T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YÖNTEMLERİ İLE BİLGİSAYAR AĞLARINDA GÜVENLİĞE YÖNELİK ANORMALLİK TESPİTİ

RÜSTEM CAN AYGÜN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
DOÇ. DR. ALİ GÖKHAN YAVUZ

İSTANBUL, 2017

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YÖNTEMLERİ İLE BİLGİSAYAR AĞLARINDA GÜVENLİĞE YÖNELİK ANORMALLİK TESPİTİ

Rüstem Can AYGÜN tarafından hazırlanan tez çalışması 05.06.2017 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Ali. Gökhan YAVUZ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Doç. Dr. Ali. Gökhan YAVUZ

Yıldız Teknik Üniversitesi

Prof. Dr. Selim AKYOKUŞ

Doğuş Üniversitesi

Doç. Dr. Songül Varlı Albayrak

Yıldız Teknik Üniversitesi

Bu çalışma, Yıldız Teknik Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü' nün 2016-04-01-YL01 numaralı projesi ile desteklenmiştir.

ÖNSÖZ

Öncelikle yüksek lisans sürecim boyunca bana her konuda destek olan ve benimle ilgilenen değerli hocam, Doç. Dr. Ali Gökhan YAVUZ'a sonsuz teşekkürlerimi sunarım.

Tüm hayatım boyunca maddi ve manevi olarak yanımda olan ve iyi bir eğitim alabilmem için çok emek veren değerli aileme sonsuz sevgilerimi ve şükranlarımı sunarım.

Son olarak, eğitim hayatıma yön vermiş olan rahmetli ilkokul öğretmenim Hüseyin YAVUZ'a üzerimdeki emeklerinden dolayı şükranlarımı sunuyorum ve kendisini saygıyla anıyorum.

Mayıs, 2017

Rüstem Can AYGÜN

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

a	Basit nöron çıkış değeri
b	Bias değeri
d_i	Veri kümesine ait bir örnek
D	Tüm veri kümesi
J	Hata Fonksiyonu
L	Yapay sinir ağı katman seviyesi
LL	Likelihood fonksiyonunun logaritması
m_i	Küme merkezinden bir örneğe olan uzaklık
P	Likelihood
R	Regülarizasyon fonksiyonu
x	Yapay sinir ağına verilen giriş değeri
W	Nöronlar arasındaki bağlantının ağırlık değeri
W^T	Ağırlığın transpozu
Y	Aktivasyon fonksiyonundan geçirilmiş nöron çıkış değeri
z	Yeniden oluşturulmuş giriş verisi
α	Öğrenme sabiti
Δ	Türev
λ	Regülarizasyon sabiti
μ_y	Yeniden oluşturma hatalarının ortalaması
σ_y	Yeniden oluşturma hatalarının standart sapması
δ	Geri yayılım algoritması için bulunan ağırlık güncelleme değeri
thr_a	Anomali eşik değeri

KISALTMA LİSTESİ

CPU	Merkezi işlem Ünitesi
CUDA	Compute Unified Device Architecture
ÇKP	Çok katmanlı perceptron
DSA	Derin yapay sinir ağı
DVM	Destek Vektör Makineleri
FN	Yanlış Negatif
FP	Yanlış Pozitif
GGOK	Gürültü Giderici Otomatik Kodlayıcı
GPU	Grafik işlem Ünitesi
HKT	Tekrar oluşturma hatalarının kareleri toplamı
KGOMA	Küçük grup olasılıksal meyilli azaltım
LSTM	Long short-term memory
MA	Meyilli azaltım
OK	Otomatik Kodlayıcı
OMA	Olasılıksal Meyilli azaltım
OS	Operating System – İşletim Sistemi
PCA	Principal Component Analysis
RBM	Sınırlandırılmış Boltzman makinesi, restricted Boltzman machine
R2L	Remote to Local
R2U	Remote to User
SGD	Stokastik meyilli azaltım
STS	Saldırı Tespit Sistemi
TN	Doğru Negatif
TP	Doğru Pozitif
U2R	User to Root Attack
YOH	Yeniden oluşturma hatası
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

DERİN ÖĞRENME YÖNTEMLERİ İLE BİLGİSAYAR AĞLARINDA GÜVENLİĞE YÖNELİK ANORMALLİK TESPİTİ

Rüstem Can AYGÜN

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Doç. Dr. Ali Gökhan YAVUZ

Bilgisayar ağlarına yapılan saldırıların etkin bir biçimde tespit edilmesine yönelik bir sistem tasarlanması aktif bir araştırma konusudur. Günümüzde bilinen saldırıların tespiti konusunda yüksek başarım ile çalışabilen statik ve yapay zeka tabanlı yöntemler var olmasına rağmen sıfırıncı gün saldırılarını etkin bir biçimde tespit edecek bir yöntem tasarlanması konusundaki çalışmalar devam etmektedir. Bununla beraber, sıfırıncı gün saldırılarının tespiti konusunda makine öğrenmesi yöntemlerinin geçmişte önerilmiş statik yöntemlere göre çok daha başarılı sonuçlar verdiği bilinmektedir. Bu sebeple makine öğrenmesi tabanlı ağ saldırı tespiti yöntemleri her geçen gün daha da popülerleşmektedir. Bu tez kapsamında makine öğrenmesi yöntemlerinin bir alt kolu olan ve mevcut makine öğrenmesi yöntemlerine göre birçok problemin çözümü konusunda çok daha başarılı sonuçlar veren derin öğrenme yöntemleri kullanılarak, NSL-KDD veri kümesi üzerinde anomali tespiti tabanlı ağ saldırı tespiti yapılmıştır.

Tez kapsamında kullanılan yöntemler uç değer tespiti tabanlı anomali tespiti ve sınıflandırıcı tabanlı anomali tespiti olarak iki ayrı kategori altında incelenmiş ve sonuçları kıyaslanmıştır. Uç değer tespiti yaklaşımı için derin öğrenme yöntemlerinin bir alt bileşeni olan otomatik kodlayıcıların anomali tespiti konusundaki performanslarını arttırmak amacı ile stokastik bir anomali eşik değeri tespit yöntemi önerilmiş ve önerilen yöntemi kullanan deterministik otomatik kodlayıcı ve gürültü giderici otomatik kodlayıcı modelleri, NSL-KDD'nin içinde bulunan KDDTest+ kümesi üzerinde sırası ile %88,28 ve %88,65 doğruluk oranları ile başarım sağlamıştır.

Sınıflandırıcı yaklaşımı için ise mevcut derin öğrenme yöntemlerinden deterministik otomatik kodlayıcı ve gürültü giderici otomatik kodlayıcı tabanlı derin yapay sinir ağlarının KDDTest+ üzerindeki anomali tespit performansları ölçülmüş ve sırası ile %79,25 ve %83,39 doğruluk oranları ile başarım sağlanmıştır. Elde edilen sonuçlar önerilen eşik değeri belirleme yöntemini kullanan, gürültü giderici otomatik kodlayıcı yönteminin tekil bir yöntem olarak literatürdeki diğer tüm tekil yöntemlerden daha başarılı olduğunu ve hibrit yöntemlerin de en başarılı olanlarına eş değer performansa sahip olduğunu göstermiştir.

Anahtar Kelimeler: Otomatik kodlayıcı, derin öğrenme, anomali, tespiti, ağ güvenliği, saldırı tespit sistemleri, yapay sinir ağları

ABSTRACT

DEEP LEARNING BASED NETWORK ANOMALY DETECTION

Rüstem Can AYGÜN

Department of Computer Engineering

MSc. Thesis

Advisor: Assoc. Prof. Dr. Ali Gökhan YAVUZ

Designing a system which can detect network intrusions effectively is an active research topic. There are several successful intrusion detection systems which rely on either static methods or artificial intelligence methods, for detecting known attacks. However, the research which is about designing a system to detect zero-day attacks with a high success rate still continues. In addition, the popularity of the machine learning algorithms based intrusion detection models is increasing day by day due to their better performance on zero-day attack detection compared to previously proposed static methods. Within the context of this thesis, in order to detect the intrusions of the NSL-KDD dataset in anomaly detection manner, we used deep learning methods which are the part of machine learning algorithms and perform better than the any other machine learning algorithms at creating solutions for the variety of problems.

The implemented methods in this thesis are examined in outlier detection and classification based anomaly detection categories separately. For outlier detection methods, we proposed a stochastic anomaly threshold determination technique to improve the anomaly detection performance of the autoencoders which are part of deep learning methods. The proposed anomaly detection models based on deterministic autoencoder and denoising autoencoder performed with 88.28% and 88.65% accuracy scores respectively on KDDTest+ dataset which is the test dataset of the NSL-KDD dataset. For classification methods, we used denoising autoencoder

based deep neural network and deterministic autoencoder based deep neural network methods which are already existing deep learning approaches, to detect the intrusions of the KDDTest+ dataset and they performed with 79.25% and 83.39% accuracy scores respectively. The obtained results from the denoising autoencoder model which uses our proposed threshold determination technique shows that, as a singular model it outperforms any other singular anomaly detection methods and it also performs almost the same as the newly suggested hybrid anomaly detection models.

Keywords: Autoencoder, deep learning, anomaly detection, network security, intrusion detection systems, artificial neural networks

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Bilgisayar ağları ve mobil ağların giderek yaygınlaşması ve bu alanda yaşanan gelişmeler, bu teknolojilerin; e-ticaret, bankacılık ve savunma sanayi gibi birçok önemli alanda yaygın ve etkin olarak kullanılmasını sağlamıştır. Ağ kullanımındaki bu yaygınlaşma, bilgisayar ağlarına yapılan saldırıların sayısını ve çeşitliliğini her geçen gün arttırmaktadır[1]. Bu sebeple bilgisayar ağlarının güvenliğinin sağlanması; veri bütünlüğünün korunması ve hizmetin sürekliliğinin devamı açısından büyük önem taşımaktadır.

Ağ güvenliğinin sağlanması amacıyla yaygın bir şekilde kullanılan Saldırı Tespit Sistemleri (STS), yazılımsal ve/veya donanımsal olarak ağdaki ve ağa bağlı cihazlardaki faaliyetleri analiz eden ve bu analizler sonucunda çeşitli savunma mekanizmalarını tetikleyen güvenlik sistemleridir.

Saldırı tespit sistemlerinde kullanılan yöntemler iki başlık altında incelenebilir[2];

- İmza Tabanlı Saldırı Tespiti
- Anomali Tabanlı Saldırı Tespiti (Anomali Tespiti)

İmza tabanlı STS'ler, daha önceden meydana gelmiş saldırılara ait belirgin özellikler olan; sistem çağrıları, izinler, ağ erişimi ve çalışma periyodu gibi bilgileri bir imza veri tabanında saklarlar ve sonrasında programları veya ağ akışını daha önceden kaydedilmiş bu imzalar aracılığıyla sınıflandırırlar[3]. Bu şekilde statik olarak analiz yapan STS'lerde, imza veri tabanının sürekli güncel tutulması sistemin başarısı

açısından kritik önem taşımaktadır. Ek olarak, bu sistemler güncel bir veri tabanına sahip olsalar bile sıfır gün saldırıları (zero-day attacks) karşında başarısız sonuç vermektedirler. Bu yüzden değişen saldırı doğasına uyum sağlayabilen ve yeni saldırılara karşı başarılı sonuç veren akıllı bir STS geliştirmek adına makine öğrenmesi tabanlı anomali tespit sistemleri üzerinde yapılan çalışmalar önem kazanmıştır[2].

Anomali tespit modelleri eğitim yöntemi bakımından, *gözetimli eğitim, gözetimsiz eğitim*[4] ve *yarı gözetimli eğitim*[5] yöntemlerini kullananlar olarak üç alt başlığa ayrılmaktadır. Gözetimli eğitim yöntemlerinde model, normal ve anormal olarak etkilenmiş olan eğitim verisi kullanılarak eğitilir ve bu şekilde saldırı tespiti yapılır. Bu yöntemler her ne kadar imza tabanlı saldırı tespit sistemlerine göre daha başarılı sonuçlar verseler de eğitim kümesinde bulunmayan yeni saldırıların tespiti konusunda başarılı değildirler[2]. Bu yöntemlerin aksine gözetimsiz veya yarı gözetimli eğitim yöntemleri, sıfırıncı gün saldırılarının tespitinde daha başarılı sonuçlar vermektedirler[3]. Gözetimsiz eğitimde, model tamamen etiketsiz bir veri kümesi ile eğitilmektedir ve veri kümesindeki normal ve anormal verilerin ayrı ayrı kümelenmesi amaçlanmaktadır[4]. Yarı gözetimli eğitimde ise model, sadece normal verilerden oluşan bir veri kümesi ile eğitilir ve daha sonra verilen test örneklerinden kendisine benzemeyenleri anormal olarak sınıflandırması beklenir[5].

Anomali tespiti; sınıflandırma, istatistiksel yöntemler, kümeleme – uç değer tespiti ve hibrit-birleştirilmiş yöntemler gibi farklı yaklaşımlar kullanılarak yapılabilir. Sınıflandırma tabanlı; *Bayes ağları, Yapay Sinir Ağları* ve *Destek Vektör makineleri* gibi yöntemler gözetimli veya yarı gözetimli olarak eğitilerek anomali tespiti amacıyla kullanılmıştırlar[3]. Ek olarak, istatistiksel tabanlı yöntemlerden; *mixture model yöntemi*[6], *principal component analysis (PCA)*[7] ve *istatistiksel sinyal işleme teknikleri*[8] anomali tespiti amacıyla kullanılmıştır. İstatistiksel yöntemler genel olarak normal verinin profilini oluşturup bu profile uymayan verileri anormal olarak adlandırdıkları için genel olarak yarı-gözetimli bir şekilde eğitildikleri söylenebilir. Ek olarak, kümeleme-uçdeğer tespiti yöntemleri; *k-ortalama (k-means)*, *k-medoids ve EM yöntemleri* anomali tespiti amacıyla kullanılmıştır[9], [10].

Makine öğrenmesi yöntemlerinin başarımının arttırılabilmesi için kritik bir gereksinim olan önemli özelliklerin seçilmesi ve etiketli veri oluşturulması işlemleri; alanında uzman kişiler tarafından yapılan, uzun zaman alan ve çok zahmetli bir iştir. Her geçen gün artan veri boyutları ve özellik boyutları da dikkate alındığında bu işlemlerin insan eliyle yapılması daha da zorlaşmaktadır. İşte bu probleme bir çözüm olarak geliştirilmekte olan derin öğrenme yöntemlerinin, görüntü ve ses işleme, doğal dil işleme, sinyal işleme ve bilgi çıkarımı gibi birçok alanda uygulanıp büyük ölçekteki veriler için klasik yöntemlere göre daha başarılı sonuçlar vermesi[11] ve bu yöntemlerin son yıllarda ağ anomalilerinin tespitinde de kullanılmaya başlanması, derin öğrenme yöntemlerini kullanarak genel ağ anomali tespiti başarımının arttırılmasına yönelik bir çalışma yapmamız konusundaki motivasyonumuzu oluşturmuştur.

Ağ anomalilerinin tespiti amacıyla; CAIDA[12], DEFCON[13], KDDCUP'99 [14], NSL-KDD [15],[16], ISCX UNB[17], gibi birçok performans değerlendirme veri kümesi kullanılmaktadır. Bu veri kümelerinden KDDCUP'99, ağ anomali tespiti konusunda çalışan araştırmacıların en yaygın olarak kullandığı veri kümesidir [18]. Ancak KDDCUP'99 veri setinin bulunan bazı hataların bu veri seti üzerinde yapılan değerlendirmeleri etkilediği saptanmış ve bunun üzerine KDDCUP'99 veri setinin hatalarından arındırılmış hali olan NSL-KDD veri kümesi oluşturulmuştur[15],[16]. NSL-KDD veri kümesi günümüzde saldırı tespiti, anomali tespiti konusunda çalışan araştırmacılar tarafından aktif bir şekilde kullanılmaktadır. Bu tez kapsamında uygulaması yapılan yöntemlerin, mevcut literatürdeki çalışmaların geniş bir bölümüyle karşılaştırılabilir olması adına değerlendirme veri kümesi olarak NSL-KDD seçilmiştir ve NSL-KDD üzerinde makine öğrenmesi yöntemlerini kullanarak anomali tespiti yapan çalışmalar detaylı olarak incelenmiştir.

NSL-KDD veri kümesi; KDDTrain+, KDDTrain+_%20, KDDTest+ ve KDDTest-21 alt veri kümelerinden oluşmaktadır. Literatürdeki çalışmalar modelin eğitimi ve değerlendirilmesi bakımından çeşitlilik göstermektedirler. Bu çalışmaların bir bölümünde KDDTrain+'nın %20'si eğitim için kullanılmış ve eğitilen model KDDTest+ ve KDDTest-21 üzerinde test edilmiştir. Bu şekilde değerlendirme yapılan bir çalışmada KDDTest+ üzerinde en yüksek doğruluk oranı %82,02 ve KDDTest-21 üzerindeki en yüksek doğruluk oranı ise %66,16 olarak bildirilmiştir [16]. Bu çalışmaya göre en başarılı

yöntemler ise; NB Tree, C4.5 ve Random Tree algoritmalarıdır. Benzer şekilde değerlendirme yapılan bir çalışmada KDDTest+ üzerinde bulanık sınıflandırıcı (fuzzy classifier) - genetik programlama yöntemi ile ise %82,74 doğruluk oranı elde edilmiştir[19]. Bu çalışmalara ilaveten NB Tree - Random Tree hibrit yöntemi ile KDDTest+ üzerinde %89,24 doğruluk oranı elde edilmiştir[20].

İlk olarak Hawkins'in yaptığı bir çalışmada [21] anomali tespiti amacı ile kullanılmış olan otomatik kodlayıcıların (OK) kullanımı son yıllarda daha da yaygınlaşmıştır[22]. Karşılaştırmalı bir çalışmada; *OK, Gürültü Giderici Otomatik Kodlayıcı (GGOK), doğrusal PCA* ve *kernel PCA* yöntemlerinin telemetri verisindeki anomalilerin tespiti konusundaki başarıları kıyaslanmıştır[23]. Buna ek olarak OK ve çekirdek yoğunluk (kernel density) tespiti yöntemleri hibrit bir biçimde KDDTest+ üzerinde uygulanmış ve her saldırı tipinin ayrı ayrı tespit edilmesine yönelik başarılı sonuçlar elde edilmiştir[22].

Bu çalışmaların yansıra derin öğrenme yöntemlerinden Seyrek OK (sparse-autoenoder) - YSA yöntemi de NSL-KDD üzerinde uygulanmıştır ve KDDTest+ üzerinde %88,39 doğruluk oranı elde edilmiştir[24].

1.2 Tezin Amacı

Bu tez kapsamında;

- Ağ anomali tespiti başarımının arttırılabilmesi adına temel yapay sinir ağı modelleri ve çeşitli derin öğrenme yöntemlerinin araştırılması ve gerçeklenmesi,
- Mevcut ağ anomali tespit başarımını arttırmaya yönelik yeni bir anomali tespit modeli önerilmesi,
- Bu uygulamalar sonucunda elde edilen sonuçların, mevcut literatürdeki makine öğrenmesi tabanlı anomali tespit yöntemleri arasından en başarılı olanlar ile kıyaslanarak, gerçeklenen derin öğrenme yöntemlerinin ağ anomali tespiti konusundaki eksi ve artı yönlerinin tespit edilmesi,

- Derin öğrenme yöntemlerinin en performanslı şekilde eğitilebilmesine olanak sağlayan yazılım ve donanımların araştırılması ve bu doğrultuda bir hesaplama sunucusu kurulması

amaçlanmaktadır.

1.3 Hipotez

Bu tez kapsamında; literatürde mevcut olan derin öğrenme yöntemlerinden; deterministik OK tabanlı derin yapay sinir ağı, GGOK tabanlı derin yapay sinir ağı ve derin öğrenme yöntemlerinin bir alt bileşeni olan otomatik kodlayıcıların tekil yöntem olarak[25] ağ anomali tespiti konusundaki performansı karşılaştırılmıştır. Yapılan çalışma kapsamında OK'ların tekil yöntem olarak anomali tespit performansını arttırmak amacı ile güncel literatürde tariflenen yöntemlerden farklı olarak stokastik bir anomali eşik değeri tespit yöntemi (SAEDT) önerilmiştir. Önerilen yönteme dayalı anomali tespit modelleri ve bahsi geçen mevcut yöntemler, güncel bir STS değerlendirme veri kümesi olan NSL-KDD[16],[15] üzerinde test edilmiş ve elde edilen sonuçlar, birbirleri ile ve literatürde benzer değerlendirme ölçütlerine sahip olan en başarılı tekil ve hibrit anomali tespit yöntemleri [16], [20],[19] ile karşılaştırılmıştır. Testlerin NSL-KDD veri kümesi içinde yer alan KDDTest+ üzerinde yapılması, önerilen yöntemlerin, eğitim kümesinden farklı olasılıksal dağılıma sahip olan ve yeni atak tiplerini içeren bir test ortamındaki başarımlarının tespiti açısından önem taşımaktadır. Ek olarak, önerilen SAEDT yöntemini kullanan OK'ların anomali tespiti konusunda hibrit yöntemlere başarılı birer alternatif olabilecekleri gösterilmiştir.

BÖLÜM 2

SALDIRI TESPİT SİSTEMLERİ

Bilgisayar ağlarına ve sistemlerine yapılan saldırılar, yetkilendirilmemiş bir erişim ile ağın ve/veya ağa bağlı sistemlerdeki servis devamlılığını, veri bütünlüğünü ve gizliliğini tehdit ederler. Bu saldırılar her ne kadar uzmanlar tarafından konfigürasyonu yapılan güvenlik duvarları ile engellenmeye çalışılsa da gelişen saldırı doğası sadece güvenlik duvarına dayalı bir ağ savunma yöntemini etkisiz kılmıştır [1]. Bu yüzden her yönü ile etkin bir ağ ve/veya sistem savunma mekanizması olması adına oluşturulmuş olan saldırı tespit sistemleri;

- Kullanıcıların sistem ve ağ aktivitelerini izlenmesi
- Sistemin olası saldırılara karşı hazır olacak şekilde düzenlenmesi
- Sistemin ve veri bütünlüğünün korunması
- Anormal faaliyet tespiti (Saldırı örüntülerinin tanınması)
- Kullanıcı ihlallerinin tespiti

Şeklinde önemli fonksiyonalitelere sahiptirler[18].

Saldırı tespit sistemleri, savunulması gereken ortamın şartlarına göre farklı şekilde özelleştirilmiştirler. Saldırı tespit sistemleri uygulama alanları bakımından;

- Sistem Tabanlı Saldırı Tespit Sistemleri
- Ağ Tabanlı Saldırı Tespit Sistemleri

Şeklinde iki başlık altında incelenebilirler.

2.1 Sistem Tabanlı Saldırı Tespit Sistemleri

Sistem tabanlı STS'ler (Host Based Intrusion Detection Systems) sistemin harici çıkışlarında meydana gelen faaliyetlerden daha çok; bellek erişimi, işlemlerin durumları, disk erişimi ve kullanılan sistem çağrıları gibi sistemin iç bileşenlerinin takip edilerek sistemdeki zararlı yazılım ve kullanıcı faaliyetlerinin tespit edilmesine yönelik bir analiz yapmaktadırlar. Buradan da anlaşılacağı gibi sistem tabanlı STS'ler her sistemin kendi özelinde koruma sağlayan güvenlik uygulamalarıdır[26].

