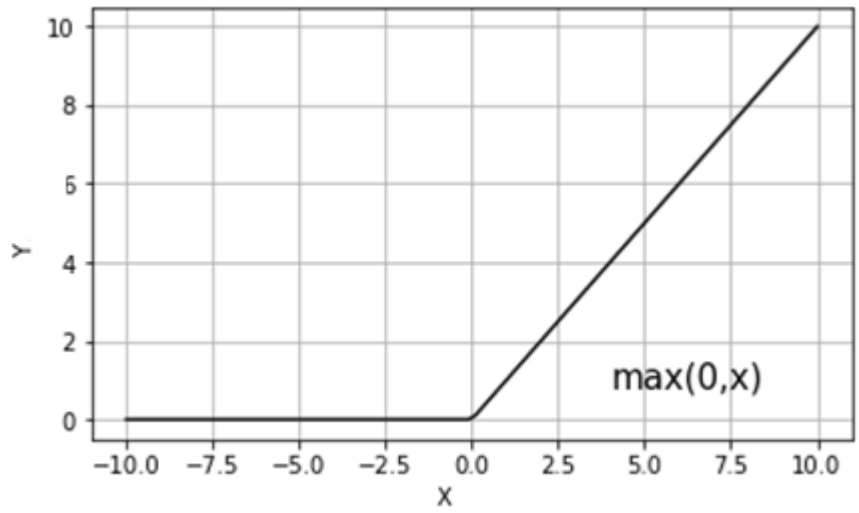
Şekil 2.7 Hiperbolik Tanjant Aktivasyon Fonksiyonu (Sharma, 2017)

2.2.5. Softmax Aktivasyon Fonksiyonu

Softmax aktivasyon fonksiyonu ikili sınıflandırma yerine sürekli verilere uygulanabilen lojistik regresyonun genelleştirilmesidir. İkiden fazla sınıf için çıkış ürettiğinden genellikle bir sınıflandırıcının çıktı katmanında yer almaktadır. Eğer sınıflandırıcı için bin adet sınıf gibi yüksek sayıda sınıf öngörülüyorsa, softmax aktivasyon fonksiyonun bir türü olan hiyerarşik softmax aktivasyon fonksiyonu kullanılmaktadır. Hiyerarşik softmax aktivasyon fonksiyonu öngörülen binlerce sınıfı ağaç yapısı kullanarak ayırmaktadır (Patterson ve Gibson, 2017: 68).

2.2.6. ReLU Aktivasyon Fonksiyonu

Son zamanlarda, rektifiye edilmiş lineer birim (ReLU) olarak adlandırılan basit bir fonksiyon, çok iyi deneysel sonuçlar ürettiği için çok popüler hale gelmiştir. ReLU aktivasyon fonksiyonu $f(x) = \max(0, x)$ şeklinde tanımlanmaktadır. Şekil 2.8'de görüldüğü gibi negatif değerler için sıfırdır ve pozitif değerler için doğrusal olarak artmaktadır (Gulli ve Pal, 2017: 68).

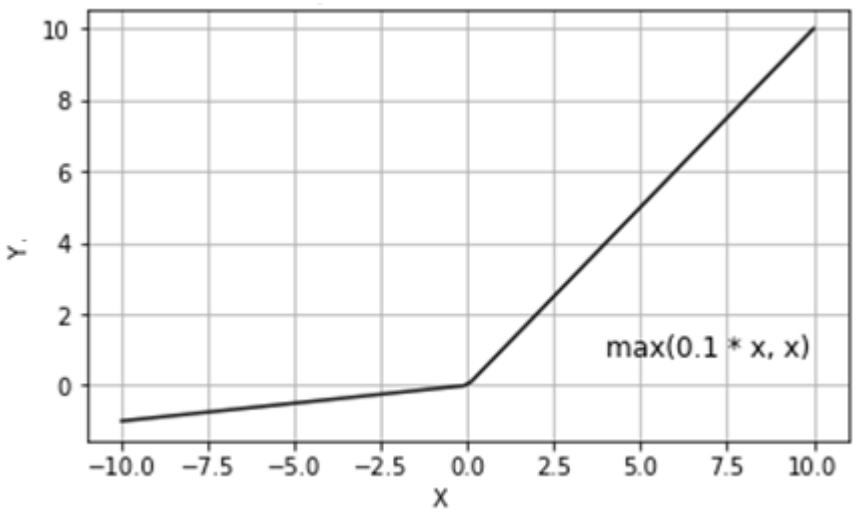


Şekil 2.8 ReLU Aktivasyon Fonksiyonu (Sharma, 2017)

Bu aktivasyon fonksiyonu ağın daha hızlı bir şekilde tahminleri istenen çıktılara yaklaştırmaktadır. ReLU aktivasyon fonksiyonun hesaplanması çok kolay olması ve şekli dolayısıyla kaybolan eğim sorununa maruz kalmadığı için derin ağlarda avantajlı bir fonksiyon olarak kabul edilmektedir. Ancak ReLU aktivasyon fonksiyonunun da bazı dezavantajları vardır. Aynı şekilde üretilen çıkışların sıfır merkezli olmamakla beraber, negatif değerler için nöron pasif kalmakta ve geriye geçiş sırasında eğimi yok etmektedir. Böylece ağırlıklar güncellenemez ve ağ öğrenemez (Sharma, 2017).

2.2.7. Leaky ReLU Aktivasyon Fonksiyonu

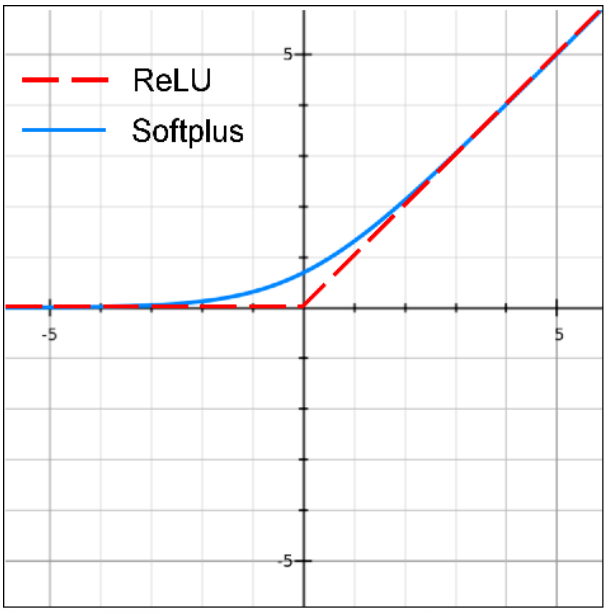
ReLU fonsiyonundaki ölen ReLU (“dying ReLU”) sorununu çözmek için Leaky (Sızdıran) ReLU aktivasyon fonksiyonu kullanılmaktadır. Leaky ReLU aktivasyon foksiyonu nöron aktif olmadığı durumda sıfırdan farklı küçük eğim sağlamaktadır. Leaky ReLU aktivasyon fonksiyonu $f(x) = \max(0.1x, x)$ şeklinde tanımlanmaktadır. Fonksiyonun grafiği şekil 2.9’da verilmiştir (Maas vd., 2013).



Şekil 2.9 Leaky ReLU Aktivasyon Fonksiyonu (Sharma, 2017)

2.2.8. Softplus Aktivasyon Fonksiyonu

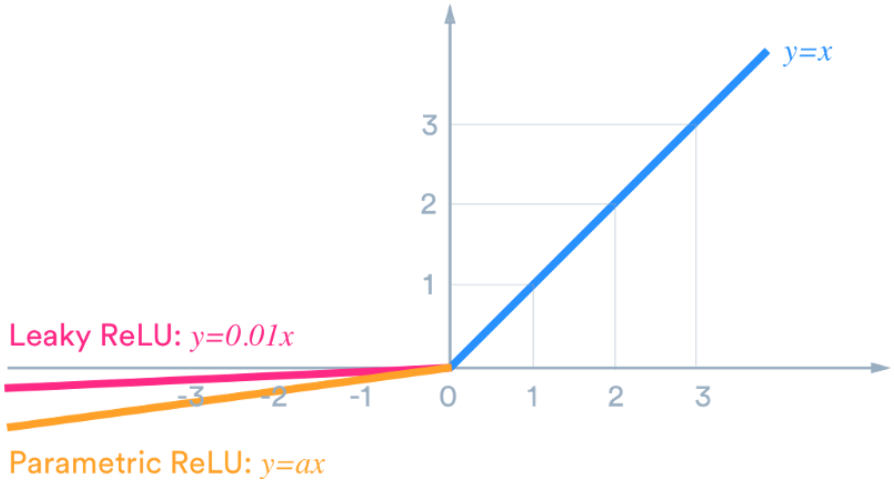
Leaky ReLU aktivasyon fonksiyonu ölen ReLU sorununa çözüm getirmekle beraber uygulamada sonuçların her zaman tutarlı olmadığı görülmektedir. Bu nedenle ReLU fonksiyonunu iyileştirme üzerine farklı çalışmalar devam etmiştir. Softplus aktivasyon fonksiyonu “ReLU'nin pürüzsüz versiyonu” olarak kabul edilmektedir. Softplus aktivasyon fonksiyonu $f(x) = \ln(1 + \exp(x))$ olarak tanımlanmakta ve Şekil 2.10’dan görüldüğü gibi ReLU'ye benzer bir şekle sahiptir (Patterson ve Gibson, 2017: 70).



Şekil 2.10 Softplus Aktivasyon Fonksiyonu (Patterson ve Gibson, 2017: 70)

2.2.9. PReLU Aktivasyon Fonksiyonu

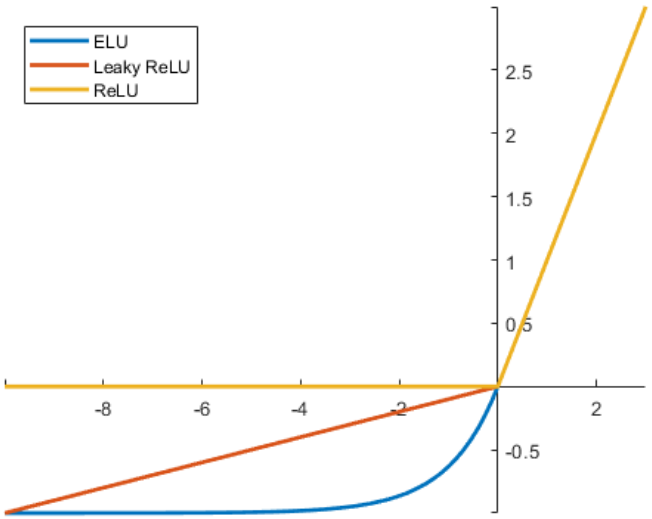
PReLU parametrik aktivasyon fonksiyonu da leaky ReLU aktivasyon fonksiyonun bir uzantısı şeklinde yorumlanabilir. Leaky ReLU fonsiyonundan farkı giriş değerlerini sabit bir değerle çarpmak yerine hiperparametre olan α ile çarpmasıdır. Parametrik ReLU aktivasyon fonksiyonu $f(x) = \max(\alpha x, x)$ şeklinde tanımlanmaktadır. Buradaki fikir, keyfi bir hiperparametre olan α'yı ağa tanıtmaktır. Bu da nöronlara negatif bölgede en iyi eğimi seçme yeteneğini vermektedir. Böylece parametrik ReLU duruma göre ReLU ya da leaky ReLU olarak davranabilir. Çalışmanın niteliğine göre ReLU aktivasyon fonksiyonlarının tüm varyasyonları denenerek problem için en iyi olan seçilmelidir (Sharma, 2017).



Şekil 2.11 PReLU Aktivasyon Fonsiyonu (Liu, 2017)

2.2.10. ELU Aktivasyon Fonksiyonu

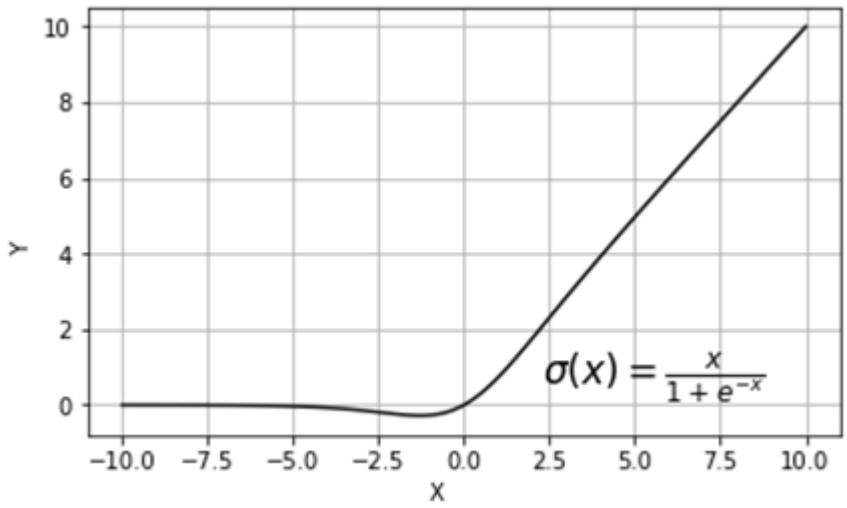
ELU Aktivasyon Fonksiyonun açılımı Exponential Linear Units (ELUs) şeklinde olup tercümesi üstel doğrusal birimler şeklindedir. ELU Aktivasyon Fonksiyonunda x değeri sıfırdan farklı olduğunda $f(x) = \alpha(\exp(x) - 1)$ şeklinde tanımlanmaktadır. Burada α hiperparametre olup, negatif değerleri kontrol altında tutarak kaybolan gradyan etkisine çözüm getirmektedir. ReLU fonksiyonundan farklı olarak, ELU fonksiyonunda aktivasyonların ortalamasını sıfıra yaklaştıran negatif değerler mevcuttur. Sıfıra yaklaşan ortalamalar ise ağın daha hızlı eğitilmesini sağlarken doğal gradyan inişini de takip etmektedir. ELU fonksiyonu Şekil 2.12'de verilmiştir (Clevert vd., 2016).



Şekil 2.12 ELU Aktivasyon Fonksiyonu (Eddins, 2018)

2.2.11. Swish Aktivasyon Fonksiyonu

Google araştırmacıları tarafından geliştirilen ve 2017 yılında yayınlanan Swish aktivasyon fonksiyonu, daha derin ağlarda daha iyi sonuçlar vermektedir. Swish aktivasyon fonksiyonu $f(x) = x\sigma(x)$ şeklinde tanımlanmaktadır. Ayrıca β eğitim parametresi kullanılarak $f(x, \beta) = 2x\sigma(\beta . x)$ şeklinde tanımlanan Swish-β versiyonu da önerilmiştir (Ramachandran vd., 2017). Swish Aktivasyonu Şekil 2.13'te de görüldüğü gibi ReLU fonksiyonuna çok benzemesine rağmen eksenin negatif bölgesinde farklı eğimli bir bölge oluşturarak nöronu pasif olma durumundan kurtarmaktadır.

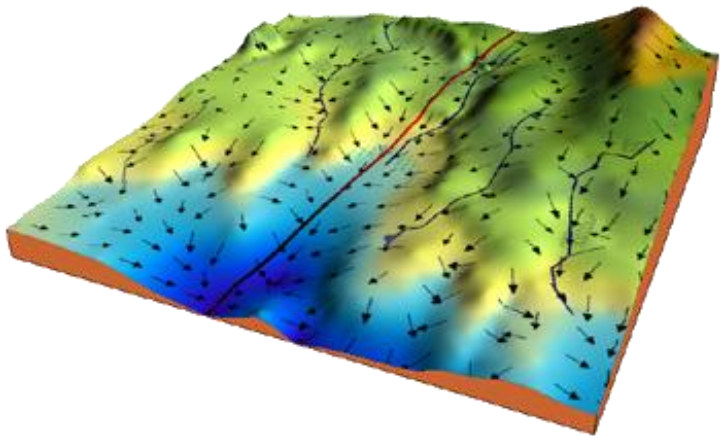


Şekil 2.13 Swish Aktivasyon Fonksiyonu (Sharma, 2017)

2.3. Gradyan İniş Algoritması

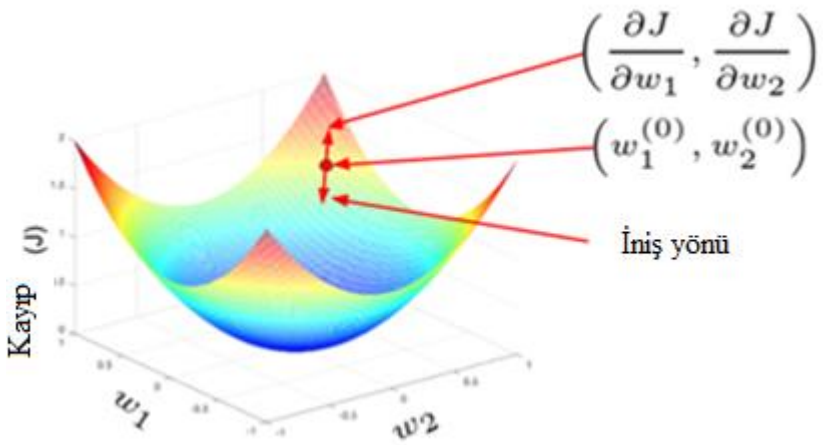
Makine öğrenimindeki tüm algoritmalar, "amaç fonksiyonu" olarak tanımlanan değeri en aza indirgeme veya en üst düzeye çıkarma esasına dayanmaktadır. Minimize edilmesi amaçlanan fonksiyonlar grubu "kayıp fonksiyonlar" olarak adlandırılmaktadır. Kayıp fonksiyonu, bir tahmin modelinin beklenen sonucu tahmin etme başarısının bir ölçütüdür (Grover, 2018). Gradyan iniş (Gradient Descent) algoritması ise kayıp fonksiyondaki eğimin negatif olduğu taraftan en dik inişten iteratif olarak hareket ederek kayıp fonksiyonlarını en aza indirgemek için kullanılan bir optimizasyon algoritmasıdır (Gradient Descent, 2017).

Kayıp fonksiyonların çok tepeli dağınık dağlar olduğu düşünülürse, "gradyan iniş" algoritması, dağın en alçak noktasına ulaşmak için dağdan aşağı kaymak gibidir (Grover, 2018). Şekil 2.14'te verilen kayıp fonksiyonun 3 boyutlu temsili olduğu varsayılırsa, buradaki amaç dağdan kahverengi ile gösterilen sağ üst köşesinden sol altta koyu mavi renk ile gösterilen denize inmektir. Oklar herhangi bir noktadan en dik inişi, yani negatif eğim yönünü göstermektedir. Başka bir deyişle kayıp fonksiyonun değerini olabildiğince hızlı azaltan yönü göstermektedir (Gradient Descent, 2017).



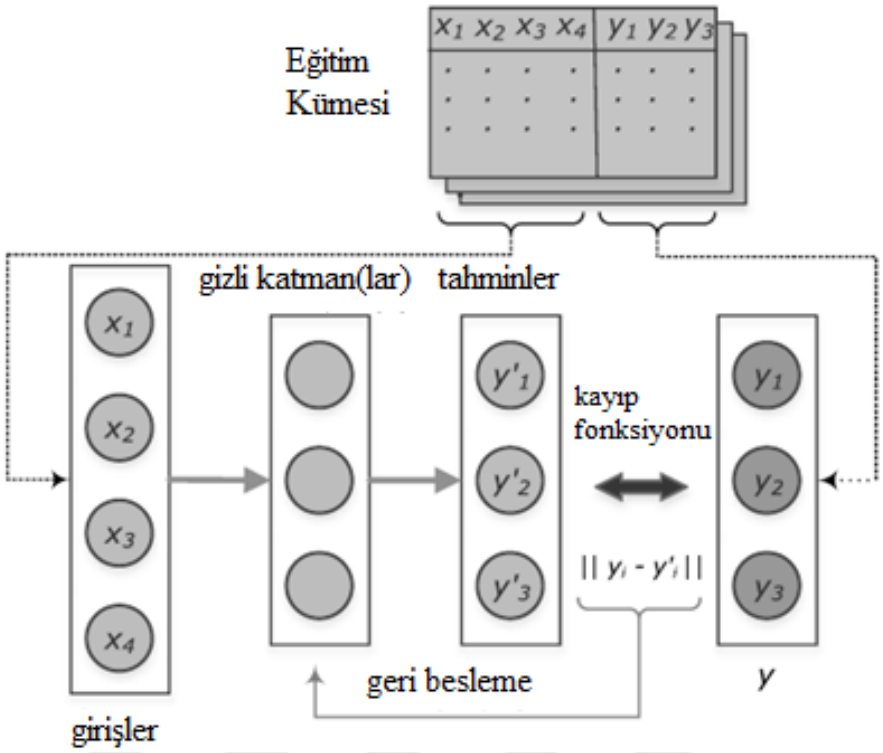
Şekil 2.14 Gradyan İniş Algoritması için Üç Boyutlu Temsil (Gradient Descent, 2017)

Yapay sinir hücresinde sadece iki ağırlık olduğunda Şekil 2.15'te görüldüğü üzere kayıp fonksiyonu bir kase şeklindedir. Eğitim sürecinin herhangi bir noktasında, kayıp fonksiyonunun ağırlıklara göre kısmi türevi kasenin o noktadaki eğimidir. Kısmi türevlerin öngördüğü yönde hareket ederek kasenin tabanına ulaşıldığında kayıp fonksiyonun değeri en aza indirgenmektedir. Bir fonksiyonun minimum seviyesini bulmak için kısmi türevlerini iteratif olarak kullanma işlemi gradyan inişi olarak adlandırılır (Sharma, 2017).



Şekil 2.15 Gradyan İniş Algoritmasının Grafiği (Sharma, 2017)

Eğitim sürecinde, gradyan iniş algoritması, kayıp fonksiyonunun türevini hesaplayarak nöronların ağırlığını ayarlamak için kullanılmaktadır. Hata oranı, ağ boyunca tekrar giriş katmanına yayılmaktadır. Her devirde her bir nöron üzerindeki ağırlıkları dengeledikten sonra, hata oranı istenen eşiğin altına düşene kadar bu eğitim döngüsünü tekrarlanmaktadır. Şekil 2.16'da, derin öğrenme modelleri için eğitim mekanizması gösterilmektedir (Mohammadi vd., 2018).



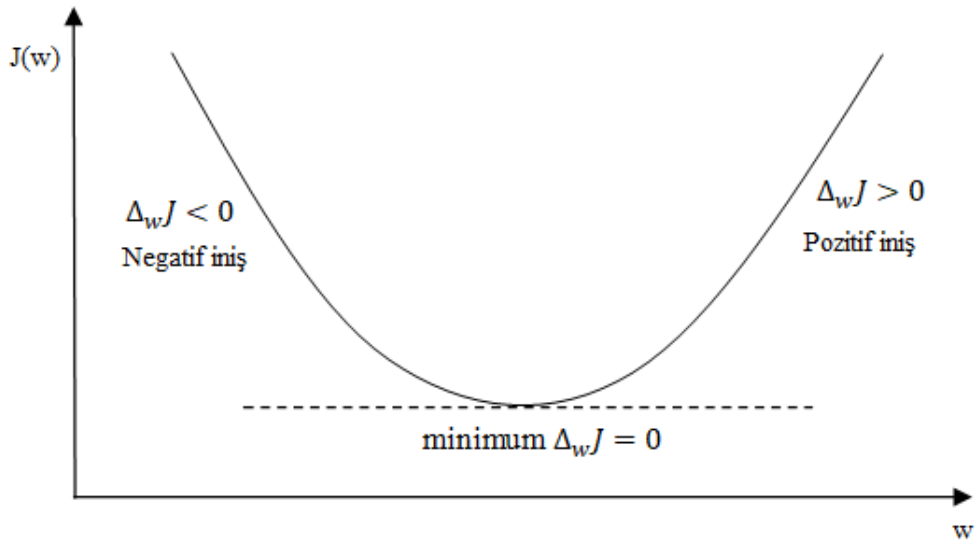
Şekil 2.16 Derin Öğrenme Modeli (Mohammadi, Al-Fuqaha, Sorour, & Guizani, 2018)

Eğitim sırasında gerçekleşen her tekrarda her ağırlık için maliyet fonksiyonunun kısmi türevini aldıktan sonra parametreler güncellenmektedir. Matematiksel ifadeleri (2.1) ve (2.2)'de verilmiştir (Dabbura, 2017). Buradaki α öğrenme katsayısıdır.

$$\frac{\partial}{\partial w}J(w) = \nabla_{\mathrm{w}}\mathrm{J} \quad \text{ve} \quad \frac{\partial}{\partial b}J(w) = \nabla_{\mathrm{b}}\mathrm{J} \qquad (2.1)$$

$$w = w - \alpha\nabla_{\mathrm{w}}\mathrm{J} \quad \text{ve} \quad b = b - \alpha\nabla_{\mathrm{b}}\mathrm{J} \qquad (2.2)$$

Gradyan İniş algoritmasını iki boyutlu düzlemde açıklamak amacıyla yanlılık parametresi b olmadığı varsayılırsa Şekil 2.17'de verildiği şekilde gösterilebilir. Eğimin pozitif olması durumunda, yani $\nabla_{\mathrm{w}}\mathrm{J} > 0$ olduğunda (2.2) kullanılarak ağırlık değeri azalmaktadır. Eğimin negatif olması durumunda, yani $\nabla_{\mathrm{w}}\mathrm{J} < 0$ olduğunda (2.2) kullanılarak ağırlık değeri artmaktadır. Bu döngü eğim sıfır oluncaya kadar devam ederek optimum ağırlık değerleri elde edilmektedir (Dabbura, 2017).



Şekil 2.17 Gradyan İniş İki Boyutlu Gösterim (Dabbura, 2017)

Hedef fonksiyonun türevini hesaplamak için kullanılan 3 farklı gradyan iniş algoritması veriye göre farklılık göstermektedir. Veri miktarına bağlı olarak, parametreyi güncelleyerek modelin doğruluğunun arttırılması ile bu güncelleme için gereken süre arasında bir denge kurulması zorunludur (Ruder, 2016).

2.3.1. Yığın Gradyan İniş (Batch Gradient Descent)

Yığın gradyan iniş algoritmasında, her döngüde parametrelerin güncellemeleri gerçekleştirilirken veri setindeki tüm örnekler dikkate alınarak yapılmaktadır. Matematiksel ifadesi (2.3)'te verilmiştir (Dabbura, 2017).

$$w = w - \alpha \nabla_w J \qquad (2.3)$$

Yığın gradyan iniş algoritmanın avantajları; sabit öğrenme katsayısı kullanıldığı için öğrenme hızında bir yavaşlama yaşanmaması, kayıp fonksiyonu dışbükey ise minimum noktaya ulaşılmasını garanti etmesi ve örnek sayısının fazla olmasıyla standart hatanın düşük olmasına sebep olması şeklinde ifade edilebilir. Yöntemin dezavantajı ise veri seti büyük olduğunda tüm örnekler işleme gireceği için algoritmanın yavaşlamasıdır. Ayrıca bazı örnekler fazla katkıda bulunmadığı halde gereksiz işlenmektedir (Dabbura, 2017).

2.3.2. Stokastik Gradyan İniş

Stokastik gradyan iniş algoritmasında tüm örnekleri işleme almak yerine eğitim verisi içinden rastgele küçük bir kesit alınarak bu veri kümesi için ortalama kayıp hesaplanmaktadır (Ganguly, 2017).

$$w = w - \alpha \nabla_w J\,(x^i, y^i; w) \qquad (2.4)$$

Stokastik gradyan iniş algoritmasının bir diğer yolu da eşitlik (2.4)'te gösterildiği gibi her örnek için kayıp değeri hesaplamaktır. Stokastik gradyan iniş algoritması diğer algoritmalara göre daha hızlı olduğundan çevrimiçi öğrenmede oldukça kullanışlıdır. Özellikle veri setinin büyük olduğu durumda yığın gradyan iniş algoritması fazlaca gereksiz hesaplamalar yaparken, stokastik gradyan iniş algoritması her seferinde güncelleme yaptığından bu gereksiz hesaplamanın önüne geçmektedir. Ayrıca yığın gradyan iniş algoritmasındaki değişimler daha az seviyede olduğundan minimuma ulaşması uzun sürerken stokastik gradyan iniş algoritmasında bu değişim daha büyük olduğundan potansiyel minimuma sıçrama gerçekleştirmektedir. Öte yandan, bu sıçramaların minimumu ıskalama ihtimali olduğundan minimuma yakınlaşmayı zorlaştırmaktadır. Ancak, öğrenme katsayısını yavaşça düşürdüğümüzde yığın gradyan iniş algoritması ile hemen hemen aynı yakınsama davranışını göstermektedir (Ruder, 2016).

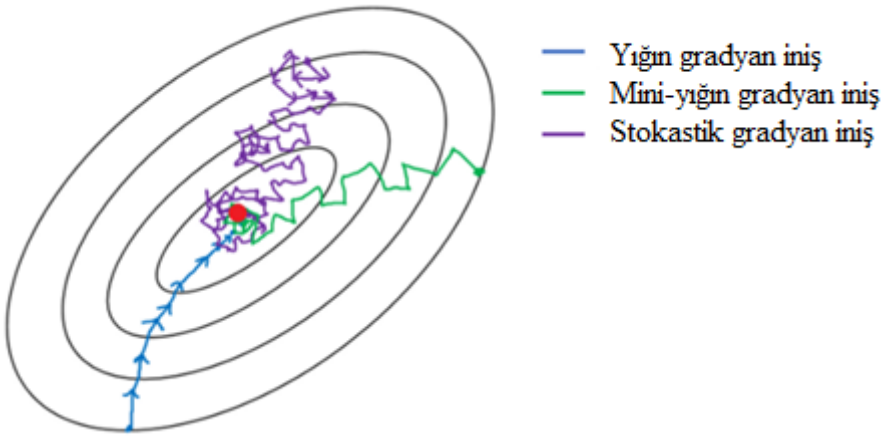
2.3.3. Mini-yığın Gradyan İniş

Mini-yığın (batch) gradyan iniş algoritmasında tüm örnekleri işleme almak yerine, *b* sayıda örneği dahil ederek mini eğitim kümeleri oluşturulduktan sonra bu kümelerdeki örnekler kullanılarak öğrenme gerçekleşmektedir. Bu işlem için önceden var olan sıralamadan etkilenmemek için veri rastgele karıştırılmaktadır. Daha sonra *b* sayıda kümeye bölünmektedir. Eğer veri seti belirtilen küme sayısına bölünemiyorsa, en son kalan küme eksik haliyle işleme alınır. Küme boyutları donanımsal nedenlerden dolayı 32,64,128 vb. sayılarda ikinin katları olacak şeklinde seçilmektedir. Mini-yığın gradyan iniş algoritmasının matematiksel gösterimi (2.5)'te verilmiştir (Dabbura, 2017).

$$w = w - \alpha \nabla_{\mathrm{w}} \mathrm{J}\left(x^{i:i+b}, y^{i:i+b}; w\right) \qquad (2.5)$$

Mini-yığın gradyan iniş algoritmasının avantajları tüm örnekleri işleme almak yerine daha az sayıda örnek işleme dahil edildiğinden dolayı daha hızlı çalışmakta ve örneklerin rastgele seçilmesinden dolayı öğrenmeye fazla katkıda bulunmayan veya benzer örneklerden kaçınmaya fayda sağlamaktadır. Yöntemin dezavantajı öğrenme sürecinde her döngüde ileri ve geri sıçramalar yapacağından dolayı minimum bölgenin etrafında dolaşmakta ama yaklaşmamaktadır. Bu nedenle minimuma yaklaştıkça öğrenme katsayısının düşürülmesi gerekmektedir (Dabbura, 2017). Fakat öğrenme katsayısını düşürme işlemi tüm parametreler için aynı anda yapılacağından farklı frekansa sahip parametreler de bu durumdan etkilenmektedir.

Bunun yanında eğer kayıp fonksiyonu dışbükey değilse lokal minimum noktasında hapsolup global minimum noktasına hiçbir zaman ulaşamama ihtimali bulunmaktadır (Ruder, 2016). Şekil 2.18'de üç faklı gradyan iniş algoritmasının minimuma ulaşırken attığı adımlar gösterilmiştir (Dabbura, 2017).



Şekil 2.18 Gradyan İniş Algoritmasının Minimuma Ulaşma Yörüngesi (Dabbura, 2017)

2.4. Kayıp Fonksiyonları

Kayıp fonksiyonu (Loss function) terimi bazı kaynaklarda hata fonksiyonu veya maliyet fonksiyonu olarak ifade edilmektedir. Bazı kaynaklar bu terimleri birbirinin yerine kullanırken, bazı makine öğrenimi kaynakları da bu terimlerin bazılarına özel bir anlam yüklemektedir (Goodfellow vd., 2016: 82). Bu terimleri açıklamak amacıyla en yaygın kullanılan regresyon kaybı fonksiyonu olan mean square error (ortalama karesel hata-MSE) fonksiyonu eşitlik (2.6)'da gösterilmiştir.

$$\mathcal{L}(\hat{y} - y) = \frac{1}{2}(\hat{y} - y)^2 \qquad (2.6)$$

Burada $\mathcal{L}$ tek giriş vektörü için Hata Kareler Ortalaması fonksiyonunu kullanarak hata değeri hesaplamaktadır. Örneklemdeki tüm giriş vektörleri için kayıp hesaplanarak eşitlik (2.7)'de gösterildiği gibi ortalaması alındığına Maliyet fonksiyonu (J) olarak ifade edilmektedir (Ng, 2018).

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^i - y^i) \qquad (2.7)$$

Maliyet fonksiyonu hesaplandıktan sonra eşitlik (2.8)'de verildiği gibi ağırlıklara göre türevi alınarak ağırlıklardan çıkarılmaktadır. Böylece ağırlıklar güncellenir ve bir sonraki iterasyona hazır hale gelir. Bu döngü maliyet fonksiyonu için minimun değeri bulunana kadar devam etmektedir (Agrawal, 2017).

$$w^{k+1} = w^k - \frac{\partial}{\partial w^k} J(w, b) \qquad (2.8)$$

Kayıp fonksiyonları Şekil 2.19'da gösterildiği gibi iki grup altında toplanabilir: Sınıflandırma ve regresyon kayıp fonksiyonları (Grover, 2018).

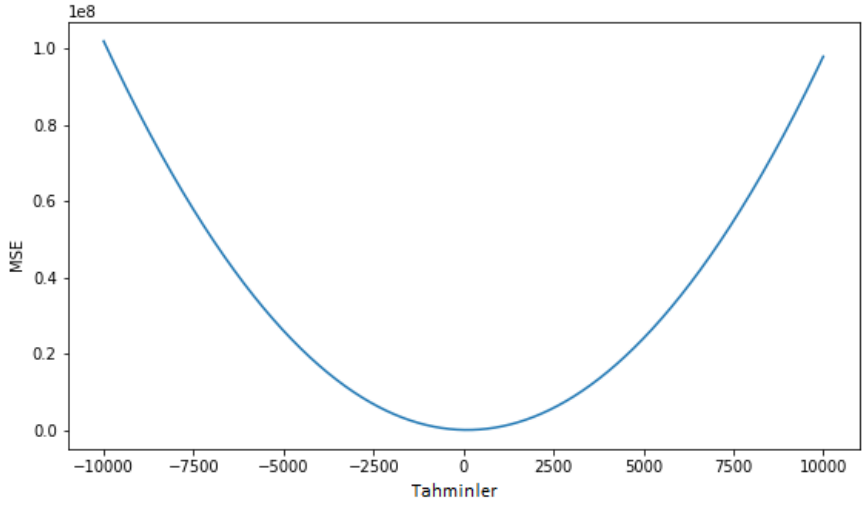
Şekil 2.19 Kayıp Fonksiyonların Sınıflandırılması (Grover, 2018)

2.4.1. Ortalama Karesel Hata

Makine öğrenmesinde en yaygın kayıp fonksiyonu olan ortalama karesel hata (Mean Square Error – MSE), hedef değişken ile tahmin edilen değerler arasındaki mesafelerin karesel toplamıdır. Literatürde aynı zamanda kuadratik kayıp veya L2 kaybı olarak da yer alan kayıp fonksiyonu eşitlik (2.9) kullanılarak ifade edilmektedir (Parmar, 2018).

$$OKH = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \qquad (2.9)$$

Şekil 2.20'de verilen grafikte gerçek değeri 100 olan problem için tahminler verilmiştir. Burada 100 değerinden uzaklaştıkça hatanın hızla artmakta olduğu gösterilmiştir (Grover, 2018).

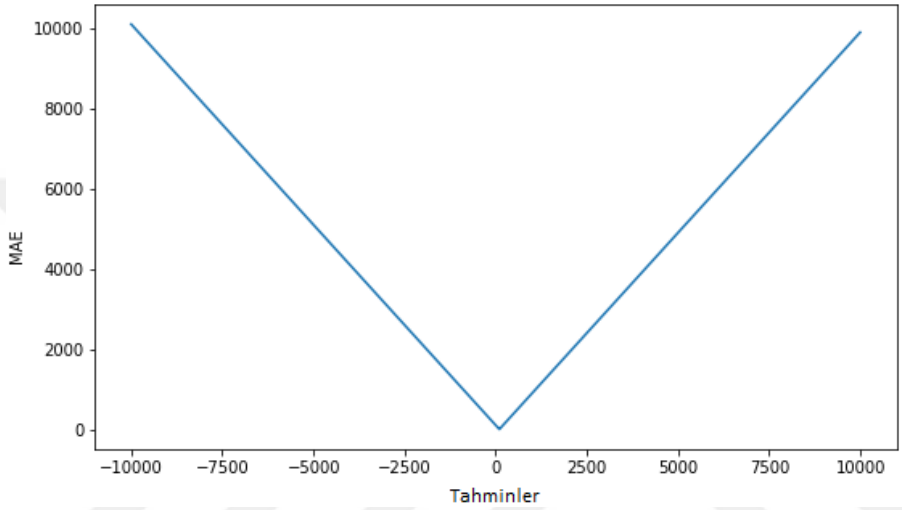


Şekil 2.20 Ortalama Karesel Hata (Grover, 2018)

2.4.2. Mutlak Hatalar Ortalaması

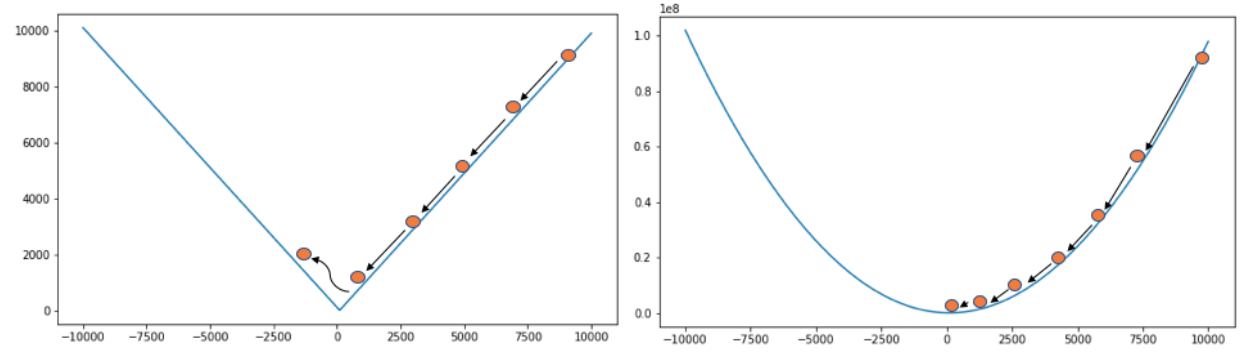
Mutlak hatalar ortalaması (Mean Absolute Error - MAE), regresyon modellerinde kullanılan bir diğer kayıp fonksiyonu olup hedef ve tahmin edilen değişkenler arasındaki mutlak farkların toplamı alınarak hesaplanır. Literatürde L1 kayıp fonksiyonu olarak da geçen mutlak hatalar ortalaması kayıp fonksiyonunun matematiksel ifadesi (2.10)'da, grafiği ise Şekil 2.21'de verilmiştir (Parmar, 2018).

$$MHO = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \qquad (2.10)$$



Şekil 2.21 Mutlak Hatalar Ortalaması (Grover, 2018)

Gradyan iniş algoritmasınındaki inişleri daha hızlı gerçekleştirdiği için ortalama karesel hata fonksiyonundan daha hızlı sonuç vermekte, ancak bu iniş sırasında doğru değere yaklaştıkça minimum noktasını atlama riski bulunmaktadır. Bu nedenle mutlak hatalar ortalaması kayıp fonksiyonunun kullanıldığı durumda dinamik öğrenme katsayısı kullanılmalıdır. Bu durum Şekil 2.22'de gösterilmiştir. Ayrıcı hesaplamalarda değerlerin karesi alınmadığında aykırı değerlere karşı daha dirençlidir (Grover, 2018).



Şekil 2.22 MSE ve MAE Karşılaştırması (Grover, 2018)

2.4.3. Ortalama Yanlılık Hatası

Ortalama yanlılık hatası kayıp fonksiyonu makine öğrenimi alanında, diğer fonksiyonlara oranla daha az kullanılmaktadır. Matematiksel ifadesi (2.11)'de verilmiştir. Bu fonksiyon, mutlak hatalar ortalaması fonksiyonunun mutlak değerinin alınmamış hali olduğundan negatif değerle pozitif değerler toplandığında toplam küçük görüneceğinden pratikte çok tercih edilmemektedir. Ancak modelin pozitif veya negatif yanlılığa sahip olup olmadığını belirlemede oldukça faydalıdır (Parmar, 2018).

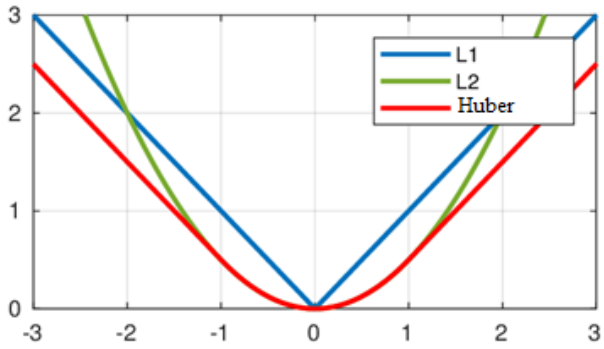
$$OYH = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n} \qquad (2.11)$$

2.4.4. Huber Kayıp Fonksiyonu

Huber kayıp fonksiyonu (Huber Loss Function), karesel ortalama hata ve mutlak hatalar ortalaması fonksiyonlarının avantajlarını kullanmaktadır. Karesel ortalama hata gibi verideki aykırı değerlere daha az duyarlı olup, aynı zamanda 0 noktasında minimumu atlama riskini azaltmaktadır. Hata arttığında mutlak hatalar ortalaması kayıp fonksiyonu (L1) gibi davranmakta, hata azaldığında karesel ortalama hata kayıp fonksiyonu (L2) gibi davranmaktadır. Bu nedenle pürüzsüz mutlak hatalar ortalaması (Smooth Mean Absolute Error) olarak da adlandırılmıştır (Owen, 2007).

L1 ve L2 kayıp fonksiyonlarını birleştiren Huber kayıp fonksiyonunun ya da diğer bir değişle pürüzsüz L1 fonksiyonunun matematiksel ifadesi (2.12)'de verilmiştir. Grafiksel gösterimi Şekil 2.23'de verilmiştir (Feng vd., 2018).

