ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

ÇATIŞMA YÖNELİMLİ MAKİNE ÖĞRENME ÜZERİNE İSTATİSTİKSEL BİR ÇALIŞMA

Deniz DÜDÜKCÜ

İSTATİSTİK ANABİLİM DALI

ANKARA
2023

ÖZET

Yüksek Lisans Tezi

ÇATIŞMA YÖNELİMLİ MAKİNE ÖĞRENME ÜZERİNE İSTATİSTİKSEL BİR ÇALIŞMA

Deniz DÜDÜKCÜ

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
İstatistik Anabilim Dalı

Danışman: Doç. Dr. İhsan KARABULUT

Bu çalışmada makine öğrenmede yaklaşımlar, kullanılan yöntem ve temel kavramların bir kısmı gözden geçirilmiştir. Kullanılan algoritmalar özetlenerek örneklerle verilmiştir. Makine öğrenme ile istatistik disiplinin kullandığı yol, yöntem ve değerlendirme anlayışları kullandıkları terminoloji temelinde kısaca karşılaştırılmıştır. Gözlem altında öğrenmede yapay sinir ağı yönteminin kullanılışı öne çıkarılmıştır.

Çatışmalı makine öğrenme, örüntü ve şekil tanımada yaygınca kullanılan el yazısı rakamları veri kümesi MNIST üzerinde uygulanarak ele alınmıştır.

El yazısı rakamlarını tanıma konusu; beyaz kutu ortamında, hedefsiz, kaçınma türünde saldırı çatışmalı makine uygulamasına örnek olarak seçilmiştir. Bu örnekte, öğrenmiş sisteme yanlış sınıflandırma yaptırarak, makine öğrenimi yanıltılmaya çalışılmıştır. Saldırı başarısı makine ile insan algı ve estetiğinin karşılaştırılmalarıyla ve saldırı gerçekleştirme hızı bakımlarından değerlendirilmiştir. Saldırı algoritması PSNR (Peak Signal Noise Ratio - tepe sinyal gürültü oranı) ölçütüyle ve Carlini ve Wagner (2017)'de önerilen tanjant hiperbolik sınıflandırma fonksiyonunun Maclaurin yaklaşımı kullanılarak oluşturulmuştur. Yeni yöntem, Carlini ve Wagner saldırı yönteminden %25.53 daha hızlı çalışmış ancak tahminlerde yanıltma oranında %11.6 düşüş gözlenmiştir.

Ocak 2023, 67 sayfa

Anahtar Kelimeler: İstatistiksel karar verme, sağlam istatistik, kayıp fonksiyonu,öğrenme algoritması, veri ve görüntü işleme

ABSTRACT

MSc Thesis

A STATISTICAL STUDY ON ADVERSARIAL MACHINE LEARNING

Deniz DÜDÜKCÜ

Ankara University
Graduate School of Natural and Applied Sciences
Department of Statistics

Supervisor: Assoc. Prof. Dr. İhsan KARABULUT

In this thesis, some of the methods and basic concepts of machine learning are reviewed. Some of the algorithms are summarized with given examples. The approaches, methods and evaluation processes of machine learning and statistics discipline are briefly compared on the basis of the terminology they use. Use of artificial neural network method in supervised learning has been highlighted.

Adversarial machine learning is studied by using handwritten numbers in MNIST dataset which are widely used in pattern and shape recognition problems.

The recognition of handwritten numbers task is chosen as an example of adversarial machine learning application by considering white box, no-target, evasion type attack environment. In this example, machine learning was tried to be mislead by making the learned system misclassify. Attack success was evaluated by comparing machine and human perception and aesthetics, and in terms of attack execution speed. The attack algorithm was created using the PSNR (Peak Signal Noise Ratio) criterion and the Maclaurin approach of the tangent hyperbolic classification function proposed in Carlini & Wagner (2017). The new method worked 25.53% faster than the Carlini & Wagner attack method, but the prediction error rate decreased by 11.6%.

January 2023, 67 pages

Key Words: Statistical decision, robust statistics, loss function, learning algorithm, data and image processing

TEŞEKKÜR

Bilgi ve tecrübeleriyle bu tez çalışmasının ortaya çıkmasında büyük katkıları ve emeği olan sayın Doç. Dr. İhsan KARABULUT'a (Ankara Üniversitesi İstatistik Bölümü); çalışmalarım süresince her türlü desteği ve fedakarlığı esirgemeyen aileme en derin duygularla teşekkür ederim.

Deniz DÜDÜKCÜ
Ankara, Ocak 2023

İÇİNDEKİLER

SİMGELER DİZİNİ

$ln(x)$	$x > 0$ olmak üzere doğal logaritma
$log(x)$	$x > 0$ olmak üzere 10 tabanlı logaritma
$loss(.)$	Kayıp fonksiyonu
$relu$	Değeri girdisini oluşturan bağımsız değişken x ile 0 arasında maksimumunu bulan fonksiyon $max(x, 0)$
$sgn(.)$	İşaret fonksiyonu
$softplus(x)$	Değeri $log(1 + e^x)$ 'e eşit fonksiyon
$clip(.)$	Girdisini oluşturan bir resim x'in çözünürlü değerlerine göre belirlenen miktarlarda, piksel aralıklarını yok eden fonksiyon

Kısaltmalar

KiB	Kilobayt
png	(Portable Network Graphics) kayıpsız sıkıştırma ısal görüntü saklama biçimi
jpeg	(Joint Photographic Experts Group) kayıplı sıkıştırma kullanan sayısal görüntü saklama biçimi
RGB	Red Green Blue (Kırmızı Yeşil Mav)
RAM	Random Access Memory (Rasgele Erişimli Bellek)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Makine öğrenme; bilgisayarların verilerden elde ettiği "tecrübelerle" kendini geliştirip uygulanan probleme daha iyi tahmin ve kestirimler vermesini sağlayan, benzer problemleri sonradan cevaplamak üzere genelleyici kapasitesi olan algoritmaları çatısı altında toplayan bilgisayar bilimleri uygulamasıdır. Günümüzde ilerleyen paralel hesaplama teknikleri ve bilgisayar teknolojileri sayesinde makine öğrenme teknikleri ilerlemiş; bu sayede birçok gerçek hayat problemine cevap üretilmeye başlanmıştır. İstatistik teorisinden farklı olarak; istatistik, örneklemden sonuç çıkarımı ile uğraşırken makine öğrenme genelleştirilebilir örüntüleri kestirmeye uğraşır: makine öğrenme bunu yaparken bir yöntem istenilen tahmin sonucunu vermeyince öğrenme sistemi baştan tasarlanır; istatistikte olduğu gibi elde edilmiş bilgi kümülatif ve sistemli şekilde ilerlemez. Tipik olarak modelin hatası hakkında varsayımlar, modelin dağılımı ile ilgili varsayımlar, modelin doğrusal olup olmadığı ile ilgili ön kabuller ve modele konu olan değişkenler arası ilişkileri ve dağılım varsayımları istatistik yöntemler için hayati iken, makine öğrenme yöntemlerinde probleme yaklaşımda objektiflikten sapma olarak değerlendirilir. Diğer yandan bilgisayar çağı ile birlikte istatistiksel problemlerin büyüklüğü ve kompleksliği artmıştır. Bu nedenle çok fazla detay, patern ve trendlerin araştırılmasını gerekli kılmıştır. Nitekim verilerin saklanarak işlenmesi ile gözlem yapıldıkça işlenmesi arasındaki fark istatistik yöntemlerin sorunlara cevap hızını etkilemekte ve hafıza ilgili problemlere yol açmaktadır.

Genel olarak gözetim altında öğrenme ve gözetim altında olmayan öğrenme şeklinde ayrılan makine öğrenme algoritmaları; makine öğrenmesi geliştiricilerine geniş bir olanak sunar. Algoritmalar da çoğu kez regresyon, kümeleme ve sınıflandırma olarak karşımıza çıkar. Böylece hızla gelişen makine öğrenme sonucunda birçok makine öğrenme tekniğinin aynı anda kullanılması olarak tanımlanan derin öğrenme teknikleri ve gelişen işlemci teknolojisi yardımıyla birçok probleme cevap üretilmeye başlanmış, her türlü insan davranışını taklit eden çeşitli aygıtlar, robotlar tasarlanmıştır. İlgilenilen problemler bilgisayarla görüntü tanıma ve sınıflandırma, doğal dil işleme, otonom araba, sağlık, dolandırıcılık ve sahtecilik tespiti gibi pek çok problemle ilgilenmektedir (Daumé 2017).

Uygulama problemin doğasına göre ses, görüntü veya sensör verisi gibi çeşitli kaynaklardan elde edilen girdiler, matematik ve istatistik yöntemler kullanılarak, problemde yer alan örüntüler bu veriler kullanılarak bilgisayar tarafından öğrenilmeye çalışılır. Daha sonra buradan elde edilen makine öğrenmesi çıktılarına uygulanan testlerle, öğrenilmiş genellemeler ve bağlantılar sınamaya tabî tutulur. Eldeki verinin öğrenme verisi ve test verisi şeklinde parçalara ayrılmasına çoğu istatistiki yöntemde başvurulmaz.

Veri elde etmede, bu verilerin sanal ortama aktarılmasında, öğrenme ve test aşamalarının her bir basamağında ve ilgilenilen problemin doğasında kaçınılmaz olarak bulunan muhtelif yıkıcı fakat genellemeye katkısı yadsınamayacak birçok bilgi örüntüsü de makine öğrenme algoritmaları tarafından işlenip öğrenilmektedir. Buna karşılık makine öğrenme algoritmalarının bahsi geçen yıkıcı etkileri algılayacak kapasitesi yoktur; nitekim burada öğrenmenin özel koşullarının ne olacağını belirleyen makine öğrenmeyi geliştiren devreye girer (Vorobeychik ve Kantarcıoğlu 2018). Algoritmanın öğrendiği ortama yönelik; makinenin gözden kaçıracağı bu yerlerin kontrol altında tutulmasına insan bilişsel yeteneğinde yer alan sınıflandırmaya ve algısına ihtiyaç duyulmaktadır. Tam da bu yüzden öğrenme algoritmalarının güvenilirliği sorgulanmaya başlamıştır. İşleyen makine öğrenme sistemlerini kandıracak ve yanlış kararlar vermelerini sağlayacak kötü amaçlı yazılımlar üretilmeye başlanmıştır. Bu yazılımlar veri yığınlarını tarayacak yazılımları savuşturup insan dikkatinden de kaçabilecek şekilde evrimleştirerek, makine öğrenme algoritmasının yaptığı sınıflandırmaları kötü amaçlı kişinin isteği doğrultusuna çekme işinde oldukça başarılı bir seviyeye gelmişlerdir (Carlini ve Wagner 2017). Buna karşı verilerin güvenliği ve algoritmaların sağlamlığını arttırmaya yönelik yeni metodlar geliştirilmeye başlanmıştır. Hızla ortaya çıkan bu alan siber güvenlik literatüründe çatışmalı makine öğrenme (adversarial machine learning) olarak kendine yer bulmuştur. Makine öğrenme ve derin öğrenme sistemlerine saldıracak kişilerin sistem hakkında daha önceden bilgi sahibi olup olmadığı, sistemin hangi parçaları hakkında istihbaratı olduğu veya saldırısında belirli bir hedefi gözetip gözetmediği gibi çeşitli sorular sorularak teşhis yapılması; olasılık ve istatistiksel yöntemler yardımıyla öğrenilen sistemlerin açıklarına karşılık tedbir alınması; daha önceki saldırıların ve algoritmaların uygulanan probleme göre başarı ve başarısızlıklarının bilimsel olarak incelenmesi için sınıflandırılması ve düzene

konulması gibi birçok güvenlik sorunu bu alanın ilgilendiği konulardır (Daumé 2017).

Makine öğrenmenin gözden geçirildiği bu çalışmanın bir amacı da bir örnekle çatışmalı makine öğrenme tekniklerini tanıtmaktır. Makine öğrenme ile öğrenilmiş görüntüye ilişkin piksel değerleri ve değerler arasındaki uzaklıkları kullanarak belirlenen hedefe göre yanlış sınıflandırmayı amaç edinen Carlini ve Wagner (2017) algoritması üzerinde durulacaktır. Carlini ve Wagner (2017) algoritmasında basitleştirme sağlayan değişiklikler yapılarak daha başka bir algoritma sunulacaktır. Uygulama verisi olarak 0-9 rakamlarını içeren MNIST (Modified National Institute of Standards and Technology database) el yazısı verileri kullanılacaktır. İnsan algısındaki orjinal görüntü ve saldırı sonucu bozulmuş görüntü arasındaki fark, insan tarafından ve makine öğrenmesi tarafından fark edilemeyecek şekilde bozulmaya çalışılacaktır. Bunu yaparken Carlini&Wagner'den farklı olarak sınıflamayı yapan tanjant hiperbolik fonksiyon yerine daha hızlı sonuç verecek şekilde Maclaurin açılımı kullanılmıştır.

Bu tezin çalışma planı şu şekildedir: ikinci bölümde makine öğrenmenin temel kavramları çeşitli algoritmalar örnek verilerek açıklanacaktır. Üçüncü bölümde çatışmalı makine öğrenme incelenecek ve bazı saldırı algoritmaları örnek verilecektir. Dördüncü bölüm uygulama bölümü olup, çatışmalı makine öğrenme kullanan Carlini ve Wagner (2017) algoritmasının nasıl değiştirildiği ve çıkan sonuçların açıklandığı bölümdür. Son bölüm olan beşinci bölüm ise tezden çıkarılan sonuçları kapsamaktadır.

2. MAKİNE ÖĞRENMEDE BAZI TEMEL KAVRAMLAR VE YÖNTEMLER

Makine öğrenme yöntemlerin çokluğu ve birçok disiplinden aldığı kavramları bir araya toplaması ve gelişmekte olan bir disiplin olması nedeniyle terminolojisi değişebilmektedir. Özellikle matematik ve istatistik gibi disiplinlerden aldığı kavramlar, makine öğrenme içerisinde orjinal bağlamlarından farklı olarak değerlendirilmektedir. Probleme göre çözüm yöntemini değiştirme ve/veya sonuç yetersiz görüldüğü takdirde sürece müdahalede bulunabilir. Bu sayede en iyi kestirim ya da sonuç bulununcaya kadar öğrenme geliştirilir. Bu bölümde istatistik ve makine öğrenme kavramlarının açıklanması için, istatistik kavramlarına zaman zaman başvurulacaktır. İlk önce makine öğrenmenin genel tanımı verilecektir. Diğer kavramların karşılıkları açıklamaları yapılarak verilmiştir. Son olarak makine öğrenme sürecinin adımları ve nasıl işlediği, temel makine öğrenme türlerinden örnekler verilerek anlatılacaktır.

2.1 Makine Öğrenmenin Tanımı

Makine öğrenme, verinin sağladığı deneyimle otomatik olarak kendini geliştiren bilgisayar programlarını konu edinir. Buna göre, verilen T gibi bir görev için hazırlanan programının bu görevdeki performansı, P performans ölçütüne göre E deneyimiyle artıyorsa bu programa bir makine öğrenme programı denir (Mitchell 1997). Burada makine, belirli bir görevi yerine getiren; mekanik veya elektrik birçok aksamla çalışan cihazlardır. Öğrenme terimi, tekrar ve deneyim sonucu davranış ve bilgi düzeyinde meydana gelen değişimi anlatır. Makine öğrenme, istatistik, matematik ve bilgisayar bilimleri disiplinlerinin karar teorisi, enformasyon teorisi ve optimizasyon teorisi gibi birçok teoriyi birleştirerek, veriden çıkarımlar yapma uğraşıdır.

Bilgisayar çağı ile birlikte istatistiksel problemlerin kapsamı ve kompleksliği artmıştır. Bu durum çok fazla detayın, örüntü ve trendlerin incelenmesini gerekli kılmıştır (Daumé 2017). Makine öğrenme birçok farklı amaçla kullanılabilir. Günlük yaşamda kullanımına örnek olarak kişiye özel haber içeriği sunan internet sitelerinde olduğu gibi otomatik olarak kendini geliştiren ve kullanıcıya göre uyum sağlayan sistem örneği verilebilir. Bir metindeki el yazısı ve bilgisayar yazısını ayırma işi gibi insan davranışlarını taklit edip yerine geçebilecek sistemler bir diğer uygulama örneğidir.

Karayolu trafik denetimi gibi belirli bir görevi yerine getirebilecek çok karmaşık ve büyük sistemleri zor ve maliyetli mühendislik tasarımlarından kurtarma işi bir başka örnek olarak verilebilir. İnternetin yaygınlaşması, işlem gücünün artması ve bu alana yapılan yatırımlar sayesinde makine öğrenmenin kullanım alanları genişlemektedir (Daumé 2017).

Makine öğrenme yöntemlerinde işlem gücü, çıktı sisteminin daha kaliteli tahmin üretmesinden ziyade, elde edilecek programın ne kadar hızlı çalıştığını ifade eder. Dolayısıyla 10 sene önce 15 dakikada çıktı üreten bir makine öğrenme algoritması, gelişen paralel işlem donanımlarıyla beraber 15 milisaniyede çıktı üretebilir hale gelmiştir. Bu sayede günümüzde makine öğrenme yöntemlerinin kullanımı ve üzerine sistemler geliştirilmesi daha ekonomik hale gelmiştir.

2.1.1 Makine öğrenme temel kavramları

Bazı terim ve kavramlar, bilgisayar bilimlerinden, istatistikten ve matematikten alınarak makine öğrenme konusu içinde çoğu kez kendine has biçimde kullanılagelmektedir.

Her türlü ses kaydedici, sensör, kamera gibi aygıtlarla sağlanan ses, yazı, resim ve değer gibi gözlemler *veri* (data) olarak adlandırılır. Veriyi işeyebilmek için bir ortama aktarmak, sayısal hale getirmek, sıralamak için çeşitli yöntem, veri tabanları gibi aracı aygıtlar ve çeviriciler kullanmak gerekir. Bu veri içinde, her bir *özellik* (feature) öğrenilecek kavramın onu betimleyen ölçülebilir parçalarına verilen isimdir. Ayrıca veri içinde her bir gözleme ait özelliklerin yanında bir veya birden fazla *etiket* (label) bulunabilir. Etiketler özelliklerin tanımlayıcılarıdır. İstatistikteki tanımlarla ilişki kurulacak olursa özellikler, istatistik modelinde açıklayıcı değişkenler rolünde yer alır, etiketler ise açıklanan değişken rolünde yer alırlar. Bu özellikler ve etiketler skalar şeklinde ya da bir vektörel değerler alabilirler. Anlamlı hale getirilmiş öğretilecek bilgilerin bir araya toplanmasıyla elde edilen bu küme *eğitim seti* olarak isimlendirilir. Veri makine öğrenmenin temel girdisi olup, diğer girdilerle birlikte öğrenme sonucu olarak *model* ortaya konur. Model, veriyi alıp, işleyip; özelliklerini anladıktan sonra elde edilen kurallardan, sayılardan ve veri yapılarından oluşan kompleks yapıdır. *Makine öğrenme algoritmaları* ise model oluşturulurken kullanılan her bir küçük adıma

verilen addır. Aynı zamanda modelde kullanılan regresyon, kümeleme vb. yöntemler de makine öğrenme algoritmaları içinde değerlendirilir. Eğitim tamamlandıktan sonra bilgilerin öğrenilip öğrenilmediğini anlamak için sistemi sınava tabi tutan veri kümesine *test seti* denir. Bir öğretmenin sınıfta anlattığı kitap, sorular ve çözümleri bir eğitim seti gibi düşünülürse daha sonra yapılacak ders içeriğiyle örtüşmek kaydıyla yapılan değerlendirme sınavındaki bilgi ve sorular da test seti olarak değerlendirilebilir (Daumé 2017). Test verisinin öğrenme sonuçlarına uygulanmasından sonra öğrenme sonuçlarını yadsıyan sonuçlar elde edilirse makine öğrenme nihai *hedef*e ulaşmamış demektir ve öğrenmeye ilişkin model baştan kurulur. Öğrenme işinin test sonucu başarılı olup olmadığı bir *performans* ölçütü kullanılarak belirlenir. Test setine verilen doğru ve yanlış cevapların değerlendirilmesi modelin performansını verir. Modelin *doğruluk* (accuracy), *hassaslık* ve *özgüllüğü* (sensitivity and specifity), doğru yanlış cevapların oluşturduğu *hata matrisi* (confussion matrix) aşağıda değinilecek olan gözlem altında olmayan öğrenme için *eşleme matrisi* (matching matrix) ile gözlemlenebilir (Daumé 2017, Stehman 1997). Hata matrisi ikiden fazla kategoriler için de oluşturulabilmektedir.