2.2 Ağ Tabanlı Saldırı Tespit Sistemleri

Ağ saldırı tespit sistemlerinde (Network Based Intrusion Detection), ağ ve ağa bağlı bulunan cihazlar için zararlı olan paketlerin tespit edilmesi amaçlamaktadır. Ağ saldırılarının oluşma sebebi, ağa yetkisiz bir şekilde erişim sağlanarak verilerin çalınması veya ağın çalışmasını engellemeyi amaçlayan aktivitelerdir. Standart bir bilgisayar/mobil ağ ortamında dış dünyaya internet aracılığıyla bağlanılır. Ağ tabanlı saldırı tespit sistemleri dış dünyadan yerel ağa gelen (gelen trafik) ve yerel ağdan dış dünyaya (giden trafik) çıkış yapan tüm paketleri anormal bir faaliyetin varlığının tespiti amacıyla izlerler[18].

Ağ saldırısı, belirli bir ağı ele geçirmek, işleyişini bozmak amacı taşıyan zararlı faaliyetler olarak adlandırılmaktadır. Ağın belirli güvenlik konfigürasyonlarının doğru yapılmaması veya yanlış ağ tasarımı gibi sebepler, bu kusurları barındıran ağların saldırılara açık olmasına sebep olmaktadır. Ağ tabanlı saldırılar çok geniş çeşitliliğe sahip olsalar da genel olarak 5 ana kategori altında toplanmaktadırlar[15];

Hizmet Reddi (Denial of Service): Belirli bir ağın veya sunucunun hizmet vermesini engelleyecek şekilde çok sayıda istek gönderilmesi soncunda sunucunun legal hiçbir isteğe cevap veremeyecek duruma getirilmesidir.

Sorgulama (Probe): Belirli bir ağdan veya sistemden, saldırı için ön hazırlık yapmak amacı ile kullanılması planlanan bilgilerin elde edilmesi olarak tanımlanır. Bu saldırı tipinde sisteme doğrudan bir zarar verilmemektedir ancak bu saldırı neticesinde elde

edilen bilgiler daha güçlü bir saldırı için zemin oluşturacağından dolaylı yoldan bir zarar teşkil etmektedir.

Yönetici Yetkilerini Ele Geçirme (User to Root Attack): Saldırganın, belirli bir ağdaki veya sistemdeki yönetici kullanıcı (root/ administrator) yetkilerini illegal yollardan (*packet sniffing, session hijacking, password attack*) ele geçirerek sistemde yetkisi olmayan işlemleri gerçekleştirmesidir.

Uzak Bağlantı ile Yönetici Yetkilerini Ele Geçirme(Remote User to Root (R2U /R2L)): Saldırganın uzaktaki bir hedef makineye o makineye ait legal bir kullanıcı gibi giriş yaparak hedef makinenin ağından bilgi transferi yapmasıdır.

Her ne kadar tüm ağ tabanlı saldırı tespitlerindeki ortak hedef, ağ üzerindeki zararlı faaliyetlerin tespit edilmesi olsa da saldırı tespit yöntemindeki yaklaşım bakımından ağ tabanlı saldırı tespitleri ikiye ayrılmaktadır;

- Kötüye Kullanım – İmza Tabanlı Saldırı Tespiti
- Anomali Tespiti / Anomali Tabanlı Saldırı Tespiti

2.2.1 Kötüye Kullanım – İmza Tabanlı Saldırı Tespiti

Kötüye kullanım – imza tabanlı saldırı tespitinde (Misuse-Signature Based Intrusion Detection), geçmiş saldırılara ait paketlerin seçilmiş olan özelliklerinden imzalar oluşturularak bir imza veri tabanında saklanır ve daha sonra ağ üzerinden taşınan paketlerin kaydedilmiş olan imzalar ile eşleşme durumları kontrol edilir. Bu tip sistemler bilinen tüm saldırı tiplerini çok yüksek bir başarım ile tespit edebilmektedirler [27], [28] . Ancak herhangi bir saldırının tüm kombinasyonlarını kapsayacak şekilde bir imza çıkarılması hem çok zaman alan hem de güvenlik alanında uzman olan kişiler tarafından yapılması gereken bir iş olduğundan imza veri tabanının güncellemesi yüksek maliyetlidir.

Kötüye kullanım tespitindeki amaç bilgisayar ağlarına veya sistemlerine yapılan spesifik saldırıları tespit etmek iken anomali tespitindeki amaç, normal sistem profiline uymayan her türlü faaliyetin tespitini yapmaktadır.

Kötüye kullanım tespiti tabanlı sistemlerin en popüler olanlarından biri SNORT [27] ağ paketlerini IP, transport katman protokolü ve paketin veri yükü (payload) bölümlerini kullanarak değerlendirir ve bu bölümleri önceden meydana gelmiş saldırılara ait imzalar üzerinden tanımlanmış kurallar ile karşılaştırarak saldırı tespiti yapar. SNORT 15 yıldır açık kaynaklı olarak kural veri tabanını yeni saldırılara karşı güncel tutmaya çalışmaktadır. Bugün SNORT sisteminde 20.000'i aşkın kural tanımı bulunmaktadır. Ancak bu sistemin paket seviyesinde saldırı tespiti yapması, çok geniş ölçekli ve yoğun trafiğe sahip ağlara uygulanabilmesini engellemektedir[18].

2.2.2 Anomali Tespiti Tabanlı Saldırı Tespiti

Anomali veririnin büyük çoğunluğundan farklı bir yapıya sahip ve genellikle sistem dışında başka bir mekanizma tarafından üretildiğinden şüphe duyulan veriler olarak tanımlanmaktadır[3]. Normal veriden farklı karakteristiğe sahip bu verilerin tespit edilmesi için geliştirilmiş ağ anomali tespit sitemleri sadece ağdaki zararlı faaliyetlerin tespiti için değil; *tıp ve toplum sağlığı, finans sahtekarlığı tespiti, askeri uygulamaları, endüstriyel uygulamalar, sensor ağlar, robotik uygulamalar* ve *astronomik verilerin analizi* gibi bir çok alanda da kullanılmaktadırlar[29]. İmza tabanlı STS'lerin, yeni saldırı tiplerinin tespiti konusunda düşük başarım göstermelerinin bir sonucu olarak STS'ler üzerine yapılan araştırmalar, istatistiksel yöntemler ve makine öğrenmesi algoritmalarını temel alan anomali tespit yöntemleri üzerine yoğunlaşmıştır.

Sistemde bulunan anomalilerin tespit edilmesi ve bu tespitin sonucuna göre çeşitli aksiyonlar alınması kritiktir. Çünkü bir anomali, siteme yapılan bir saldırıyı veya sistemdeki bir performans problemini işaret ediyor olabilir [18].

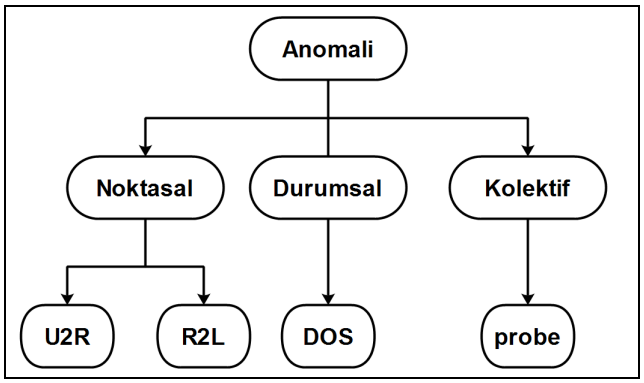
Sistemde meydana gelen anomaliler birbirlerinden farklı karakteristiğe sahiptirler ve genel olarak anomali türleri üç temel kategoriye ayrılmaktadırlar[3];

Noktasal Anomali: Belirli bir örnek, normal verinin tamamından farklı bir karakteristiğe sahipse noktasal anomali olarak kabul edilir.

Durumsal Anomali: Belirli bir örnek, sadece belirli durumlarda normal veriden farklı oluyor ise durumsal anomali olarak kabul edilir.

Kolektif Anomali: Belirli bir grup veriyi oluşturan örnekler bireysel olarak değerlendirildiklerinde normal veriden farklı olarak nitelendirilmezken, toplu şekilde değerlendirildiklerinde normal veriden farklı bir yapı ortaya koyuyorlar ise bu şekildeki veri grupları kolektif anomali olarak adlandırılırlar.

Şekil 2. 1'de daha önceden tariflenmiş olan DOS, probe, U2R, R2U/R2L saldırıları tiplerinin anomali türlerine göre sınıflandırılması gösterilmektedir.



Şekil 2. 1 Ağ saldırı çeşitlerinin anomali perspektifinden sınıflandırılması

2.2.2.1 Anomali Tespiti Tabanlı Saldırı Tespit Sisteminin Alt Bileşenleri

Anomali tespiti tabanlı saldırı tespit sistemi; *anomali tespit motoru, eşleştirme mekanizması, referans veri, konfigürasyon verisi, alarm mekanizması*, *insan analist*, *post-processing*, *paket yakalama* ve *güvenlik yöneticisi* gibi alt bileşenlerden oluşmaktadır[18] .

Anomali tespit motoru: Tüm çevrimiçi ve çevrimdışı saldırıların tespiti bu mekanizma üzerinden yapılmaktadır. Bilinen saldırılar için kural tabanlı yöntemler, sıfırıncı-gün-saldırıları için ise anomali tespit yöntemleri uygulanmaktadır.

Eşleştirme mekanizması: İzlenen ağdaki verilerin bilinen saldırı profillerine uyup uymadığının tespit edilmesini sağlayan sistemdir.

Referans veri: Saldırı tespit sistemindeki referans veri; *saldırı imzaları, normal veri profili, yetkiler* ve *rollerden* oluşmaktadır.

Konfigürasyon verisi: STS'nin referans verilerinin güncellenmesi ile alakalı tutulan verilerdir. Yeni oluşturulan imzalar ile mevcut referans verisi arasında bir uyuşmazlık

oluştuğunda konfigürasyon verisinin yardımı ile imzalar*, kurallar* ve *profiller* güncellenir.

Alarm mekanizması: Anomali tespit motorundan alınan anormal veri tespitlerine ilişkin çeşitli uyarılar oluşturan mekanizmadır.

İnsan analist: Alarmlar ve tespitler sonucunda çeşitli aksiyonları alan yetkili kişi.

Post-processing: Tespit edilen saldırıların gerçekten saldırı olup olmadığının araştırılması bu modül içerisinde yönetilir.

Paket yakalama modülü: Paket yakalama modülünde ağ verileri, paket veya akış (flow) seviyesinde yakalanırlar. Ağ paketleri, wireshark gibi araçlar ile yakalanabilen salt bilgi iken, akış (*flow*), birden çok paketin özet bilgisi olarak kabul edilmektedir. Yakalanan ağ paketleri birleştirilerek akış çıkarılabileceği gibi doğrudan akış şeklinde veri yakalamaya imkan tanıyan; *Nfdump2, NfSen3* ve *Cisco Netflow V.94*. gibi araçlar da vardır.

Güvenlik yöneticisi: İmza veri tabanının güncelleme işlemlerini yapan kişi.

2.2.2.2 Anomali Tespit Başarımını Etkileyen Faktörler

Anomali tespitinde; *veri yapısı, önemli verilerin seçilmesi, mesafe ölçüm metrikleri* ve *modelin eğitim türü* gibi kritik öneme sahip birçok kriter bulunmaktadır[18], [26].

Veri yapısı: Anomali tespit yöntemlerinin bir veri kümesi üzerinde uygulanabilir olmasını doğrudan etkileyen faktör veri yapısıdır[26]. Birçok makine öğrenmesi yöntemi giriş verisi olarak nümerik değerler kullanmaktadır. Ancak veri kümeleri kategorik özellikler de içermektedir. Bu kategorik özelliklerin anomali tespit modelinde etkin bir biçimde kullanılabilmesi için nümerik değerlere dönüştürülmesi gerekmektedir.

Önemli Özelliklerin seçilmesi: Veri kümesinin gereksiz özelliklerden arındırılmasıyla modelin karmaşıklığı azalır, başarımı artar ve bunun sonucunda genelleştirilmiş bir model ortaya çıkartılır[18]. Özellik seçimi uzman kişiler tarafından el ile yapılabileceği gibi makine öğrenmesi yöntemlerinden yararlanılarak otomatik olarak da yapılabilir.

Mesafe ölçümü yöntemleri: Özellikle uç değer ve kümeleme yöntemlerinde, test edilecek verilerin normal profile olan uzaklığının ölçülerek anormal veya normal şekilde

etiketleme yapılmasında kullanılan en temel araç mesafe ölçüm metrikleridir. Bu yüzden kullanılan mesafe ölçüm metriği anomali tespit başarımını doğrudan etkilemektedir[4]. Bununla beraber, her veri kümesine ve probleme ilişkin gereksinim farklıdır. Dolayısıyla kullanılacak olan mesafe ölçüm metriğinin belirlenmesi, problem ve veri kümesi bazında değişmektedir.

Eğitim Türü: Anomali tespit başarımı konusunda en az diğer faktörler kadar etkin başka bir metrik ise modelin eğitim türüdür. Anomali tespiti konusunda üç farklı eğitim modeli uygulanabilir[26];

***Gözetimli eğitim tabanlı anomali tespiti*:** Model, normal ve anormal sınıflara ait örneklerin ikisini de içerecek şekilde gözetimli olarak eğitilir ve sonrasında verilen test örneklerinin hangi sınıfa ait olduğunu tahmin etmeye çalışır. Modelin, eğitildiği veri ve sınıfları tanıma konusundaki başarımı diğer yöntemlere göre daha yüksek olsa da genellikle etiketli veri kümelerinin az sayıda bulunması, özellikle de anormal olarak etiketlenmiş verilerin yetersizliği bu yöntemlerin her probleme uygulanabilmesini engellemektedir.

Yarı-gözetimli eğitim tabanlı anomali tespiti: Sadece normal veriler kullanılarak eğitilen model kendisine belirli bir eşik değerinden uzaktaki verileri anormal olarak etiketler. Eğitim sadece normal veri üzerinden yapıldığından hem etiketli veri bulma konusundaki sıkıntı daha azdır hem de bilinmeyen saldırıların tespiti konusundaki başarım daha yüksektir. Bunun yanı sıra bazı çalışmalarda sadece anormal veri üzerinden eğitim yapılarak anomali tespiti yapılmıştır. Ancak anormal verinin az sayıda olması ve zor elde edilmesi, sadece anormal veriye dayalı eğitim yönetiminin çok nadir olarak uygulanmasına sebep olmuştur[26].

Gözetimsiz eğitim tabanlı anomali tespiti: Gözetimsiz eğitim tabanlı yöntemlerde kullanılan eğitim kümesindeki verilerin etiketleri bilinmemektedir. Ancak, veri kümesinde anormal veri yoğunluğunun normal veri yoğunluğundan çok daha az olduğu ön kabulü yapılarak anomali tespiti yapılabilmektedir. Burada amaç, verinin anormal ve normal olarak ayrı ayrı gruplanmasıdır. Tamamen gözetimsiz olarak eğitilen modellerin birçoğu yüksek yanlış pozitif oranına sahip olduğu için bu yöntemler genellikle yarı-gözetimli yöntemler ile beraber kullanılmaktadırlar.

Aktif bir araştırma konusu olan ağ anomali tespitinde henüz evrensel olarak kabul görmüş bir yöntem bulunamamıştır. Ağ verisi içerisinde bulunan gürültü, genellikle bu yöntemlerin yanlış alarm vermelerine sebep olmaktadır. Ayrıca anomali tespit yöntemlerinin performans değerlendirmelerinin yapılması amacı ile oluşturulan etiketli kümelerin azlığı da bu alandaki çalışmaların hızını azaltmaktadır. Bunlara ek olarak, gelişen saldırılarla beraber ağ verisindeki normal veri kavramı hızla değişmektedir. Yani bugün normal olarak kabul edilen bir örüntü gelecekte değişen şartlar dolayısıyla anormal olarak kabul edilebilir. Bu yüzden, bu dönüşüme ayak uydurabilecek dinamik çözümlere ihtiyaç duyulmaktadır[3].

2.2.2.3 Anomali Tespit Yöntemleri

Anomali Tespit yöntemleri, modelin eğitim türüne göre gözetimli, gözetimsiz veya yarı-gözetimli anomali tespit yöntemleri şeklinde gruplanabilecekleri gibi modelinin temelini oluşturan yöntemin türüne göre; *sınıflandırma tabanlı, istatistiksel model tabanlı, uç değer tespiti - kümeleme tabanlı* ve *hibrit - birleştirilmiş* (hybrid-ensemble) yöntemler tabanlı anomali tespit yöntemleri şeklinde de gruplanabilirler.

Makine Öğrenmesi Tabanlı Saldırı Tespit Sistemleri

Makine öğrenmesi yöntemleri, belirli bir problemin çözümüne yönelik olarak bilgisayarların, sürekli olarak programlanmaya ihtiyaç duymadan, geçmiş verilerin bilinen özellikleri üzerinden eğitilerek; sınıflandırma, kümeleme veya tahmin yapmasına olanak tanırlar[28]. Makine öğrenmesi algoritmalarını kullanan anomali tespit yöntemleri; sınıflandırıcılar, uç değer tespiti - kümeleme tabanlı yöntemler ve hibrit - birleştirilmiş yöntemler olmak üzere üç alt başlık altında incelenmektedir.

a) Sınıflandırma Yöntemleri Tabanlı Anomali Tespiti

Sınıflandırma, gözlemlenen verilerin daha önceden sayısı ve karakteristik özellikleri bilinen kategorilere ayrılması işlemidir. Belirli bir veri kümesini sınıflandıracak olan model, verileri ait oldukları sınıflara ayırmak için sınıf karar fonksiyonlarını belirler. Sınıf karar fonksiyonu, basit sınıflandırma yöntemleri için doğrusal iken daha karmaşık problemler için doğrusal olmayan bir yapıya sahiptir[18].

Sınıflandırma yöntemi ile anomali tespiti yapacak olan model, eğitim için hem normal hem de anormal olarak etiketlenmiş veri ile gözetimli olarak eğitilmektedir. Ancak bu modeller genelde yeni saldırıların tespiti konusunda başarılı değildirler. Çünkü gözetimli olarak eğitilmiş bir model kendi eğitim kümesinde olmayan bir sınıfa ait verileri sınıflandıramamaktadır. Gözetimli sınıflandırma ile anomali tespiti yapan modelin yüksek başarımla çalışabilmesi için yeni saldırıları sınıflarının sürekli olarak modelin eğitim verisine dahil edilmesi ve modelin tekrardan eğitilerek güncellenmesi gerekmektedir. Ancak bu işlem çok fazla zaman aldığı için güncelleme işlemi bitene kadar yapılan yeni saldırılar sisteme hedefledikleri zararı verebilirler. Dolayısıyla yeni saldırı tiplerinin tespiti için gözetimli eğitim yöntemleri tercih edilmez. Buna bir alternatif olarak yarı-gözetimli olarak eğitilen sınıflandırıcılar sadece normal veri üzerinden eğitilip normal profile uymayanları anormal olarak kabul ettikleri için yeni saldırıların tespiti konusunda daha başarılıdırlar. Ancak bu yöntemde de modelin eğitimi sonucu çıkan normal profilin güvenilir olması gerekmektedir. Çünkü modelin oluşturduğu normal profile dahil edilmemiş bir normal veri, model tarafından anormal olarak etiketlenirken, modelin oluşturduğu normal profile, normal olduğu düşünülerek eklenmiş bir anormal veri de modelin, yanlışlıkla eklenmiş olan bu anomaliye benzer verileri doğru şekilde sınıflandıramamasına sebep olur[3]. Sınıflandırma yöntemleri bilinen saldırıları tespit etme ve kendilerini değişen saldırılara karşı güncelleme esnekliklerine rağmen çok fazla kaynak tüketmeleri ve etiketli veriye ihtiyaç duymaları gibi dezavantajlara da sahiptirler.

Bayes Ağları: Bayes ağları, verilere ait özellikler arasında olasılıksal ilişkileri modellerler. Bu yöntemin istatistiksel yöntemler ile kullanıldığında anomali tespiti konusunda daha avantajlı olduğu tespit edilmiştir[27],[30]. Ek olarak, Bayes ağlarının anomali tespiti konusunda yüksek yanlış pozitif oranına sahip oldukları da bilinmektedir[31].

Yapay Sinir Ağları: Yapay sinir ağı, insan beynindeki nöron ve sinapsların çalışma mekanizmaları ve genel yapılarından ilham alınarak oluşturulmuş bir makine öğrenmesi yöntemidir[32]. YSA'lar durum değişimlerine uyum sağlayabilmelerine olanak tanıyan esnek bir yapıya sahiptirler[27]. Yapay sinir ağı yöntemleri bu avantajları ve ek olarak

oldukça başarılı bir makine öğrenmesi yaklaşımı olan derin öğrenmenin de alt yapısını oluşturmaları sebebiyle anomali tespitinde yaygın olarak kullanılmaktadırlar[33],[21] .

Bulanık Mantık Teknikleri: Bulanık mantık teknikleri, belirli kararlara varmak için klasik mantık yöntemlerindeki kesinlik yerine için yakınsama mantığı ile çalışmaktadırlar. Bulanık mantık tabanlı anomali tespitinde, belirli bir aralıkta normal veriye yakınsayan veriler normal, diğer veriler ise anormal olarak kabul edilmektedir. Bulanık mantık tekniklerinin, port taraması ve probe saldırılarının tespiti konusunda başarılı oldukları kanıtlanmıştır. Buna karşın bulanık mantık yöntemleri, yüksek çalışma karmaşıklığına sahiptirler[27].

Genetik Algoritmalar: Genetik algoritmalar evrimsel algoritmaların bir alt koludur. Evrimsel algoritmalar evrim konusundaki mutasyon, seçilim ve gen oluşumu gibi konulardan ilham alınarak tasarlanmıştır[27]. Genetik algoritmalar ile geçmiş kullanıcı aktiviteleri kullanılarak kullanıcı profilinden saldırı tespiti[34] ve süper kullanıcı yetkisine izinsiz erişimin tespitine yönelik çalışmalar yapılmıştır[35].

Destek Vektör Makinesi: İlk adımda giriş vektörünü çok boyutlu özellik uzayıyla ifade edecek şekilde dönüştürür. Sonrasında verileri optimum hiper düzlem ile ikiye ayırır. DVM iki sınıflı bir sınıflandırıcı olması nedeniyle anomali tespiti probleminin çözümüne çok uygundur. Bu yüzden DVM bir sınıflandırıcı olarak[36],[37],[38] anomali tespitinde bir çok çalışmada kullanılmıştır[1].

Karar Ağaçları: Ardışık karar verme mekanizmasına sahip bir ağaç yapısı ile sınıflandırma yapan karar ağaçlarında her bitiş düğümü bir kategoriyi temsil etmektedir. Genel mantık olarak ağaç içinde bir önceki adımda alınan karar, bir sonraki adımda alınacak olan kararı etkiler ve bu şekilde sınıflandırma yapılır[1]. Karar ağaçları protokol analizine dayalı anomali tespiti amacıyla kullanıldıklarında diğer sınıflandırıcılarda olduğu gibi bilinmeyen saldırıların tespiti konusunda başarısız oldukları görülmüştür[39].