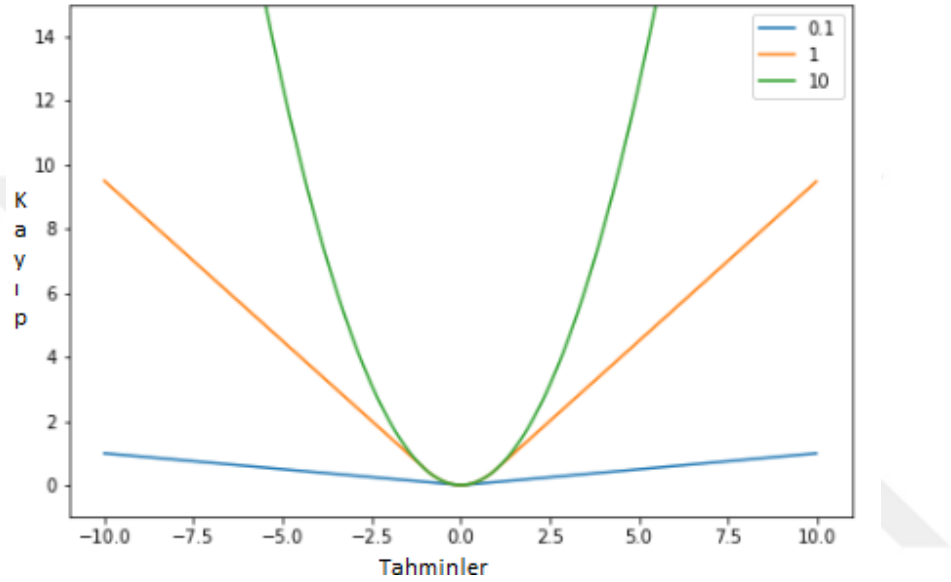
$$Huber\ Loss = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & |y_i - \hat{y}_i| < 1 \\ |y_i - \hat{y}_i| - \frac{1}{2} & |y_i - \hat{y}_i| \geq 1 \end{cases} \qquad (2.12)$$



Şekil 2.23 Huber Kayıp Fonksiyonu (Feng, Kittler, Awais, Huber, & Wu, 2018)

Veri içinde hangi durumların aykırı durum olarak sayılacağını belirlemek için δ (Delta) hiperparametresi kullanılarak yazılan Huber fonksiyonu eşitlik (2.13)'te verilmiştir. Yani delta hiperparametresi hangi fonksiyona daha yakın davranacağını belirlemektedir. Farklı katsayıları için farklı renklerle çizilen Huber kayıp fonksiyonun grafiksel gösterimi Şekil 2.24'te verilmiştir (Grover, 2018).

$$Huber\ Loss\ \delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & |y_i - \hat{y}_i| < \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & |y_i - \hat{y}_i| \geq \delta \end{cases} \quad (1.13)$$

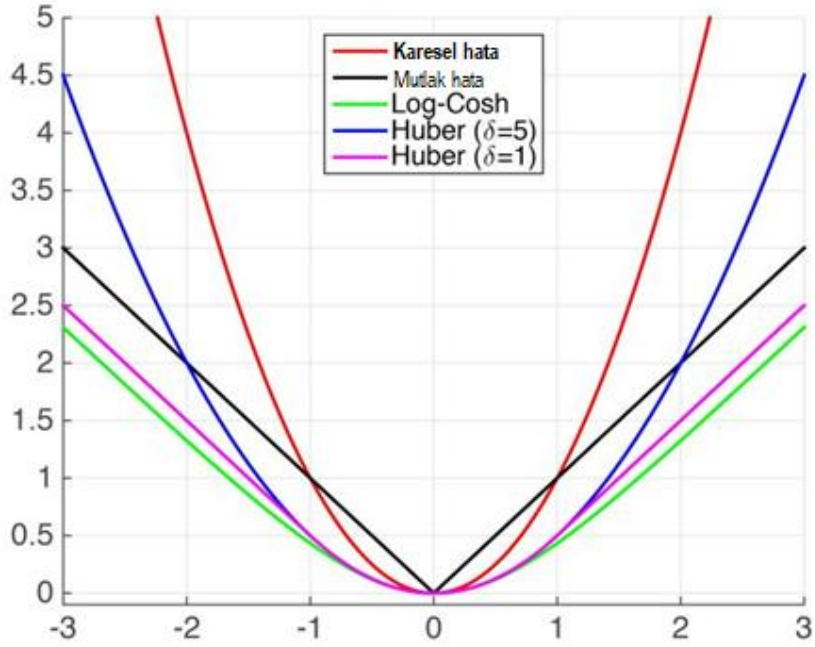


Şekil 2.24 Farklı δ Katsayıları için Huber Kayıp Fonksiyonu (Grover, 2018)

2.4.5. Log-Cosh Kayıp Fonksiyonu

Log-Cosh kayıp fonksiyonu Huber kayıp fonksiyonu ile aynı özelliğe sahip ancak iki kez türevi alınabildiği için bazı makine öğrenme algoritmalarında daha kullanışlıdır. Log-Cosh kayıp fonksiyonun matematiksel gösterimi eşitlik (2.14)'te verilmiştir. Şekil 2.25'ten görüldüğü gibi beklenen ile tahmin edilen değerler arasındaki fark küçüldükçe karesel ortalama hata kayıp fonksiyon grafiğine yakın, büyüdükçe mutlak hatalar ortalaması kayıp fonksiyonuna yakındır (Weinberger, 2018).

$$Log - Cosh\ Kaybı = \log(\cosh(\hat{y}_i - y_i)) \quad (2.14)$$



Şekil 2.25 Log-Cosh Kayıp Fonksiyonu (Weinberger, 2018)

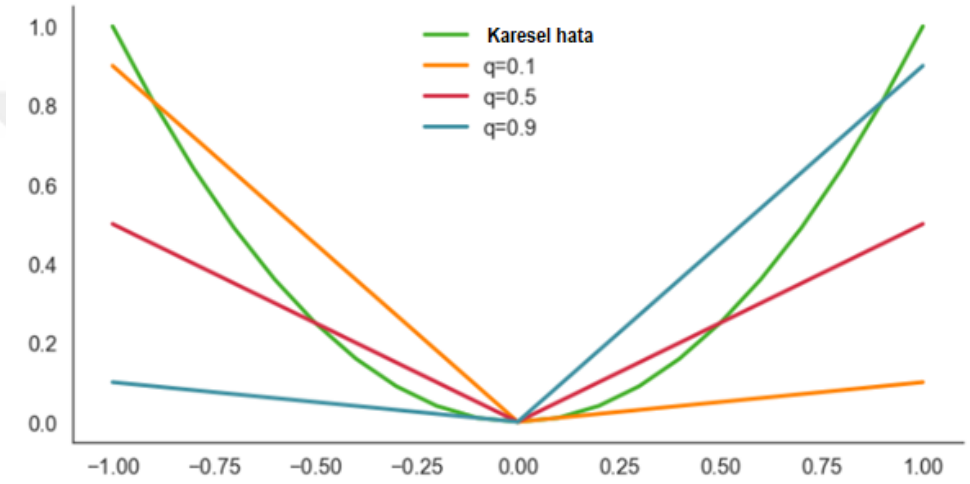
2.4.6. Kuantil Kayıp Fonksiyonu

Gerçek dünyada tahmin problemlerinin çoğunda belirsizlik olduğu için mükemmel sonuçlar almak oldukça zordur. Ancak tahmin aralığı bilindiğinde birçok problemde büyük kolaylık sağlamaktadır. Kuantil kayıp fonksiyonu (Quantile Loss Function), yalnızca nokta tahminleri yerine bir aralığın tahmin edilme gerekli olduğu durumlarda kullanılmaktadır. En küçük kareler regresyonunda aralık tahmini hataların sabit varyansa sahip olduğu varsayımına dayanmaktadır. Dolayısıyla bu varsayım bozulduğunda doğrusal regresyon modellerine güvenilmemektedir. Bu durumda doğrusal olmayan regresyon modellerini ya da ağaç tabanlı modellerini kullanarak, doğrusal çizgiye yakın model olsa bile doğrusal çizgiyi takip fikrinden fedakârlık yapılmaktadır (Grover, 2018).

Hataların değişken varyanslı olma durumlarının yanı sıra aykırı değerlerin varlığı, özellikle uzun kuyruklu modellerde en küçük karelerin tahmincisini çok zayıf bir tahmin ediciye dönüştürmektedir. Bu durumlara direnç gösteren Kuantil kayıp fonksiyonu, hangi tür hataları (negatif ya da pozitif) daha ön planda tutmak istediğimiz fikrine dayanmaktadır. Kuantil kayıp fonksiyonun matematiksel ifadesi eşitlik (2.15)'te verilmiştir (Koenker ve Bassett, 1978).

$$Kuantil\ Kayıp = \sum_{i=y_i \geq \hat{y}_i} \theta \left|y_i - \hat{y}_i\right| + \sum_{i=y_i < \hat{y}_i} (1-\theta) \left|y_i - \hat{y}_i\right| \quad (2.15)$$

Burada θ hangi kuantile odaklandığımızı göstermektedir. Örneğin bir öğrencinin sınavdan aldığın not, sınıfın θ oranının aldığı nottan daha yüksek ve (1-θ) oranının aldığı nottan daha düşükse, öğrencinin θ. kuantilde olduğu şeklinde değerlendirme yapılmaktadır. Yani sınıfın yarısı bu öğrenciden daha iyi not aldıysa, yarısı daha düşük aldıysa bu öğrenci 50. kuantilde olduğu şeklinde değerlendirme yapılmaktadır (Koenker ve Hallock, 2001). Şekil 2.26'da Kuantil kayıp fonksiyonunun farklı θ değerleri için grafiksel gösterimi verilmiştir. Buradan negatif hatalardan doğan kayıpları azaltmak da önemli ise, mavi renk ile gösterilen grafiği elde etmek için θ=0.9 seçilmektedir (Ripert, 2018).



Şekil 2.26 Kuantil Kayıp Fonsiyonu (Ripert, 2018).

2.4.7. Log-Loss Kayıp Fonksiyonu

Lojistik kayıp fonksiyonu sınıflandırmadan daha çok sınıflara ait olma olasılığı önemli olduğu durumlarda kullanılmaktadır. Örneğin reklamcılık sektöründe "bir reklama tıklama olasılığı" hesaplanarak elde edilen bilgi potansiyel belirlemede kullanılmakta ve daha sonra paraya dönüştürülmektedir. Olasılıkların tahmin edilmesi, 0 ile 1 arasında sayıların üretilmesi anlamına gelmektedir. Olasılıklarla çalışma söz konusu olduğunda logaritmik dönüşümler kullanılmaktadır (Patterson ve Gibson, 2017). Lojistik kayıp fonksiyonu literatürde çapraz entropi kayıp fonksiyonu olarak da yer almaktadır (Parmar, 2018).

$$\log \text{loss} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \qquad (2.16)$$

Eşitlik (2.16)'da verilen N değişkeni eğitim setindeki eleman sayısını veya test içerisindeki soru sayısını temsil etmektedir. Logaritma 0 ile 1 arasında negatif değer alan bir fonksiyon olup pozitif değere dönüştürmek için eşitlik negatif işaretle

başlamaktadır, çünkü kayıp fonksiyonların minimum noktasını bulmak için pozitif değerler kullanılmaktadır. Eşitlikteki (+) ile ayrılan iki terim y_i değerine bağlı olarak 0 veya 1 değerini almaktadır. y_i = 0 ise ilk terim 0 olacak, y_i=1 ise ikinci terim 0 olacaktır (Heaton, 2015).

2.4.8. Dengeli Çapraz Entropi

Birçok görüntü tanıma tekniğinde ağın eğitilmesinde büyük çoğunluğu obje içermeyen veri seti kullanılmaktadır. Veri setindeki bu dengesizlik iki soruna neden olmaktadır. Birincisi birçok negatif sinyal ürettiğinden eğitim aşaması verimsiz hale gelmektedir. İkincisi negatif sinyallerin çok fazla olması durumunda model bozulmuş hale gelebilir. Bu durumda bir veri madenciliği yaparak daha karmaşık verileri içeren veri seti çalışması yapmak dengesizliği gidermenin bir yoludur. Bir diğer yolu ise dengelenmiş çapraz entropi (Balanced Cross Entropy) kayıp fonksiyonunu kullanmaktır. Çapraz Entropinin matematiksel ifadesi (2.17)'de verildiği şekilde yazıldığında ve modelin tahmin edilen olasılık fonksiyonu (2.18)'de verildiği şekilde tanımlandığında Çapraz Entropi eşitlik (2.19) 'da verildiği şekle dönüşmektedir (Lin vd., 2017).

$$Çapraz\ Entropi = \begin{cases} -\log(p) & eğer\ \ y = 1 \\ -\log(1-p) & diğer\ durumda \end{cases} \quad (2.17)$$

$$p_t = \begin{cases} p & eğer\ \ y = 1 \\ 1-p & diğer\ durumda \end{cases} \quad (2.18)$$

$$Çapraz\ Entropi = -\log(p_t) \quad (2.19)$$

Veri setindeki dengesizliği gidermek için pozitif sinyaller için α katsayısı, negatif sinyaller için 1-α değeri tanımlandığında bir α_t parametre elde edilmiş olur. Bu parametre çapraz entropi denklemi ile birleştiğinde (2.20) eşitliğine dönüşmektedir. Böylece negatif sinyaller ile pozitif sinyallerin önemini bu katsayı ile ayarlayarak öğrenme sürecinde dengesizlik giderilmektedir (Lin vd., 2017).

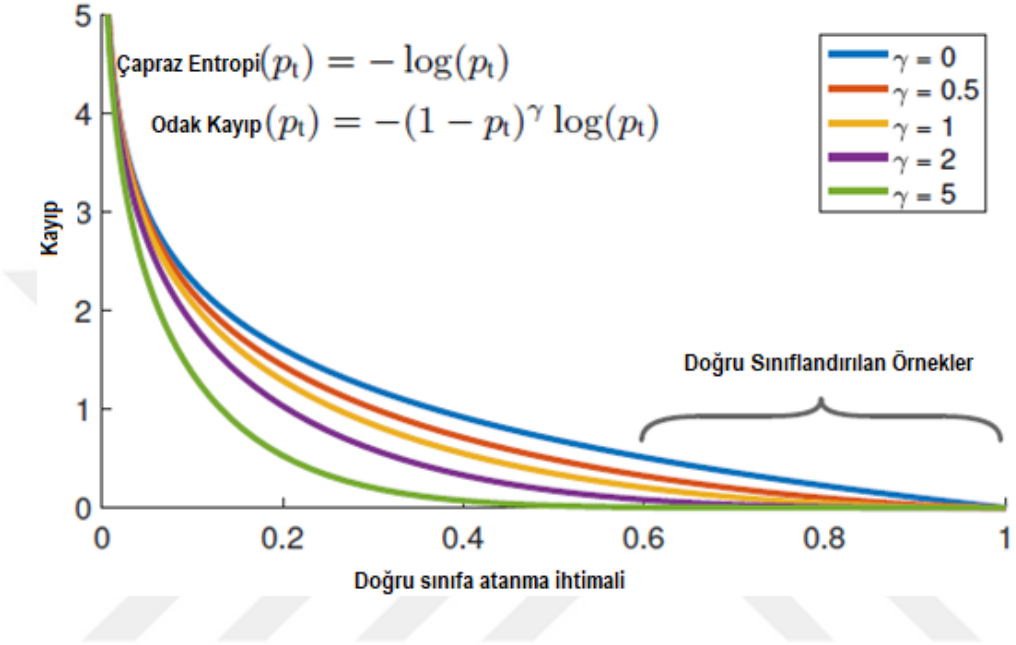
$$Dengeli\ Çapraz\ Entropi = -\alpha_t \log(p_t) \quad (2.20)$$

2.4.9. Odak Kayıp Fonksiyonu

Dengeli çapraz entropi kayıp fonksiyonu negatif sinyaller ile pozitif sinyallerin önemini dengelemekte, ancak eğitim setindeki kolay ve zor örnekler arasında ayırım yapamamaktadır. Google araştırmacıları tarafından geliştirilen odak kayıp (Focal Loss) fonksiyonu, kolay örneklerin aralığını azaltmak ve zor örneklere

odaklanmak amacıyla çapraz entropi kayıp fonksiyonuna $(1-p_t)^{\gamma}$ ayarlama faktörü eklenmiştir. Burada $\gamma \geq 0$ odaklanma parametresidir. Odak kayıp fonksiyonun matematiksel ifadesi (2.21)'de verilmiştir. Odak kayıp fonksiyonun farklı γ için çizilen grafik Şekil 2.27'de verilmiştir (Lin vd., 2017).

$$Odak\ Kayıp\ Fonksiyonu = -(1-p_t)^{\gamma}\log(p_t) \qquad (2.21)$$



Şekil 2.27 Odak Kayıp Fonksiyonu (Lin vd., 2017)

2.4.10. Kullback-Leibler Iraksaması

Bir değişken üzerinde iki olasılık dağılımının arasındaki farkı ölçmek için kullanılan Kullback-Leibler (KL) ıraksaması (KL Divergence/Relative Entropy) veri madenciliği literatüründe yaygın olarak kullanılmaktadır. Göreceli entropi kavramına çok yakın olan KL ıraksaması, tahmin edilen değerlerin beklenen değere göre bilgi kaybının ölçüsüdür. KL ıraksaması mesafe kavramı gibi algılanmamalıdır, çünkü p ve q olasılık dağılımları arasındaki fark simetrik değildir. Yani $\mathrm{d(p,q)} \neq \mathrm{d(q,p)}$. İki olasılık dağılımı arsındaki farkı bulabilmek için eşitlik (2.22)'de verildiği gibi logaritmik fark alınmaktadır (Kurt, 2017).

$$D_{KL} = \sum_{i=1}^{N} p(\log(p) - \log(q)) \qquad (2.22)$$

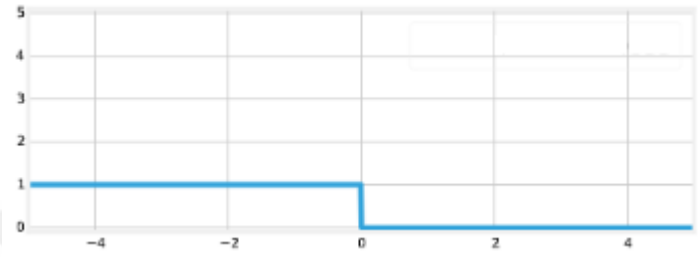
Buradan $\log(a) - \log(b) = \log(\frac{a}{b})$ olduğundan . KL ıraksaması (2.23)'teki eşitliğe dönüşmektedir.

$$D_{KL} = \sum_{i=1}^{N} p\log(\frac{a}{b}) \qquad (2.23)$$

2.4.11. Sıfır-Bir Kayıp Fonksiyonu

Sıfır-bir kayıp fonksiyonu (Zero-one Loss Function) sürekli bir fonksiyon olmadığından optimizasyon yapılamamaktadır. Sıfır-bir kayıp fonksiyonunun matematik gösterimi (2.24)'te ve grafiksel gösterimi Şekil 2.28'de verilmiştir (Weinberger, 2018).

$$Sıfır - Bir\ Kayıp = \begin{cases} 0 & eğer\ \ y_i * \hat{y}_i > 0\ ise \\ 1 & diğer\ durumda \end{cases} \qquad (2.24)$$



Şekil 2.28 Sıfır-Bir Kayıp Fonksiyonu

2.4.12. Üstel Kayıp Fonksiyonu

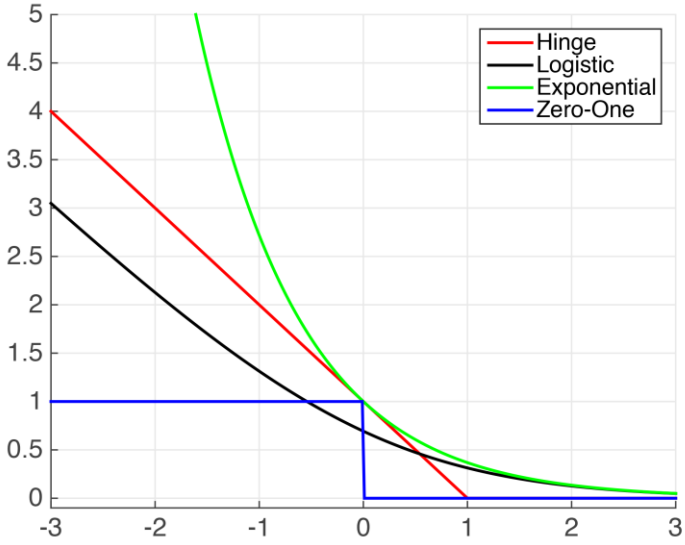
Üstel kayıp fonksiyonu (Exponential Loss Function) tahmin edilen değerler ile beklenen değerler arasındaki fark üstel olarak arttığından literatürde agresif fonksiyon olarak tanımlanmaktadır. Adaboost gibi algoritmalarda güzel yakınsama sonuçları sağlamakta, ancak gürültülü verilerde sorunlara neden olabilmektedir. Üstel kayıp fonksiyonunun matematik gösterimi (2.25)'te verilmiştir (Weinberger, 2018).

$$Üstel\ Kayıp = \sum_{i=1}^{N} e^{-y_i * \hat{y}_i} \qquad (2.25)$$

2.4.13. Hinge Kayıp Fonksiyonu

Makine öğrenmede zor örneklerin sınıflandırması söz için ağın optimize edilmesi gerektiğinde "Hinge Loss" ya da diğer ismiyle "Çok Sınıflı SVM (Support Vector Machines)" en çok kullanılan kayıp fonksiyonudur. Standart SVM (Vektör Destek Makineler) için kullanıldığında, kayıp fonksiyonu, doğru ile her iki sınıftaki en yakın noktalar arasında mesafeyi göstermektedir. Hinge kayıp fonksiyonunun matematik gösterimi (2.26)'da verilmiştir. Şekil 2.29'da Hinge kayıp fonksiyonların grafiksel gösterimi diğer sınıflandırmada kullanılan kayıp fonksiyonlarla karşılaştırmalı olarak verilmiştir (Weinberger, 2018).

$$SVM\ Kayıp = \frac{1}{N}\sum_{i=1}^{N} \max(0, -y_{ij} * \hat{y}_{ji}) \qquad (2.26)$$



Şekil 2.29 Sınıflandırma için Kayıp Fonksiyonlar (Weinberger, 2018)

2.5. Optimizasyon Algoritmaları

Makine öğrenimi optimizasyon yoluyla denklemdeki hatayı en aza indirgeme tekniklerine dayanmaktadır. Optimizasyonda, beklenen değerlere en yakın tahminleri veren şartlar ve parametreler üzerine odaklanılmaktadır. Daha iyi tahminler üretmek için ağırlıkları ayarlama sürecine ise parametre optimizasyonu denmektedir. Bilimsel yöntem geliştirme sürecinde olduğu gibi kurulan hipotez, dünyadaki gerçek olaylar için en iyi tanımlama bulunana kadar tekrar tekrar değiştirilip test edilmektedir. Her ağırlık seti hangi girdilerin hangi anlama geldiğini ifade eden hipotezi temsil etmektedir. Ağırlıklar, ağın girişleri ile tahmin etmek istenen çıktılar arasındaki korelasyona ilişkin varsayımlardır. Tüm olası ağırlıklar ve bunların kombinasyonları, bu problemin hipotez alanı olarak tanımlanabilir. Bu hipotez alanı içinde en iyi hipotezi bulma girişimi makine öğreniminde kayıp fonksiyonları ve optimizasyon algoritmaları kullanılarak yapılmaktadır. Ağın girdi sayısı yükseldikçe alan artmakta ve arama zorlaşmaktadır. Bu nedenle öğrenme sürecini büyük bir kısmı hangi girdinin önemli olduğuna, hangi girdinin görmezden gelinmesi gerektiğine karar verme sürecidir (Patterson ve Gibson, 2017: 27).

Optimizasyon, uygulamalı matematikte merkezi bir rol oynamakta ve gerçek dünya problemlerinin modellemesinde yaygın olarak kullanılmaktadır. Optimizasyon, görüntü bilimi, finans, sinyal işleme ve makine öğrenimi gibi çok uygulama alanını kapsayan karmaşık sistemlerdir. Optimizasyon problemleri dışbükey, pürüzlü ve hatta dışbükey olmayan yüksek boyutlu problemlerdir. Verinin

faydalı bir şekilde kullanılabilir olması ve bu karmaşık problemlerin çözülmesi için basit ve ölçeklenebilir algoritmalar üretme ihtiyacı duyulmuştur (Teboulle, 2018).

Çoğu optimizasyon algoritması iterasyon (tekrarlama) şeklindedir, yani ardışık hesaplamalar yaparak istenen çözüme yakınsama göstermektedir. Programlamada iterasyon için döngü ifadesi kullanılmaktadır. Döngünün i. tekrarında girdinin x_i değeri hesaplanmaktadır. Bu döngü ancak yakınsama kriteri ya da durma kriteri yerine getirildiği takdirde sona ermektedir (Gentle, 2004: 32).

Derin öğrenmede yaygın olarak kullanılan optimizasyon algoritmaları birinci derece metotları (First Order Methods – FOM) ve ikinci derece metotları (Second order methods – SOM) şeklinde iki başlık altında incelenmektedir. Birinci derece metotları gradyan tabanlı algoritmalardır. Bunlar gradyan iniş algoritması, momentumlu gradyan iniş algoritması, Nestorov momentumlu gradyan iniş algoritması, Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam, AMSGrad şeklinde sıralanabilir. İkinci derece metodları ise Newton, Quasi-Newton (BFGS), Eşlenik gradyan, Hessian-free (Hesse Serbest), Levenberg – Marquardt algoritmalarıdır (Toussaint, 2018).

Jacobian matrisi vektörlerin birinci dereceden kısmi türevlerini içeren bir m × n matrisidir. Hessian matrisi ise fonksiyonun ikinci dereceden kısmi türevlerinin kare matrisidir. Birinci derece metotlarında Jacobian matrisi kullanılmaktadır. İkinci derece metotlarında Hessian matrisi kullanılmaktadır (Patterson ve Gibson, 2017: 33).

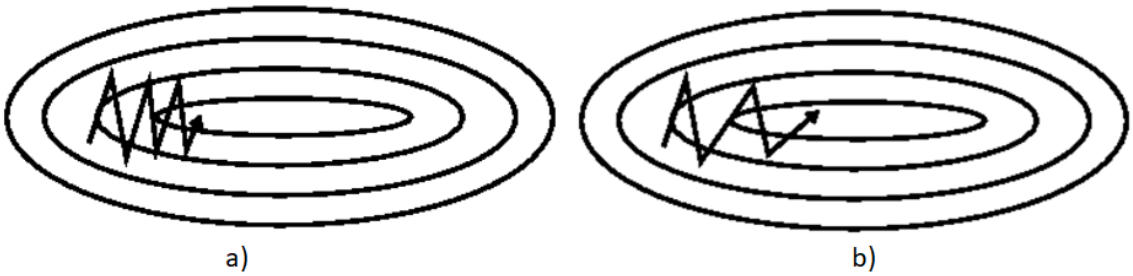
2.6. Birinci Derece Optimizasyon Algoritmaları

Birinci derece optimizasyon algoritmaları gradyan tabanlı algoritmalardır. Yığın gradyan iniş algoritması, mini yığın gradyan iniş algoritması ve stokastik gradyan iniş algoritmalarının her birinin dezavantajı bulunması literatürde birçok farklı algoritmanın geliştirilmesine neden olmuştur. Bu algoritmalar tek olumsuzluğa çözüm getirirken, bazıları ise birden fazla olumsuzluğu ortadan kaldırmaktadır (Ruder, 2016).

2.6.1. Momentumlu Gradyan İniş Algoritması

Stokastik gradyan iniş algoritmasında bir boyut için hesaplanan değer diğerine göre daha dik olabilir. Bu nedenle ilk döngüde bir parametre için hesaplanan iniş daha dik olup, ikinci döngüde diğer parametre için hesaplanan iniş daha dik

olabilir. Bu durumda gradyan iniş algoritması bir parametreden diğer parametreye sürekli salınım göstererek ilerleyeceği için minimum noktasına ulaşması yavaşlayacaktır. Momentumlu stokastik gradyan iniş algoritmasında, bu salınım giderildiğinden, algoritma hızlanmaktadır. Şekil 2.30'da bu salınım etkisi gösterilmiştir (Ruder, 2016).



Şekil 2.30 a) Stokastik Gradyan ve b) Momentumlu Stokastik Gradyan (Orr, 2006).

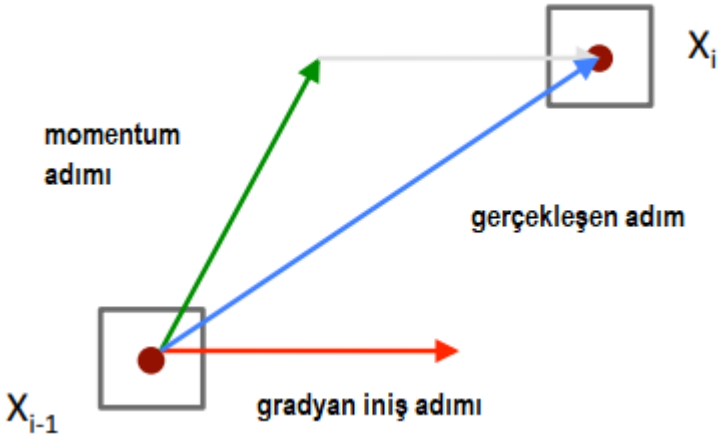
Momentum, bir önceki ağırlığın bir kısmını mevcut ağırlığa eklemektedir. Gradyan inişi aynı yöne doğru devam ettiğinde minimum noktaya doğru atılan adımların boyutunu arttıracaktır. Algoritma momentum içerdiği durumda öğrenme katsayısı azaltılmalıdır, çünkü yüksek öğrenme katsayısı ile momentum bir arada kullanıldığında minimuma büyük adımlarla koşarken algoritmanın minimumu ıskalaması muhtemeldir. Momentumlu gradyan iniş algoritmasının matematiksel gösterimi (2.27) ve (2.28)'de verilmiştir (Orr, 2006).

$$v_t = \gamma v_{t-1} + \alpha \nabla_w J(w) \qquad (2.27)$$

$$w = w - v_t \qquad (2.28)$$

Momentumlu gradyan iniş algoritmasındaki γ parametresi genellikle [0,5;0,9;0,95;0,99] gibi değerlere ayarlanır. Genellikle uygulana yaklaşım 0,5'lik bir momentumla başlamak ve sonraki döngülerde 0,99'a kadar arttırmaktır (Karpathy, 2018).

Gradyan inişi için dağdan aşağı doğru inen topun inişi benzetmesinde momentum sağdan ve soldan topa uygulanan hava direncine benzetilmektir. Bu rüzgar direnci topu ana yörüngesinden sağa sola gereksiz hareketler yapmasını engelleyerek aşağı doğru daha hızlı inmesini sağlamaktadır. Şekil 2.32'de görüldüğü gibi güncellen ağırlık vektörü momentum vektörü ile gradyan iniş vektörü arasında yer almaktadır (Ruder, 2016).

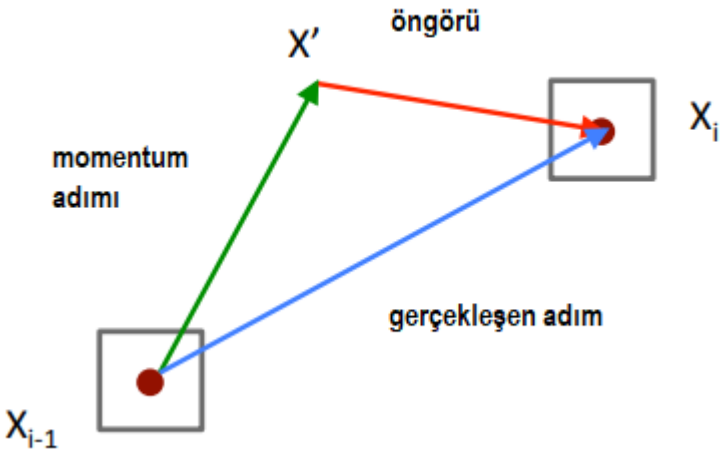


Şekil 2.31 Momentum Güncellemesi (Vansteenkiste, 2016)

2.6.2. Nesterov Momentumlu Gradyan İniş Algoritması

Nesterov Momentumlu Gradyan iniş algoritmasındaki amaç her adımda fazladan yapılan hesaplamaları azaltarak algoritmayı hızlandırmaktır (Nesterov, 1983).

Gradyan inişi için dağdan aşağı doğru inen topun inişi benzetmesinde Nesterov momentumu topa öngörü yeteneği kazandırarak daha sonraki adımı tahmin ederek ilerlemesini sağlamaktadır. Şekil 2.32'de görüldüğü gibi güncellen ağırlık vektörü momentum vektörü kullanılarak bir sonraki adım için öngörü oluşturtmaktadır (Ruder, 2016).



Şekil 2.32 Nesterov Momentum Güncellemesi (Vansteenkiste, 2016)

Nesterov momentumlu gradyan iniş algoritmasının olağan momentumdan farkı gradyan inişi hesaplanmadan önce ağılık vektörünün momentum vektörünü dikkate almasıdır. Eşitlik (2.29) ve (2.30) verilen Nesterov momentumlu gradyan iniş

algoritmasının matematiksel ifadesinden de görüldüğü gibi maliyet fonksiyonu $\nabla_w J$ bir önceki momentum dikkate alınarak hesaplanmaktadır. Yani ağırlık vektörü w henüz güncellenmeden önce maliyet fonksiyonu etkilenmektedir. Bu da bir sonraki adıma bir öngörü katmaktadır (Vansteenkiste, 2016).

$$v_t = \gamma v_{t-1} + \alpha \nabla_w J\,(w - \gamma v_{t-1}) \qquad (2.29)$$

$$w = w - v_t \qquad (2.30)$$

2.6.3. ADAGRAD Algoritması

ADAGRAD kısaltmasının açılımı İngilizce'de "ADAptive GRADient algorithm" şeklinde olup "uyarlanmış gradyan algoritması" olarak tercüme edilmektedir. ADAGRAD algoritması gradyanın önceki adımlarını dikkate alarak gradyan inişinin nasıl gerçekleştiğini analiz etmekte ve öğrenme hızı ona göre ayarlamaktadır. Yani daha önceki iterasyonlarda gözlemlenen verilerin geometrisini dinamik olarak birleştirerek bilgilendirici gradyan temelli öğrenme gerçekleştirmektedir (Duchi vd., 2011).

ADAGRAD algoritması daha önceki iterasyonlarda gözlemlenen verileri analizi sırasında çok rastlanan örnekler için daha düşük öğrenme katsayısı uygularken, az rastlanan örnekleri için daha yüksek öğrenme katsayısı uygulamaktadır. Bu durum her bir boyuttaki ilerlemenin zaman içinde ortaya çıkması açısından yararlı bir özelliktir. Ayrıca her katmandaki gradyanların ölçeği farklı olmasına rağmen optimal öğrenme oranı hesaba katılmalıdır. ADAGRAD algoritmasının avantajı, öğrenme katsayısını kendisi ayarladığından, başlangıç değerininin doğru ayarlama probleminin ortadan kalmasıdır (Zeiler, 2012).

$$g_{t,i} = \nabla_w J(w_{t,i}) \qquad (2.31)$$

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{G_{t,i}+\epsilon}} \odot g_{t,i} \qquad (2.32)$$

Her adımda farklı öğrenme katsayısı uygulanması gerektiğinden, öğrenme katsayısı ilgili adımın eğimi olan $g_{t,i}$ parametresiyle çarpılmaktadır. $g_{t,i}$ parametresi amaç fonksiyonunun ilgili adımdaki w_i ağırlığının kısmı türevidir. ADAGRAD algoritmasının matematiksel ifade (2.31) ve (2.32)'de verilmiştir. Burada G_t geçmiş gradyanların karelerinin toplamıdır. $G_{t,i}$ bir matris olduğundan $g_{t,i}$ vektörü ile vektörel çarpma işlemi yapılmaktadır (Duchi vd., 2011).

2.6.4. ADADELTA Algoritması

ADAGRAD algoritmasının daha gelişmiş bir uzantısı olan ADADELTA algoritması, önceki gradyanların tümünü toplamak yerine, öğrenme katsayısını "kayan pencere" mantığına ayarlamaktadır. ADADELTA algoritması birçok güncelleme yapıldığında bile öğrenmeye devam etmektedir (Usage of optimizers, 2018).

ADAGRAD algoritmasında gradyan büyüklükleri dikkate alındığından, başlangıç koşullarına duyarlıdır. Başlangıçtaki gradyanlar büyükse, eğitimin kalan kısmı için öğrenme oranları düşük olacaktır. Ayrıca, paydadaki kare gradyanların sürekli birikimi nedeniyle, öğrenme oranı eğitim boyunca düşmeye devam edecek ve sonunda sıfıra inerek eğitimi tamamen durduracaktır. ADADELTA algoritması bu iki soruna çözüm getirmek için geliştirilmiştir (Zeiler, 2012).

ADADELTA algoritmasında önceki gradyanların tümünü ($G_{t,i}$) toplamak yerine, geçmişten sadece sınırlı bir kesit (pencere) alınarak paydanın büyümesine çözüm getirmektedir. Bu sayede birçok güncelleme yapılsa da öğrenme ilerlemeyi sürdürmektedir. Geçmişten alınan kesit için gradyanların karelerinin toplamı verimsiz olduğundan, ADADELTA algoritmasında kare gradyanların bozulan ortalaması $E[g^2]$ kullanılmaktadır. Eşitlik (2.33)'te γ katsayısı momentuma benzer şekilde belirli oranda mevcut gradyanı dikkate alırken, belirli oranda ise önceki gradyanı dikkate almaktadır. Hesaplan $E[g^2]$ kullanılarak ağırlıklar (2.34)'te verildiği şekilde ağırlıklar hesaplanmaktadır (Zeiler, 2012).

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma){g_t}^2 \qquad (2.33)$$

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} * g_t \qquad (2.34)$$

2.6.5. RMSprop Algoritması

RMSprop algoritması Geoff Hinton tarafından önerilen yayınlanmamış bir öğrenme katsayısının uyarlama yöntemidir. ADAGRAD algoritmasındaki azalan öğrenme katsayısı probleminde çözme ihtiyacının sonucunda, aynı zamanda ve bağımsız olarak RMSprop ve ADADELTA algoritmaları geliştirilmiştir. RMSprop algoritmasının matematiksel ifadesi (2.35) ve (2.36)'da verilmiştir. RMSprop algoritmasının ADADELTA algoritmasından tek farkı γ parametresi yerine sabit katsayısı kullanılmasıdır (Ruder, 2016).

$$E[g^2]_t = 0.9 * E[g^2]_{t-1} + 0.1 * {g_t}^2 \qquad (2.35)$$

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} * g_t \qquad (2.36)$$

2.6.6. ADAM Algoritması

ADAM kısaltmasının açılımı İngilizce'de "ADAptive Moment estimation" şeklinde olup "Uyarlanmış Momentum Tahmini" olarak tercüme edilmektedir. ADAM algoritması uyarlanabilir öğrenme katsayısını hesaplayan başka bir yöntemdir. ADAM algoritması momentumlu gradyan inişi ve RMSProp algoritmaların avantajlarını bir araya toplayarak geliştirilmiş bir algoritmadır. Bu algoritma gradyanın üstel hareketli ortalamasını (m_t) ve gradyan vektörünü (v_t) kullanarak daha verimli iniş sağlamaktadır. Momentumlu gradyan iniş algoritmasında olduğu gibi g_t parametresi amaç fonksiyonun ilgili adımdaki w ağırlığının kısmı türevidir ($g_t = \nabla_w J(w_t)$). Gradyanın üstel hareketli ortalaması (m_t) ve gradyan vektörü (v_t) eşitlik (2.37) ve (2.38) kullanılarak hesaplanmaktadır. Burada $\beta_1$ve β_2 hiperparametreleri ADADELTA algoritmasına benzer bir şekilde mevcut gradyan ile önceki gradyan arasındaki verimli dengeyi yakalamak için kullanılmaktadır. Yani bu hiperparametreler hareketli ortalamanın üstel bozulma oranlarını kontrol altında tutmaktadır (Kingma ve Ba, 2015).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \qquad (2.37)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \qquad (2.38)$$

Burada hareketli ortalama birinci momenti (ortalamayı) ve ikinci momenti (merkezsiz vasyansı) tahmin etmek için kullanılmaktadır. Ancak başlangıçta m_t ve v_t parametrelerine 0 değeri verilmesi özellikle ilk adımlarda yanlı tahminlerin yapılmasına yol açmaktadır. Ayrıca $\beta_1$ve β_2 hiperparametrelerin bire yakın olması m_t ve v_t değerlerinde bir değişiklik sağlamayacağı için de benzer sonuçlar doğurmaktadır. Bu nedenle yanlılığı ortadan kaldırmak için (2.39)'da verildiği gibi düzeltme yapılmaktadır. Daha sonra (2.40)'da gösterildiği gibi ağırlıkla güncellenmektedir (Kingma ve Ba, 2015).

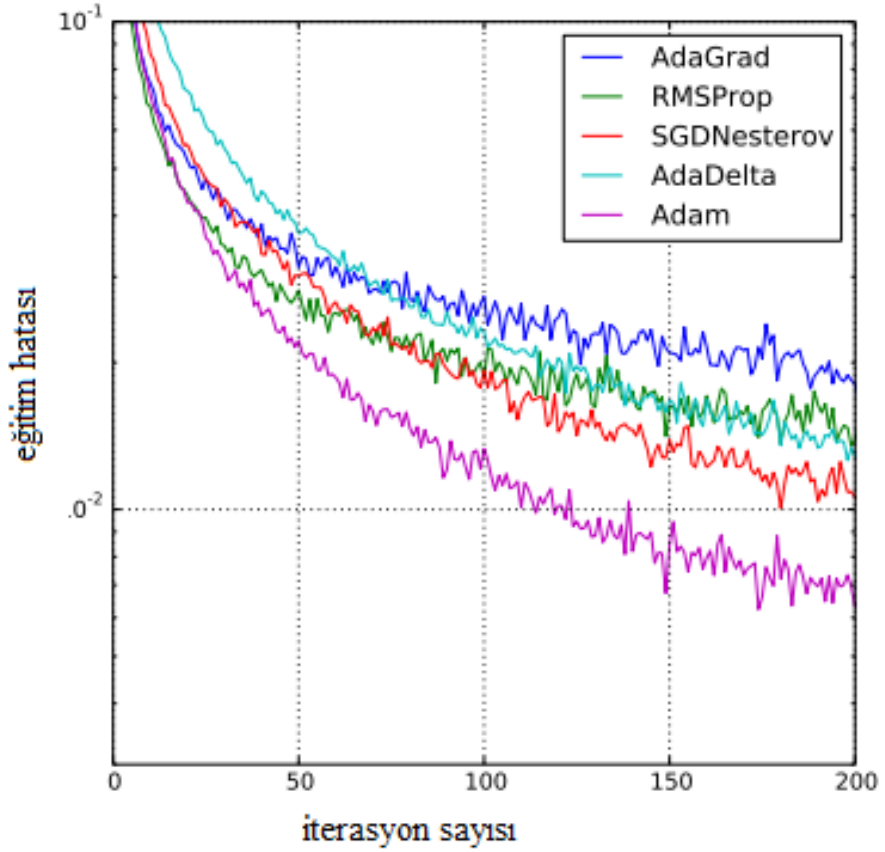
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad (2.39)$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t \qquad (2.40)$$

Gradyan inişi için dağdan aşağı doğru inen topun inişi benzetmesinde momentum hava direncine benzetilirken, ADAM algoritması aynı benzetmede topun

daha ağır olmasını sağlayarak aşağıya daha hızlı inmesini sağlamaktadır (Ruder, 2016).

Kingma ve Ba (2015) çalışmalarında diğer optimizasyon algoritmalarla karşılaştırdığında ADAM algoritmasının daha iyi sonuç verdiğini göstermişlerdir. Şekil 2.33'de görüldüğü gibi ADAM algoritmasın AdaGrad, RMSProp, SGDNesterov, AdaDelta gibi algoritmalarla karşılaştırıldığında maliyet fonksiyonu daha düşük seviyede seyretmektedir (Kingma ve Ba, 2015).