Öğrenme oluşturulurken algoritmaların içinde yer alan ve öğretici tarafından veya deneyimle belirlenen bazı parametreler vardır. Bunlara *hiperparametrelerdir*; geliştiricinin belirlediği, makine öğrenme süreci üzerinde ayarlama yapılmasını sağlayan, öğrenmenin performansını veya ürettiği çıktıyı şekillendirebilecek yardımcı sabitlerdir. Öğrenme sürecinde verinin bir kısmı hiperparemtrelerin deneysel olarak belirlenmesinde kullanılabilir. *Öğrenme oranı* (learning rate) hiperparametreler arasında önemli bir yere sahiptir, öğrenmenin hızını belirleyen sabittir. Yüksek bir öğrenme oranı hızlı yani ezberlemeye meyilli bir eğitim demek iken, düşük bir öğrenme oranı uzun süren yavaş bir eğitim demektir. Küçük bir öğrenme oranı ile veriden olabildiğince detay elde edilmesi sağlanırken, büyük bir öğrenme oranında ana hatlar kaçırılıp gerçekte öğrenmeye katkısı olmayan birçok ilişki öğrenilmiş olur (Bishop 2006). *Kayıp fonksiyonu* (loss/cost function), gerçek çıktılarla makinenin öğrendiği çıktılar arasındaki farkın bir fonksiyonudur. Makine öğrenmesi her bir veri girişinde bu fonksiyonda minimum yapılmaya çalışılarak; gerçekle öğrenilen arasındaki fark azaltılmaya çalışılır. Böylece bu matematiksel fonksiyon yardımıyla öğrenmenin nerede duracağı geliştirici tarafından ayarlanabilir. Kayıp fonksiyonu seçimi makine öğrenmenin arzu edilen

performansı ile ilgilidir. Kolay olması ve türevlenebilmesi açısından karesel kayıp fonksiyonları sıkça kullanılır (Bishop 2006). Öğrenme sürecinde eldeki öğrenme verisinin tüm önerilen makine öğrenme modelinin sonucunun ortaya çıkarılmasıyla son bulması bu öğrenmenin *devir* (epoch) sayısının 1 olması olarak adlandırılır. Aynı veri setinin L defa bu işleme sokularak yapılması ile elde edilen öğrenme sonucu bu öğrenmenin L devirli bir öğrenme gerçekleştirdiği anlaşılmalıdır. Aynı veri setinin her devirde gözlem sırasının bir başka permütasyonun alınması önerilmektedir (Daumé 2017). Devir sayısı ileride bahsedilecek olan yapay sinir ağlarında önemli bir işleve sahip olduğu görülecektir. Devir sayısı öğrenme sürecinin bir hiperparametresidir. Yapay sinir ağlarında devir sayısı başka hiperparametrelerle birlikte işleme tabi tutulur. Başka hiperparametrelere de bağlı olarak devir sayısının arttırılıp azaltılmasıyla makinenin genelleyebilme yeteneği geliştiricinin isteği doğrultusunda değiştirilebilir. Devir sayısının gereğinden az olması eksik öğreneme; fazla olması ise ezberleme ile sonuçlanabilir. *Grup boyutu* (batch size) ise algoritmaların eğitim setini kaçar kaçar işleyeceğini belirten kullanılan işlemcinin teknik özelliklerinin hesaba katıldığı bir hiperparametredir. İşlemcinin işleme kapasitesi ile ilgilidir, örneğin bir işlemcinin aritmetik işlem biriminde aynı anda kaç bit işlem görüldüğü bu grup sayısını belirlemekte etkilidir. Eğitim yapılan makinedeki öğrenmenin süresini ayarlamakta etkilidir. Grup boyutunun eğitimin sonucuna etkisi yoktur. Veri ile test setinin boyutlarının oranı da bir hiperparametre olarak düşünülebilir. Daumé (2017) eldeki verinin işlenmesini %70'ini öğrenme seti, %20'sini test seti, %10'unu hiperparametre belirlemek için kullanılmaktadır.

Makine öğrenme algoritmaları genellikle çok fazla işlem gücü gerektirmektedir. Özellikle kayıp fonksiyonlarının çıktılarının işlenebilmesi için optimizasyon algoritmasının seçimi de bir hiperparametre olarak görülebilir. *Dereceli alçalma* (gradient descent) makine öğrenmede birçok popüler sayısal yöntem arasında en çok kullanılan araçtır. Birinci dereceden türev gerektiren iteratif bir yöntemdir. Benzer bir sayısal analiz yöntem stokastik dereceli alçalma yöntemidir. Bu yöntemde veri setinden rasgele bir alt küme alınarak çözümleme yapılır. Adam optimizasyonu ise her bir özelliğin uygun şekilde öğrenme oranlarının değiştirilerek dereceli alçalmanın kullanıldığı bir sayısal analiz yöntemidir (Bishop 2006). Bu çalışmanın uygulama kısmında Adam yöntemi kullanılmıştır. Dereceli alçalma sayısal yöntemi örnek

uygulama bu bölümde ilerde anlatılacaktır.

Bir model eğitim setinden anlam çıkaramamış, yeniden tekrar edilen deneylerde veya test seti verilerinde yanlış çıktılar veriyor ise bu durum *eksik öğrenme* (underfit) olarak adlandırılır. Genellikle veriye uygun olmayan model seçimi veya veri yetersizliği sebebiyle oluşmaktadır. Bir öğrencinin sınava çalışırken ders içeriğini dikkate almayıp sınavdan düşük not alması olarak görülebilir. *Aşırı öğrenme* (overfit) ise model eğitim setindeki verilerle sadece veri seti için doğru çıktı sağlayıp ezberleme yapılmasıdır. Model farklı bir senaryo ile karşılaştığında hep aynı çıktılar verilir. Bir öğrencinin aynı konuları tekrar edip ezberleme yapması olarak düşünülebilir, öğrenci farklı soru ile karşılaştığında doğru cevap veremeyecektir. Bir çeşit kopya gibi de düşünülebilir. Bu duruma istatistikte de karşılaşılır; örneğin regresyonun tahminin verilere tam fit sağlanması bir aşırı öğrenme olarak düşünülebilir. Örneğin elde dört veri varsa dört veriye üçüncü dereceden bir polinom tam olarak fit edilebilir; ama böyle bir regresyon doğru çıktılar veremeyecektir. Aşırı öğrenme durumunu engellenmesi, algoritmanın genelleştirmeyi daha iyi yapması için kayıp fonksiyonuna yapılan düzenlemeler *regüralizasyon* olarak adlandırılır. Aşırı öğrenmenin engellenmesine yönelik olarak devirlerin verilerin değişik permütasyonlarına uygulanması da yukarıda değinilen bir diğer yöntemdir. Eksik öğrenmeye karşı ise çapraz doğrulama (cross validation) yöntemi kullanılabilir. Çapraz doğrulamada öğrenme veri seti k tane partisyona (ayrık setlere) ayrılır. Her seferinde bunlardan biri test için bir kenara ayrılıp, geriye kalan $k-1$ tanesi ile eğitim yapılır. Eğitimden sonra test için ayrılan veride kestirim hatası değerlendirmesi yapılır. Bu yöntem her bir partisyon için (k kez) tekrarlanır ve elde edilen sonuçların ortalamasıyla yargıya varılır (Clarke vd. 2009).

Makine öğrenme sonucu oluşan çıktılar reel bir sayıyı tahmin etme işlemi ise kullanılan algoritma regresyon olarak adlandırılır. Birçok objeyi ilgilerine göre seçme amacıyla kullanılan algoritmalar sınıflandırma olarak adlandırılır. Buradaki her bir objeyi belirleyen fonksiyon diskriminant fonksiyonu olarak adlandırılır. Sınıf sayıları ikili ise ikili evet- hayır diskriminasyonu; ikiden fazla ise çoklu diskriminasyon olarak adlandırılabilir.

Makine öğrenme terminolojisinde veride, modelde ve hatta uygulamayı geliştiren kişide

bulunabilen, olumlu ya da olumsuz, engellenemez taraflılık durumu *bias* (yanlılık) olarak adlandırılır. Birçok çeşidi bulunabilir. Örneğin, örneklem yanlılığı verileri toplarken elde edilen yanlılıktır. İnternet üzerinden yapılan bir ankette, internete erişimi az olan kitlelerin görüş ve girdilerinin yansımaması bu duruma bir örnektir. Hariç tutma yanlılığı yığını doğru temsil etmediği düşünülen gözlemlerin çıkarılmasıyla oluşan yanlılığa bir örnektir. Örneğin, seçim anketinde %1~2 alan partileri sonuçtan silmek buna bir örnektir. Ölçüm yanlılığı veriyi toplayan aletlerdeki yanlılığı ifade eder. Bozuk bir kamerayla çekilen resimlerden bir öğrenme algoritması geliştirmeye çalışmak böyle bir yanlılığı içerebilir. Endüktif yanlılık ise (tür olarak) insanda bulunan ön kabullerden doğan yanlılıktır. "Penguenler memeli midir yoksa kuş mudur?" sorusuna çoğu insanın yanlış cevap verip "memeli" demesi endüktif yanlılığa örnektir (Daumé 2017). Bütün makine öğrenmelerde endüktif yanlılığın olması beklenebilir bir yanlılık durumudur.

2.1.2 Makine öğrenme süreci

Makine öğrenme tipleri eldeki veri ve amaca göre sınıflandırılarak incelenebilir. Bilinen makine öğrenme tipleri gözlem altında öğrenme (supervised learning), gözlem altında olmayan öğrenme (unsupervised learning) ve pekiştirmeli öğrenmedir (reinforcement learning). Makine öğrenmede kullanılacak yöntemler makine öğrenme algoritması olarak adlandırılır. Değişik makine öğrenme tiplerinin ve değişik algoritmaların bir arada kullanıldığı bir makine öğrenme derin öğrenme (deep learning) olarak adlandırılır. Makine öğrenme süreci ardışık adımlar biçiminde aşağıda anlatılacaktır.

Ölçme aygıtlarıyla gözlem birimlerinin ölçümlenir. Öğrenme için önce veri gerekir. Veri gözlem birimlerindeki değişik özelliklerin (değişkenlerin, features) bir ölçümlenmesi (gözlemlenmesi) ile elde edilir. Veriler düzensiz ve insan eli değmediği müddetçe anlamsızdır. Bu nedenle gözlemleme kamera, sensör, veritabanı gibi araçlarla başlayan ve verilerin uygun bir organizasyona sokulmasıyla (veri analizi) ile devam eden bir süreçtir. Bu anlamda istatistik disiplini pek çok araca sahiptir ve makine öğrenme bu araçları yeri geldikçe kullanır.

Veri makine öğrenmenin amacına göre ön işleme sokulur. Çalışılan konu ve amaca göre veri anlamlandırılabilir öğelere dikkate alınarak organize edilir. Bu aşamada gözlemler

filtrelenir (gürültü filtreleme). Gözlem birimine ilişkin özelliklerin (değişkenlerin) eklenip çıkarılması yapılır. Kamera görüntülerinde olduğu gibi bozulmalar normalizasyona tabi tutulur. Veri bilgisayar tarafından işlenebilecek hale getirilir.

Öğrenme algoritması ve öğrenme modeli seçilir. Düzene sokulmuş veri makine öğrenmesinin amacına ve sınırlamalarına göre bir takım algoritmalar aracılığı ile işlenir. Verinin öğrenme seti olarak ayrılan kısmında öğrenme gerçekleştirilir. Bu aşama makine öğrenme aşamasıdır. Regresyon, kümeleme vb. istatistik yöntemler makine öğrenmede algoritmalar olarak adlandırılır. Verinin algoritmalar ile işlenmesi sonucu ortaya çıkan çıkarımlar model olarak adlandırılır. Örneğin, regresyon algoritma olarak kullanılmış ise ortaya çıkan matematiksel yapısıyla birlikte regresyon katsayıları öğrenme modelini, oluşturur. Öğrenme tamamlandıktan sonra test verisi ile öğrenme sınava tabi tutulur (Mitchell 1997).

Model testi yapılır. Öğrenme verisinden ayrı tutulan öğrenmeyi gerçekleştirenin de hiçbir şekilde verilerin doğası hakkında haberdar olmadığı test verisi ile model teste tabi tutulur. Performansının ne kadar başarılı olduğu kontrol edilir. Aşırı öğrenme veya eksik öğrenme var ise, geçerli olmayan çıktılar gözleniyor, kestirimler isabetli değilse vb. performans sonuçları tatmin edici değilse önceki adımlara dönülür. Performans, kayıp fonksiyonuna göre veya çoğu kez sınıflandırmaların amaçlı makine öğrenmelerinde yapıldığı gibi hata matrisleri kullanılarak izlenebilir (Mitchell 1997). Aşağıda bu tezde öne çıkan gözlem altında öğrenmeye ilişkin bilgiler verilecek ve kullanılan araçlar tanıtılacaktır.

2.2 Gözlem Altında (Supervised) Öğrenme

Veri setindeki her bir veri (girdi) özellikler (istatistikteki bağımsız, açıklayıcı değişkenler) ile bunlara karşılık gelen etiketlerden (istatistikteki bağımlı, açıklanan değişken) oluşuyorsa değişkenlerin eğiteceği doğru cevaplar (etiketler) biliniyor demektir. Biliniyor olan etiketlerin yönlendirmesiyle veri öğrenmeye tabi tutuluyorsa bu tür öğrenme gözlem altında (supervised) öğrenme denir. İstatistikteki yöntemlerin çoğu gözlem altında öğrenmede bir algoritma olarak kullanılırlar. Bir çeşit koşullu dağılım (özellikler etiketlere koşuldur) aracılığıyla öğrenmedir. Özelliklerle sonlu sayıdaki

kategorileri olan etiketler arasındaki ilişkinin genellenmeye çalışıldığı makine öğrenmesidir. Algoritma etiket verisinin ölçme düzeyine göre sınıflandırma veya regresyon olarak biçimlenir. Gözlem altında öğrenmede etiket verisinin kategorik olması durumunda algoritmaya sınıflandırmadır, lojistik regresyon da bu şekilde değerlendirilir. Etiket değerleri sürekli olması halinde algoritmaya regresyon adı verilmektedir. Gözlem altında öğrenme yöntemleri diğer öğrenme metotlarına göre genellikle karmaşık hesaplamalar gerektirmekle birlikte isabetliliği (accuracy) yüksektir. Bu da yüksek veri boyutu, yoğun işlem ve zaman gerektirmektedir (Daumé 2017). Uygun bir kayıp fonksiyonuna göre en az hata ile etiketler sınıflandırılır veya kestirilir. Bu tip öğrenmede değişik makine öğrenme algoritmaları kullanılabilir. Bunlardan bazıları destek vektör makineleri (support vector machine), perseptron, yapay sinir ağları, karar ağaçları ve rasgele ormanlar (random forest) örnek verilebilir.

2.2.1 Karar ağacı

$A = \{Q_1, Q_{2,} \dots Q_r\}$ n tane nominal (adlandırma sınıflandırma) veya ordinal (sıralama) değerli $Q_i, i = 1, 2, \cdots, n$ değişkenlerin (features) kümesini göstersin. Yapılan gözlemler sonucunda değişkenlerin gözlem değerlerine göre bir gözlem biriminin hangi sınıfa (Y etiketinin nominal ya da ordinal değerlerine) atanacağı belirlenir. Bu işlem en az değişken (feature) kullanılarak ve en az sınıflama hatası ile yapılmak istendiğinde gözlemlerin hangi sırada değerlendirileceği ya da sorulacağı önem kazanır. Çünkü bazı sorular (feature) diğerlerine göre az hata ile sınıflama yapılmasında daha etkili olabilirler. A'nın içerdiği değişkenler arasında $n!$ tane sıralama yapmak mümkündür. Bu sıralamalardan birine j denirse her bir sıralama $(j_1, j_2, \dots j_n)$ $\{1,2 \dots n\}$'nin bir permütasyonu olacaktır. A kümesinin j. sıralaması için gösterimi

$$A_j = \{Q_{j1}, Q_{j2}, \dots Q_{jn}\}$$

olsun. Karar ağaçları makine öğrenme yöntemi ile Y'ye ait etiketler, A_j sıralı kümesi ile ayrık parçalara bölünmeye çalışılır. Burada öğrenme verisinin parçalanması işlemleri m sıralı soru sonucunda oluşacak yanıt kümesi Y değerlerine göre yapılacaktır. A_j gözlemleri alındıkça, her bir Q_{ji}'nin kategorilerine göre dallar gelişecek ve sonunda bu dallar izlenerek Y değişkeninin değerlerine ulaşılacaktır. İnformasyon, herhangi bir

sistemden elde edilen derlenmiş bilginin değeridir. Öğrenme sırasında en süpriz olan gözlemin informasyon bakımından sisteme katkısı diğerlerine göre daha fazladır. Yapılan sınıflamanın uygunluğu için değişik ölçütler kullanılır. Bu ölçütlerin çoğu informasyon ölçüsüne dayanır. İnformasyonu en aza çeken sınıflama en iyi sınıflama olarak düşünülebilir. Saf olmama temelli (impurity based) ölçüt, enformasyon kazancı (information gain), Gini katsayısı, enformasyon kazanç oranı, olabilirlik oranı ki-kare ölçütü gibi değişik ölçütlerle sınıflamanın yeterli olup olmadığına karar verilebilir. Bu ölçütler hakkında bilgi Maimon ve Rokach (2005) kaynağında bulunabilir.

Sorulacak maksimum soru sayısı, her bir soru için seçenek sayısı (ağacın maksimum derinliği) oluşturulacak karar ağacının hiperparamatreleridir. Model, makinede işletilirken tüm ağacı büyük dallara ayırarak çözmek zaman ve işlem gücü bakımından külfetli olduğundan gereksiz sorular budanmalıdır. Örneğin ders seçimi için öğrencilere önerilerde bulunan bir makine öğrenme uygulaması için karar ağaçları yöntemi uygun olabilir. Böyle bir uygulamada öğrencinin yeni alacağı ders hakkında daha önceki derslerin ön koşulu olup olmadığı, aynı alan dersi olup olmadığı, dersi seçip kalan öğrencilerin oranı ve dersin sabah veya öğleden sonra oluşu gibi sorular bu uygulamanın özellikleri olabilir. Bu soruların doğru sırada sorulması ile en az soruda ders seçimine yardımcı bir karar ağacı makine öğrenmesi uygulaması elde edilebilir (Daumé 2017).

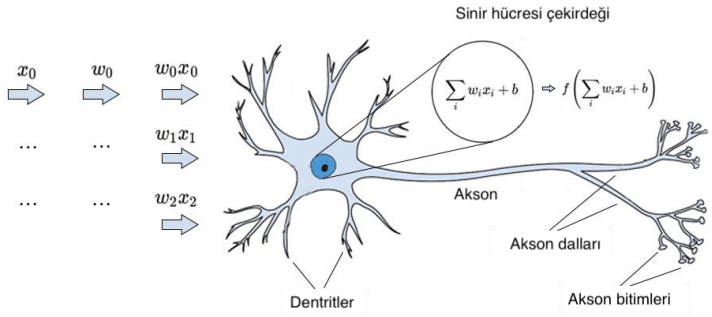
2.2.2 k-Yakın komşuluklar

Bu yöntemde verilerin sisteme yüklenmesi ile bir öğrenme gerçekleşmiş kabul edilir. Sonra gelen her veri ile bu öğrenme test edilerek geliştirilir. Gözlem altında öğrenme ile bunu tarif etmek için $A = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ şeklindeki bir öğrenme setini ele alalım. Buna göre yeni gelen bir gözlem verisi x^q olsun. Bu gözlemin hangi y^q sınıfına ait olacağı sorgulanacak olsun. x^q'ya yakın çevrede var olan gözlemlerin çoğunluğunun y kategorisi yada sayısal değeri ne ise söz konusu verinin sınıfı da benzer olacaktır. Bunu yapmak için yakın çevre tanımı bir uzaklık ölçütü ile operasyonel hale getirilir. x^q noktası ile bu noktaya en yakın diğer k tane veri arasındaki uzaklıklar hesaplanır. Bu k tam sayısı önceden belirlenen bir hiperparametredir ve test bitene

kadar aynı kalır. Bu noktanın komşuluğundaki k tane gözlem dahil edilecek şekilde uzaklıklar sıralanır. En yakın k gözlem içinde en fazla bulunan sınıf bu x^q gözlemin de sınıfı olur. Çoğunluk, iki sınıfın olduğu bir çalışmada gözlemlerin %50'si olarak, dört sınıfın olduğu bir uygulamada ise çoğunluk gözlemlerin %25'i olarak belirlenebilir. Sayısal veriler için öklidyen uzaklığı $d_1 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ veya Manhattan uzaklığı $d_2 = \sum_{i=1}^{n} |x_i - y_i|$ vb.; kategorik veriler için de Hamming uzaklığı veya Jaccard/Tanimato benzerliği bu amaçla kullanılabilir. k sayısının belirlenmesi kritik öneme sahiptir; k sayısının tek olarak seçilmesi çoğunluğu belirlemede kolaylık sağlar. k'nın büyüklüğü öğrenmenin ezberleme ya da öğrenememe problemlerine yol açabilir. k sayısı büyük olduğunda modelde düşük varyans ve büyük yanlılığa neden olurken, küçük k yüksek varyans ve düşük yanlılığa neden olur. Eğer veri çok sayıda sıradışı (outlier) gözlemler içeriyorsa genel olarak k'nın büyük seçilmesi önerilir.

2.2.3 Yapay sinir ağları

Doğada bulunan sinir hücrelerini taklit ederek makine öğrenme modeli elde etmek için kullanılan yöntemlerdir. Beyindeki sinir hücreleri birbirlerine bağlanıp öğrenme ve hafızayı geliştirerek zamanla yeni ve daha fazla bağlar oluştururlar. Yapay sinir ağları da bu sistemi benimseyip bu bağları ağırlıklarla matematiksel olarak ifade eder. Öyle ki bir bağ olmayacaksa o uçtaki ağırlık sıfır olur; buna karşın bir ağ diğerinden daha kuvvetli ise matematiksel olarak o uca daha fazla ağırlık verir. Bu ağırlık, katsayı çarpanı olarak ile ifade edilir. Bu şekilde ağın sürekli olarak kendini güncellemesiyle değişen ağırlıklarla çalışan bir yapay sinir ağı elde edilir. Diğer bir deyişle her bir hücre kendisine gelen veriyi toplamla ifade edilen girdi olarak alır ve bu toplamın bir fonksiyonu olarak işleme sokar ve bir çıktı üretir. Aşağıdaki şekile göre bir tek gözlem biriminden alınan d boyutlu bir verinin nöron ve makine öğrenmede benzeştirmesi (anoloji) olan perseptron algoritmasının işlenmesi gösterilmiştir **(Şekil 2.1).**



Şekil 2.1 Tek bir sinir hücresi (nöron) ile makine öğrenmede veri işleme birimi perseptronun bağdaştırılması (Kaynak: https://krisbolton.com/2018/08/22/a-practical-introduction-to-artificial-neural-networks-with-python.html-27.07.2022:11.35 adapte edilmiştir)

Tek bir nöron için toplamı işleyen ve nihai olarak tek bir değer üreten f fonksiyonu bir aktivasyona neden olacak sinyal üretir. f, aktivasyon fonksiyonu olarak adlandırılır. Bu, nöron hücresini taklit etmekten de uzak basit bir görünüm sunar, sadece bir sinyal verir. Ancak nihai aktivasyon tek bir sinyale bağlı olmayabilir. Bu nedenle farklı nöronlardan gelen sinyalleri bir arada işleyecek bir sistem kurulması gerekir.