K-En Yakın Komşu algoritması: K-en yakın komşu yöntemi çevrimiçi öğrenme yöntemi ile çalışır, modelin önceden gözetimli veya gözetimsiz olarak eğitilmesine gerek yoktur. K-en yakın komşu algoritması parametresiz olarak sınıflandırma yapabilen en temel makine öğrenmesi yöntemlerinden biridir. Seçilen örneğin en yakın k komşusu

çoğunluk olarak hangi sınıfa aitse seçilen örnek de o sınıfın bir elemanı olarak kabul edilir[1]. Bu yöntem anomali tespiti amacıyla tekil[40] ve hibrit olarak çeşitli çalışmalarda kullanılmıştır [18].

b) Uç Değer Tespiti ve Kümeleme Tabanlı Anomali Tespiti

Kümeleme, birbirine benzer verileri aynı grup altında toplama işlemidir. Grup içindeki veriler özellikleri bakımından birbirlerine grup dışındaki verilere kıyasla çok daha yakındırlar. Bu sebeple verilerin özelliklerine göre yakınlıklarının saptanması, kümeleme ve uç değer tespiti yöntemleri için büyük önem taşımaktadır Kümeleme ve uç değer tespiti konusunda en çok kullanılan mesafe ölçüm metrikleri; Öklid, manhattan, mahalanobis ve divergence yöntemleridir.

Kümeleme Yöntemi: Kümeleme yönteminde normal ve anormal verileri içeren bir eğitim kümesi gözetimsiz olarak kümelenir ve bunun sonucunda normal ve anormal verilere ait kümeler oluşturulur. Daha sonra, test edilecek örnek veri hangi kümenin merkezine daha yakın ise o kümenin elemanı olarak etiketlenir(Şekil 2. 2). Bu yöntemde genellikle eğitim kümesinde normal verilerin sayısı çok, anormal verilerin sayısı ise normallere kıyasla daha az olmaktadır. Bu sayede büyük olan kümenin normal olduğu anlaşılmaktadır[3].

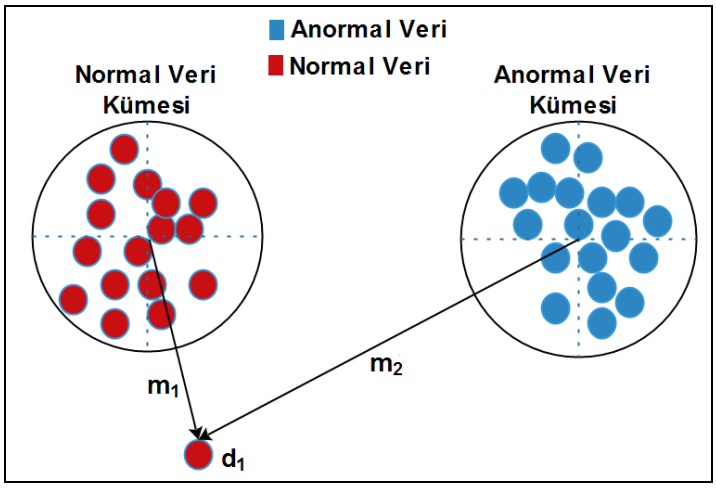
Kümeleme yöntemi, standart kümeleme ve ortaklaşa kümeleme olarak iki alt başlık altında incelenmektedir. Standart kümeleme ile ortaklaşa kümeleme arasındaki temel fark standart kümeleme yöntemleri, sadece veri kümesindeki satırları dikkate alarak kümeleme yaparken, ortaklaşa kümeleme yöntemleri hem satırları hem de sütunları dikkate alarak kümeleme yapar.

Kümeleme yöntemlerinden; k-ortalama, geliştirilmiş k-ortalama, k-medoids, Em kümeleme anomali tespitinde yaygın olarak kullanılmaktadır[3].

Kümeleme yöntemleri, “k” değerinin bilindiği anomali tespit problemlerinde kolaylıkla gruplara ayırma işlemini yapabilmektedirler. Ek olarak, kümeleme yöntemlerinin sınıflandırma yöntemlerine göre hesaplama maliyetlerinin daha düşük olması sebebiyle bu yöntemler, sınıflandırma yöntemlerine kıyasla çok daha kısa sürede stabil anomali tespit sonuçları üretebilirler. Ancak kümeleme yöntemlerinde uygun mesafe ölçüm

yönteminin belirlenememesi sistemin başarısını direk olarak düşürmektedir. Bunun yanı sıra, sınıflandırıcı yöntemlerine kıyasla model profilinin güncelleme işlemleri çok uzun zaman almaktadır[3].

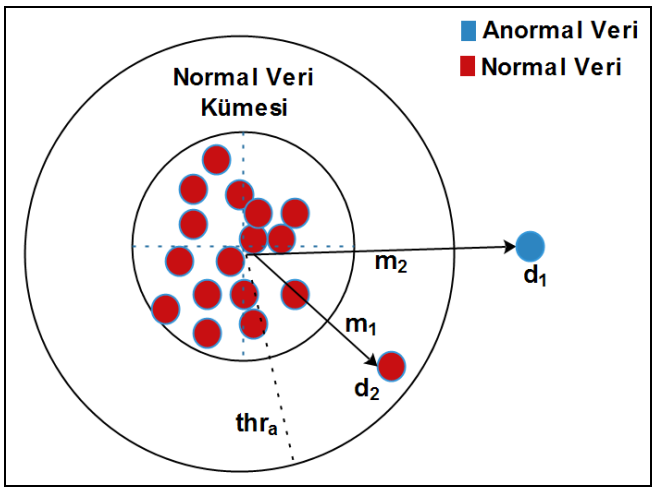
Kümeleme tabanlı anomali tespiti konusunda; aday mekanizmasına dayalı iki aşamalı kümeleme[41], coğrafi bölge bilgisine dayalı bilgisayar solucanı (computer worm) tespiti [42] gibi çalışmalar yapılmıştır.



Şekil 2. 2 Kümeleme tabanlı anomali tespiti

Uç Değer Tespiti: Ağ anomalilerinin tespiti konusunda yaygın bir şekilde kullanılan uç değer tespit yöntemleri[21], anomali tespitine benzer şekilde bir grup veri içerisinde verinin geri kalanından çok farklı örüntüye sahip verilerin aranması işlemidir[4].Uç değer tespitinde sadece normal veriler kullanılarak yarı-gözetimli bir şekilde eğitilen model, normal verilerin bir kümesini oluşturur (veya normal veriye dayalı bir sınıflandırıcı oluşuturulur). Sonrasında verilen test örneği, merkez noktası normal veri kümesinin merkezi olacak şekilde thr_a uzunluğunda yarı çapa sahip bir dairenin dışında kalıyorsa anomali olarak etiketlenir (Şekil 2. 3). Burada thr_a yarıçapındaki daire anomali eşik değerini temsil etmektedir.

Uç değer tespit yöntemleri, küçük veri kümeleri için çok başarılı sonuçlar vermektedirler. Bu yöntemler, veri kümesi içerisinde izole olmuş anomalilerin dahi tespit edilebilmesine olanak tanımaktadır. Ancak, hem kümeleme (veya sınıflandırıcı eğitimi) hem de uç değer hesabını içinde barındıran bu yöntemin zaman karmaşıklığı diğer anomali tespit yöntemlerine göre genellikle daha fazla olmaktadır. Ayrıca bu yöntemlerin başarısı doğru eşik değerinin belirlenmesine direk olarak bağımlıdır[4].

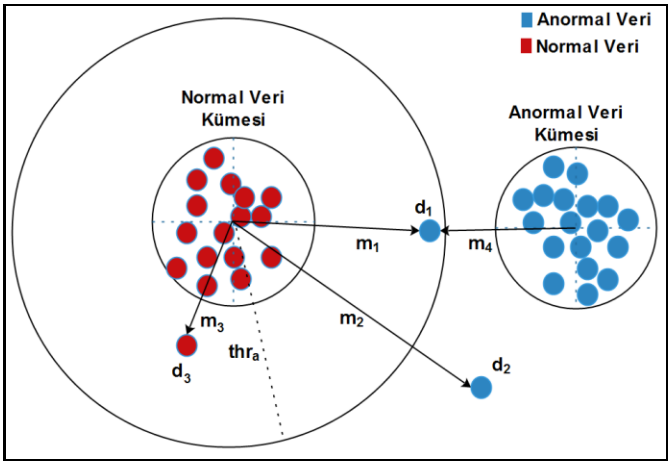


Şekil 2. 3 Uç değer tespiti tabanlı anomali tespiti

Kümeleme – Uç Değer Hibrit Yöntemi: Bu yöntemde model, normal ve anormal verileri içeren bir eğitim kümesi ile kümeleme yönteminde olduğu gibi eğitilir. Sonrasında uç değer tespiti yöntemdeki gibi bir anomali eşik değeri belirlenir (thr_a çapında merkezi normal veri kümesinin merkezi olacak şekilde bir daire) ve sonrasında verilen bir test örneği için Şekil 2. 4'te gösterildiği üzere ;

1. Anormal grubun merkez düğümüne, normal grubun merkez düğümüne olduğundan daha yakınsa anormal (öncelik değeri=1)
2. Normal eşik değerini temsil eden dairenin içinde kalmıyorsa anormal(öncelik değeri=2)
3. Geriye kalan durumlarda normal

Olarak etiketleme yapılır[43].



Şekil 2. 4 Kümeleme ve uç değer tespiti hibrit yöntemi

c) Hibrit ve Birleştirilmiş Yöntemler (Hybrid and Ensemble Methods)

Birleştirilmiş (ensemble) yöntemler, iki ya da daha fazla sınıflandırıcının bir araya gelerek oluşturdukları sınıflandırıcı grubunun elde ettiği başarımın bu grubu oluşturan sınıflandırıcıların her birinin bireysel başarımından daha yüksek olmasını hedefleyen bir makine öğrenmesi yaklaşımıdır[1]. Sınıflandırıcılar, test verisini ayrı ayrı değerlendirirler ancak son aşamada bu kararlar birleştirilerek ortak bir sonuca ulaşılır. Burada kararları birleştirmek için; bagging, boosting ve yığın genelleştirme yöntemi olmak üzere üç farklı yaklaşım bulunmaktadır [44], [1].

Birleştirilmiş yöntemler sayesinde zayıf sınıflandırıcıların kombinasyonları yapılarak başarılı sonuçlar elde edilebilir. Birleştirilmiş yöntemlerin takip edilen verideki değişime tekil yöntemlere göre daha çabuk adapte olabildikleri ve tekil yöntemlere göre daha başarılı sonuçlar verdikleri tespit edilmiştir[44]. Bu gibi kesin avantajlarına karşın birleştirilmiş yöntemler birden çok sınıflandırıcıdan sonuç aldıkları için genel olarak çalışma hızları tekil yöntemlere göre daha yavaş kalmaktadır. Ek olarak, birleştirilmiş yöntemlerde sınıflandırıcı seçimi için kesin olarak izlenmesi gereken bir yol olmadığından genellikle greedy yaklaşım kullanılmaktadır.

Hibrit yöntemler, anomali tespiti konusunda farklı yöntemlerin farklı avantajlarından faydalanabilmek için yaygın bir şekilde kullanılmaktadır. SVM - YSA hibrit yöntemi[24], DBN - SVM hibrit yöntemi[37], Random Tree - Nb Tree hibrit yöntemi[20] gibi bir çok hibrit yöntem anomali tespiti amacı ile kullanılıp yüksek başarımlar elde edilmiştir.

İstatistiksel Model Tabanlı Anomali Tespiti

Anomali tespiti amacıyla kullanılan istatistiksel yöntemler, normal verinin istatistiki bir profilini oluştururlar ve bu profile uymayan verileri anormal olarak adlandırırlar[3]. Bu yöntemler parametrik ve parametrik olmayanlar olarak iki ayrı kategori altında incelenmektedir. Parametrik yöntemler, veri içeresindeki dağılımları tespit edebilirken parametrik olmayan yöntemler verinin iç dağılımı hakkında bilgi sahibi olmazlar[18]

İstatistiksel yöntemler uygulanacakları ağın karakteristiği hakkında önceden bilgi sahibi olmak zorunda değildirler[27]. Ağa uygulandıkları andan itibaren o ağın normal yapısını öğrenip ona göre anomali tespiti yaparlar.

İstatistiksel yöntemler, normal profile belirli bir eşik değeri üstünde uzak olan verileri anormal olarak tespit etmektedirler. Bu yüzden istatistiksel yöntemlerde, uzmanlık ve zaman gerektiren bu parametrelerin ve metriklerin belirlenmesi modelin başarımı açısından kritik önem taşımaktadır. Ayrıca bu yöntemler izledikleri veri üzerinden sürekli öğrenen bir yapıda oldukları için saldırganlar tarafından eğitilebilirler[27].

İstatistiksel yöntemlerden, expectation maximization ile ağ içerisindeki solucanların tespitine yönelik bir çalışma yapılmıştır[45]. Ayrıca, PCA yöntemi; veri sıkıştırma, anlamlı özelliklerin seçilmesi ve hesaplama maliyetini düşürmesi gibi avantajlarından dolayı istatistiksel yöntem tabanlı anomali tespitinde kullanılmıştır [23].

2.2.2.4 Anomali Tespiti Değerlendirme Veri Kümeleri

Ağ anomalilerinin tespiti, 20 yılı aşkın bir süredir devam eden aktif bir araştırma konusu olduğundan bu problemin çözümüne ilişkin önerilen modellerin performanslarının değerlendirilmesi adına birçok farklı değerlendirme veri kümesi oluşturulmuştur. Yıllar içinde değişen işletim sistemleri konseptleri, ağ teknolojilerinde yaşanan gelişmeler ve hızlı bir şekilde artmakta olan ağ saldırıları, değerlendirme veri kümelerinin kendilerini sürekli güncellemelerini zorunlu kılmıştır. Günümüzde yaygın olarak kullanılan saldırı tespiti veri kümelerinden bazıları şunlardır[3],[18];

CAIDA (2011): Basit olarak yakalanmış ağ paketlerini içermektedir. Ancak veriler etiketli değildir ve çoklu saldırı senaryoları bulunmamaktadır[12].

DEFCON (2011): Yaygın olarak kullanılan bir diğer değerlendirme veri kümesidir. Bu veri kümesi, hacker yarışmaları sırasında oluşturulan trafikten meydana gelmektedir. Bu yüzden geneli saldırı verisinden oluşturulmuştur ve gerçek ağ verisinden farklı veriler içeren bir veri kümesidir[13].

ISCX UNB (2012): Gerçek ağ paketleri analiz edilerek gerçek; *http, smtp, ssh, imap, pop3* ve *ftp* trafiği oluşturacak profiller yaratılmıştır. Ek olarak, veri kümesi çeşitli çok katmanlı atak senaryolarına ilişkin verileri de içermektedir[17].

ADFA-LD12 (2012): Günümüzün en çok kullanılan işletim sistemlerinden biri olan Ubuntu'nun 11.04 sürümünü kullanan sistemlerin bulunduğu bir ağdan toplanan paketlerden oluşturulmuştur ve en güncel saldırı tiplerini içermektedir. KDDCUP'99 veri

kümesinin, günümüzde çok fazla rağbet görmeyen Solaris işletim sistemini kullanan simülasyon ortamlarında oluşturulmuş olması ve günümüzde KDDCUP'99'un oluşturulduğu tarihe kıyasla bir çok yeni saldırı türünün ortaya çıkmış olması, ADFA-LD12 veri kümesine olan ilgiliyi her geçen gün arttırmaktadır[46].

KDDCUP'99 (1999): KDDCUP'99 veri kümesi hala en yaygın olarak kullanılan saldırı tespiti performans değerlendirme kümesidir. Bu veri kümesi DARPA98 IDS değerlendirme programı kapsamında yakalanan paketlerden oluşturulmuştur. Veri kümesi eğitim veri kümesi ve test veri kümesi olarak iki ayrı kümeden oluşmaktadır. Eğitim kümesinin içeresinde 4.900.000 veri bulunmaktadır. Her verinin 41 özelliği vardır. Test veri kümesi ise 300,000 veriden oluşmaktadır ve 14 ü sadece test kümesi mahsus olmak üzere 24 farklı atak tipini ve normal verileri içermektedir. Bu saldırılar; *DOS,R2L,Probe* ve *U2R* saldırı grupları altında birleştirilerek sınıflandırılabilecekleri gibi ayrı ayrı da sınıflandırılabilirler[47],[14].

NSL-KDD (2009): Bu tez kapsamında oluşturulan modellerin performanslarının değerlendirilmesi adına KDDCUP'99 veri kümesinin geliştirilmiş hali olan NSL-KDD veri kümesi kullanıldığından NSL-KDD veri kümesine ilişkin detaylara daha geniş yer ayırılmıştır.

NSL-KDD veri kümesi KDDCUP'99 veri kümesinin hatalarından arındırılmış ve geliştirilmiş halidir. Saldırı tespit yöntemlerinin başarılarının kıyaslanmasında kullanılan KDDCUP'99 veri kümesinin çok fazla tekrarlı veri içermesinin, algoritmaların başarı oranlarını etkilediği ve normale göre çok daha yüksek sonuçlar alınmasına sebep olduğu tespit edilmiştir[16]. Bunun sonucunda KDDCUP'99 veri kümesinden NSL-KDD veri kümesi oluşturulmuştur. NSL-KDD veri kümesindeki her veri 41 özelliğe sahiptir. Bu özelliklerden üç tanesi kategorik, 38 tanesi ise nümeriktir.

NSL-KDD veri kümesi KDDTrain+ ve KDDTrain+'nın %20'lik versiyonu olan KDDTrain+_%20 olmak üzere iki adet eğitim kümesi ile KDDTest+ ve 21 sınıflandırıcı tarafından başarılı olarak sınıflandırılmış olan örneklerden temizlenmiş KDDTest-21 isminde iki test kümesi içermektedir. KDDCUP'99 veri kümesindeki tüm tekrarlı veriler silindikten sonra veri kümesi içerisinde çeşitli zorluk seviyeleri oluşturulması adına tekrarlardan arındırılmış KDD eğitim kümesi 3 parçaya bölünmüş ve her parça üzerinde

7 ayrı sınıflandırıcı eğitilerek (21 sınıflandırıcı) eğitim ve test kümeleri sınıflandırılmıştır. Test ve eğitim kümesindeki her bir örnek için "başarılı sınıflandırma sayısı" adında bir değişken, sıfır değeri ile ilklendirilmiş ve her bir başarılı sınıflandırma sonucunda bu sayaç bir arttırılmıştır. Bu analiz sonucunda eğitim kümesinin %98'i ve test kümesinin %86'sı 21 adet sınıflandırıcının tamamı tarafından doğru sınıflandırılmıştır. Bu da KDDCUP'99 üzerinde uygulanan en basit makine öğrenmesi algoritmalarının bile eğitim kümesinde minimum %98 ve test kümesinde de minimum %86'ya yakın başarı göstermesini açıklamaktadır[16]. Analiz sonucunda elde edilen "başarılı sınıflandırma sayısı" değişkenlerinin değerine göre veri kümesi; *0-5 puan arası*, *6-10 puan arası, 11-15 puan arası, 16-20 puan arası* ve *21 puan* alan örneklere göre gruplandırılmış ve her gruptan belirli yüzdede veri seçilerek KDDTrain+ ve KDDTest+ veri kümeleri oluşturulmuştur. Bunun yanı sıra 21 sınıflandırıcı tarafında da başarılı sınıflandırılmış olan örneklerin temizlenmiş olduğu KDDTest-21 adında daha zor bir test kümesi de oluşturulmuştur.

Bu çalışmada önerilen modellerin eğitiminde kullanılan KDDTrain+_%20, normal trafiği ve 21 ayrı saldırı trafiğini içermektedir. Saldırı tipleri; *DOS, probing user-to-root ve root-to-local* saldırı grupları altında toplanarak veya tamamı anormal sınıfına ait olarak kabul edilerek değerlendirilebilir[16]. KDDTest+, KDDTrain+%20'daki 19 saldırı tipine ek olarak 18 yeni saldırı tipini içermektedir. Bu test veri kümesi, kullanılan yöntemlerin sıfırıncı gün saldırılarındaki başarısı hakkında fikir vermesi açısından önemlidir[16]. Bu çalışmada Çizelge 2. 1'de gösterilen şekilde KDDTrain+%20 ve KDDTest+ içerisindeki saldırı verileri anormal etiketi altında birleştirilerek iki sınıflı anomali tespiti yapılmıştır.

Çizelge 2. 1 KDDTrain+_%20 ve KDDTest+'nın sınıf bazında veri dağılımları

		KDDTrain+%20	KDDTest+
Normal		13449	9711
ANORMAL	DOS	9233	7458
	U2R	11	67
	R2L	347	2887
	PROBE	2289	2421

BÖLÜM 3

DERİN ÖĞRENME TABANLI AĞ ANOMALİ TESPİT MODELLERİ

Bu bölümde, derin öğrenme yöntemlerinin temelini oluşturan; perceptron ve lojistik sınıflandırıcı yapısı, çok katmanlı perceptron yapısı (ÇKP), otomatik kodlayıcı yöntemi ve yapay sinir ağlarının genelleştirilmesi ve başarımın arttırılmasına yönelik yöntemler açıklanmıştır. Sonrasında bu temel yöntemleri esas alan, derin yapay sinir ağları (DSA) ve önerilen uç değer tespitine dayalı otomatik kodlayıcı tabanlı anomali tespit modellerinin çalışma prensipleri tariflenmiştir.

3.1 Derin Öğrenme

Bir makine öğrenmesi yöntemi olan yapay sinir ağlarının temelleri, 1950'lerde perceptron algoritmasının geliştirilmesi ile atılmıştır. Doğrusal aktivasyon fonksiyonu kullanan perceptron algoritmasının karmaşık fonksiyonlara yakınsamada başarısız olması nedeniyle 1970'lerde bu algoritmanın doğrusal olmayan aktivasyon fonksiyonunu kullanan sürümü tasarlanmıştır ve daha sonra, 1980'lerde başarımın arttırılabilmesi adına bu doğrusal olmayan perceptron katmanları birbirlerine eklenerek çok katmanlı perceptron modeli oluşturulmuştur. Ek olarak, bu çok katmanlı yapının eğitimi için türeve dayalı bir yöntem olan geri yayılım (back propagation) algoritması geliştirilerek YSA'ların karmaşık fonksiyonlara yakınsama başarımı arttırılmaya çalışılmıştır[48]. Ancak, etiketli veri kümelerinin eksikliği ve hesaplamalar için gerekli güçlü donanımların bulunmaması nedeniyle yapay sinir ağları 2000'li yılların sonlarına kadar çok fazla ilgi görmemiştir. 2000'li yıllarda, paralel programlama ve hızlı grafik işlemcilerin üretilmesi gibi hesaplama gücünde yaşanan büyük gelişmeler sayesinde yapay sinir ağları kavramı tekrar ilgi görmüştür. Hızla gelişen grafik işlemci

teknolojisi sayesinde bugün, derin öğrenme algoritmaları grafik işlemciler üzerinde genel amaçlı işlemcilere kıyasla 10 ile 20 kat arasında daha hızlı bir şekilde eğitilebilmektedir[49]. Buna ek olarak, bugün derin öğrenme yöntemlerinin temelini oluşturan birçok yöntem 1990'lı yıllarda geliştirilen yöntemlerden çok da farklı değildir[47]. Burada vurgulanması gereken nokta, yapay sinir ağlarındaki teorik gelişmelerden ziyade bugün elimizde bulunan donanımların sahip oldukları işlem yeteneği sayesinde eğitimin hızlandırılması, derin öğrenme algoritmaları ve yapay sinir ağları konusunun kullanım alanının yaygınlaşmasını sağlamıştır[49].