Şekil 2.33 ADAM Algoritmasının Diğer Algoritmalarla Karşılaştırılması (Kingma ve Ba, 2015)

2.6.7. AdaMax Algoritması

ADAM algoritmasında ağırlıklar için güncelleme kuralı L^2 normuna ters orantılı şekilde gerçekleşmektedir (L^2 normu hedef değişken ile tahmin edilen değerler arasındaki mesafelerin karesel toplamıdır). AdaMax algoritmasında ise L^2 norm tabanlı güncelleme kuralı yerine L^p norm tabanlı güncelleme kuralı kullanılmaktadır. L^p kuralına göre gradyan verktörü tekrar yazıldığında (2.41) eşitliği elde edilmektedir (Kingma ve Ba, 2015).

$$v_t = \beta_2^p v_{t-1} + (1 - \beta_2^p)|g_t|^p \qquad (2.41)$$

Bu durumda algoritma büyük p değerleri için sayısal olarak kararsız hale geldiğinden literatürde L^2 ve L^1 kullanımı daha yaygındır. Ancak, p → ∞ şeklinde özel durumda algoritma şaşırtıcı derecede istikrarlı hale gelmektedir. L^∞ kuralına göre gradyan vektörü tekrar yazıldığında (2.42) eşitliği elde edilmektedir (Kingma ve Ba, 2015).

$$v_t = \beta_2^\infty v_{t-1} + (1 - \beta_2^\infty)|g_t|^\infty \qquad (2.42)$$

Gerekli dönüşümler yapmak için u_t parametresi tanımlandıktan sonra (2.43)'de verilen eşitlik elde edilmektedir. Daha sonra u_t parametresi kullanılarak ağırlıklar (2.44)'te verildiği şekilde güncellenmektedir (Kingma ve Ba, 2015).

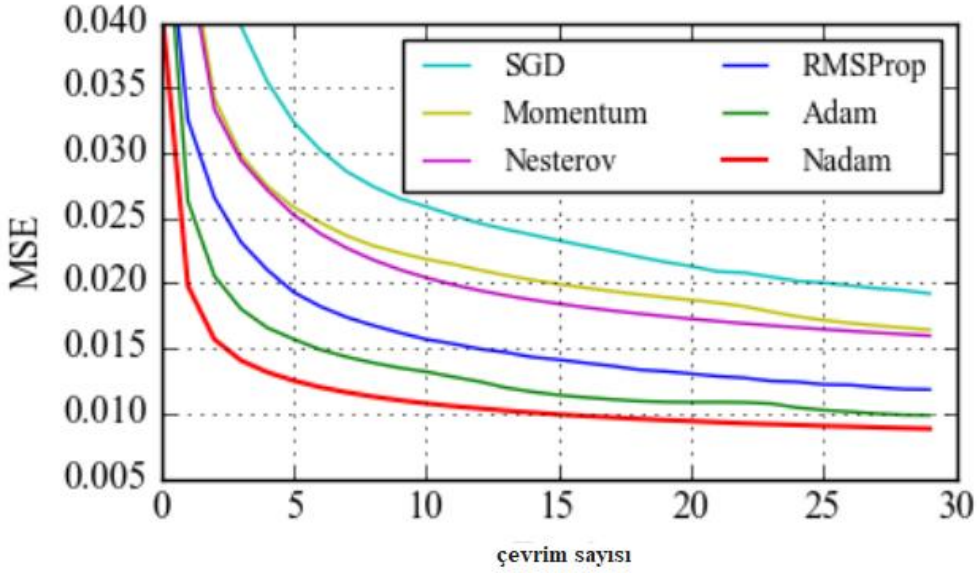
$$u_t = \lim_{p\to\infty}(v_t)^{\frac{1}{p}} = \max(\beta_2 * u_{t-1}, |g_t|) \qquad (2.43)$$

$$w_{t+1} = w_t - \frac{\alpha}{u_t} * \hat{m}_t \qquad (2.44)$$

2.6.8. NADAM Algoritması

NADAM kısaltmasının açılımı İngilizcede "Nesterov-accelerated ADAM" şeklinde olup "Nesterov ile hızlandırılmış ADAM" algoritması olarak tercüme edilmektedir. NADAM algoritması ise Nesterov momentumlu gradyan inişi ve RMSProp algoritmalarını bir araya getirerek geliştirilmiştir. Nesterov momentumlu gradyan iniş algoritması momentumlu gradyan iniş algoritmasından daha iyi performans gösterdiğinden ADAM algoritmasını geliştirmek için Nesterov momentumundan yararlanarak NADAM algoritması geliştirilmiştir (Dozat, 2016).

ADAM algoritmasında m_t parametresinin hesaplaması için gereken g_t değeri $g_t = \nabla_w J(w_t)$ ifadesine göre hesaplanmaktadır. NADAM algoritmasında ise Nesterov momentum kullanıldığı için $g_t = \nabla_w J(w_t - \gamma m_{t-1})$ ifadesine göre hesaplanmaktadır. Şekil 2.34'te görüldüğü gibi NADAM algoritmasını stokastik gradyan inişi (SGD), momentumlu gradyan inişi (Momentum), Nesterov, RMSProp, ADAM gibi algoritmalarla karşılaştırıldığında maliyet fonksiyonu daha düşük seviyede seyretmektedir (Dozat, 2016).



Şekil 2.34 NADAM Algoritmasının Diğer Algoritmalarla Karşılaştırılması (Dozat, 2016)

2.6.9. AMSGrad Algoritması

Gradyan iniş algoritmalarında en yaygın olarak kullanılan ADAM algoritması Reddi ve arkadaşları (2018) tarafından incelendiğinde, ADAM algoritmasının bazı durumlarda optimal çözüme yakınsama göstermediğini tespit etmiştir. ADAM algoritmasında m_t ve v_t gibi iki parametreyi kullanarak öğrenme katsayısına etki ederek büyük adımlar atarak veya küçük adımlar atarak gradyan inişi gerçekleştirmektedir. Atılan büyük adımın yanlış olması ve sonraki doğru adımların çok küçük olması durumunda, yanlışı telafi etmeyip yanlış yöne sapmasına neden olmaktadır. Sabit olmayan öğrenme katsayısı kullanılan diğer yöntemlerde asimptotik olarak ortalama hata sıfıra yaklaştığından yanlış adımı atılsa bile sonraki adımlarda hata telafi edilebilmektedir. ADAM algoritmasındaki bu sapmaların etkileri β_2 katsayısı çok büyük olduğunda azalmaktadır. Bu nedenle ADAM algoritması pratikte uygulanırken β_2 büyük seçilmesi önerilmektedir (Reddi vd., 2018).

AMSGrad algoritması, ADAM algoritmanın yapısını koruyarak bazı değişikliklerle yakınsama sorununu ortadan kaldırmaktadır. AMSGrad ve ADAM algoritmaların farkı, bulunulan adımdan önceki adımlarda en yüksek v_t değerini koruması ve gradyanın hareketli ortalamanın normalleşmesinde kullanılmasıdır. Böylece bazı adımlarda öğrenme katsayısı eski halini korumaktadır. Yani AMSGrad algoritması yanlış adım atmaktansa adım atmamayı tercih ederek ADAM algoritmasında karşılaşılan sorunu ortadan kaldırmaktadır. AMSGrad algoritmasının

matematiksel ifadesi (2.45), (2.46), (2.47), (2.48)'de görüldüğü gibi ADAM algoritmasını tüm adımları uygulanmaktadır, tek değişiklik (2.47)'de verilen adımın ilave edilmesidir (Reddi vd., 2018).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (2.45)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2 \quad (2.46)$$

$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t) \quad (2.47)$$

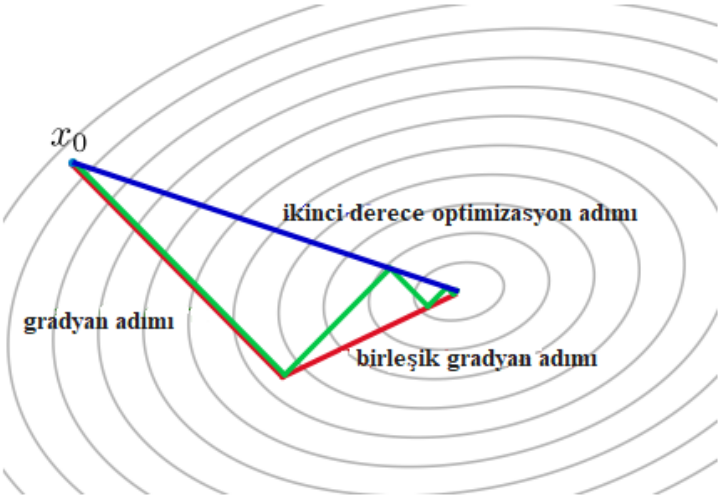
$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} * m_t \quad (2.48)$$

Derin öğrenmede hangi gradyan iniş algoritmalarının kullanılması gerektiğinin kararını vermek için problemi ve veriyi dikkatli incelemek gerekmektedir. Örneğin verinin sık örneklerden oluşmaması durumunda değişken öğrenme katsayılı algoritmalardan tercih edilmelidir. Verinin çok fazla olmadığı ve hesaplama zamanı önemsenmediği durumda yığın gradyan iniş algoritması kullanılmalıdır. Günümüzde ADAM algoritması çok yaygın olarak kullanılmaktadır. AMSGrad algoritması yeni ve daha gelişmiş, birçok soruna çözüm getiren algoritma olmasına rağmen, literatürde çok örnek bulunmadığından etkinliği konusunda bir değerlendirme yapılamamaktadır (Ruder, 2016).

2.7. İkinci Derece Optimizasyon Algoritmaları

İkinci derece yöntemler, eğri bilgisini kullandıklarından stokastik gradyan inişi gibi birinci derece tekniklerinden daha az sayıda adımda minimuma ulaştıkları için derin sinir ağlarının eğitimi konusunda büyük bir potansiyel taşımaktadırlar. Ayrıca ikinci derece optimizasyon yöntemleri nöral ağların eğitiminde daha fazla paralellik kazandırmak için bir fırsat sunmaktadır. Stokastik gradyan inişi algoritmasına kıyasla, ikinci derece optimizasyon metotları ayarlamak için nispeten daha az hiperparametreler kullanılmaktadır (Ramamurthy ve Duffy, 2017).

İkinci derece yöntemler minimuma kestirme adımlarla ulaştıkları için toplamda daha az adım atmaktadırlar. Atılan adımlar iyi sezgisel yaklaşım sergiledikleri için adımları hep hedefe doğrudur. Şekil 2.35'te görüldüğü gibi gradyan tabanlı optimizasyon algoritması minimum noktaya dört adımda ulaşmakta iken ikinci derece optimizasyon algoritması tek adımda ulaşmaktadır (Toussaint, 2018).



Şekil 2.35 İkinci Derece Optimizasyon Algoritma Adımı (Toussaint, 2018)

2.7.1. Newton Optimizasyon Tekniği

Newton yönteminde Hessian matrisi kullanılmakta olup ikinci derece optimizasyon algoritmasıdır. Bu yöntemin amacı kayıp fonksiyonun ikinci türevlerini kullanarak bir sonraki adım için daha iyi yön bulmaktır (Quesada, 2018).

Newton optimizasyon yöntemi, ikinci mertebeden Taylor serisinin genişletmesine dayalı optimizasyon algoritmasıdır. Newton optimizasyon yönteminin matematiksel gösterimi (2.49) ve (2.50)'de verilmiştir (Goodfellow vd., 2016: 311).

$$J(w) \approx J(w_0) + (w - w_0)^T \nabla_w J(w_0) + \frac{1}{2}(w - w_0)^T H(w - w_0) \quad (2.49)$$

$$w = w_0 - H^{-1} \nabla_w J(w_0) \quad (2.50)$$

2.7.2. Levenberg – Marquardt Optimizasyon Algoritması

Newton optimizasyon algoritması yalnızca Hessian matrisinin pozitif olduğu durumlarda uygundur. Derin öğrenmede, amaç fonksiyonunun yüzeyi çoğu zaman dışbükey değildir. Amaç fonksiyonunda eyer noktalarının mevcut olması Newton yöntemi için sorun teşkil etmektedir. Hessian değerleri eyer noktasının yakınında pozitif değilse Newton yönteminin yanlış yönde hareket etmesine neden olmaktadır. Bu durumda Hessian algoritmasına regülasyon (düzeltme) uygulanarak bu sorun önlenmektedir. Yaygın regülasyon stratejileri arasında Hessian diyagonal boyunca bir sabit katsayının eklemesi bulunmaktadır. Bu regülasyon stratejisi literatürde Levenberg – Marquardt optimizasyon algoritması olarak yer almaktadır. Levenberg –

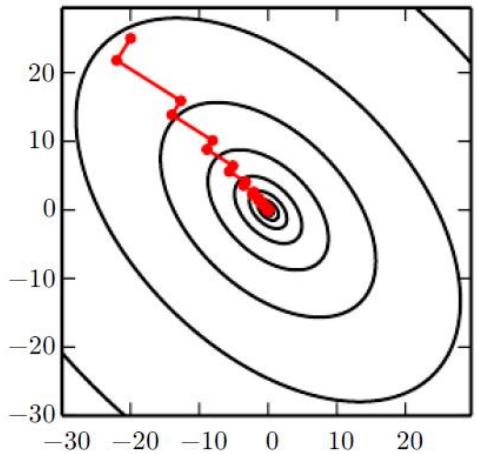
Marquardt optimizasyon yönteminin matematiksel gösterimi (2.51)'de verilmiştir (Goodfellow vd., 2016: 311).

$$w = w_0 - \left[H\left(f(w_0)\right) + \lambda I\right]^{-1} \nabla_{\mathrm{w}} \mathrm{J}(w_0) \qquad (2.51)$$

2.7.3. Eşlenik Gradyan Optimizasyon Algoritması

Eşlenik gradyan algoritması, gradyan iniş algoritması ile Newton algoritması arasında kabul edilen bir algoritmadır. Gradyan inişi algoritmasındaki yavaş yakınsamayı hızlandırmak amacıyla geliştirilmiştir. Aynı zamanda Newton yönteminin gerektirdiği Hessian matrisinin tersinin alınmasına, depolanma ve değerlendirilmesine ilişkin bilgi ihtiyacını ortadan kaldırmaktadır. Eşlenik gradyan algoritmasında minimum arama işlemi eşlenik yönler boyunca gerçekleşerek gradyan iniş algoritmasına göre genellikle daha hızlı yakınsama göstermektedir. Bu minimum arama yönleri Hessian matrisine göre türetilmiştir (Quesada, 2018).

Eşlenik Gradyan algoritması, dik gradyan İnişi algoritmasındaki problemleri gidermek için geliştirilmiştir. Dik gradyan inişi algoritmasındaki atılan her adım sonraki adıma göre dik olmak zorundadır. Örneğin önceki adımın yönü d_{t-1} vektörü ve mevcut adımın yönü d_t vektörü olsun. Minimuma ulaşmak d_t vektörü yine d_{t-1} vektörünü gösterse bile aynı yönü takip edememektedir. Çünkü dik gradyan inişi algoritması ortogonal ilişki temelli bir algoritmadır ve d_t vektörü d_{t-1} vektörüne dik olmak zorundadır. Yani d_{t-1}vektörün yönünü izleyerek minimuma biraz daha yaklaşma seçeneği yerine geriye doğru çekilerek yaklaşmak için sonraki adım beklenmektedir. Bu nedenle dik gradyan İnişi algoritması minimuma zigzag çizerek yaklaşmaktadır. Bu durum Şekil 2.36'da gösterilmiştir (Goodfellow vd., 2016, s. 312).



Şekil 2.36 Dik Gradyan İniş Algoritması (Goodfellow vd., 2016: 312)

Eşlenik gradyan algoritmasında bu zigzag şeklindeki yakınsamanın önüne geçmek için bir önceki adımda işlenen yön ile eşlenik yön bulunmaya çalışılmaktadır. Eşlenik gradyan algoritmasının matematiksel gösterimi (2.52)'de verilmiştir.

$$d_t = \nabla_{\mathrm{w}} \mathrm{J}(w_0) + \beta_{\mathrm{t}} \mathrm{d}_{\mathrm{t}-1} \qquad (2.52)$$

Burada β_{t} ilerleme vektörünü kontrol eden katsayıdır. β_{t} katsayını hesaplamak için Fletcher-Reeves ve Polak-Ribière yaklaşımları kullanılmaktadır. Ayrıca H Hessian matris olup, $\mathrm{d}_{\mathrm{t}}^{\mathrm{T}} \mathrm{H} \mathrm{d}_{\mathrm{t}-1} = 0$ durumundan d_t vektörü ile d_{t-1} vektörü birbiri ile eşlenik vektörlerdir (Goodfellow vd., 2016: 315).

2.7.4. Quasi-Newton Optimizasyon Algoritması

Newton optimizasyon algoritması, Hessian matrisini değerlendirmek ve tersini hesaplamak için birçok operasyon gerektirdiği için uygulanma açısından pahalıdır.

Quasi-Newton algoritması, Hessian matrisini doğrudan hesaplamak ve tersini çevirmek yerine, algoritmanın her yinelemesinde ters Hessian matrisine bir yaklaşık değer (M_T) oluşturmaktadır. Bu yaklaşık değer M_T sadece kayıp fonksiyonunun ilk türevleri kullanılarak hesaplanmaktadır. M_T değerleri hesaplandıktan sonra ağırlıkların güncellemesinde kullanılmaktadır. Bu değerleri hesaplamak için literatürde iki farklı yaklaşım bulunmaktadır: Davidon–Fletcher–Powell formülü (DFP) ve Broyden-Fletcher–Goldfarb-Shanno formülüdür (BFGS). Quasi-Newton algoritmasının matematiksel gösterimi (2.53)'te verilmiştir. Burada ϵ adım büyüklüğünü ifade etmektedir (Quesada, 2018).

$$w = w_0 + \epsilon M_T \nabla_{\mathrm{w}} \mathrm{J}(w_0) \qquad (2.53)$$

2.8. Aşırı Öğrenme ve Regülasyon

Model seçimi sırasında modelin hızlı çalışmasının yanında, model veri yapısı "yeterince" uyum sağlayacak şekilde seçilmelidir. Modelin veri yapısının yetersiz uyum sağlaması ya da aşırı uyum sağlaması tahminleri önemli derecede etkilemektedir. Ağın "yetersiz" ve "aşırı" öğrenme gibi sorunların önüne geçmek için yanlılık ve varyans dengesinin bulunması gerekmektedir (Patterson ve Gibson, 2017: 102).

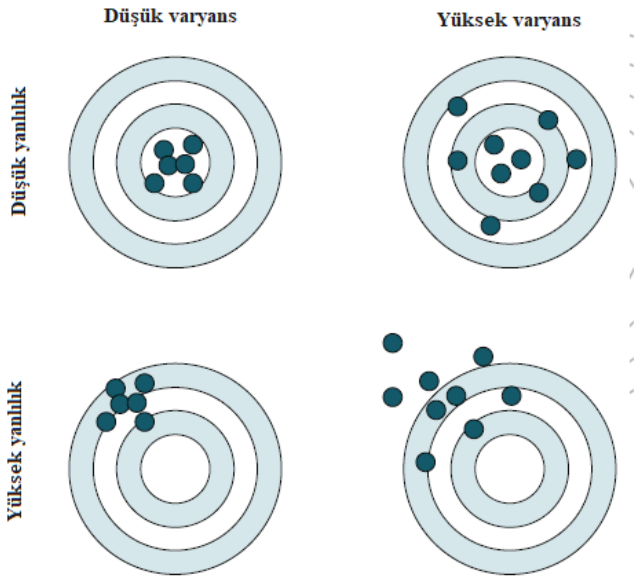
Aşırı Öğrenme sorununu çözmek için daha fazla veri toplayarak model değişikliğine gidilebilir ya da veri toplama imkanı yoksa mevcut eğitim setini veri

çoğaltma (data augmentation) teknikleri ile çoğaltılabilir. Veri arttırmanın dışında seçilen karmaşık model yerine daha basit model seçilebilir, regülasyon uygulanabilir ya da modelin doğrulama kümesi üzerindeki performansı analiz ederek algoritmaya erken durma tekniği uygulanabilir. Aşırı öğrenme sorununa karşı bir diğer çözüm seyreltme tekniği uygulamaktır (Despois, 2018). Bu çözümler alt bölümlerde detaylı şekilde açıklanacaktır.

2.8.1. Yanlılık ve Varyans Dengesi

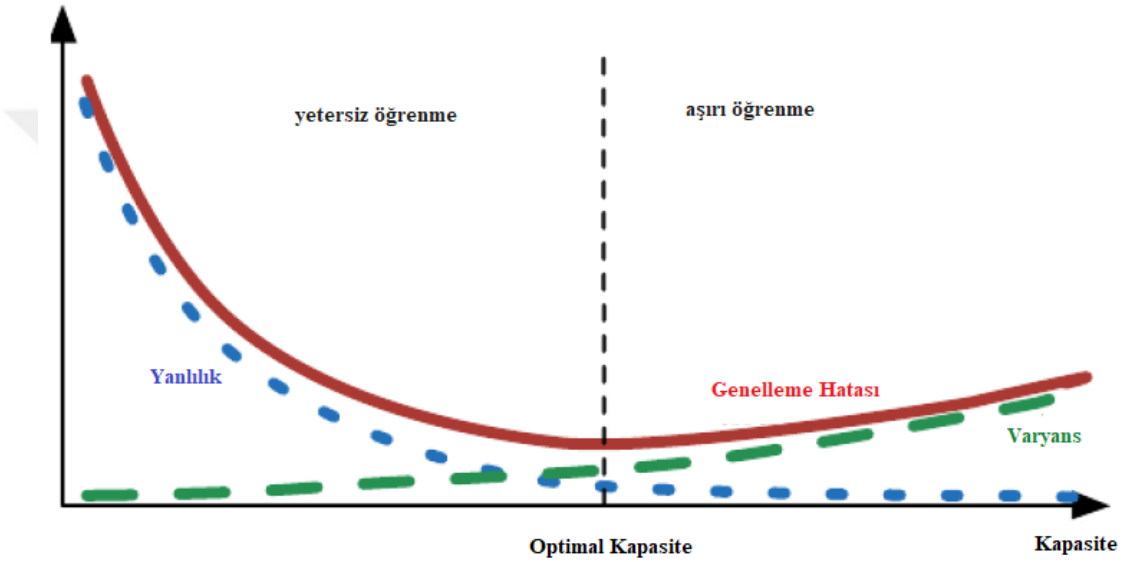
Yanlılık (Bias) ve varyans, bir tahmin edicideki iki farklı hata kaynağını ölçmektedir. Yanlılık, tahmin edicinin ürettiği değerlerin gerçek değerinden sapmayı, varyans ise herhangi bir veri kümesinin neden olabileceği tahmin edicinin ürettiği değerlerin gerçek değerinden sapmayı ölçmektedir (Goodfellow vd., 2016: 130).

Şekil 2.37'de verilen hedef tahtası yanlılık ve varyans arasındaki ilişkiyi daha net açıklamaktadır. İlk satırda yanlılığın düşük olması nedeniyle noktalar hedef tahtayı isabet etmiştir. İkinci satırda ise yanlılığın yüksek olması nedeniyle hedef tahtayı isabet etmeyen noktalar mevcut. Aynı şekilde ilk kolonda varyansın daha düşük olması nedeniyle noktalar birbirine daha yakın, ikinci kolonda ise daha dağınıktır. En doğru tahminleri üst sol köşedeki hedef tahtası gibidir. Hem tüm noktalar hedefi isabet edecek hem de hedef tahtasının merkezine yakın olacaktır. Ne yazık ki yanlılık ve varyans bir biriyle ters orantılı olarak değişmektedir (Michelucci, 2018: 98).



Şekil 2.37 Yanlılık ve Varyans Gösterimi (Michelucci, 2018)

Yanlılık ve varyans örneklem parametre sayısına göre değişim göstermektedir. Parametre sayısı arttıkça daha doğru değerler tahmin edileceği için yanlılık azalmaktadır. Aynı zamanda parametre sayısı arttıkça veri kümesinden kaynaklı hatalar olabileceği için varyans artmaktadır. Şekil 2.38'de ortalama karesel hata fonksiyon grafiğin kapasite arttıkça yanlılık eğrisinin azaldığını, varyans eğrisinin arttığı gösterilmektedir. Yanlılık ve varyansın dengede olduğu durum optimal kapasiteyi temsil etmektedir. Optimum kapasiteyi arama süreci "Yanlılık ve Varyans Pazarlığı" (Bias and Variance Trade Off) olarak literatürde yer almaktadır. (Goodfellow vd., 2016: 130).



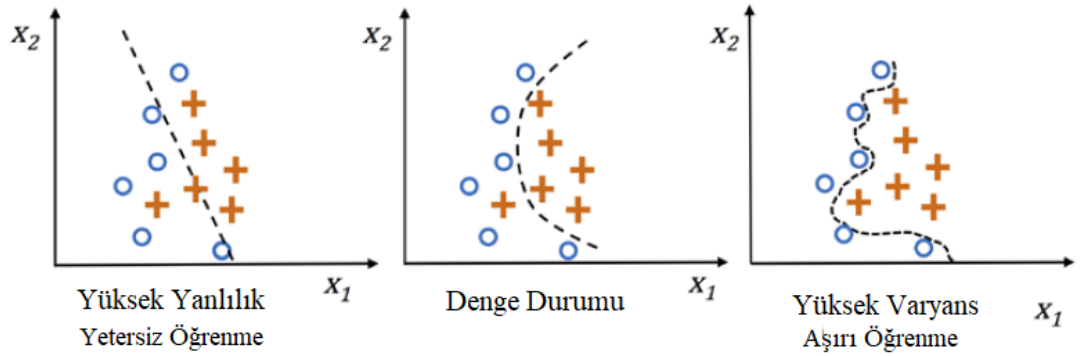
Şekil 2.38 Kapasiteye Göre Yanlılık ve Vasyans Dengesi (Goodfellow vd., 2016: 130)

Yanlılık ve varyans arasındaki ilişki makine öğrenim literatürüne "yetersiz öğrenme ve aşırı öğrenme" kavramı olarak girmiştir. Şekil 2.38'de optimal kapasite çizgisinin sol tarafı "yetersiz öğrenme" bölgesi, sağ tarafı ise aşırı öğrenme bölgesidir (Goodfellow vd., 2016: 130).

2.8.2. Yetersiz Öğrenme ve Aşırı Öğrenme

Aşırı öğrenmede, bir model eğitim verileri üzerinde iyi performans gösterirken, daha önce hiç görmediği veriler (test verileri) için genelleme yapamamaktadır. Bir modelde aşırı öğrenme varsa modelin yüksek bir varyansa sahip olduğunu gösterir şeklinde yorum yapılmaktadır. Aşırı varyans durumunda model çok fazla parametre içerdiğinden karmaşıklaşarak farklı veriye uyum sağlayamamaktadır. Benzer şekilde, yüksek yanlılık modelin yetersiz öğrenmesine neden olmaktadır. Yetersiz öğrenmede, modelin eğitim verilerindeki detayları

yakalamak için yeterince karmaşık olmadığından ağ görmediği verilerde düşük performans göstermektedir (Raschka ve Mirjalili, 2017: 73).



Şekil 2.39 Yetersiz Öğrenme ve Aşırı Öğrenme (Raschka ve Mirjalili, 2017: 73).

Şekil 2.39'daki veri kümesi için doğrusal model kullanıldığında noktaları iki gruba ayırmak yetersiz kalmaktadır. Bu nedenle modelin hiç görmediği veriler için tahminler yaptığında hata oranı yüksek olacaktır. Bu durum ağın yetersiz öğrenme gerçekleştirdiğini göstermektedir. Yüksek varyansın olduğu son resimde ise grupları ayıran çizgi eğitim verisinde tüm detayları dikkate alarak çok fazla uyum sağladığı görülmektedir. Fakat bu model hiç görmediği veriler için tahminler yaptığında hata oranı yine yüksek olacaktır. Bu durum ağın aşırı öğrenme gerçekleştirdiğini göstermektedir (Raschka ve Mirjalili, 2017: 73).

Ağın aşırı öğrenmesini önlemenin ilk yolu ağın eğitimini erken sonlandırmaktır. Yani doğrulama setindeki performansa bakarak eğitimde gerileme yaşandığı anda eğitimi durdurmaktır. Ağın aşırı öğrenmesini önlemenin diğer yolu da regülasyon uygulamaktır (Ganguly, 2017: 25). Erken sonlanma ve regülasyon kavramların açıklamak için öncelikle doğrulama kümesinin açıklanması gerekmektedir.

2.8.3. Eğitim, Doğrulama ve Test Kümeleri

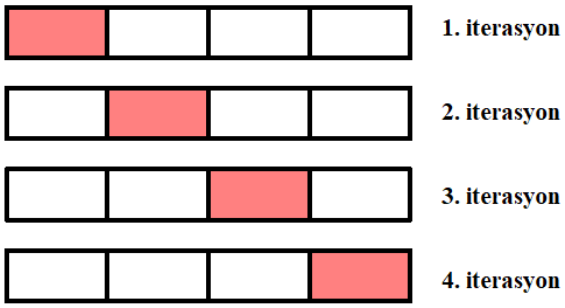
Veri kümesinden alınan ve hiperparametrelerin seçiminde kullanılan verilerin alt kümesi, doğrulama kümesi olarak adlandırılmaktadır. Bir ağın eğitim aşaması tamamlandıktan sonra modelin kalitesini ölçmek için daha önce hiç görmediği örneklerden oluşan test veri kümesi kullanılmaktadır. Fakat hiperparametrelerin doğru değerlerin seçilmesi için bir doğrulama (onaylama) kümesine ihtiyaç duyulmaktadır. Eğer doğrulama için test verisinden birkaç örnekler kullanılırsa, ağ test aşamasında doğru sonuçlar vermeyecektir. Bu nedenle doğrulama ve test kümesi ortak örnek bulundurmamalıdır. Veri kümesinin öncelikle eğitim kümesi %70, test

kümesi %30 olacak şekilde bölünmesi tavsiye edilmektedir. Daha sonra eğitim algoritması uygulanırken, eğitim kümesinin bir kısmı eğitim için, bir kısmı doğrulama için kullanılmaktadır. Veri kümesinin büyüklüğüne bağlı olarak %70 eğitim, %30 doğrulama ya da %80 eğitim, %20 doğrulama için kullanılabilir (Heaton, 2015: 257).

2.8.4. Çapraz Doğrulama

Verilerin rasgele eğitim, doğrulama ve test şeklinde alt kümelerine ayrılarak, ağın performans ölçümlerinde doğrulama kümesi kullanılarak yapılması gizli çapraz doğrulama (Holdout Cross-Validation) olarak adlandırılmaktadır. Bu yöntemin dezavantajı bölümlenme şekline hassasiyet göstermesidir. Yani alt kümelere bölünme işlemi dengeli yapılmaz ise performans ölçümleri kullanışsız olacaktır (Raschka ve Mirjalili, 2017: 189). Bazen de veri kümesi sınırlı olduğunda, eğitim, doğrulama ve test kümelerine bölmek için yeterli olmadığından çok küçük kümeler oluşmaktadır. Küçük doğrulama ve test kümeleri ise ağın düşük performans göstermesine neden olacaktır. Bu duruma çözüm getirmek için K-katlı çapraz doğrulama (K-fold Cross-Validation) tekniği kullanılmaktadır (Bishop, 2006: 32).

K-katlı çapraz doğrulamada eğitim verisi rastgele *k* sayıda bölümlere ayrılmakta ve her iterasyonda modelin eğitimi için *k - 1* bölümü kullanılırken performans değerlendirmesi için bir kat kullanılmaktadır. Bu prosedür *k* kez tekrarlanarak k sayıda modeller ve performans değerleri elde edilmektedir. Modellerin ortalama performansı bu bağımsız katlara bağlı olarak hesaplandığından verilerinin alt bölümlere ayrılmasına daha az duyarlıdır. Şekil 2.40'da eğitim verisi rastgele dört kümeye bölünerek, her iterasyonda renkli küme doğrulama için diğer 3 küme ise eğitim için kullanılmaktadır. Her iterasyonda kullanılan kümeler farklı örneklere denk geleceğinden gelen performans değerleri daha güvenilir olacaktır (Raschka ve Mirjalili, 2017: 191).



Şekil 2.40 K-Katlı Çapraz Doğrulama (Bishop, 2006: 33)

2.8.5. Veri Çoğaltma Tekniği

Mevcut veri setini arttırmak adına veri toplamak pahalı ve emek isteyen bir süreç olduğu için her zaman başvurulan bir yöntem değildir. Bunun yerine verilerin daha çeşitli görünmesini sağlamaya yarayan veri çoğaltma (Data Augmentation) teknikleri kullanılmaktadır (Despois, 2018).

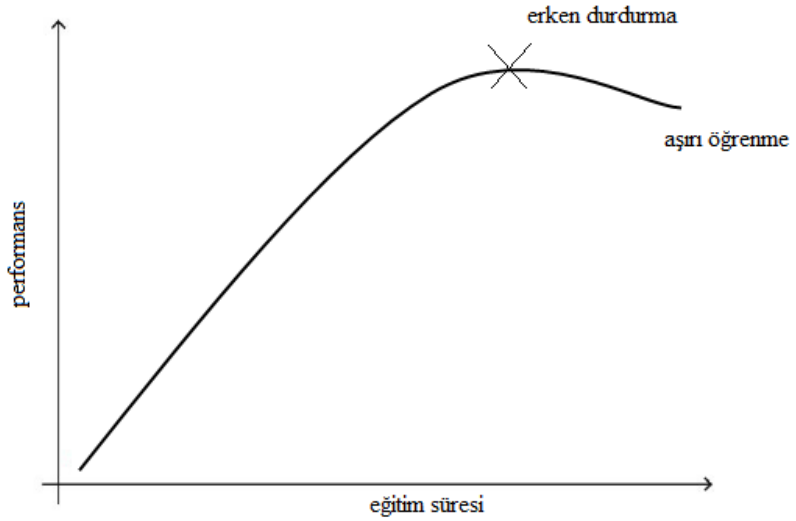
Görüntü işlemede verileri çoğaltmak için mevcut resimlerin simetrisi alınmakta, ters çevrilmekte, ölçeği değiştirilmekte, gürültü eklenmekte, bulanıklaştırılma, resmi kırpma gibi teknikler uygulanmaktadır. Yalnız bu teknikleri uygularken eğitim setine gereksiz veriler ekleyerek veri çöplüğüne dönüştürmekten kaçınılmalıdır. Örneğin kediyi tanıma uygulaması için her resmin pembe ve yeşil versiyonunu eklemek ya da kedi olduğunu çıplak gözle seçemediğimiz görüntüleri eklemek gereksizdir. Çünkü gerçek veride kırmızı kedi ya da paravanın arkasından sadece kulakları görünen kedi olmayacak, ya da olsa bile ağın bunları tanıması beklenmez. Ayrıca ölçek değişikliği yaparken resmin eni ve boyu aynı oranda artırılmalı, uzatma ya da genişletme yapılırsa yarardan daha çok zarar verecektir. Bunun dışında resim çeşitlerine dikkat edilmelidir. Örneğin sadece gözü açık kediler olan küme ile ağ eğitildikten sonra uyuyan kediyi tanıyamaması normaldir. Veri çoğaltma teknikleri uygularken sınıfların çeşitliliğine her sınıf sayısındaki veri miktarına bakılmalıdır. Sadece siyah kedilerin resimlerini çoğaltırsak diğer kedi resimleri sabit kalırsa sınıf dengesizliği olacaktır (Despois, 2018).

2.8.6. Erken Sonlandırma

Genellikle ağın ilk eğitim evrelerinde eğitim hataları gittikçe azalmaktadır, fakat bir noktada ağ eğitim verisini "ezberlemeye" başlayacaktır. Bu durumda eğitim verisi için hata azalmaya devam ederken, test verisi için artmaya başlayacaktır. Yani aşırı öğrenme gerçekleşecektir. Eğitim sürecini izlemek ve test hatası arttığında eğitimi durdurmak yaygın olarak uygulanan stratejidir. Bu strateji erken durdurma olarak adlandırılmaktadır. Şekil 2.41'de erken durdurma grafik üzerinde gösterilmiştir (Albon, 2018: 316).

Erken Sonlandırma için her döngünün sonuna performansı ölçecek hiperparametre eklenmektedir. Bu hiperparametre, ağın performans değerinde bir azalma söz konusu olduğunda döngünün sonlanmasını sağlayacaktır. Ayrıca döngünün sonuna ağın mevcut durumu hakkında bilgi veren bir kontrol noktası gibi

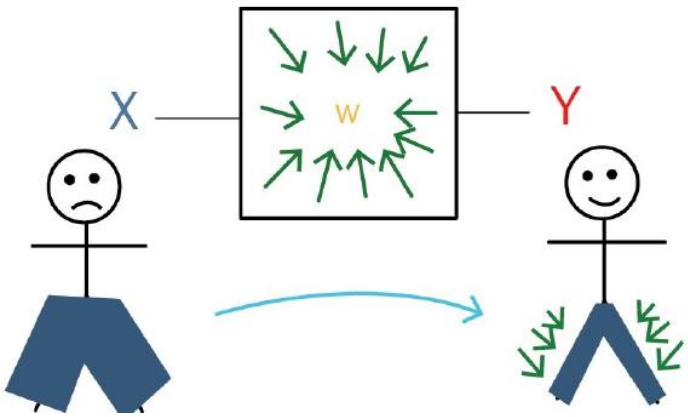
işlev gören bir hiperparametre daha tanımlanabilir. Böylede ağın erken sonlanması durumunda elde edilen en iyi değerle kayıt altına almaktadır (Albon, 2018: 316).



Şekil 2.41 Erken Sonlandırılma (Ganguly, 2017: 26)

2.8.7. L1 ve L2 Regülasyonları

Regülasyon işlemi, ağa yapay kısıtlamalar uygulayarak parametrelerin sayısını dolaylı olarak azaltmaktadır. Regülasyon işlemini daha iyi anlamak için esnek kumaş örneği verilmektedir. Şekil 2.42'de gösteriliği gibi kıyafet esnek kumaştan üretildiğinde farklı ölçülere uyum sağlaması daha kolaydır. Yani kayıp fonksiyona regülasyon uygulandığına yüksek ağırlıklar cezalandırılacağı için varyans azalacaktır ve böylece aşırı öğrenmenin önüne geçilmiş olacaktır (Ganguly, 2017: 25).



Şekil 2.42 Regülasyon Açıklama Görseli(Ganguly, 2017, s. 27)

Aşırı öğrenme problemini çözmek için, bir modelin karmaşıklığı yakalamanın bir yolu bulunmalıdır. Modelin karmaşıklı hakkında bilgi sahibi olunursa modelin veriye uyumu dengede tutulabilir. Bu nedenle mümkün olduğunca basit modeller

seçilmelidir. Model seçimi sırasında, eğer farklı karmaşıklık derecesi olan iki model, kayıp fonksiyonu açısından hemen hemen aynı performansı veriyorsa, basit olan seçilmelidir. Karmaşık modele regülasyon işleminde ceza terimi eklenerek modelin karmaşıklığı azaltılmaktadır. Bu ceza parametresinin önemini kontrol etmek için $\lambda \geq 0$ hiperparametresi kullanılmaktadır (Gulli ve Pal, 2017: 105).

Makine öğrenmesinde kullanılan üç farklı regülasyon türü vardır:

- **L1 Regülasyonu** (Literatürde Lasso olarak da geçmektedir): Modelin karmaşıklığı, ağırlıkların mutlak değerlerinin toplamı olarak ifade edilmektedir. L1 regülasyon terimi (2.54)'te Maliyet fonksiyonuna eklenmiştir.

$$Maliyet = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij} w_j)^2 + \lambda \sum_{j=0}^{M} |w_j| \qquad (2.54)$$

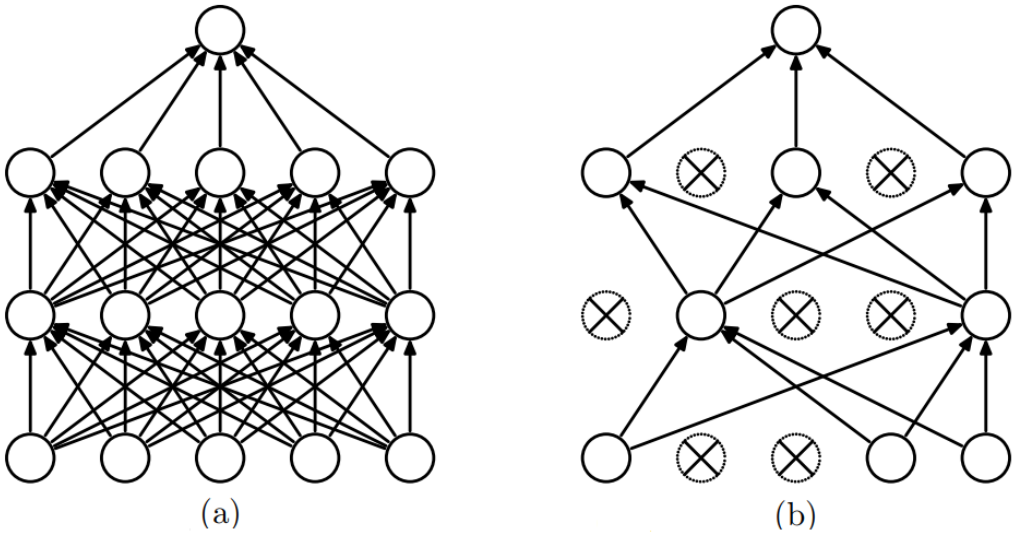
- **L2 Regülasyonu** (Literatürde Ridge olarak da geçmektedir): Modelin karmaşıklığı, ağırlıkların karelerinin toplamı olarak ifade edilmektedir.

$$Maliyet = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij} w_j)^2 + \lambda \sum_{j=0}^{M} w_j^2 \qquad (2.55)$$

- **Elastik net regülasyon**: Modelin karmaşıklığı, önceki iki tekniğin bir kombinasyonu ile yakalanmaya çalışılmaktadır.

2.8.8. Seyreltme

Aşırı öğrenme ile baş etmenin diğer bir yolu da seyreltme tekniği uygulamaktır. Seyreltme (Dropout) tekniği, sinir ağının eğitimi sırasında bazı nöronların devre dışı bırakılmasıdır. Böylece modelin eğitim verilerine çok fazla uyum sağlamasını engellemekte, yani aşırı öğrenmeye çözüm getirmektedir (Lemberger, 2017). Şekil 1.43'te görüldüğü gibi her katmanda belirli sayıda nöron devre dışı bırakıldığı için, bu nöronların diğer nöronlarla bağlantısı da devre dışı kalmaktadır. Hangi nöronların devre dışı bırakılacağı p olasılığı ile rastgele seçildiğinde, her eğitim döngüsü için farklı nöron kombinasyonları kullanılmış olacaktır (Srivastava vd., 2014).



Şekil 2.43 a)Seyreltme Uygulanmadan ve b) Seyreltme Uygulanan Sinir Ağı

Seyreltme tekniğinin amacı gizli katmandaki diğer nöronlar tarafından gölgelenmiş nöronların etkisini ön plana çıkarmaktır. Her seferinde farklı nöronlar aktif olacağı için sinir ağı her seferinde farklı eğitim seti kullanılıyor gibi etki yaratacak ve ağın veriyi ezberlemesini önlenecektir. Verinin kısıtlı olduğu durumlarda oldukça kullanışlı bir tekniktir (Deng ve Yu, 2014: 286).

Seyreltme aynı zamanda karmaşık modelleri daha küçük modeller halinde çalıştırarak sonucu genelleme mantığına dayanmaktadır. Bazen daha iyi sonuçlar alabilmek için küçük ölçekli ağların avantajlarından yararlanılmaktadır. Küçük ölçekli ağları entegre ederek daha genelleştirilmiş bir yapı elde edilmektedir. Böylece elde edilen sonuçlar tesadüfi hatadan daha az etkilenmektedir. Çok sayıda ağı birleştirilerek oluşan yapıda her ağ için ayrı bir döngüye ihtiyaç olacağından çok fazla hesaplama yapılmalıdır. Seyreltme tekniğinde ise bu ağlar sabit olmayıp her döngüde değiştiğinden hesaplamalar tek ağ için yapıldığından daha hızlı sonuç vermektedir (Heaton, 2015: 211).