Eğer çıktı çok değişkenli veya çoklu doğrusal regresyonu olarak ifade edilseydi aktivasyon y ya da y_k değerlerinin bir tahmini veya kestirimi olarak düşünülecekti. Çıktı bir sınıflama işlemi olsaydı bir diskriminant fonksiyonu ile veya yine bir sınıflama sağlayan lojistik regresyon ile aktivasyon yapılacak; aktivasyonun sonucunda girdilere karşılık gelen y sınıfları bulunacaktı. Bu yapılan işlemlerin tümü bir doğrusal regresyon modelinde olduğu gibi y'yi belirli parametrelere göre girdilerin doğrusal kombinasyonlarının oluşturulmasına benzetilebilir. Ama bu varsayım araştırılan sistemin hep doğrusal modele uygun olduğunu, bütün işlemlerin bu düzgünlükte işlediği anlamına gelir. (x_1, x_2, y) gözlem altında öğrenme verisi olsun. w ağırlıklar vektörü, doğrusal regresyon modelinin deterministik kısmı $x = [x_1, x_2]$ olmak üzere:

$$y(x, w) = w_0 + w_1x_1 + w_2x_2^2 + w_3x_1x_2$$

gibi olduğunu varsayalım. Deterministik model $\phi_0(x_1, x_2) = 1$, $\phi_1(x_1, x_2) = x_1, \phi_2(x_1, x_2) = x_2^2, \phi_3(x_1, x_2) = x_1x_2$ olmak üzere aşağıdaki gibi de yazılabilirdi:

$$y(x, w) = w_0 + w_1\phi_1(x_1, x_2) + w_2\phi_2(x_1, x_2) + w_3\phi_3(x_1, x_2)$$

$$= w_0 + \sum_{j=1}^{3} w_j \phi_j (x_1, x_2)$$

Son eşitlik görüldüğü gibi y, $\phi_j(x_1, x_2)$ gibi baz fonksiyonlarının lineer kombinasyonu şeklinde ifade edilmiştir. Regresyon, aktivasyonu doğrusal bir çıktı olarak verilen bir sistem modeli olarak düşünüldüğünde;

$$y(x, w) = f\left(w_0 + \sum_{j=1}^{3} w_j \, \phi_j \, (x_1, x_2)\right)$$

$$= w_0 + \sum_{j=1}^{3} w_j \, \phi_j \, (x_1, x_2)$$

olacaktır. Fakat makine öğrenmede amaç daha iyi kestirim ve modeller karmaşık olabileceği için f aktivasyonun fonksiyonunun ve ϕ_j baz fonksiyonlarının ne olduğu yukarıdaki gibi çoğu kez bilinmez. Genel hali ile baz fonksiyonları

$$y(x, w) = f\left(w_0 + \sum_{j=1}^{M} w_j \, \phi_j \, (x_1, x_2 \ldots \, x_d)\right)$$

biçiminde yazılabilir.

Yapay sinir ağlarında amaç aktivasyon fonksiyonu istenilen karmaşıklıkta ve istenilen isabetlilikte kestirim verecek şekilde ve yine parametrelere bağlı olacak, ve parametrelerinin veri geldikçe yeni duruma uyum sağladığı ϕ_j baz fonksiyonları kullanmaktır. Bu amaç için izlenecek bir yol gözlemlerin lineer kombinasyonlarının doğrusal olmayan baz fonksiyonları seçmektir (Bishop 2006). Her katmanda seçilecek nonlineer baz fonksiyonun verilerin lineer kombinasyonlarındaki değerleri yeni bir girdi olarak işlev görecektir. Son katmanda kullanılacak olan aktivasyon fonksiyonu verinin doğasına göre çıktı üretecek yapıda seçilmelidir. Eğer problem bir regresyon problemi ise aktivasyon fonksiyonu birim fonksiyon, bir sınıflama problemi ise bir sigmoid fonksiyon vb. seçilebilir. Ağırlıkların (parametre tahminlerinin) veri geldikçe değişimi problemde kullanılacak olan kayıp fonksiyonu dikkate alınarak en az hatayı verecek şekilde yapılması sağlanır.

Bu bölüm boyunca $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$, $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ öğrenme verisinden bir gözlemi veya d boyutlu gözlem vektörünü gösterecektir. Çıktı olarak da $\boldsymbol{y} = (y_1, y_2, \cdots, y_k)$ verilecektir. M ağırlık vektörünün boyutu veya sabit terim dahil olmak üzere bir doğrusal regresyondaki parametre sayısı gibi düşünülebilir ve modelle birlikte

belirlenmiş bir hiperparemetredir. $w = (w_1, w_2, \cdots, w_{M-1})$ başlangıçta uygun olarak belirlenmiş daha sonraki adımlarda da güncellenecek olan ağırlık vektörünün veya bir regresyon (çoklu veya çok değişkenli olabilir) modelindeki sabit parametresi hariç diğer parametrelerin vektörünün gösterimidir. w_0 sabiti de yanlılık (bias) sabitini veya bir regresyon modelindeki sabit parametresini ifade edecektir.

Sinir hücreleri gelen uyaranları algılar ve alınan uyarı yada uyarılar bir merkezde toplanarak belirli bir eşiğe göre değerlendirilir; bu değerlendirme sinir hücresinin doğasına göre yapılır. Bunun sonucunda sinir hücresi akson bitimlerine sinyal veya sinyaller gönderir veya göndermez. Vücudun dinamikleri de bu sinyale göre tepki verir. Perseptron bu işlemlerin matematiksel uyarlamasıdır. Gözlem verisi başlangıçta verilen, rasgele olabileceği gibi deterministik olarak da belirlenmiş ağırlıklarla çarpılarak toplanır. Bu toplam tıpkı nöronlardaki gibi belirli bir eşik değerle karşılaştırılır. Buna göre bu karşılaştırma işlevi bir fonksiyon aracılığıyla gerçekleştirir ve bu sonuca bakılarak bir karara varılır. Karar herhangi bir çıkarım olabilir; bu bir tahmin veya sınıflandırma işlemi olabilir. Yukarıdaki şekilde verilen f fonksiyonu eşik değerinin geçilip geçilmediğini belirleme işlevine sahiptir ve *aktivasyon fonksiyonu* olarak adlandırılır (Daumé 2017). Aktivasyon fonksiyonunun argümanı *net toplam* olarak ifade edilir. w_0 tıpkı ağırlıklar gibi süreç içinde belirlenebilecek veya öğrenme sürecinin başında belirlenmiş bir sabit olup; öğrenmede varılması istenen sonuca göre değişiklik yapılabilecek bir ağırlık yada parametredir. w_0 yan (bias) katsayısı olarak adlandırılır. Makine öğrenme terminolojisinde w_i'ler ağırlıklar olarak adlandırılacaktır.$w_0 + \sum_{j=1}^{M} w_j \phi_j (x_1, x_2 \ldots x_d)$ net toplamında fonksiyonlar doğrusal olmak zorunda da değildir; çoğunlukla parametrelerine göre doğrusal değildirler (Bishop 2006). Örneğin $\phi_j(x_1, x_2, \ldots x_d)$'ler M dereceli polinomiyal bir regresyon modelinde çarpımsal terimleri veya üstel terimleri gösteren fonksiyonlar olarak düşünülebileceği gibi çok daha karmaşık fonksiyonlar da olabilir. $\phi_j(x_1, x_2, \ldots x_d)$ fonksiyonları baz fonksiyonları (basis functions) olarak adlandırılır.

Nöronların paralel olarak dizilimleri ile bir katman oluşturulur. Girdi vektörünün kendisi de bir katmandır, ancak işlem yapılmaz. Aşağıdaki gösterimde j, bir katmandaki nöronların indisini göstersin. $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ gözlem vektörünün w_{ji} i ağırlıklarına

göre doğrusal fonksiyonuna

$$z_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0}$$

göre z_j herhangi bir katmandaki $j.$ nörondaki net toplamı ifade etsin. $h\ (z_j)$ aktivasyon değeri eğer son katmanda değilse gizli birimde (hidden unit) yer alır. Son katmanda ise çıktı katmanı olarak adlandırılır. Örneğin: $h(.)$ gizli katmandaki aktivasyon fonksiyonunu, $\sigma(.)$ çıktı katmandaki aktivasyon fonksiyonunu göstermek üzere gözlem altında bir öğrenme için iki katmanlı bir yapay sinir ağının modeli:

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma(\sum_{j=1}^{M} w_{kj} h\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0})$$

Olarak yazılır. $y_k(\boldsymbol{x}, \boldsymbol{w}), w$ ağırlıkları ve x girdi vektörü verildiğinde çıktı katmanında oluşacak k boyutlu vektörün gösterimidir. Örneğin Türkiye için araba plakası tanıyacak bir sistem için plaka görüntülerinden resimler girdi x ise; k = 81 il için ayrı katagorik çıktı oluşacaktır. Eşitliğin son satırı en içteki katman 1. katman en dıştaki katman 2. katman olduğunu göstermek üzere yazılmıştır. Çıktı katmanı normal regresyon problemleri için $h(.)$ birim fonksiyon olacak şekilde alınır.

Verilecek ağırlıkları güncelleme işlemi backpropagation (geri yayılım) terimi ile ifade edilir. Matematiksel olarak sinir ağları, yapay sinir ağları çalışırken ileri geçiş (forward pass) ve geri geçiş (backward pass) olarak iki aşamada incelenebilir. İleri geçişte işlem yönü girdilerden çıktılara doğrudur, ve ileriye yayılım (forward propagation) olarak adlandırılır (Bishop 2006). her bir adımda yanlılık ve ağırlıklar, her bir nörondaki aktivasyon çıktılarıyla çarpılarak matematiksel değer üretilir; son nörona gelindiğinde ise çıktının istenen değer ile farkına bakılır, bu fark

$$E_k = y_k - \sum_{j=1}^{M} w_j \phi_j(x_1, x_2, \ldots x_d) + w_0 = y_k - \hat{y}_k$$

ile gösterilir. Hata miktarı önceden belirlenen ve kabul edilebilir bir mutlak değerden çoksa girdi olarak alınan w_j ağırlıkları daha az mutlak hata vermesi amacıyla geri yayılım (backpropagation) çalıştırılarak güncellenir. Geri yayılım, kabul edilemeyecek büyüklüklerdeki hata miktarıyla tetiklenen ve geriye doğru işleyen öğrenmenin

güncellenmesi işlemidir. İleriye doğru öğrenme sırasında ortaya çıkan net toplam, ağırlıkların katsayı olduğu doğrusal bir fonksiyondur. Bu güncelleme net toplamın son durumuna türevde zincir kuralı uygulanarak yapılır. Bu işlem ϕ_j baz fonksiyonlarının türevlenebilir olmasını gerektirir. Eğer böyle değilse öğrenmenin amacını bozmayacak şekilde baz fonksiyonlarına yakın olan türevlenebilir ϕ_j fonksiyonları belirlenmelidir. Bu güncellemeler sonucunda oluşan yeni ağırlıklar ile tekrardan ileri geçiş çalıştırılıp yapılan iş bir döngü halinde istenilen sonuca yaklaşıncaya kadar devam ettirilir. τ adım sayısı, her τ geri yayılımındaki güncel ağırlık vektörü $w^{(\tau)}$ ile gösterilmek üzere elde edilen E_k' deki *değişim miktarı* $\nabla E_k(w^{(\tau)})$ vektörü ile gösterilirse bir sonraki $(\tau + 1)$. adımda güncellenmiş olan ağırlık vektörü

$$w^{(\tau+1)} := w^{\tau} - \eta \nabla E_k(w^{(\tau)})$$

olarak elde edilir. Güncelleme $\nabla E_k(w^{(\tau)})$ gradientinin ters işaretlisi -$\nabla E_k(w^{(\tau)})$ ile yapılır; hata negatif ise ağırlıklar artırılarak, pozitif ise ağırlıklar azaltılarak yapılır. Ancak hatanın en az olabileceği ağırlık vektörü güncelleme sırasında atlanabilir; (iterasyon işleminde gözden kaçmış olabilir). Bu durumun yaşanmaması için gradientin adım büyüklüğü optimal olduğu düşünülen ve önceden deneme yanılma ile belirlenebilir. Böylece η hiperparametresi ayarlanmış olur. Tüm öğrenme verilerine göre elde edilen hataların toplamı, çıktı kategori sayısı K ile gösterilmek üzere

$$E = \sum_{k=1}^{K} E_k = y - \hat{y}$$

öğrenme işinin toplam hatasını gösterir. Nöron sayısı ve girdi miktarı arttıkça, yapılan işlem sayısı ve karmaşıklık üstel olarak artar, bu da oldukça fazla işlem gücü gerektirir. Geri yayılım için; E hata fonksiyonu w_{ji}^{l} l'inci katmandaki j'inci nöronu ile i'inci nörona bağlayan ağırlıkları, z_j^l l'inci katmandaki j'inci nörona ait aktivasyonu ifade etmek üzere bu zincir:

$$\frac{\partial E}{\partial w_{ji}^{l}} = \frac{\partial E}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{ji}^{l}}$$

olarak yazılır. burada ∂ önüne geldiği terime göre kısmi türev işlemini göstermektedir. Kısmi türev alma ve bunların her bir ağırlıkta güncellenmesi işlemi oldukça zaman ve işlemci gücü gerektirmesi, öğrenmenin doğruluğundan ödün verilerek daha hızlı sonuç veren güncelleme biçimleri geliştirilmesini gerektirmiştir (Bishop 2006). Buna örnek olarak batch (grup) öğrenme verilebilir. Veri sayısına göre örneğin bütün öğrenme seti ξ

parçaya bölünüp; güncelleme öğrenme verisinde sadece $(\xi + 1)$. veriye gelindiğinde uygulanır. İlk ξ grubunda öğrenme seti aynı ağırlıklar ile çarpılarak bütün hataların ortalaması $(\xi + 1)$. veride geri yayılım çalıştırılmasıyla güncellenebilir.

Aşağıdaki örneğin verilmesinin iki amacı vardır: birincisi bir veri işlemede verilerin tümü n çaplı bir örneklem olarak elde olduğunda ve verilerin tek tek gelmesi durumunda istatistiksel çıkarımın (tahminin) nasıl yapılabileceğini gösterilmesidir. Verilerin tek tek gelmesi istatistikte ardışık çıkarımın da bir konusu olmakla birlikte, konudan sapmamak amacıyla detaya inilmemiştir. Diğer amaç perseptron algoritmasının çok sayıda nörona ve katmana karşılık gelen sinir ağları içine bir işlem birimi olarak kullanılmasına örnek vermek olacaktır. Bu durum tıpkı biyolojik birimlerde nöronların birbirlerine bağlandığı gibi perseptronlar birbirlerine bağlanacak ve "öğrenmeyi ve bilgi akışını" zenginleştirecek, karar vermeyi kolaylaştıracaktır.

Aşağıdaki $n = 12$ olan bir veri setinde x ve Y değişkenlerine ait gözlemler vardır. Y bağımlı rasgele değişkeni, x açıklayıcı değişkeni verilmiş sabit olmak üzere; $Y = a + bx + \epsilon$ modeli ele alınacak, ϵ hata terimi ihmal edilerek parametre tahminleri yapılacaktır. İlk aşamada verilerin tümü birlikte alınıp, tahminler alışılagelen istatistik metodolojisine göre elde edilecektir. İkinci aşamada yine aynı modelin, gözlemlerin teker teker tahmin sürecine alındığı azalan gradyan yöntemi ile tahminlerin elde edilmesi örnekle anlatılacaktır **(Çizelge 2.1)**.

Çizelge 2.1 – x ve Y değişkenlerine ait gözlemler

x: 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42
Y: 8.4, 9.5, 11.8, 10.4, 13.3, 14.8, 13.2, 14.7, 16.4, 16 5, 18.9, 18.5

İstenen basit doğrusal regresyon modeli aşağıdaki varsayımlar altında istatistiksel olarak elde edilecektir. $Y = a + bx \;\; N(0, \sigma^2)$ normal dağılımlı iid Buna göre $\hat{Y} = 0.28928 + 0.45664x$ bulunmaktadır. Azalan gradyan algoritması için formül:

$$w^{(\tau+1)} := w^{\tau} - \eta \nabla E(w^{(\tau)})$$

olmak üzere τ adım sayısını, η burada öğrenme oranı sabitini, w (1xn)'lik parametre vektörünü, ∇ türev operatörünü, E fonksiyonu hata fonksiyonlarını ifade eder. Burada

hataların açılımı, w parametreler (1xn)'lik vektör olmak üzere ve N tüm hatalar sayısı olmak üzere bağımlı rasgele değişkenler ile açıklayıcı değişkenler arasındaki farktır. Yukarıdaki verilen veri seti durumunda; iki parametreli basit doğrusal regresyon modeli için azalan gradyan uygulanması için formülün açık hali:

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(\tau+1)} := \begin{bmatrix} a \\ b \end{bmatrix}^{(\tau)} - \begin{bmatrix} \eta(Y - a - bx_i) \\ \eta(Y - a - bx_i)x_i \end{bmatrix}$$

dönüşür. Gradyan azalış algoritmasının çalışabilmesi için sabit değerler ve parametrelerin başlangıç değerleri önceden belirlenmelidir. Bunun için bu problemde sabit başlangıç değerleri şu şekilde seçilmiştir: öğrenme oranı = 0.01, adım sayısı = 1. Parametrelerin başlangıç değerleri ise $a = 0, b = 0$ seçilmiştir. İlk veri çifti $(x, Y) = (20,8.4)$ olarak geldiğinde;

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^{(0)} - \begin{bmatrix} 0.01(8.4 - 0 - 0) \\ 0.01(8.4 - 0\_0)20 \end{bmatrix}$$

olacaktır. Bu matris hesaplandığında [0,084 1.68] ilk sonucu elde eder. İkinci veri çifti $(x, Y) = (22,9.5)$ olarak işleme konulduğunda ise matrisler:

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.084 \\ 1.68 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-9.5 + 0.084 + 1.68) \\ 0.01(-9.5 + 0.0084 + 1.68)22 \end{bmatrix}$$

şeklinde oluşacaktır. Bu durumda bir sonraki üçüncü veri çiftine verilecek matris [0.00664 0.02192] olacaktır. Üçüncü çift $(x, Y) = (24,11.8)$

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.00664 \\ 0.02192 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-11.8 + 0.00664 + 0.02192) \\ 0.01(-11.8 + 0.00664 + 0.02192)24 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.1111 2.8044]'tür. Dördüncü çift $(x, Y) = (26,10.4)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.1111 \\ 2.8044 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-10.4 + 0.1111 + 2.8044) \\ 0.01(-10.4 + 0.1111 + 2.8044)26 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0362 0.8584]'tür. Beşinci çift $(x, Y) = (28,13.3)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.0362 \\ 0.8584 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-13.3 + 0.0362 + 0.8584) \\ 0.01(-13.3 + 0.0362 + 0.8584)28 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0880 2.6151]'tür. Altıncı çift $(x, Y) = (30,14.8)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.0880 \\ 2.6151 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-14.8 + 0.0880 + 2.6151) \\ 0.01(-14.8 + 0.0880 + 2.6151)30 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0330 1.0206]'tür. Bu çift $(x, Y) = (32,13.2)$

yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.033 \\ 1.0206 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-13.2 + 0.0330 + 1.0206) \\ 0.01(-13.2 + 0.0330 + 1.0206)32 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.088 2.8670]'tür. Sekizinci çift $(x, Y) = (34,14.7)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.088 \\ 2.867 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-14.7 + 0.088 + 2.8670) \\ 0.01(-14.7 + 0.088 + 2.8670)34 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0305 1.1696]'tür. Dokuzuncu $(x, Y) = (36,16.4)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.0305 \\ 1.1696 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-16.4 + 0.0305 + 1.1696) \\ 0.01(-16.4 + 0.0305 + 1.1696)36 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.1221 4.3066]'tür. Onuncu çift $(x, Y) = (38,16.5)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.1221 \\ 4.3066 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-16.5 + 0.1221 + 4.3066) \\ 0.01(-16.5 + 0.1221 + 4.3066)38 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0008 0.0002]'tür. On birinci çift $(x, Y) = (40,18.9)$ yerleştirilirse

$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.0008 \\ 0.0002 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-18.9 + 0.0008 + 0.0002) \\ 0.01(-18.9 + 0.0008 + 0.0002)40 \end{bmatrix}$$

Cevap matrisi olarak bu adım [0.0188 4.5594]'tür. Son çift $(x, Y) = (42,18.5)$ yerleştirilirse

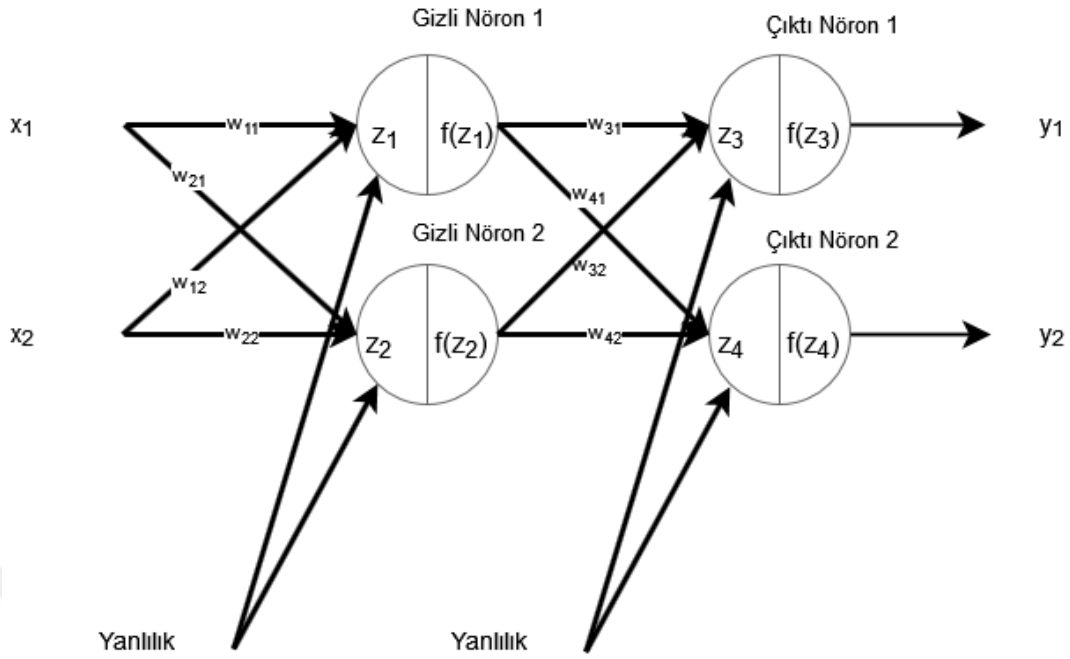
$$\begin{bmatrix} a \\ b \end{bmatrix}^{(1)} = \begin{bmatrix} 0.0188 \\ 4.5594 \end{bmatrix}^{(0)} + \begin{bmatrix} 0.01(-18.5 + 0.0188 + 4.5594) \\ 0.01(-18.5 + 0.0188 + 4.5594)42 \end{bmatrix}$$

Bir adımlık gradyan azalışın sonuç matrisi [0.0506 1.2166] olarak bulunur. Bu sonuç virgülden sonra dört basamak gözetilecek şekilde hesaplandığı için virgülden sonra altı basamak gözeten bir bilgisayar çıktısına göre daha az hassastır. Altı basamakta sonuç [0.008400 1.680000] bulunmaktadır.