YSA'larda başarımın en yüksek seviyeye ulaşabilmesi için yapay sinir ağında kullanılacak olan özelliklerin çok dikkatli bir şekilde seçilmesi gerekmektedir. Çünkü özellik boyutu doğrusal olarak arttığında öğrenme karmaşıklığı üstel olarak artmaktadır. Bunun sonucunda, hem eğitim zamanını uzamakta hem de YSA'nın başarımı negatif yönde etkilenmektedir. Bu konu literatürde boyutluluğun laneti (curse of dimensionality) olarak adlandırılan önemli bir problemdir[50]. Bu problemin çözümü için modelin eğitildiği veri kümesi içerisindeki bağlantıların ve ilişkilerin uzman kişiler tarafından çıkartılarak özellik boyutunun el ile daraltılması gerekmektedir[49]. Ancak günümüzde ciddi şekilde artmış olan veri kümesi boyutları ve özellik uzay genişlikleri sebebiyle bu işlemlerin el ile etkin ve hızlı bir şekilde yapılması uygulanabilir değildir. Derin öğrenme yöntemleri, veri kümesi içerisindeki bağlantıları otomatik olarak çıkarması ve özellik boyutu daraltımı yaparak sınıflandırma modelinin başarımını arttırması açısından büyük önem taşımaktadır. Derin öğrenme modelleri, verinin temsilinin çok katmanlı özetleme sayesinde öğrenilmesini sağlayan bir yapıya sahiptirler[49].

Derin öğrenme modelleri ileri beslemeli yapay sinir ağı (feed forward neural network) yapısı üzerine kurulmuştur. Bu modeller; bir giriş katmanından, basit bir yapıya sahip ve doğrusal olmayan şekilde dönüşüm yapabilen gizli katman/katmanlardan ve bir tane doğrusal çıkış katmanından oluşmaktadır. Derin öğrenme yöntemleri, temsil öğrenmesi (representation learning) olarak da adlandırılmaktadır[51]. Çok katmanlı bir yapıya sahip olan bu modeller, basit ama doğrusal olmayan modüller yardımıyla her katmanda veri içindeki temsili daha yüksek ve soyut olarak ifade ederler. Bu şekilde çalışan katmanların sonucunda çok karmaşık fonksiyonlar dahi öğrenilebilir[25]. Sonrasında sınıflandırma aşamasında verinin yüksek katmanlarındaki en soyut temsili kullanılarak

veriye ait en önemli özellikler vurgulanırken gereksiz özelliklerinde üzerine yüklenen anlam da azaltılmış olur ve bu sayede derin öğrenme yöntemleri, çok yüksek derecede doğrusal olmayan fonksiyonlara bile yakınsayarak başarılı sonuçlar vermektedirler[47].

Derin öğrenme yapısındaki derin kelimesi farklı uzmanlar tarafından; birden çok gizli katmana sahip olan yapay sinir ağları, etiketsiz verilerin etkin bir biçimde kullanılmasına olanak tanıyan modeller gibi farklı kavramları temsil ediyor olsa da en kapsayıcı şekilde bakıldığında derin öğrenme kavramının, çeşitli makine öğrenmesi yöntemlerini kullanarak verinin soyutlaştırılmış temsilini çıkaran ve bu sayede önemli özelliklerin otomatik olarak öğrenilmesini sağlayan modelleri temsil ettiği söylenebilir[47] ,[11].

1990'ların başlarında geri yayılım yöntemi ile eğitilen yapay sinir ağları makine öğrenmesi topluluğu tarafından tercih edilmemekteydi. Çünkü çok boyutlu özellik uzayına sahip veriler için etkili özelliklerin seçiminin el ile yapılması gerekiyordu ve meyilli azaltım(gradient descent) yöntemi lokal minimum değerlerine takılarak düşük başarım göstermekteydi. 2006 yılında Canadian Institute for Advanced Research (CIFAR) kurumunda yapay sinir ağları üzerine çalışan bir grup tarafından bu iki problemi gidererek yapay sinir ağı eğitimini iyileştirmek için gözetimsiz olarak özellik tespit eden iki aşamalı bir model oluşturulmuştur. Tasarlanan bu modelin ilk aşaması olan ön-eğitimde (pre-training), model her katmanındaki aktivasyon fonksiyonları yardımıyla verilen girdinin tekrar oluşturulmasına ve özellik boyutunun daraltılmasına yönelik eğitilmiştir ve ön-eğitim aşamasından elde edilen ağırlıklarla yapay sinir ağındaki bağlantılar ilklendirilmiştir. Sonrasında ikinci aşama olan ince-ayar (fine-tunning) bölümünde yapay sinir ağı gözetimli olarak tekrar eğitilerek tüm ağırlıklar tekrar güncellenmiş ve model optimum ağırlıklara ulaşacak şekilde eğitilmiştir. Bu yöntem derin yapay sinir ağı olarak adlandırılmaktadır. Bu çalışmayla el yazısı karakterlerinin tanınması konusunda yüksek başarım elde edilmiştir[49]. Bu yönteme benzer şekilde ön-eğitim ve ince-ayar aşamalarından oluşan ancak geri yayılım yönteminin gereksinimini ortadan kaldıran sınırlandırılmış Boltzman makineleri (RBM) tabanlı derin inanç ağları yöntemi oluşturulmuş ve MNSIT veri kümesi üzerinde mevcut yöntemleri geride bırakan sonuçlar elde etmiştir[52]. Derin yapay sinir ağları ve derin inanç ağları yöntemlerinin yanı sıra özellikle görüntü işleme ve ses işleme konusunda özellik

boyutunun azaltılarak başarılı sınıflandırma yapılması için tasarlanmış olan konvolüsyonel yapay sinir ağları (convolutional neural networks) (CNN), bu alanlarda diğer tüm makine öğrenmesi yöntemlerinden daha yüksek sonuçlar vermektedir[50]. Ek olarak, doğal dil işleme gibi bir biriyle ilişkili veri zincirlerinin ve zaman serilerinin sınıflandırılması amacıyla tekrarlı yapay sinir ağlarının (reccurent neural networks) bir çeşidi olan long short-term memory (LSTM) yöntemi tasarlanmış ve bu yöntemin kullanımından yüksek başarım elde edilmiştir[49].

Geçmişi 1950'li yıllara kadar dayanan derin öğrenme yöntemleri, geleneksel makine öğrenmesi yöntemleriyle yıllardır çözülmeye çalışılan; görüntü tanıma, ses tanıma, ilaç içeresindeki moleküllerin aktivitelerinin tespiti, parçacık hızlandırıcı verilerinin analizi, beyin ağlarının tekrar oluşturulması, DNA mutasyon tespiti, doğal dil işleme alanında; konu sınıflandırma, duygu analizi, soru cevap sistemleri ve otomatik çeviri programları gibi birçok problemin çözümünde geleneksel yöntemlere kıyasla daha yüksek başarımlar elde edilmesini sağlamıştır[49]

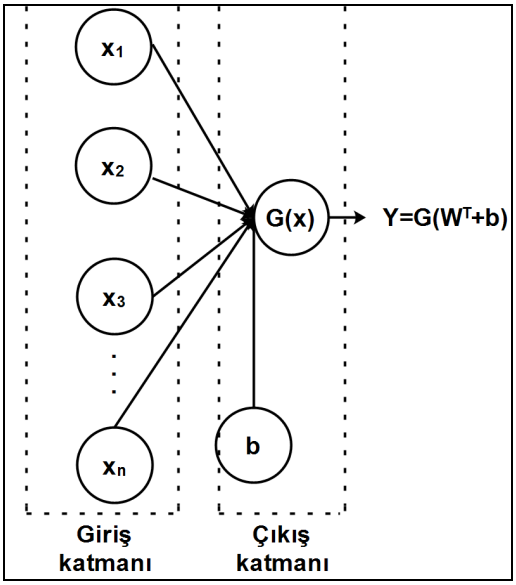
3.2 Yapay Sinir Ağları

Bu bölümde derin öğrenme modellerinin temelini oluşturan yapay sinir ağlarının çalışma prensipleri, basitten modellerden başlanarak daha karmaşık yapılara doğru adım adım tariflenmiştir. Öncelikle yapay sinir ağının temelini oluşturan perceptron yapısı, lojistik sınıflandırıcı, çok katmanlı perceptron yapısı, otomatik kodlayıcı modelleri ve yapay sinir ağı eğitiminin iyileştirilmesine ilişkin yöntemler açıklanmıştır.

3.2.1 Perceptron

Perceptron, lineer olarak sınıflandırma yapan ve çok katmanlı yapay sinir ağlarının temelini oluşturan yapıdır. (3.1) eşitliğinde "W^T "ağırlık değerinin transpozunu, "*x*" giriş verisini, "*b*" bias değerini ve *"a"* değeri de perceptronun çıkış değerini vektörel notasyonda ifade etmektedir. Perceptron, (3.1) belirtilen karar fonksiyonu ile *"x"* verisini eğitim verisindeki sınıflardan en uygun olanına atamaya çalışır[32],[47]. Perceptronun genel yapısı Şekil 3. 1 'de gösterilmiştir.

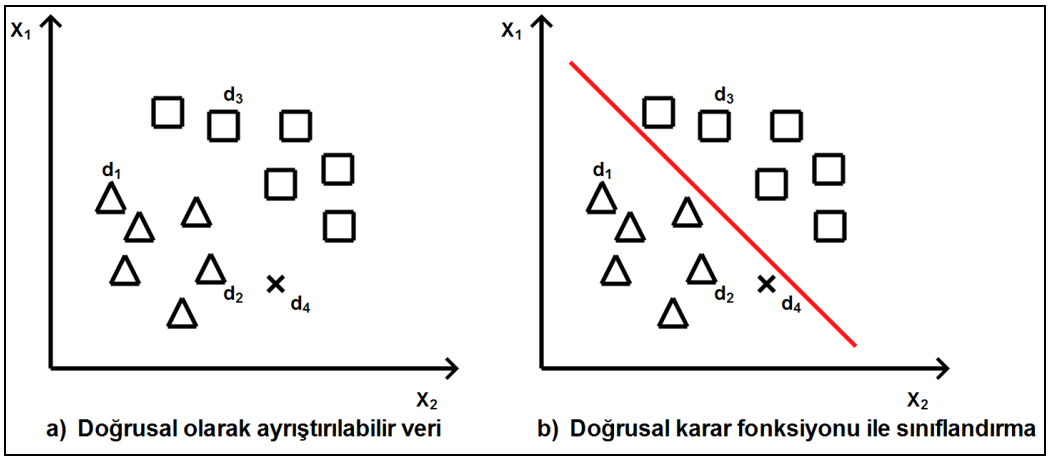
$$a = (W^T x + b) \tag{3.1}$$



Şekil 3. 1 Perceptron yapısı

(3.1) eşitliğinde *"a"* çıkış fonksiyonundan alınan değerler verilen giriş verisinin değerine bağlı olarak göre çok büyük veya çok küçük olabileceğinden bu değerin belirli bir aralığa ölçeklenmesi perceptronun eğitimi için kritik önem taşımaktadır. Bu yüzden nöronlarda çıkış değerini sabit bir aralığa ölçeklemek için (3.2) eşitliğinde gösterildiği üzere *"G"* aktivasyon fonksiyonu kullanılır. Aktivasyon fonksiyonu olarak *sigmoid, tanh ve softmax* fonksiyonları kullanılmaktadır[53].

$$Y = G(W^T x + b) \tag{3.2}$$



Şekil 3. 2 İki sınıf için veri dağılımı

Şekil 3. 2'de d_1, d_2 ve d_3 verilerinin x_1 ve x_2 özelliklerine ait değerleri ve ait oldukları sınıflar verilmiştir (üçgen ve kare şekilleri ile). Buna ek olarak d4 verisinin de x_1 ve x_2

özellikleri belirtilmiş ancak sınıfı belirtilmemiştir. Perceptronun çıkış değeri veren, Y karar fonksiyonu $d_1, d_2, d_3 \ldots d_n$ için;

$$\begin{bmatrix} Y^{(1)} = G(W_{11}x^{(1)}{}_1 + W_{21}x^{(1)}{}_2 + b) \\ Y^{(2)} = G(W_{11}x^{(2)}{}_1 + W_{21}x^{(2)}{}_2 + b) \\ Y^{(3)} = G(W_{11}x^{(3)}{}_1 + W_{21}x^{(3)}{}_2 + b) \\ . \\ . \\ . \\ . \\ Y^{(\mathrm{n})} = G(W_{11}x^{(\mathrm{n})}{}_1 + W_{21}x^{(\mathrm{n})}{}_2 + b) \end{bmatrix} \tag{3.3}$$

(3.3) eşitliğinde gösterildiği şekilde $Y^{(1)}$, $Y^{(2)}$, $Y^{(3)}$ … $Y^{(n)}$ sonuçlarını üretir. Skalar notasyonda verilmiş olan W_{ij}, L katmandaki i. nöronun L+1. katmandaki j. nöron ile arasındaki bağlantının ağırlığını temsil etmektedir. (3.4) eşitliğinde verilen eğitim verisinin her biri için karar fonksiyonunun ürettiği $Y^{(i)}$ değerleri ile eğitim verilerinin gerçek değerleri olan $y^{(i)}$ değerleri arasındaki farkların karelerinin toplamı $j(W,B)$ hata fonksiyonu olarak adlandırılır (eşitlik (3.4)). Perceptronun eğitimi bu hata fonksiyonunun ürettiği değerlerin meyilli azaltım (MA) yöntemi ile azaltılmasına yönelik yapılır[47].

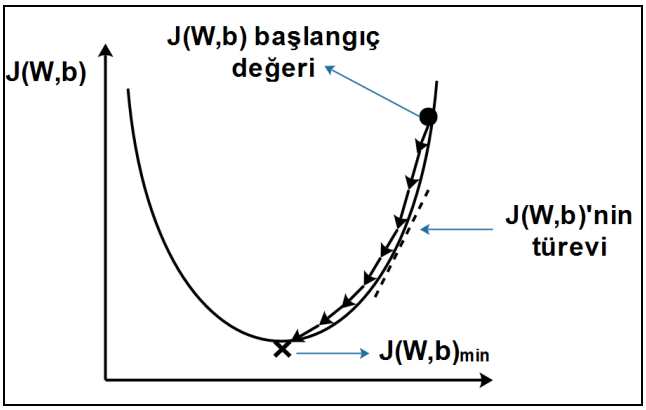
$$\begin{aligned} J(W,b) &= \left[(Y^{(1)} - y^{(1)})^2 + (Y^{(2)} - y^{(2)})^2 + (Y^{(3)} - y^{(3)})^2 \ldots\ldots ((Y^{(\mathrm{n})} - y^{(\mathrm{n})})^2 \right] \\ J(W,b) &= \sum_{i=1}^{n} (Y^{(\mathrm{i})} - y^{(\mathrm{i})})^2 \end{aligned} \tag{3.4}$$

3.2.2 Meyilli Azaltım

Meyilli azaltım (gradient descent) yönteminde, (3.5) ve (3.6) eşitliklerinde belirtildiği üzere hata fonksiyonunun W ve b parametrelerine göre türevleri olan, ΔW ve Δb değerlerinin sabit bir çarpan olan öğrenme katsayısı α ile çarpılarak W ve b değerlerinin azaltılması ve bunun sonucunda hata fonksiyonunun ürettiği değerlerin de azaltılması hedeflenmektedir. Şekil 3. 3 'te hata fonksiyonunun meyilli azaltım yöntemiyle adım adım minimum noktasına gelişi gösterilmiştir[32].

$$W_{ij} = W_{ij} - \alpha \Delta W_{ij} \tag{3.5}$$

$$b = b - \alpha \, \Delta b \tag{3.6}$$



Şekil 3. 3 Meyilli azaltım

Meyilli azaltım yöntemi verinin tamamı için $Y^{(i)}$ sonuçlarını ürettikten sonra fonksiyonunun türevini alarak güncelleme yaptığından modelin eğitimi çok uzun sürmektedir. Bu problemin çözümü amacı ile olasılıksal meyilli azaltım (OMA) yöntemi geliştirilmiştir. Olasılıksal meyilli azaltım (stochastic gradient descent) yönteminde hata fonksiyonun türevi, meyilli azaltımda olduğu gibi bir seferde tüm veri üzerinden hesaplanmak yerine her adımda sadece tek bir veri kullanılarak hesaplanır. Sonrasında *W* ve *b* değerleri bu fonksiyonun türevine göre güncellenir ve bu işleme veri kümesindeki tüm veriler için devam edilir. Bu şekilde yapılan eğitim çevrimiçi eğitim (online training) olarak adlandırılır. Olasılıksal meyilli azaltım yönteminde *ΔW* ve *Δb* değerleri zincir kuralına göre hesaplanır. Zincir kuralına göre eğer *y* fonksiyonu *x* değişkenine göre türevi alınabilen bir fonksiyon ise (3.7) eşitliğinde olduğu gibi türev alınabilir.

$$\frac{dy}{dx} = \frac{dy}{dx}\frac{dx}{dz} \tag{3.7}$$

$$\Delta W_{11} = \frac{d(G(W^T x^{(i)} + b) - y^{(i)})^2}{dW_{11}} \tag{3.8}$$

$$\Delta W_{11} = 2(G(W^T x^{(i)} + b) - y^{(i)}) \frac{d(G(W^T x^{(i)} + b))}{d\,W} \tag{3.9}$$

(3.8) ve (3.9)'da yapılan türev alma işlemlerinden sonra (3.9)'da sağ kısımda kalan türev işlemi ve zincir kuralı ve (3.10)'da tariflenen kurala göre (3.11)'deki formuna getirilir.

$$\frac{dy}{dx} = \left[1 - y(\mathrm{x})\right] y(\mathrm{x}) \tag{3.10}$$

$$\begin{aligned} \frac{d(G(W^T x^{(\mathrm{i})} + b))}{dW_{11}} &= \frac{d(G(W^T x^{(\mathrm{i})} + b))}{d(W^T x^{(\mathrm{i})} + b_1)} \frac{d(W^T x^{(\mathrm{i})} + b)}{dW_{11}} \\ \frac{d(G(W^T x^{(\mathrm{i})} + b))}{dW_{11}} &= \left[1 - G(W^T x^{(\mathrm{i})} + b)\right] G(W^T x^{(\mathrm{i})} + b) \frac{d(W^T x^{(\mathrm{i})} + b)}{d\,\mathrm{W}} \end{aligned} \tag{3.11}$$

(3.12)'de *W* değeri, matris notasyonu açılarak skalar notasyon ile gösterildiğinde, kısımda W_{11}'e göre türev alınırsa (3.13)'de görüldüğü üzere sadece $x_1^{(i)}$ elde edilir.

$$\frac{d(G(W^T x^{(\mathrm{i})} + b))}{dW_{11}} = \left[1 - G(W^T x^{(\mathrm{i})} + b)\right] G(W^T x^{(\mathrm{i})} + b) \frac{d(W_{11} x^{(\mathrm{i})}{}_1 + W_{21} x^{(\mathrm{i})}{}_2 + b)}{dW_{11}} \tag{3.12}$$

$$\Delta W_{11} = \frac{d(G(W^T x^{(\mathrm{i})} + b))}{dW_{11}} = \left[1 - G(W^T x^{(\mathrm{i})} + b)\right] G(W^T x^{(\mathrm{i})} + b)\, \mathrm{x}_1^{(\mathrm{i})} \tag{3.13}$$

Sonuç olarak ΔW_{11} (3.13) gösterildiği şekilde bulunur. Buna ek olarak diğer ΔW_{jk} ve Δb değerleri de benzer olarak (3.14)'de gösterildiği şekilde bulunur.

$$\begin{aligned} \Delta W_{jk} &= \frac{d(G(W^T x^{(\mathrm{i})} + b))}{dW_{jk}} = \left[1 - G(W^T x^{(\mathrm{i})} + b)\right] G(W^T x^{(\mathrm{i})} + b)\, \mathrm{x}_j^{(\mathrm{i})} \\ \Delta b &= \frac{d(G(W^T x^{(\mathrm{i})} + b))}{db} = \left[1 - G(W^T x^{(\mathrm{i})} + b)\right] G(W^T x^{(\mathrm{i})} + b) \end{aligned} \tag{3.14}$$

Tanım 3.1 Olasılıksal meyilli azaltım algoritması**:**

1. W ve b değerlerini rastgele ata.
2. Rastgele bir $x^{(i)}$ verisi için hata fonskiyonunu hesapla.
3. (3.14) eşitliğinde belirtildiği üzere ΔW_{jk} ve Δb değerlerini hesaplayarak (3.5) ve (3.6) eşitliklerine göre *W* ve *b* değerlerini güncelle.
4. Tüm örnekler tamamlanana kadar 2. adımdan devam et.

Çevrimiçi eğitim yaklaşımı ile olasılıksal meyilli azaltım yönteminin uygulanması modelin kapasitesine göre en başarılı yakınsama sonuçlarını verecektir ancak bu yöntemde işlenmek üzere tek tek veri aktarımı yapılması grafik işlemciler gibi matris – matris çarpımı için özelleşmiş donanımların bu avantajının kullanılamamasına sebep olmaktadır. Bu yüzden olasılıksal meyilli azaltım yönteminin bir alt kolu olan küçük-grup (mini-batch) olasılıksal meyilli azaltım (KGOMA) yöntemi geliştirilmiştir. Bu

yöntemde veriler grafik işlemciye birer birer beslenmek yerine 10'luk, 20'lik veya 300'lük gibi gruplar halinde beslenerek ve eğitim tur sayısı da grup elaman sayısı ile doğru orantılı olacak şekilde arttırılarak meyilli azaltım yapılmaktadır. Örneğin, bir model için çevrimiçi öğrenme ile 10 turluk eğitim sonucunda yakalanan hata oranını yaklaşık olarak yakalamak için, grup eleman sayısı 10 olarak belirlenmiş KGOMA yöntemi ile 100 turluk bir eğitim gerekmektedir. Bu yöntem, GPU'ların paralel işlem gücünün tam verimle kullanılmasına olanak tanımaktadır. KGOMA ile çevrimiçi olasılıksal meyilli azaltım yöntemine kıyasla 2 ile 10 kat arası daha hızlı eğitim yapılabilmektedir[32].

Model OMA veya KGOMA yöntemi ile eğitilip hata fonksiyonu minimum değere indirildikten sonra karar fonksiyonu, eğitim verisindeki tüm veriler için gerçek değerlerine yakınsayan sonuçlar üretir.