Seyreltme tekniği bütünleştirilmiş modellerden çok daha hızlı çalışmasına rağmen yine de tüm veri setini kullanılarak uygulandığından eğitim sürecinin yavaşlamasına neden olmaktadır. Bu dezavantajı ortadan kaldırmak veya daha iyi performans elde etmek için hızlı seyreltme, adaptif seyreltme, evrimsel seyreltme, uzaysal seyreltme, iç içe seyreltme, max pooling seyreltme, bağlantı seyreltme (dropconnect) gibi farklı versiyonlar geliştirilmiştir (Rawat ve Wang, 2017).

2.9. Hiperparametreler

Makine öğrenmede, modelin girdileri parametre olarak adlandırılmaktadır. Bir de modele daha iyi ve daha hızlı bir şekilde eğitmek için uygulanan iyileştirme sırasında ayarlanan hiperparametreler bulunmaktadır. Başka bir ifadeyle performansı etkileyebilecek herhangi bir yapılandırma ayarı hiperparametredir (Patterson ve Gibson, 2017: 102).

Birçok derin öğrenme algoritması kendine özgü hiperparametre ile algoritmanın davranışını kontrol etmektedir. Algoritmanın çalışma zamanını ve bellek maliyetini kontrol altına alan bu hiperparametreler modelin kalitesini de, yani model eğitim seti ile ne kadar iyi eğitildiğini ve yeni girdilerle ne oranda doğru sonuçlar verdiğini de etkilemektedir (Goodfellow vd., 2016: 427).

Hiperparametrelerin seçiminde iki temel yaklaşım vardır: manüel seçim ve otomatik seçim. Manüel hiperparametre seçimi, makine öğrenim modellerinin nasıl iyi genelleme sağladığını ve Hiperparametrelerin modeli nasıl etkilediğini anlamayı gerektirmektedir. Otomatik hiperparametre seçimi ise modeli anlama ihtiyacını büyük ölçüde azaltmaktadır, ama çoğu zaman çok fazla hesaplama gerektirdiği için maliyetli olabilmektedir (Goodfellow vd., 2016: 427). Derin öğrenme algoritmaları, hem maliyetin hem de hızın önemli olduğu birçok durumda kullanıldığı için Hiperparametrelerin ve veri setine etkilerinin detaylı öğrenilmesi gerekmektedir.

Hiperparametreler öğrenme katsayısı gibi sadece nümerik değer alan parametreler değildir. Ağın daha iyi çalışmasını sağlayacak her ayar hiperparametre olarak düşünülebilir. “Hiperparametre ayarı” süreci ise ağın daha iyi sonuç vermesini sağlayacak şartları bulma sürecidir. Modellerde ayarlanabilecek parametreler aşağıdaki gibi üç sınıfta incelenebilir (Michelucci, 2018: 275).

- Sürekli değerler alabilen hiperparametreler: öğrenme hızı, regulasyon parametreleri.
- Kesikli fakat teorik olarak sonsuz kabul edilebilen hiperparametreler: gizli katman sayısı, her katmandaki nöron sayısı ve çevrim sayısı.
- Kategorik hiperparametreler: optimizasyon teknikleri, aktivasyon fonksiyonu.

2.9.1. Öğrenme Katsayısı

Yapay sinir ağının kayıp fonksiyonu bir yüzey olarak düşünüldüğünde, ağırlıklar gidebileceğimiz yönü temsil etmektedir. Gradyan iniş mevcut eğimdeki

basamakları temsil ederken öğrenme hızı ise atılan adımın uzunluğu temsil etmektedir. Öğrenme katsayısını seçmek bazı durumlarda zordur. Çünkü daha yüksek öğrenme katsayısı kullanmak her zaman daha fazla şey öğrenmek ya da daha hızlı öğrenmek anlamına gelmemektedir. Bazen bir model öğrenme katsayısı küçüldüğünde daha hızlı öğrenen bir model haline gelmektedir (Ganguly, 2017: 24). Çünkü öğrenme katsayısının çok büyük seçilmesi durumunda yöntemin sürekli minimum etrafında ileri geri sıçrayarak asla minimuma ulaşamaması muhtemeldir. Öğrenme katsayısının çok küçük seçilmesi durumunda algoritma çok yavaş ilerlediğinden, minimum noktayı bulmak için çok fazla iterasyon yapma zorunluluğu doğmaktadır (Michelucci, 2018: 52).

Gradyan iniş algoritmasının matematiksek ifadesi (2.56)'da yer alan α öğrenme katsayısı her döngüde hesaplanan hatanın ağırlıkları ne ölçüde etkileyeceğine karar vermektedir. Öğrenme katsayısı öğrenme sürecinde sabit tutulabildiği gibi her döngüde azalan adaptif öğrenme katsayı da kullanılmaktadır (Bengio, 2012).

$$w = w - \alpha \nabla_{\mathrm{w}} \mathrm{J} \qquad (2.56)$$

Algoritmanın maliyet fonksiyonunun Nan değeri (tanımlanamayan sayı) vermesi öğrenme oranının çok büyük olduğuna işaret etmektedir. Bu tür problemleri kontrol etmenin en iyi yolu, eğitim süreci boyunca maliyet fonksiyonunun değerini düzenli aralıklarla ekranda göstermektir. Bu yol süreci durdurma ve zaman kaybını önleme şansı verecektir (Michelucci, 2018: 52).

2.9.2. Gizli Katman Sayısı

Derin öğrenmede düşük seviyeli özellikleri kullanılarak yüksek seviyeli özellikleri öğrenme amaçlanmaktadır. Örneğin bir görüntü tanıma için pikseller birleşerek kenarlar, kenarlar kullanılarak ilkel objeler, ilkel objeler bir arada değerlendirilerek nesneler tanımlanmaktadır. İnsanlar da kavramları bu tür çoklu seviyeli hiyerarşik yollarla açıklamaktadır. Bu nedenle insan beyninden esinlenerek araştırmacılar daha derin mimarilere yönelerek daha iyi sonuçlar almayı amaçlamışlardır. Fakat 2006 yılına kadar yapılan çalışmalarda katman sayısı arttıkça daha kötü sonuçlar elde edilmiştir (Bengio, 2009: 6). Hinton ve arkadaşlarının 2006 yılında her bir katmanı tek tek eğiten bir öğrenme algoritması geliştirmesi umutsuz olarak sayılan bu alanda bir sıçrama noktası olarak tanımlanmaktadır (Hinton vd.,

2006). Bu gelişmeden sonra birçok araştırmacı bu alana yönelerek daha derin mimariler üzerinde çalışarak daha başarılı sonuçlar elde dilmiştir.

Yapılan çalışmalarda sinir ağlarının katmanlarının sayısı arttıkça, gradyan kaybolması veya gradyan patlaması sorunu oluştuğu ortaya çıkmıştır. Gradyan kaybolması, eğimin sıfıra yakın olduğunda ağırlıkların güncellenmesini engelleme durumudur. Gradyan patlaması ise genellikle başlangıç ağırlıkları çok büyük seçildiğinde meydana geldiği bir durumdur. Buna çözüm olarak ReLU aktivasyon fonksiyonu ve küçük öğrenme oranları kullanılmakta ya da yığın normalizasyonu yapılmaktadır. Aynı zamanda aşırı öğrenme, ağın mimarisi derinleştikçe daha ciddi bir sorun haline geldiği görülmüştür. Aşırı öğrenmeyi önlemek için seyreltme gibi regülasyon tekniklerin kullanılması gerekmektedir (Zhong vd., 2019). Yapılan bu çalışmalar katman sayısı arttıkça derin ağın eğitilmesi zor olduğu gerçeğini vurgulamaktadır. Ağın başarısını arttırmak için sadece katman sayısını arttırmak çözüm değildir, bu katmanlarla başa çıkabilecek yöntemler kullanılması gerekmektedir.

Öte yandan daha derin mimarilere yönelmek yerine daha sığ mimariler kullanarak yakın sonuçlar elde edilebildiği çalışmalar da mevcuttur. Üstelik sığ mimariler çalışma süresi açısından daha kısa sürede sonuç verdiğinden daha az maliyetli olduğundan derin mimarilere ihtiyaç durumu mutlaka sorgulanmalıdır. Mevcut problem için sadece yeterli sayıda katman kullanıldığında başarı ve maliyet kriterleri karşılanmış olacaktır (Ba ve Caruana, 2014).

Gizli katman sayısının her probleme göre özel olarak belirlenmesi gerekmektedir. Probleme özel belirleme yapılırken literatürde mevcut olan bazı yaklaşımların kullanılması önerilmektedir. Bunlardan en basit ve en zahmetli olanı deneme yanılma yoluyla belirlemedir. Farklı sayıda katmanlar belirleyerek denemeler yaptıktan sonra en iyi sonuç veren sayıyı kullanmaktır. Bir diğer yaklaşım ise tecrübelere dayalıdır. Derin öğrenme mimarileri üzerine çalışmalar incelenerek ya da araştırmacının kendi tecrübeleri dikkate alınarak başarı ihtimali yüksek olan birkaç seçenek kurgulanabilir. Deneme işlemi bu seçeneklerle sınırlandırıldığında daha hızlı sonuç elde edilir. Son olarak doğru sayıda katman için bir otomatik arama algoritması kullanılmaktadır. Bu arama algoritması tesadüfi veya sistematik olarak alternatifleri denedikten sonra en iyi sonuç vereni raporlamaktadır. Ya da arama algoritması bir optimizasyon problemi olarak düşünülüp genetik algoritma veya Bayesian optimizasyonu gibi teknikler kullanılmaktadır (Brownlee, 2018).

2.9.3. Gizli Katmandaki Nöron Sayısı

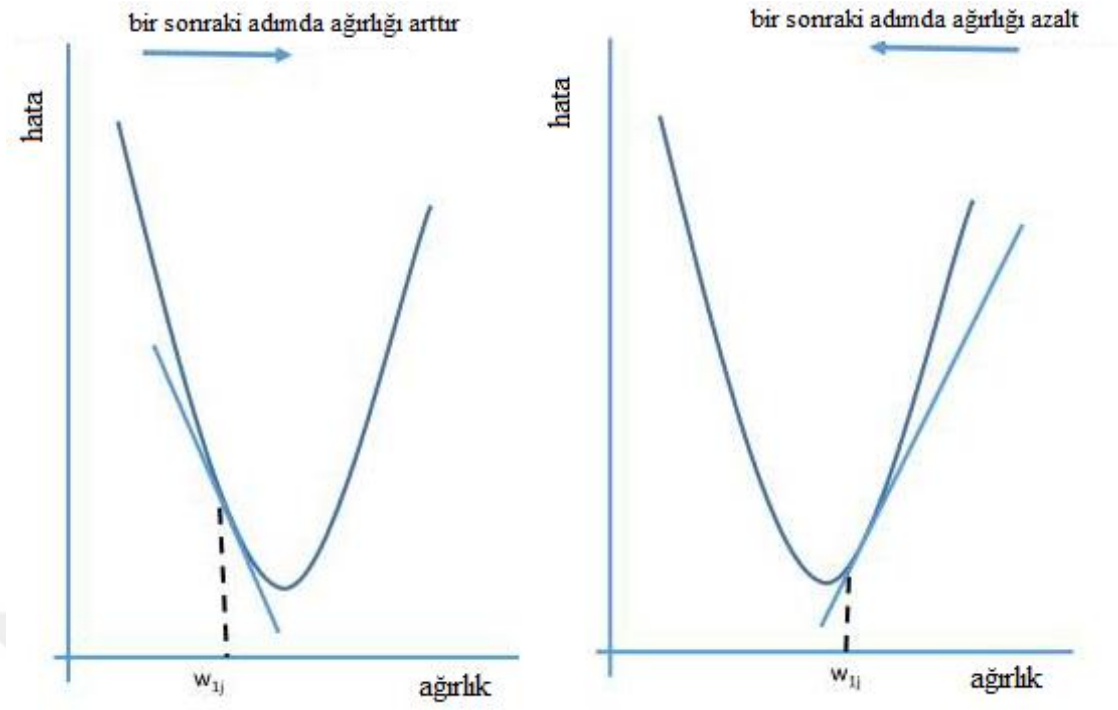
Yapay sinir ağlarında giriş ve çıkış katmanlarındaki nöron sayısına karar vermek oldukça basittir. Giriş katmandaki nöron sayısı giriş vektöründeki değişken sayısına eşleşecektir. Çıkış katmanında ise tek çıkış nöronu ya da tahmin etmeye çalıştığımız sınıf sayısına uyan bir dizi nöron olacaktır. Ancak her gizli katman için nöron sayımlarına karar vermek, hiperparametre ayarının zorlaştığı bir noktadır. Nöron sayının ne kadar büyük veya küçük olabileceğine dair hiçbir kural olmadığından rastgele sayıda nöron kullanarak denemeler yapılmalıdır (Patterson ve Gibson, 2017: 100).

Bir katmandaki nöronlar ne kadar fazlaysa verileri ağ tarafından o kadar net anlaşılabilir. Problem basit ise az sayıda nöron yeterliyken karmaşık problemlerde daha fazla nöron eklemek ağın başarısını arttırmaktadır. Veri girişi ne kadar fazlaysa, ilk gizli katmanda ihtiyaç duyulan nöron sayısı o kadar fazladır. Aynı şekilde ne kadar çok çıktı nöronu varsa, son gizli katmanda ihtiyaç duyulan nöron sayısı o kadar fazladır. Böyle yorumlar yaparak faklı sayıda nöron sayısı deneyerek en başarılı sonuç veren nöron sayısı kullanılmalıdır (Cook, 2016: 192).

Bu durum araştırmacıları başlangıçtan itibaren çok sayıda nöronla başlamaya itmektedir, ancak fazla sayıda kullanılan nöronların bir bedeli vardır. Modele daha fazla parametre eklediğinde ağı eğitmek için gereken çaba artmaktadır. Büyük parametre sayıları uzun eğitim süreleri yakınsamayı bulmakta zorlanan modeller anlamına gelmektedir (Patterson ve Gibson, 2017: 100). Bu nedenle gizli katman sayısı ve gizli katmanlardaki doğru nöron sayısına karar verirken ayrılan süre genellikle ağı eğitmek için ayrılan süreden daha fazladır (Cook, 2016: 192).

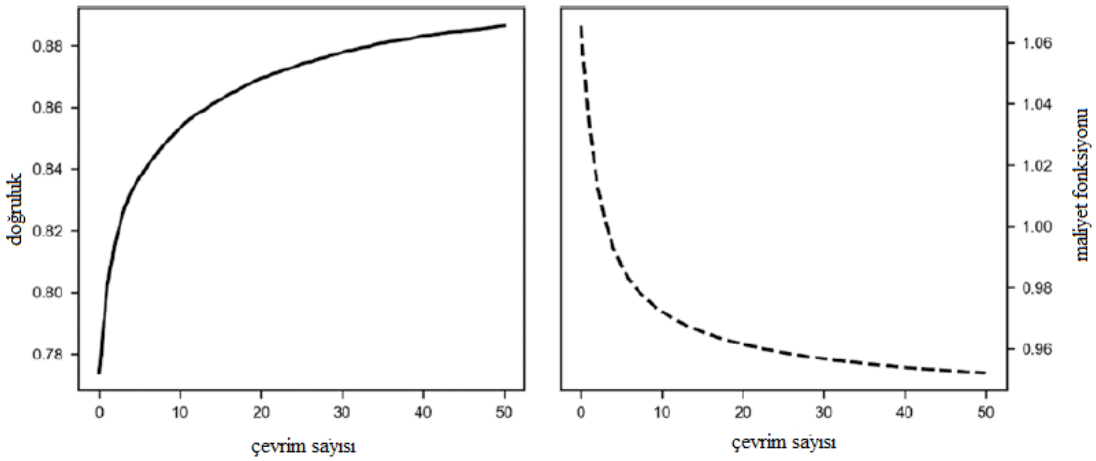
2.9.4. Çevrim Sayısı

Şekil 2.44'te gösterildiği gibi hata fonksiyonun kısmi türev negatifse, ağırlık artar (şeklin sol kısmı); kısmi türev pozitif ise, ağırlık azalır (şeklin sağ kısmı). Bu öğrenme sürecindeki her bir güncellemeye iterasyon denir (Lewis, 2016: 20). Bir iterasyon sırasında veri kümesinin tamamının hesaplamalarda dahil olması halinde aynı zamanda bu güncelleme çevrim olarak adlandırılmaktadır. Başka bir deyişle tüm veri kümesinin başından sonuna kadar bir kez geçmek bir çevrimi tamamlamaktır (Michelucci, 2018: 114).



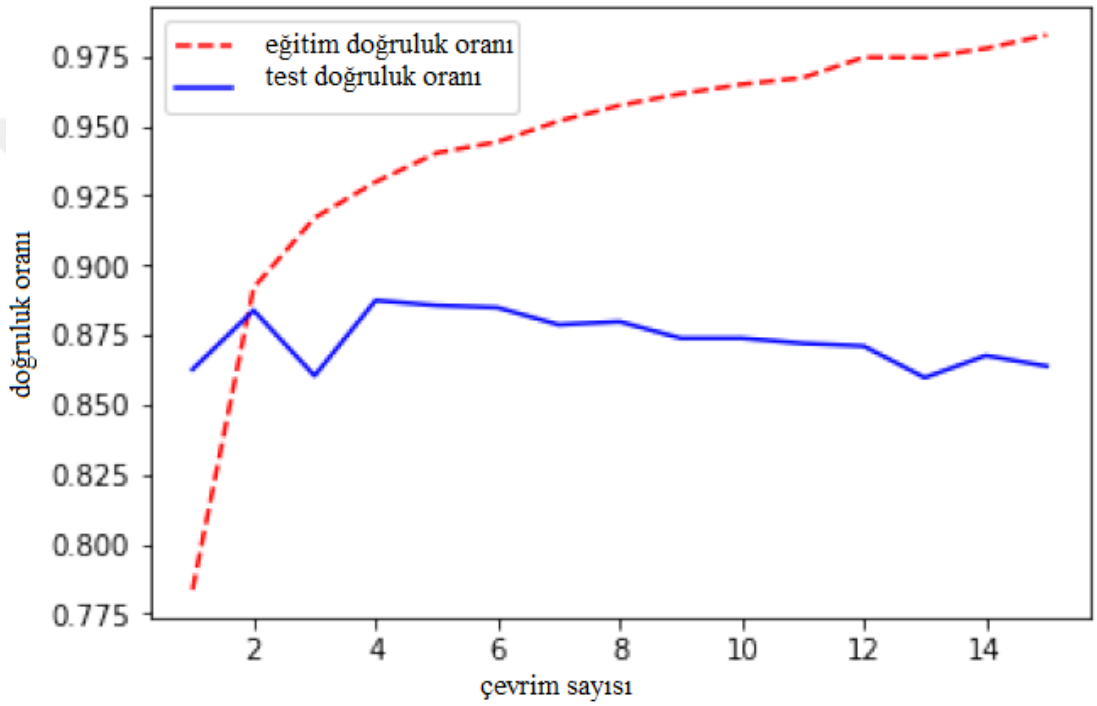
Şekil 2.44 Ağırlıkları Güncelleme (Lewis, 2016: 21).

Veri kümesi çok büyük olduğunda her güncelleme işlemi için tamamı dahil etmek eğitim sürecini yavaşlatacaktır. Bu nedenle büyük veri kümesi daha küçük veri kümelerine bölünmekte ve her güncelleme için bu küçük kümeler sırasıyla kullanılmaktadır. Böylece bir çevrim tamamlandığında bu küçük veri kümeleri sayısı kadar güncelleme ya da iterayon gerçekleştirilmiş olacaktır. Her çevrimde tüm veri setini kapsayacak genellemeye yaklaştıracak güncellemeler yapılacağı için, Şekil 2.45'te görüldüğü gibi her çevrimde tahmin ya da sınıflandırmanın doğruluk oranı artacak, hata fonksiyonun değeri düşecektir. (Michelucci, 2018: 139).



Şekil 2.45 Çevrim Sayısına Göre Doğruluk Oranı ve Hata Değeri (Michelucci, 2018: 184)

Sinir ağı eğitimi devam ederken modelin hem eğitim hem de test setindeki doğruluk oranı artma eğiliminde olacaktır. Bununla birlikte, belirli bir noktada sinir ağı, gerektiğinden fazla uyum sağladığından eğitim verilerini "ezberlemeye" başlar. Bu durumda Şekil 2.46'da de görüldüğü gibi eğitim başarısı artarken, test başarısı azalmaktadır. Eğitim süreci görselleştirildiğinde en optimal çevrim sayısına karar verilmesi kolaylaşacaktır. Yapay sinir ağlarının başarısını belirleyen asıl kriter test aşaması olduğundan test hatasının en düşük olduğu noktada eğitim süreci sonlandırılmalı ya da ağlın ezberlenmesini engelleyen regülasyon teknikleri kullanılması önerilmektedir (Albon, 2018: 313).



Şekil 2.46 Eğitim ve Test Doğruluk Oranı (Albon, 2018: 313)

2.9.5. Ağırlıklara Başlangıç Değerler Atama

Basit bir sinir ağı oluştururken bile daha iyi bir sonuç elde etmek için yapılan ayarlamalar oldukça zahmetlidir. Ancak, bir sinir ağı tasarlarken göz önünde bulundurulması gereken ilk adım, parametrelere başlangıç değerlerin atanmasıdır. Eğer başlangıç değerleri doğru bir şekilde atanırsa en az zamanda iyi sonuçlar verecek şekilde optimizasyonu sağlanacaktır, aksi takdirde gradyan inişini kullanarak bir minim hataya yakınlaşmak imkansız olacaktır (Yadav, 2018). Büyük başlangıç ağırlıkları ara katmanlarda çok büyük bir çıktıya ve aşırı öğrenen bir ağ haline gelmesine neden olmaktadır. Küçük başlangıç ağırlıkları orta katmanlarda çok küçük

bir çıktıya ve hiçbir şey öğrenmeyen bir ağa haline gelmesine neden olmaktadır (Karpathy, 2018).

Tüm ağırlıkların başlangıç değerlere sıfır değeri verildiğinde kayıp fonksiyonuna göre türev her ağırlık için aynı olur, bu nedenle tüm ağırlıklar sonraki iterasyonda aynı değerlere sahiptir. Böylece ağırlıklara başlangıç değeri olarak sıfıra ayarlamak ağı doğrusal bir modele eşdeğer model haline getirir. Bunun yanında yanlılık parametresinin sıfıra ayarlanması, herhangi bir sıkıntı yaratmayacaktır. Ağırlıklara rasgele başlangıç değer atamak, kaybolan gradyanlar veya patlayan gradyanlar gibi iki soruna yol açma ihtimalini taşımaktadır. Bu nedenle bu iki sorunun oluşma ihtimaline karşı sigmoid fonksiyonu yerine RELU aktivasyon fonksiyonu kullanılması önerilmektedir. Ayrıca aktivasyon fonksiyonuna bağlı olarak ağırlıkları başlangıç değerleri atamak için sezgisel teknikler de kullanılmaktadır (Doshi, 2018).

2.9.6. Mini-Yığın Boyutu

Mini-yığın sinir ağı hesaplamasını hızlandırmak için kullanılan yaygın bir yaklaşımdır. Stokastik gradyan iniş algoritmasında olduğu gibi tek tek her örnek için gradyan eğimi hesaplamak yerine tüm küme üzerinde gradyan eğimi hesaplanmaktadır. Yığın büyüklüğü arttıkça, veri kümesi üzerinde her ileri ya da geri geçiş, modelin çalıştırmak için gereken hafızaya ihtiyacı arttıracaktır. Veri kümesinin daha küçük kümelere bölündükten sonra her küçük küme için gradyan hesaplanıp ağırlıklar güncellenmektedir. Böylece veri kümesinin tamamı üzerinde her ileri ya da geri geçiş sırasında mini-yığın sayısı kadar ağırlıklar güncellendiğinden model daha hızlı sonuç vermektedir. Mini-yığın kullanımı derin ağdaki nöron sayısı ve eğitim verisinin büyüklüğü arttıkça büyük yarar sağlamaktadır. Ancak mini-yığınların olması gereken büyüklük modele göre değişiklik gösterdiğinden farkı büyüklükte mini-yığınlar kullanılarak model çalıştırılmalıdır. En iyi performansı olan büyüklük seçilerek kullanılmalıdır (Lewis, 2016: 55).

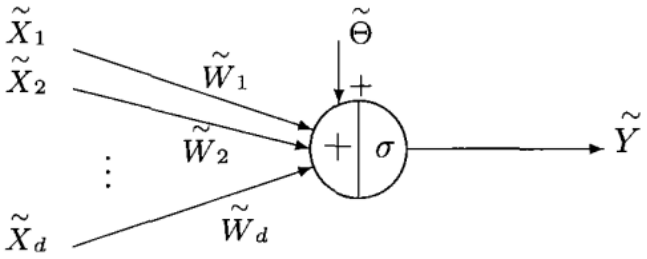
2.10. Bulanık Derin Öğrenme

Bulanık sinir ağlarının mimarisi geleneksel çok katmanlı sinir ağları ile aynı olup giriş sinyalleri, bağlantı ağırlıkları ve yanlılık parametresi bulanık kümelerdir. Bulanık sinir ağlarının işlemler bulanık aritmetik teorisine dayanmaktadır. Geleneksel sinir ağları gibi Bulanık sinir ağları da son zamanlarda sistem

modellemesi, örüntü sınıflandırma, sistem tanımlaması, işlem kontrolü, sinyal işleme gibi birçok alanda başarıyla uygulanmıştır (Liu ve Li, 2004: 131).

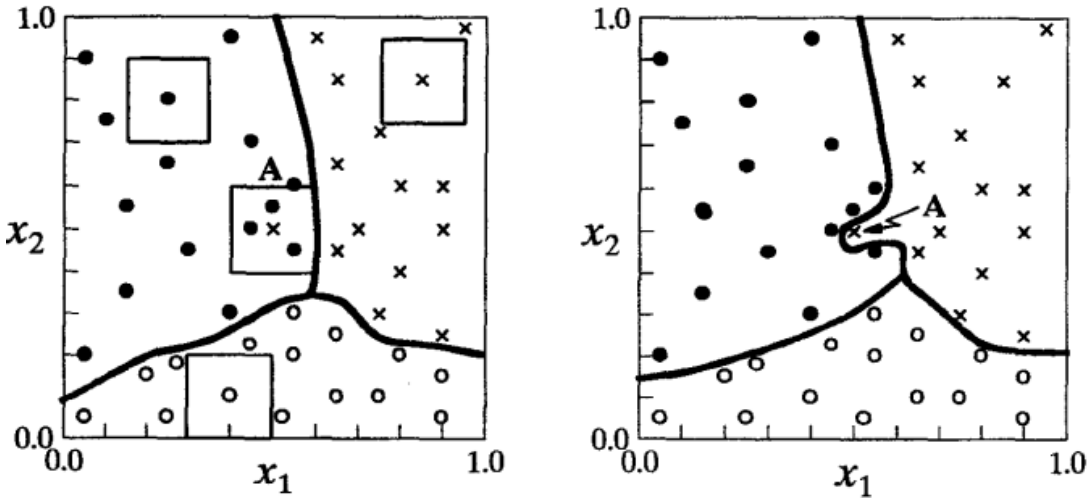
Bulanık sinir ağlarını giriş sinyalleri ve bağlantı ağırlıklarının değerlerine göre üç sınıfa ayrılmaktadır. Bunlardan ilki, girişleri gerçek sayılar ve bağlantı ağırlıkları bulanık kümelerdir. İkincisi, girişleri bulanık kümeler ve bağlantı ağırlıkları gerçek sayılardır. Üçüncüsü, girdileri ve ağırlıkları bulanık kümelerdir (Liu ve Li, 2004: 131).

Bulanık sinir ağı bulanık nöronlardan oluşmaktadır. Bulanık nöron ise standart bir nöronun bulanıklaştırılmasıyla elde edilmektedir. Bulanık nöronun yapısı Şekil 2.47'de gösterilmiştir. Burada x_i bağımsız tam sayılı parametreler yerine bulanık sayıları ifade eden $\tilde{x}_i$ parametresi ve w_i ağırlıkları yerine bulanık sayıları ifade eden $\tilde{w}_i$ bulanık ağırlıklar, θ_i yanlılık parametresi yerine bulanık sayıları ifade eden $\tilde{\theta}_i$ bulanık yanlılık parametresi kullanılmıştır. Üçgensel bulanık girişler $\tilde{x}_i = (\mathrm{a}_i, \mathrm{b}_i, \mathrm{c}_i)$ olarak tanımlanmakta olup a_i sol sınırı değerini, b_i merkez değerini, c_i sağ sınır değerini temsil etmektedir (Liu ve Li, 2004: 133).



Şekil 2.47 Bulanık Yapay Nöron (Liu & Li, 2004, s. 133)

Çok katmanlı sinir ağlarının bulanıklaştırılması girişleri ve ağırlıkların bulanık sayılara genişletilmesiyle mümkündür. Bulanık sinir ağları, sinir ağlarının bulanık kurallardan öğrenilmesi, bulanık girdilerin sınıflandırılması, doğrusal olmayan bulanık sistemlerin modellenmesi ve sinir ağlarından dilsel kural çıkarılması gibi birçok uygulama alanına sahiptir. Bulanıklaştırılmış sinir ağlarının eğitimi için birçok algoritmanın yanında girdi vektörlerinin bulanıklaştırılması, sinir ağlarının genelleme yeteneğini geliştirmektedir. Giriş vektörlerinin bulanıklaştırılması, aynı zamanda sinir ağlarının aşırı öğrenmesinin önüne geçerken yanlış sınıflandırma oranlarını da azaltmaktadır. Şekil 2.48'de verilen örnekte görüldüğü gibi bulak sinir ağı veriyi başarılı bir şekilde sınıflandırırken, standart sinir ağı ise veriyi ezberlemektedir (Ishibuchi ve Nii, 1998).



Şekil 2.48 (a) Bulanık Ağ ve (b) Standart Ağ Sınıflandırma Örneği (Ishibuchi & Nii, 1998).

2.11. Derin Öğrenme ile ilgili Literatür taraması

Derin öğrenme, insan beynindeki bilgi işlem prensiplerinden ilham alan makine öğrenimi araştırma alanıdır. Derin öğrenme, 30 yıl önce tanıtılan derin sinir ağlarının mimarisine dayanmaktadır. Derin sinir ağlarının eğitme zorluğu dolayısıyla bu alandaki gelişmelerin önüne geçmiştir. Ancak 2006 yılında Hinton ve arkadaşlarının derin ağların her katmanın ayrı eğitme fikri bu alandaki araştırmalara hız vermiştir (Hinton vd., 2006).

Derin öğrenme algoritmaların görüntü işleme alanında yoğun hesaplamalardan dolayı ağın yavaş sonuç vermesi ayrı bir zorluk oluşturmaktadır. Bu zorluk Grafik işleme ünitesi (GPU) ile aşılarak, son birkaç yıl içinde çeşitli zorlu problemlere derin öğrenme uygulanmış, çığır açan bir ilerleme kaydedilmiştir (Zhang vd., 2016).

Bu gelişmelerin yanı sıra 1990'lı yıllarda L. A. Zadeh tarafından önerilen "Esnek Hesaplama" (Soft Computing) kavramı derin öğrenmeyi de içine alarak daha anlamlı hale gelmiştir. Esnek bilgi işlem teknikleri, hem mantıksal hem de sezgisel bilgi işlemeyi içeren insan tipi bilgi işleme mantığına dayanmaktadır. Esnek bilgi işlemleri tekniklerinin amacı, genetik algoritmalar, bulanık mantık, yapay sinir ağları gibi teknikleri bir araya getirerek bu teknikleri kombinasyonu ile daha geniş uygulama alanlarında daha iyi sonuçlar elde etmektir (Zhang vd., 2016).

Derin öğrenme, diğer makine öğrenmesi tekniklerinden farklı olarak, programcının tanımlamasına gerek kalmadan ihtiyaç duyulan öznitelikleri (feature) eğitim veri setinden bağımsız olarak öğrenmektedir. Ayrıca diğer makine öğrenmesi

tekniklerinin gereğinden daha büyük eğitim veri seti ve daha yüksek hesaplama gücü gerektirmektedir (Kappor vd., 2018).

Büyük veri çağında, her gün satış işlemleri, sosyal medya, sensörler gibi pek çok kaynaktan üretilen verinin yansıra küçümsenmeyecek boyutta sağlık verisi üretilmektedir. Örneğin, 2015 yılında tipik bir hastane tarafından 665 terabayt veri oluşturulduğu görülmüştür. Bu boyutlardaki veri bir kütüphanenin web arşivinden çok daha büyüktür. Büyük veri akışı sadece iş alanı ile sınırlı olmayıp, arama motorları, e-postalar, sosyal medya ağları, arabaların coğrafi izleme sistemleriyle günlük hayatımızın olmazsa olmaz parçası haline gelerek davranışlarımızı bile etkilemektedir. Bununla birlikte, "büyük veri" ifadesi sadece hacme değil, aynı zamanda hız ve çeşitliliğine göre de değerlendirilmektedir (Wang vd. 2017). Mobil internetin gelişiminde keskin bir yükseliş ile beraber hızlı ve yüksek kalitede veri akışı bir ihtiyaç haline gelmiştir. Akıllı ev, duygu tanıma, sağlık izleme gibi gerçek zamanlı izleme hizmetlerin, kişiselleştirilmiş ve etkileşimli servislerle yüksek veri hızında bu kadar çok kullanıcıya sunulması gerekmektedir (Hossain vd., 2018).

Son yıllarda, bilişim alanında geliştirilen makine öğrenimi teknikleri büyük veriyi hızlı ve doğru işleme konusunda mesafe kat etmiştir. Derin öğrenme yaklaşımı ise nispeten yeni bir makine öğrenme yöntemi olup Google gibi büyük veriyi işleyen bilgi teknolojisi şirketlerinin önemli bileşeni haline gelmiştir (Kaneko ve Yada, 2016).

Büyük veriyi hızlı ve en doğru şekilde işleyen tekniklerden biri olan Derin öğrenme, işletme, yönetim, tıp, sağlık, mühendislik, bilimsel araştırma gibi birçok alanda büyük fayda sağlamaktadır (Wang, Xu, ve Pedrycz, 2017).

Büyük veriyi hızlı ve doğru işleme ihtiyacına cevap veren derin öğrenme tekniklerinin yaygın kullanıldığı alanlardan biri de memnuniyetin tahminidir. Bir işletmenin rekabet üstünlüğü kazanması için üstün müşteri değeri ve memnuniyet sunmak çok önemlidir. İşletmenin hizmet ve ürün kalitesinin geliştirmesinde müşterilerin hizmetleri ve ürünleri detaylı değerlendirmesi büyük katkı sağlarken, karlılık ve müşteri sadakati konusunda genel memnuniyet dikkate alınmaktadır. Literatürde genel memnuniyetin tahmin edilmesinde kullanılan birçok yöntem bağımsız değişkenlerin genel memnuniyeti üzerine etkisini incelemektedir. Ancak genel memnuniyet, müşterinin beklentisini de içinde barındırarak müşterinin genel izlenimi anlamına gelmektedir (Deng ve Pei, 2009).

Birçok işletme, hem hizmetlerini hem de müşteri deneyimlerini değerlendirmek için müşteri memnuniyeti anketleri toplamaktadır. Giderek artan bu veriler, istatistiksel yöntemlerle geleneksel olarak analiz edilmektedir. Ancak belirli bir hipotez test eden bu tekniklerin çoğu zaman veri setine uymayan çoklu varsayımları olduğundan, büyük veri için verimli değildir (Tabrizi vd., 2016).

Girdi değişkenleri ve çıktı değişkenleri arasındaki karmaşık ilişkiyi incelemek için yaygın olarak kullanılan derin öğrenme temelini oluşturan yapay sinir ağları, müşteri memnuniyet analizleri için uygun teknikler arasındadır. Literatürde genel memnuniyetin tahmin edilmesinde yapay sinir ağlarını kullanan birçok çalışma bulunmaktadır. Örneğin Jahandideh ve arkadaşları (2013) çalışmalarında güvenilirlik, sigorta, fiziksel koşullar, empati, duyarlılık gibi faktörleri kullanarak hastaların hastane hizmetleri genel olarak nasıl değerlendirdiklerini tahmin eden yapay sinir ağlarına dayalı bir model önermişleridir (Jahandideh vd., 2013).

Bunun yanında literatürde genel memnuniyetin tahmin edilmesinde geleneksel teknik olan lojistik regresyon modelinin yapay sinir ağları ile karşılaştıran çalışmalar da bulunmaktadır. Örneğin Tsaur, Chiu ve Huang (2002), dokuz uluslararası oteldeki hizmetlerin önem puanlarını ölçmek için yapay sinir ağları ve lojistik regresyon uygulamışlardır. Çalışmalarında yapay sinir ağların lojistik regresyondan daha iyi performans gösterdiği sonucuna varmışlardır (Tsaur vd., 2002).

Tayyar (2010) çalışmasında 364 kişiye uygulanan 7'li Likert ölçeği ilde toplanmış anket verisi kullanarak Hasta Memnuniyetinin tahmininde Yapay Sinir Ağları, Lojistik Regresyon ve Ayırma Analizi tekniklerinin performanslarını karşılaştırmıştır. Çalışmasında en iyi performansın yapay sinir ağları olduğunu göstermiştir (Tayyar, 2010).

Derin öğrenme ise yapay sinir ağları temelleri üzerine kurulan daha derin mimariler kullanarak müşteri memnuniyetin tahminin de önemli avantajlar sunmaktadır. Derin mimari ile beraber kısalan eğitim süresi gerçek zamanlı hizmet sunan çağrı merkezlerinde büyük katkı sağlamaktadır. Örneğin Cong ve arkadaşları (2016) müşteri memnuniyet skorunu ölçmek için akustik ve semantik özellikler, duygu değişkenliği özellikleri, konuşma ritmi gibi parametreleri sözlü değerlendirmelere ilave ederek derin öğrenme teknikleri ile tahmin etmişlerdir (Cong vd., 2016).

Müşteri hizmetleri algıları genel anlamda belirsizlik içermektedir. Dilsel değerlendirmelere dayalı müşteri algılarını temsil etmek için Likert ölçeğini uygulamak bu belirsizliğe çözüm getirmemektedir. İnsan algıları ve tutumları öznel ve belirsizdir. Ayrıca, bireysel algı ve kişilikteki farklılıklar bu belirsizliğe etki etmektedir. Literatürde bu duruma çözüm getirmek için Likert ölçeğini bulanık sayılarla ifade ederek daha başarılı sonuçlar alabildiği çalışmalar bulunmaktadır. Örneğin Deng ve Pei (2008) çalışmalarında yapay sinir ağlarının tekniğini bulanık mantıkla entegre edildiğinde baha başarılı sonuçlar verdiğini göstermişlerdir (Deng ve Pei, 2009).

Müşteri memnuniyeti tahmininde derin öğrenme tekniğini bulanık mantıkla entegre eden bir tekniğe literatürde rastlanmamıştır.

ÜÇÜNCÜ BÖLÜM
BULANIK MANTIK VE DERİN ÖĞRENME ENTEGRASYONU

1932 yılında Rensis Likert tarafından tanıtılan Likert ölçeği, anket araştırmasında en yaygın kullanılan psikometrik ölçek olup, katılımcıların beyan edilen ifadeye katılma seviyelerine dayanmaktadır. Örneğin, 5'li Likert ölçeği için 1-Kesinlikle katılmıyorum, 2-Katılmıyorum, 3-Ne katılmıyorum ne de katılıyorum, 4-Katılıyorum ve 5-Kesinlikle katılıyorum şeklinde seviyeler tanımlanmaktadır. Katılımcılar seviyelerden birini seçerek anket sorularını cevaplamaktadır (Li, 2013).

Likert ölçeğindeki en büyük problemlerden biri, bu ölçeğin sıralı mı yoksa aralıklı mı olduğu tartışmasının halen sürüyor olmasıdır. Her ne kadar Rensis Likert, ölçeği tanımlarken kalite aralıkları olduğunu varsaysa da, birçoğunda Likert ölçeğinin uygulamada sıralı olduğu varsayılmıştır. Diğer bir sorun ise Likert ölçeğinin kapalı cevap formatına sahip olmasıdır. Yani katılımcılar duygularını bu formatla sınırlandırmak zorunda kaldığından duygularını veya tam olarak emin olmadıkları noktaları yansıtamamaktadır. Bu sorunları aşmak için seviyelerin artırılması ya da iki aşamalı Likert ölçeği kullanılması önerilmiştir. Bunun yanında her seviye için daha detaylı ve duyguları yansıtacak cümlelerin kurulması da önerilmiştir. Ancak katılımcılara bu tür anketlerin uygulamasındaki zorluklar araştırmacıları daha pratik yöntemlerin arayışına itmiştir (Li, 2013).

Likert ölçeğini iyeleştirmenin alternatif bir yolu Likert ölçeğine karşılık gelen basit görsel analog skalanın kullanılmasıdır. Likert ölçeği bir çizgi ile temsil edilmekte ve cevaplayanlardan cevaplarını en iyi yansıtan iki noktayı çizgi üzerinde bir çarpı ile işaretlemesi istenmektedir. Bu teknikteki amaç Likert ölçeğindeki kategorilerin tamsayı ile kodlamasından doğan algı farklılıklarının önüne geçmektir. Bu tekniğe benzer bir şekilde Likert ölçeğin kategorilerine bulanık sınırları eklenerek Bulanık Likert Ölçeği geliştirilmiştir (Gil ve González-Rodríguez, 2012)

Bu tez çalışmasının amacı, 5'li Likert tipi ölçeğiyle üretilen 100, 200, 300, 400, 500, 600, 700, 800, 900 ve 1000 örnek içeren yapay veri setlerinin üçgensel ya da yamuk bulanık sayılar kullanılarak bulanık forma dönüştürülmesi ve bu yolla verilerin çoğalması durumunda derin öğrenme tekniklerinin performansının analiz edilmesidir.

Bulanık sayılarla oluşturulan veri setleri ile normal veri setinden en az üç ya da dört kat daha fazla parametre sayısına ulaşılmakta ve büyük veri ile optimizasyon

çalışmalarında global optimizasyona ulaşmak yerine yerel optimum tuzağına düşme sorunu ortadan kalkmaktadır. Bu çalışmada verilerin analizinde öncelikle birçok sınıflandırma tekniğinin temelini oluşturan lojistik regresyon gibi klasik bir yöntemle pilot çalışma yapılmıştır. 5'li Likert tipi ölçek ve bulanık Likert tipi ölçekle üretilen yapay veriler analiz edilmiştir. Bulanık Likert tipi ölçeğin analizinde öncelikle literatürde rastlandığı şekliyle minimum, maksimum ve tepe noktası şeklinde alınan ve üç ayrı sonuç dizisi üreten lojistik regresyon modelinin durulaştırmayla tek sonuçta birleştirilmesi ele alınmıştır. Literatürden farklı olarak bu tez çalışmasında minimum maksimum ve tepe noktasının tek bir veri seti şeklinde ifade edilerek doğrudan tek bir lojistik regresyon modeline ulaşılması ve tek sonuç dizisinin üretilmesi önerilmektedir. Ardından yapay zeka teknikleri içerisinde literatürde sağladığı avantajlardan ötürü ilgi odağı olan derin öğrenme tekniğinin performansı bulanık mantık çerçevesinde değerlendirilip karşılaştırılmıştır. Derin öğrenme ile yapılan analizlerde de hem literatürdeki bulanıklaştırmaya örnek teşkil eden tepe, maksimum ve minimum değerler için ayrı ayrı sonuçlarla durulaştırmaya gidilmiş hem de literatürden farklı olarak bulanık sayıların tek sonuç dizisi üretmesi önerilerek derin öğrenme modelinin performansları araştırılmıştır. Bulanık mantık ile derin öğrenme tekniğinin entegrasyonu ile klasik yöntemlere kıyasla daha başarılı test sonuçlarına ulaşılıp ulaşılamayacağı bu tez çalışmasının cevap aradığı en önemli olgudur. Literatürde tezde yapılan çalışmaya benzer bir çalışma olmaması bu tezin motivasyonunu oluşturmaktadır.