Aynı veri setinde bu işlemler öğrenme oranı = 0.01, adım sayısı = 10000. Parametrelerin başlangıç değerleri ise $a = 0, b = 0$ seçilen başka bir deneyde i5 ikinci nesil işlemcide 2.083 saniyede sonuç vermiştir. Burada batch (bir dizi) sayısı alımı ile yapılan hesaplamanın gradyan azalışta farkı olmadığı gösterilmek istensin. Aynı hiper parametreler kullanılıp bir dizi elde edilsin. Tek döngüde 2.083 saniyede $a = 0.0.008400, b = 1.680000$ olarak sonuç vermiştir. Bir diğer deneyde ise yığın sayısı

dörderli olduğunda $a = 0.020948$ ve $b = 0.418953$ olmaktadır. Halbuki adım sayısı yeteri kadar arttırıldığında ve zaman kıstası ortadan kaldırıldığında gradyan azalış algoritmaları birbirine benzer sonuçlar vermektedir. Öyle ki süre kıstası ortadan kalkarsa $a = 0.289222$ ve $b = 0.45665$ bulunmaktadır. Bütün C dili kodları aynı makinede 323.11 KiB yer kaplamış, 0.20 saniyede derlenmiş ve 9.69 saniyede yukarıda verilen bu cevap elde edilmiştir. Gradyan azalışla görüldüğü gibi her bir t anında cevap elde edilebilmektedir ancak bu cevabın bilgisayarın diziyi kaçar kaçar aldığı yapılan işin sonucunu değiştirmemektedir. Aksine a ve b'in yakınlığı öğrenme oranı ve adım sayısına bağlıdır. Uygulamada ise bu yığın seçimi paralel işlemcilerin ve uygulamanın yazıldığı dilin derleyicisinin hafızadan tasarruf ve işletim sisteminin daha hızlı yazmaç kullanma adına işlemcinin farklı bölümlerine göre optimize edilmesine göre modern işlemcilerde oldukça farklı zamanlarda çıktı verebilmesine yol açmaktadır. İşlemcilerin veri yolları ve işlem birimlerinin büyüklüğüne göre de çalışma zamanı değiştirmesinden dolayı en basit gradyan azalış probleminde bile hiperparametre seçiminde dikkatli olmak gerekmektedir.

Yapay sinir ağlarının çalışmasının incelenmesi için de iki elemanı olan bir veri seti için ($i = 1,2$) ile iki katmanlı, dört nörondan oluşan bir sinir ağı verilmiş olsun. Gözlem altında öğrenme ile girdi vektörü $x = [x_1, x_2]$= [0.65,0.15] ve hedef çıktı vektörü $y = [y_1, \ y_2] = [0.99, \ 0.01]$ olarak verilmiştir. İki gizli nöronlu ve iki çıktı nöronlu bir yapay sinir ağı aşağıda verilmiş olup sistemin ağırlık vektörleri rasgele olarak başlatılmayıp, ilk katmanda w_1 =[w_{11}, w_{12} w_{21} w_{22}] = [0.1,0.2,0.2,0.1] ve w_2 =[w_{31} w_{32} w_{41} w_{42}] = [0.1,0.3,0.1,0.3] sabit olarak verilmiş olsun. Yapay sinir ağlarının çalışabilmesi hiperparametreler, öğrenme oranı η, döngü sayısı τ, kullanılacak aktivasyon fonksiyonu f ve yanlılık yani bias yapay sinir ağı çalışmadan önce belirlenmelidir. Gizli ve çıktı nöronlarının yanlılık oranları yani bias = 0.45 olsun. Döngü sayısı τ = 1, öğrenme oranı η = 0.1 ve kullanılacak aktivasyon fonksiyonu f sigmoid olsun.



Şekil 2.2 İki katmanlı, tam bağlı ikisi gizli 4 nörondan oluşan, gözlem altında öğrenme yapan yapay sinir ağı

Şekil 2.2'de verilen ağ yapısı çizimde gösterilmiş olup; her bir j. nöronda oluşan hesaplama sonucu (net sum) aşağıdaki gibi ifade edilir:

$$z_j = \sum_{i=1}^{2} x_i \, w_{ji} + bias$$

Burada z_j her bir katmandaki j'inci nöronun sayısal değerini, bias terimi yanlılık sabitlerini, x girdi vektörünün bileşen indeksi i olmak üzere x vektörü bir önceki adımdaki girdi vektörünü ve w_{ji} 'de gizli katmandaki j'inci nörona ait her bir x vektörünün i. elemanından gelen ağırlık katsayılarını ifade etmektedir. Örneğin birinci ve ikinci nöron için;

$$z_1 = 0.1 \times 0.65 + 0.2 \times 0.15 + 0.45 = 0.545$$

$$z_2 = 0.2 \times 0.65 + 0.1 \times 0.15 + 0.45 = 0.595$$

bulunur. Daha sonra bu değerler $f(z_j)$ şeklinde sigmoid fonksiyonunda işlenir:

$$f(z_j) = (1 + e^{-z_j})^{-1}$$

$f(z_1) = 0.633$ ve $f(z_2) = 0.6445$ bulunur. Sonra bu nöronların sonucları kendisinden sonra gelecek nöronlar için girdiler olacaktır. Yukarıdaki işleyişte bu sonuçlar çıktı nöronlarına girdi olacaktır. Öyle ki çıktı nöronlarının değerleri ikinci katmanın ağırlıklarıyla işlem görecektir:

$$z_3 = 0.633 \times 0.1 + 0.6445 \times 0.1 + 0.45 = 0.5777$$

$$z_4 = 0.633 \times 0.3 + 0.6445 \times 0.3 + 0.45 = 0.8333$$

Bu nöronlara da öncekiler gibi sigmoid fonksiyonuna sokulduğunda $\hat{y}_1 = f(z_3)$ = 0.6405, $\hat{y}_2 = f(z_4)$ = 0.6971 olarak bulunur; dolayısıyla çıktı vektörü $\hat{y}_k$ = [0.6405 0.6971] olarak ifade edilir. Bu değerler ile hedef vektörü y_k = [0.99 0.01]'ne yeterince yakın olmadığı için bir sonraki adımda hesaplanan değer ile istenen değer arasındaki fark bulunup, backpropagation (geri yayılım) uygulayarak başlangıçtaki ağırlık vektörleri değiştirilecektir. Hesaplanan ve hedef değer arasındaki fark hata kareler ile şu şekilde ifade edilir:

$$Hata(\hat{y}_k) = \frac{1}{2}(y_k - \hat{y}_k)^2$$

Toplam hata ise hata vektörün tüm bileşenleri toplanarak elde edilir.

$$Toplam\ hata = hata(\hat{y}_1) + hata(\hat{y}_2) \ldots + hata(\hat{y}_k) = \sum_{k=1}^{K} hata(\hat{y}_k)$$

$$Hata(\hat{y}_1) = 0.5 \times (0.99 - 0.6405)^2 = 0.0611$$

$$Hata(\hat{y}_2) = 0.5 \times (0.01 - 0.6971)^2 = 0.2361$$

Yukarıdaki örnekteki döngü için toplam hata 0.2972 olur. Yapılan toplam hata kullanıcı için yeterince az ise başka döngüye ihtiyaç duyulmaz, eğer ihtiyaç varsa döngü geri yayılım yapılarak devam ettirilir. Geri yayılım amacı her katmandaki ağırlıklar yukarıda verilen türev ve yorumu dikkate alarak güncellenecektir. Geri yayılım türevlere dayanıyor olup, aktivasyon fonksiyonunun türevlenebilir olması gerekmektedir. Yukarıdaki süreç tersine yönlendirilerek geri yayılım yapılır. Geri yayılımdaki kısmi bileşenler, kısmi türevde zincir kuralı yardımıyla yazılabilir durumda olmalıdır. Burada geri yayılımda ağırlık adlandırmada tanımlar girdi çıktı olarak korunacaktır; yani *j* ve *i* indisleri kendi aralarında yer değiştirmez. Geri yayılımda yön tersine çevrilirse ağın yönü daima ileriye doğru olur; ancak ağırlıklardaki girdi ve çıktı nesneleri olduğu gibi kalır. Bir nöron için bu geri yayılım η öğrenme oranı olmak üzere *j*. nörona *i*. kaynaktan (nöron, girdi veya geri yılımda çıktı) gelen ağırlığın değişim miktarı şu şekilde ifade edilebilir:

$$\Delta w_{ji} = -\eta \times \left(\frac{\partial Toplamhata}{\partial f(z_j)}\right) \times \left(\frac{\partial f(z_j)}{\partial z_j}\right) \times \left(\frac{\partial z_j}{\partial w_{ji}}\right)$$

$$\Delta w_{ji} = -\eta \times (\frac{\partial Toplamhata}{\partial f(z_j)}) \times (\frac{\partial f(z_j)}{\partial z_j}) \times (\frac{\partial z_j}{\partial w_{ji}})$$

Sayısal örnek devam ettirilir ve w_{11}'in yeni ağırlığını hesaplamak için sondan başlanmalıdır, öyle ki ilk önce w_{31} hesaplanmalıdır. Burada w_{31} ifade edildiği üzere 3 nolu nörona yani 1 nolu çıktı nöronuna giren 1 nolu girdi nöronundan gelen ağırlık olarak adlandırmaya devam edilecektir. w_{31} için değişim miktarı Δw_{31} şu şekilde hesaplanır:

$$Toplamhata = Hata(\hat{y}_1) + Hata(\hat{y}_2)$$

$$Toplamhata = \frac{1}{2} \times (y_1 - f(z_3))^2 + \frac{1}{2} \times (y_2 - f(z_4))^2$$

$$\frac{\partial Toplamhata}{\partial f(z_3)} = 2(-1)\left(\frac{1}{2}\right)\left(y - f(z_3)\right)^{(2-1)} + 0$$

$$= -(0.99 - 0.6405) = -0.3495$$

Sigmoid fonksiyonu için kısmi türev:

$$\frac{\partial f(z_3)}{\partial z_3} = f(z_3) \times (1 - f(z_3)) = 0.6405(1 - 0.6405) = 0.2303$$

$$z_3 = w_{31} \times f(z_1) + w_{32} \times f(z_2) + bias$$

$$\frac{\partial z_3}{\partial w_{31}} = w_{31}^{(1-1)} \times f(z_1) + 0 + 0 = f(z_1) = 0.633$$

Öğrenme oranı η = 0.1 olarak yukarıda verildiğine göre bulunan bütün kısmi türevler ile birlikte yukarıda yerine yazılırsa değişim miktarı w_{31} için :

$$\Delta w_{31} = -0.1 \times 0.3495 * 0.2303 \times 0.633 = -0.005095$$

olarak bulunur. Bu değerin güncellenmesi sistem üzerindeki diğer ağırlıklar da bulunduğunda yerine konulur. Burada w_{11} için toplam hata için benzer işlemlerle kısmi türevler şu şekilde bulunur:

$$\Delta w_{11} = -\eta \times (\frac{\partial Toplamhata}{\partial f(z_1)}) \times (\frac{\partial f(z_1)}{\partial z_1}) \times (\frac{\partial z_1}{\partial w_{11}})$$

$$\frac{\partial Toplamhata}{\partial f(z_1)} = \frac{\partial hata(\hat{y}_1)}{\partial f(z_1)} + \frac{\partial hata(\hat{y}_2)}{\partial f(z_1)}$$

$$= (-0.00804 + 0.0158)$$

$$= 0.015$$

$$\frac{\partial hata(\hat{y}_1)}{\partial f(z_1)} = \frac{\partial hata(\hat{y}_1)}{\partial f(z_1)} \times \left(\frac{\partial f(z_1)}{\partial z_1}\right) \times \left(\frac{\partial z_1}{\partial f(z_1)}\right)$$
$$= -0.3495 \times 0.2303 \times 0.1$$
$$= -0.00804$$

$$\frac{\partial hata(\hat{y}_2)}{\partial f(z_1)} = \frac{\partial hata(\hat{y}_2)}{\partial f(z_1)} \times \left(\frac{\partial f(z_1)}{\partial z_1}\right) \times \left(\frac{\partial z_1}{\partial f(z_1)}\right)$$
$$= -(0.01 - 0.6971) \times 0.2303 \times 0.1$$
$$= 0.0158$$
$$\Delta w_{11} = -0.1 \times 0.0158 \times 0.633 \times (1 - 0.633) \times 0.65$$
$$= 0.00023$$

Sonuç olarak yeni ağırlıklar aşağıdaki formülde yerine konur ve güncellemeler elde edilir:

$$w_{ji} := w_{ji} - \Delta w_{ji}$$

Örneğin birinci döngüde w_{11}'in güncel hali = $0.1 - 0.00023$ = 0.09977 olur. w_{31}'de 0.10509 bulunur. Bu yeni durumda ilk katmana w_1 = [0.09977 0.19987 0.19947 0.09987] ve ikinci katman w_2 = [0.10509 0.10518 0.29081 0.29064] şeklinde güncelleme getirilmiş olur. Ancak burada tek döngüde sistemin toplam hatası 0.2948 çıkar; nitekim bu çok büyük bir orandır. Bütün sistem 10000 defa döngü ile ağırlıkların güncellenmesiyle ancak belirli bir seviyeye kadar istenen hata seviyesine inebilir. 10000 döngüde yukarıdaki durumda 0.00018224 toplam hata elde edilmiştir. 10000 döngü sonunda ise öğrenilmiş ağırlıklar ilk katmanda w_1 = [1.13787 0.43951 1.23564 0.33900] ve ikinci katmanda w_2 = [2.07304 2.10321 2.65180 2.69921] olmaktadır. Sistemin toplam hatası istenilen seviyeden düşükse makine öğrenme süreci durdurulup, öğrenilmiş ağırlıkların ve oluşmuş sinir ağlarının yapısı benzer başka bir problemin çözümünde kullanılmak üzere saklanır.

Yukarıda iki boyutlu bir gözlem vektörü ile iki katmanlı bir yapay sinir ağı ile öğrenme yapılmıştır. Ilk döngüdeki 0.2972 olan toplam hata 9999 döngüden sonra yaklaşık olarak 0.00002 toplam hatası olarak gerçekleşmiştir. Öğrenme için kullanılan tek veri vektörü için $\hat{y} = [\hat{y}_1, \ \hat{y}_2]$, her katmanında 2 nöron bulunan iki katmanlı yapay sinir ağı (neural network) kullanılarak öğrenme tamamlanmıştır. Bu öğrenmeye dayalı olarak

yeni bir $x = [x_1, x_2] = [0.40,0.30]$ gözlemi yapıldığında bu yeni veri bir test işlemi olarak dikkate alınır ve iki katmandaki öğrenilen ağırlıklar ve sigmoid fonksiyon kullanılarak ilk katmandan

$$z_1 = 1.13787 \times 0.40 + 1.23564 \times 0.30 + 0.45 = 1.27584$$

$$z_2 = 0.43951 \times 0.40 + 0.339 \times 0.30 + 0.45 = 0.727504$$

$$f(z_1) = (1 + e^{-z_1})^{-1} = 0.781741, \qquad f(z_2) = (1 + e^{-z_2})^{-1} = 0.674257$$

elde edilir. Çıktılar ikinci katmanda

$$z_3 = 2.07304 \times 0.781741 - 2.65180 \times 0.674257 + 0.45 = 0.282586$$

$$z_4 = 2.10321 \times 0.781741 - 2.69921 \times 0.674257 + 0.45 = 0.274204$$

$$f(z_3) = (1 + e^{-z_3})^{-1} = 0.57018, \qquad f(z_4) = (1 + e^{-z_4})^{-1} = 0.568125$$

$\hat{y}_{test} = [\hat{y}_1, \quad \hat{y}_2] = [0.57018, \ 0.568125]$ bulunur. Sonuç olarak 10000 döngü sonucu oluşan modele uygulanan test verisi ile 0.00035 hata elde edilirken 1 döngü sonucu modele uygulanan test veri setinden 0.29235 hata elde edilmektedir. Burada yapay sinir ağı için bir öğrenme söz konusudur. Girdi sayısından bağımsız olarak ve değişik yan değerleri kullanılarak veya nöron sayıları ve katman sayıları değiştirilerek hata değerlerinde küçülme gözetlenerek bu ağ geliştirilebilir ve öğrenme sonucundan algoritmayı geliştiren kişi memnun oluncaya kadar bu yenileme ve seçme kararları devam eder. Öyle ki yeni gelecek veri setleri ile mevcut bulunan öğrenilmiş sinir ağına yeni girdiler üst üste verilerek de öğrenmeye katkı yapmak mümkündür (Daumé 2017).

2.3 Gözlem Altında Olmayan (Unsupervised) Öğrenme

Gözlem altında olmayan öğrenme; gözlemci ile öğrenmenin aksine etiketlenmemiş veri ile öğrenmeye çalışan yöntemlere denir. Algoritma doğru verilmiş sınıfları bilmediği için bu çıkarımları kendisinin yapması beklenir. Yani $S = (x_1, x_2 \ldots x_n)$ ile öğrenme algoritması oluşturulur. Etiketlerin yok olması ile birlikte algoritmaların ölçüm yapabileceği kayıp fonksiyonu ortadan kalkmış olur. Kullanımları arasında gruplama ve aykırı değer tespiti bulunur. Gruplama benzer verileri bir araya toplama işi; aykırı değer tespiti ise veri setindeki diğer verilerle uyuşmayan verilerin belirlenmesi işidir. Avantajları arasında etiketlerle uğraşmayarak insan emeğini ve yanlışlarını ortadan kaldırır. Buna ek olarak insanın fark edemeyeceği örüntüleri yakalamada gözlem altına göre daha iyi sonuç verir ve boyut azaltılması ile de idealdir. Dezavantajları olarak

zaman ve algoritma karmaşıklığı artar (Bishop 2006).

2.4 Pekiştirmeli (Reinforcement) Öğrenme

Pekiştirmeli öğrenme, ardışık kararlarla makine öğrenme gerçekleştiren yöntemlerdir. Gözlem altında öğrenme veya gözetimsiz öğrenmeden farklı olarak deneme yanılma ve öğrenilmek istenen olgunun süreç içinde tanımlanabilmesi beklenmektedir. Bu yöntemde öğrenilecek olgu; karar kümesi, karar, karar verici ve kayıp/kazanç gibi terimleri içerecek şekilde modellenmeye uğraşılır. Bundan dolayı diğer yöntemlere göre modellenmesi ve algoritmalarının hızlı biçimde oluşturulması genelde zordur (Bishop 2006).