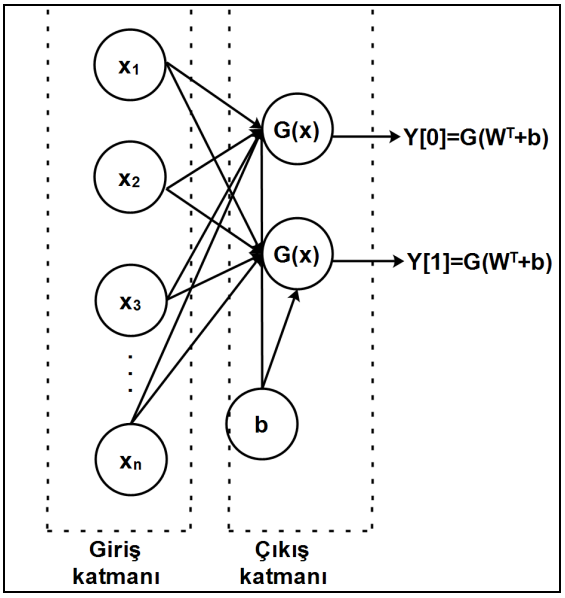
Tekil perceptron ile iki sınıflı sınıflandırma işlemi yapılırken (3.2) eşitliğinde tariflenmiş karar fonksiyonu, sigmoid olarak belirlendiğinde tüm çıkışlar [0-1] aralığına ölçeklenir ve 0,5 değerinden büyük $Y^{(i)}$ değerine sahip veriler "kare" sınıfına ait olurlarken 0.5 değerinden daha büyük çıkış değerleri ise "üçgen" sınıfına ait olurlar.

3.2.3 Softmax Aktivasyon Fonksiyonu Tabanlı Lojistik Sınıflandırıcı

Softmax aktivasyon fonksiyonu tabanlı lojistik sınıflandırıcıda tek nöronlu sınıflandırmadan farklı olarak sınıf sayısı kadar nöron bulunur ve tüm nöronların çıkışı (3.15) eşitliğinde tariflendiği gibi [0-1] aralığında bir olasılık değerine sahip olacak şekilde ölçeklenir[32].

Sırası ile (3.16),(3.17) ve (3.18) eşitliklerinde belirtildiği üzere en yüksek çıkış değerine sahip olan nöron, $x^{(i)}$'nin sınıfı olarak belirlenir. Bu tez kapsamındaki tüm gözetimli eğitime dayalı sınıflandırıcılarda softmax aktivasyon fonksiyonu tabanlı lojistik sınıflandırıcı kullanılmıştır. Şekil 3. 4'te softmax fonksiyonu kullanan lojistik sınıflandırıcıya ait görsel verilmiştir.

$$Y = softmax(W^T x + b) \tag{3.15}$$



Şekil 3. 4 Softmax lojistik sınıflandırıcı

$$P(Y=i \mid x,W,b) = softmax(W_i^T x + b) \tag{3.16}$$

$$P(Y=i \mid x,W,b) = \frac{e^{W_i^T x + b_i}}{\sum_{j=1}^{n} e^{W_j^T x + b_j}} \tag{3.17}$$

$$y_{pred} = argmax_i P(Y=i \mid x,W,b) \tag{3.18}$$

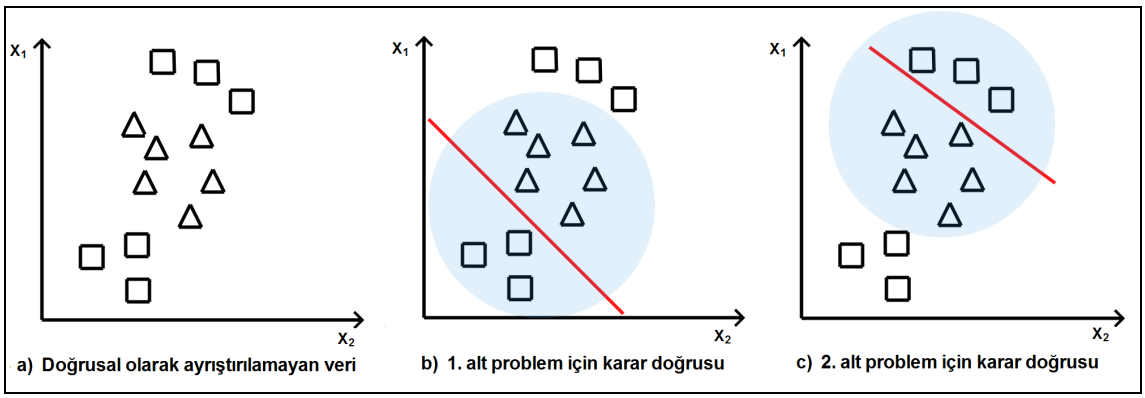
$$LL(W,b) = \sum_{i=0}^{|D|} log\left(P(Y=y^{(i)} \mid x^{(i)},W,b)\right) \tag{3.19}$$

$$J(W,b) = -LL(W,b) \tag{3.20}$$

Lojistik sınıflandırıcıların hata fonksiyonu (3.4) eşitliğindeki hataların farkının kareleri toplamı yönteminden farklıdır. (3.19) eşitliğinde belirtildiği üzere veri kümesindeki verilerin gerçekte ait oldukları sınıflara ilişkin nöronların ürettiği çıkış değerlerinin logaritmalarının toplamı log-likelihood fonksiyonu olarak adlandırılmaktadır. Bu fonksiyondan elde edilen değerlerin maksimize edilmesi gerekmektedir. Bu fonksiyon üzerinden OMA yöntemi ile eğitim yapılabilmesi için, eşitlik (3.20)'de belirtilen negatif log-likelihood fonksiyonun minimize edilmesi gerekmektedir.

3.2.4 Çok Katmanlı Perceptron Yapısı

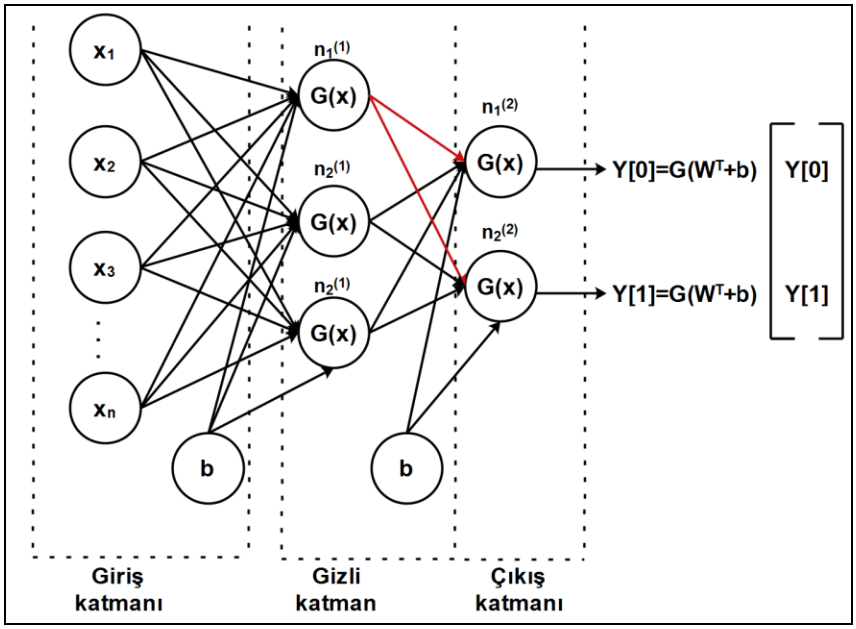
Lineer olarak ayrıştırılabilen veriler için perceptron tarafından oluşturulan lineer bir karar fonksiyonu yeterlidir (Şekil 3. 5). Ancak Şekil 3. 5 (a)'ya bakıldığında ise iki sınıfın örneklerinin tek bir lineer doğru ile ayrıştırılamayacağı açıkça görülmektedir. Bu şekilde çözümü doğrusal olmayan, kompleks karar fonksiyonlarına ihtiyaç duyulan problemlerin, önce doğrusal olarak çözülebilecek alt parçalara bölünerek ayrı ayrı nöronlarda işlenmesi (Şekil 3. 5 (b),(c)) ve bu alt parçaların birleşiminin doğrusal olarak çözüm sağlayabilen çıkış nöronuna verilmesi gerekmektedir. Bu problemin çözümü için çok katmanlı perceptron yapısı kullanılmaktadır [54], [47].



Şekil 3. 5 Doğrusal olarak ayrıştırılamayan veri

ÇKP, Şekil 3. 6' da gösterildiği üzere bir giriş katmanı bir veya daha fazla gizli katman ve bir çıkış katmanından oluşmaktadır. ÇKP aslında bir çeşit lojistik sınıflandırıcıdır ancak klasik bir lojistik sınıflandırıcıdan farklı olarak, öncelikle giriş katmanındaki verileri gizli katmanındaki nöronlarında bulunan aktivasyon fonksiyonları sayesinde doğrusal olmayan transformasyon ile doğrusal olarak ayrıştırılabilir hale getirir ve çıkış katmanındaki nöron/nöronlara iletir. Çıkış katmanında da basit lojistik sınıflandırıcıda olduğu gibi veri doğrusal olarak sınıflandırılır.

Aktivasyon fonksiyonu olarak gizli katmanlarda *sigmoid, tanh ve Relu* fonksiyonları kullanılırken çıkış katmanında bu fonksiyonlara ilaveten softmax fonksiyonu da kullanılır[53].



Şekil 3. 6 Çok katmanlı perceptron yapısı

Şekil 3. 6'da gösterilen ÇKP'deki nöronların çıkış fonksiyonları;

Gizli katmanın 1.nöronunda:

$$Y_1 = G(W_{11}^{(1)} x_1 + W_{21}^{(1)} x_2 + ... + W_{n1}^{(1)} x_n + b_1) \quad (3.21)$$

Gizli katmanın 2.nöronunda:

$$Y_2 = G(W_{12}^{(1)} x_1 + W_{22}^{(1)} x_2 + ... + W_{n2}^{(1)} x_n + b_1) \quad (3.22)$$

Çıkış katmanın 1.nöronunda:

$$Y_2 = G(W_{11}^{(2)} Y_1 + W_{21}^{(2)} Y_2 + b_1) \quad (3.23)$$

Şeklinde hesaplanmaktadır. Buradan anlaşılacağı gibi birinci katmandaki nöronlardan elde edilen Y_1 ve Y_2 verileri ikinci katmandaki çıkış nöronuna giriş verisi olarak beslenmektedir. Çözülmesi gereken problemin lineer olmama (non-linearity) seviyesi arttıkça, problemin yönetilebilir alt parçalara bölünebilmesi adına ÇKP yapısının derinleştirilmesi gerekmektedir[47].

ÇKP eğitimi için lojistik sınıflandırıcıya benzer şekilde OMA yöntemi kullanılır. Lojistik sınıflandırıcıda eğitilmesi gereken tek katman olan çıkış katmanındaki nöronların hata fonksiyonlarının düzetilmesi yeterlidir ancak ÇKP yapısında ise gizli katmanlardaki nöronların hata oranları bir sonraki katmanda girdi olarak kullanıldıkları için sonraki

katmanlardaki nöronların hata oranlarını da etkilemektedirler. ÇKP yapısında eğitimin bu yapıya uygun şekilde yapılması gerekmektedir. Şekil 3. 6'da gizli katmandaki birinci nöronun ($n_1^{(1)}$) bir sonraki katmanda etkilediği nöronlar kırmızı bağlantılar ile gösterilmiştir. Bu yapının eğitimi için geri yayılım algoritması kullanılmaktadır[55].

3.2.4.1 Geri Yayılım Algoritması

Geri yayılım (back probagation) algoritmasının amacı, ÇKP içerisindeki tüm nöronların hata fonksiyonlarının *W* ve *b* parametrelerine göre kısmi türevlerinin alınarak ağırlık güncellemelerinin yapılmasıdır. Geri yayılım yönteminin temeli, meyilli azaltım yöntemine dayalıdır ve hata fonksiyonlarının türevlerinin alınması ile alakalı kurallar burada da aynı şekilde uygulanır. Algoritma temel olarak iki aşamadan oluşur; ileri geçiş (forwardpass) ve geri geçiş (backwardpass). İler geçiş aşamasında tüm nöronlar için çıkış değerleri (3.24)-(3.27) arasındaki eşitliklerinde gösterildiği şekilde üretilir. (3.24)- (3.27) arasındaki eşitliklerde matris notasyonu kullanılmıştır ve $Y^{(\mathrm{L})}$ "*L*" .inci seviyedeki nöronun çıkış değerini göstermektedir[54],[47].

$$Y^{(0)} = x \tag{3.24}$$

$$Y^{(1)} = G(W^{T(1)}x + b^{(1)}) \tag{3.25}$$

$$Y^{(2)} = G(W^{T(2)}Y^{(1)} + b^{(2)}) \tag{3.26}$$

$$Y^{(\mathrm{L})} = G(W^{T(\mathrm{L})}Y^{(\mathrm{L}-1)} + b^{(L)}) \tag{3.27}$$

İleri geçiş sonrasında (3.28) eşitliğinde belirtilen formüle göre çıkış katmanı için türev değeri hesaplanır. Formülde θ şeklinde ifade edilmiş değer, *W* ve *b* parametrelerini ifade etmektedir. Geri geçiş aşamasında çıkış katmanı "*L*" için hata fonksiyonunun türevi:

$$\delta_i^{(\mathrm{L})} = 2(Y^{(\mathrm{L})} - \mathrm{y})\frac{d}{d\theta}\left(\sum_{j=1}^{(L-1)} W_{ji}^{(L-1)} Y_j^{(L-1)} + b_i^{(L)}\right) \tag{3.28}$$

Şeklinde hesaplandıktan sonra "*L-1"*. gizli katmandan başlanarak 1. Gizli katmana kadar olan tüm gizli katmanlardaki nöronlar için hata fonksiyonunun türevi, (3.29) eşitliğinde "*L-1*" gizli katmandaki "*i*" nöronu için verilen fonksiyona göre hesaplanır.

$$\delta_i^{(L-1)} = \sum_{j=1}^{S_{L+1}} W_{ij}^{(L-1)} \delta_j^{(L)} \frac{d}{d\theta} \left(\sum_{j=1}^{S_{L-2}} W_{j1}^{(L-2)} Y_j^{(L-2)} + b_i^{(L-1)} \right) \quad (3.29)$$

Her gizli katman için bu değerler hesaplandıktan sonra, tüm nöronlara ilişkin *W* ve b değerleri için hata fonksiyonu nihai türev değerleri, (3.30) ve (3.31) eşitliklerine göre hesaplanır ve bu değerler (3.5) ve (3.6) eşitliklerinde belirtildiği gibi güncellenir.

$$\Delta W_{ji}^{(L)} = \delta_i^{(L+1)} Y_j^{(L)} \quad (3.30)$$

$$\Delta b_i^{(L)} = \delta_i^{(L)} \quad (3.31)$$

3.2.5 Otomatik Kodlayıcılar

Bu bölümde yapay sinir ağlarının bir çeşidi olan ve DSA'ların temelini oluşturan modellerden olan; deterministik otomatik kodlayıcı ve gürültü giderici otomatik kodlayıcı yöntemleri açıklanmıştır.

3.2.5.1 Otomatik Kodlayıcı (Deterministik Otomatik Kodlayıcı)

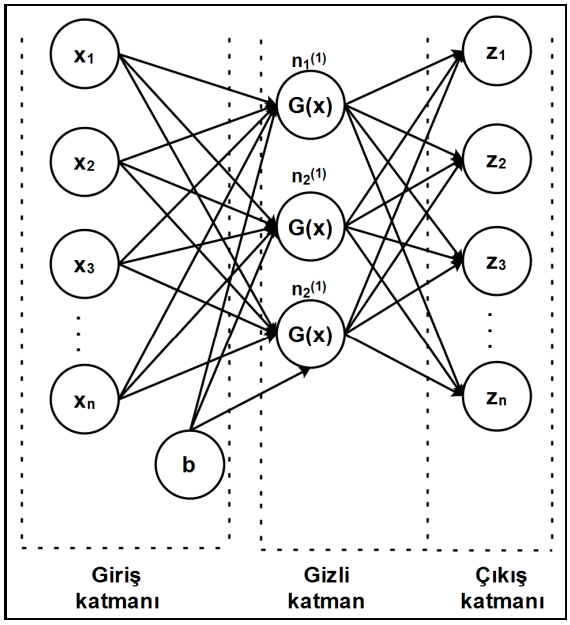
YSA'ların bir çeşidi olan OK'lar; giriş katmanı, bir veya daha fazla gizli katman ve bir çıkış katmanından oluşurlar[25]. OK'larda genellikle gizli katmanlardaki nöron sayısı giriş katmanına göre daha az sayıdadır. Buna ek olarak, çıkış katmanındaki nöron sayısı da giriş katmanındaki nöron sayısıyla aynıdır (Şekil 3. 7). OK'lar derin YSA'larda sınıflandırıcı eğitimine yardımcı olarak kullanılabilecekleri gibi doğrudan anomali tespitinde de kullanılabilirler[34]. OK, kodlama ve çözümleme isminde iki ayrı süreçten oluşur. Kodlama sürecinde girilen veriler kendi özellik boyutlarından daha az sayıda özellik boyutuyla ifade edilirler. Çözümleme sürecinde ise veriler gizli katmandaki kodlama bilgileri kullanılarak çıkış katmanında, giriş katmanıyla aynı özellik boyutuna sahip olacak şekilde tekrar oluşturulurlar.

$$Y = G(W^T x + b) \quad (3.32)$$

$$z = G(W Y + b^T) \quad (3.33)$$

$$HKT = \sum_{i=1}^{D} (z^{(i)} - x^{(i)})^2 \quad (3.34)$$

OK, sırası ile eşitlik (3.32) ve (3.33)'da verildiği şekilde kodlama ve çözümleme işlemleri ile deterministik eşleme yapar. *"G"* fonksiyonu doğrusal olmayan bir aktivasyon fonksiyonunu simgeler ve genellikle *sigmoid* veya *tanh* fonksiyonları kullanılır[25]. W^T bir nöronun kodlama aşamasında *x* giriş değerinden *Y* çıkışını üretmek için gereken ağırlık değerini temsil ederken *b* "bias" değerini temsil eder. *z* değeri ise *Y* gizli katman değeri kullanılarak *x* girişinin yeniden oluşturulmuş halidir. *W* ve b^T ise çözümleme aşamasındaki ağırlık ve bias değerini temsil eder. Verinin tekrar oluşturulmuş hali ile orijinal hali arasındaki fark, yeniden oluşturma hatası (YOH) olarak adlandırılır[25] ve OK eğitimi bu değerin meyilli azaltılması ile yapılır. OK için YOH değeri, tekrar oluşturma hatalarının kareleri toplamı (HKT) veya çapraz entropi yöntemi ile bulunur. Bu tez kapsamında gerçeklenen modellerde YOH hesaplaması için (3.34) 'de verilen HKT yöntemi seçilmiştir ve *G* için *sigmoid* fonksiyonu kullanılmıştır.



Şekil 3. 7 Otomatik kodlayıcı

3.2.5.2 Gürültü Giderici Otomatik Kodlayıcılar

OK'nın eğitim verisini aşırı öğrenmesi sonucunda model, eğitim verisinin birim fonksiyonu haline gelir. Bu sorunun çözülmesi için genelleştirilmiş bir modele ihtiyaç duyulmaktadır. Genelleştirilmiş model oluşturulması ve verilerin birbirleriyle olan istatistiksel bağlantılarının çıkarılması için stokastik yapıdaki; Seyrek OK, GGOK[25], ve sınırlandırılmış Boltzman makineleri (restricted Boltzman machine) yöntemi

kullanılabilir[32]. Bu tez kapsamında anomali tespitinde hem deterministik OK hem de stokastik GGOK kullanılmıştır. GGOK, deterministik OK'dan farklı olarak girdilerin belirli bir yüzdesini rastgele seçerek sıfırlar ve bu bozulmuş olan girdiden verinin orijinalini tekrar oluşturmaya çalışır. Bu yüzden GGOK stokastik bir yapıya sahiptir.

3.2.6 Yapay Sinir Ağı Eğitimini Güçlendiren Ek Konfigürasyonlar

ÇKP'ler, literatürde aşırı öğrenme diye tariflenen sorundan dolayı eğitim verisi üzerinde yüksek başarım ile sınıflandırma yapabilirlerken kontrol ve test verisi üzerinde başarısız sonuç vermektedirler. Bu, modelin eğitim verisini ezberlediğini gösteren bir problemdir ve temel olarak iki sebepten kaynaklanmaktadır[47];

- Eğitimin sadece parametre değerlerine bağlı olması, yani modelin parametre bağımlı hale gelmesi.
- Eğitimin gereğinden fazla uzatılması.

3.2.6.1 L1 ve L2 Regülarizasyonu

Yapay sinir ağı eğitiminde modelin tamamen *W* ve *b* değerlerine bağımlı olmasını ortadan kaldırma amacı ile L regülarizasyonu yapılır. L regülarizasyonu hata fonksiyonuna *W* ve *b* parametrelerinden bağımsız olarak L1 ve L2 isminde iki ayrı parametre daha eklenir. (3.35) eşitliğinde; λ regülarizasyon parametresinin önemini belirleyen sabiti, R(θ) hata fonksiyonu J(θ)'ya eklenmiş regülarizasyon değerini, θ modelde bulunan tüm parametrelerin kümesini ({*W, b*}), E_L(θ) hata fonksiyonunun regülarizasyon parametresi eklenmiş halini temsil etmektedir[32].

$$E_L(\theta) = J(\theta) + \lambda R(\theta) \tag{3.35}$$

$$R(\theta) = \|\theta\|_p$$
$$\|\theta\|_p = \sum_{i=1}^{\theta} \left(|\theta_i|^p \right)^{\frac{1}{p}} \tag{3.36}$$

(3.36) eşitliğinde belirtilen *p* değeri genellikle 1 veya 2 değerini alır. Bu yüzden yöntem L1/L2 regülarizasyonu şeklinde adlandırılmıştır. E_L(θ) Fonksiyonundaki J(θ) kısmı modelin eğitiminin keskinliğini sağlarken, λR(θ) kısmı modelin genelleştirilmesini sağlar. Bu sayede modelin daha önceden gördüğü örnekleri tespit etme başarımı ile hiç

görmediği örnekleri sınıflandırma başarımı yani genelleştirilmesi arasında bir denge sağlanmaya çalışılır. L1 ve L2 regülarizasyon parametreleri birlikte kullanılabilirler. Bu tez kapsamında yapılan gözetimli yapay sinir ağı eğitimlerinde L1/L2 regülarizasyonu yöntemi kullanılmıştır.

3.2.6.2 Erken Durma

Modelin aşırı öğrenmesini engellemek ve test verisine özel bir model eğitiminden kaçınmak adına bütün veri kümesi; *eğitim, kontrol* ve *test veri kümesi* olmak üzere üç ayrı parçaya ayrılır. Modelin hata fonksiyonu eğitim veri kümesinde elde edilen sonuçlar neticesinde OMA yöntemi ile iyileştirilirken, modelin aşırı öğrenmediğinin kontrolü kontrol veri kümesi üzerinden yapılır. Eğitim ve kontrol kümeleri kullanılarak tamamlanan eğitim sürecinden sonra modelin gerçek performansı test veri kümesi üzerinden ölçülür[32].