Memnuniyet tahmininde çoklu lineer regresyon analizi, korelasyon analizi, faktör analizi, kümeleme analizi, çok boyutlu ölçekleme, karar ağacı ve yapay sinir ağları gibi tekniklerin kullanımı oldukça yaygındır (Tayyar, 2010). Müşteri memnuniyeti tahmininde, tekniklerin veri yapısına bağlı olmadan yapay veri ile test edilmesi, 5'li Likert ölçeğini bulanık ölçeğe dönüştürülerek tekniklerin performanslarının arttırılması ve bu tekniklerin veri miktarına göre değişen davranışların incelenmesi bu çalışmayı özgün kılacak özelliklerdir.

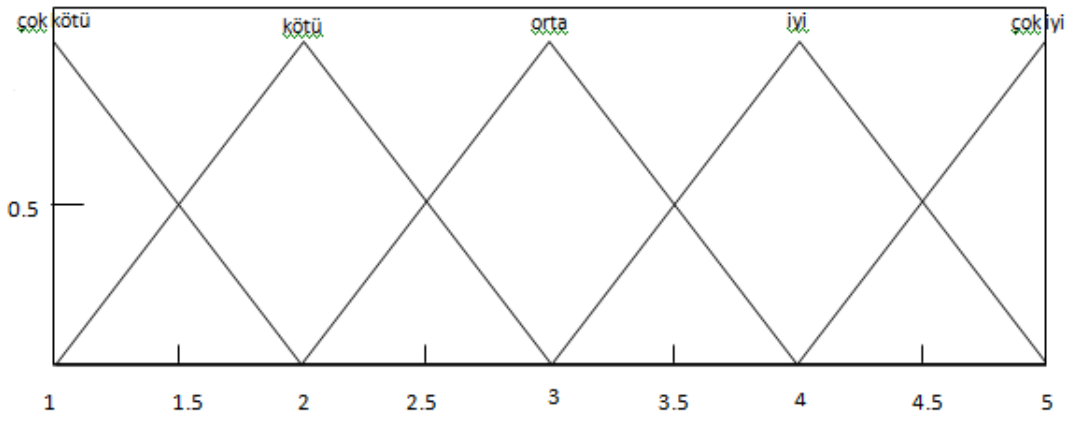
Bu bölümde tezin teorik kısmında açıklanan modellerin test edilmesi için yapılan tüm hazırlıklar ve uygulamaya ilişkin detaylar anlatılmıştır. Deneysel çalışma için üretilen yapay verinin özelliği açıklanmış, oluşturulan Python kodu verilmiştir. Oluşturulan yapay veri bulanık ölçek kullanılarak dönüştürülmüş ve uygulama için gereken eğitim, kontrol ve test veri kümelerinin oluşturulmasına ilişkin detaylar açıklanmıştır. Literatürde müşteri memnuniyetinin tahmininde yaygın

bir teknik olarak kullanılan lojistik regresyon modeli ile derin öğrenme temelini oluşturan yapay sinir ağları tekniklerini karşılaştıran çalışmalara dayanarak bu çalışmada öncelikle lojistik regresyon yöntemi kullanılarak bir pilot çalışma uygulanmıştır. Pilot çalışmanın sonuçlarına göre bulanık Likert ölçeğinin kullamının, lojistik regresyonun ürettiği sonuçları iyileştirdiği görülmüştür. Ayrıca literatürde yer alan bulanık lojistik regresyonu tekniğine göre daha kısa sürede sonuç üretmiştir. Bulanık Likert ölçeğinin lojistik regresyon performansını iyileştirdiği görüldükten sonra bulanık Likert ölçeği kullanılarak derin öğrenme mimarisi test edilmiştir.

3.1. Bulanık Likert Ölçeği

Özellikle sosyal bilimlerin en önemli tekniklerinden biri olan anket çalışmalarında Likert ölçeğinin bulanıklaştırılması araştırmalara büyük katkı sağlamıştır. Likert ölçeğinin doğasından kaynaklanan bilgi kaybı ve kapalı yanıt biçiminden kaynaklanan bilgi uyuşmazlığı sorunları bulanık kümeler yardımıyla aşılmıştır. Bulanık Likert ölçeğinde, iletişim teorisinin konsensüs kavramı uygulandığından geleneksel Likert ölçeğinden daha doğru bir ölçüm sonucu sağlayabileceği görülmüştür (Li, 2013).

Üçgensel bulanık sayılarla tanımlanan bulanık Likert ölçeğinin grafiksel gösterimi Şekil 3.1'de verilmiştir. Üçgensel sayının tepe noktası Likert ölçeğindeki kategorilere uygun şekilde belirlenmiştir. Tablo 3.1'de Likert ölçeğindeki kategoriler, dilsel ifadeler ve bulanık karşılıkları verilmiştir (Sreekumar ve Mahapatra, 2015).



Şekil 3.1. Üçgensel Bulanık Likert ölçeği (Sreekumar ve Mahapatra, 2015).

Tablo 3.1 Bulanık Likert Ölçeği (Sohn vd., 2016)

Likert Ölçeği	Dilsel İfadeler	Bulanık Karşılıklar
1	Çok Kötü	(0.5 , 1 , 1.5)
2	Kötü	(1.5 , 2 , 2.5)
3	Kararsızım	(2.5 , 3 , 3.5)
4	İyi	(3.5 , 4 , 4.5)
5	Çok İyi	(4.5 , 5 , 5.5)

3.2. Deneysel Çalışmada Kullanılan Yazılımlar

Deneysel çalışmalar Intel Core i7-7700HQ CPU, 16GB RAM, 2TB, 5,400-rpm HDD, 256GB SSD, Nvidia GTX 1050 Ti (4GB of VRAM) özelliklere sahip bir bilgisayarda yapılmıştır. Çalışmada modellerin tasarımı için Python, TensorFlow ve Keras yazılım platformları kullanılmıştır. Uygulama yazılım geliştirme ortamı olarak Spyder yazılımı kullanılmıştır.

3.3. Yapay Verinin Üretilmesi

Yapay zeka tekniklerini araştırmak için genellikle gerçek veriler kullanılmaktadır. Ancak teknikleri test etme noktasında gerçek verileri kullanmanın bazı dezavantajları vardır. Çoğu alandaki gerçek verilerin bütçe, teknik veya etik gibi çeşitli nedenlerle elde edilmesinin zor olması dezavantajlarından biridir. Diğeri ise gerçek verilerinin çoğunun kullanımının sınırlı olmasıdır. Yani kullanılacak veri kümeleri amaca uygun modeller içermemekte ya da, içindeki deseni bulmak için özel hazırlık gerektirmektedir. Bunun çözüm olarak yapay veri ya da diğer bir ifadeyle sentetik veri üreterek deneysel sonuçlar elde etmektir (Peng ve Hanke, 2016).

Yapay verilerin kullanımı, gerçekleşmesi beklenen, ancak henüz uygulamada gerçekleşmemiş olan değişkenliğin tanımlanması ve kontrol etme imkanı vermektedir. Parametreleri kontrol etme yeteneği, farklı koşullar altında sınıflandırma modellerinin performansının kapsamlı bir şekilde araştırılmasını sağlamaktadır. Yapay verilerle test tekniği, tekrarlanabilir deneysel bulguların oluşturulması açısından önemli bir tekniktir (Kennedy vd., 2011).

3.3.1 Yapay Veri Üretme Algoritması

Çalışmada yapay veri setini oluşturmak için Python kodu ile yazılmış bir fonksiyon oluşturulmuştur. Fonksiyonun algoritma adımları aşağıda verilmiştir:

1. Adım: Bağımsız ve bağımlı değişkenlerin değerlerini saklamak için iki boş dizi tanımlanmıştır.
2. Adım: Parametre sayısı kadar tekrar edecek bir döngü oluşturulmuştur. Döngü içerisinde normal dağılım oluşturacak belirtilen örnek sayısı kadar rasgele değerler üretilmektedir. Bu sayı üretme işlemi için daha sonra açıklanacak "truncnorm" kullanılmıştır. Döngünün her çevriminde bir satır oluşarak veri matrisi oluşturmak için daha önce tanımlanan bağımsız değişkenler dizisine aktarılmaktadır. Tablo 3.2'de bağımsız değişkenler için tanımlana dizi örneği verilmiştir.

Tablo 3.2 Bağımsız Değişlenler Dizisi

Örneklem büyüklüğü kadar üretilen sayılar

Parametre sayısı kadar döngü çevrimi	**.**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**…**
	0	4	3	5	1	2	3	3	3	1	2	5	4	3	5	5	4
	1	5	4	4	4	4	3	1	3	4	3	4	3	1	4	4	2
	2	2	3	4	3	4	3	5	4	5	2	4	3	4	2	4	4
	3	5	3	4	2	2	3	2	3	3	2	1	4	4	4	4	2
	4	2	3	3	4	5	5	3	1	1	5	3	4	3	4	3	4
	5	3	1	3	1	2	2	5	2	4	4	2	3	2	2	4	4
	6	1	1	2	3	2	5	2	4	5	4	3	2	2	1	2	3
	7	3	2	2	4	4	2	1	1	3	3	3	5	2	2	1	4
	8	2	3	1	3	5	5	3	4	2	4	5	4	5	3	4	3
	9	1	3	1	1	4	2	3	2	5	4	2	4	1	3	5	4

3. Adım: Bağımsız değişkenler dizisi matris haline getirilerek transpozu alındıktan sonraki hali Tablo 3.3'de verilmiştir.
4. Adım: Değişkenler arası ilişkilerin simetrik olduğu yani karmaşık olmayan veri kümeleriyle deneysel çalışma yapmak amacıyla (3.1)'de verilen denklem kullanılarak bağımsız değişkenler elde edilmiştir.

$$X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8 + X_9 + X_{10} \quad (3.1)$$

Bağımlı değişken için dengeli sınıf oluşturulması hedeflendiğinden Likert ölçeğinde yer alan üç değeri kullanılmıştır. (3.1)'de verilen ifade üç değeri için hesaplandığında sonuç 30 olarak bulunmuştur. 30 değeri eşik kabul edilerek 30 ve üstü memnun=1, 30 altı ise memnun değil=0 olarak kabul edilmiştir.

5. Adım: Bağımlı değişkenleri için elde edilen değerler bir eşik değeri kullanılarak "memnun=1" ve "memnun değil=0" şeklinde etiketlendirilmiştir. Bu etiketler veri matrisine eklendikten sonraki hali Tablo 3.4'de verilmiştir.

Tablo 3.3 Bağımsız Değişlenler Matrisi

Veri İndisleri	Bağımsız Değişkenler									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	4	5	2	5	2	3	1	3	2	1
1	3	4	3	3	3	1	1	2	3	3
2	5	4	4	4	3	3	2	2	1	1
3	1	4	3	2	4	1	3	4	3	1
4	2	4	4	2	5	2	2	4	5	4
5	3	3	3	3	5	2	5	2	5	2
6	3	1	5	2	3	5	2	1	3	3
7	3	3	4	3	1	2	4	1	4	2
8	1	4	5	3	1	4	5	3	2	5
9	2	3	2	2	5	4	4	3	4	4
10	5	4	4	1	3	2	3	3	5	2
11	4	3	3	4	4	3	2	5	4	4
12	3	1	4	4	3	2	2	2	5	1
13	5	4	2	4	4	2	1	2	3	3
14	5	4	4	4	3	4	2	1	4	5
…	4	2	4	2	4	4	3	4	3	4

Tablo 3.4 Veri Matrisi

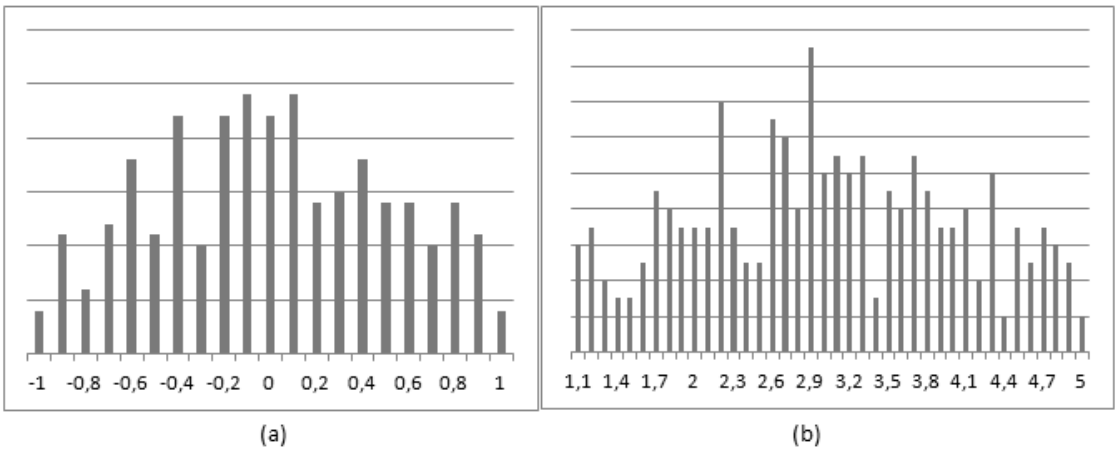
Veri İndisleri	Bağımsız Değişkenler										Bağımlı Değişken
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	2	5	2	1	3	2	4	5	3	0
1	1	3	4	3	3	1	3	3	3	2	0
2	2	3	4	1	1	3	4	5	4	2	0
3	3	4	4	3	1	1	3	1	2	4	0
4	2	5	4	5	4	2	4	2	2	4	1
5	5	5	3	5	2	2	3	3	3	2	1
6	2	3	1	3	3	5	5	3	2	1	0
7	4	1	3	4	2	2	4	3	3	1	0
8	5	1	4	2	5	4	5	1	3	3	1
9	4	5	3	4	4	4	2	2	2	3	1
10	3	3	4	5	2	2	4	5	1	3	1
11	2	4	3	4	4	3	3	4	4	5	1
12	2	3	1	5	1	2	4	3	4	2	0
13	1	4	4	3	3	2	2	5	4	2	0
14	2	3	4	4	5	4	4	5	4	1	1
...	3	4	2	3	4	4	4	4	2	4	1

3.3.2 5'li Likert Ölçeği Formatında Veri Üretme

Normal dağılıma uyan 5'li Likert ölçeği formatında veri üretmek için kullanılan Python scipy.stats kütüphanesinde hazır bulunan "truncnorm" fonksiyonu kullanılmıştır. Truncnorm fonksiyonu öncelikle -1 ile 1 arasında normal dağılıma

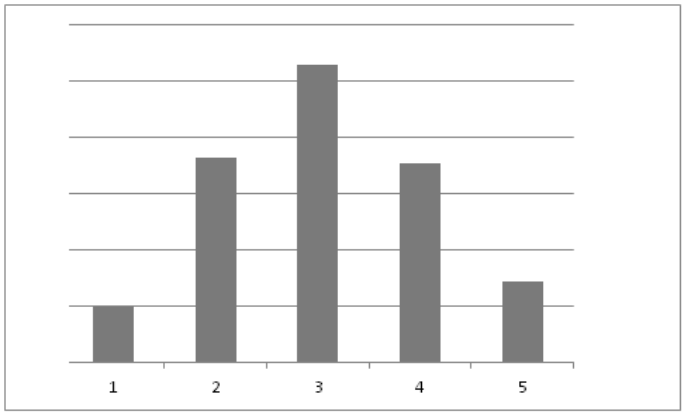
uygun olarak rastgele sayılar üretilmektedir. Bu sayıların toplamı ve ortalaması 0 olacak şekilde üretilmektedir. Daha sonra Şekil 3.2'de gösterildiği gibi üretilen sayılar 1 ile 5 aralığına dönüştürülmüştür. Bu dönüşüm için (3.2)'de verilen ifade kullanılmıştır. -1 ile 1 aralığında sayıların ortalaması olan 0 değerinden uzaklık 1 iken, 1 ile 5 aralığında sayıların ortalaması olan 3 değerinden uzaklık ise 2'dir. Bu nedenle önce üretilen değerler 2 ile çarpılmakta ve daha sonra 3 sayı eklenerek sayı doğrusunda 3 birim kaydırılmaktadır.

$$X = x * 2 + 3 \qquad (3.2)$$



Şekil 3.2 Sayı Aralıkları

Elde edilen 1 ile 5 aralığındaki sayılar tam sayıya dönüştürülerek Likert tipi veri kümesi oluşturulmuştur. Şekil 3.3'te verilen 100 örnek için oluşturulmuş grafik, üretilen veri kümesinin normal dağılıma uyduğunu göstermektedir.



Şekil 3.3 Likert Tipi Veri Örneği

5'li Likert ölçeği formatında yapay veri üretme algoritması için Python kodu aşağıda verilmiştir.

```
def yapay_veri(parametre=10,ornek_say=100):
    import numpy as np
    sonuc_listesi=[]
    parametre_listesi=[]
    for i in range(parametre):
        parametre_listesi.append(truncnorm(-1,1,loc=3,
                                 scale=2).rvs(ornek_say).round().astype(int))
    parametre_listesi=np.array(parametre_listesi)
    parametre_listesi=parametre_listesi.T
    for i in range(ornek_say):
        sonuc=sum(parametre_listesi[i])
        sonuc_listesi.append(sonuc)

    sonuc_listesi=np.array(sonuc_listesi)
    cutoff=int(parametre*3)
    sonuc_ikili=sonuc_listesi>cutoff
    sonuc_ikili=np.array(sonuc_ikili,dtype=int)
    sonuc_ikili=sonuc_ikili.reshape(ornek_say,1)
    x=np.hstack((parametre_listesi,sonuc_ikili))
    return (x)
```

3.4. Eğitim ve Test Veri Setlerinin Oluşturulması

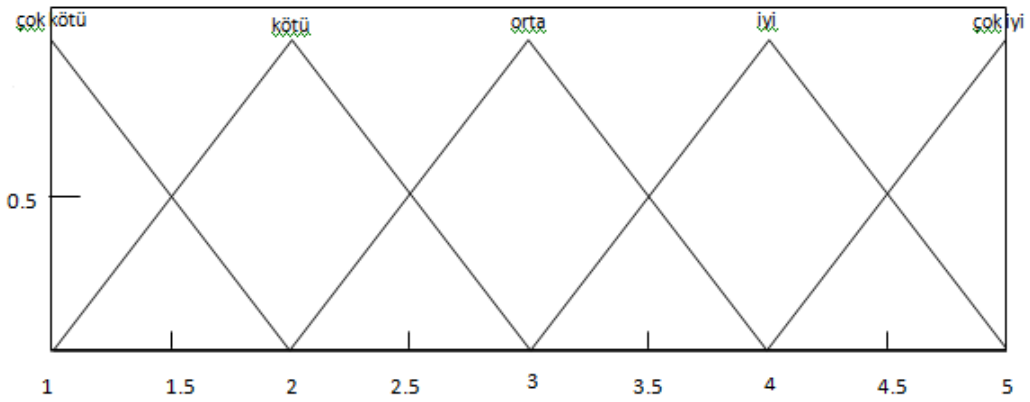
Yapay veri setleri oluşturulduktan sonra, bu veri kümeleri eğitim, doğrulama ve test setlerine bölünmüştür. Yapay veri setinden %70 oranında eğitim ve %30 oranında test seti oluşturmak için *sklearn.model_selection* kütüphanesindeki hazır olarak bulundan *train_test_split* fonksiyonu kullanılmıştır. Daha sonra doğrulama veri setini oluşturmak için eğitim setinin %30'u kullanılmıştır.

Bu işlem için kullanılan Python kodu aşağıda verilmiştir.

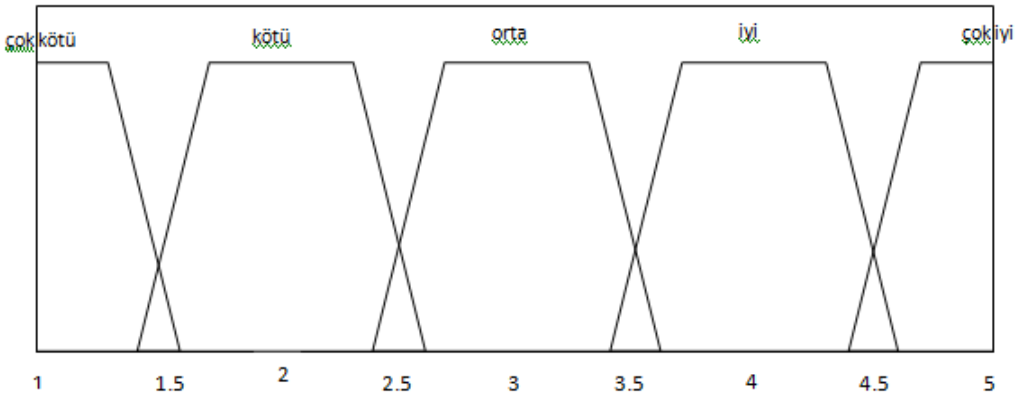
```
df=pd.DataFrame(x)
df_x = df.loc[:,0:9]
df_y = df.loc[:,[10]]
x_train, x_test, y_train, y_test=train_test_split(df_x,
                           df_y, test_size=0.3,random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_train,
                           y_train, test_size=0.3, random_state=42)
```

3.5. Bulanık Veri Setinin Oluşturulması

Eğitim ve Test veri setleri bulanık sayı fonksiyonları kullanılarak üçgensel ve yamuk bulanık sayılardan oluşan veri setlerine dönüştürülmüştür. Üçgensel Likert ölçeği için Sreekumar ve Mahapatra (2015) çalışmalarında verilen üçgensel bulanık sayılar, yamuksal Likert ölçeği için Güner ve Çomak (2014) çalışmalarında verilen yamuksal bulanık sayılar kullanılmıştır.



Şekil 3.4 Üçgensel Bulanık Likert Ölçeği



Şekil 3.5 Yamuk Bulanık Likert Ölçeği

Bulanık üçgensel Likert ölçeğin grafiksel gösterimi Şekil 3.4'te, yamuk bulanık Likert ölçeğin grafiksel gösterimi Şekil 3.5'te verilmiştir. Likert ölçeğinin bulanık karşılıkları Tablo 3.5'te verilmiştir.

Tablo 3.5 Bulanık Sayılar Tablosu

Likert Ölçeği	Dilsel İfadeler	Üçgensel Bulanık Karşılıklar	Yamuksal Bulanık Karşılıklar
1	Çok Kötü	(0.5 , 1 , 1.5)	(0.5, 0.75, 1.25, 1.5)
2	Kötü	(1.5 , 2 , 2.5)	(1.5 , 1.75 , 2.25, 2.5)
3	Kararsızım	(2.5 , 3 , 3.5)	(2.5 , 2.75, 3.25 , 3.5)
4	İyi	(3.5 , 4 , 4.5)	(3.5, 3.75, 4.25, 4.5)
5	Çok İyi	(4.5 , 5 , 5.5)	(4.5 , 4.75, 5.25 , 5.5)

Bulanık üçgensel sayılar ve bulanık yamuksal sayılar için iki ayrı dizi tanımlanmıştır.

3.6. Sınıflandırmada Kullanılan Başarı Ölçütleri

İkili karar problemlerinde sınıflandırıcı, örnekleri pozitif veya negatif olarak etiketlemektedir. Sınıflandırıcı tarafından verilen karar karışıklık matrisi veya beklenmedik durum tablosu olarak bilinen bir yapıda temsil edilmektedir. Şekil 3.6'da verilen karışıklık matrisi dört kategoriye sahiptir (Davis ve Goadrich, 2006):

- Doğru pozitifler (DP), doğru olarak etiketlenmiş pozitif örneklerdir.
- Yanlış pozitifler (YP), hatalı olarak etiketlenmiş pozitif örneklerdir.
- Doğru negatifler (DN) doğru olarak etiketlenmiş negatif örneklerdir.
- Yanlış negatifler (YN) hatalı olarak etiketlenmiş negatif örneklerdir.

		Tahmin Edilen Sınıf	
		Pozitif Sınıf	Negatif Sınıf
Gerçek Sınıf	Pozitif Sınıf	a	b
	Negatif Sınıf	c	d

a: TP (True Positive) **b: FN (False Negative)**
DP (Doğru Pozitif) **YN (Yanlış Negatif)**

c: FP (False Positive) **d: TN (True Negative)**
YP (Yanlış Pozitif) **DN (Doğru Negatif)**

Şekil 3.6 Karışıklık Matrisi (confusion matrix)

Model başarımının ölçülmesinde kullanılan en popüler ve basit yöntem, modele ait doğruluk oranı (accuracy) olarak bilinmektedir. Doğruluk oranı doğru sınıflandırılmış örnek sayısının (DP + DN), toplam örnek sayısına oranı şeklinde hesaplanmaktadır. Hata oranı ise yanlış sınıflandırılmış örnek sayısının (YP + YN), toplam örnek sayısına oranı şeklinde hesaplanmaktadır. Doğruluk oranı ve hata olanı (3.3) ve (3.4) ifadesinde verilen formül yardımıyla hesaplanmıştır (Canbaz, 2015).

$$Doğruluk\ Oranı = \frac{DP + DN}{DP + DN + YP + YN} \tag{3.3}$$

$$Hata\ Oranı = \frac{YP + YN}{DP + DN + YP + YN} \tag{3.4}$$

Dengesiz veri kümesinin sınıflandırma başarısını değerlendirirken sadece doğruluk oranı yeterli ölçüt değildir. Bunun yanı sıra seçicilik, kesinlik, duyarlılık ve F-ölçütü değerlerinin de hesaplanması gerekmektedir.

Kesinlik (precision), gerçek sınıfı pozitif olan ve pozitif tahmin edilen örnek sayısının, pozitif tahmin edilen örneklerin tamamına bölünmesi ile elde edilmektedir. Yani kesinlik ile pozitif tahmin edilen örnekler içeresinde gerçekte sınıf etiketi pozitif olan örneklerin oranı ölçülmektedir. Kesinlik (3.5) ifadesinde verilen formül yardımıyla hesaplanmıştır (Halepmollası, 2016)

$$Kesinlik = \frac{DP}{DP + YP} \qquad (3.5)$$

Duyarlılık (recall) ise gerçek sınıfı pozitif olan ve pozitif tahmin edilen örnek sayısının, gerçek sınıfı pozitif olan örneklerin tamamına oranı alınarak hesaplanmaktadır. Yani duyarlılık ile gerçekte sınıf etiketi pozitif olan örnekler içeresinde doğru tahmin edilen örneklerin oranı ölçülmektedir. Duyarlılık (3.6) ifadesinde verilen formül yardımıyla hesaplanmıştır (Halepmollası, 2016).

$$Duyarlılık = \frac{DP}{DP + YN} \qquad (3.6)$$

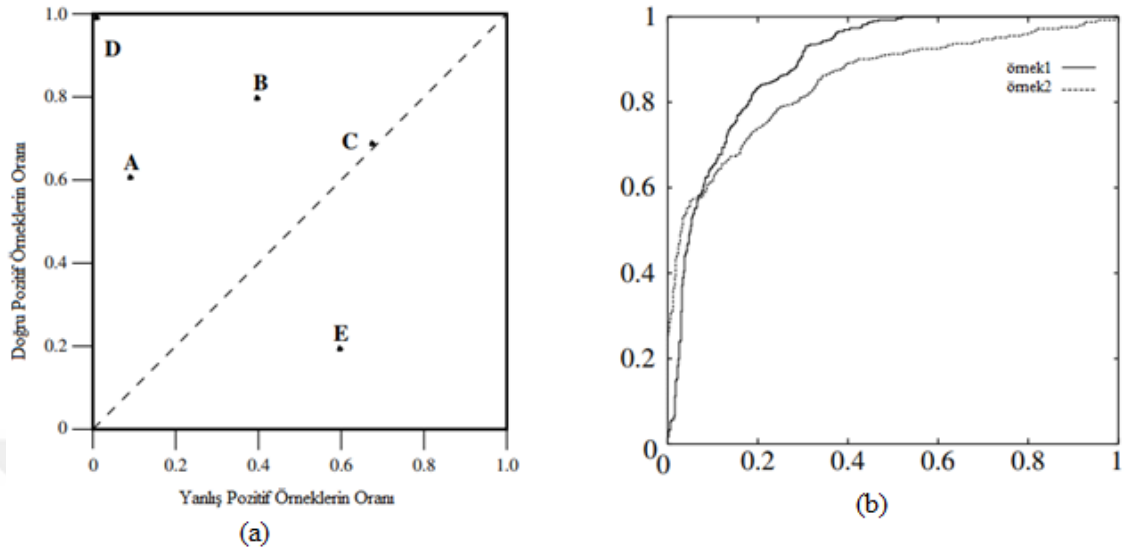
Her iki ölçütü beraber değerlendirmede kullanılan F-ölçütü (F-measure), kesinlik ve duyarlılığın harmonik ortalamasıdır. Geliştirilen model ne kadar iyi ise F-ölçütü o kadar büyük bir değer almaktadır. F-ölçütü (3.7) ifadesinde verilen formül yardımıyla hesaplanmıştır (Canbaz, 2015).

$$F - \text{Ölçütü} = \frac{2 * Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \qquad (3.7)$$

Bu ölçütler farklı sınıflardaki etiketlerin doğru sınıflandırılmasını değerlendirir. Ancak bu ölçütler tek sınıfa (olumlu örneklere) odaklanarak sonuç vermektedir. ROC eğrisi ise sınıflandırıcı performansını kapsamlı olarak değerlendirmektedir (Sokolova vd., 2006). ROC eğrisi (Receiver Operating Characteristic- Alıcı Çalışma Karakteristiği) sınıflandırma performansının iki boyutlu bir ölçüsüdür. ROC eğrisi altındaki alan (AUC) performansın bir yönünü ölçen skaler bir ölçüdür (Marzban, 2004).

ROC grafikleri, gerçek pozitif oranın Y ekseninde çizildiği ve yanlış pozitif oranın X ekseninde çizildiği iki boyutlu grafiklerdir. Bir ROC grafiği, fayda (gerçek

pozitifler) ve maliyet (yanlış pozitifler) arasındaki göreceli değişmeleri göstermektedir. Şekil 3.7 (a)'da gösterilen kesikli çizgi %50 olasılığı temsil etmektedir. Buradaki en iyi tahmin "D" noktasıdır (Fawcett, 2006).



Şekil 3.7 (a) ROC Alanı , (b) ROC Eğrisi

Örneklemden farklı oranlarda negatif ve pozitif örnekler alınarak eşik değerler oluşturulmaktadır. Her eşik deşerler için tahmin sonucu ROC alanına işlenmektedir. Daha sonra bu noktalar birleştirilerek ROC eğrisini oluşturmaktadır. Şekil 3.7 (b)'de gösterilen iki örnek içerisinde hangisini daha başarı olduğuna kapladıkları alana göre karar verilmektedir.

3.7. Lojistik Regresyon Deneysel Çalışması

Sınıflandırma problemlerinde en klasik yöntemlerden biri lojistik regresyon yöntemidir. Lojistik regresyonun tahmin etme başarısı büyük ölçüde bağımlı değişkenlere ve verinin yapısına bağlı olup, derin öğremmeye modeline göre daha az sayıda faktörden etkilenmektedir.

3.7.1. Likert Tipi Veri ile Lojistik Regresyon Modeli

Lojistik regresyon modeli oluşturmak için Python'un sklearn.linear_model kütüphanesinde bulunan LogisticRegression fonksiyonlar kullanılmıştır. Oluşturulan modeli eğitim veri ile beslemek için fit komutu kullanılmıştır. Daha sonra test verisi içerisinde bulunan bağımlı değişken değerleri kullanılarak predict komutu ile bağımlı değişkenin tahmini değerleri üretilmiştir. Python'un sklearn kütüphanesinde bulunan

metrics fonksiyonu ile ağı başarısı ölçülmüştür. Açıklanan bu aşamaların Python kodu aşağıda verilmiştir.

```
lr_model =LogisticRegression(random_state=42)
lr_model.fit(x_train, np.ravel(y_train))
lr_predict_test = lr_model.predict(x_test)
print("Başarı:{0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test)))
```

Keras kütüphanesinde lojistik regresyon modelinin başarısını iyileştirmek için regulasyon parametreleri gibi farklı fonksiyon ayarları mevcuttur. Ancak pilot çalışmasındaki amaç veri kümesinin bulanık kümeye dönüştürülmesi durumunda performans artışının olup olmayacağını gözlemlemek olduğundan, fonksiyonun varsayılan ayarları kullanılarak test işlemi yapılmıştır.

3.7.2. Üçgensel Bulanık Lojistik Regresyon Modeli

Bulanık lojistik regresyon modeli tezin literatür kısmında açıklanan tekniğe göre yapılmıştır. Üçgensel sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için ayrı veri seti elde edilmiştir. Böylece her veri seti için ayrıca tahmini değerler oluşturulmuştur. Yani çıktı da bulanık üçgensel olarak üretilmiştir. Daha sonra ağırlık merkezi yöntemi kullanılarak durulama işlemi uygulanmış ve tek çıktı değeri elde edilmiştir. Açıklanan bu aşamaların Python kodu aşağıda verilmiştir.

```
lr_model =LogisticRegression(random_state=42)
lr_model.fit(l_x_train, np.ravel(y_train))
l_lr_predict_test = lr_model.predict_proba(l_x_test)[:,1]

lr_model =LogisticRegression(random_state=42)[:,1]
lr_model.fit(m_x_train, np.ravel(y_train))
m_lr_predict_test = lr_model.predict_proba(m_x_test)

lr_model =LogisticRegression( random_state=42)[:,1]
lr_model.fit(u_x_train, np.ravel(y_train))
u_lr_predict_test = lr_model.predict_proba(u_x_test)

predict_test=(l_lr_predict_test+m_lr_predict_test+u_lr_predict_test)/3
predict_test=(predict_test>0.5)
print("Başarı: {0:.4f}".format(metrics.accuracy_score(y_test, predict_test)))
```

3.7.3. Yamuk Bulanık Lojistik Regresyon Modeli

Bulanık lojistik regresyon modeli üçgensel sayılar içeren verilerle test edildiği gibi yamuk bulanık sayılar içeren veri seti ile de test edilmiştir. Aynı şekilde minimum değeri, tepe değeri ve maksimum değeri için ayrı veri seti elde edilmiştir. Böylece her veri seti için ayrıca tahmini değerler oluşturulmuş ve durulama işlemi uygulanmıştır. Açıklanan bu aşamaların Python kodu aşağıda verilmiştir.

```
lr_model =LogisticRegression(random_state=42)
lr_model.fit(lt_x_train, np.ravel(y_train))
l_lr_predict_test = lr_model.predict_proba(lt_x_test)[:,1]

lr_model =LogisticRegression(random_state=42)
lr_model.fit(mt_x_train, np.ravel(y_train))
m_lr_predict_test = lr_model.predict_proba(mt_x_test)[:,1]

lr_model =LogisticRegression(random_state=42)
lr_model.fit(nt_x_train, np.ravel(y_train))
n_lr_predict_test = lr_model.predict_proba(nt_x_test)[:,1]

lr_model =LogisticRegression(random_state=42)
lr_model.fit(ut_x_train, np.ravel(y_train))
u_lr_predict_test = lr_model.predict_proba(ut_x_test)[:,1]

predict_test=(l_lr_predict_test+m_lr_predict_test+n_lr_predict_test+u_lr_predict_test)/4
predict_test=(predict_test>0.5)
print("Başarı: {0:.4f}".format(metrics.accuracy_score(y_test, predict_test)))
```

3.7.4. Bulanık Likert Tipi Veri ile Lojistik Regresyon Modeli

Literatürde genellikle 3.6.2 ve 3.6.3'te anlatıldığı gibi üçgensel ve yamuk sayıların her bir elemanı için ayrı veri setleri oluşturularak ayrı modeller analiz edilmiş ve durulaştırma yapılarak birleştirilmektedir. Bu tez çalışmasında üçgensel bulanık sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için tek veri seti elde edilmiştir (bölüm 3.5'de veri setinin oluşturulması ayrıntılı olarak açıklanmıştır). Bu bulanık veri seti ile lojistik regresyon uygulanmış ve ikili (binary) tahmini değerler oluşturulmuştur. Çıktı bulanık sayı olmadığından durulama işlemine ihtiyaç duyulmamıştır. Açıklanan bu aşamaların Python kodu aşağıda verilmiştir.

```
lr_model =LogisticRegression(random_state=42)
lr_model.fit(fuzzy_x_train, np.ravel(y_train))
lr_predict_test = lr_model.predict(fuzzy_x_test)
print("Başarı:{0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test)))
```

3.7.5. Lojistik Regresyon Sonuçlarının Raporlanması

Lojistik Regresyonun sınıflandırma performansını gözlemlemek için her biri 30 adet olacak şekilde 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 örnek içeren bir yapay veri kümeleri oluşturulmuştur. Bu kümeler kullanılarak lojistik regresyon modelleri test edilmiş ve modellerinin performans ölçütleri kayıt altına alınmıştır. Her biri 100 örnek içeren 30 adet Likert tipi verilerin kullanıldığı lojistik Regresyon sınıflandırmanın performans değerleri Tablo 3.6'da verilmiştir.

Tablo 3.6 100 Örnekli 30 Adet Veri Setinin Lojistik Regresyon Başarı Ölçütleri Örneği

Lojistik Regresyon	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**30**	**Ortalama**
Hesaplama Süresi (milisaniye)	0,002	0,002	0,001	0,002	0,002	0,001	…	0,002	0,0017
Eğitim Doğruluk	0,714	0,816	0,674	0,735	0,714	0,939	…	0,796	0,7497
Eğitim Kesinlik	0,655	0,750	0,677	0,667	0,680	0,882	…	0,750	0,7141
Eğitim Duyarlılık	0,826	0,600	0,821	0,700	0,739	0,938	…	0,923	0,7611
Eğitim F-ölçütü	0,731	0,667	0,742	0,683	0,708	0,909	…	0,828	0,7340
Eğitim ROC alanı	0,721	0,756	0,649	0,729	0,716	0,938	…	0,788	0,7374
Eğitim Karışıklık Matrisi	[16 10] [4 19]	[31 3] [6 9]	[10 11] [5 23]	[22 7] [6 14]	[18 8] [6 17]	[31 2] [1 15]	… …	[15 8] [2 24]	
Test Doğruluk	0,567	0,367	0,500	0,433	0,500	0,600	…	0,500	0,5278
Test Kesinlik	0,667	0,500	0,450	0,400	0,357	0,611	…	0,353	0,4737
Test Duyarlılık	0,375	0,105	0,692	0,429	0,455	0,688	…	0,600	0,5447
Test F-ölçütü	0,480	0,174	0,546	0,414	0,400	0,647	…	0,444	0,4789
Test ROC alanı	0,580	0,462	0,523	0,433	0,490	0,594	…	0,525	0,5392
Test Karışıklık Matrisi	[11 3] [10 6]	[9 2] [17 2]	[6 11] [4 9]	[7 9] [8 6]	[10 9] [6 5]	[7 7] [5 11]	… …	[9 11] [4 6]	

Tablodan da görüldüğü gibi 100 adet Likert tipi veri ile yapılan lojistik regresyon sınıflandırmasında başarı kriteri doğruluk oranı olarak seçildiğinde, eğitim başarısı %75 iken test başarısının %52 olduğu görülmüştür. Karışıklık matrisi incelendiğinde de yanlış sınıfa atanan verilerin sayısı oldukça fazladır. Buradan test sonucunu dikkate alarak eğitim sürecini başarısız olduğu ve modeli test verisini sınıflandıramadığı sonucuna ulaşılmıştır. Aynı analizler 200-1000 arası örnek için de yapılmış ve Tablo 3.8'de verilmiştir. 200-1000 adet veri kümeleri ile yapılan sınıflandırmada test başarısı %67'den %92'ye yükselmiştir. Örnek sayısı arttıkça başarının arttığı görülmüştür.

Üçgensel bulanık Likert tipi veri kümeleri ile yapılan lojistik regresyon sınıflandırmanın performans değerleri Tablo 3.7'de verilmiştir. Burada eğitim başarısı %85 iken test başarısı %65 olduğu görülmüştür. Karışıklık matrisi incelendiğinde yanlış sınıfa atanan verilerin sayısı lojistik regresyona göre azdır. Test başarısı yüksek olmasa da model bir sınıflandırma yapmıştır. Aynı analizler 200-1000 arası örnek için de yapılmış ve Tablo 3.8'de verilmiştir. 200-1000 adet veri

kümeleri ile yapılan sınıflandırmada test başarısı %84'den %98'ye yükselmiştir. Örnek sayısı arttıkça başarının arttığı görülmüştür.

Tablo 3.7 Bulanık Likert Tipi Lojistik Regresyon Başarı Ölçütleri

Bulanık Likert Tipi Lojistik Regresyon	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**30**	**Ortalama**
Hesaplama Süresi (milisaniye)	0,003	0,003	0,003	0,003	0,003	0,003	…	0,003	0,0030
Eğitim Doğruluk	0,898	0,857	0,796	0,857	0,857	0,959	…	0,878	0,8571
Eğitim Kesinlik	0,875	0,786	0,781	0,810	0,833	0,938	…	0,864	0,8373
Eğitim Duyarlılık	0,913	0,733	0,893	0,850	0,870	0,938	…	0,864	0,8596
Eğitim F-ölçütü	0,894	0,759	0,833	0,829	0,851	0,938	…	0,864	0,8460
Eğitim ROC alanı	0,899	0,823	0,780	0,856	0,858	0,954	…	0,876	0,8491
Eğitim Karışıklık Matrsi	[23 3] [2 21]	[31 3] [4 11]	[14 7] [3 25]	[25 4] [3 17]	[22 4] [3 20]	[32 1] [1 15]	… …	[24 3] [3 19]	
Test Doğruluk	0,800	0,467	0,700	0,533	0,633	0,633	…	0,600	0,6500
Test Kesinlik	0,917	0,800	0,611	0,500	0,500	0,632	…	0,643	0,6098
Test Duyarlılık	0,688	0,211	0,846	0,500	0,546	0,750	…	0,563	0,6442
Test F-ölçütü	0,786	0,333	0,710	0,500	0,522	0,686	…	0,600	0,5985
Test ROC alanı	0,808	0,560	0,717	0,531	0,615	0,625	…	0,603	0,6546
Test Karışıklık Matrsi	[13 1] [5 11]	[10 1] [15 4]	[10 7] [2 11]	[9 7] [7 7]	[13 6] [5 6]	[7 7] [4 12]	… …	[9 5] [7 9]	

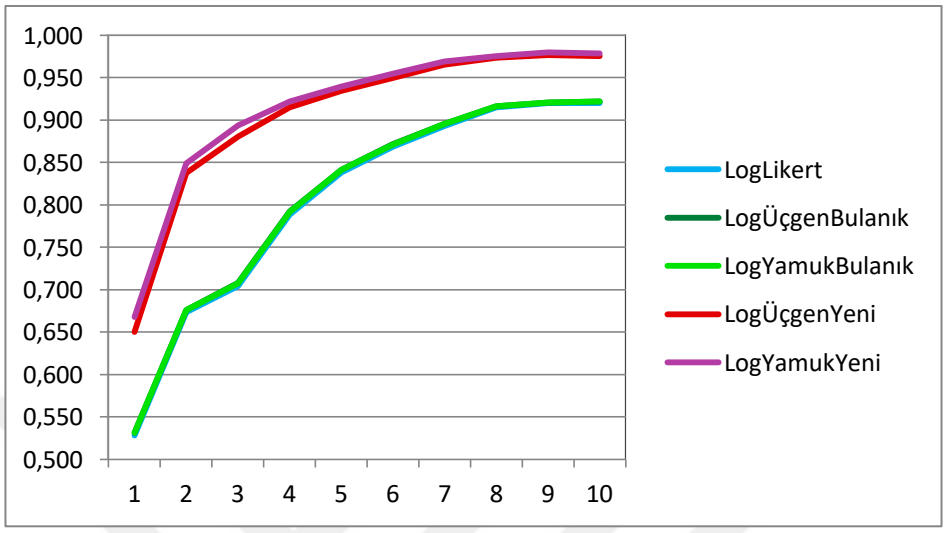
Tüm veri kümeleri için yapılan sınıflandırma sonucunda elde edilen test için doğruluk oranların ortalama değerleri Tablo 3.8'de toplanmıştır. Diğer ölçütlerin ortalama değeri Ek 1'de verilmiştir.