2.5 Makine Öğrenme Yöntemlerinin Karşılaştırması

Bütün makine öğrenme problemlerinde, ilgilenilen problemin veri boyutu ve problemin karmaşıklığı arttıkça eldeki verilerde de saçılma görülür. Bu sebeple geliştirilen her bir yöntemde daha fazla özellik öğrenmek isteniyorsa, bu özelliğe ek olarak daha büyük sayıda veriye ihtiyaç duyulur; fakat veri sayısı arttıkça işlemlerdeki efor, hesabı tutulması gereken veri ve depolamanın katlanarak artması problemleri ortaya çıkar. Boyut arttıkça verilerin saçılımı seyrekleşmeye, her bir boyutta yapılacak işlem artmaya,bulunan benzerlikler önemini yitirmeye başlar. Bu fenomene boyutsallık laneti denir (Daumé 2017). Sorunu çözmek için özellik sayısını düşürmek (boyut indirgeme), ölçüm araçlarını geliştirmek, gereksiz verileri temizlemek, birbirine benzer özellikleri çıkarıp tekrardan eğitmek gibi akıl yürütmelerle sorun çözülmeye çalışılır. Ancak gereksiz ve alakasız özellikler eğitim süresinin uzamasına ve sonuçların kirlenmesine yol açmaktadır. Bu sebeple yanlılık ve varyans arasında bir seçim yoluna gitmek gerekmektedir. Günümüzde benzer şekilde tek bir makine öğrenme algoritması uzun eğitim sürelerine yol açtığından, aynı anda birden çok yöntem kullanılarak bir tekniğin açığı diğer teknikle kapatılmaya çalışılmaktadır. Bunun sonucunda veri sayısının ve boyutlarının da artmasıyla derin öğrenme metodları doğmuştur. Yapay sinir ağları; yetersiz bilgi olduğunda bile, diğer algoritmalara göre iyi sonuçlar verecek şekilde çalışması, hataya karşı töleransının yüksek olması dolayısıyla hafızanın dağıtık olması, bilgisayarlar için paralel hesaba yatkın olması ve maliyetlerin günden güne düşmesi

bakımından diğer makine öğrenme yöntemleri arasında farklı bir yere sahiptir. Bunların yaninda bilginin, her bir anda ağırlıklarda nereye gidip, nerede olduğu belirlenebilecek şekilde göz önünde olması sayesinde güvenilir olduğundan derin öğrenme uygulamalarında diğer makine öğrenme yöntemlerine göre günümüzde daha ön plandadır (Bishop 2006).

3. ÇATIŞMALI MAKİNE ÖĞRENME

Makine öğrenme algoritmalarını veya sistemlerini yanıltmak, belirli hedefler doğrultusunda yapılmış sınıflandırmanın güvenirliğini düşürmek veya sınıflandırılmış bilgileri değiştirme işi gibi uğraşlar çatışmalı makine öğrenmenin alanına girer. Çatışmalı makine öğreniminde verilerin ve algoritmaların kimin elinde olup, kimin ele geçirip değiştirmeye çalıştığına göre taraflar belirlenir ve isimlendirilir. Saldırgan; tehdidi oluşturan, verilere ve algoritmalara dolaylı ya da dolaysız yoldan etki edip, öğrenme sonucu oluşmuş modele yanlış kararlar verdirmeye zorlayan taraftır. Savunan; her bir saldırgana karşı veriyi ve algoritmayı koruyan, ataklara karşı geliştirdiği yöntemler ve makine öğrenme algoritmasında yaptığı iyileştirmelerle hem saldırıyı önceden tespit etmeye uğraşan hem de önleme faaliyetlerinde bulunan taraftır. Bu iki tarafın birbirine zıt amaçları olduğundan devamlı bir çatışma hali söz konusu olup; her zaman önce geliştirilen yönteme karşı onu aşacak yöntemler bulunmaya çalışılan bir güvenlik problemi alanı oluşur. Makine öğrenmenin güvenliğine saldırgan tarafından bakılırsa, saldırganın genel olarak iki amacı olduğu söylenebilir: saldırgan hem savunan tarafa yakalanmaktan kaçmaya çalışırken hem de değiştireceği karar veya verileri savunan tarafa masum göstermeye çalışır. Bu durum saldırgan tarafında iki karşıtlık meydana getirmektedir. Masum gözükmeye çalışarak yapmaya uğraştığı amacından ödünler verecek ya da saldırıda kullandığı yöntemleri geliştirerek kendisinden daha çok zaman ve bilgi isteyen yeni yöntemler geliştirmeye çalışacaktır. Makine öğrenmenin güvenliğine savunan tarafından bakılırsa da durum benzer şekildedir. Daha fazla güvenlik için eklenen her kıstas öğrenmenin kullanılacağı ortamda yavaşlamalara yol açabilecek, yapılan işin güvenliği için de ek maliyet ve iş gücü ayrılmasını gerektirecektir. Aynı zamanda bu alanda yapılan her bir çalışma da farklı veri setleri ve farklı yöntemler temel alınarak oluşturulmuş makine öğrenme algoritmalarında değişik sonuçlar vermesine yol açar. Öyle ki bazı modeller diğerlerine göre saldırganların sonuç elde etmesine daha yatkın iken, model öğrenilirken kullanılan veri setleri değiştirildiğinde bu sefer saldırıların savuşturulmasına yol açabilir ya da bu durumun tam tersi olarak veriler sabit iken saldırıya açık olan bir öğrenme, model değiştirildiğinde sağlam bir hale gelebilir. Makine öğrenmenin doğası gereği de geliştiricinin her bir yaptığı değişiklikte güvenliğin ne kadar sağlam olacağı önceden kestirilemez. Gün geçtikçe ilerleyen ve yeni geliştirilen makine öğrenme ve derin

öğrenme algoritmaları ile sorun katlanarak büyümektedir. Buna binaen saldırganlar için bilgi çalmaya ve değiştirmeye yönelik birçok yeni yöntem bulmaktadırlar. Bahsedilen olay bu sebeple siber güvenlik alanında oldukça fazla yer bulmuş ve yapılan saldırılar matematiksel ve istatistiksel olarak ifade edilip yorumlanmaya ve anlamlandırılmaya çalışılmıştır (Yuan vd. 2019). Bu sebeple çeşitli taksonomiler ile saldırılar birbirlerinden ayırt edilmeye çalışılır. Örneğin bu taksonominin on iki grupta veya dokuz grupta incelenmesi gibi çeşitli fikirler öne sürülmüştür. Buna göre zamanlama, bilgi ve hedef bakımından saldırının üç boyutu ele alınabilir. Zamanlama bakımından kısaca özetlenirse: bir saldırgan ya daha veriler öğrenme işlemine sokulmadan önce ya da öğrenme modeli oluşturulduktan sonra tekrardan eğitme veya sürekli deneme yanılmalarla sistemi bozma gibi iki dala ayrılır. Saldırganın saldırı niteliği, saldırı yapacağı model hakkında elde ettiği ya da elde edebileceği bilgiye de bağlıdır. Bilgi bakımından: saldırgan, sistem hakkında (hangi model, hangi veri seti, hangi hiperparametre vb.) önceden ne kadar bilgisi olduğunu ölçer. Saldırının amacı da saldırganın nasıl bir yol izleyebileceğini belirleyici bir diğer unsurdur. Hedef bakımından ise: saldırgan, belirlediği bir hedefe mi yoksa bütün sistem çapına mı zarar vermek istediği gözlemlenmeye çalışılır (Yuan vd. 2019). Saldırgan, saldırı zamanlaması, model ve öğrenme-test verisi hakkındaki ön bilgisi ve saldırı amacı unsurlarını dikkate alarak bunların kombinasyonlarından birinin üzerine saldırısını biçimlendirir. Aşağıdaki sunumda bu unsurlar tek tek ve diğerlerinden soyutlanmış olarak ele alınacaktır.

3.1 Zamanlamaya Göre Ataklar: Zehirleme ve Kaçınma

Makine öğrenme algoritmalarına yapılan saldırılar saldırı zamanı bakımından sınıflandırılırsa, saldırgan, model öğrenilmeden önce ya da öğrenildikten sonra amaçlarını gerçekleştirmek için harekete geçebilir. Öğrenme işinden önceki (öğrenme ve\veya test) veri setini bozmaya yönelik yöntemler genel olarak zehirleme saldırısı ismini alırlar. Öğrenmeden sonra yapılan saldırı yöntemleri kaçınma saldırıları olarak isimlendirilir. Saldırıya maruz kalan modelin dolayısıyla sınıflama fonksiyonunun zaten belirlenmiş olduğu bu durumda saldırgan bu fonksiyona dokunmadan amaçlarını gerçekleştirmeye çalışacaktır. Bunu yapabilmesi ancak x öğrenme ve/veya test verilerinin "göze batmayacak" şekilde değiştirilip yanlış sınıflandırılması ile olanaklıdır.

Sınıflandırıcıdan kaçınma atakları şu şekilde açıklanabilir: x bir girdi, $f(x)$ bir sınıflandırıcı ve makine öğrenme sonucu x'in sınıflandırılması y olsun. Buradaki saldırganın sınıflandırma fonksiyonu $f(x)$, saldırganın kendi öngörü ve/veya model hakkındaki bilgilerine dayanarak, muhtemelen deneme yanılmalar yaparak belirlediği herhangi bir fonksiyon olabilir. $f(x) = y$ sonucunu verecek şekilde w ağırlıkları göstermek üzere örneğin $f(x) = sgn(w^T x)$, çok basit seçilmiş olabilir. Bu durumda x, $y = -1$ veya $y = 1$ olarak sınıflandırılacaktır. Saldırgan, x'in orjinali üzerinde yapılan çeşitli bozulmalarla elde ettiği yeni veri x''i kullanarak y şeklinde doğru sınıflanabilecek veriyi diğer $y' \neq y$ sınıfına atamaya uğraşır. Öyleki bu işlem $f(x') = y'$ şeklinde ifade edilebilir. Böylece x'ten x' elde edilip y' şeklinde yanlış sınıflandırma işi kaçınma atakları olarak adlandırılır. Saldırganın x' 'i elde etmesi ve matematiksel modellemesi oldukça karmaşık ve zaman gerektiren bir süreç olduğundan, bu saldırı tipinde birçok ideal x' adayı oluşturup deneme yanılma ile sınıfın değişip değişmediğini kontrol etmek önemlidir (Vorobeychik ve Kantarcıoglu 2018). Bu saldırı tiplerine karşı savunmalara örnek olarak elektronik posta servisi sağlayan kurumların koyduğu spam filtreleri verilebilir. "100000$ kazandınız! Ödülü almak için tıklayın" şekline içerisinde zararlı bağlantının bulunduğu zararlı bir e-maili ele alalım. Elektronik posta hizmeti veren firmalar filtreleri genellikle bu gibi saldırıları engellemek için birçok kötü amaçlı e-maili makine öğrenmesi kullanarak filtrelemeyi amaçlarlar. Bu sebeple yukarıdaki metin filtreye takılarak e-mailin kurbana gösterilmemesini sağlanır. Ancak saldırgan yeni e-mailler tasarlayarak bir sayısı yerine I harfi, sıfır sayısı yerine O harfi kullanarak, noktalama işareti olarak ünlem kullanmayarak vb. yazımlar ve bunların kombinasyonlardan yarattığı e-mailleri sürekli deneyerek filtrelerden kaçınmaya çalışması kaçınma saldırısına örnektir. Bir başka örnek olarak makine öğrenme ile çalışan yüz tanıma algoritmalarından maske, gözlük, makyaj, lens gibi yüzde değişiklikler yaratarak kişinin kendisini başka biri gibi tanıtmaya çalışması bu saldırı türüne örnek verilebilir (Yuan vd. 2019). Zehirleme atakları şu şekilde özetlenebilir: model eğitilmeden evvel x veri matrisini, model ağırlıklarını, hiperparametrelerini, x'lerin eşlendiği etiketleri vb. gibi birçok öğrenme bileşenine farklı değerler vererek uygulanan saldırılardır. Böylece savunan taraf olarak geliştiricinin elde ettiği $f(x)$'in sağlıklı sonuçlar vermemesi ve istenilen yanlış kararları verdirmesi saldırgan tarafından öğrenme yapılmadan önceden planlanır. Dışarıdan eğitim setine zararlı veri seti

karıştırmak, bir etiketi diğeri ile değiştirmek, öğrenme sırasında belirlenen ağırlıkları yer değiştirmek gibi birçok yöntemi vardır. Genellikle saldırgan için kolay olan bu yöntemler, makine öğrenmesi tamamlanmamış olduğu için hemen sonuç veremeyecek özellikteki saldırılardır. Savunan taraf açısından bakıldığında birçok veri arasında zehirlenmiş veri-bilgiyi arama ve belirleme güçlüğü zaman ve mali kayıplara yol açabilir. Örneğin; bir internet sitesi yöneticileri kullanıcılarının beğenilerinin sayısını öğrenme veri seti olarak toplayarak siteye giren kişilere site içindeki diğer bağlantılara yönlendiren önerilerde bulunan bir makine öğrenme sistemi oluşturmuş olsun. Protesto veya kötü amaçlı olarak kullanıcıların sürekli beğenmeme sayısını artırarak öneri sisteminin eğitim setini bozup, siteye yeni girenlerin yanlış bağlantılara yönlendirmesi bir örnek olarak verilebilir. Bu saldırılara başka bir örnek olarak otomatik sürüş için eğitilen bir arabanın kullandığı yolda giderken çarpılmaması istenen görüntüleri içeren eğitim setinin içinden trafik işaretleri silinerek veya bozularak ortaya çıkacak olan otomatik aracın sürekli tabelalara çarpan bir model oluşturması verilebilir (Yuan vd. 2019).

3.2 Model Ve Öğrenme-Test Verisi Hakkındaki Ön Bilgiye Göre Atak:Beyaz-Siyah Kutu

Çatışmalı makine öğrenmede diğer bir ayrım saldırganın saldırı yapılacak makine öğrenme sistemi hakkında ne kadar bilgisi olduğudur. Bu bilginin ($f(x)$ gibi sınıflandırma algoritmaları, x girdi vektörleri, hiperparametre vb.) ne kadar çok olduğu kabaca ikiye ayrılabilir: Beyaz kutu saldırıları yöntemlerinde saldırganın tamamen oluşturulmuş sistem hakkında bilgisi olduğu varsayılır. Öyle ki hiperparametrelerin, ağırlıkların, sonuç sınıf sayısı vb gibi öğrenme ve test sürecindeki her bilgiye erişimi vardır. Burada öğrenme verilerine ve kullanılan yöntemlere erişim her zaman sınıflandırıcının atlatılacağına veya saldırganın istediği sınıfa girdi verisini koyabileceği anlamına gelmez. Saldırganın bildiği sisteme dair çıkarımlarda bulunarak x' ile yanlış sınıflandırma yapması da kendisine saldırdığı sistem hakkında yeni bir bilgi kazandıracak, sonraki denemesinde sisteme daha ciddi zarar vermesi de olasıdır. Savunan taraf buna karşılık olarak en az bilgi sızdıracak şekilde makine öğrenme haricinde sistemlere başvurarak (veri şifreleme, verileri daha güvenli bir ortama alma vb.) ve sürekli denemelere izin vermeyip sınırlayan makine öğrenme modelleri

kullanacaktır.

Gerçek hayattaki beyaz kutu saldırılarına örnek olarak el yazısı tanıma işi için oluşturulan veri setleri ve buna uygun algoritmalar genellikle açık kaynaklı olup herkesin kullanımına sunulmuştur. Topluma açık bilgilerden oluşturulmuş bu sistemleri kullanan ve el yazısından çeşitli tanıma işlemleri ile iş yapan bir firma, beyaz kutu saldırılarına açık durumdadır (Vorobeychik ve Kantarcıoglu 2018).

Siyah kutu saldırı yöntemlerinde saldırganın algoritma ve veriler hakkında, hiperparametre ve ağ topolojileri gibi (yıldız, seri, paralel vb. bağlanmış sistemler) makine öğrenimini sağlayan iç dinamikler hakkında bilgisi bulunmaz veya çok azdır. Bu az bilginin içinde veri setleri daha oluşturulmadan önceki fiziksel ortam hakkında (kamera, ses sistemleri vb.) veya "örneklem alınan kitle" hakkında bilgi sahibi olmak, girdileri farklı olan bir makine öğrenme sürecine tabii aynı sistemin çıktıları hakkında bilgi sahibi olmak, farklı sistemlerin aynı veri setine nasıl tepkiler verdiğini gözlemlemek gibi doğrudan sisteme ilişkin olmayan, birçok dolaylı yollardan elde edilen bilgi parçacığı da bunların içinde sayılabilir (Vorobeychik ve Kantarcıoglu 2018). Beyaz kutularda olduğu gibi siyah kutu saldırılarında da başarı garantisi her durum ve/veya koşulda mümkün değildir; ancak beyaz kutulara göre başarı ihtimalleri daha düşüktür. Bu nedenle saldırgan bilmediği sisteme karşı daha özgürce ataklarını planlayabilmekte ve bu saldırılara karşı savunan taraf genellikle önlem almayacağı için bilinenlerden daha farklı biçimlerde saldırılar çeşitlenebilmektedir. Örneğin makine öğrenme sisteminin çalıştığı bir bilgisayarın RAM'inde yer alan bilgilerdeki değişimi elektronik olarak inceleyerek sistemde yanlış sınıflandırma yaptıracak veri elde etme işlemi ve bundan yararlanarak oluşturulan bir saldırı sisteme dolaylı yoldan yapılmış bir saldırı olup, siyah kutu ataklarına bir örnektir.

3.3 Belirlenen (Spesifik) Amaca Göre Ataklar: Hedefli veya Hedefsiz Saldırılar

Saldırganların makine öğrenme sistemlerine saldırıları için farklı nedenleri olabilir. x girdiler (veri-resim, ses vb.), x' bozulmaya uğratılmış girdiler, $f(x)$ sınıflandırma algoritması ve $f(x) = y$ makinenin doğru yaptığı bir sınıflandırma olsun. Eğer saldırgan belirlediği bir y''i, y'nin yerine geçirecek şekilde $f(x') = y'$ spesifik olarak

belirlemiş ise, bu bir hedefli saldırı olarak belirli bir y' yoksa bu hedefsiz saldırı olarak isimlendirilir. Hedeflerin sadece bir tek y olarak seçilmesi yerine, aynı anda birden çok hedef de seçilebilir. Hedefli saldırılarda saldırgan makine öğrenme algoritmalarına önceden spesifik olarak belirlediği bir amaçla ataklarını gerçekleştirir. Örneğin makine öğrenme ile sosyal medya üzerindeki topluma açık yazılardan insan dilini öğrenmeye çalışan bir yapay zeka uygulamasını düşünelim. Bu uygulamanın saldırganlar tarafından spamlanarak, belirli bir etnik grup veya din isminin küfür olarak öğrettirilmesi ve bunun sonucunda yapay zekanın çalışması sırasında belirli bir din ve/veya etnik grubu cümle içinde yazmasını sağlamak hedefli bir saldırıdır. Hedefsiz saldırılarda saldırgan öğrenme algoritmasının veya üstünde çalıştığı makinenin hatalarının daha fazla olmasını, böylece bütün sınıflandırmalara aynı anda etki ederek karışıklık çıkarmaya çalışır. Bu sayede hedefli saldırılarda olduğu gibi, tek bir çıktı üzerine yoğunlaşmak yerine bütün çıktıları etkileyerek doğru sınıflandırmaları yok eder (Vorobeychik ve Kantarcıoglu 2018). Örneğin makine öğrenme kullanan anti-virüs yazılımlarına, saldırganların yeni çıkartacakları virüsleri yanlış olarak sınıflandırtmak için kodunu değiştirerek kendi formatını sürekli değiştiren solucan örnek verilebilir. Bir süreliğine sıradan bir dosya gibi daha sonra anti-virüs firmasının bildiği başka bir virüs gibi davranarak esas virüsün gizlenmesi genel olarak hedefsiz bir saldırıdır.

3.4 Görsel Verilerle Çatışmalı Makine Öğrenmede Başarı Değerlendirmesi

Çatışmalı makine öğrenmede, görsel (resim vb.) kullanan bir algoritmaya yapılan saldırının ne denli başarılı olduğunu belirleyebilmek için analitik bir yöntem yoktur. Bu nedenle saldırının başarı değerlendirmesi çeşitli değerlendirme araçları kullanılarak bilgisayar ortamındaki denemelerle yapılır.

Görsel alanda çalışan çatışmalı makine öğrenmede çıktılar değerlendirilirken iki ana yol izlenir. İlki ortaya çıkan resmin insan için ne kadar kaliteli göründüğü dolayısıyla insan gözlemciyi kandırıp kandıramayacağı, ikincisi oluşturulmuş resmin makine öğrenme sistemine verildiğinde resmi ne kadar yanlış sınıfa atadığı yani makinenin kandırılıp kandırılamayacağıdır. Makineyi kandırma işi, insanı kandırmaya göre modellenmesi ve matematiğini oluşturması daha kolaydır. Çünkü insanın bir resimdeki hataları, netliği, ışık oranlarını algılaması ve değerlendirmesi şu andaki makineden çok üstündür ve bu

algılamanın matematiksel olarak formüllere dökülmesi de zordur. Bu nedenle yeni ortaya çıkan birçok çatışmalı örnek üreten algoritma olmasına karşın, birçoğunun yaptığı değişiklikler makineleri kandırmasına rağmen insan gözü ile algılanabilmektedir. Yazılan algoritmaların matematik olarak modellenemeyecek insan estetiği ve oran algısını da yenmesi beklenmektedir. Resimler; ya karşılıklı olarak ikili farklarına bakacak şekilde (full reference-referanslı) ya da tekil olarak her resme farklı değerler verecek şekilde (no reference-referanssız) iki farklı yöntemle kalitesi açısından değerlendirilebilir. Görsellerin insan için kalite değerlendirilmesi üzerinde çok uzun süredir çalışılan ancak günden güne iyileşen çözünürlük, aydınlanma ve renk gibi birçok görsel alandaki ilerlemelere rağmen yetersiz kalmaktadır (Vorobeychik ve Kantarcıoglu 2018). Bu değerlendirmeler görsel sanat eserlerinin orjinallerini taklit veya sahtelerinden ayırt etmede de kullanılabilmektedir.