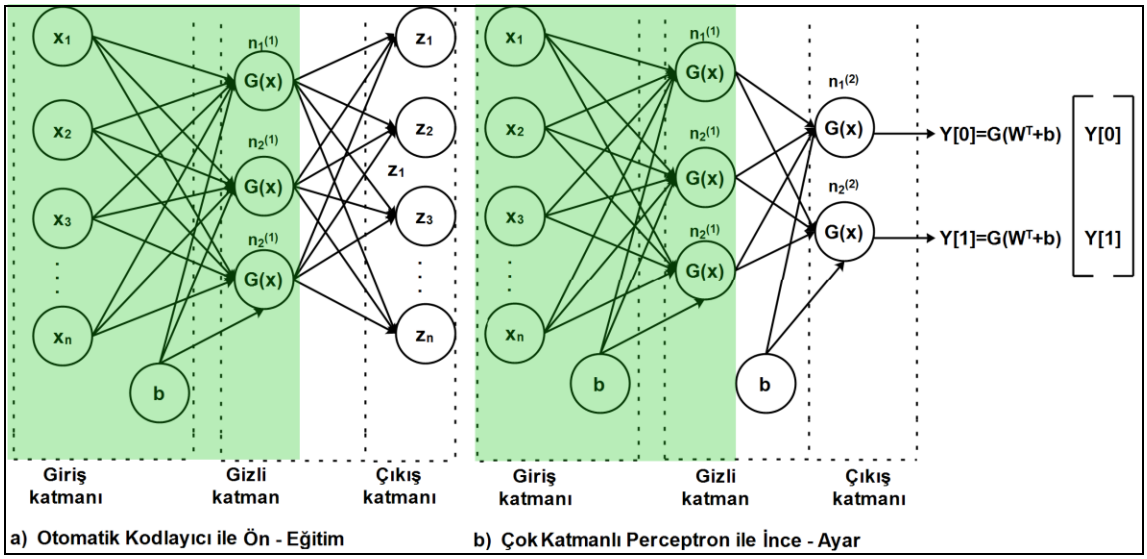
Erken durma (Early Stopping) modelin aşırı öğrenmesini engelleyen bir yöntemdir. Bu yöntemde, eğitimin başlangıcında model, eğitim kümesi ile sabit bir tur sayısı boyunca eğitilir. Sonrasında kontrol veri kümesi kullanılarak modelin kontrol verisi üzerindeki başarımı ölçülür. Daha sonra model tekrar sabit bir tur boyunca eğitilir ve kontrol verisi üzerindeki başarımı tekrar ölçülür. Son elde edilen başarım bir önceki başarıma kıyasla belirli bir sabit oran miktarınca artış gösterdiği sürece modelin eğitim turu miktarı arttırılır ve eğitime devam edilir. Ancak model kontrol verisi üzerinde bir önceki başarımına göre beklenen artışı gösteremediyse bu, modelin eğitim verisini ezberlemeye başladığı ancak kontrol verisi üzerinde gelişme kaydetmediğini gösterir ve eğitim durdurulur. Sonrasında en son performans artışı gösteren model, eğitim sonucunda elde edilmiş olan nihai model olarak belirlenir.

3.3 Anomali Tespit Modelleri

3.3.1 Derin Yapay Sinir Ağı Modeli

Derin yapay sinir ağı modelleri, gözetimsiz eğitime dayalı çok katmanlı otomatik kodlayıcılar ve gözetimli eğitime dayalı ÇKP yapılarının birlikte kullanılmasından oluşur. Derin YSA'ların eğitimi; ön-eğitim ve ince-ayar adında iki aşamadan meydana gelir.

Ön-eğitim aşamasında modelin otomatik kodlayıcı bölümü, Şekil 3. 8 (a)'da belirtildiği üzere etiketsiz verilerin birbirleri ile olan ilişkilerini tespit eder ve bu sayede veri kümesinin altında yatan temsili öğrenir. Bu işlem deterministik OK yöntemi kullanıldığında boyut daraltılarak yapılabileceği gibi GGOK veya sparse-autoencoder gibi stokastik yöntemler kullanıldığında ise boyut genişletme şeklinde de yapılabilir. İnce-ayar aşamasında ise çok katmanlı perceptron yapısı otomatik kodlayıcının ön-eğitim aşamasında elde ettiği bilgilerden yararlanılarak gözetimli olarak eğitilmektedir (Şekil 3. 8 (b)).



Şekil 3. 8 Derin Yapay Sinir Ağı

Derin YSA'da tek gizli katmana sahip otomatik kodlayıcılar arka arkaya eklenerek çok katmanlı bir otomatik kodlayıcı modeli oluşturulur. ön-eğitim aşamasında, birbirlerine bağlanmış olan bu otomatik kodlayıcıların her biri en dış katmandaki otomatik kodlayıcıdan başlanarak sırası ile bağımsız olarak eğitilir(greedy-wise training)[56],[57].

Ön-eğitim işleminin amacı, veri kümesini daha iyi temsil edebilen özellikleri belirlenmesi ve ince-ayar aşamasından önce ÇKP yapısının ağırlıklarına uygun değerler atanmasıdır. Gözetimli ÇKP eğitiminde rastgele ilklendirilmiş ağırlıklar kullanılması, olaslıksal meyilli azaltım yöntemi ile iyileştirilen hata fonksiyonunun yerel minimum değerlerine takılmasına sebep olmakta ve bunun sonucunda modelin kendi kapasitesinin altında daha düşük bir başarım ile çalışmasına yol açmaktadır.

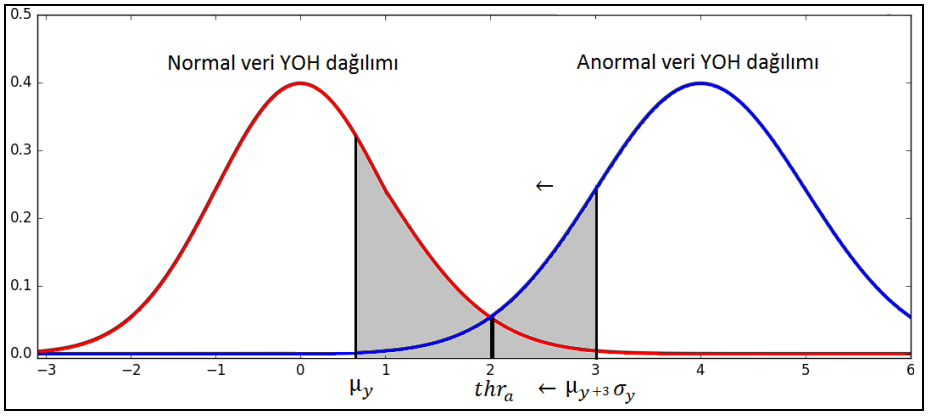
Modelin sınıflandırma görevini yürüten ÇKP yapısı, Şekil 3. 8 (a)'da gösterilen OK'nın kodlama bölümlerinin tamamını aynı şekilde kullanır(Şekil 3. 8 (b)). Buna karşın otomatik kodlayıcıların çözümleme bölümlerinin hiç biri ÇKP yapısına dahil edilmez. Sadece kodlama kısımlarından oluşan otomatik kodlayıcıların son katmanına lojistik sınıflandırıcı eklenir ve çok katmanlı ok yapısı bu şekilde ÇKP yapısına dönüştürülür. Bu sayede ÇKP, veriye ait en önemli özellikleri kullanır ve ağırlıkları da otomatik kodlayıcıların kodlama bölümlerinin ağırlıklarıyla ilklendirilir.

İnce-ayar aşamasında ÇKP etiketli veri ile gözetimli olarak, sınıflandırma hatasını düzeltmek üzere eğitilir ve tüm modelin ağırlıkları ideal değerlere ulaşıncaya kadar eğitime devam edilir[54],[55]. Bu çalışmada OK'lar içerisindeki nöronların aktivasyonu için sigmoid fonksiyonu seçilmiştir ve sınıflandırıcının çıkış katmanında softmax aktivasyon fonksiyonu kullanılmıştır. Ayrıca, modelin aşırı öğrenmesini engellemek amacıyla erken durma ve L1, L2 regülarizasyonu (weight decay) yöntemleri kullanılmıştır. Bu tez kapsamında; OK ve GGOK yöntemlerine dayalı 2 ayrı DSA tabanlı anomali modeli oluşturulmuştur Bu modeller; OK – DSA ve GGOK – DSA şeklinde isimlendirilmiştir.

3.3.2 Önerilen Otomatik Kodlayıcı Tabanlı Anomali Tespit Modeli

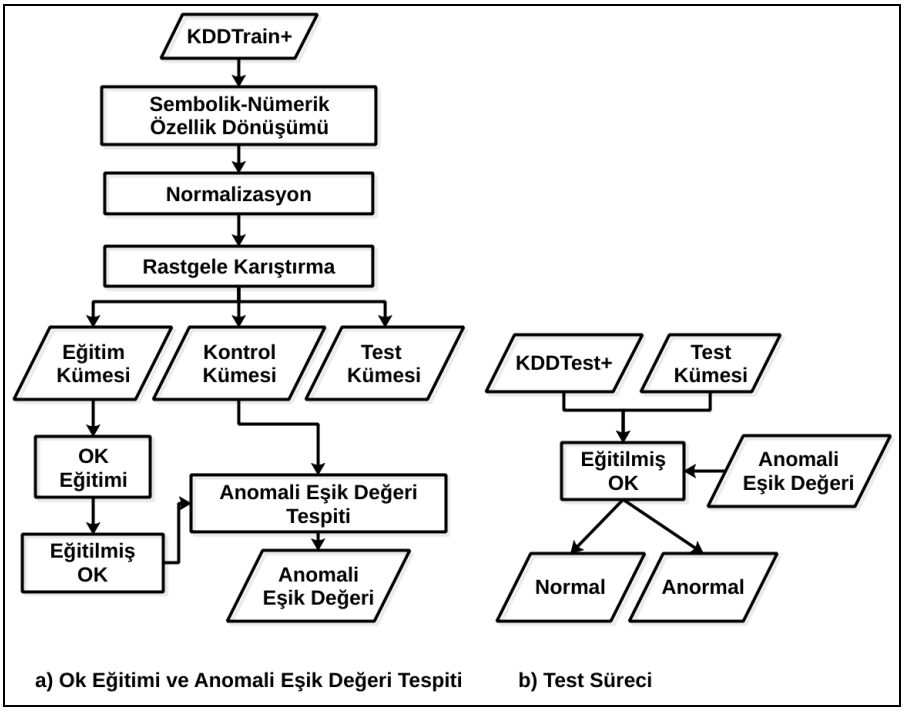
Bir OK, normal ve anormal etiketine sahip bir veri kümesindeki normal veriler kullanılarak yarı gözetimli bir şekilde eğitildiğinde sadece normal veriyi yeniden oluşturmayı öğrendiğinden normal veriler için düşük YOH değerleri üretirken anormal veriler için normallere kıyasla daha yüksek YOH değerleri üretecektir. Bu durumda OK'nın başarımı YOH değerinin doğru yorumlanmasına bağlıdır. Belirli bir eşik değerinin (thr_a) altındaki YOH değerleri normali ifade ederken bu eşiğin üzerindeki değerler anomaliyi ifade edecektir. Mevcut çalışmalarda bu eşik değerinin eğitim verisindeki en

yüksek YOH'a bağlı olması[5] veya çekirdek yoğunluk hesabıyla belirlenmesi[34] yönünde öneriler yapılmıştır. Bu çalışmada ise bahsi geçen çalışmalarından farklı olarak OK modelini genelleştirmek adına stokastik bir anomali eşik değeri tespit (SAEDT) yöntemi önerilmiştir. Bu yöntemde, toplam veri kümesinin rastgele seçilmiş bir kısmı eşit sayıda normal ve anormal örnek içerecek şekilde doğrulama kümesi olarak ayrılır.



Şekil 3. 9 Eğitim kümesi ile bulunmuş μ_y ve σ_y değerlerinin yardımı ile doğrulama kümesi üzerinden en başarılı anomali eşik değerinin (thr_a) tespiti

Şekil 3. 9'da ilk aşamada sadece normal örnekler ile eğitilmiş olan OK'nın doğrulama kümesindeki normal ve anormal veriler için oluşturduğu Gauss dağılımları sırası ile kırmızı ve mavi eğriler ile verilmiştir. Bu eğrilerin birbirlerine göre durumlarından da görüleceği üzere ilk aşama sonucunda eğitim verisindeki normal veriler için elde edilen ortalama YOH μ_y değerinin üç standart sapma σ_y fazlasının(μ_y +3 σ_y) eşik değeri (thr_a) olarak kullanılması durumunda normal verilerin %99,9'u başarılı bir şekilde sınıflandırılırken anormal veriler için yanlış pozitif değeri yüksek olacaktır. Bu açıdan thr_a değerinin optimize edilmesi gereklidir ve bu çalışma kapsamında μ_y 'den başlanarak üç σ_y sağına kadar olan bölge 100 eşit parçaya bölünerek incelenmiş ve doğrulama verisi içindeki hem normal hem de anormal verilerin en yüksek başarımla sınıflandırılmasını sağlayan bölgeye ait değer thr_a değeri olarak seçilmiştir (Tanım 3.2). Önerilen anomali tespit yöntemine ilişkin genel akış Şekil 3. 10'da gösterilmiştir. Bu tez kapsamında; önerilen SAEDT yöntemini kullanan OK ve GGOK tabanlı iki adet anomali tespit modeli önerilmiştir. Bu modeller OK-SAEDT ve GGOK-SAEDT şeklinde isimlendirilmişlerdir.



Şekil 3. 10 Otomatik kodlayıcı tabanlı anomali tespiti

Tanım 3.2 Anomali Eşik Değeri Tespit Algoritması**:**

"*ts*" (eğitim kümesi), "*vs*" (kontrol kümesi),"*ok*"(otomatik kodlayıcı)", "*YOH*" (yeninden oluşturma hatası), "*acc*" (doğruluk)

1. *ok'*yı *ts* ile eğit
2. *ts* için üretilen *YOH* değerlerinin μ_y ve σ_y değerlerini hesapla
3. $thr_a \leftarrow \mu_y + 3\,\sigma_y$
4. Eğer $thr_a > \mu_y$ değilse 9. adıma git.
5. *ok* 'nın *vs* üzerindeki *acc* değerini hesapla
6. *sonucListesi* 'ne (thr_a - *acc*) ikilisini ekle.
7. $thr_a \leftarrow thr_a$ - 0.01
8. 4. Adıma git
9. *sonucListesi* 'nden En iyi *acc* değerine sahip *thr* değerini seç
10. $thr_a \leftarrow thr$
11. thr_a değerini dön.

BÖLÜM 4

DENEYSEL ORTAMIN HAZIRLANMASI VE PERFORMANS DEĞERLENDİRMESİ

Bu bölümün ilk kısmı olan "Deneysel Ortamın Hazırlanması" başlığı altında; tez kapsamında uygulaması yapılmış olan anomali tespit modellerinin kodlanması için kullanılan araçlar, modellerin test edilmesi için yapılan ön hazırlıklar, modellerin konfigürasyonu ve parametrelerine ilişkin detaylar anlatılmıştır. Sonrasında deneysel sonuçlar bölümünde ise; performans ölçüm metrikleri tariflenmiş, değerlendirme için kullanılacak olan eğitim-kontrol-test veri kümelerinin oluşturulmasına ilişikin detaylar açıklanmış ve gerçeklenen modellerin NSL-KDD üzerindeki anomali tespit başarımları bu ölçüm metrikleri bazında verilmiştir. Elde edilen sonuçlar hem birbirleri ile hem de literatürde NSL-KDD üzerinde en başarılı sonuçları veren benzer değerlendirme ölçütlerini kullanan diğer çalışmalarla karşılaştırılmıştır.

4.1 Deneysel Ortamın Hazırlanması

Bu tez kapsamında kullanılan tüm anomali tespit modelleri, Theano kütüphanesi kullanılarak Python dili ile kodlanmıştır. Kodlanmış olan modellerin eğitim süreçlerinin hızlandırılması adına grafik işlemci tabanlı bir sunucu kurulmuştur. Ek olarak, NSL-KDD veri kümesini YSA eğitimine daha elverişli hale getirmek adına veri kümesinde bir dizi ön hazırlık yapılmıştır.

4.1.1 Theano Kütüphanesi

Theano, temel olarak matematiksel fonksiyonların sembolik graflar olarak ifade edilebilmesine olanak tanıyan ve bu sembolik grafların hesaplama karmaşıklığını düşürmek adına optimizasyonunu yapan, matris hesaplamaları ve kompleks matematiksel işlemler üzerine özelleştirilmiş bir Python kütüphanesidir. İlk sürümü 2008 yılında yayınlanan Theano, New(3-close) BSD altında lisanslanmış açık kaynaklı ve ücrestiz bir kütüphanedir[58].

Theano'nun, sembolik graflar üzerinden otomatik türev alma özelliği sayesinde manuel olarak hesaplanan türev sonucunda meydana gelebilecek hataların önüne geçilmektedir. Ayrıca Theano, bu özelliği sayesinde derin öğrenme ve yapay sinir ağlarının eğitiminde kullanılan ve temeli türev alma işlemine dayalı, meyilli azaltım yönteminin etkin bir şekilde gerçeklenmesine imkan tanımaktadır[58].

Matris hesaplamaları üzerine özelleşmiş bir kütüphane olan Theano, yapay sinir ağlarının temelini oluşturan matris işlemlerini yüksek performansla yapabilmek adına numPy ve CUDA kütüphanelerini kullanır. CUDA, n-boyutlu matrisleri grafik işlemci hafızasında saklamak ve doğrudan grafik işlemci üzerinde hesaplanmalar yapmak için kullanılır. Theano, CUDA kodlarını kendi içerisinde barındırdığı CUDA kod üreticisi sayesinde otomatik olarak ürettiği için ekstradan CUDA için kod yazmaya gerek kalmadan yüksek karmaşıklık içeren fonksiyonlar grafik işlemci üzerinde çalışacak hale getirilebilmetekdir. Theano üzerinde gerçeklenen programlar kodda herhangi bir değişikliğe gerek kalmadan hem CPU da hem de GPU da çalıştırılabilir. Ek olarak, Theano CPU için üretilecek kodun derlenmesinde gcc kullanırken GPU için nvcc kullanır[59].

Theano kütüphanesi ile gerçeklenen programların numpy, scipy ve C++ dillerine göre CPU üzerinde 6,5 kat ve GPU üzerinde de 44 kat daha hızlı çalıştıkları tespit edilmiştir. Theano kütüphanesi; c dil üreticisi, CUDA kod üreticisi ve numpy kütüphanesini etkin bir şekilde birleştiren, bütüncül bir sistem olması sayesinde böylesine yüksek bir performans ile çalışabilmektedir[59].

4.1.2 CUDA Kütüphanesi

Grafik işlemciler (GPU) bilgisayar grafikleri/video oyunlarına göre özelleştirilmiş ve bu alanlarda yaygın olarak kullanılan cihazlardır. GPU'larda matris ve vektörel tabanlı matematiksel işlemler çok hızlı şekilde yapılabilmektedir. Bu avantaj GPU'ların bilgisayar grafikleri/bilgisayar oyunlarını yanı sıra, temeli matris işlemlerine ve türevlerine dayanan yapay sinir ağlarının eğitimi için de yaygın bir biçimde kullanılmalarını sağlamıştır.

Ayrıca donanımsal alanda her geçen yıl artan teknolojik gelişmeler sayesinde bireysel kullanıcılar için üretilen CPU ve GPU'lar çok çekirdekli hale gelmiştir. Bu gelişmeler, CPU ve GPU'ların paralel programlama işlemlerinde kullanılan etkin bir enstrüman olmalarını sağlamıştır[60]. Bunların yanı sıra, artan bilgisayar oyun pazarı bu cihazların fiyatlarının düşmesine sebep olmuştur[48].

CUDA (compute unified device architecture) 2007 yılında nvidia firması tarafından çok çekirdekli GPU'lar üzerinde paralel programlama işlemlerini daha etkin yapabilmesi adına geliştirilmiş bir kütüphanedir. CUDA ve CUDA destekli grafik kartları, paralel programlama yapan programcılara CPU ve GPU arasındaki veri aktarımını yönetebilmeleri adına birçok fonksiyonalite sağlamıştır[61]. CUDA kütüphanesi; hesaplamaya dayalı kimya (computational chemistry), boşluklu matris çözümü (sparse matrix solver), arama, sıralama, yapay sinir ağları (derin öğrenme) ve fiziksel modelleme gibi birçok konuda kullanılmaktadır[60].

Tez kapsamında gerçeklenen modellerin eğitimlerinin hızlı yapılabilmesi adına CUDA destekli grafik işlemci içeren bir hesaplama sunucusu kurulmuştur. Kurulmuş olan sunucuya ilişkin detaylar Çizelge 4. 1'de verilmiştir.

Çizelge 4. 1 Grafik işlemci destekli hesaplama sunucusu

CPU	İ5-3470 @3.2GHZ x 4
RAM	16 GB
GPU	GTX TITAN X 12GB 384 bit, 3072 CUDA Cores
OS	Ubuntu 16.04 LTS

4.1.3 NSL-KDD Üzerinde Ön Hazırlık

NSL-KDD veri kümesini YSA eğitimine daha elverişli bir hale getirmek adına veri kümesinde bir dizi ön hazırlık yapılmıştır. İlk olarak veri kümesindeki 41 özellikten alfanümerik olan üç adet özellik; "*protocol-type*", "service" ve "*flag*", 1-n kodlama yöntemiyle 84 ayrı ikili özelliğe dönüştürülmüştür. Ek olarak, tüm veriler için sıfır değerine sahip olan "*num-outbound-cmds*" isimli özellik silinmiştir. Bu şekilde veri kümesindeki toplam özellik sayısı 121'e çıkarılmıştır. Sonrasında maksimum-minimum normalizasyonu yapılarak tüm özellikler [0,1] aralığına ölçeklenmiştir.

4.1.3.1 1-N Kodlama Yöntemi

Kategorik özelliklerin nümerik özelliklere dönüştürülmesi amacı ile kullanılan 1-n kodlama yönteminde bir kategorik özelliğin veri kümesindeki tüm farklı değerleri ayrı birer ikilik özelliğe dönüştürülür.

Örneğin Çizelge 4. 2'de gösterildiği üzere, F_x özelliği için veri kümesinde bulunan tüm farklı değerlerin kümesi S={v_1, v_2, ve v_3} olarak kabul edilsin. 1-n kodlama yöntemi bu özellik için uygulandığında veri kümesine v_1, v_2, ve v_3 adında üç ayrı kolon eklenir. Sonrasında veri kümesindeki her bir verinin F_x özelliğinin değeri; v_1, v_2, ve v_3 değerlerinden hangisine eşit ise o değere ilişkin kolonun ilgili hücresine 1 değeri yazılırken yeni eklenen diğer kolonlara ilişkin hücrelere ise 0 değeri yazılır.