Tablo 3.8 Lojistik Regresyon Modellerinin Ortalama Test Doğruluk Değerleri

Veri Sayısı	Likert Tipi Lojistik Regresyon	Üçgensel Bulanık Lojistik Regresyon	Yamuk Bulanık Lojistik Regresyon	Üçgensel Bulanık Likert Tipi Lojistik Regresyon	Yamuk Bulanık Likert Tipi Lojistik Regresyon
100	0,528	0,531	0,531	0,650	0,668
200	0,673	0,676	0,676	0,838	0,849
300	0,704	0,708	0,707	0,881	0,894
400	0,788	0,792	0,792	0,915	0,922
500	0,838	0,842	0,841	0,934	0,940
600	0,869	0,872	0,871	0,949	0,955
700	0,893	0,896	0,896	0,966	0,969
800	0,915	0,916	0,916	0,974	0,976
900	0,920	0,921	0,921	0,977	0,980
1000	0,920	0,922	0,922	0,976	0,979

Yapay veri içindeki sınıflar dengeli olacak şekilde oluşturulduğundan modellerin sınıflandırma başarısı için doğruluk oranları karşılaştırmak yeterlidir. Bu tablodan elde edilen grafikler Şekil 3.8'de verilmiştir. Buradan tüm modellerin sınıflandırma başarısı veri sayısı ile doğru orantılı olarak artmaktadır. Lojistik regresyon ile bulanık lojistik regresyon modelleri birbirine yakın değerler

verdiğinden grafikte tek eğri gibi görünmektedir. Çalışmada önerilen bulanık Likert tipi veri ile uygulanan lojistik regresyon her veri seti için daha başarılı sonuçlar üretmiştir. Özellikle veri kümesinin küçük olduğu durumlarda büyük avantaj sağlamaktadır.

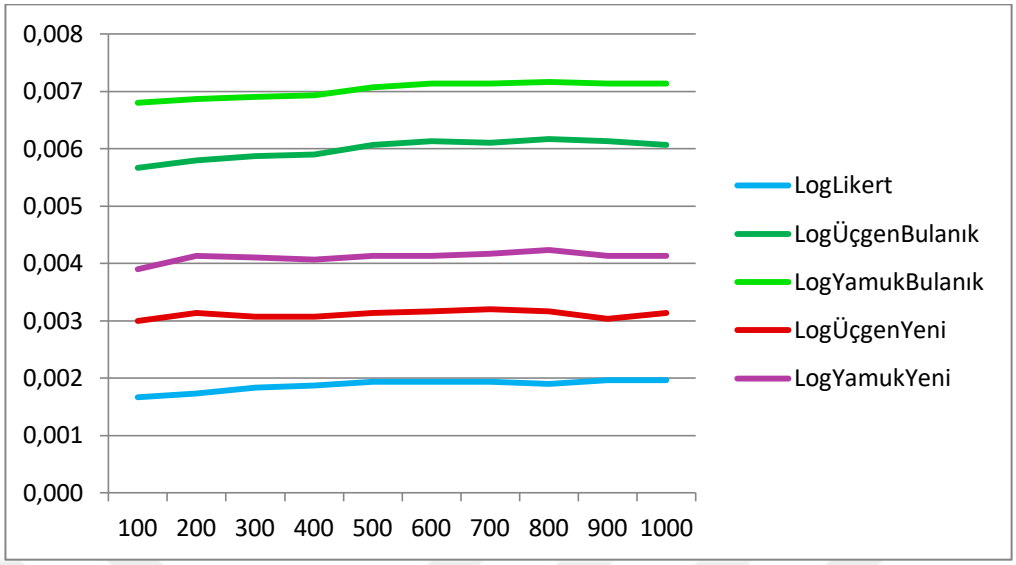


Şekil 3.8 Lojistik Regresyon Modellerinin Doğruluk Oranlar Grafiği

Kullanılan yaklaşım modelin tahmin etme başarısını iyileştirmekle kalmayıp hesaplama hızına da katkı sağlamaktadır. Literatürde yer alan yaklaşımda üçgensel sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için ayrı lojistik regresyon modeli kurularak ayrı tahmini değerler oluşturulmuştur. Yani çıktı da bulanık üçgensel olarak üretilmektedir. Daha sonra durulama tekniği kullanılarak kesin sayıya dönüştürülmektedir. Bu tez çalışmasında önerilen yaklaşımla tek lojistik regresyon için hesaplama yapıldığından daha kısa süre de sonuç üretilmektedir. Hesaplama süreleri Tablo 3.9 ve grafiği Şekil 3.9'da verilmiştir.

Tablo 3.9 Lojistik Regresyon Modellerinin Hesaplama Süreleri (milisaniye)

Veri Sayısı	Likert Tipi Lojistik Regresyon	Üçgensel Bulanık Lojistik Regresyon	Yamuk Bulanık Lojistik Regresyon	Üçgensel Bulanık Likert Tipi Lojistik Regresyon	Yamuk Bulanık Likert Tipi Lojistik Regresyon
100	0,002	0,006	0,007	0,003	0,004
200	0,002	0,006	0,007	0,003	0,004
300	0,002	0,006	0,007	0,003	0,004
400	0,002	0,006	0,007	0,003	0,004
500	0,002	0,006	0,007	0,003	0,004
600	0,002	0,006	0,007	0,003	0,004
700	0,002	0,006	0,007	0,003	0,004
800	0,002	0,006	0,007	0,003	0,004
900	0,002	0,006	0,007	0,003	0,004
1000	0,002	0,006	0,007	0,003	0,004



Şekil 3.9 Lojistik Regresyon Modellerinin Hesaplama Süresi Grafiği

Elde edilen sonuçlara göre Likert tipi ölçeğin bulanık ölçeğe dönüştürülmesi lojistik regresyon sınıflandırma başarısını artırmaktadır. Lojistik regresyon ile yapılan deneysel çalışma hedeflenen sonuca ulaştığından aynı tekniği derin öğrenme modeline uygulayarak test sonuçları değerlendirilmesi hedeflenmektedir.

3.8. Derin Öğrenme Modelleri Oluşturulması

Derin Öğrenme modelinin performansı ise ağ mimarisinin seçimine duyarlıdır. Her katmandaki farklı katman sayısı ve düğüm sayısı kombinasyonu, derin ağın farklı mimarilerini oluşturmaktadır.

3.8.1 Derin Öğrenme Modeli

Çalışmada bir veri kümesi için en iyi mimariyi bulmak için farklı katman sayısı, düğüm sayısı ve aktivasyon fonksiyonu kombinasyonu ile denemeler yapılmıştır. Bu denemeler için rasgele bir dizi mimari üretilmiştir. Giriş katmanında on bağımsız değişken ve çıkış katmanında bir bağımsız değişken yer almaktadır. Giriş katmanından çıkış katmanına doğru gidildikçe veri analiz edilerek tek değişkenle özetlenmektedir (Goodfellow vd., 2016). Bu nedenle gizli katmandaki sayılar için bir ile dokuz arasındaki değer seçilmiştir. Aktivasyon fonksiyonlarından yapay sinir ağlarında yaygın olarak kullanılan “Sigmoid” ve “Tanh” fonksiyonları, derin öğrenme mimarilerinde sağladığı avantajlardan dolayı yangın olarak kullanılan “Relu” aktivasyon fonksiyonu ve bulanık sayıların kullanıldığı için relu aktivasyon

fonksiyonun pürüzsüz hali olan “Softplus” aktivasyon fonksiyonları seçilmiştir (Patterson ve Gibson, 2017: 70).

Her mimari 30 adet 500 örnek içeren veri kümesi kullanılarak eğitilmiş olup, eğitim ve test başarısı hesaplanarak en yüksek başarı skorunu veren mimari seçilmiştir. Mimari seçimi için yapılan deneysel çalışmanın sonuçları Tablo 3.10’da verilmiştir.

Elde edilen sonuçlara göre en başarılı sonuçları en kısa sürede hesaplayan mimari seçilmiştir. Tek katmanlı yapay sinir ağlarında Tanh fonksiyonu en iyi sonuç verirken mimarı derinleştikçe performansında düşüş olduğu gözlemlenmiştir. Sigmoid fonksiyonu ise üç katmana kadar sınıflandırma başarısı arttırmıştır, ancak seçilen fonksiyonlar arasından en yavaş fonksiyon olduğu gözlemlenmiştir. Bu sonuç literatürle karşılaştırıldığında sebebi, çıkış değerleri 0 ve 1 değerlerine yaklaştıkça ağırlıkların güncelleme miktarları azalacağı için öğrenme süreci yavaşladığı şeklinde açıklanmaktadır. Bu nedenle Sigmoid fonksiyonu yapay sinir ağlarında etkili bir şekilde kullanılmasına rağmen derin ağlarda çok fazla tercih edilmemektedir (Goodfellow vd., 2016: 183). Softplus fonksiyonu Relu fonksiyonun ile aynı sonuçlar üretmesine rağmen hesaplama süresi daha yüksektir. Bu nedenle Likert tipi veri kümesine uygulanacak derin öğrenme mimarisi için “Relu” aktivasyon fonksiyonu seçilmiştir. Relu aktivasyon fonksiyonun dikkate alarak sonuçlar incelendiğinde iki katmanlı, sırayla beş ve iki nöron bulunan ağ seçilmiştir. Derin öğrenme kavramı yapay sinir ağılarına dayanmakta olup, birden fazla gizli katman içeren yapay sinir ağı derin ağ olarak tanımlanmaktadır (Deng ve Yu, 2014: 206). Bu nedenle çalışmada seçilen mimari Derin öğrenme mimarisi olarak kabul edilmiştir.

Tablo 3.10 Derin Öğrenme Mimari Seçimi

Katman Sayısı	Nöron Sayısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı
Ortlama süre		**Relu 62 sn**		**Sigmoid 82 sn**		**SoftPlus 79 sn**		**Tanh 69 sn**	
1	**[9]**	0,949	0,895	0,964	0,909	0,987	0,935	0,986	0,959
1	**[8]**	0,991	0,928	0,973	0,907	0,985	0,942	0,986	0,956
1	**[7]**	0,949	0,909	0,967	0,938	0,985	0,943	0,988	0,954
1	**[6]**	0,989	0,948	0,958	0,896	0,981	0,927	0,987	0,956
1	**[5]**	0,985	0,939	0,964	0,929	0,981	**0,963**	0,984	0,953
1	**[4]**	0,989	0,945	0,962	0,921	0,976	0,932	0,987	0,959
1	**[3]**	0,982	0,947	0,916	0,889	0,971	0,923	0,984	0,954
1	**[2]**	0,979	0,935	0,952	0,914	0,966	0,907	0,980	0,950
Ortlama süre		**Relu 51 sn**		**Sigmoid 88 sn**		**SoftPlus 73 sn**		**Tanh 78 sn**	
2	**[8,6]**	0,994	0,953	0,992	0,950	0,994	0,953	0,959	0,959
2	**[8,4]**	0,995	0,955	0,993	0,951	0,995	0,955	0,959	0,959
2	**[8,2]**	0,991	0,957	0,993	0,951	0,991	0,957	0,959	0,959
2	**[6,4]**	0,995	0,956	0,995	0,955	0,995	0,956	0,959	0,959
2	**[6,2]**	0,994	0,958	0,993	0,949	0,994	0,958	0,959	0,959
2	**[5,2]**	0,994	**0,961**	0,993	0,955	0,994	0,957	0,959	0,959
2	**[4,2]**	0,994	0,958	0,992	0,955	0,994	0,958	0,959	0,959
Ortlama süre		**Relu 42 sn**		**Sigmoid 86 sn**		**SoftPlus 65 sn**		**Tanh 62 sn**	
3	**[8,6,4]**	0,991	0,910	0,993	0,957	0,991	0,954	0,953	0,941
3	**[8,6,2]**	0,993	0,919	0,993	0,957	0,991	0,953	0,953	0,941
3	**[8,4,2]**	0,993	0,908	0,993	0,957	0,991	0,959	0,953	0,941
3	**[6,4,2]**	0,995	0,955	0,993	0,955	0,993	0,948	0,953	0,941
3	**[6,2,2]**	0,955	0,895	0,993	0,955	0,991	0,959	0,953	0,941
3	**[5,2,2]**	0,953	0,905	0,993	0,955	0,991	**0,961**	0,953	0,941
3	**[4,2,2]**	0,955	0,901	0,993	0,955	0,993	0,948	0,953	0,941
Ortlama süre		**Relu 39 sn**		**Sigmoid 83 sn**		**SoftPlus 63 sn**		**Tanh 65 sn**	
4	**[8,6,4,2]**	0,990	0,893	0,876	0,868	0,923	0,885	0,842	0,844
4	**[8,6,2,2]**	0,988	0,899	0,876	0,868	0,923	0,885	0,842	0,844
4	**[8,4,2,2]**	0,985	0,897	0,876	0,868	0,923	0,909	0,842	0,844
4	**[6,4,2,2]**	0,977	0,891	0,876	0,868	0,923	0,881	0,842	0,844
4	**[6,2,2,2]**	0,986	0,889	0,876	0,868	0,923	0,907	0,842	0,844
4	**[4,2,2,2]**	0,990	0,905	0,876	0,868	0,923	0,907	0,842	0,844
4	**[5,2,2,2]**	0,993	0,895	0,876	0,868	0,923	0,885	0,842	0,844

Bu mimariyi oluşturmak için keras.models kütüphanesindeki Sequential fonkiyonu ile model oluşturulmuştur. Sequential fonkiyonu sayı, metin,ses gibi sıralı veri ile eğitim için uygun olan bir model oluşturmaktadır. Model oluşturulduktan sonra add fonksiyonu kullanılarak ağa katmanlar eklenmektedir. Katmanlar eklenirken katmanın türü, her katmandaki nöron sayısı, uygulanacak aktivasyon

fonksiyonu ve ağırlıklara başlangıç değerlerin ne şekilde atanacağını belirtilmesi gerekmektedir. Gizli katmanlar için Relu aktivasyon fonksiyon, çıktı katma için 1 ya da 0 sonucu üretmek için Sigmoid aktivasyon fonksiyonu kullanılmıştır. Ağırlıkların başlangıç değerlerini rastgele atanması uniform dağılımına uygun yapılmıştır.

Bu mimaride kullanılan "Yoğun (Dense)" yada diğer ismiyle tam bağlantılı katman özelliği, her girişin her çıkışa ağırlıkla bağlandığı doğrusal bir işlemin bulunmasıdır. Anket sonuçların elde edilmesi için uygun olduğundan seçilmiştir. Optimizasyon türünü ve kayıp fonksiyon türünü seçmek için compile fonksiyonu kullanılmaktadır. Çevrim sayı için 1000 değeri belirlenmiştir, ancak aşırı öğrenmeyi önlemek için erken sonlandırma tekniği ile en optimal değerlerde eğitim durdurulacaktır. Erken sonlandırma kriterinin belirlenmesinde doğrulama kümesi kullanılmıştır. Böyle test aşamasında ağın hiç görmediği veri ile test edilerek daha gerçekçi sonuçlar elde edilmiştir. Doğrulama kümesinin verinin geri kalanı ile aynı özellikte olduğunu garantilemek için k-katlı doğrulama uygulanmıştır. Ancak üretilen yapay veri homojen olduğundan bir fark yaratmamıştır. Aksine hesaplama süresini arttırdığında bu deneysel çalışma için istenmeyen sonuçlar doğurmuştur. Bu neden k-katlı doğrulama uygulanmamıştır. Aşırı öğrenmeyi önlemek için kullanılan bir diğer teknik seyreltme regülâsyonudur. Üretilen yapay verinin özelliklerinden kaynaklı mimarinin sınıflandırma performansına bir katkı sağlamamıştır. Derin öğrenme mimarisin performansını iyileştirmek için literatürde birçok yaklaşım bulunmaktadır. Ancak çalışmanın amacı en iyi sınıflandırma yapan mimariyi bulmaktan ziyade tüm mimarilerin benzer şekilde kurgulanıp Likert ölçeğinin bulanık ölçeğe dönüştürüldüğünde performanstaki değişimi gözlemlemektir. Bu mimarini Python kodu aşağıda verilmiştir.

```
for i in range (0,1):
    model = Sequential()
    model.add(Dense( activation="relu", input_dim = 10, units=5, kernel_initializer = "uniform"))
    model.add(Dense(activation="relu", units=5, kernel_initializer="uniform"))
    model.add(Dense(activation="sigmoid", units=1, kernel_initializer="uniform"))
    model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])
    monitor=EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=45,mode='auto')
    checkpointer=ModelCheckpoint(filepath="best_weights.hdf5", verbose=0, save_best_only=True)
    model.fit(x_train, y_train, validation_data=(x_val, y_val),
              callbacks=[monitor, checkpointer],verbose=0, epochs=1000)
    model.load_weights("best_weights.hdf5")
    predict_test = model.predict(x_test)
    predict_test=(predict_test>0.5)
    print("Başarı: {0:.4f}".format(metrics.accuracy_score(y_test, predict_test)))
```

3.8.2 Bulanık Derin Öğrenme Modeli

Derin Öğrenme modeli için tasarlanan mimari Bulanık derin öğrenme için de kullanılmaktadır. Bulanık Lojistik Regresyon modelinde olduğu gibi üçgensel sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için ayrı veri seti elde edilmiştir. Böylece her veri seti için ayrıca tahmini değerler oluşturulmuştur. Daha sonra ağırlık merkezi yöntemi kullanılarak durulama işlemi uygulanmış ve tek çıktı değeri elde edilmiştir.

3.8.3 Üçgensel Bulanık Veri ile Derin Öğrenme Modeli

Bu tez çalışmasında literatürden farklı olarak bulanık sayının minimum, tepe ve maksimum değerleri için ayrı veri seti elde etmek yerine bu değerler tek veri setinde birleştirilmiş ve analizler buna göre yapılmıştır. Ishibuchi ve Nii (1998) sinir ağlarının bulanıklaştırılması girişlerin ve ağırlıkların bulanık sayılara genişletilmesiyle mümkün olduğunu göstermişleridir. Bu çalışmada veriyi bulanık sayılara dönüştürerek ağın başarısına etkisi Derin öğrenme modeli üzerinde incelenmiştir (Ishibuchi ve Nii, 1998).

Parametre vektörü üçgensel bulanık sayı ile ifade edildiğinde ağın giriş nöron sayısı üç katına çıkmaktadır. Bu nedenle bu veri kümesi için en iyi mimariyi bulmak için aktivasyon fonksiyonları, farklı katman sayısı ve düğüm sayısı kombinasyonu ile tekrar denemeler yapılmıştır. Mimari seçimi için yapılan deneysel çalışmanın sonuçları Tablo 3.11'de verilmiştir. İlk katman için sonuçlara bakıldığında mimari derinleştikçe Sigmoid ve Tanh fonksiyonların başarısı düşmektedir. Bu nedenle üç katmanlı ağ için sadece Relu ve Softplus fonksiyonların katman sayısı ve düğüm sayısı kombinasyonuna bakılmıştır. Üç katmanlı ağ için sonuçlar incelendiğinde Relu fonksiyonunun başarısında düşüş, Softplus fonksiyonunun başarısında artış olduğu için sonraki denemelerde sadece Softplus fonksiyonların katman sayısı ve düğüm sayısı kombinasyonuna bakılmıştır.

Tablo 3.11 Üçgensel Bulanık Derin Öğrenme Mimari Seçimi

Katman Sayısı	Nöron Sayısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı	Eğitim Başarısı	Test Başarısı
Ortlama süre		**Relu 44 sn**		**Sigmoid 56 sn**		**SoftPlus 46 sn**		**Tanh 48 sn**	
1	**[25]**	0,987	0,939	0,992	0,919	0,993	0,954	0,998	0,924
1	**[20]**	0,993	0,905	0,994	0,940	0,993	0,956	1,000	0,929
1	**[15]**	0,996	0,931	0,992	0,950	0,987	0,960	0,999	0,899
1	**[10]**	0,984	0,934	0,987	0,913	0,991	0,952	0,994	0,896
1	**[5]**	0,979	0,942	0,979	0,943	0,974	0,928	0,995	0,927
Ortlama süre		**Relu 26 sn**		**Sigmoid 36 sn**		**SoftPlus 33 sn**		**Tanh 26 sn**	
2	**[25,20]**	0,966	0,885	0,908	0,851	0,990	0,959	0,994	0,919
2	**[25,15]**	0,982	0,904	0,908	0,851	0,994	0,956	0,995	0,923
2	**[25,10]**	0,959	0,881	0,808	0,769	0,992	0,943	0,946	0,913
2	**[25,5]**	0,985	0,895	0,863	0,811	0,991	0,937	0,995	0,927
2	**[20,15]**	0,978	0,899	0,910	0,844	0,993	0,948	0,994	0,926
2	**[20,10]**	0,974	0,895	0,820	0,772	0,993	0,966	0,953	0,888
2	**[20,5]**	0,971	0,865	0,819	0,772	0,992	0,940	0,904	0,886
2	**[15,10]**	0,975	0,892	0,794	0,739	0,993	0,946	0,956	0,931
2	**[10,5]**	0,973	0,871	0,764	0,739	0,992	0,951	0,784	0,763
Ortlama süre		**Relu 21 sn**		**Sigmoid - sn**		**SoftPlus 23 sn**		**Tanh - sn**	
3	**[25,20,15]**	0,982	0,899			0,997	0,971		
3	**[25,20,10]**	0,982	0,888			0,998	0,968		
3	**[25,20,5]**	0,980	0,894			0,991	0,960		
3	**[25,15,10]**	0,981	0,887			0,997	0,969		
3	**[25,15,5]**	0,983	0,908			0,993	0,962		
3	**[25,10,5]**	0,980	0,897			0,991	0,959		
3	**[20,15,10]**	0,988	0,899			0,997	0,971		
3	**[20,15,5]**	0,945	0,886			0,996	0,961		
3	**[20,10,5]**	0,936	0,863			0,994	0,954		
3	**[20,5,5]**	0,940	0,868			0,993	0,950		
3	**[15,10,5]**	0,936	0,884			0,990	0,959		
3	**[10,5,5]**	0,951	0,894			0,987	0,951		

Dört katmanlı mimarisi için düğüm sayıların kombinasyonlarına karar verirken sadece genel başarı ortalamasının üstünde başarı gösteren kombinasyonlar dikkate alınmıştır. Dört katmanlı ve beş mimarisi için yapılan deneysel çalışmaların sonuçları Tablo 3.12'de verilmiştir.

Tablo 3.12 Üçgensel Bulanık Derin Öğrenme Mimari Seçimi (3 ve 4 katmanlı)

Katman Sayısı	Nöron Sayısı	Eğitim Başarısı	Test Başarısı
Ortlama süre		**SoftPlus 23 sn**	
4	**[25,20,15,10]**	0,998	**0,978**
4	**[25,20,15,5]**	0,997	0,970
4	**[25,20,10,5]**	0,996	0,965
4	**[25,15,10,5]**	0,998	0,974
4	**[20,15,10,5]**	0,995	0,964

Katman Sayısı	Nöron Sayısı	Ortalama Başarı	Eğitim Başarısı
Ortlama süre		**SoftPlus 28 sn**	
5	**[25,20,15,10,5]**	0,934	0,946
5	**[25,20,15,5,2]**	0,840	0,858
5	**[25,20,10,5,2]**	0,873	0,888
5	**[25,15,10,5,2]**	0,881	0,892
5	**[20,15,10,5,2]**	0,843	0,860

Beş katmanlı mimari için elde edilen sonuçlarda performans düşüşü gözlemlendiği için daha derin katman için denemeler yapılmamıştır. Bu sonuçlara göre en başarılı sonuçları dört katmanlı olup sırayla 25, 20, 15 ve 10 nöron bulunan ağ vermiştir. Hesaplama süreleri incelendiğinde yine dört katmanlı mimari en kısa sürede sonuç üretmiştir.

3.8.4 Yamuk Bulanık Veri ile Derin Öğrenme Modeli

Bu çalışmada literatürden farklı olarak bulanık sayının minimum, tepe ve maksimum değerleri için ayrı veri seti elde etmek yerine bu değerler tek veri setinde birleştirilmiş ve analizler buna göre yapılmıştır. Parametre vektörü yamuk bulanık sayı ile ifade edildiğinde ağın giriş nöron sayısı dört katına çıkmaktadır. Bu nedenle bu veri kümesi için en iyi mimariyi bulmak için farklı katman sayısı ve düğüm sayısı kombinasyonu ile tekrar denemeler yapılmıştır. Üçgensel bulanık ve yamuk bulanık verini yapısı benzerlik gösterdiğinden dolayı model seçimi için önceki aşamada elde edilen tecrübeden yararlanılmıştır. Üçgensel bulanık veri için en iyi sonucu üç katmanlı ağ için Softplus fonksiyonu ürettiğinden sadece Softplus aktivasyon fonksiyonu denemeler yapılmıştır. Mimari seçimi için yapılan deneysel çalışmanın sonuçları Tablo 3.13'te verilmiştir.

Elde edilen sonuçlara göre en başarılı sonuçları altı katmanlı olup sırayla 35, 30, 25,20,15 ve 10 nöron bulunan ağ vermiştir. Hesaplama süreleri incelendiğinde en kısa sürede sonuç üreten beş katmanlı mimari olmuştur. Ancak başarı kriterine öncelik verildiğinden ve altı katmalı mimarinin hesaplama süresi beş katmanlı mimariye yakın olduğundan yine altı katmanlı mimari ile devam etmeye karar verilmiştir.

Tablo 3.13 Yamuk Bulanık Derin Öğrenme Mimari Seçimi

Katman Sayısı	Nöron Sayısı	Eğitim Başarısı	Test Başarısı
Ortalama süre		**SoftPlus 25 sn**	
3	**[35, 30, 25]**	0,995	0,961
3	**[35, 30, 20]**	0,993	0,935
3	**[35, 30, 15]**	0,992	0,947
3	**[35, 30, 10]**	0,994	0,948
3	**[35, 30, 5]**	0,954	0,899
3	**[35, 25, 20]**	0,995	0,944
3	**[35, 25, 15]**	0,954	0,907
3	**[35, 25, 10]**	0,958	0,903
3	**[35, 25, 5]**	0,955	0,896
3	**[35, 20, 15]**	0,957	0,907
3	**[35, 20, 10]**	0,989	0,923
3	**[35, 20, 5]**	0,954	0,902
3	**[35, 15, 10]**	0,957	0,905
3	**[35, 15, 5]**	0,946	0,897
3	**[35, 10, 5]**	0,952	0,888
3	**[30, 25, 20]**	0,989	0,955
3	**[30, 25, 15]**	0,954	0,910
3	**[30, 25, 10]**	0,901	0,901
3	**[30, 25, 5**	0,955	0,892
3	**[30, 20, 15]**	0,956	0,905
3	**[30, 20, 10]**	0,953	0,888
3	**[30, 20, 5]**	0,954	0,892
3	**[30, 15, 10]**	0,954	0,891
3	**[30, 15, 5]**	0,953	0,886
3	**[30, 10, 5]**	0,952	0,890
3	**[25, 20, 15]**	0,955	0,910
3	**[25, 20, 10**	0,955	0,894
3	**[25, 20, 5]**	0,951	0,889
3	**[25, 15, 10]**	0,957	0,908
3	**[25, 15, 5]**	0,944	0,899
3	**[25, 10 5]**	0,955	0,889
3	**[20, 15, 10]**	0,952	0,898
3	**[20, 15, 5]**	0,955	0,902
3	**[15, 10, 5]**	0,954	0,892
3	**[10, 5, 5]**	0,955	0,896

Katman Sayısı	Nöron Sayısı	Eğitim Başarısı	Test Başarısı
Ortalama süre		**SoftPlus 20 sn**	
4	**[35, 30, 25,20]**	0,994	0,961
4	**[35, 30, 25,15]**	0,992	0,968
4	**[35, 30, 25,10]**	0,990	0,953
4	**[35, 30, 25,5]**	0,982	0,918
4	**[35, 30, 20,15]**	0,994	0,966
4	**[35, 30, 20,10]**	0,981	0,930
4	**[35, 30, 20,5]**	0,978	0,932
4	**[35, 30, 15,10]**	0,984	0,919
4	**[35, 30, 15,5]**	0,978	0,922
4	**[35, 30, 10,5]**	0,975	0,909
4	**[30, 25, 20,15]**	0,989	0,962
4	**[30, 25, 20,10]**	0,982	0,935
4	**[30, 25, 20,5]**	0,980	0,922
4	**[30, 20, 10,5]**	0,976	0,911
Ortalama süre		**SoftPlus 17 sn**	
5	**[35, 30, 25,20,15]**	0,999	0,966
5	**[35, 30, 25,20,10]**	0,998	0,973
5	**[35, 30, 25,20,5]**	0,992	0,939
5	**[35, 30, 25,15,10]**	0,993	0,965
5	**[35, 30, 25,15,5]**	0,994	0,945
5	**[35, 30, 25,10,5]**	0,991	0,953
5	**[35, 30, 20,15,10]**	0,996	0,965
5	**[35, 30, 20,15,5]**	0,987	0,948
5	**[30, 25, 20,15,10]**	0,993	0,953
5	**[30, 25, 20,15,5**	0,987	0,943
Ortlama süre		**SoftPlus 20 sn**	
6	**[35, 30, 25,20,15,10]**	0,996	**0,982**
6	**[35, 30, 25,20,15,5]**	0,997	0,957
6	**[35, 30, 25,20,10,5]**	0,992	0,946
6	**[35, 30, 25,15,10,5]**	0,991	0,945
6	**[35, 30, 20,15,10,5]**	0,991	0,946
Ortalama süre		**SoftPlus 26 sn**	
7	**[35, 30, 25,20,15,10]**	0,903	0,865
7	**[35, 30, 25,20,15,5]**	0,948	0,890
7	**[35, 30, 25,20,10,5]**	0,894	0,829
7	**[35, 30, 25,15,10,5]**	0,945	0,873
7	**[35, 30, 20,15,10,5]**	0,942	0,882

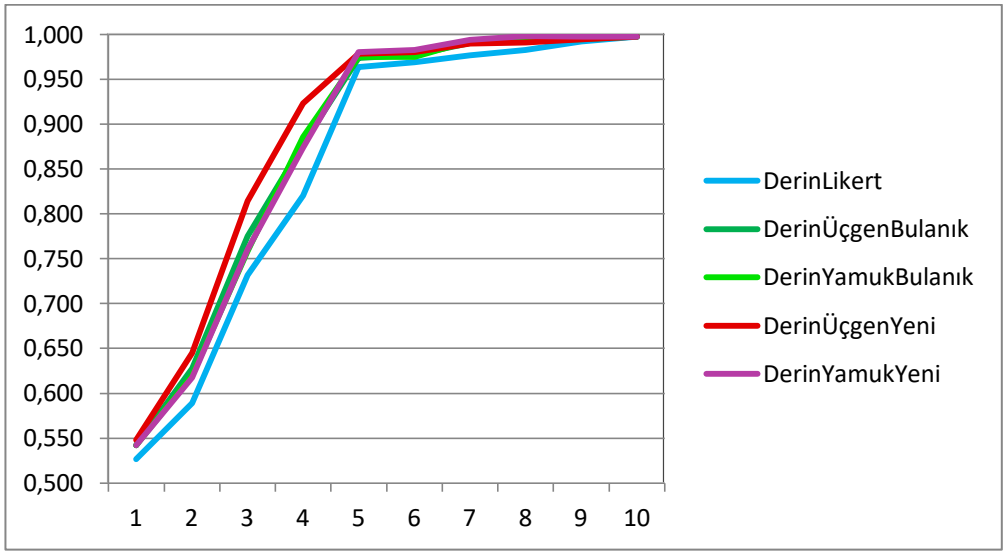
3.9. Derin Öğrenme Modellerinin Sınıflandırma Sonuçları

Derin Öğrenme Modellerinin Sınıflandırma performansını gözlemlemek için lojistik regresyon modelleri testinde kullanılan 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 örnek içeren bir yapay veri kümeleri kullanılmıştır. Deneysel çalışmada elde edilen modellerin performans sonuçları Ek 2'de verilmiştir.

Tüm veri kümeleri için yapılan sınıflandırma sonucunda elde edilen eğitim ve test için doğruluk oranların ortalama değerleri Tablo 3.14'te toplanmıştır. Diğer ölçütlerin ortalama değeri Ek 2'de verilmiştir. Bu tablo ile elde edilen grafikler Şekil 3.10'de verilmiştir.

Tablo 3.14 Derin Öğrenme Modellerinin Ortalama Test Doğruluk Değerleri

Veri Sayısı	Likert Tipi Derin Ağ	Üçgensel Bulanık Derin Ağ	Yamuk Bulanık Derin Ağ	Üçgensel Bulanık Likert Tipi Derin Ağ	Yamuk Bulanık Likert Tipi Derin Ağ
100	0,527	0,542	0,542	0,548	0,542
200	0,589	0,628	0,620	0,645	0,618
300	0,731	0,775	0,759	0,814	0,760
400	0,821	0,878	0,886	0,923	0,874
500	0,964	0,974	0,975	0,978	0,980
600	0,969	0,977	0,975	0,980	0,983
700	0,977	0,990	0,993	0,990	0,994
800	0,983	0,993	0,996	0,991	0,998
900	0,992	0,995	0,998	0,995	0,998
1000	0,997	0,998	0,997	0,998	0,998



Şekil 3.10 Derin Öğrenme Modellerinin Doğruluk Oranlar Grafiği

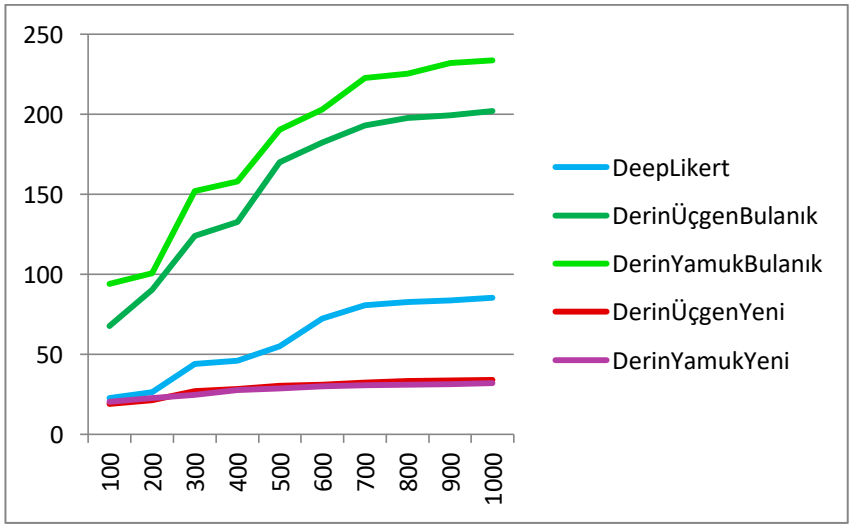
Tüm modellerin sınıflandırma başarısı veri sayısı ile doğru orantılı olarak artmaktadır. Çalışmada önerilen bulanık Likert tipi veri ile uygulanan her veri seti için başarılı sonuçlar üretmiştir. Ancak çalışmanın genelinde sonuçlar literatürde yer

alan yaklaşıma çok yakın olup, özellikle 500 örnek ve sonrası için aynı sonuçları üretmiştir. Veri kümelerinin oluşturulmasında karmaşık denklem kullanılmadığından veri sayısı artsa bile verideki ilişkilerin karmaşıklığında bir artış görülmemektedir. Bu nedenle veri sayısı artışı tekniklerin bu ilişkileri çözmesini kolaylaşmaktadır.

Bulanık Likert ölçeği modelin tahmin etme başarısını iyileştirmekle kalmayıp hesaplama hızına da katkı sağlamaktadır. Literatürde yer alan yaklaşımda üçgensel sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için model kurularak ayrı tahmini değerler oluşturulmuş, durulama tekniği kullanılarak kesin sayıya dönüştürülmektedir. Bu tez çalışmasında önerilen teknik tek model için hesaplama yapıldığından daha kısa süre de sonuç üretilmektedir. Hesaplama süreleri Tablo 3.15 ve grafiği Şekil 3.11'de verilmiştir.

Tablo 3.15 Derin Öğrenme Modellerinin Hesaplama Süreleri (sn)

Veri Sayısı	Likert Tipi Derin Ağ	Üçgensel Bulanık Derin Ağ	Yamuk Bulanık Derin Ağ	Üçgensel Bulanık Likert Tipi Derin Ağ	Yamuk Bulanık Likert Tipi Derin Ağ
100	22,82	67,81	93,98	19,02	20,33
200	26,47	90,26	100,80	21,19	22,55
300	44,09	124,11	151,86	27,17	24,75
400	46,11	132,59	158,04	28,45	27,69
500	54,95	170,17	190,45	30,37	28,78
600	72,26	182,41	203,06	31,07	29,88
700	80,57	192,89	222,54	32,49	30,60
800	82,74	197,60	225,47	33,25	31,02
900	83,57	199,33	231,86	33,67	31,36
1000	85,28	202,08	233,64	33,91	31,96



Şekil 3.11 Derin Öğrenme Modellerinin Hesaplama Süresi Grafiği

Elde edilen sonuçlara göre Likert tipi ölçeğin bulanık ölçeğe dönüştürülmesi derin öğrenme modelinin sınıflandırma hızını olumlu yönde etkilemektedir. Likert tipi veri ile derin öğrenme modelinin 100 örnek içeren küme için hesaplama süresi ortalama 22 saniye iken, 1000 örnek içeren küme için ortalana 85 saniyeye ulaşmıştır. Literatürde yer alan bulanık derin öğrenme modellerin 100 örnek içeren küme için hesaplama süresi ortalama 67 ve 93 saniye iken, 1000 örnek içeren küme için ortalana 202 ve 233 saniyeye ulaşmıştır. Derin öğrenme modelinin eğitimi için bulanık Likert tipi veri kullanıldığında ise 100 örnek içeren küme için hesaplama süresi ortalama 19 ve 20 saniye iken, 1000 örnek içeren küme için ortalana 34 ve 32 saniyeye ulaşmıştır. Önerilen yöntem en doğru ve hızlı sınıflandırma yapan yöntem olduğu sonucuna varılmıştır.

3.10. Karmaşık Veri ile Deneysel Çalışma

Bu veri seti ile işlem yapılmasının nedeni basit ve karmaşık modeller arasındaki farkı gözlemlemektir. Bölüm 3.9'da değişkenler arası ilişkilerin simetrik olduğu yani karmaşık olmayan veri kümeleriyle deneysel çalışma yapılmıştır. Bu bölümde ise (3.8)'de verilen denklem kullanılarak oluşturulan veri kümesiyle deneysel çalışma yapılmıştır.

$$X_1^2 + 2 * X_2 + X_3 + 4 * X_4 + X_5 + X_6 + 5 * X_7 + X_8 + X_9 + X_{10} \quad (3.8)$$

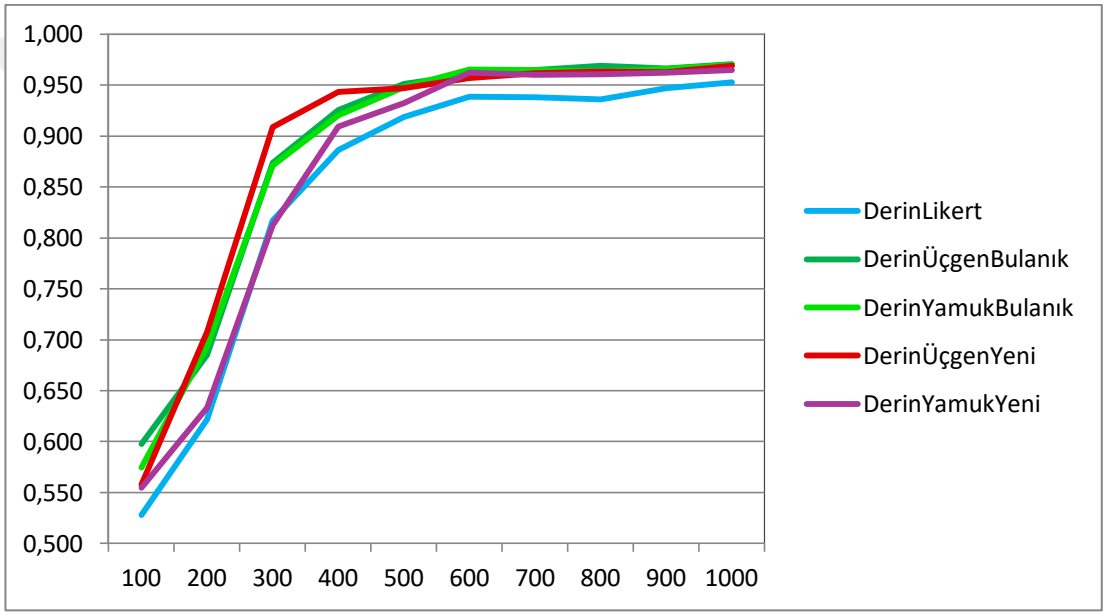
Bağımlı değişken için dengeli sınıf oluşturulması hedeflendiğinden Likert ölçeğinde yer alan üç değeri kullanılmıştır. (3.8)'de verilen ifade üç değeri için hesaplandığında sonuç 60 olarak bulunmuştur. 60 değeri eşik kabul edilerek 60 ve üstü memnun=1, 60 altı ise memnun değil=0 olarak kabul edilmiştir.

Modellerinin Sınıflandırma performansını gözlemlemek için her biri 30 adet olmak üzere 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 örnek içeren bir yapay veri kümeleri oluşturulmuş ve eğitim için kullanılmıştır. Deneysel çalışmada elde edilen modellerin performans sonuçları Ek 3'te verilmiştir.

Tüm veri kümeleri için yapılan sınıflandırma sonucunda elde edilen test ortalama değerleri Tablo 3.16'te toplanmıştır. Diğer ölçütlerin ortalama değerleri Ek 3'de verilmiştir. Bu tablodan elde edilen grafikler Şekil 3.12'de verilmiştir.

Tablo 3.16 Karmaşık Veri ile Test Edilen Modellerinin Test Doğruluk Oranları

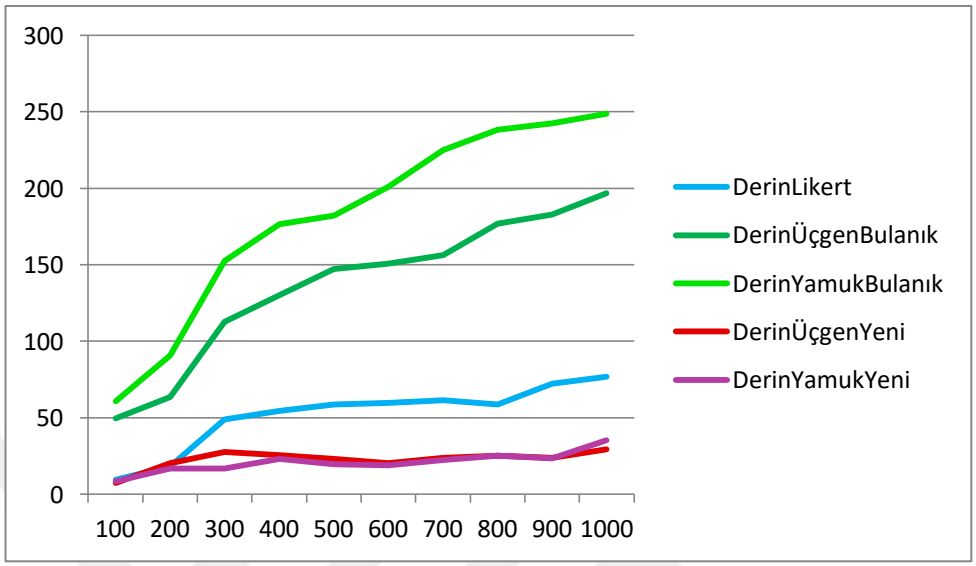
Veri Sayısı	Likert Tipi Derin Ağ	Üçgensel Bulanık Derin Ağ	Yamuk Bulanık Derin Ağ	Üçgensel Bulanık Likert Tipi Derin Ağ	Yamuk Bulanık Likert Tipi Derin Ağ
100	0,528	0,598	0,574	0,558	0,554
200	0,622	0,685	0,693	0,708	0,633
300	0,817	0,874	0,871	0,909	0,813
400	0,886	0,926	0,920	0,943	0,909
500	0,919	0,951	0,948	0,947	0,932
600	0,939	0,961	0,965	0,957	0,962
700	0,938	0,965	0,965	0,962	0,960
800	0,936	0,969	0,963	0,963	0,961
900	0,947	0,967	0,966	0,963	0,962
1000	0,953	0,970	0,970	0,969	0,965



Şekil 3.12 Karmaşık Veri ile Test Edilen Modellerinin Doğruluk Oranlarının Grafiği

Bu deneysel çalışmada elde edilen sonuçlara göre verinin karmaşık olması modellerin tahmin etme başarısını olumsuz yönde etkilediği görülmüştür. Simetrik ve dengeli veri için tahmin etme başarısı %99,8 iken, karmaşık veri için en iyi sonuç veren modelin başarın oranı %96,9 olduğu görülmüştür. Grafikte mavi renkle gösterilen eğri Likert tipi veri ile sınıflandırma yapan derin öğrenme mimarisi olup diğer modellere göre daha başarısız olduğu görülmüştür. Grafikte kırmızı renkle gösterilen üçgensel bulanık Likert tipi veri ile yapılan sınıflandırma literatürde yer alan yaklaşımla aynı sonuçları üretmiştir. Grafikte mor renkle gösterilen yamuk bulanık Likert tipi veri ile yapılan sınıflandırma 100-500 örnek için arası bu başarının gerisinde kalırken, ancak 500 örnekten sonra aynı düzeye ulaşmaktadır.