Bazı popüler değerlendirme metotları basitten karmaşığa göre şu şekildedir:

1-PSNR (Peak Signal Noise Ratio-Tepe Sinyal Gürültü Oranı-Referanslı): İki resimdeki maksimum piksel değerleri temel alınarak, resmin kalitesinin en yüksek olduğu nokta ve çevresi hakkında bir ölçümdür. Başka resim unsurları (renk, renk yoğunluğu vb.) dikkate alınarak karmaşık değerlendirme metotları da üretilebilir (Fardo vd. 2016).

2-SSIM (Structural Similarity Index – Yapısal Benzerlik Dizini-Referanslı): İki resmin parlaklık ve renklilik ölçümlerinin ortalama ve standart sapma değerlerinin yer aldığı matematiksel ifadelere konulmasıyla elde edilir (Nilsson ve Akenin-Möller 2020).

PSNR ve SSIM metotları parlaklık ve renk değerlerinin ortalamalarına göre çalışırlar. Bu nedenle insan gözüne çarpan yüksek frekanslı piksel detaylarını ve tekil piksel bozulmalarını görmezden gelip kaçırma eğilimi içerisindedirler. Bu nedenle hatalı değerlendirmelere yol açabilirler.

3- FSIM: (Fourier Similarity Index – Fourier Benzerlik Dizini-Referanssız): Tek bir resmin tümünün renk ve tonlarıyla belirlenen biçimine ait fourier dönüşümleri alınarak renk ve tonlarının faz bileşenleri elde edilir. Bu bileşenlerin Hilbert dönüşümleri kullanılarak bulunan değerlerle yapılan bir ölçümdür. Bu ölçüm değeri kullanılarak

resim betimlenir (Zhang vd. 2011).

FSIM metodu, PSNR ve SSIM metotlarından farklı olarak görüntülerin frekans bileşenlerinin temel alarak ortaya bir tahlil koyduğu için tekil piksel bozulmalarını değerlendirmede daha hassastır.

4-HDRVDP: (High Dynamic Range Visible Difference Predictor – Yüksek Dinamik Aralık Görünen Farklılık Tahmin Edicisi-Referanssız): İnsan gözü ve beyninin görüntü algılamadaki iş gören kısımları matematiksel olarak modellenmeye çalışılır. Göz retinasından itibaren derlenen ışık verisi, önceden psikofizik olarak toplanmış verilerle beraber işlenerek bir ölçüm sistemi oluşturulur. Bu ölçüm sistemine bir resim girdi olarak verilerek sonucunda bir değer elde edilir. Bu değer resmin betimlenmesinde kullanılır (Rafal vd. 2011).

5- DeepSIM: (Deep Similarity Index – Derin Benzerlik Dizini-Referanslı): Önceki verilen SSIM metodunun geliştirilmişidir. Metod bir resmin parçalanarak, her bir parçasına SSIM metodunun uygulanması olarak kısaca tanımlanabilir. Resimler aşağıda bahsedilecek bir çeşit dönüşüm işlevli matrisler olan kernellerle parçalanarak CNN'ye, oradan her bir imaj SSIM ile değerlendirilerek bir öğrenme sürecine tabi tutulur. Böylece bir resmin birçok bileşeninden SSIM elde edilerek daha hassas bir ölçüm yapılmaya çalışılır (Czolbe vd. 2020).

6-PIPAL (Perceptual Image Processing Algorithms – Algısal Görüntü İşleme Algoritmaları-Referanslı): Çok yönlü ve çok katmanlı özel olarak makine öğrenme kullanılması için geliştirilmiş bir metottur. Yapılan iki resim arasındaki farkların öğrenildiği bir model olup, iki resim içindeki cisimleri (resmi oluşturan parçalar ve\veya bileşenleri) algılayıp, konumlarına ve çevresindeki renk değişimlerine bakılarak öğrenildiği işlemler sonucunda, iki resme değer vererek ikisi arasında hangisinin nitelikli olduğuna dair bir yargının ortaya konulduğu bir metottur (Gu vd. 2020).

Makine için değerlendirme kıstasları insan algısının kalite sınıflandırmasına ve değerlendirmesine göre gelişmemiş bir alandır. Makine öğrenme modeli sonucu yapılan sınıflandırmanın hangi olasılıkla doğru tahmin edildiği makine için bir ölçümdür.

Buradaki olasılık değerlendirmesi istatistikteki gözlemlerin yapıldığı yığın dağılımı hakkındaki varsayım(lar)a değil, sadece öğrenme sürecindeki yanılma ve doğru sınıflandırma sayıları kullanılarak ampirik tahmine dayalıdır. Örneğin, bir resmin içindeki cismin at olup olmadığının belirlenmesinde seçim yapan bir öğrenme algoritması bir test verisini (resmini) 0.90 doğru olma olasılığıyla at olarak tahmin ediyor olsun. Aynı koşullar altında at resmini (test verisini) bozarak bu olasılığı 0.60'a indiren çatışmalı bir makine öğrenme tekniği algoritmasına göre, bu olasılığı 0.30'a indiren bir başka çatışmalı makine öğrenme tekniği kullanan algoritma daha başarılıdır denir. Yani 0.90 olasılıkla doğru kararı veren makine öğrenme algoritması kararını değiştirmiştir. Dolayısıyla çatışmalı makine öğrenme kullanılarak oluşturulmuş bozulmuş bir resim, hedef algoritma veya sisteme verildiğinde ne kadar yanlış sınıflandırılıyorsa veya ne kadar diğer doğru sınıflandırmaları yanıltabiliyorsa o kadar başarılıdır. İki farklı çatışmalı makine öğrenme yöntemi arasındaki fark da bu sebeple aynı sisteme uygulanan, yapılan yanlış sınıflandırmaya karşı doğru sınıflandırma olasılıklarının hangisinin daha büyük olduğudur. Ancak ikisinin de insan estetik yargısına göre farklı kaliteleri olabilir. Yanılma yüzdelerinin artması estetik açıdan daha kaliteli (hemen anlaşılıp ayırt edilemeyen) bir resim ortaya çıkacağı anlamına gelmez. Bu çalışmadaki uygulamada yukarıdaki değerlendirme metotlarından PSNR kullanılacaktır. PSNR'nin işleyişi aşağıdaki gibi detaylandırılabilir:

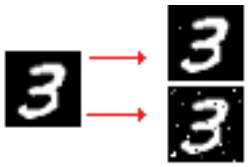
PSNR hesaplanırken, verilen bir resmin H x J çözünürlük değerlerini, R resmin maksimum alabileceği piksel değerini (örneğin siyah beyaz bir resmin [0-255] arasında gri tonu alabiliyorsa burada R = 255'tir.) $I(h,j)$ ilk resmin (h,j) 'inci piksel değerini ve $K(h,j)$ ikinci resmin hxj 'inci piksel değerini vermek üzere ve MSE (mean squared error - hata kareler toplamı) olmak üzere;

$$MSE = \frac{1}{HJ}\sum_{h=1}^{H}\sum_{j=1}^{J}(I(h,j) - K(h,j))^2$$

$$PSNR = 10\,log_{10}\left(\frac{R^2}{MSE}\right)$$

olarak hesaplanır. Alabilecek maksimum renk değerlerinin karesi ile piksel değerleri arasındaki farkların kareler toplamının logaritması alınarak elde edilir.(Fardo vd. 2016). Burada PSNR'nin yüksek olması birinci resmin kalitesinin yüksekliğini veren bir

ölçüm olarak değerlendirilir. Buna örnek olarak **şekil 3.1** verilebilir**.**



Şekil 3.1 28 x 28 pikselden oluşan 3 sayısı siyah-beyaz png resmi

(Sıkıştırmasız resim formatı) PSNR değerleri üstteki resim için 34.998 ile hesaplanırken diğer resim için 18.222 hesaplanmıştır. Üstteki resimde tek bir piksel değeri değiştirilmişken alttaki resimde birden çok piksel ile orjinal resim bozulmuştur, bu durum insan gözünün görsel algılamasıyla da fark edilebilir. Üstteki resmin PSNR değeri yüksek olduğundan orjinale yakınlığıdır ve dolayısıyla PSNR'ye göre görsel değerlendirme olarak ikinciye üstündür.

3.5 Saldırı Algoritmalarına Örnekler

Daha önce bilgi, zaman ve hedef unsurlarıyla sınıflandırılmış çatışmalı makine öğrenmesinin uygulanmasında izlenebilecek algoritmalardan söz edilecektir. Yukarıda verilen değerlendirme metotları algoritmalar içinde kullanılacaktır.

Bu algoritmalar pek çok şekilde oluşturulabilir. Bu konudaki algoritmalar verilecek olanlarla sınırlı değildir, çok sayıda algoritma bulunabilir. Bu nedenle çalışmaya konu olan uygulama çerçevesindeki algoritmalar yani, beyaz kutu, hedefsiz ve kaçınan saldırı algoritmalardan dört tanesi özetlenecektir. Aşağıda bu çalışmada yapılacak uygulamaya benzer özelliklere sahip aynı veri setinin kullanıldığı çalışmalarda yer almış basitten zor olana doğru sıralanmış seçilmiş 4 algoritma verilecektir. Aşağıdaki algoritmalardan ilk üç tanesi kaçınma-beyaz kutu-hedefli saldırılar için tasarlanmıştır. 4. saldırı ise kaçınma-beyaz kutu-hedefsiz saldırılar için kullanılmıştır.

1-FGSM (Fast Gradient Sign Metod-Hızlı Gradyan İşaret Metodu): Kayıp fonksiyonun gradyantı kullanılarak tek adımda orjinal görüntüye eklenen bozulmalar ile hesaplanır (Goodfellow vd. 2014). x orjinal görüntü, $f(x)$ sınıflandırma fonksiyonu, e kullanıcı tanımlı (hiperparametre) sabit bir pozitif reel sayı, ∇L , 2.6.3 alt başlığındaki gibi L kayıp fonksiyonunun nöronlarına göre gradyanı göstermek üzere x' şeklinde elde edilen bozulmuş görüntünün matematiksel ifadesi

$$x' = x + e \times sgn\big(\nabla L(f(x), x)\big)$$

biçimindedir.

2-IFGSM (Iterative Fast Gradient Sign Method - İteratif Hızlı Gradyan İşaret Metodu): İteratif FGSM olup tek seferde bir imaj elde etmek yerine her adımda bir önceki adımda bulunan yeni imaja belirli bir sınır içinde kırpma uygulanarak resimde küçültme uygulanır (Goodfellow vd. 2016). Bu sayede kayıp küçük hale getirilir. m adım sayısını gösteren doğal sayı, e kullanıcı tanımlı sabit bir pozitif reel sayı, kırpma ($clip$) işlem

$$x'_{m+1} := clip\{x'_m + e \times sgn(\nabla L(f(x_m), x_m))\}$$

şeklinde verilir.

3-JSMA(Jacobian Saliency Map Attack – Jakobiyen Belirgenlik Haritası Atağı): Görüntüdeki bütün piksellere değer atayarak aykırı görünen pikseller en yüksek, çevresi ile uyumlu olan piksellere en düşük tam sayı değerler atayark bir harita oluşturur. Bu haritanın belirlenen bir sınır değer üzerine çıkan piksellere göre saldırı modellenerek, bu aykırı piksellerle tüm etiketlerdeki doğruluk oranı düşürülmeye çalışılır (Papernot vd. 2016).

4- DeepFool: Hedefsiz bir saldırı olup bu saldırı, makinenin öğrendiği karar sınırlarına lineer olarak yaklaşımda bulunarak, bu sınırları geçecek en küçük değerleri bulmaya uğraşır. Genellikle uğraşılan problemler lineer olmadığından, bu değerler bulunduktan sonra orjinal imaja göre sınırlara en yakın olacak şekilde optimizasyon algoritmaları kullanılıp sonuç elde edilir (Moosavi vd. 2016).

4. ÇATIŞMALI MAKİNE ÖĞRENME UYGULAMASI

Çatışmalı makine öğrenme uygulaması olarak Carlini ve Wagner (2017) algoritması seçilmiş olup; bu algoritma zamanlaması bakımından kaçınma, atağın uygulanacağı ortama dair ön bilgi bakımından beyaz kutu ve spesifik hedefi bakımından hedefsiz saldırı türünde bir algoritmadır. Tezin bu bölümünde algoritmanın kullandığı veri seti, çalışma mantığı ve kullandığı matematiksel temeli açıklanacaktır. Uygulama olarak bu algoritmanın üzerinde değişiklikler yapılarak nasıl daha hızlı bir algoritma geliştirilebileceğine ilişkin teknik ve yöntemler 4.2.3'de açıklanarak verilecektir. Birinci bölümde kullanılan materyal olarak üzerinde çalışılan makinenin özellikleri ve uygulama dili belirtilmiştir. Sonraki bölümlerde veri setinin boyutları ve yapısı verilmiştir. Yöntem bölümünde ise orjinal saldırı algoritması verilmiş; daha sonra üzerinde yapılan değişikliklerin anlatımı yapılmıştır.

4.1 Uygulamada Kullanılan Araçlar ve Veri Seti

Bu bölümdeki kullanılan sistem Windows 7, işlemci çekirdeği i5 2. nesil 2.5 gigahertz ve RAM 4 gigabayt olup sabit disk 120 gigabayttır. Python dili versiyon 3.6.4 kullanılarak makine öğrenme geliştirmeleri için sıkça kullanılan ücretsiz açık kaynak Tensorflow ve Numpy yazılım kütüphaneleri, kaynak kodları içinde yer almaktadır.

Dijital görüntüler, elektronik bir ortam hafızasında sayısal olarak yer alır. Örneğin, bir kamerada hafıza kartında veya bir yazıcı ile taranıp bilgisayar sabit diski üzerinde tutulan bir görüntünün, kayıt formatlarının isimleri ve elektronik cihazdan gelen özellikleri (flash disk, SSD, Hard disk vb.) farklı olabilir ancak temelde tarama örüntü (bitmap) ile hafızaya işlenirler. Her bir resim burada iki boyutlu matris olarak ifade edilebilir. Matematiksel olarak ifade edecek olursak; X bu dijital resmin enini temsil eden pozitif tamsayı ve Y bu dijital resmin boyunu temsil eden pozitif tamsayıdır. Tarama örüntü resmin enini ve boyunun çarpımıyla ortaya çıkan birim alana düşen hücre sayısı, çözünürlük olarak adlandırılmaktadır. Bu $X \times Y$ boyutundaki matrisin her hücresi ise renk değerlerini verir ve bu elemanların her birine piksel denir. Bu şekilde dijital ekranı olan herhangi bir görüntüleme aygıtında veya bir kağıda basılarak tekrardan oluşturulabilirler. Bu X ve Y ne kadar büyükse, resmin hafızada kapladığı yer

o kadar büyük olacaktır. Bu sebeple kayıpsız ve kayıplı olmak üzere iki şekilde sıkıştırılıp kayıt aygıtında saklanırlar. Kayıpsız sıkıştırma olan formatlarda orjinal görüntü aynen korunur, bu sebeple genellikle hafızada görüntünün kopyasından daha çok yer kaplar. Kayıpsız görüntü sıkıştırma formatlarına günümüzde çok kullanılan örnek PNG verilebilir. Kayıplı sıkıştırmada ise görüntüden detaylar atılarak dosya boyutundan elde edilen kazanç vardır. Nitekim hafızada orjinal görüntüden daha az yer kaplarlar. Orjinal görüntü kayıplı her sıkıştırmaya maruz kaldıktan sonra tekrardan sıkıştırmadan çıkarıldığında bozulmaya uğrar. Kayıplı görüntü sıkıştırma formatlarına popüler örnek olarak JPEG verilebilir. Hafızada bu şekilde tutulan görüntülerin bir ekrana yansıtılması istenildiğinde ise aygıt yazılımı hafızadan görüntülerin alınması ve ekrana iletilmesi için yardımcı olur. Bunun için verinin hafızada nerede başlayıp nerede bittiği ilk önce işlenmekle beraber, yukarıda bahsedilen elektronik özelliklerine ve formatlarına uygun piksellerin ve çözünürlüklerin bir algoritmayla işlenmesi ile renk modelleri elde edilir. Bu renk modelleri genel olarak yaygın kullanılan piksel formatları arasında tek renk, siyah-beyaz, kırmızı-yeşil-mavi (RGB) sayılabilir. Burada da işlenen renklerin ne kadar kaliteli olacağı ile alakalı hafızaya kayıtlı bilgi ile renkler ölçeklenir ve bu konseptin ismi renk derinliğidir. Renk derinliği, n renk derinliğini ifade eden pozitif tamsayı 2^n bit ile ifade edilir. Günlük olarak kullanılan çoğu elektronik ekranda $n = 24$ bit renk derinliği vardır. Örneğin RGB için her bir renge 8 bit verilir. Böylece örneğin kırmızı rengin dijital temsili için $2^8 = 256$ tane kırmızı tonu elde edilir ve her bir kırmızı tonu 0-255 arasındaki bir tamsayı ile ifade edilmiş olur. Aynı şekilde diğer renklerin kombinasyonu ile ortaya her bir pixele eşlik eden 3×1'lik RGB'nin her bir renk bileşenini temsil eden bir vektör ortaya çıkar. Siyah-beyaz resim için ise kırmızı, yeşil ve mavi değerleri aynı tamsayı ile ifade edilip tek bir 0-255 tamsayı değerine sabitlenerek üç boyutlu vektör ile temsil yerine, tek bir tamsayıya indirilmiş haliyle elde edilebilir. Böylece aygıt yazılımı grafik işleme birimini çalıştırarak her bir pikseli tek tek ekrana işler. Ekranın ilk neresine ve hangi büyüklükte grafiğin işlenecek olduğu da yine aygıtın işletim sisteminin kararına bağlıdır.

Bu tez çalışmasında kullanılacak MNIST (Modified National Institute of Standards and Technology Database) el yazısı ile yazılmış rakamlardan oluşturulmuş büyük bir veritabanıdır. Günümüzde bu veritabanı sıkça makine öğrenme alanında çalışan algoritmalar ve uygulamaların kabiliyetlerinin ve doğruluklarının test edildiği popüler

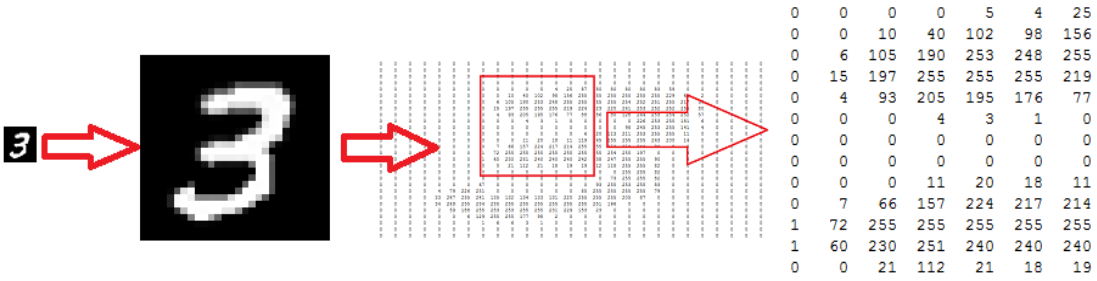
bir veri setidir (Carlini ve Wagner 2017). Bu çalışmada da bu veri seti çatışmalı (adversarial) makine öğrenme için kullanılacaktır.

MNIST verileri ilk oluşturulurken Amerika Birleşik Devletleri'nde NIST ismi ile Amerikan Nüfus Sayım Bürosu çalışanları ve Amerikan lise öğrencilerinin el yazması rakamları ile oluşturulup çalışanların el yazısı örnekleri SD3, öğrencilerin el yazısı örnekleri SD1 olarak isimlendirilmiştir. NIST veri setinin özellikleri şunlardır: SD1 58527 el yazısıyla yazılmış, 500 farklı öğrenciden alınmış sıfır ve dokuz arasındaki rakamların 12 kere tekrarla yazılmasıyla oluşturulmuş bir veri kümesidir, ancak eksikler vardır. SD3 60000 tane sıfır ve dokuz arası rakamların el yazısıyla 24 kere tekrarıyla 250 çalışandan alınmıştır. Öğrenciler ve çalışanların el yazısı arasındaki farklar gözle görülebilir düzeydedir. Bu nedenle bu el yazısı örnekleri için bu iki grubun oluşturduğu veri setlerinin karıştırılması gerekmiştir. Nitekim SD1 veri setinin eksiklerinin giderilmesi gerekmiştir. Böylece bu 500 öğrenci içerisinden 250 öğrenciye tekrar gidilerek bu kez 250 öğrenciden sıfır ve dokuz arasındaki rakamların el yazısıyla 24 kere tekrarı ile 30000 veri eksiksiz olarak toplanmıştır. MNIST veri seti 250 öğrenci ve 250 nüfus bürosu çalışanının verileriyle oluşturulmuştur. MNIST veri setinden 30000 SD3 örneği ve 30000 SD1 örneği karıştırılmıştır. 60000 örnekten oluşan bu veri makine öğrenme çeşitli amaçlarla uygulamaları için eğitim seti olarak kullanılmaktadır. MNIST veri setine 10000 verilik bir test seti de eklenmiştir. Bu amaçla 5000 SD3 ve 5000 SD1 de toplanıp karıştırılmış ve 10000 test seti olarak geliştiricilere sunulmuştur. Hem test hem de eğitim seti için tam 250 öğrenci ve 250 devlet çalışanının el yazısı karışmıştır. MNIST veri setinin dijital özellikleri ise şunlardır: 28×28 pikselden oluşan siyah-beyaz dijital görüntülerdir. El yazıları 28×28'lik alana sığdırılması için ortalanmıştır (bounding box normalization). El yazısı ne kadar büyük veya küçük olsun 28×28'lik bir matriste sığacak şekilde yer alır. Dolayısıyla anlamlandırılamayacak şekilde rakamların kırpılmış olması söz konusu değildir. Her bir görüntüde tek bir rakam bulunmaktadır. Orjinal dosyalar CSV (Comma seperated values) formatıyla tutuluyor olup, internet üzerinden paylaşım yapıldığı zaman dosya bozuntuya uğramaması istenmektedir.