Çizelge 4. 2 F_x özelliği için 1-n kodlama

	F_x	v_1	v_2	v_3
$Veri_1$	v_1	1	0	0
$Veri_2$	v_2	0	1	0
$Veri_3$	v_3	0	0	1
.				
.				
$Veri_n$	v_1	1	0	0

1-n kodlama yerine, v_1, v_2, ve v_3 değerlerine sırası ile 0, 1 ve 2 değerleri verilerek bir dönüşüm yapılsaydı, v_1 özelliği ile v_2 özelliği arasındaki fark 1 birim olurken v_1 özelliği ile v_3 özelliği arasındaki fark ise 2 birim olacaktı. Ancak gerçekten v_1 değerinin, v_2 değerine yakınlığının v_3 değerine olan yakınlığından daha fazla olup olmadığı bilinmemektedir. Bu tercih edilen dönüşüm işleminin bir yan etkisidir. Dolayısıyla yapılan bu varsayımlar veri kümesindeki verilerin bu şekilde dönüşümü yapılmış özellikler bakımından birbirlerinden uzaklaşmasına sebep olur. Bu da yapay sinir ağına ilişkin ağırlıkları etkileyeceği için modelin başarımını düşürebilir. Bu yüzden daha doğru bir dönüşüm yapılabilmesi adına 1-n kodlama yöntemi tercih edilmiştir.

KDDTrain+ veri kümesindeki alfanümerik üç özellikten;

- "*protocol-type*", 3 farklı değere,
- "*service*", 70 farklı değere,
- "*flag*", 11 farklı değere

Sahiptir.

Bu kolonlar yukarıda tariflendiği şekilde 1-n kodlama yöntemi ile alfanümerik özellikten nümerik özelliğe dönüştürüldüklerinde 84 ayrı ikilik özellik oluşur. Mevcut 3 alfanümerik özellik silinir ve "num-outbound-cmds" özelliğinin de silindiği dikkate alınırsa veri kümesi toplamda 121 ayrı özelliğe sahip olur[24].

4.1.3.2 Maksimum-Minimum Normalizasyonu

Maksimum-minimum normalizasyonunda veri kümesindeki her bir özelliğe ait maksimum ve minimum değerleri bulunduktan sonra veri kümesindeki her bir verinin her bir özelliği için eşitlik (4.1)'de gösterildiği şekilde [0-1] aralığına ölçeklenmiş yeni değerler bulunur. Değerlerin küçük bir sayı aralığına ölçeklenmesi YSA'nın eğitimi açısından çok önemlidir. Çünkü YSA'ya çok büyük değerler girdi olarak verildiğinde modelin ağırlıklarının hesaplanması çok zaman almaktadır[62].

$$x' = \frac{x - min\_deger}{max\_deger - min\_deger} \tag{4.1}$$

Anomali tespit modelleri, KDDtrain+_%20'nin bölünmesi ile oluşturulmuş eğitim kümesi ile eğitildikten sonra ilk olarak KDDTrain+_%20'nin belirli bir yüzdesinden oluşturulmuş olan test kümesi üzerinden test edilir ve sonrasında KDDTest+ üzerinde modelin gerçek performans değerlendirmesi yapılır. Burada normalizasyonun KDDTrain+_%20 ve KDDTest+ için farklı maksimum-minimum değerleri ile yapılması modelin başarımının olumsuz olarak etkiler. Çünkü Eğitilen YSA'nın ağırlıkları KDDTrain+_%20'nın normalize edilmiş özelliklerine göre ayarlanmış olur. Buna karşın, KDDTest+'nın bağımsız olarak normalize edilmesi, modelin eğitim sırasında özelliklere yüklediği ağırlıkların doğru yorumlanamamasına sebep olur. Örneğin, KDDTrain+_%20 kümesinde F_1 özelliği için maksimum değer "255" iken minimum değer "0" olsun. maksimum-minimum normalizasyonundan sonra F_1 özelliği "255" olan verilerin artık yeni F_1 değerleri "1" olacaktır. Benzer şekilde KDDTest+'nında bağımsız bir şekilde maksimum-minimum normalizasyon işleminden geçirildiği düşünülsün ve F_1 özelliği için veri kümesindeki maksimum değer "70" minimum değer ise "0" olsun. KDDTest+'da F_1 özelliği "70" olan tüm verilerin değerleri, normalizasyondan sonra 1 olacaktır. KDDTest+'nın normalize edilmiş verileri, eğitilmiş model üzerinde test edildiğinde model, KDDTest+'da gerçek değeri "70" olan veriyi normalize edildikten sonra "1"'e dönüştüğü için gerçekte "255" değerine sahipmiş gibi algılar ve bu da başarımı olumsuz olarak etkiler. Normalizasyon için kullanılacak maksimum minimum değerleri NSL-KDD içindeki en kapsayıcı veri kümesi olan KDDTrain+ üzerinden bulunmuştur. KDDTrain+_%20 ve KDDTest+ veri kümelerinin normalizasyonu bu maksimum-minimum değerleri kullanılarak yapılmıştır.

4.2 Performans Değerlendirmesi

4.2.1 Performans Değerlendirme Metrikleri

Bu tez kapsamında gerçeklenen yöntemlerin performanslarının değerlendirilmesi amacı ile kesinlik (precision), duyarlılık (recall), f-ölçütü (f-measure) ve doğruluk (accuracy) ölçüm metrikleri kullanılmıştır. Bu metrikler; doğru pozitif (TP - true positive), yanlış pozitif (FP - false positive) , doğru negatif (TN - true negative) ve yanlış negatif (FN - false negative) ölçütleri üzerinden hesaplanmaktadırlar[18].

Doğru Pozitif: Gerçekte anormal olan bir veri model tarafından anormal olarak sınıflandırılıyorsa TP olarak kabul edilir.

Yanlış Pozitif: Gerçekte normal olan bir veri model tarafından anormal olarak sınıflandırılıyorsa FP olarak kabul edilir.

Doğru Negatif: Gerçekte normal olan bir veri model tarafından normal olarak sınıflandırılıyorsa TN olarak kabul edilir.

Yanlış Negatif: Gerçekte anormal olan bir veri model tarafından normal olarak sınıflandırılıyorsa FN olarak kabul edilir.

Doğruluk: Başarılı sınıflandırılmış verilerin sayısının tüm verilerin sayısına oranıdır.

$$Doğruluk = \frac{TP + TN}{NormalÖrnekSayısı + AnormalÖrnekSayısı} \tag{4.2}$$

Kesinlik: Anormal olarak başarılı sınıflandırılmış verilerin sayısının, tüm anormal olarak sınıflandırılmış verilerin sayısına oranı.

$$Kesinlik = \frac{TP}{TP + FP} \tag{4.3}$$

Duyarlılık: Anormal olarak başarılı sınıflandırılmış verilerin tüm anormal verilerin sayısına oranı.

$$Duyarlılık = \frac{TP}{TP + FN} \tag{4.4}$$

F-Ölçütü: Kesinlik ve duyarlılık metriklerinin harmonik ortalamasıdır. Sistemin genel başarımını tariflemek için kullanılır.

$$F - Ölçütü = \frac{2 * Doğruluk * Kesinlik}{(Doğruluk + Kesinlik)} \tag{4.5}$$

4.2.2 Eğitim - Kontrol - Test Veri Kümelerinin Oluşturulması

OK - DSA yöntemi ile GGOK-DSA yöntemlerinin KDDTrain+_%20 üzerindeki anomali tespit başarımları ölçümü için 10 kat çapraz doğrulama tekniği kullanılmıştır. Bu yüzden, KDDTrain+_20'nin %80'lik bölüme eğitim kümesine , %10'luk bölümü kontrol kümesine ve kalan %10'luk bölümü de test kümesine ayrılmıştır. 10 kat çaprazlama

tekniğinin her bir aşamasında test veri kümesi için KDDTrain+_20'nin farklı %10'luk bölümü kullanılır. Dolayısıyla eğitim ve kontrol kümeleri de her aşamada KDDTrain+_20'nin farklı bölümlerinden oluşturulmaktadır.

Önerilen OK - SAEDT ve GGOK - SAEDT anomali tespit modelleri için KDDTrain+_%20 içerisinde bulunan normal ve anormal veriler kendi içlerinde rastgele karıştırıldıktan sonra karıştırılmış olan verilerden normal verilerin tamamı ve anormal verilerin bir bölümü seçilerek sistemi eğitmek, doğrulamak ve test etmek adına üç adet alt veri kümesi oluşturulmuştur. Bu üç veri kümesi bir bütün olarak düşünüldüğünde eğitim kümesi bu bütünün %60'ını oluşturmaktadır ve sadece normal örnekleri içermektedir. Test ve doğrulama kümelerinin her biri bu bütünün %20'sini oluşturmaktadır ve her biri eşit sayıda normal ve anormal örnek içermektedir. Oluşturulan test kümesi, KDDTest+ kümesi ile karıştırılmaması adına test%20 olarak adlandırılmıştır.

4.2.3 Anomali Tespit Modelleri için Test Konfigürasyonları

Bu bölümde uygulaması gerçeklenmiş olan mevcut ve önerilen anomali tespit yöntemlerinin konfigürasyonlarına ilişkin detaylar açıklanmıştır.

4.2.3.1 Mevcut Yöntemlere Dayalı Anomali Tespit Modelleri

Deterministik Otomatik Kodlayıcı Tabanlı Derin Yapay Sinir Ağı (OK - DSA):

- 121 nöronluk giriş katmanı, sırası ile 60, 30 ve 15 nöron içerecek şekilde üç gizli katman ve iki nöronluk bir çıkış katmanından oluşmaktadır.
- Ön-eğitim aşaması için deterministik OK kullanılmıştır.
- Aktivasyon fonksiyonu olarak, gizli katmanlarda sigmoid, çıkış katmanında ise softmax fonksiyonu kullanılmıştır.
- Ön-eğitim aşamasında 200 tur boyunca gözetimsiz olarak eğitilmiştir.
- İnce-ayar aşamasında 3000 tur boyunca gözetimli olarak eğitilmiştir.
- Ön-eğitim ve ince-ayar aşaması için α=0.1 olarak belirlenmiştir.

Gürültü Giderici Otomatik Kodlayıcı Tabanlı Derin Yapay Sinir Ağı (GGOK - DSA) :

- 121 nöronluk giriş katmanı, her biri 150 nörondan oluşan üç gizli katman ve iki nöronluk bir çıkış katmanından oluşmaktadır.
- Ön-eğitim aşaması için GGOK kullanılmıştır.
- Aktivasyon fonksiyonu olarak, gizli katmanlarda sigmoid, çıkış katmanında ise softmax fonksiyonu kullanılmıştır.
- Ön-eğitim aşamasında 200 tur boyunca gözetimsiz olarak eğitilmiştir.
- İnce-ayar aşamasında 3000 tur boyunca gözetimli olarak eğitilmiştir.
- Ön-eğitim ve ince-ayar aşaması için α=0.1 olarak belirlenmiştir.
- Bozulma oranı 1., 2. ve 3. Katmanlardaki bozulma oranları sırasıyla %10,%20,%30 olarak belirlenmiştir.

4.2.3.2 Önerilen Anomali Tespit Modelleri

Stokastik Eşik Değeri Belirleme Yöntemine Dayalı Otomatik Kodlayıcı Tabanlı Anomali Tespit Modeli(OK - SAEDT):

- 121 nöronluk giriş katmanı, 30 nöronluk bir gizli katman ve 121 nöronluk bir çıkış katmanından oluşmaktadır.
- Aktivasyon fonksiyonu olarak, gizli katmanlarda sigmoid fonksiyonu kullanılmıştır.
- YOH değerini minimuma indirgemek için 3000 tur boyunca yarı gözetimli olarak eğitilmiştir.
- α=0.01 olarak belirlenmiştir.

Stokastik Eşik Değeri Belirleme Yöntemine Dayalı Gürültü Giderici Otomatik Kodlayıcı Tabanlı Anomali Tespit Modeli (GGOK - SAEDT) ;

- 121 nöronluk giriş katmanı, 60 nöronluk bir gizli katman ve 121 nöronluk bir çıkış katmanından oluşmaktadır.
- Aktivasyon fonksiyonu olarak, gizli katmanlarda sigmoid fonksiyonu kullanılmıştır.

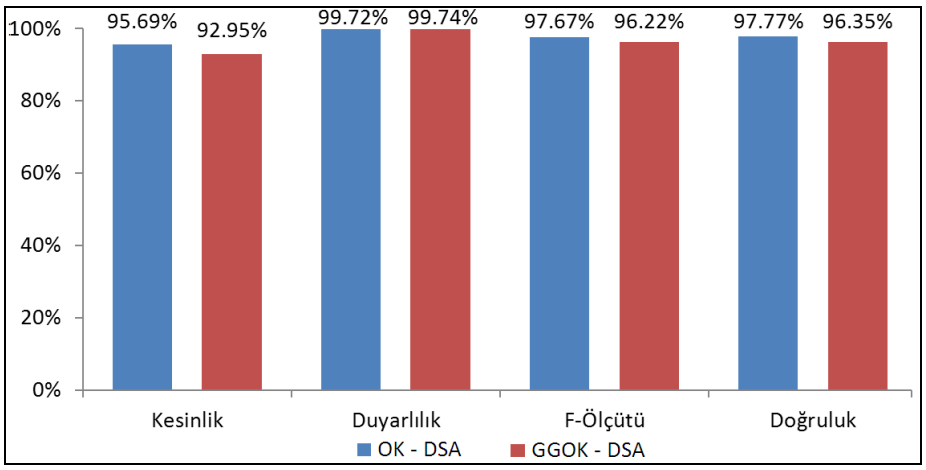
- YOH değerini minimuma indirgemek için 2500 tur boyunca yarı gözetimli olarak eğitilmiştir.
- Bozulma oranı %10 ve α=0.01 olarak belirlenmiştir.

4.2.4 Deneysel Sonuçlar

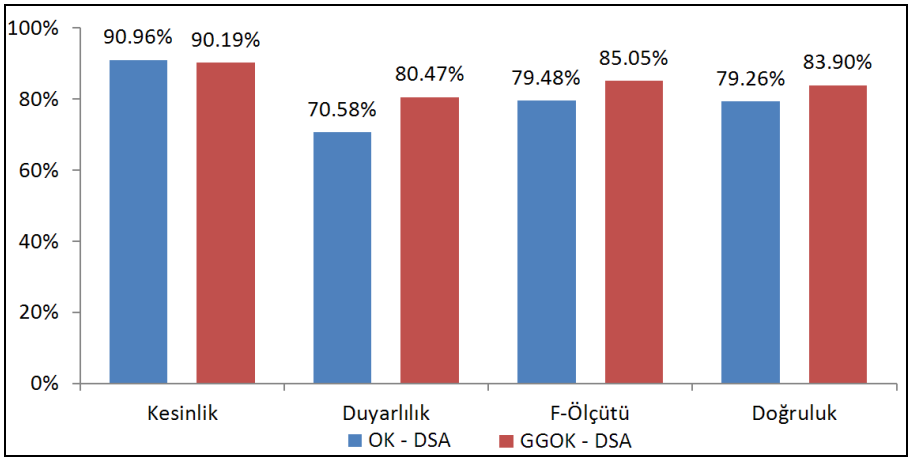
Bu bölümde tez kapsamında gerçeklenmiş olan mevcut ve önerilen yöntemlerin NSL-KDD kümesindeki anomali tespit performanslarına ilişkin sonuçlar; doğruluk, kesinlik, duyarlılık ve F-ölçütü metrikleri bazında verilmiştir.

OK - DSA ve GGOK – DSA yöntemleri KDDTrain+_%20 veri kümesi kullanılarak eğitilmiş ve bu yöntemlerinin değerlendirilmesi için 10 katmanlı çapraz doğrulama yöntemi kullanılmıştır. Ek olarak, modellerin sıfırıncı gün saldırının tespiti konusundaki başarımlarının ölçülebilmesi adına 10 kat doğrulamanın her bir turunda eğitilen model KDDTest+ üzerinde de test edilmiştir. Modelin KDDTest+ üzerindeki genel başarımı, bu 10 sınıflandırıcıdan elde edilen sonuçların ortalaması olarak kabul edilmiştir.

Şekil 4. ’de OK-DSA ve GGOK-DSA modellerinin KDDTrain+_%20 üzerindeki anomali tespit performanslarına bakıldığında OK-DSA yönteminin F-ölçütü ve doğruluk metrikleri bakımından GGOK-DSA’dan çok az farkla daha iyi olduğu görülmektedir.



Şekil 4. 1 OK - DSA ve GGOK – DSA yöntemlerinin performanslarının KDDTrain+_%20 üzerinde 10-fold çapraz doğrulama tekniği kullanılarak karşılaştırılması

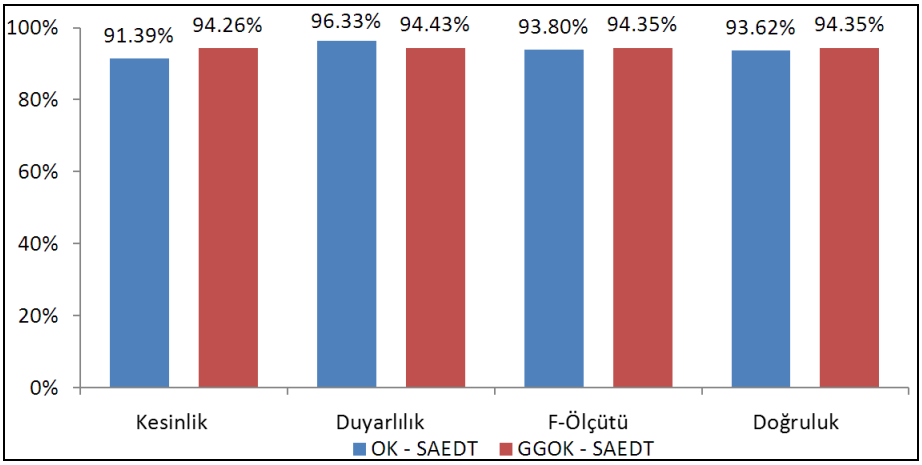


Şekil 4. 2 DSA ve GGOK – DSA yöntemlerinin performanslarının KDDTest+ üzerinden karşılaştırılması

Ancak bu iki yöntemin Şekil 4. 2 'de KDDTest+ üzerindeki test sonuçlarına bakıldığında ise GGOK-DSA'nın, OK-DSA'ya göre çok daha yüksek doğruluk ve F-ölçütü yüzdeleri ile anomali tespiti yaptığı görülmektedir. Bu yöntemlerin KDDTrain+_%20 üzerindeki performanslarının da bir birine çok yakın olduğu göz önünde bulundurulduğunda GGOK-DSA'nın OK-DSA'ya göre genel anomali tespit başarımının çok daha yüksek olduğu sonucuna varılmıştır.

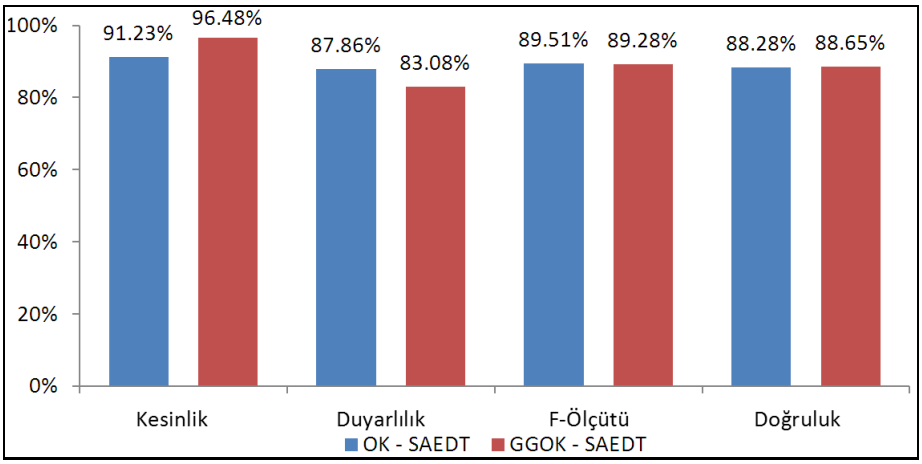
Önerilen yöntemlerden OK-SAEDT ve GGOK-SAEDT modelleri, stokastik eşik değeri belirleme yöntemine dayandıklarından rastgele karıştırılmış veri kümeleri üzerinden; eşik değerleri belirlenmiş ve test edilmiştirler. Bu yöntemler için değerlendirme işlemleri 100 tur boyunca tekrarlanmıştır. Modellerin test sonuçlarına ilişkin verilen değerler tüm turlar boyunca elde edilen sonuçların ortalamasıdır.

GGOK-SAEDT ve OK-SAEDT modellerinin, KDDTrain+_%20'dan ayrılmış olan test veri kümesi üzerindeki anomali tespit performanslarına ilişkin sonuçlar Şekil 4. 3'te verilmiştir. Burada iki yöntemin de birbirlerine çok yakın performans gösterdikleri ancak GGOK-SAEDT yönteminin OK-SAEDT yöntemine kıyasla biraz daha yüksek başarımla tespit yaptığı görülmektedir.



Şekil 4. 3 Önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerinin test%20 üzerindeki başarımlarının kıyaslaması

Bunun yanı sıra Şekil 4. 4'te önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerin KDDTest+ veri kümesi üzerindeki anomali tespit performanslarına ilişkin verilen sonuçlarda bu yöntemlerin KDDTest+ veri kümesi üzerinde de birbirlerine çok yakın performans gösterdikleri görülmektedir. Ancak burada da yine GGOK-SAEDT yönteminin OK-SAEDT yöntemine göre doğruluk metriği bazında çok az farkla daha yüksek başarım ile tespit yaptığı gözlemlenmiştir.



Şekil 4. 4 Önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerinin KDDTest+ üzerindeki başarımlarının kıyaslaması

Şekil 4. 1 ve Şekil 4. 4'deki sonuçlara bakıldığında OK-DSA ve GGOK-DSA yöntemlerinin KDTrain+_%20 üzerindeki testlerde, önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerine göre daha yüksek doğruluk oranlarına sahip oldukları görülmektedir. OK-DSA ve GGOK-DSA yöntemleri gözetimli olarak eğitilen sınıflandırıcılar olduklarından eğitim kümesi ile aynı tip saldırıları içeren bir test veri kümesi üzerinde test edildiklerinde, yarı gözetimli

olarak eğitilmiş olan önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerine göre daha başarılı sonuçlar üretmektedirler.

Çizelge 4. 3 Gerçeklenen Modellerin Eğitim Sürelerinin Karşılaştırılması

Yöntem	Eğitim Süresi (Dakika)
OK-DSA	4,97
GGOK-DSA	9,34
OK-SAEDT	2,79
GGOK-SAEDT	2,35

Çizelge 4. 3 gerçeklenen modellerin eğitimi için gerekli olan süreler dakika cinsinden karşılaştırmalı olarak verilmiştir. Buna göre gerçeklenen yöntemler arasındaki en başarılı yöntem olan GGOK-SAEDT yönteminin diğer yöntemlere kıyasla daha kısa sürede eğitilebildiği tespit edilmiştir.

Çizelge 4. 4'te tez kapsamında gerçeklenen OK-DSA, GGOK-DSA, OK-SAEDT ve GGOK-SAEDT yöntemlerinin birbirleri ile ve literatürdeki en başarılı yöntemlerle doğruluk metriği bazında KDDTest+ üzerindeki karşılaştırmalı sonuçları verilmiştir. OK-DSA yöntemi ile GGOK-DSA yöntemlerinin MLP yöntemine göre daha yüksek başarıma sahip olduğu görülmektedir. Bunun yanı sıra, önerilen OK-SAEDT ve GGOK-SAEDT yöntemlerinin KDDTest+ üzerinde OK-DSA ve GGOK-DSA yönteminden çok daha yüksek başarımla tespit yapabildikleri görülmektedir. Bu sonuç, yarı gözetimli olarak eğitilmiş olan önerilen yöntemlerin, eğitim sürecinde hiç karşılaşılmamış olan saldırı tiplerini içeren KDDTest+ veri kümesindeki anomalilerin tespiti konusunda, gözetimli olarak eğitilmiş sınıflandırıcılar olan; OK-DSA ve GGOK-DSA'ya göre çok daha başarılı olduklarını göstermektedir. Ayrıca, önerilen GGOK-SAEDT yönteminin literatürdeki diğer tüm tekil yöntemlerden daha başarılı olduğu ve en yüksek başarıma sahip olan Random Tree - NB Tree yöntemine de çok yakın bir başarım ile tespit yapabildiği görülmektedir.