Verinin karmaşıklığı derin öğrenme modellerinin hesaplama hızını nasıl etkilediğin incelemek için Şekil 3.13'te verilen grafik oluşturulmuştur. Bu grafik 3.11'de verilen grafik ile karşılaştırıldığında verinin karmaşıklığı modellerin hesaplama hızını fazla etkilemediği görülmüştür.



Şekil 3.13 Karmaşık veri ile Derin Öğrenme Modellerinin Hesaplama Süresi Grafiği

3.11. Dengesiz Sınıflar İçeren Veri ile Deneysel Çalışma

Bir önceki bölümde dengeli sınıf içeren karmaşık veri oluşturulmuş ve bu veri kümeleriyle deneysel çalışma yapılmıştır. Bu bölümde ise bağımlı değişken için dengesiz sınıf oluşturulmuştur. Likert ölçeğinde yer alan üç değeri yerine, bulanık sayının üst değeri olan 3,5 sayısı kullanılmıştır. (3.7)'de verilen ifade 3,5 değeri için hesaplandığında sonuç yaklaşık 72 olarak bulunmuştur. 72 değeri eşik kabul edilerek 72 ve üstü memnun=1, 72 altı ise memnun değil=0 olarak kabul edilmiştir.

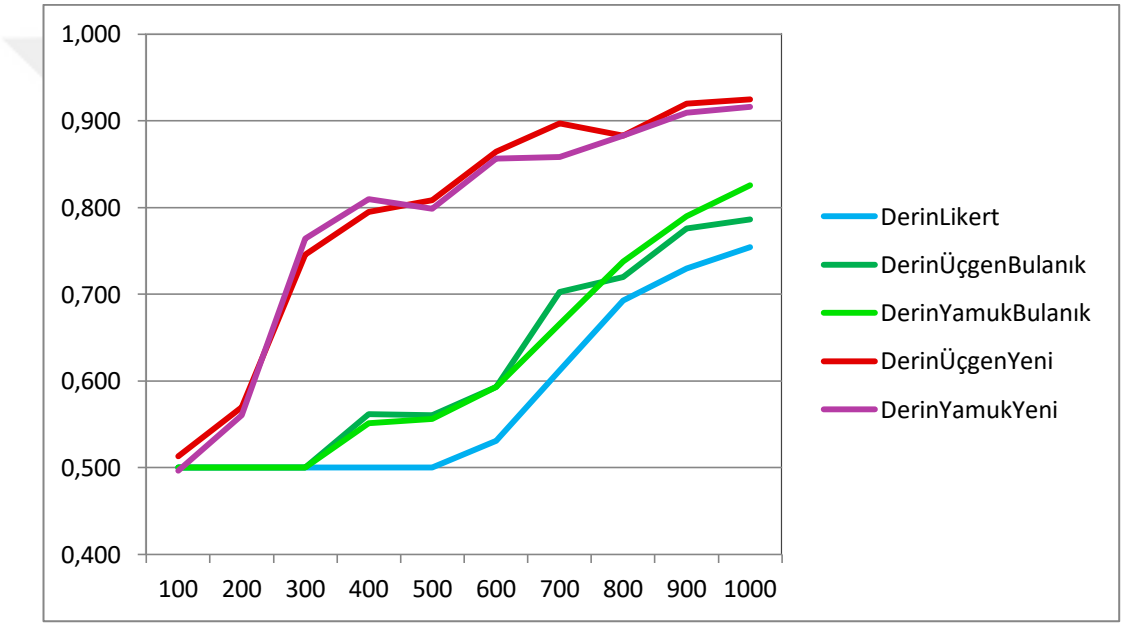
Modellerinin Sınıflandırma performansını gözlemlemek için her biri 30 adet olmak üzere 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 örnek içeren bir yapay veri kümeleri oluşturulmuş ve eğitim için kullanılmıştır. Deneysel çalışmada elde edilen modellerin performans sonuçları Ek 4'de verilmiştir.

Veri kümelerinin dengesiz sınıf içermesi sebebiyle doğruluk oranları yerine başarı ölçütü olarak ROC eğrisi altındaki alanı dikkate alarak değerlendirmeler yapılmıştır.

Tüm veri kümeleri için yapılan sınıflandırma sonucunda elde edilen test için ROC alanlarının ortalama değerleri Tablo 3.17'de toplanmıştır. Diğer ölçütlerin ortalama değerleri Ek 4'de verilmiştir. Bu tablodan elde edilen grafikler Şekil 3.14'de verilmiştir.

Tablo 3.17 Dengesiz sınıf içeren Veri ile Test Edilen Modellerinin ROC alanı değerleri

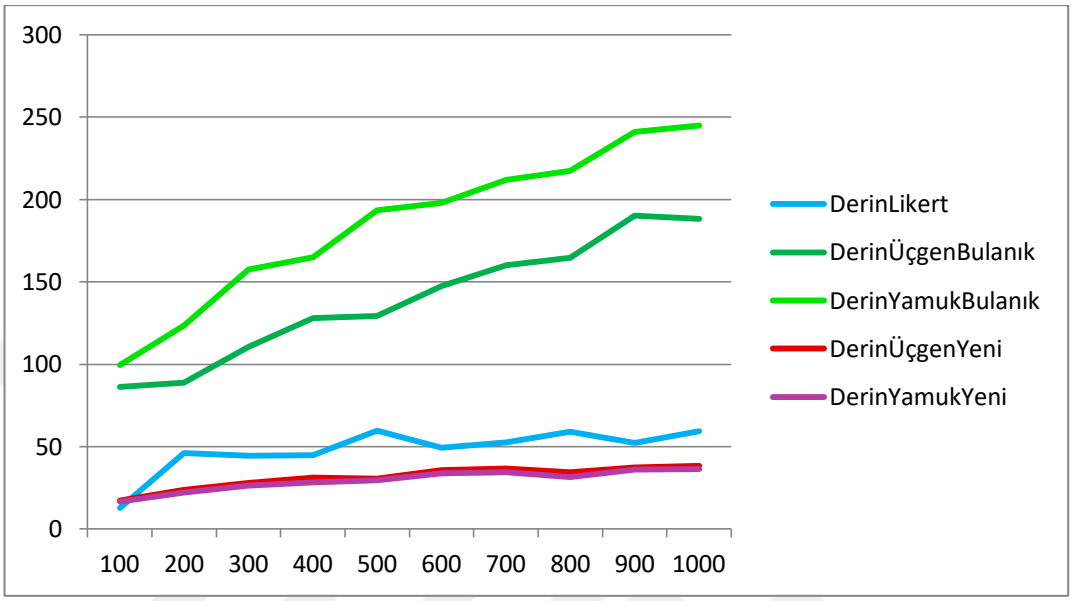
Veri Sayısı	Likert Tipi Derin Ağ	Üçgensel Bulanık Derin Ağ	Yamuk Bulanık Derin Ağ	Üçgensel Bulanık Likert Tipi Derin Ağ	Yamuk Bulanık Likert Tipi Derin Ağ
100	0,500	0,500	0,500	0,513	0,497
200	0,500	0,500	0,500	0,570	0,561
300	0,500	0,500	0,500	0,746	0,764
400	0,500	0,561	0,551	0,795	0,810
500	0,500	0,560	0,556	0,809	0,799
600	0,531	0,593	0,593	0,864	0,857
700	0,612	0,703	0,666	0,897	0,859
800	0,693	0,720	0,738	0,883	0,883
900	0,730	0,776	0,790	0,920	0,909
1000	0,754	0,787	0,826	0,925	0,916



Şekil 3.14 Dengesiz Sınıf İçeren Veri ile Testin ROC Alanı Değerlerinin Grafiği

Şekilde 3.14'te verilen sonuçlara göre Likert ölçeğinin üçgensel ya da yamuk bulanık sayılar kullanılarak Likert ölçeğine dönüştürülmesi derin öğrenme modellerinin sınıflandırma performanslarında %20 üzerinde bir artış sağlamıştır. Bunun yanında tüm modeller 100-200 örnek içeren veri kümeleri için sınıflandırma başarısı %60 altında kalmaktadır. Derin öğrenme modellerinin başarılı olması için veri mikarının önemli olduğunu gçstermektedir. Bulanık Likert tipi veri derin öğrenme modeli ile sınıflandırdığında 300 örnekten itibaren %75, 900 örnekten itibaren %90 başarı göstererek en iyi sınıflandırma yapan teknik olduğu görülmüştür. Literatürde yer alan bulanık derin öğrenme modelleri ise derin öğrenme modeline göre daha iyi performans göstermesine rağmen önerilen tekniğe göre başarısız olduğu görülmüştür.

Dengesiz sınıf içeren veri derin öğrenme modelleri ile sınıflandırıldığında hesaplama hızını nasıl etkilediğini incelemek için Şekil 3.15'te verilen grafik oluşturulmuştur. Bu grafik 3.11 ve 3.13'te verilen grafik ile karşılaştırıldığında verinin dengesiz sınıf içermesi modellerin hesaplama hızını etkilemediği görülmüştür.



Şekil 3.15 Dengesiz Veri ile Derin Öğrenme Modellerinin Hesaplama Süresi Grafiği

Dengesiz sınıf içeren veri ile yapılan deneysel çalışmanın sonuçlarına göre önerilen tekniğin en hızlı ve en doğru sınıflandırma yapan teknik olduğu görülmüştür.

3.12. Veriye Gürültü Eklenmesi

Bazı modeller gürültüye hassastır. Bu çalışmada gürültü içeren yapay veri ile modeller test edilerek, sonuçlar karşılaştırılmıştır. Gürültü eklemek için "gurultu_ekle" fonksiyon yazılmıştır. Bu fonksiyon için gürültü oranını belirlemek için "gurultu=0.05" parametresi tanımlanmıştır. Aşağıdaki kod içerisinde kırmızı dikdörtgenle işaretli olan satır ilave edilmiştir. Yani öncelikle yapay veri üretilmekte, daha sonra içerisine gürültü eklenmektedir.

```
def yapay_veri(parametre=10,ornek_say=100,gurultu=0.05):
    import numpy as np
    sonuc_listesi=[]
    parametre_listesi=[]
    for i in range(parametre):
        parametre_listesi.append(truncnorm(-1,1,loc=3,
                                 scale=2).rvs(ornek_say).round().astype(int))
    parametre_listesi=np.array(parametre_listesi)
    parametre_listesi=parametre_listesi.T
    for i in range(ornek_say):
        sonuc=sum(parametre_listesi[i])
        sonuc_listesi.append(sonuc)

    sonuc_listesi=np.array(sonuc_listesi)
    cutoff=int(parametre*3)
    sonuc_ikili=sonuc_listesi>cutoff
    sonuc_ikili=np.array(sonuc_ikili,dtype=int)
    sonuc_ikili=gurultu_ekle(sonuc_ikili,p=gurultu)
    sonuc_ikili=sonuc_ikili.reshape(ornek_say,1)
    x=np.hstack((parametre_listesi,sonuc_ikili))
    return (x)
```

Aşağıda "gurultu_ekle" fonksiyonunu Python kodları yer almaktadır. Burada gürültü değerini olasılık kabul edilerek 1 veya 0 değerleri seçilmektedir. Gürültü değerini "0,05" seçildiği için "0,95" oranında sıfır, "0,05" oranında bir sayısı üretilerek bir gürültü dizisi oluşturulmuştur. Daha sonra bu dizi ile bağımlı değişken dizisine özel veya (xor) işlemi uygulanmıştır.

```
def gurultu_ekle(y,p):
    import numpy as np
    gurultu_dizisi=[]
    for i in range(len(y)):
        sayi=np.random.choice([1,0],p=[p,1-p])
        gurultu_dizisi.append(sayi)
    gurultu_dizisi=np.array(gurultu_dizisi)
    return np.array(np.logical_xor(y,gurultu_dizisi),dtype=int)
```

Tablo 3.18'de uygulanan "xor" işlemi için örnek verilmiştir. Gürültü dizisinde 0 olduğu sürece bağımlı değişkene herhangi bir etkisi olmayıp, bir olduğunda bağımlı değişkenin değerinin tersini almaktadır. Böylece yapay veriye bir değerlerinin sayısı kadar gürültü eklenmiş olacak.

Tablo 3.18 Xor İşlemi

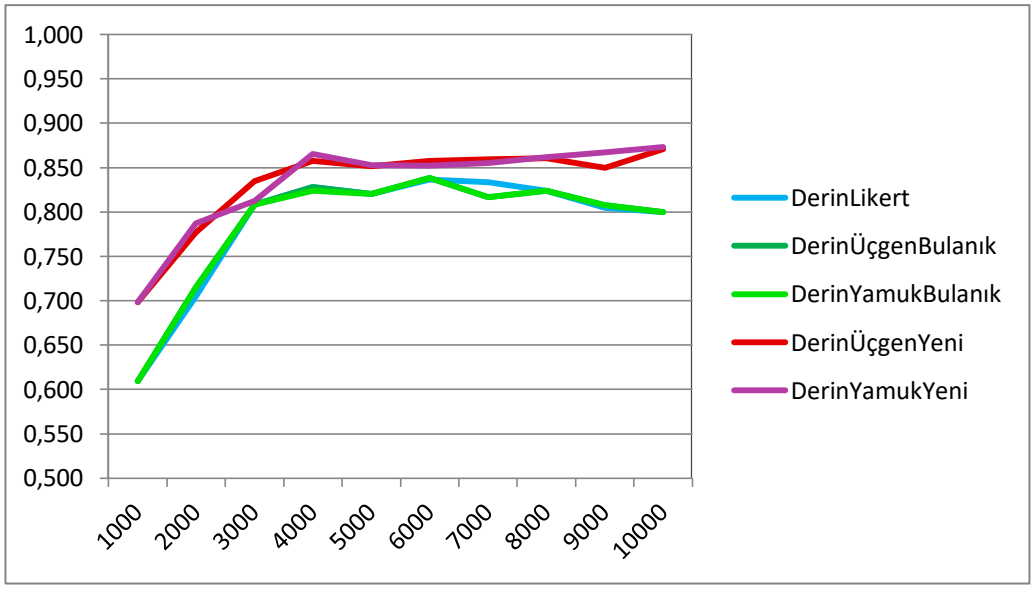
Yapay Veri	Gürültü Dizisi	Sonuç Dizisi
1	0	1
0	0	0
1	1	0
1	0	1
0	0	0
1	0	1
0	0	0
0	0	0
0	1	1
0	0	0

Bir deneysel çalışmada dengesiz sınıf içeren karmaşık veriye ek olarak %5 oranında gürültü eklenerek veri kümeleri oluşturulmuş ve bu veri kümeleriyle deneysel çalışma yapılmıştır. Modellerinin sınıflandırma performansını gözlemlemek için her biri 30 adet olmak üzere 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 örnek içeren bir yapay veri kümeleri oluşturulmuş ve eğitim için kullanılmıştır. Ancak 1000 örnek içeren veri kümeleri hariç hiçbir kümede %70 üstünde başarı elde edilememiştir. Çalışmadaki tüm modeller için geçerli olmak üzere, veri kümelerinin karmaşık ilişki içerirken aynı zamanda dengesiz sınıf ve gürültü içermesi sınıflandırma başarısı olumsuz yönde etkilediği ve başarılı bir sıralama için daha fazla veriye ihtiyaç duyulduğu görülmüştür. Bu nedenle yöntemlerin performans farklarını daha iyi gözlemlemek amacıyla her biri 30 adet olmak üzere 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000 örnek içeren bir yapay veri kümeleri oluşturulmuş ve eğitim için kullanılmıştır. Deneysel çalışmada elde edilen modellerin performans sonuçları Ek 5'te verilmiştir.

Tüm veri kümeleri için yapılan sınıflandırma sonucunda elde edilen eğitim ve test için ROC alanlarının ortalama değerleri Tablo 3.19'da toplanmıştır. Diğer ölçütlerin ortalama değerleri Ek 5'te verilmiştir. Bu tablodan elde edilen grafikler Şekil 3.16'da verilmiştir.

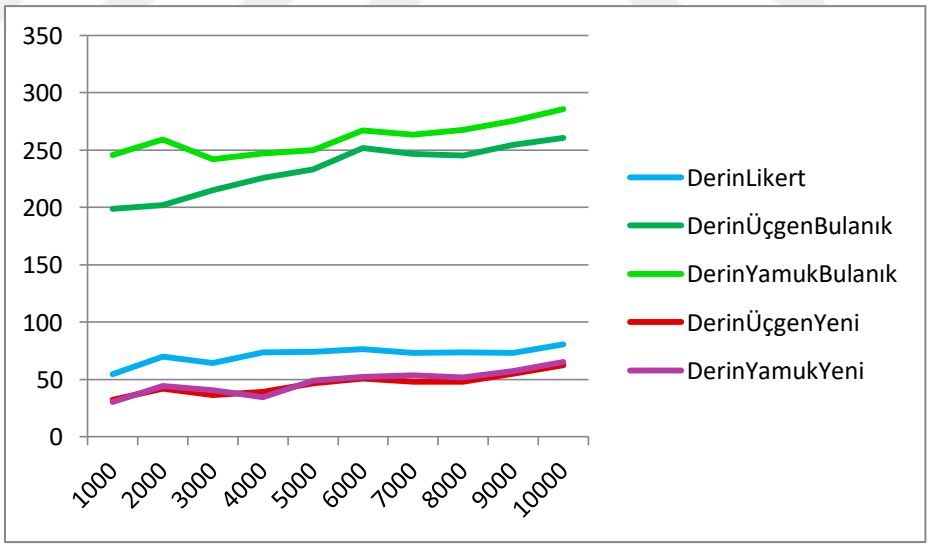
Tablo 3.19 Gürültü İçeren Veri ile Test Edilen Modellerinin ROC Alanı Değerleri

Veri Sayısı	Likert Tipi Derin Ağ	Üçgensel Bulanık Derin Ağ	Yamuk Bulanık Derin Ağ	Üçgensel Bulanık Likert Tipi Derin Ağ	Yamuk Bulanık Likert Tipi Derin Ağ
1000	0,609	0,609	0,609	0,698	0,698
2000	0,705	0,715	0,715	0,777	0,787
3000	0,808	0,808	0,808	0,835	0,813
4000	0,828	0,828	0,824	0,857	0,865
5000	0,820	0,820	0,820	0,852	0,853
6000	0,837	0,838	0,838	0,858	0,852
7000	0,834	0,817	0,817	0,859	0,855
8000	0,824	0,824	0,824	0,860	0,862
9000	0,804	0,807	0,807	0,850	0,867
10000	0,800	0,800	0,800	0,871	0,873



Şekil 3.16 Gürültü İçeren Veri ile Test Edilen Modellerinin ROC Alanı Değerleri Grafiği

Şekilde 3.16'da verilen sonuçlara göre Likert ölçeğinin üçgensel ya da yamuk bulanık sayılar kullanılarak Likert ölçeğine dönüştürülmesi derin öğrenme modellerinin sınıflandırma performanslarında artışa neden olmaktadır. Literatürde yer alan bulanık derin öğrenme modelleri ise derin öğrenme modeline göre daha iyi performans göstermesine rağmen önerilen tekniğe göre başarısız olduğu görülmüştür.



Şekil 3.17 Gürültü İçeren Veri ile Test Edilen Modellerinin Hesaplama Süreleri Grafiği

Dengesiz sınıf ve gürültü içeren veri ile derin öğrenme modellerinin sınıflandırma hızını incelemek için Şekil 3.17'de verilen grafik oluşturulmuştur. Verinin dengesiz sınıf ve gürültü içermesi modellerin hesaplama hızını az etkilediği görülmüştür.

SONUÇ

Çalışmada derin öğrenme ve bulanık mantık tekniklerinin entegrasyonu sonucunda modellerin performansı memnuniyet tahmin problemi üzerinde test edilmiştir. Memnuniyet tahmininde yaygın olarak kullanılan Likert ölçeği, bulanık sayılar kullanılarak bulanık Likert ölçeğine dönüştürülmüş ve analizlerde kullanılmıştır. Deneysel çalışmada farklı koşullar altında sınıflandırma modellerinin performansının kapsamlı bir şekilde araştırılması için yapay veri üretilmiştir. Etkinliği birçok parametreye bağlı olan derin öğrenme mimarisinin başarısının tesadüfî olmadığını göstermek amacıyla üretilen yapay veri geleneksel teknik olan lojistik regresyon tekniği ile test edilmiştir. Üretilen Likert tipi veri ile bulanık Likert tipi veri lojistik regresyon modeli kullanılarak sınıflandırılma yapıldığında verinin bulanık hale dönüştürme işleminin modelin hem başarısını hem hızını arttrdığı görülmüştür. Yapılan testler hem üçgensel hem de yamuk bulanık sayılar kullanılarak yapılmıştır.

100 adet Likert tipi veri ile yapılan lojistik regresyon sınıflandırmasında başarı kriteri doğruluk oranı olarak seçildiğinde, eğitim başarısı %75 iken test başarısının %52 olduğu görülmüştür. Karışıklık matrisi incelendiğinde de yanlış sınıfa atanan verilerin sayısı oldukça fazladır. Buradan test sonucunu dikkate alarak eğitim sürecini başarısız olduğu ve modeli test verisini sınıflandıramadığı sonucuna ulaşılmıştır. Aynı analizler 200-1000 arası örnek için de yapılmış ve Tablo 3.8'de verilmiştir. 200-1000 adet veri kümeleri ile yapılan sınıflandırmada test başarısı %67'den %92'ye yükselmiştir. Örnek sayısı arttıkça başarının arttığı görülmüştür.

100 adet Üçgensel bulanık Likert tipi veri kümeleri ile yapılan lojistik regresyon sınıflandırmanın performans değerlerine göre eğitim başarısı %85 iken test başarısı %65 olduğu görülmüştür. Karışıklık matrisi incelendiğinde yanlış sınıfa atanan verilerin sayısı lojistik regresyona göre azdır. Test başarısı yüksek olmasa da model bir sınıflandırma yapmıştır. Aynı analizler 200-1000 arası örnek için de yapılmış ve Tablo 3.8'de verilmiştir. 200-1000 adet veri kümeleri ile yapılan sınıflandırmada test başarısı %84'den %98'ye yükselmiştir. Örnek sayısı arttıkça başarının arttığı görülmüştür.

Keras kütüphanesinde lojistik regresyon modelinin başarısını iyileştirmek için regulasyon parametreleri gibi farklı fonksiyon ayarları mevcuttur. Ancak pilot çalışmasındaki amaç veri kümesinin bulanık kümeye dönüştürülmesi durumunda

performans artışının olup olmayacağını gözlemlemek olduğundan, fonksiyonun varsayılan ayarları kullanılarak test işlemi yapılmıştır.

Çalışmada önerilen bulanık Likert tipi veri ile uygulanan lojistik regresyon her veri seti daha başarılı sonuçlar üretmiştir. Özellikle veri kümesinin küçük olduğu durumlarda büyük avantaj sağlamaktadır.

Literatürde yer alan yaklaşımda üçgensel sayıyı ifade eden minimum değeri, tepe değeri ve maksimum değeri için ayrı lojistik regresyon modeli kurularak ayrı tahmini değerler oluşturulmuştur. Yani çıktı da bulanık üçgensel olarak üretilmektedir. Daha sonra durulama tekniği kullanılarak kesin sayıya dönüştürülmektedir. Bu tez çalışmasında önerilen yaklaşımla tek lojistik regresyon için hesaplama yapıldığından daha kısa süre de sonuç üretilmektedir.

Likert tipi verinin bulanık veriye dönüştürülmesinin modelin başarısını arttırdığı ve verinin derin öğrenme modelinin testi için uygun olduğu sonucuna varıldıktan sonra derin öğrenme mimarisinin tasarımı aşamasına geçilmiştir. Likert tipi veri ile bulanık Likert tipi verinin parametre sayılarında farklılık söz konusu olduğunda her mimari için ayrı ayrı mimari seçim prosedürü takip edilmiştir. Likert tipi veri ile derin öğrenme modeli için sırayla beş ve iki nöron içeren iki gizli katmanlı mimari seçilmiştir. Bu mimaride gizli katmanların aktivasyon fonksiyonları için Sigmoid, Tanh, Relu ve SoftPlus gibi fonksiyonları denenmiş ve Relu fonksiyonu seçilmiştir.

Üçgensel bulanık Likert tipi veri ile derin öğrenme modeli için sırayla 25, 20, 15 ve 10 nöron içeren dört gizli katmanlı mimari seçilmiştir. Bu mimaride gizli katmanların aktivasyon fonksiyonları için Sigmoid, Tanh, Relu ve SoftPlus gibi fonksiyonları denenmiş ve SoftPlus fonksiyonu seçilmiştir.

Yamuk bulanık Likert tipi veri ile derin öğrenme modeli için sırayla 35, 30, 25, 20, 15 ve 10 nöron içeren altı gizli katmanlı mimari seçilmiştir. Bu mimaride gizli katmanların aktivasyon fokksiyonları için Sigmoid, Tanh, Relu ve SoftPlus gibi fonksiyonları denenmiş ve SoftPlus fonksiyonu seçilmiştir.

Daha iyi mimari seçmek için aktivasyon, nöron ve katman sayıları dışında birçok hiperparametre kombinasyonunun denenmesi gerekmektedir. Bu yol ile hiperparametre ayarı ve ideal mimari seçimi tecrübe ve yetenek gerektiren uzun zaman ayrılması gereken bir iştir. Mimariyi tasarlayan kişi aynı tip uygulama ve mimaride çalışmış olması durumunda, doğru strateji belirleyerek daha hızlı sonuca ulaşabilmektedir (Goodfellow vd. 2016: 432). Bu nedenle üçgen bulanık sayılar için

mimari seçiminde elde edilen tecrübeden üçgen bulanık sayılar için kullanımlacak mimari seçiminde yararlanılmıştır. Ancak çalışmanın amacı sonucu tahmin edecek en iyi mimariyi bulmak olmadığı için mimari seçimi için en temel hiperparametrelere bakılarak modeller için deney ortamının benzerliği önemsenmiştir. Çalışmada modellerin benzer şartlarda farklı ölçek kullanması durumunda iyileştirme olup olmadığı incelenmesi amaçlanmıştır.

Tüm modellerin sınıflandırma başarısının veri sayısı ile doğru orantılı olarak artığı görülmüştür. Çalışmada önerilen bulanık Likert tipi veri ile uygulanan her veri seti için başarılı sonuçlar üretilmiştir. Ancak çalışmanın genelinde sonuçlar literatürde yer alan yaklaşıma çok yakın olup, özellikle 500 örnek ve sonrası için aynı sonuçları üretmiştir.

Elde edilen sonuçlara göre Likert tipi ölçeğin bulanık ölçeğe dönüştürülmesi Derin öğrenme modelinin sınıflandırma hızını olumlu yönde etkilemektedir. Likert tipi veri ile derin öğrenme modelinin 100 örnek içeren küme için hesaplama süresi ortalama 22 saniye iken, 1000 örnek içeren küme için ortalana 85 saniyeye ulaşmıştır. Literatürde yer alan bulanık derin öğrenme modellerin 100 örnek içeren küme için hesaplama süresi ortalama 67 ve 93 saniye iken, 1000 örnek içeren küme için ortalana 202 ve 233 saniyeye ulaşmıştır. Derin öğrenme modelinin eğitimi için bulanık Likert tipi veri kullanıldığında ise 100 örnek içeren küme için hesaplama süresi ortalama 19 ve 20 saniye iken, 1000 örnek içeren küme için ortalana 34 ve 32 saniyeye ulaşmıştır. Önerilen yöntem en doğru ve hızlı sınıflandırma yapan yöntem olduğu sonucuna varılmıştır. Bu problem için hesaplama süresi çok önemli değildir. Ancak derin öğrenme uygulamalarının arasında hesaplama süresinin çok önemli olduğu durumlar da mevcuttur. Böyle durumlarda önerilen tekniğin önemi artacağı düşünülmektedir.

Farklı koşullar altında modellerin sınıflandırma performansını kapsamlı bir şekilde araştırmak için önce kuadratik denklem kullanılarak veri daha karmaşık veriye dönüştürülmüştür. Bu veri kümeleri kullanılarak modelleri test edilmiş ve yorumlanmıştır. Deneysel çalışmanın bu aşamasında elde edilen sonuçlara göre verinin karmaşık olması modellerin tahmin etme başarısını olumsuz yönde etkilediği görülmüştür. Simetrik ve dengeli veri için tahmin etme başarısı %99,8 iken, karmaşık veri için en iyi sonuç veren modelin başarın oranı %96,9 olduğu görülmüştür.

Daha sonra veri sınıf dengesizliği ve gürültü içerecek şekilde üretilmiş veriyle modeller daha zorlu hale getirilerek analiz edilmiştir. Bu sonuçlara göre Likert ölçeğinin üçgensel ya da yamuk bulanık sayılar kullanılarak Likert ölçeğine dönüştürülmesi derin öğrenme modellerinin sınıflandırma performanslarında %20 üzerinde bir artış sağlamıştır. Bunun yanında tüm modeller 100-200 örnek içeren veri kümeleri için sınıflandırma başarısı %60 altında kalmaktadır. Derin öğrenme modellerinin başarılı olması için veri mikarının önemli olduğunu göstermektedir. Bulanık Likert tipi veri derin öğrenme modeli ile sınıflandırdığında 300 örnekten itibaren %75, 900 örnekten itibaren %90 başarı göstererek en iyi sınıflandırma yapan teknik olduğu görülmüştür. Literatürde yer alan bulanık derin öğrenme modelleri ise derin öğrenme modeline göre daha iyi performans göstermesine rağmen önerilen tekniğe göre başarısız olduğu görülmüştür.

Dengesiz sınıf içeren veri derin öğrenme modelleri ile sınıflandırıldığında ve modeller karşılaştırıldığında verinin dengesiz sınıf içermesi modellerin hesaplama hızını etkilemediği görülmüştür.

Çalışmadaki tüm modeller için geçerli olmak üzere, veri kümelerinin karmaşık ilişki içerirken aynı zamanda dengesiz sınıf ve gürültü içermesi sınıflandırma başarısı olumsuz yönde etkilediği ve başarılı bir sıralama için daha fazla veriye ihtiyaç duyulduğu görülmüştür. Bu nedenle yöntemlerin performans farklarını daha iyi gözlemlemek amacıyla her biri 30 adet olmak üzere 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000 örnek içeren bir yapay veri kümeleri oluşturulmuş ve eğitim için kullanılmıştır. Bu sonuçlara göre Likert ölçeğinin üçgensel ya da yamuk bulanık sayılar kullanılarak Likert ölçeğine dönüştürülmesi derin öğrenme modellerinin sınıflandırma performanslarında artışa neden olmaktadır. Literatürde yer alan bulanık derin öğrenme modelleri ise derin öğrenme modeline göre daha iyi performans göstermesine rağmen önerilen tekniğe göre başarısı olduğu görülmüştür. Verinin dengesiz sınıf ve gürültü içermesi modellerin hesaplama hızını az etkilediği görülmüştür.

Derin öğrenme modelinin tasasımı sırasında tüm etkenlerin dikkatle incelenmesi ve veriye uygun tekniklerin kullanılması modelin başarısı için oldukça önemlidir. Böyle durumda konuyu kapsamlı anlatak kaynakları kullanılması önemlidir. Derin öğrenme konusunda Türkçe literatürün sınırlı olması sebebiyle Türkçe literatür katkı sağlayacağı düşünülmektedir.

KAYNAKÇA

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, A., Corrado, G. S., Zheng, X. (2015). *Tensorflow: Largescale machine learning on heterogeneous systems.* Google, ABD.

Agrawal, A. "Loss functions and optimization algorithms". https://medium.com: https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c (Erişim Tarihi: 18.02.2019)

Albayrak, A. S., & Koltan Yılmaz, Ş. (2009). "Veri madenciliği: karar ağacı algoritmaları ve İMKB verileri üzerine bir uygulama". *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 14(1)*: 31-52.

Albon, C. (2018). *Machine Learning with Python Cookbook.* O'Reilly Media, Inc, ABD.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688.* The Theano Development Team, ABD

anakonda.og. "Spyder*"*. https://anaconda.org/anaconda/spyder (Erişim Tarihi: 24.04.2019)

Ba, J., & Caruana, R. (2014). "Do deep nets really need to be deep?" *In Advances in neural information processing systems:* 2654-2662.

Bengio, Y. (2009). *Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1*). Now publishers inc., Boston.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *In Neural networks: Tricks of the trade.* Springer, Berlin, Heidelberg.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, Singapore.

Boden, M. A. (1996). *Artificial intelligence.* Academic Press, London.

Brownlee, J. "How to configure the number of layers and nodes in a neural Network". https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/ (Erişim Tarihi: 27.03.2019)

Canbaz, N. (2015). *Nesneye Dayalı Yazılımların Tasarım Kalitesini Ölçmek için Öğrenme Tabanlı Bir Yöntem.* İstanbul Teknik Üniversitesi, Bilişim Enstitüsü, Bilgisayar ve Bilişim Fakültesi, Yüksek Lisans Tezi.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2016). "Fast and accurate deep network learning by exponential linear units (elus)". *ICLR 2016*: 1-14.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). "Torch7: A matlab-like environment for machine learning". *In BigLearn, NIPS workshop:* 1-6.

Cong, P., Wang, C., Ren, Z., Wang, H., Wang, Y., & Feng, J. (2016). "Unsatisfied customer call detection with deep learning". *In 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP):* 1-5.

Cook, D. (2016). *Practical machine learning with H2O: powerful, scalable techniques for deep learning and AI.* . O'Reilly Media, Inc, ABD.

Coppin, B. (2004). *Artificial intelligence illuminated.* . Jones & Bartlett Learning, Boston.

Dabbura, I "Gradient descent algorithm and its variants". https://towardsdatascience.com: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3 (Erişim Tarihi: 15.-0.2019)

Davis, J., & Goadrich, M. (2006). "The relationship between precision-recall and ROC curves". *In Proceedings of the 23rd international conference on Machine learning*: 233-240.

Deng, L. (2014). "A tutorial survey of architectures, algorithms, and applications for deep learning". *APSIPA Transactions on Signal and Information Processing, 3, e2.*: 1-29.

Deng, L., & Yu, D. (2014). "Deep learning: methods and applications". *Foundations and Trends® in Signal Processing, 7(3–4):* 197-387.

Deng, W. J., & Pei, W. (2009). "Fuzzy neural based importance-performance analysis for determining critical service attributes". . *Expert Systems with Applications, 36(2)*: 3774-3784.

Despois, J. "Memorizing is not learning!—6 tricks to prevent overfitting in machine learning". https://ai-odyssey.com: https://ai-odyssey.com/2018/03/22/memorizing-is-not-learning%E2%80%8A-%E2%80%8A6-tricks-to-prevent-overfitting-in-machine-learning/ (Erişim Tarihi 18.01.2019)

Despois, J. "Stop Feeding Garbage To Your Model!—The 6 biggest mistakes with datasets and how to avoid them". https://hackernoon.com: https://hackernoon.com/stop-feeding-garbage-to-your-model-the-6-biggest-mistakes-with-datasets-and-how-to-avoid-them-3cb7532ad3b7 (Erişim Tarihi: 15.01.2019)

Doshi, N. "Deep Learning Best Practices (1)—Weight initialization". https://medium.com/: https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94 (Erişim Tarihi: 13.01.2019)

Dozat, T. (2016). "Incorporating nesterov momentum into adam". *Workshop track - ICLR 2016*: 1-4.

Duchi, J., Hazan, E., & Singer, Y. (2011). "Adaptive subgradient methods for online learning and stochastic optimization". *Journal of Machine Learning Research, 12(Jul)*: 2121-2159.

Eddins, S. "Deep Learning Network Analyzer". https://blogs.mathworks.com/deep-learning/page/2/ (Erişim Tarihi: 16.01.2019)

El Hatri, C., & Boumhidi, J. (2018)." Fuzzy deep learning based urban traffic incident detection". *Cognitive Systems Research, 50*: 206-213.

Elmas, Ç. (2001). *Yapay zeka uygulamaları:(yapay sinir ağı, bulanık mantık, genetik algoritma).* Seçkin Yayıncılık, Ankara.

Emir, Ş. (2013). *Yapay Sinir Ağları ve Destek Vektör Makineleri Yöntemlerinin Sınıflandırma Performanslarının Karşılaştırılması: Borsa Endeks Yönünün Tahmini Üzerine Bir Uygulama.* İstanbul: İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü.

Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., & Philbrick, K. (2017). "Toolkits and libraries for deep learning". *Journal of digital imaging, 30(4)*: 400-405.

Ersoylu, İ. (2011). *Bulanık VIKOR ve Bulanık AHP Yöntemleri ile Performans Ölçümü.* İstanbul: Yayınlanmamış Yüksek Lisans Tezi, Havacılık ve Uzay Teknolojileri Enstitüsü.

Fawcett, T. (2006). "An introduction to ROC analysis". *Pattern recognition letters, 27(8):* 861-874.

Feng, Z. H., Kittler, J., Awais, M., Huber, P., & Wu, X. J. (2018). "Wing loss for robust facial landmark localisation with convolutional neural networks". *IEEE Conference on Computer Vision and Pattern Recognition*: 1-11.

Freeman, J. A., & Skapura, D. M. (1991). *Neural Networks Algorithms, Applications,and Programming Techniques.* Addison-Wesley Publishing Company, Inc, ABD

Fulcher, J. (2006). *Advances in applied artificial intelligence.* IGI Global, ABD

Fyfe, C. (2000). *Artificial Neural Networks and Information Theory.* Department of Computing and Information System, The university of Paisley, İngiltere.

Ganguly, K. (2017). *Learning Generative Adversarial Networks.* Packt Publishing Ltd, Birmingham.

Gao, Y., & Lu, Q. (2018). "A fuzzy logistic regression model based on the least squares estimation". *Computational and Applied Mathematics, 37(3)*: 3562-3579.

Gentle, E. J. (2004). *Optimization Methods For Applications in Statistics.* Springer, Virginia.

Gil, M. A., & González-Rodríguez, G. (2012). "Fuzzy vs. Likert scale in statistics". *InCombining experimentation and theory*: 407-420.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press, ABD.

Gradient Descent*," "* Gradient Descent", ml-cheatsheet.readthedocs.io: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html (Erişim Tarihi: 02.03.2019)

Graupe, D. (2007). *Principles of artificial neural networks.* World Scientific, Chicago

Grover, P. "Regression loss functions all machine learners should know". heartbeat.fritz.ai: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0 (Erişim Tarihi: 29.12.2018)

Gulli, A., & Pal, S. (2017). *Deep Learning with Keras.* . Packt Publishing Ltd, Birmingham.

Güner, N., & Çomak, E. (2014). "Lise öğrencilerinin matematik dersine yönelik tutumlarının bulanık mantık yöntemi ile incelenmesi". *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 20(5)*: 189-196.

Halepmollası, R. (2016). *Alt Sekans Profil Haritalari Kullanılarak Protein Katlanması Tanıma.* İstanbul: İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 2016, İstanbul

Hamzaçebi, C. (2011). *Yapay sinir ağları.* Ekin Yayınevi, Bursa.

Hatcher, W. G., & Yu, W. (2018). "A survey of deep learning: platforms, applications and emerging research trends." *IEEE Access, 6:* 24411-24432.

Heaton, J. (2015). *Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning.* Heaton Research Inc, Chesterfield, ABD.

Hinton, ,. G., Osindero, S., & Teh, Y. W. (2006). "A fast learning algorithm for deep belief nets." *Neural computation, 18(7)*: 1527-1554.

Hossain, M. D., Muhammad, G., & Amin, S. U. (2018). "Improving consumer satisfaction in smart cities using edge computing and caching: A case study of date fruits classification". *Future Generation Computer Systems, 88*: 333-341.

Ishibuchi, H., & Nii, M. (1998). "Fuzzification of input vectors for improving the generalization ability of neural networks". *In 1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence*: 1153-1158.

Jager, R. (1995). *Fuzzy Logic in Control.* Delft Teknoloji Üniversitesi, Doktora Tezi.

Jahandideh, S., Asefzadeh, S., Jahandideh, M., Asadabadi, E. B., & Jafari, A. (2013). "The comparison of methods for measuring quality of hospital services by using neural networks: A case study in Iran". *International Journal of Healthcare Management, 6(1)*: 45-50.

Jebara, T. (2012). *Machine learning: discriminative and generative.* Columbia University, Springer Science & Business Media, ABD

Jensen, W. A. (2008). "Decision trees for business intelligence and data mining: using SAS® enterprise miner". *Technometrics, 50(3)*: 409-410.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Darrell, T. (2014). "Caffe: Convolutional architecture for fast feature embedding". *In Proceedings of the 22nd ACM international conference on Multimedia. ACM:* 675-678.

Kaneko, Y., & Yada, K. (2016). "A deep learning approach for the prediction of retail store sales". *In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW),* 531-537.

Kappor, R., Walters, S. P., & Al-Aswad, L. A. (2018). "The current state of artificial intelligence in ophthalmology". *Survey of ophthalmology:* 233-240.

Karakuş, B. A. "Derin Sinir Ağları için Aktivasyon Fonksiyonları". http://buyukveri.firat.edu.tr/2018/04/17/derin-sinir-aglari-icin-aktivasyon-fonksiyonlari/ (Erişim Tarihi: 04.04.2019)

Karpathy, A. "CS231n: Convolutional neural networks for visual recognition". http://cs231n.stanford.edu/: http://cs231n.github.io/neural-networks-3/ (Erişim Tarihi: 07.04.2019)

Kennedy, K., Delany, S. J., & Mac Namee, B. (2011). "A Framework for generating data to simulate application scoring". *Credit Scoring and Credit Control XII, Conference Proceedings* (s. 1-21). Edinburg: Credit Research Centre, Business School, University of Edinburgh.

Keras.io. "Keras: The python deep learning library". https://keras.io/ (Erişim Tarihi: 16.04.2019)

Kingma, D. P., & Ba, J. L. (2015). "Adam: Amethod for stochastic optimization". *ICLR 2015*: 1-15.

Koenker, R., & Bassett Jr, G. (1978). "Regression quantiles". *Econometrica: journal of the Econometric Society*: 33-50.

Koenker, R., & Hallock, K. F. (2001). "Quantile regression". *Journal of economic perspectives, 15(4)*: 143-156.

Kröse, B., & van der Smagt, P. (1996). *An introduction to neural networks.* The University of Amsterdam, Amsterdam.

Kurt, W. "Kullback-leibler divergence explained". https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained (Erişim Tarihi: 29.12.2018)

Lemberger, P. (2017). "On generalization and regularization in deep learning". *arXiv preprint arXiv:1704.01312.*: 1-11.

Lewis, N. D. (2016). *Deep Learning Made Easy with R.* Auscov, ABD

Li, D., & Du, Y. (2008). *Artificial intelligence with uncertainty.* CRC press, Boca Raton.