MNIST veri seti ile uğraşan çeşitli makine öğrenme kullanan bazı popüler algoritmalarının performansları verilecektir. Öklidyen uzaklık kullanan Yakın k-komşuluklar metodu %5 test hata oranı, 2 katmanlı yapay sinir ağı, 300 gizli

birimli %4.7 test hata oranı, konvolüsyonel 16x16 örnekleme ile %1.7 test hata oranı elde edilmiştir (Goodfellow vd. 2014).

Bu tez çalışmasında MNIST veri setine dayanarak sonuçlar ve sınıflandırmaların nasıl saptıralacağı gösterilecektir. Bunun için resimdeki detaylar yok edilerek, sınıflandırma sonucu oluşan kayıpların tekrardan ihmal edilerek detay seviyesinde sağlanan öğrenmenin yok sayılması amaçlanacaktır. Burada hem görsel açıdan tatmin edici sonuçlar verilmesi, yani resmin insan algısında net bir karşılığı bulunması hem de makinenin kandırılması yani verilen resmi insan algısındaki halindekinden farklı olarak sınıflandırması beklenmektedir. Dolayısıyla hem el yazısını gören insan gözü hem de sınıflandırmayı yapan yapay sinir ağı birlikte kandırmak nihai amaç olarak belirlenmiştir. Buna uygun bir örnek **Şekil 4.1** verilebilir**.**



Şekil 4.1 3 sayısının bilgisayar hafızasındaki görüntüsü

(sıkıştırmasız resim formatı) dijital olarak bilgisayar ortamına aktarılışı. Resim matris olarak ifade edildiğinde siyah değerler 0'ı, grinin tonları 1-255 arasında değerler almaktadır. Üçüncü görünüm bilgisayar hafızasında bir dizi olarak yer alır.

Konvolüsyonel yapay sinir ağları görüntü işleme ve tanımada kullanılan derin öğrenme (kısaca birden fazla öğrenme yönteminin aynı öğrenme problemi içinde kullanılması anlaşılacaktır) yöntemlerinden bir tanesi olup, temelini matematikteki konvolüsyon işleminden alır. Konvolüsyon görüntü işlemede resmi değiştirmek için kullanılan önemli araçlardan biridir. Bir f fonksiyonunun bir başka g fonksiyonu tarafından dönüştürülmesi ile elde edilen yeni fonksiyon bu iki fonksiyonun konvolasyonu olarak adlandırılır. x bir girdi resmi ve y ise resmi manipüle etmek için x'e uygulanacak matris olmak üzere x ve y'nin her bir matris elemanının nokta çarpımlarıyla elde edilen konvolüsyon işlemi

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$

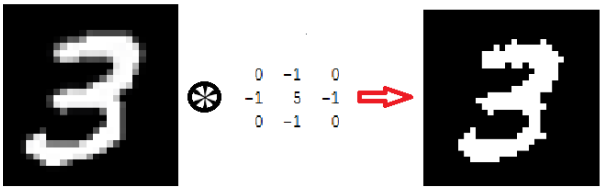
formülü ile verilir . Burada τ bağımsız değişken, t sabit olmak üzere f ve g birer fonksiyon olmak üzere sonuç bu iki fonksiyonun konvolüsyon integrali olarak adlandırılır. Bu integralin toplam formülü ile yazımı, ayrık konvolüsyon

$$(f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau)$$

formülü ile verilir. İşleme alınacak fonksiyonlar iki veya daha fazla boyutlu olduğunda da konvolüsyon formülü diğer boyutlar formüle konularak çözülebilir. Örneğin bu çalışma için ve genel olarak dijital görüntülerde kullanılan iki boyutlu konvolüsyonlar

$$(f * g)(t,j) = \sum_{k=-\infty}^{\infty} \sum_{\tau=-\infty}^{\infty} f(\tau,k)g(t-\tau,j-k)$$

gösterilebilir ve yapılan iş iki boyutlu ayrık konvolüsyon işlemi olarak adlandırılır. Konvolüsyon işleminin görüntü işlemedeki uygulanışı şu şekildedir: konvolüsyon resmin maniplasyonu için kullanılan resimde bulanıklaştırma, keskinleştirme, görüntü içindeki kenarları ortaya çıkarma (**Şekil 4.1**), resim kanvasının (çerçevesinin) şeklini değiştirme gibi resimle alakalı birçok değişiklik yapılabilmesini sağlayan matematiksel işlemdir. f fonksiyonu çözünürlüğü $h \times j$ olan iki boyutlu bir resmi ifade ediyor olsun. f ile konvolüsyon işlemine girecek olan g fonksiyonuna görüntü işlemedeki özel ismi olan kernel denir. Kernel g x $\times$ x çözünürlüğe sahip olsun. x dönüşümü yapılacak olan ilk resmin boyutlarını geçemez yani $0 \leq x \leq h$ ve $0 \leq x \leq j$ birlikte sağlanmış olmalıdır. g'nin içeriğine göre resme uygulanan kerneller konvolüsyon sonucu çeşitli şekiller ortaya çıkaracaktır. Seçilen kernele göre ve taramanın kaç adımda bir resim üzerinde ne kadar atlatılacağına göre (kaydırma-tarama aralığı) ortaya çıkan sonuç genellikle orjinal resmin boyutundan küçük olur. Bu sebeple dolgu (padding) denilen orjinal resmin çevresine sıfırlardan oluşan çerçeve eklenip $h \times j$'lik çözünürlük $(h+p) \times (j+p)$ şeklindeki çözünürlüklü, p padding boyutu olmak üzere daha büyük çözünürlüklü çerçeveli bir resme dönüştürülür. Resmi bulanık hale getirmek, resimdeki kenarları bulmak, renkleri patlatmak, dikdörtgen bir resmi yamultarak paralelkenar hale getirmek gibi birçok farklı matris seçimi ile resmi çok farklı manipülasyon tekniği uygulanabilir **(Şekil 4.2)**. Bu konvolüsyon tekniklerinin tümü CNN (Konvolüsyonel Yapay Sinir Ağları) derin öğrenmesinin temelini oluşturmaktadır.



Şekil 4.2 28 x 28 pikselden oluşan 3 sayısı siyah-beyaz png resminin konvolüsyon ile keskinleştirilmesi

Soldaki resim [0 -1 0 ; -1 5 -1 ; 0 -1 0] kerneli ile konvolüsyon işlemi sonucu soldaki 3 yazısı elde edilmiştir.

4.2 Yöntem

Carlini ve Wagner (2017) algoritması geliştirilirken ilk olarak resme eklenecek küçük bir değişim ile resmin ait olduğu sınıfı değiştirmeyi göz önüne alınmıştır. Daha sonra var olan optimizasyon yöntemleriyle bu küçük değişimin yeri ve miktarı bulunmaya uğraşılmıştır. Değişimin yeri ve miktarının belirlenmesi için birçok olası denklem ve deneme yanılma yolu ile elde edilebilecek çok fazla sayıda resim oluşturulabilir. İşlem sonucunda elde edilecek resimleri sınırlamak, en iyi sonuçları elde etmek ve sonuçlar arasında iyi seçim yapmak için Carlini ve Wagner (2017)'de çeşitli yaklaşımlar izlenmiştir. Aşağıdaki bölümlerde yukarıda değinilen küçük değişim miktarının resim üzerinde nereye konulacağının belirlenmesi için temel uzaklık ölçütleri kavramı verilmiştir. Sonraki bölümde orjinal saldırı algoritmasında kullanılan optimizasyondaki optimal çözüm kriterleri ve objektif fonksiyonu seçimlerinin nasıl olduğu açıklanmıştır. Son bölümde ise orjinal yöntemdeki optimal çözüm kriterlerine getirelen yorum ve yöntem gösterilmiştir.

4.2.1 Uzaklık ölçütleri

Carlini ve Wagner (2017) uzaklık ölçütlerini tanımlamak için normları kullanmış olup, L_p uzaklığı $||\ x - \ x'||_p$ ile gösterilir. Burada x orjinal, x' değiştirilmiş görüntüyü ifade eder. Norm ise: $||V|| = \sum(\ |vi|p) - p$ verilir. Bu durumda L_0 normu mutlak değer olur ve $|x - x'|$ ile bir resimdeki değişen piksellerin sayısını ölçer. L_2 uzaklığı ise $||\mathrm{x} - \mathrm{x}'||^{\ 2}$ ile kareler toplamının karekökünü bulur ve bu da görüntü işlemede piksel değişiminin

uzaklığını bulur. L_∞ uzaklığı görüntüde oluşmuş herhangi bir değişikliğin maksimum olanını bulur. Değiştirilecek piksel değerleri ve uzaklıklar matematiksel olarak ölçülebilir durumda olmasına rağmen, çatışmalı makine öğrenmede verilen bir yönteme karşılık hangi veri setinde hangi sonucu vereceğini kestirmek mümkün değildir. Ancak gerçek hayatta yapılan deneyler ile algoritmalar birbirinden ayrılabilir. Uzaklık ölçütleri değiştirilerek var olan algoritmaların çeşitli varyasyonları elde edilip, farklı optimizasyon problemleri oluşturulabilmektedir. Örneğin el yazısı için değişen piksel sayısı daha önemli bir uzaklık bileşeni olarak atak sonuçlandırmada iyi sonuç verebilecekken, yüz tanıma sisteminde diğer bir metrik daha anlamlı olabilmektedir (Vorobeychik ve Kantarcıoglu 2018).

4.2.2 Orjinal atak algoritması

Carlini ve Wagner (2017) orjinal atak algoritmasının nasıl modellenip işletildiği aşağıda özetlenecektir. $x \in R^m$, $m = o \times p$'lik çözünürlük değerli ve t olarak sınıflandırılmış orjinal resim olmak olsun. x'i d kadar bir miktar bozarak t' yanlış sınıfına atanmasını sağlayacak bir saldırının yapılması ele alınacaktır. Ataktaki amaç bozulmuş resim x' 'in yine $x' \in R^m$ ve bu kez ait olduğu sınıfın t' olmasıdır. Verinin bozulması işleminin $D: R^m \rightarrow R^m$ fonksiyonu aracılığı ile gerçekleştirildiğini varsayalım. Buna göre t sınıfına ait olan x 'in D fonksiyonu aracılığıyla $x' = D(x)$ olarak bozulup yanlış olarak t' 'ne sınıflandırılması işlemi çatışmalı atak olarak adlandırılır. Eğer söz konusu D fonksiyonu orjinal x girdisine bozulmayı doğrusal ve toplamsal olarak

$$x' = D(x)$$
$$= x + d$$

biçiminde sağlıyorsa bu atak toplamsal çatışmalı saldırı olarak adlandırılır.

x girdileri için, k tane sınıfın var olduğu durumda, $k-1$ tane $g_j(.)$ sınıflandırma (diskriminant) fonksiyonu yazılabilir. Sınıflandırma fonksiyonları üzerinde genelde bir kısıtlama olmamakla birlikte doğrusal sınıflandırma fonksiyonları ile çalışmanın bir takım kolaylıkları ve üstünlükleri var olduğunu hatırlatalım. x girdisi için $g_j(\mathrm{x})$ diskriminant değeri, diğer sınıflandırma fonksiyonlarının değerlerinden daha büyükse resim j sınıfına atanır. Bu durum aşağıdaki gibi ifade edilebilir:

$$g_j(\mathrm{x}) \geq g_l(\mathrm{x}) \ , \qquad \mathrm{j} \neq \mathrm{l}$$

Söz konusu sınıflandırma işlemi aşağıdaki yol da izlenebilir:

$$g_j(x) \geq max_{\ \mathrm{j}\neq\mathrm{l}}\{g_l(\mathrm{x})\} \Leftrightarrow max_{\ \mathrm{j}\neq\mathrm{l}}\{g_l(\mathrm{x})\} - g_j(\mathrm{x}) \leq 0$$

Bu gerektirme x'i x' olarak bozup yapılabilecek potansiyel saldırı koşullarını belirlemek için kullanılabilir. d bozulmasının en küçük olması ya da x''ın orjinal görüntüye çok benzer olması bu koşullardan biridir. O halde resmin fark edilmeyecek kadar bozulması ve istenilen t' sınıfına atanması optimizasyon problemi olarak ortaya konabilir. Bu optimizasyonda amaç yapılan saldırının başarılı olma olasılığının yüksek olmasıdır. Böyle bir problem bozulmanın fark edilmeyecek boyutta olması ve yanlış sınıflandırma olasılığının yüksek olması olarak da bir olasılık problemi biçiminde ortaya konulabilirdi (Yuan vd. 2019).

Eğer bozulma miktarının en düşük olarak tutulması isteniyorsa, saldırı minimum norm saldırısı olarak adlandır ve aşağıdaki

$$minimize\ ||x' - x||_p$$
$$sk:\ max_{\ \mathrm{j}\neq\mathrm{l}}\{g_l(\mathrm{x})\} - g_j(\mathrm{x}) \leq 0$$

optimizasyon problemi çözülür. $x' \notin R^m$ söz konusu olduğunda yani böyle uygun bir bozulma sağlanamayacaksa bu durumda belirli koşullar altında bozulma derecesinin sınırları zorlanmak maksimum izin verilebilir saldırı olarak adlandırılıp η sabit olmak üzere

$$minimize\ max_{\ \mathrm{j}\neq\mathrm{l}}\{g_l(\mathrm{x}')\} - g_j(\mathrm{x}')$$
$$sk: ||x' - x||_p \leq \eta$$

olacak şekilde çözüme ulaşılır. Fakat bu kullanılarak amaç fonksiyonu minimum yapılamayabilir. Bu nedenle üçüncü bir alternatif regüralizasyona dayanan kısıtlamasız bir optimizasyondur. Bu Lagrange çarpanı kullanılarak formüle edilen bir optimizasyondur. Bu optimizasyon problemi ile bulunan x' regüralizasyon temelli saldırı olarak adlandırılır. Optimizasyon problemi λ Lagrange çarpanını göstermek üzere;

$$minimize_{x'}\ ||x' - x||_p + \lambda(max_{\ \mathrm{j}\neq\mathrm{l}}\{g_l(\mathrm{x}') - g_j(\mathrm{x}')\})$$

olarak ifade edilir (Vorobeychik ve Kantarcıoglu 2018).

Carlini ve Wagner (2017)'da regülerizasyon temelli bir optimizasyon tercih edilerek x'

saldırı resminin bulunması amaçlanır. Yukarıdaki optimizasyon ifadesinde $||x' - x||_p$ yerine daha genel bir ifade olarak bozulma miktarı orjinal metin için şu optimizasyonla hesaplanır. Yukarıdaki x' için çözüm uzayı yine x'in bulunduğu çözüm kümesi olacaktır. $x \epsilon [0,1]^n$ olduğundan, $x + d \epsilon [0,1]^n$ da bir resmi bulunduracak şekilde sınırlandırılır. Optimizasyon problemi

$$minimize_d \; ||d||_p + cf(x + d)$$

olarak ifade edilir. Dikkat edilirse $f(x + d)$, $max_{\; j \neq l} \{g_l(x') - g_j(x')\}$ yerine kullanılmıştır.

Uzaklık metriği $D: R^m \rightarrow R^m$ $||.||_p$ l_p normu olarak alınmıştır. t, x'in gerçek sınıfı olmak üzere t' saldırganın, x'i t'den koparark yeniden atamak istediği yeni sınıf ya da sınıflardır. Eğer t' belirli ise hedefli saldırı, değilse hedefsiz saldırı söz konusudur. Carlini ve Wagner (2017) optimizasyon problemindeki sınıflandırıcı fonksiyonlarını lineer sınıflandırma fonksiyonları dışında da seçmektedir. Sınıflandırıcı fonksiyon lineer olmadığında da söz konusu optimizasyon probleminin çözümü zahmetli ve zaman alıcıdır.

f için birden çok fonksiyon önerilmiştir, kötü amaçlı saldırılar oluşturulurken en iyi sonucu veren (sınıflandırma algoritmasının verdiği cevapların güvenilirliğini en çok aşağı indiren veya birisinin resmin kötü amaçlı biri tarafından oluşturulup oluşturulmadığını ayırt edememesi bakımından üstün) bu fonksiyonlar önerilmiştir. Carlini ve Wagner (2017) tarafından önerilen bazı f sınıflandırma fonksiyonları

$$f_1(x') = -loss_{F,t}(x') + 1$$
$$f_2(x') = \max(max_{i \neq t}(F(x')_i) - F(x')_t, 0)$$
$$f_3(x') = \max(0.5 - F(x')_t, 0)$$
$$f_4(x') = softplus(max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)$$

biçimindedir. Burada $x' = x + d$ olup orjinal resim x'e eklenen ufak değişim d'yi; $softplus(x) = log(1 + e^x)$, $-loss_{F,t}(x')$ çapraz entropi kaybını, $F(x')_t$ t'inci sınıfa ait olarak atanan sınıflandırılmayı verir. $F(x')_s$ şeklindeki s sınıfına ait atanan sınıflandırılmayı verir. $F(x')$'ler son katman olup; $F(x') = softmax(Z(x'))$ ile ifade edilir. t sınıfına ait olarak atanan $Z(x')_t$ en son katmandan bir önceki durumunu vermekteyken, $Z(x')_i$ i sınıfınaa ait olarak atanan en son katmandan bir önceki logitleri

göstermektedir.

c burada sabit olup genellikle en küçük olacak şekilde alınır. c'yi seçmek için ikili arama algoritması kullanılmıştır. c en küçük 0.01 alınıp optimal sonuç elde edinceye kadar 10 ile çarpılır, minimumlar karşılaştırılır. Elde edilecek modifikasyonların matematiksel olarak tutarlı (Örneğin resim siyah beyaz ise renk değeri 255'in üstüne çıkarsa bu hem resim için hem de sınıflandırma için bir anlam ifade etmez.) olması için değiştirilecek piksel değerleri sınırlandırılmıştır. Her bir değiştirilecek x_i için $i = 1..n$ sınır $0 < x_i + d_i < 1$ sınırı uygulanmıştır. f fonksiyonu seçimi serbest olup en etkili sonuçları veren seçimler K kararlılık, Z aktivasyon fonksiyonu (RELU,sigmoid vb.) olmak üzere buna Carlini ve Wagner (2017) temel olarak

$$f_4(x') = softplus(max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)$$
$$d = 0.5(tanh(w_i) + 1) - x_i$$

p uzaklığı L_2 almıştır. c ise ikili arama algoritması ile c = 0.001'den başlanarak c = 1000'e kadar sabit tutulmuştur. c her seferinde 10 ile çarpılarak $min\ ||d||_p + c\ f(x + d)$ en küçük yapan d değeri bulunmaya çalışılır. $||\mathrm{d}||_p$ normunun çözümü için 2 tane yöntem daha önerilmiş olup, bunlar normun ∞, 0 ve 2 olarak alınmasıyla d'nin hesaplanması sonucu ortaya çıkar. Ancak $p = \infty$ alınması durumunda d uzaklığı için türevi alınabilir bir denklem ortaya çıkmaz, sürekliliğin olmadığı durumlar vardır. Öyle ki $max(d)$ ile gradyan azalış uygulanmaya çalışılsa dahi en büyük iki değer arasında sıkışacaktır. $p = 0$ alınması durumunda yeniden süreklilik sorunu ortaya çıkar ve türev alınmaması sonucu gradyan azalış çalışmayacaktır. İki yöntemde de türevi alınabilen $p = 2$ normu çağrılmaktadır. Birinci durumda yani $p = \infty$'da iteratif olarak $p = 2$ durumunun algoritması çağrılır ve bu duruma göre amaç fonksiyonu ve sınır çözülür. Belirlenen bir sabit değere göre tüm pikseller için $p = \infty$ tekrardan geri çağrılarak önceki çözümün bulunanları yerine konularak bir sonraki adıma geçilir. Aynı şekilde $p = 0$'da da iteratif olarak $p = 2$ ile aynı işlemler yapılır.