Çizelge 4. 4 KDDTest+ üzerinde en yüksek başarıma sahip olan diğer yöntemlerle karşılaştırmalı sonuçlar

Yöntem	Türü	Doğruluk Oranı
MLP[16]	Tekil	% 77,41
OK-DSA(Gerçekelenen Yöntem)	Hibrit	% 79,25
NB Tree[16]	Tekil	% 82,02
Fuzzy Classifier[19]	Tekil	% 82,74
GGOK-DSA(Gerçekelenen Yöntem)	Hibrit	% 83,39
SAE- MLP[24]	Hibrit	% 88,39
OK-SAEDT (Önerilen Yöntem)	Tekil	% 88,28
Random Tree[20]	Tekil	% 88,46
GGOK-SAEDT (Önerilen Yöntem)	Tekil	% 88,65
Random Tree - NB Tree[20]	Hibrit	% 89,24

BÖLÜM 5

SONUÇ VE ÖNERİLER

Bu tez kapsamında, derin öğrenme yöntemleri kullanılarak ağ anomali tespiti konusundaki başarımın arttırılması hedeflenmiştir. Başarımın arttırılabilmesi adına ilk olarak mevcut derin öğrenme yöntemlerinden OK-DSA ve GGOK-DSA yöntemlerinin başarımları NSL-KDD veri kümesi üzerinde ölçülmüştür. Sonrasında anomali eşik değerinin belirlenmesi için stokastik bir anomali eşik değeri tespit (SAEDT) yöntemi önerilmiş ve önerilen eşik değeri belirleme yöntemini kullanan deterministik otomatik kodlayıcı tabanlı modeller ile NSL-KDD veri kümesi üzerinde anomali tespiti yapılmıştır.

Modellerin test edildiği KDDTest+ veri kümesinin, modellerin eğitildiği KDDTrain+_%20 veri kümesindeki saldırı tiplerine ek olarak farklı saldırı tiplerini de içermesi, önerilen yöntemin daha önceden hiç karşılaşmadığı saldırıları (sıfırıncı gün saldırıları) tespit etme konusundaki başarımının ölçülebilmesine imkan tanımıştır. Önerilen GGOK-SAEDT ve OK-SAEDT yöntemlerinin KDDTest+ üzerindeki testlerde %88,65 ve %88,28 doğruluk oranları ile birbirlerine çok yakın performans gösterdikleri tespit edilmiştir. Ek olarak, gözetimli eğitime dayalı sınıflandırıcılar olan GGOK-DSA ve OK-DSA yöntemlerinin bilinen saldırıların tespiti konusunda önerilen yarı gözetimli eğitime dayalı GGOK-SAEDT ve OK-SAEDT yöntemlerine kıyasla daha başarılı oldukları ancak sıfırıncı gün saldırılarında ise çok daha düşük başarım gösterdikleri tespit edilmiştir.

Önerilen GGOK-DSA yönteminin KDDTest+ üzerinde güncel literatürdeki tüm tekil yöntemlerden daha başarılı olduğu ve %89,24 doğruluk oranı ile en başarılı hibrit yöntem olan Random Tree - NB Tree yöntemine de çok yakın bir performans gösterdiği tespit edilmiştir. Bunun sonucunda uygun eşik değeri belirlendiğinde, otomatik

kodlayıcıların tekil birer yöntem olarak anomali tespiti konusunda en başarılı olan hibrit yöntemlere alternatif olabilecekleri saptanmıştır.

KAYNAKLAR

[1] Tsai, C.F., Hsu, Y.F., Lin, C.Y. ve Lin, W.Y., (2009). "Intrusion detection by machine learning: A review", Expert Systems with Applications, 36 (10): 11994–12000.

[2] Leung, K. ve Leckie, C., (2005). "Unsupervised anomaly detection in network intrusion detection using clusters", Proceedings of the Twenty-eighth Australasian conference on Computer Science, Ocak 2005, Newcastle, Australia 333–342.

[3] Ahmed, M., Mahmood, A.N. ve Hu, J., (2016). "A survey of network anomaly detection techniques", Journal of Network and Computer Applications, 60: 19–31.

[4] Gogoi, P., Bhattacharyya, D.K., Borah, B. ve Kalita, J.K., (2011). "A Survey of Outlier Detection Methods in Network Anomaly Identification ", The Computer Journal, 54 (4): 570–588.

[5] Dau, H.A., Ciesielski, V. ve Song, A., (2014). "Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class", Simulated Evolution and Learning: 10th International Conference, SEAL 2014, 15-18 Aralık 2014, Dunedin, New Zealand, 311—322.

[6] Eskin, E., (2000). "Anomaly detection over noisy data using learned probability distributions", Proceedings of the 17th International Conference on Machine Learning, 29 Haziran-2 Temmuz 2000, San Mateo, CA, 255–262.

[7] Shyu, M., Chen, S. , Sarinnapakorn, K. ve Chang, L., (2003). "A novel anomaly detection scheme based on principal component classifier", Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM03), 19-22 Kasım 2003, Melbourne, Florida, USA, 172–179.

[8] Thottan, M. ve Ji, C., (2003). "Anomaly Detection in IP Networks", IEEE Trans. Signal Processing, 51 (8): 2191-2204.

[9] Syarif, I., Prugel-Bennett, A. ve Wills, G., (2012). "Unsupervised Clustering Approach for Network Anomaly Detection", Fourth International Conference on Networked Digital Technologies (NDT 2012), 29 Mart-2 Nisan 2012, Dubai, AE, 24–26.

[10] Aygün, R.C. ve Yavuz A. G., (2017).,"A Stochastic Data Discrimination Based Autoencoder Approach For Network Anomaly Detection", 25th Signal Processing and Communications Applications Conference, 15-18, Mayıs 2017, Antalya, Turkey.“(baskıda).”.

[11] Deng, L., (2014). "A tutorial survey of architectures, algorithms, and applications for deep learning", APSIPA Transactions on Signal and Information Processing,

[12] Caida DataSet, http://www.caida.org, 28 Ocak 2017.

[13] DEFCON CTF PCAPs from DEF CON 17 to 24, https://media.defcon.org/, 28 Ocak 2017.

[14] KDDCUP’99 Data Set, https://kdd.ics.uci.edu/databases/kddcup99/task.htmls, 28 Ocak 2017.

[15] NSL-KDD Dataset, http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html, 25 Kasım 2016.

[16] Tavallaee, M., Bagheri, E., Lu, W. ve Ghorbani, A.A., (2009). "A detailed analysis of the KDD CUP 99 data set", IEEE International Conference on Computational Intelligence for Security and Defense Applications, Ottawa, 8-10 Temmuz 2009, Ottawa, ON, Canada, 53-58.

[17] ISCX UNB Dataset, http://www.unb.ca/cic/research/datasets/ids.html, 28 Ocak 2017.

[18] Bhuyan, M.H., Bhattacharyya, D.K. ve Kalita, J.K., (2014). "Network anomaly detection: Methods, systems and tools", IEEE Commun. Surveys Tuts, 16(1): 303-336.

[19] Kromer, P., Platos, J., Snasel, V. ve Abraham, A., (2011). "Fuzzy classification by evolutionary algorithms", IEEE International Conference on Systems, Man, and Cybernetics. Anchorage, 9-12 Ekim 2011, AK, USA, 313–318.

[20] Kevric, J., Jukic, S. ve Subasi, A., (2016). "An effective combining classifier approach using tree algorithms for network intrusion detection", Neural Computing and Applications: 1-8.

[21] Hawkins, S., He, H.X., Williams, G.J. ve Baxter, R.A., (2002). "Outlier detection using replicator neural networks", Proc. of the Fifth Int. Conf. and Data Warehousing and Knowledge Discovery (DaWaK02), 4-6 Eylül 2002. Aix-en-Provence, France.

[22] Nicolau, M. ve McDermott, J., (2016). "A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection", International Conference on Parallel Problem Solving from Nature, 17-21 Eylül 2016, Edinburgh, United Kingdom.

[23] Sakurada, M. ve Yairi, T., (2014). "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction", Proc. of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA), 1-5 Aralık 2014, Gold Coast, Australias, 4–11.

[24] Niyaz, Q., Sun, W., Javaid, A.Y. ve Alam, M., (2015). "A Deep Learning Approach for Network Intrusion Detection System", Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 3-5 Aralık 2015, New York, NY, USA.

[25] Bengio, Y., (2009). "Learning Deep Architectures for AI", Foundations and Trends in Machine Learning, 2 (1): 1-127.

[26] Chandola, V. Banerjee, A. ve Kumar, V., (2007). "Anomaly Detection: A Survey, Technical Report", ACM Computing Surveys (CSUR), 41(3) : 1-58.

[27] García-Teodoroa, P., Díaz-Verdejoa, J., Maciá-Fernándeza, G. ve Vázquez, E., (2009). "Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges", Computers & Security, 28 (1): 18-28.

[28] Buczak, A.L. ve Guven, E., (2016). "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", IEEE Communications Surveys & Tutorials, 18 (2): 1153-1176.

[29] Panda, M., Abraham, A. ve Patra, M.R., (2011). "A Hybrid Intelligent Approach for Network Intrusion Detection", International Conference on Communication Technology and System Design, 7-9 Aralık 2011, Coimbatore India

[30] Heckerman, D., (2008). "A Tutorial on Learning With Bayesian Networks", Innovations in Bayesian Networks: Theory and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, 33-82

[31] Kruegel, C., Mutz, D. Robertson, W. ve Valeur, F., (2003). "Bayesian event classification for intrusion detection",Proceedings of 19th annual computer security applications conference, 8-12 Aralık 2003, Las Vegas, USA, 14-23.

[32] LISA lab University Of Montreal, Deep Learning Tutorial (2015), http://deeplearning.net/tutorial/deeplearning.pdf, 28 Ocak 2017.

[33] Poojitha, G., Kumar, K. ve Reddy, P., (2010). "Intrusion detection using artificial neural network", 2010 International conference on computing communication and networking technologies (ICCCNT), 29-31 Temmuz 2010, Karur, India, 1-7.

[34] Balajinath, B. ve Raghavan, S.V., (2001). "Intrusion detection through learning behavior model", Computer Communications, 24(12): 1202–1212.

[35] Khan, M.S.A., (2011). "Rule based Network Intrusion Detection using Genetic Algorithm", International J. Computer Applications, 18(8): 26–29.

[36] Balabine, I. ve Velednitsky, A., (2015). "A. Method and system for confident anomaly detection in computer network traffic", Google Patents.

[37] Salama, M., Eid, H., Ramadan, R., Darwish, A. ve Hassanien, A., (2011). "Hybrid intelligent intrusion detection scheme", Soft Computing in Industrial Applications, 96: 293–303.

[38] Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J. ve Williamson, R.C., (2001). "Estimating the Support of a High-Dimensional Distribution", Neural Computation, 13(7): 1443–1471.

[39] Abbes, T., Bouhoula, A. ve Rusinowitch, M., (2010). "Efficient decision tree for protocol analysis in intrusion detection", International J. Security and Networks, 5 (4): 220–235.

[40] Liao, Y. ve Vemuri, V.R., (2002). "Use of K-nearest neighbor classifier for intrusion detection", Computers & Security, 21 (5): 439–448.

[41] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. ve Stolfo, S., (2002). "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data", Applications of Data Mining in Computer Security, 6: 77-101.

[42] Zhuang, Z., Li, Y. ve Chen, Z., (2010). "Enhancing Intrusion Detection System with proximity information", International J. Security and Networks, 5 (4): 207-219.

[43] Münz, G., Li, S. ve Carle, G., (2007). "Traffic Anomaly Detection Using K-Means Clustering", 4. GI/ITG-Workshop MMBnet 2007, 13-14 Eylül 2007, Hamburg, Germany.

[44] Syarif, I., Zaluska, E., Prugel-Bennett, A. ve Wills, G., (2012). "Application of Bagging, Boosting and Stacking to Intrusion Detection", Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012,15-20 Temmuz 2012, Berlin, Germany, 593-602.

[45] Wang, K. ve Stolfo, S.J., (2004). "Anomalous Payload-Based Network Intrusion Detection", Recent Advances in Intrusion Detection. RAID 2004. Lecture Notes in Computer Science: 203–222.

[46] adfa ld12 DataSet, https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets, 28 Ocak 2017.

[47] Quoc, V.L., A Tutorial on Deep Learning Part 1: Nonlinear Classifiers and The Backpropagation Algorithm, http://robotics.stanford.edu/quocle/tutorial1.pdf, 01 Mart 2017.

[48] Schmidhuber, J., (2015). "Deep learning in neural networks: An overview", Neural Networks, 61: 85-117.

[49] LeCun, Y., Bengio, Y. ve Hinton, G., (2015). "Deep learning", Nature, 521: 436–444.

[50] Arel, I., Rose, D.C. ve Karnowski, T.P., (2010). "Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]", IEEE Computational Intelligence Magazine, 5 (4): 13-18.

[51] Bengio, Y., Courville, A. ve Vincent, P., (2013). "Representation learning: A review and new perspectives", IEEE Trans. Pattern Anal. Mach. Intell., 3 (8): 1798–1828.

[52] Hinton, G.E. ve Salakhutdinov, R.R., (2006). "Reducing the dimensionality of data with neural network", science, 313(5786): 504-507.

[53] Bengio, Y. ve Glorot, X., (2010). "Understanding the difficulty of training deep feedforward neuralnetworks", Sardinia, Italy: citeseerx.

[54] LeCun, Y.A., Bottou, L., Orr, G.B. ve Müller, K.R., (2010). "Efficient backprop", Neural networks: Tricks of the trade, Springer Berlin Heidelberg: 9-48.

[55] Ng, A., Sparse autoencoder, university CS294ALecturenotes (2011), https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf, Stanford University, 01 Mart 2017.

[56] Bengio, Y., Lamblin, P., Popovici, D. ve Larochelle, H., (2007). "Greedy Layer-Wise Training of Deep Networks", Proceedings of the 19th International Conference on Neural Information Processing Systems , 12-15 Kasım 2007, Doha, Qatar.

[57] Vincent, P., Larochelle, H., Bengio, Y. ve Manzagol, P.A., (2008). "Extracting and Composing Robust Features with Denoising Autoencoders", Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08), 5-9 Temmuz 2008, New York, NY, USA, 1096 - 1103.

[58] Bastien, F., Lamblin, P. ve diğerleri, v., (2012). "Theano: new features and speed improvements.", arXiv preprint arXiv:1211.5590: 9-48.

[59] Bergstra, J. Breuleux, O. , vd, (2010). Theano: A CPU and GPU math compiler in Python, Proceedings of the 9th Python in Science Conference (SciPy 2010), 28 Haziran-3 Temmuz 2010, Austin, 1-7.

[60] Nickolls, J., Buck, I., Garland, M. ve Skadron, K., (2008). "scalable parallel programming with cuda", Queue - GPU Computing, 6: 40-53.

[61] Z.Ueng, S., Lathara, M., Baghsorkhi, S.S. ve Wen-mei, W.H., (2010). "CUDA-lite: Reducing GPU programming complexity", International Workshop on Languages and Compilers for Parallel Computing, 7-9 Ekim 2010, Edmonton, AB, Canada, 1-15.

[62] Wang, W., Zhang, X., Gombault, S. ve Knapskog, S., (2009). Attribute normalization in network intrusion detection, Pervasive systems, algorithms, and networks (ISPAN), 13-16 Aralık 2009, Hong Kong, China, 448-453.

EK-A

NSL-KDD VERİ KÜMESİ ÖZELLİK TABLOSU

A-1 KDDTrain+_%20 Veri Kümesine Ait Özellikler

	Özellik	Değer		Özellik	Değer
1	duration	Nümerik [0,42862]	22	is_guest_login	{0, 1}
2	protocol_type	{'tcp','udp', 'icmp'}	23	count	Nümerik [1,511]
3	service	{'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'http_2784', 'http_443', 'http_8001', 'imap4', 'IRC', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois', 'X11', 'Z39_50'}	24	srv_count	Nümerik [1,511]
4	flag	{ 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' }	25	serror_rate	Nümerik [0,1]
5	src_bytes	Nümerik [0,381709090]	26	srv_serror_rate	Nümerik [0,1]
6	dst_bytes	Nümerik [0,5151385]	27	rerror_rate	Nümerik [0,1]
7	land	{0, 1}	28	srv_rerror_rate	Nümerik [0,1]
8	wrong_fragment	Nümerik [0,3]	29	same_srv_rate	Nümerik [0,1]
9	urgent	Nümerik [0,1]	30	diff_srv_rate	Nümerik [0,1]
10	hot	Nümerik [0,77]	31	srv_diff_host_rate	Nümerik [0,1]
11	num_failed_logins	Nümerik [0,4]	32	dst_host_count	Nümerik [0,255]
12	logged_in	{0, 1}	33	dst_host_srv_count	Nümerik [0,255]
13	num_compromised	Nümerik [0,884]	34	dst_host_same_srv_rate	Nümerik [0,1]
14	root_shell	Nümerik [0,1]	35	dst_host_diff_srv_rate	Nümerik [0,1]
15	su_attempted	Nümerik [0,2]	36	dst_host_same_src_port_rate	Nümerik [0,1]
16	num_root	Nümerik [0,975]	37	dst_host_srv_diff_host_rate	Nümerik [0,1]
17	num_file_creations	Nümerik [0,40]	38	dst_host_serror_rate	Nümerik [0,1]
18	num_shells	Nümerik [0,1]	39	dst_host_srv_serror_rate	Nümerik [0,1]
19	num_access_files	Nümerik [0,8]	40	dst_host_rerror_rate	Nümerik [0,1]
20	num_outbound_cmds	Nümerik [0,0]	41	dst_host_srv_rerror_rate	Nümerik [0,1]
21	is_host_login	{0, 1}	42	class	{'normal', 'anomaly'}

Şekil A. 1 KDDTrain+_%20'ye ait özelliklerin veri tipleri ve değer aralıkları

A-2 KDDTest+ Veri Kümesine Ait Özellikler

	Özellik	Değer		Özellik	Değer
1	duration	Nümerik [0,57715]	22	is_guest_login	{0, 1}
2	protocol_type	{'tcp','udp', 'icmp'}	23	count	Nümerik [0,511]
3	service	{'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'http_2784', 'http_443', 'http_8001', 'imap4', 'IRC', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois', 'X11', 'Z39_50'}	24	srv_count	Nümerik [0,511]
4	flag	{ 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' }	25	serror_rate	Nümerik [0,1]
5	src_bytes	Nümerik [0,62825648]	26	srv_serror_rate	Nümerik [0,1]
6	dst_bytes	Nümerik [0,1345927]	27	rerror_rate	Nümerik [0,1]
7	land	{0, 1}	28	srv_rerror_rate	Nümerik [0,1]
8	wrong_fragment	Nümerik [0,3]	29	same_srv_rate	Nümerik [0,1]
9	urgent	Nümerik [0,3]	30	diff_srv_rate	Nümerik [0,1]
10	hot	Nümerik [0,101]	31	srv_diff_host_rate	Nümerik [0,1]
11	num_failed_logins	Nümerik [0,4]	32	dst_host_count	Nümerik [0,255]
12	logged_in	{0, 1}	33	dst_host_srv_count	Nümerik [0,255]
13	num_compromised	Nümerik [0,796]	34	dst_host_same_srv_rate	Nümerik [0,1]
14	root_shell	Nümerik [0,1]	35	dst_host_diff_srv_rate	Nümerik [0,1]
15	su_attempted	Nümerik [0,2]	36	dst_host_same_src_port_rate	Nümerik [0,1]
16	num_root	Nümerik [0,878]	37	dst_host_srv_diff_host_rate	Nümerik [0,1]
17	num_file_creations	Nümerik [0,100]	38	dst_host_serror_rate	Nümerik [0,1]
18	num_shells	Nümerik [0,5]	39	dst_host_srv_serror_rate	Nümerik [0,1]
19	num_access_files	Nümerik [0,4]	40	dst_host_rerror_rate	Nümerik [0,1]
20	num_outbound_cmds	Nümerik [0,0]	41	dst_host_srv_rerror_rate	Nümerik [0,1]
21	is_host_login	{0, 1}	42	class	{normal, anomaly}

Şekil A. 2 KDDTest+'ya ait özelliklerin veri tipleri ve değer aralıkları

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	:Rüstem Can AYGÜN
Doğum Tarihi ve Yeri	:20.11.1989 / Bakırköy - İstanbul
Yabancı Dili	:İngilizce
E-posta	:canaygun10@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Müh.	İstanbul Üniversitesi	2012
Lise	Fen Bilimleri	Şehremini Lisesi	2006

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2016-Devam Ed.	Yıldız Teknik Üniversitesi/Bilgisayar Müh.	Araştırma Görevlisi
2014-2016	Yapı Kredi Bankası / BT.	Uzman Yazılım Geliştirici
2012-2014	Yapı Kredi Bankası / BT.	Yazılım Geliştirici

YAYINLARI

Bildiri

[1] Aygün, R.C. ve Yavuz A. G., (2017).,"A Stochastic Data Discrimination Based Autoencoder Approach For Network Anomaly Detection", 25th Signal Processing and Communications Applications Conference, 15-18, Mayıs 2017, Antalya, Turkey.“(baskıda).”.

[2] Aygun, R.C. ve Yavuz A. G., (2017).,"Network Anomaly Detection with Stochastically Improved Autoencoder Based Models", The 4th IEEE International Conference on Cyber Security and Cloud Computing, 26-28, Haziran 2017, New York, USA.“(baskıda).”

Proje

[1] Yapı Kredi Bankası/BT, " Harmoni - Vadeli Mevduat Altyapı Altyapı Dönüşüm Projesi", Yazılım Geliştirici, (2012-2014).

[2] Yapı Kredi Bankası/BT, "Vadeli Mevduat Tutar Aşımı Hazine Referans Talep Projesi",Uzman Yazılım Geliştirici, (2013-2014).

[3] Yapı Kredi Bankası/BT, "YKB Front-End Dönüşüm Projesi",Uzman Yazılım Geliştirici, (2014-2015).

[4] Yapı Kredi Bankası/BT, "Vadeli Mevduat Bire Bir Fiyatlama Projesi", Yazılım Ekibi Lideri, (2015).

[5] Yapı Kredi Bankası/BT, "Çeyiz ve Konut Mevduatı (Birikimli Mevduat) Projesi",Yazılım Ekibi Lideri (YKB/Vadeli Mevduat Grubu Yazılım Mimarı), (2015-2016).

[6] YÜLAP 2016-04-01-YL01, “Derin Öğrenme Yöntemleri ile Bilgisayar Ağlarında Güvenliğe Yönelik Anormallik Tespiti”, Araştırmacı, (2017-Devam ediyor).