Li, Q. (2013). "A novel Likert scale based on fuzzy sets theory". *Expert Systems with Applications, 40(5)*: 1609-1618.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). "Focal loss for dense object detection". *arXiv preprint arXiv:1708.02002*: 1-10.

Liu, D. C. "A Practical Guide to ReLU". https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7 (Erişim Tarihi: 21.04.2019)

Liu, P., & Li, H. X. (2004). *Fuzzy neural network theory and application.* World Scientific, Singapore.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). "Rectifier nonlinearities improve neural network acoustic models". *In Proc. icml (Vol. 30, No. 1)*: 3-10.

Marzban, C. (2004). "The ROC curve and the area under it as performance measures". *Weather and Forecasting, 19(6)*: 1106-1114.

McNeill, F. M., & Thro, E. (2014). *Fuzzy logic: a practical approach.* Academic Press, New York

Mehrotra, K., Mohan, C. K., & Ranka, S. (1996). *Elements of Artificial Neural Networks.* USA: MIT Press.

Michelucci, U. (2018). *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. Apress.* Apress, Dübendorf, İsviçre.

Mitchell, T. M. (1997). Chapter 3: Decision tree learning. T. M. Mitchell içinde, *Machine Learning (ISBN 0-07-115467-1)* (s. 55-64). McGraw-Hill, Boston.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). "Deep learning for IoT big data and streaming analytics: a survey". *IEEE Communications Surveys & Tutorials*: 2923-2960.

Mousavi, S. S., Schukat, M., & Howley, E. (2016). "Deep reinforcement learning: an overview". *In Proceedings of SAI Intelligent Systems Conference*: 426-440.

Namdari, M., Abadi, A., Taheri, S. M., Rezaei, M., Kalantari, N., & Omidvar, N. (2014). “Effect of folic acid on appetite in children: Ordinal logistic and fuzzy logistic regressions”. *Nutrition, 30(3)*: 274-278.

Namdari, M., Yoon, J. H., Abadi, A., Taheri, S. M., & Choi, S. H. (2015). “Fuzzy logistic regression with least absolute deviations estimators”. *Soft Computing, 19(4)*: 909-917.

Nesterov, Y. (1983). “A method for unconstrained convex minimization problem with the rate of convergence O”. *In Doklady AN USSR (Vol. 269)*: 543-547.

Ng, A. “Generative learning algorithms”. http://cs229.stanford.edu/: http://cs229.stanford.edu/notes/cs229-notes2.pdf (Teslim Tarihi: 21.11.2018)

Ng, A. “CS229 Lecture notes”. http://cs229.stanford.edu: http://cs229.stanford.edu/notes/cs229-notes1.pdf (Teslim Tarihi: 30.04.2019)

NRC, N. R. (1997). “Computer science and artificial intelligence”. *Panel on Computer Science and Artifical Intelligence*: 1-18.

Orr, G. (2006, 9 1). “Momentum and learning rate adaptation”. https://www.willamette.edu/~gorr/classes/cs449/momrate.html (Erişim Tarihi: 28.03.2019)

Owen, A. B. (2007). “A robust hybrid of lasso and ridge regression”. *Contemporary Mathematics, 443(7)*: 59-72.

Öğücü, M. O. (2006). *Yapay Sinir Ağlari ile Sistem Tanıma.* İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

Parmar, R. “Common loss functions in machine learning”. https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23 (Erişim Tarihi: 27.03.2019)

Patterson, J., & Gibson, A. (2017). *Deep Learning: A Practitioner's Approach.* O'Reilly Media, Inc, ABD.

Pedrycz, W., Ekel, P., & Parreiras, R. (2011). *Fuzzy Multicriteria Decision-Making: Models, Methods and Applications.* John Wiley & Sons, West Sussex.

Peng, T., & Hanke, F. (2016). "Towards a synthetic data generator for matching decision trees". *ICEIS (1)*: 135-141.

PSF, P. S. (2019, 04 14). "What is python? Executive summary". https://www.python.org/doc/essays/blurb/ (Erişim Tarihi: 22.02.2019)

Quesada, A. "5 algorithms to train a neural network". https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network (Erişim Tarihi: 23.04.2019)

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). "Swish: a self-gated activation function". *arXiv preprint arXiv:1710.05941*: 1-12.

Ramamurthy, V., & Duffy, N. (2017). L-SR1: "A second order optimization method for deep learning". *ICLR 2017*: 1-15.

Raschka, S., & Mirjalili, V. (2017). *Machine Learning and Deep Learning with Python, scikit-learn and TensorFlow.* Packt Publishing, Birmingham, İngiltere.

Rawat, W., & Wang, Z. (2017). "Deep convolutional neural networks for image classification: A comprehensive review". *Neural computation, 29(9:* 2352-2449.

Reddi, S. J., Kale, S., & Kumar, S. (2018). "On the convergence of adam and beyond". *ICLR 2018*: 1-23.

Ripert, M. "How Instacart delivers on time (using quantile regression)". https://tech.instacart.com: https://tech.instacart.com/how-instacart-delivers-on-time-using-quantile-regression-2383e2e03edb (Erişim Tarihi: 19.03.2019)

robotomy.eu. "Literature review of deep machine learning for feature extraction" . http://www.robotomy.eu/literature-review-of-deep-machine-learning-for-feature-extraction/ (Erişim Tarihi: 12.08.2018)

Ross. (2010). *Fuzzy logic with engineering applications.* John Wiley & Sons, USA.

Ruder, S. (2016). "An overview of gradient descent optimization algorithms". *arXiv preprint arXiv:1609.04747:* 1-14.

Rusli, N. M., Ibrahim, Z., & Janor, R. M. (2008). "Predicting students' academic achievement: Comparison between logistic regression, artificial neural network, and Neuro-fuzzy". *In 2008 International Symposium on Information Technology (Vol. 1, pp. 1-6). IEEE:* 1-6.

Russell, R. (2018). *Machine Learning Step-by-Step Guide To Implement Machine Learning Algorithms with Python.* CreateSpace Independent Publishing Platform, ABD.

Salmani, F., Taheri, S. M., Yoon, J. H., Abadi, A., Majd, H. A., & Abbaszadeh, A. (2017). "Logistic regression for fuzzy covariates: Modeling, inference, and applications". *International Journal of Fuzzy Systems, 19(5)*: 1635-1644.

Sharma, A. "Understanding activation functions in deep learning". https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/ (Erişim Tarihi: 12.01.2019)

Shi, S., Wang, Q., Xu, P., & Chu, X. (2016). "Benchmarking state-of-the-art deep learning software tools". *In 2016 7th International Conference on Cloud Computing and Big Data (CCBD). IEEE:* 99-104.

Sohn, S. Y., Kim, D. H., & Yoon, J. H. (2016). "Technology credit scoring model with fuzzy logistic regression". *Applied Soft Computing, 43*: 150-158.

Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation". *In Australasian joint conference on artificial intelligence*: 1015-1021.

Sreekumar, S., & Mahapatra, S. (2015). "Service quality of Indian banks: A fuzzy inference system approach". *Asian Acad. Manag. J, 20(2)*: 59-80.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). "Dropout: a simple way to prevent neural networks from overfitting". *The Journal of Machine Learning Research, 15(1)*: 1929-1958.

Şen, Z. (2009). *Bulanık Mantık İlkeleri ve Modelleme.* Su Vakfı Yayınları, İstanbul.

Tabrizi, T. S., Khoie, M. R., Sahebkar, E., Rahimi, S., & Marhamati, N. (2016). "Towards a patient satisfaction based hospital recommendation system". *In 2016 International Joint Conference on Neural Networks*: 131-138.

Tayyar, N. (2010). "Müşteri memnuniyeti tahmininde yapay sinir ağlari, lojistik Regresyon Ve Ayirma Analizinin Performanslarinin Karşilaştirilmasi". *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 15(1)*: 339-355.

Teboulle, M. "First order optimization methods". https://www.math.u-psud.fr/IMG/pdf/abstract-courpgmo-poytechnique_cle0fcb4d.pdf (Erişim Tarihi: 21.12.2018)

Thomas, P. (2005). *Artificial Intelligence.* Thomson Gale, ABD

Toussaint, M. "Introduction to optimization. second order optimization methods". https://ipvs.informatik.uni-stuttgart.de/mlr/marc/teaching/13-Optimization/04-secondOrderOpt.pdf (Erişim Tarihi: 04.02.2019)

Tsaur, S. H., Chiu, Y. C., & Huang, C. H. (2002). "Determinants of guest loyalty to international tourist hotels—a neural network approach". *Tourism Management, 23(4)*: 397-405.

Unruh, A. "What is the TensorFlow machine intelligence platform?" https://opensource.com: https://opensource.com/article/17/11/intro-tensorflow (Erişim Tarihi: 18.09.2019)

Usage of optimizers ,"Usage of optimizers". https://keras.io: https://keras.io/optimizers/ (Erişim Tarihi: 12.09.2108)

Vansteenkiste, E. (2016). *New FPGA Design Tools and Architectures. Doktora Tezi.* Gent, Belçika: Ghent Üniversitesi.

Wang, H., Xu, Z., & Pedrycz, W. (2017). "An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities". *Knowledge-Based Systems, 118*: 15-30.

Wang, Y. M. (2009). "Centroid defuzzification and the maximizing set and minimizing set ranking based on alpha level sets". *Computers & Industrial Engineering, 57(1)*: 228-236.

Warwick, K. (2013). *Artificial intelligence: the basics.* Routledge, London.

Weinberger, K. "*CS4780/CS5780: Machine Learning for Intelligent Systems"*. http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote10.html (Erişim Tarihi: 20.12.2018)

Yadav, S. "Weight initialization techniques in neural networks". https://towardsdatascience.com: https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78 (Erişim Tarihi: 19.03.2019)

Yu, D., Eversole, A., Seltzer, M., Yao, K., Huang, A., Guenter, B., Slaney, M. (2014). "An introduction to computational networks and the computational network toolkit". *Technical report, Microsoft Research:* 1-179.

Zeiler, M. D. (2012). "ADADELTA: an adaptive learning rate method". *arXiv preprint arXiv:1212.5701:* 1-6.

Zhang, J., Tao, C., & Wang, P. (2016). "A review of soft computing based on deep learning". *In 2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration:* 136-144.

Zhong, G., Ling, X., & Wang, L. N. (2019). "From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9(1): e1255.*

EK1 –Lojistik Regresyon Modellerinin Sınıflandırma Sonuçları

Hesaplama Süreleri

Örnek Sayısı	LogLikert	LogÜçgenBulanık	LogYamukBulanık	LogÜçgenYeni	LogYamukYeni
100	0,002	0,006	0,007	0,003	0,004
200	0,002	0,006	0,007	0,003	0,004
300	0,002	0,006	0,007	0,003	0,004
400	0,002	0,006	0,007	0,003	0,004
500	0,002	0,006	0,007	0,003	0,004
600	0,002	0,006	0,007	0,003	0,004
700	0,002	0,006	0,007	0,003	0,004
800	0,002	0,006	0,007	0,003	0,004
900	0,002	0,006	0,007	0,003	0,004
1000	0,002	0,006	0,007	0,003	0,004

Duyarlılık Değerleri

Örnek Sayısı	LogLikert eğitim	LogLikert accuracy test	LogÜçgenBulanık eğitim	LogÜçgenBulanık test	LogYamukBulanık eğitim	LogYamukBulanık test	LogÜçgenYeni eğitim	LogÜçgenYeni Test	LogYamukYeni eğitim	LogYamukYeni test
100	0,761	0,545	0,763	0,552	0,763	0,552	0,860	0,644	0,870	0,659
200	0,709	0,617	0,716	0,620	0,716	0,620	0,890	0,805	0,903	0,821
300	0,768	0,652	0,775	0,656	0,775	0,656	0,919	0,855	0,931	0,872
400	0,819	0,758	0,824	0,762	0,823	0,762	0,941	0,904	0,945	0,912
500	0,878	0,836	0,880	0,839	0,880	0,838	0,964	0,932	0,968	0,939
600	0,886	0,850	0,890	0,856	0,889	0,855	0,963	0,934	0,966	0,942
700	0,908	0,881	0,910	0,886	0,909	0,886	0,979	0,963	0,982	0,967
800	0,919	0,885	0,921	0,886	0,921	0,886	0,978	0,965	0,982	0,967
900	0,931	0,904	0,932	0,904	0,932	0,904	0,978	0,968	0,980	0,973
1000	0,927	0,906	0,931	0,909	0,930	0,909	0,987	0,970	0,988	0,974

Doğruluk Oranları

Örnek Sayısı	LogLikert eğitim	LogLikert accuracy test	LogÜçgenBulanık eğitim	LogÜçgenBulanık test	LogYamukBulanık eğitim	LogYamukBulanık test	LogÜçgenYeni eğitim	LogÜçgenYeni Test	LogYamukYeni eğitim	LogYamukYeni test
100	0,750	0,528	0,751	0,531	0,751	0,531	0,857	0,650	0,867	0,668
200	0,767	0,673	0,771	0,676	0,771	0,676	0,914	0,838	0,925	0,849
300	0,812	0,704	0,817	0,708	0,817	0,707	0,939	0,881	0,946	0,894
400	0,851	0,788	0,855	0,792	0,854	0,792	0,952	0,915	0,958	0,922
500	0,886	0,838	0,887	0,842	0,887	0,841	0,965	0,934	0,969	0,940
600	0,905	0,869	0,907	0,872	0,907	0,871	0,971	0,949	0,974	0,955
700	0,922	0,893	0,923	0,896	0,923	0,896	0,978	0,966	0,982	0,969
800	0,937	0,915	0,939	0,916	0,939	0,916	0,983	0,974	0,986	0,976
900	0,943	0,920	0,944	0,921	0,944	0,921	0,984	0,977	0,986	0,980
1000	0,945	0,920	0,947	0,922	0,947	0,922	0,989	0,976	0,990	0,979

F-Ölçütü Değerleri

Örnek Sayısı	LogLikert eğitim	LogLikert accuracy test	LogÜçgenBulanık eğitim	LogÜçgenBulanık test	LogYamukBulanık eğitim	LogYamukBulanık test	LogÜçgenYeni eğitim	LogÜçgenYeni Test	LogYamukYeni eğitim	LogYamukYeni test
100	0,734	0,479	0,735	0,484	0,735	0,484	0,846	0,599	0,858	0,618
200	0,709	0,611	0,716	0,614	0,716	0,614	0,895	0,808	0,907	0,822
300	0,772	0,648	0,778	0,652	0,778	0,652	0,926	0,860	0,935	0,877
400	0,827	0,765	0,832	0,769	0,831	0,769	0,945	0,907	0,951	0,914
500	0,874	0,819	0,876	0,823	0,876	0,823	0,962	0,926	0,966	0,932
600	0,890	0,852	0,893	0,856	0,893	0,855	0,967	0,943	0,970	0,949
700	0,911	0,878	0,913	0,882	0,912	0,882	0,976	0,961	0,979	0,965
800	0,928	0,904	0,930	0,905	0,930	0,905	0,981	0,971	0,984	0,973
900	0,935	0,909	0,936	0,909	0,936	0,909	0,982	0,973	0,984	0,977
1000	0,937	0,911	0,940	0,914	0,940	0,914	0,988	0,973	0,989	0,977

Kesinlik Değerleri

Örnek Sayısı	LogLikert eğitim	LogLikert accuracy test	LogÜçgenBulanık eğitim	LogÜçgenBulanık test	LogYamukBulanık eğitim	LogYamukBulanık test	LogÜçgenYeni eğitim	LogÜçgenYeni Test	LogYamukYeni eğitim	LogYamukYeni test
100	0,714	0,474	0,716	0,477	0,716	0,477	0,837	0,610	0,850	0,631
200	0,752	0,639	0,755	0,642	0,755	0,642	0,909	0,831	0,920	0,841
300	0,792	0,687	0,797	0,690	0,797	0,690	0,935	0,882	0,942	0,894
400	0,843	0,786	0,847	0,790	0,846	0,790	0,950	0,914	0,958	0,921
500	0,874	0,811	0,875	0,815	0,875	0,815	0,961	0,922	0,964	0,927
600	0,898	0,865	0,900	0,866	0,900	0,866	0,971	0,956	0,976	0,959
700	0,917	0,882	0,918	0,883	0,917	0,883	0,973	0,961	0,977	0,965
800	0,939	0,925	0,939	0,926	0,939	0,926	0,984	0,977	0,987	0,979
900	0,940	0,916	0,941	0,917	0,941	0,917	0,986	0,979	0,987	0,982
1000	0,949	0,920	0,951	0,922	0,951	0,922	0,988	0,976	0,990	0,979

ROC-Ölçütü Değerleri

Örnek Sayısı	LogLikert eğitim	LogLikert accuracy test	LogÜçgenBulanık eğitim	LogÜçgenBulanık test	LogYamukBulanık eğitim	LogYamukBulanık test	LogÜçgenYeni eğitim	LogÜçgenYeni Test	LogYamukYeni eğitim	LogYamukYeni test
100	0,737	0,539	0,739	0,544	0,739	0,544	0,849	0,655	0,859	0,671
200	0,751	0,663	0,755	0,666	0,755	0,666	0,907	0,834	0,918	0,846
300	0,799	0,699	0,804	0,703	0,803	0,702	0,933	0,878	0,942	0,891
400	0,844	0,785	0,848	0,789	0,848	0,789	0,950	0,914	0,956	0,921
500	0,884	0,838	0,885	0,841	0,885	0,841	0,965	0,935	0,968	0,940
600	0,900	0,869	0,903	0,872	0,903	0,871	0,970	0,949	0,973	0,954
700	0,919	0,892	0,921	0,895	0,920	0,895	0,978	0,965	0,981	0,969
800	0,935	0,913	0,937	0,914	0,937	0,914	0,983	0,973	0,986	0,975
900	0,941	0,919	0,941	0,920	0,941	0,920	0,983	0,976	0,985	0,979
1000	0,941	0,919	0,944	0,921	0,943	0,921	0,988	0,975	0,990	0,979

EK2– Likert Tipi Veri ile Derin Öğrenme Modellerinin Sınıflandırma Sonuçları

Hesaplama Süreleri

DeepLikert	DerinÜggenBulanık	DerinYamukBulanık	DerinÜggenYeni	DerinYamukYeni
23	68	94	19	20
26	90	101	21	23
44	124	152	27	25
46	133	158	28	28
55	170	190	30	29
72	182	203	31	30
81	193	223	32	31
83	198	225	33	31
84	199	232	34	31
85	202	234	34	32

Duyarlılık Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,394	0,692	0,492	0,425	0,477	0,396	0,319	0,307	0,265	0,252
0,368	0,328	0,464	0,381	0,408	0,353	0,554	0,509	0,398	0,370
0,520	0,477	0,703	0,628	0,710	0,638	0,798	0,734	0,633	0,593
0,768	0,704	0,824	0,775	0,881	0,813	0,897	0,864	0,792	0,766
0,991	0,952	0,999	0,963	0,998	0,962	1,000	0,977	0,997	0,975
0,985	0,956	0,993	0,961	0,995	0,959	1,000	0,973	1,000	0,970
0,995	0,973	1,000	0,987	1,000	0,987	1,000	0,986	1,000	0,989
0,992	0,974	1,000	0,984	1,000	0,991	1,000	0,991	1,000	0,998
0,994	0,988	0,999	0,992	1,000	0,996	1,000	0,990	1,000	0,995
0,996	0,996	1,000	0,995	1,000	0,996	1,000	0,997	1,000	0,997

Doğruluk Oranları

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,609	0,527	0,669	0,542	0,667	0,542	0,582	0,548	0,554	0,542
0,641	0,589	0,722	0,628	0,697	0,620	0,697	0,645	0,640	0,618
0,780	0,731	0,843	0,775	0,832	0,759	0,879	0,814	0,798	0,760
0,865	0,821	0,921	0,878	0,934	0,886	0,957	0,923	0,885	0,874
0,991	0,964	0,999	0,974	0,998	0,975	0,999	0,978	0,998	0,980
0,989	0,969	0,997	0,977	0,998	0,975	0,998	0,980	1,000	0,983
0,995	0,977	1,000	0,990	1,000	0,993	1,000	0,990	1,000	0,994
0,994	0,983	1,000	0,993	1,000	0,996	0,999	0,991	1,000	0,998
0,996	0,992	1,000	0,995	1,000	0,998	1,000	0,995	1,000	0,998
0,998	0,997	1,000	0,998	1,000	0,997	1,000	0,998	1,000	0,998

F-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,317	0,261	0,428	0,310	0,437	0,300	0,251	0,215	0,192	0,167
0,325	0,281	0,470	0,370	0,405	0,327	0,474	0,430	0,328	0,305
0,512	0,467	0,688	0,621	0,691	0,618	0,766	0,714	0,604	0,565
0,761	0,700	0,825	0,783	0,886	0,831	0,896	0,868	0,788	0,768
0,991	0,958	0,999	0,970	0,998	0,971	0,998	0,975	0,997	0,978
0,988	0,966	0,996	0,974	0,997	0,972	0,998	0,978	1,000	0,981
0,994	0,974	1,000	0,989	1,000	0,992	1,000	0,989	1,000	0,993
0,993	0,981	1,000	0,992	1,000	0,995	0,999	0,990	1,000	0,998
0,995	0,991	1,000	0,994	1,000	0,998	1,000	0,994	1,000	0,997
0,997	0,997	1,000	0,997	1,000	0,997	1,000	0,998	1,000	0,998

Kesinlik Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,282	0,211	0,425	0,262	0,462	0,297	0,217	0,178	0,159	0,134
0,304	0,266	0,496	0,393	0,437	0,346	0,434	0,398	0,297	0,277
0,510	0,474	0,684	0,636	0,700	0,639	0,751	0,712	0,587	0,554
0,761	0,707	0,826	0,793	0,892	0,854	0,894	0,873	0,784	0,772
0,990	0,966	0,999	0,977	0,997	0,980	0,997	0,974	0,998	0,982
0,990	0,976	0,999	0,990	1,000	0,986	0,996	0,983	0,999	0,992
0,993	0,975	1,000	0,991	1,000	0,997	1,000	0,993	1,000	0,997
0,994	0,988	1,000	1,000	1,000	1,000	0,999	0,989	1,000	0,998
0,997	0,994	1,000	0,997	1,000	1,000	1,000	0,998	1,000	1,000
0,999	0,999	1,000	1,000	1,000	0,998	1,000	0,999	0,999	0,999

ROC-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,546	0,524	0,612	0,542	0,612	0,537	0,551	0,538	0,531	0,518
0,598	0,563	0,681	0,606	0,650	0,586	0,664	0,628	0,612	0,588
0,740	0,705	0,810	0,754	0,798	0,743	0,862	0,815	0,778	0,746
0,849	0,818	0,909	0,870	0,924	0,875	0,947	0,920	0,879	0,861
0,991	0,962	0,999	0,973	0,998	0,974	0,999	0,978	0,997	0,980
0,989	0,968	0,996	0,976	0,997	0,974	0,999	0,980	1,000	0,982
0,995	0,976	1,000	0,990	1,000	0,992	1,000	0,989	1,000	0,993
0,994	0,982	1,000	0,992	1,000	0,995	0,999	0,991	1,000	0,998
0,996	0,992	1,000	0,995	1,000	0,998	1,000	0,994	1,000	0,997
0,997	0,997	1,000	0,997	1,000	0,997	1,000	0,998	1,000	0,998

EK3–Karmaşık Veri ile Derin Öğrenme Modellerinin Sınıflandırma Sonuçları

Hesaplama Süreleri

DeepLikert	DerinÜggenBulanık	DerinYamukBulanık	DerinÜggenYeni	DerinYamukYeni
10	50	61	7	8
18	63	91	20	17
49	113	152	28	17
55	130	177	26	23
59	147	182	23	20
60	151	201	20	19
61	156	225	24	22
59	177	238	25	25
72	183	242	24	24
77	197	249	30	35

Duyarlılık Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,656	0,641	0,715	0,689	0,689	0,636	0,699	0,656	0,550	0,532
0,760	0,734	0,902	0,865	0,921	0,878	0,873	0,820	0,747	0,728
0,884	0,856	0,946	0,905	0,907	0,867	0,986	0,945	0,890	0,856
0,930	0,899	0,966	0,928	0,965	0,931	0,978	0,945	0,946	0,905
0,946	0,916	0,983	0,951	0,983	0,949	0,985	0,954	0,986	0,946
0,985	0,976	0,983	0,961	0,987	0,965	0,989	0,963	0,988	0,967
0,941	0,918	0,979	0,960	0,981	0,955	0,986	0,953	0,979	0,947
0,984	0,965	0,973	0,971	0,979	0,973	0,980	0,964	0,973	0,960
0,941	0,932	0,979	0,965	0,977	0,965	0,982	0,969	0,983	0,965
0,974	0,968	0,983	0,969	0,977	0,969	0,983	0,967	0,986	0,963

Doğruluk Oranları

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,618	0,528	0,693	0,598	0,693	0,574	0,633	0,558	0,601	0,554
0,681	0,622	0,759	0,685	0,757	0,693	0,753	0,708	0,652	0,633
0,860	0,817	0,927	0,874	0,921	0,871	0,956	0,909	0,858	0,813
0,918	0,886	0,960	0,926	0,957	0,920	0,972	0,943	0,946	0,909
0,942	0,919	0,981	0,951	0,980	0,948	0,979	0,947	0,966	0,932
0,949	0,939	0,980	0,961	0,981	0,965	0,978	0,957	0,979	0,962
0,958	0,938	0,979	0,965	0,982	0,965	0,982	0,962	0,982	0,960
0,961	0,936	0,974	0,969	0,977	0,963	0,982	0,963	0,980	0,961
0,962	0,947	0,981	0,967	0,980	0,966	0,981	0,963	0,981	0,962
0,954	0,953	0,980	0,970	0,976	0,970	0,981	0,969	0,983	0,965

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,446	0,376	0,561	0,489	0,542	0,436	0,476	0,411	0,358	0,330
0,565	0,487	0,740	0,651	0,732	0,653	0,697	0,645	0,545	0,513
0,805	0,770	0,904	0,856	0,883	0,838	0,957	0,914	0,808	0,769
0,894	0,864	0,958	0,929	0,955	0,917	0,969	0,945	0,930	0,897
0,923	0,906	0,979	0,955	0,977	0,951	0,973	0,946	0,979	0,953
0,942	0,935	0,978	0,963	0,975	0,968	0,969	0,952	0,971	0,959
0,939	0,931	0,978	0,973	0,982	0,977	0,978	0,973	0,984	0,976
0,958	0,924	0,978	0,968	0,978	0,955	0,984	0,964	0,988	0,961
0,947	0,935	0,985	0,970	0,983	0,968	0,980	0,959	0,981	0,962
0,952	0,954	0,980	0,974	0,977	0,974	0,980	0,972	0,982	0,968

F-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,524	0,462	0,613	0,541	0,597	0,503	0,561	0,495	0,429	0,397
0,639	0,577	0,803	0,728	0,807	0,737	0,760	0,705	0,612	0,583
0,833	0,799	0,921	0,874	0,892	0,849	0,967	0,923	0,836	0,798
0,909	0,878	0,961	0,927	0,960	0,923	0,973	0,944	0,935	0,897
0,932	0,908	0,981	0,952	0,980	0,949	0,979	0,949	0,982	0,949
0,960	0,950	0,981	0,962	0,981	0,966	0,979	0,958	0,979	0,963
0,940	0,924	0,978	0,966	0,982	0,966	0,982	0,962	0,982	0,961
0,968	0,942	0,975	0,969	0,978	0,964	0,982	0,963	0,981	0,961
0,944	0,933	0,982	0,967	0,980	0,967	0,981	0,964	0,982	0,963
0,961	0,959	0,981	0,971	0,977	0,971	0,981	0,970	0,984	0,966

ROC-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,562	0,529	0,642	0,592	0,640	0,572	0,592	0,550	0,545	0,534
0,665	0,622	0,741	0,684	0,737	0,691	0,754	0,703	0,651	0,627
0,847	0,818	0,920	0,876	0,916	0,874	0,953	0,910	0,855	0,817
0,916	0,887	0,959	0,926	0,957	0,920	0,972	0,944	0,945	0,909
0,941	0,917	0,981	0,951	0,980	0,948	0,979	0,946	0,966	0,932
0,943	0,936	0,980	0,960	0,981	0,965	0,978	0,957	0,979	0,962
0,957	0,940	0,978	0,965	0,982	0,965	0,982	0,961	0,982	0,960
0,960	0,934	0,974	0,969	0,977	0,963	0,982	0,963	0,981	0,961
0,961	0,949	0,982	0,967	0,980	0,966	0,981	0,963	0,982	0,962
0,954	0,950	0,980	0,970	0,976	0,970	0,981	0,969	0,983	0,964

EK4–Dengesiz Veri ile Derin Öğrenme Modellerinin Sınıflandırma Sonuçları

Hesaplama Süreleri

DeepLikert	DerinÜçgenBulanık	DerinYamukBulanık	DerinÜçgenYeni	DerinYamukYeni
13	86	99	17	17
46	89	124	24	22
44	111	157	28	26
45	128	165	31	28
60	129	194	31	30
49	147	198	36	34
53	160	212	37	35
59	165	217	34	31
52	190	241	37	36
59	188	245	38	36

Duyarlılık Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,000	0,000	0,000	0,000	0,000	0,000	0,167	0,033	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,258	0,161	0,349	0,145
0,000	0,000	0,000	0,000	0,000	0,000	0,768	0,532	0,787	0,582
0,000	0,000	0,190	0,126	0,115	0,106	0,824	0,636	0,806	0,663
0,000	0,000	0,188	0,121	0,174	0,113	0,854	0,645	0,834	0,630
0,076	0,069	0,273	0,206	0,267	0,194	0,972	0,762	0,981	0,753
0,237	0,229	0,395	0,407	0,336	0,332	0,988	0,823	0,987	0,758
0,470	0,401	0,538	0,604	0,556	0,486	0,937	0,800	0,945	0,803
0,537	0,467	0,620	0,558	0,666	0,581	0,995	0,870	0,988	0,856
0,554	0,523	0,665	0,581	0,715	0,654	0,980	0,879	0,984	0,862

Doğruluk Oranları

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,871	0,867	0,871	0,867	0,871	0,867	0,888	0,867	0,871	0,861
0,852	0,822	0,852	0,822	0,852	0,822	0,878	0,842	0,893	0,836
0,845	0,839	0,845	0,839	0,845	0,839	0,952	0,894	0,949	0,893
0,821	0,821	0,860	0,840	0,845	0,838	0,966	0,894	0,958	0,899
0,838	0,856	0,869	0,873	0,867	0,872	0,970	0,926	0,965	0,920
0,845	0,835	0,869	0,850	0,878	0,855	0,983	0,933	0,981	0,925
0,875	0,885	0,902	0,912	0,895	0,899	0,989	0,949	0,987	0,930
0,918	0,901	0,927	0,908	0,933	0,919	0,978	0,941	0,979	0,939
0,927	0,905	0,949	0,924	0,956	0,930	0,987	0,954	0,986	0,945
0,914	0,911	0,941	0,928	0,954	0,945	0,984	0,956	0,984	0,953

F-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,000	0,000	0,000	0,000	0,000	0,000	0,154	0,048	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,259	0,187	0,359	0,168
0,000	0,000	0,000	0,000	0,000	0,000	0,755	0,559	0,747	0,572
0,000	0,000	0,242	0,165	0,136	0,123	0,808	0,628	0,787	0,643
0,000	0,000	0,238	0,177	0,225	0,168	0,895	0,717	0,880	0,691
0,085	0,081	0,332	0,263	0,346	0,270	0,947	0,790	0,941	0,769
0,241	0,229	0,473	0,486	0,394	0,390	0,967	0,830	0,959	0,762
0,477	0,411	0,600	0,681	0,615	0,550	0,933	0,800	0,936	0,796
0,594	0,543	0,654	0,613	0,702	0,651	0,958	0,858	0,954	0,834
0,588	0,561	0,688	0,624	0,778	0,730	0,954	0,859	0,952	0,849

Kesinlik Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,000	0,000	0,000	0,000	0,000	0,000	0,143	0,083	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,266	0,224	0,403	0,199
0,000	0,000	0,000	0,000	0,000	0,000	0,747	0,626	0,716	0,578
0,000	0,000	0,333	0,311	0,167	0,146	0,795	0,635	0,773	0,635
0,000	0,000	0,333	0,333	0,333	0,333	0,958	0,843	0,942	0,808
0,097	0,097	0,449	0,438	0,529	0,513	0,929	0,842	0,908	0,805
0,246	0,228	0,617	0,616	0,500	0,496	0,950	0,845	0,935	0,776
0,486	0,425	0,729	0,904	0,744	0,700	0,934	0,811	0,933	0,798
0,688	0,684	0,736	0,701	0,750	0,747	0,925	0,851	0,923	0,825
0,663	0,645	0,727	0,702	0,869	0,861	0,930	0,852	0,923	0,845

ROC-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜggenBulanık eğitim	DerinÜggenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜggenYeni eğitim	DerinÜggenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,500	0,500	0,500	0,500	0,500	0,500	0,581	0,513	0,500	0,497
0,500	0,500	0,500	0,500	0,500	0,500	0,624	0,570	0,669	0,561
0,500	0,500	0,500	0,500	0,500	0,500	0,875	0,746	0,881	0,764
0,500	0,500	0,595	0,561	0,558	0,551	0,907	0,795	0,896	0,810
0,500	0,500	0,594	0,560	0,587	0,556	0,923	0,809	0,912	0,799
0,535	0,531	0,628	0,593	0,630	0,593	0,979	0,864	0,981	0,857
0,618	0,612	0,697	0,703	0,668	0,666	0,989	0,897	0,987	0,859
0,733	0,693	0,767	0,720	0,777	0,738	0,962	0,883	0,966	0,883
0,765	0,730	0,809	0,776	0,833	0,790	0,990	0,920	0,987	0,909
0,771	0,754	0,830	0,787	0,857	0,826	0,983	0,925	0,984	0,916

EK5–Gürültü İçeren Veri ile Derin Öğrenme Modellerinin Sınıflandırma Sonuçları

Hesaplama Süreleri

DeepLikert	DerinÜçgenBulanık	DerinYamukBulanık	DerinÜçgenYeni	DerinYamukYeni
55	199	246	32	30
70	202	259	42	44
64	215	242	37	40
74	226	247	39	34
74	233	250	47	49
77	252	267	51	52
73	247	263	48	54
74	245	268	48	52
73	255	275	55	57
80	260	286	63	65

Duyarlılık Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜçgenBulanık eğitim	DerinÜçgenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜçgenYeni eğitim	DerinÜçgenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,195	0,227	0,195	0,227	0,195	0,227	0,546	0,409	0,546	0,409
0,440	0,410	0,462	0,430	0,462	0,430	0,599	0,560	0,665	0,580
0,653	0,624	0,653	0,624	0,653	0,624	0,741	0,697	0,709	0,636
0,646	0,661	0,646	0,661	0,632	0,652	0,713	0,731	0,716	0,745
0,651	0,653	0,651	0,653	0,651	0,653	0,784	0,740	0,782	0,744
0,646	0,681	0,653	0,684	0,653	0,684	0,732	0,734	0,750	0,721
0,634	0,678	0,604	0,644	0,604	0,644	0,730	0,740	0,727	0,729
0,664	0,653	0,664	0,653	0,664	0,653	0,751	0,740	0,758	0,740
0,602	0,622	0,609	0,628	0,609	0,628	0,690	0,718	0,735	0,766
0,566	0,606	0,566	0,606	0,566	0,606	0,736	0,756	0,745	0,764

Doğruluk Oranları

DerinLikert eğitim	DerinLikert test	DerinÜçgenBulanık eğitim	DerinÜçgenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜçgenYeni eğitim	DerinÜçgenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,871	0,823	0,871	0,823	0,871	0,823	0,925	0,860	0,925	0,860
0,891	0,902	0,894	0,905	0,894	0,905	0,912	0,922	0,930	0,925
0,932	0,924	0,932	0,924	0,932	0,924	0,936	0,922	0,938	0,924
0,929	0,933	0,929	0,933	0,926	0,931	0,936	0,936	0,935	0,940
0,927	0,925	0,927	0,925	0,927	0,925	0,936	0,922	0,936	0,922
0,924	0,936	0,926	0,937	0,926	0,937	0,934	0,937	0,943	0,936
0,925	0,932	0,920	0,925	0,920	0,925	0,936	0,935	0,937	0,934
0,928	0,930	0,928	0,930	0,928	0,930	0,937	0,935	0,940	0,938
0,919	0,919	0,920	0,920	0,920	0,920	0,931	0,933	0,930	0,930
0,913	0,917	0,913	0,917	0,913	0,917	0,940	0,941	0,940	0,939

F-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜçgenBulanık eğitim	DerinÜçgenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜçgenYeni eğitim	DerinÜçgenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,323	0,361	0,323	0,361	0,323	0,361	0,694	0,563	0,694	0,563
0,599	0,582	0,618	0,601	0,618	0,601	0,717	0,704	0,778	0,721
0,766	0,752	0,766	0,752	0,766	0,752	0,798	0,767	0,796	0,755
0,768	0,787	0,768	0,787	0,758	0,781	0,803	0,812	0,801	0,824
0,763	0,764	0,763	0,764	0,763	0,764	0,816	0,778	0,816	0,779
0,763	0,793	0,769	0,795	0,769	0,795	0,808	0,808	0,833	0,801
0,759	0,786	0,738	0,761	0,738	0,761	0,812	0,807	0,812	0,804
0,777	0,778	0,777	0,778	0,777	0,778	0,820	0,812	0,827	0,817
0,735	0,741	0,740	0,745	0,740	0,745	0,789	0,799	0,796	0,803
0,712	0,742	0,712	0,742	0,712	0,742	0,823	0,834	0,825	0,833

Kesinlik Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜçgenBulanık eğitim	DerinÜçgenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜçgenYeni eğitim	DerinÜçgenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,938	0,882	0,938	0,882	0,938	0,882	0,955	0,900	0,955	0,900
0,941	1,000	0,933	1,000	0,933	1,000	0,893	0,949	0,938	0,951
0,927	0,945	0,927	0,945	0,927	0,945	0,865	0,852	0,908	0,929
0,947	0,974	0,947	0,974	0,946	0,974	0,918	0,912	0,908	0,924
0,920	0,919	0,920	0,919	0,920	0,919	0,851	0,820	0,853	0,818
0,933	0,948	0,936	0,949	0,936	0,949	0,902	0,898	0,937	0,900
0,947	0,936	0,949	0,929	0,949	0,929	0,914	0,889	0,919	0,896
0,938	0,961	0,938	0,961	0,938	0,961	0,902	0,901	0,910	0,913
0,945	0,915	0,942	0,916	0,942	0,916	0,922	0,900	0,867	0,844
0,958	0,956	0,958	0,956	0,958	0,956	0,934	0,931	0,924	0,915

ROC-Ölçütü Değerleri

DerinLikert eğitim	DerinLikert test	DerinÜçgenBulanık eğitim	DerinÜçgenBulanık test	DerinYamukBulanık eğitim	DerinYamukBulanık test	DerinÜçgenYeni eğitim	DerinÜçgenYeni test	DerinYamukYeni eğitim	DerinYamukYeni test
0,596	0,609	0,596	0,609	0,596	0,609	0,770	0,698	0,770	0,698
0,717	0,705	0,727	0,715	0,727	0,715	0,791	0,777	0,827	0,787
0,821	0,808	0,821	0,808	0,821	0,808	0,859	0,835	0,847	0,813
0,819	0,828	0,819	0,828	0,812	0,824	0,849	0,857	0,850	0,865
0,819	0,820	0,819	0,820	0,819	0,820	0,877	0,852	0,876	0,853
0,817	0,837	0,821	0,838	0,821	0,838	0,857	0,858	0,869	0,852
0,813	0,834	0,798	0,817	0,798	0,817	0,857	0,859	0,856	0,855
0,827	0,824	0,827	0,824	0,827	0,824	0,866	0,860	0,870	0,862
0,797	0,804	0,800	0,807	0,800	0,807	0,838	0,850	0,855	0,867
0,780	0,800	0,780	0,800	0,780	0,800	0,862	0,871	0,865	0,873

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı	**Zeynep**	**Uyruğu**	**T. C.**
Soyadı	**ÜNAL**	**Tel no**	**0544 2150800**
Doğum tarihi	**10.10.1981**	**e-posta**	**zeynepunal1010@hotmail.com**

Eğitim Bilgileri

Mezun olduğu kurum		**Mezuniyet yılı**
Lise	**Kazakistan**	**1999**
Lisans	**Selçuk Üniversitesi, Mühendislik-Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü**	**2004**
Yüksek Lisans	**Akdeniz Üniversitesi, Sosyal Bilimler Enstitüsü, Ekonometri A.B.D.**	**2015**
Yüksek Lisans	**Akdeniz Üniversitesi, Sağlık Bilimleri Enstitüsü, Biyoistatistik ve Tıp Bilişimi A.B.D.**	**2017**
Doktora	**Akdeniz Üniversitesi, Sosyal Bilimler Enstitüsü, Ekonometri A.B.D.**	**2019**

İş Deneyimi

Görevi	**Kurum**	**Süre (yıl-yıl)**
MCT (Eğitmen)	**BESAY Bilişim Akademisi**	**2006-2008**
AR&GE Müdürü	**Ela Quality Resort Hotel**	**2008-2014**
Proje Uzmanı	**Batı Akdeniz İhracatçılar Birliği**	**2016-2017**

Yabancı Dilleri	**Sınav türü**	**Puanı**
İngilizce	**YDS**	**88,75**
Rusça	**YDS**	**96,25**

Proje Deneyimi

Proje Adı	**Destekleyen kurum**	**Süre (Yıl-Yıl)**
Batı Akdeniz Tarım ve Gıda Kümesi	**EKONOMİ BAKANLIĞI**	**2016-2017**

Burslar-Ödüller:

1999-2004 Karşılıksız Milli Eğitim Bursu
2000-2004 Başbakanlık Bursu

Yayınlar ve Bildiriler:

Yaman, H., & Ünal, Z. (2018). The validation of the PRISMA-7 questionnaire in community-dwelling elderly people living in Antalya, Turkey. Electronic Physician, 10(9), 7266-7272.

Ekşili N., Ünal Z., İpekçi Çetin E., "Beş Yıldızlı Otel İşletmelerinde Kriz Dönemlerinde Uygulanan Tasarruf Stratejilerinin Bulanık DEMATEL Yöntemiyle İncelenmesi", Business and Economics Research Journal Cilt:6 Sayı:2, Nisan 2017, 259-273

Ünal Z., Asilkan Ö., Canbazoğlu E., "Sağlık Alanında Mobil Uygulama Örneği: Çocuklardaki Gelişimin Büyüme Eğrilerine Göre Değerlendirilmesi", Akademik Bilişim Kongresi, Aydın, 30 Ocak-5 Şubat 2016

Canbazoğlu E., Asilkan Ö., Ünal Z., "Sağlık Alanında Mobil Uygulama Geliştirme Çalışması:Muayene Sırası Bilgilendirme", TBD 32. Ulusal Bilişim Kurultayı, Ankara, 3-5 Aralık 2015, ss.8-13

Ünal Z., Asilkan Ö., Çetin E., Ekşili N., "Otel İşletmelerinde Tedarikçi Seçimi Optimizasyonuna Yönelik Bir Uygulama", Yönetim Bilişim Sistemleri Kongresi, Erzurum, 8-10 Ekim 2015, ss. 423-432.

Ünal Z., Asilkan Ö., Ekşili N., Yardımcı A., "Konaklama İşletmelerinin Verimliliğini Artırma Amacıyla Çağrı Merkezlerinin Kurulumu", Yönetim Bilişim Sistemleri Kongresi, İstanbul, 16-17 Ekim 2014.

Ekşili N., Ünal Z., Batur Z., "İş Yaşamında X Kuşağı Yöneticilerin Algılama Farklılıklarından Kaynaklanan Performans Değerlemeleri Üzerine Bir Araştırma", I.Uluslararası İktisat Kongresi, Prag, Çek Cumhuriyeti, 3-5 Eylül 2014, ss.104-106