4.2.3 Önerilen saldırı algoritması

$c\ f(x+d)$ terimi ile birçok f fonksiyonu tanımı yapılabilir. Resmin yanlış sınıflandırıldığını kontrol eden bu terime Carlini ve Wagner (2017) orjinal atak algoritmasında önerilen fonksiyonların varyasyonlarını verdirecek şekilde yeni algoritmalar oluşturulabilir. Bu optimizasyon problemini çözecek algoritmalardan biri oluşturulacak kötü niyetli resmin renk değerlerinin yarıya düştüğü yerlerin hesaplanmasıdır. Öyle ki; MNIST veri seti ve el yazısı örnekleri grinin tonları şeklinde kodlandığından ve bu çatışmalı makine öğrenme sürecini etkileyen en önemli unsur, optimizasyon sonucu bulunacak kötü niyetli resmin tonlarının siyah ve beyaza ne kadar yakın olduğudur. Bilgisayarın dizi olarak gördüğü resim için farkların kayıp fonksiyonlarında d ile hesaplanması için kayıp fonksiyonlarından biri aşağıdaki hale çevrilebilir:

$$x + d = 0.5\{\tan(w) + 1\}$$

$$w = \ tanh^{-1}(2(x+d) - 1)$$

Maclaurin açılımından $Maclaurin(a) = tanh^{-1}(a) = a + \frac{1}{3}a^3 + \frac{1}{5}a^5 \ldots$ şeklinde yazılabildiğinden $2(x+d) - 1$ ifadesi a olarak yazılırsa w değeri:

$$w = \ tanh^{-1}(2(x+d) - 1) = tanh^{-1}(a)$$

$$= (2x + 2d) - 1 + \frac{1}{3}(2x + 2d - 1)^3 + \frac{1}{5}(2x + 2d - 1)^5 \ldots$$

$d \in [0,1]$ olduğuna göre ve piksel değerlerinin yarısına göre hesaplanmak istendiğinde $(x+d)$ içeren yüksek terimli ifadelerin çok küçük olacağı öngörülmüştür. Bu durumda sınırlı sayıda Maclaurin açılımı yapılarak birçok farklı optimizasyon denklemi oluşturulabilir. Buna bir örnek

$$min\ ||a||_2 \ + \ c\ f(a)$$

$$a = \ (x+d) - 1 + \frac{1}{3}(x + d - 1)^3$$

$$sk\text{:}\ f_5(x') = softplus(max_{i \neq t}(Z(x')_i) - \ Z(x')_t) - \log(2)$$

olarak verilebilir. PSNR ölçütü ise daha kaliteli resimlerin seçimi ve optimizasyon algoritmasındaki üstel terimlerin nerede sonlanacağının bulunmasını sağlar. Yeni çatışmalı makine öğrenme ile elde edilmiş örnekler arasında eğer PSNR oransal olarak belirli bir ihtiyatın üzerinde ise bir sonraki aşamada Maclaurin açılımı hesaplanırken

açılımın daha uzun hali kullanılacaktır. Burada PSNR hedef fonksiyonu için güncelleme yaptırmakta kullanılmaktadır öyle ki toplam kayıplar ile oluşturulan saldırgan resim arasındaki farkın oranı ikili arama sonuçları Maclaurin denklemindeki terimlerin ne kadar fazla olacağını belirler. Böylece τ adım sayısı olmak üzere

$$min\ ||a||_2\ +\ c\ \ f(A)$$

$$\mathrm{A} := Maclaurin(a)$$

$$a := \{a :|\ \frac{PSNR(x + d_\tau)}{PSNR(x + d_{\tau+1})}\ > l\ \}$$

$$sk\colon f_5(x') = softplus(max_{i \neq t}(F(x')_i) -\ F(x')_t) -\ \log(2)$$

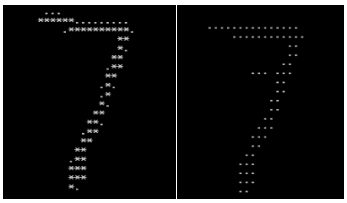
biçimindedir. Bütün bu yöntemlerin kullanılmasının sebebi $tanh$ ile yapılan işlemlerin ve türevlerin lineer olarak yazılan üslü ifadelere nazaran bilgisayar ve işlemci birimlerinde daha zor hesaplanması ve sayısal olarak yuvarlama hatalarına üslü sayılardan daha çok maruz kalmasıdır. Öyle ki PSNR sonucuna göre daha düşük sayıda sabit değerler kullanılarak daha hızlı çalışan aşağıdaki algoritma önerilmiştir **(Algoritma 3.1)**.

Algoritma 3.1 OnerilenCarliniWagner(Veriler, Girdiler)

1: hiperparametreler ← Girdiler
2: for all hiperparametreler **do**
3: loss ← []
4: Maclaurin_tanh ← {(x_n,y_n) ∈ Veriler: n}
5: **if** 0 >= loss – l2_dist **then**
6: gradyan = encok+gradyan+ loss – l2_dist /2
7: ikili_arama ← MaclurinDenklemi (gradyan)
8: **else**
9: gradyan = encok+gradyan+ loss – l2_dist *2
10: ikili_arama ← MaclurinDenklemi (gradyan)
11: **endif**
12: PSNR ← ikili_arama
13: **for in** PSNR mod K = k − 1
14: loss ← yeni_MaclurinDenklemi
15: adv_resim ← eniyiskor(loss)
16: **endfor**
17: endfor

4.2.3.1 Uygulama ile elde edilen bulgular

Bu uygulamada orjinal atak algoritmasına göre daha hızlı çalışan ancak sınıflama yeteneği bakımından daha başarısız bir algoritma üretilmiştir. Buna göre üzerinde çalışılan ortam; MNIST veri seti ile çalışan, öğrenme oranı 0.1, grup (batch) boyutu 128 olan 3x3 kernelli 32 tane nörondan oluşan ilk katmanlı, 3x3 kernelli 64 tane nörondan oluşan $RELU$ aktivasyonlu konvolüsyonel yapay sinir ağı ve çıktı olarak bu MNIST el yazısı verilerinin 50 döngüde 10 tane rakam ile çıktı oluşturduğu $softmax$ ile tam bağlı son katmandan oluşmaktadır. Çatışmalı makine öğrenme hiperparametreleri orjinal algoritma aynı bilgisayar ortamında %99.5 doğruluk oranı ile bu veri setinde 26 dakika 34 saniyede sonuç elde etmektedir. Önerilen yöntemde ise bu oran %87.9 doğruluk oranı ile 19 dakika 47 saniyede sonuç elde etmektedir (**Şekil 4.3).** Dolayısıyla bu koşullar altında yeni algoritma %25.53 daha hızlı çalışmakta ancak tahminlerde %11.6 doğruluk oranında düşüş yaşamaktadır. Buna ek olarak diğer doğru sınıflandırmalar için de orjinal duruma göre ortalama %6.039 bir artış sağlanmaktadır **(Çizelge 4.1).** Doğru olmayan sınıflandırmalar için makinenin %6.039 fark ile doğru tahmin ettiği sınıfı değiştirebileceği bir açık yakalanmaktadır. Bundan sonraki çalışmalar için birçok saldırı şekli önerilebilir ve distilasyon kullanılarak algoritmaların çeşitli fonksiyonlar ile daha iyi sonuçlar alınabilen algoritmalar yapılabilir.



Şekil 4.3 Orjinal sınıflandırma algoritmasına oranla "7" el yazısının yeni algoritmayla görsel olarak dönüşümü

(Soldaki resim orjinal görüntü, sağdaki resim değiştirilmiş görüntüdür)

Çizelge 4.1 Algoritmaların doğruluk oranları ve çıktı üretim sürelerinin karşılaştırılması

	Doğruluk Oranı	Çıktı Üretme Süresi (saniye)
Carlini & Wagner Algoritması	**% 99.5**	**1594**
Önerilen Yeni Algoritma	**% 87.9**	**1187**

5. TARTIŞMA VE SONUÇ

Szegedy vd. (2013)'nin saptamasına göre makine öğrenme sezgilere aykırı, beklenmeyen ve yorumlanamayan öğrenmelere de açıktır. Bu anlamda makine öğrenmenin güvenliği konusuna ilişkin önemli ve detaylı bilgi için Barreno vd. (2006)'nin çalışmasına bakılabilir. Makine öğrenme sistemlerine yapılabilecek saldırıların bu sistemler üzerine ileride günümüzdekinden daha ciddi zararlar verebileceği öngörülebilir. İstenmeyen veri sızıntıları (çalıntıları) bu tür zararların oluşmasına katkıda bulunması oldukça sık karşılaşılan bir durumdur. Özellikle veri zehirleme yoluyla yapılan saldırılar ve bunlara karşı savunma konuları uzun yıllardır üzerinde çalışılan bir konu olmasına karşın verinin spesifik yerlerinin zehirlenmesi yeni bir alandır.

Yapmış olduğum bu çalışma sırasında edinilen bilgilere dayanarak bu konularda birkaç saptama yapmak yerinde olabilir. İleride geliştirilecek makine öğrenme algoritmaları için bu yeni sızıntı ve zehirlemelerin önlenmesi amacına yönelik olarak savunan tarafın makine öğrenme performansından ve isabetliliğinden vazgeçebileceği öngörülebilir. Atağı önlemeye daha fazla zaman ayrılması zaman bakımından performansta azalma anlamına gelecektir. Atağın spesifik yerlere yönelmesinden dolayı burada savunma yapan taraf için bu spesifik yerlerin öğrenilmesindeki genelleme yapabilme kabiliyeti düşürülmelidir ki öğrenme sırasında buralar görmezden gelinebilsin. Bunun sonucunda spesifik yerlerin savunulması daha kolay hale gelirken isabetlilik de azalacaktır. Tezin uygulama kısmında verilen karar zamanına göre kısıtların (zehirleme-kaçınma) verinin ön işleme tabi tutularak bilgisayarın işlem görebileceği hale getirilmesi aşamasında da saldırılar çeşitlenecektir. Olası bir diğer gelişme de makine öğrenmeye bilgi sağlayan kamera, mikrofon, ısı sensörü vb. gibi araçların makine öğrenmesi için ön işlemlerden önce zehirlenmek üzere hedef haline gelmesidir.

Makine öğrenmenin güvenliği ve sağlamlığı (robustness), saldırı tespiti, önlem alma konuları çatışmalı öğrenmede öne çıkan konulardır ve gelişmeye açıktır. Carlini ve Wagner (2017) yöntemi bir sızıntı tespit aracı olarak da kullanılabilir. Örneğin optimizasyonda kullanılan $tanh$, log, $softmax$ f sınıflandırma fonksiyonlarının

uygulamada belirtilen sınır koşulları aşılmamak kaydıyla saldırı anında zehirli veriyi ya da sızıntıyı tespit süresince rasgele olarak kullanılabilir. Bu anlamda bir fonksiyonun tespit edemediğini bir diğeri tespit edebilir. Yeni oluşturulabilecek bu fonksiyonların hangisinin sürekli deneme yapılarak modelin isabetlilik oranını aşağıya çekip çekmediği bulunabilir. Bu şekilde daha da genel olarak hedef sistem için kaçınma saldırıları tespitinde en çok zarar verici fonksiyon aileleri saptanabilir.

Uygulamada verilen $tanh$'nin Maclaurin açılımında mümkün olduğu sayıda az terim kullanılması daha basit ifade sağlamakta ve bilgisayarın işlemesini kolaylaştırmaktadır. Bunun yerine söz konusu fonksiyonun belirlenen bir hata üst sınırına sahip olacak şekilde Maclaurin açılımına tabii tutulması bir çalışmanın konusu olarak düşünülebilir. Ayrıca bu fonksiyona Maclaurin haricindeki başka yaklaşımlar da düşünülebilir. Saldıran tarafın, savunma durumundaki makine öğrenmede kullanılan f sınıflandırıcı fonksiyona göre saldırılarındaki değişimleri izleyerek atağı yapan tarafın kaçınma seçimleri öngörülebilir.

KAYNAKLAR

Bishop, C. 2006. Pattern Recognition and Machine Learning. Springer Science-Business Media LLC. 225-290.

Carlini, N., & Wagner, D. 2017. Towards evaluating the robustness of neural networks. 2017 ieee symposium on security and privacy (SP 2017), May 22-24, 39-57. San Jose, USA.

Chris Bolton. 2022. Web Sitesi: https://krisbolton.com/2018/08/22/a-practical-introduction-to-artificial-neural-networks-with-python.html Erişim Tarihi: 27.07.2022

Clarke, B., Fokoue, E., & Zhang, H. 2009. Principles and theory for data mining and machine learning. Springer Science New York, USA.

Czolbe, S., Krause, O., & Feragen, A. 2020. DeepSim: Semantic similarity metrics for learned image registration. 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 11 November, 2-3, Vancouver, Canada.

Daumé, H. 2017. A Course in machine learning. Self-published. 8-72.

Fardo, F. A., Conforto, V. H., de Oliveira, F. C., & Rodrigues, P. S. 2016. A formal evaluation of PSNR as quality measurement parameter for image segmentation algorithms, Centro Universit´ario da FEI, 2-5, Sao Paulo, Brazil.

Goodfellow, I. J., Shlens, J., & Szegedy, C. 2014. Explaining and harnessing adversarial examples. International Conference on Learning Representations (ICLR 2015), May 7-9, 8, San Dieago, USA.

Goodfellow, I., & Bengio, S. 2016. Adversarial examples in the physical world. International Conference on Learning Representations (ICLR 2017), April 24-26, 1-3, Toulon, France.

Gu, J., Cai, H., Chen, H., Ye, X., Ren, J.S., & Dong, C. 2020. PIPAL: a Large-Scale Image Quality Assessment Dataset for Perceptual Image Restoration. SIAT Branch, Shenzhen Institute of Artificial Intelligence and Robotics for Society, 3, Shenzhen, China.

Maimon, O., Rokach, L. 2005. Data mining and knowledge discovery handbook. Springer Science New York, USA.

Mitchell, T. 1997. Machine Learning. McGraw-Hill Science/Engineering/Math, 2-3, New York.

Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2574-2582.

Nilsson, J., & Akenine-Möller, T. 2020. Understanding ssim. 3-6,California,USA.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z., & Swami, A. 2016. The Limitations of deep learning in adversarial settings. European Symposium on Security and Privacy. IEEE, March 21-24, 372-387. Saarbrücken, Germany.

Rafał Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. 2011. HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. Cambridge University, ACM Trans. Graph. 30, 4, Article 40., Cambridge, UK.

Stehman S. 1997. Selecting and interpreting measures of thematic classification accuracy, Remote Sensing of Environment. SUNY College of Environmental Science and Forestry, Volume 62, Issue 1. New York, USA.

Vorobeychik, Y., Kantarcıoglu M. 2018. Adversarial Machine Learning. Morgan & Claypool. 27-41

Yuan, X., He, P., Zhu, Q., & Li, X. 2019. Adversarial examples: Attacks and defenses for deep learning. IEEE Transactions on Neural Networks and Learning Systems, vol. PP, pp. 1–20.

Zhang, L., Zhang, L., Mou, X., & Zhang, D. 2011. FSIM: A feature similarity index for image quality assessment. IEEE transactions on Image Processing, 20(8), 2378-2

EKLER

EK 1 Gradyan Azalış Örneği C Kaynak Kodu

EK 2 Yapay Sinir Ağı Örneği Python Kaynak Kodu

EK 3 PSNR ve Maclaurin Açılımı Kullanan Carlini ve Wagner Python Kaynak Kodu

EK 4 Makine Öğrenme Terimlerinin İstatistik Disiplininde Karşılıkları

EK 1 Gradyan Azalış Örneği C Kaynak Kodu

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
double x[] = {20,22,24,26,28,30,32,34,36,38,40,42};
double y[] = {8.4,9.5,11.8,10.4,13.3,14.8,13.2,14.7,16.4,16.5,18.9,18.5};
double b0 = 0;
double b1 = 0;
double ogrenme_orani = 0.01;
for ( int i = 0; i < 200 ; i ++ ) {
                    int idx = i % 5;
                     double p = b0 + b1 * x[idx];
                     double hata = p - y[idx];
                     b0 = b0 - ogrenme_orani * hata;
                    b1 = b1 - ogrenme_orani * hata * x[idx];
                    printf(" %f ", b0);
                    printf(" %f \n",b1);
}
   printf("\n\n---------------------------------");
   printf("   \n"__DATE__" "__TIME__"\t"__FILE__);
}
```

EK 2 Yapay Sinir Ağı Örneği Python Kaynak Kodu

```
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

dataset = loadtxt('network_agirliklari.csv', delimiter=',')

X = dataset[:,0:2]
y = dataset[:,2]

model = Sequential()
model.add(Dense(2, input_shape=(2,), activation='sigmoid'))
model.add(Dense(2, activation='sigmoid'))
model.add(Dense(2, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X, y, epochs=1, batch_size=1)

_, accuracy = model.evaluate(X, y)
print('Basari orani su cikmistir: %.2f' % (accuracy*100))
```

EK 3 PSNR ve Maclaurin Açılımı Kullanan Carlini ve Wagner Python Kaynak Kodu

```
import sys
import tensorflow    as tf
#import tensorflow_probability as tfp
import numpy as np
import tensorflow.compat.v1 as tfk
import pandas as pd

PSNR_SAY = 0.25
ITERASYON_SAY = 10000
HEDEFLI = True
OGRENME_ORANI = 1e-2
GUVENLI = 0
ADAM_KATSAYISI = 1e-3
ADAM_OPT_SAY = 9

class CarliniL2Modlanmis:
   def __init__(self, sess, model, grup_buyuklugu=1, guven = GUVENLI,
            targeted = HEDEFLI, ogrenme_orani = OGRENME_ORANI,
            adam_opt = ADAM_OPT_SAY, max_iter = ITERASYON_SAY,
            sabit = ADAM_KATSAYISI,
            ustsinir = -0.5, altsinir = 0.5):

      resim_cozunurluk, kanal_say, kac_rakam_var = model.resim_cozunurluk,
                                    model.kanal_say, model.kac_rakam_var
      self.sess = sess
      self.HEDEFLI = targeted
      self.OGRENME_ORANI = ogrenme_orani
      self.ITERASYON_SAY = max_iter
      self.ADAM_OPT_SAY = adam_opt
      self.GUVENLI = guven
      self.sabit = sabit
```

```
        self.grup_buyuklugu = grup_buyuklugu
        self.repeat = adam_opt >= 10
        shape = (grup_buyuklugu,resim_cozunurluk,resim_cozunurluk,kanal_say)
        modifier = tf.Variable(np.zeros(shape,dtype=np.float32))
        self.kanvas = tf.Variable(np.zeros(shape), dtype=tf.float32)
        self.cizdir_kanvas = tf.Variable(np.zeros((grup_buyuklugu,kac_rakam_var)),
dtype=tf.float32)
        self.const = tf.Variable(np.zeros(grup_buyuklugu), dtype=tf.float32)
        self.ata_kanvas = tfk.placeholder(tf.float32, shape)
        self.ata_cizdir_kanvas = tfk.placeholder(tf.float32,
(grup_buyuklugu,kac_rakam_var))
        self.ata_const = tfk.placeholder(tf.float32, [grup_buyuklugu])
        self.kosex = (altsinir - ustsinir) / 2.
        self.kosey = (ustsinir + altsinir) / 2.
        self.yenimaj = tf.tanh(modifier + self.kanvas) * self.kosex + self.kosey
        self.puha = tf.sigmoid((modifier + self.kanvas)) * self.kosex + self.kosey -0.33 *
(modifier + self.kanvas)
        self.fark = self.yenimaj - self.puha
        self.output = model.predict(self.yenimaj)
        self.l2dist = tf.reduce_sum(tf.square(self.fark-(tf.tanh(self.kanvas) * self.kosex +
self.kosey)),[1,2,3])
        gercek = tf.reduce_sum((self.cizdir_kanvas)*self.output,1)
        gercek_olmayan = tf.reduce_max((1-self.cizdir_kanvas)*self.output -
(self.cizdir_kanvas*10000),1)


        if self.HEDEFLI:
            loss1 = tf.maximum(0.0, gercek_olmayan-gercek+self.GUVENLI)
        else:
            loss1 = tf.maximum(0.0, gercek-gercek_olmayan+self.GUVENLI)


        self.loss2 = tf.reduce_sum(self.l2dist)
        self.loss1 = tf.reduce_sum(self.const*loss1)
        self.loss = self.loss1+self.loss2


        start_vars = set(x.name for x in tfk.global_variables())
        optimizer = tfk.train.AdamOptimizer(self.OGRENME_ORANI)
```

```
        self.train = optimizer.minimize(self.loss, var_list=[modifier])
        end_vars = tfk.global_variables()
        new_vars = [x for x in end_vars if x.name not in start_vars]

        self.setup = []
        self.setup.append(self.kanvas.ata(self.ata_kanvas))
        self.setup.append(self.cizdir_kanvas.ata(self.ata_cizdir_kanvas))
        self.setup.append(self.const.ata(self.ata_const))
        self.init = tfk.variables_initializer(var_list=[modifier]+new_vars)

    def attack(self, imgs, targets):
        r = []
        print('devam et',len(imgs))
        for i in range(0,len(imgs),self.grup_buyuklugu):
            print('sayac',i)
            r.extend(self.attack_batch(imgs[i:i+self.grup_buyuklugu],
targets[i:i+self.grup_buyuklugu]))
        return np.array(r)

    def attack_batch(self, imgs, labs):
        def compare(x,y):
            if not isinstance(x, (float, int, np.int64)):
                x = np.copy(x)
                if self.HEDEFLI:
                    x[y] -= self.GUVENLI
                else:
                    x[y] += self.GUVENLI
                x = np.argmax(x)
            if self.HEDEFLI:
                return x == y
            else:
                return x != y

        grup_buyuklugu = self.grup_buyuklugu
```

```
        maclaurin_gelen = (imgs - self.kosey) / self.kosex * 0.999999)

    def PSNR(orjinal, sahte):
        mse = np.mean((orjinal - sahte) ** 2)
        if(mse == 0):

            return 100
        max_pixel = 255.0
        psnr = 20 * log10(max_pixel / sqrt(mse))
        return psnr

    def MACLaurin(hesap):
        s = pd.Series([1,2,3,4,5,6],index = [1, -0.333333, 0.133333, -0.053968, 0.021869, -
0.008863])
        kount = kount + 1
        if (PSNR(imgs,targets) > PSNR_SAY):
            hesap = np.power(maclaurin_gelen,s[kount])*s['kount']
        elif (PSNR(imgs,targets) < PSNR_SAY):
            hesap = np.power(maclaurin_gelen,s[kount])/s['kount']

        imgs = MACLaurin((imgs - self.kosey) / self.kosex * 0.999999)
return hesap

        cevap_en_az = np.zeros(grup_buyuklugu)
        CONST = np.ones(grup_buyuklugu)*self.sabit
        cevap_en_cok = np.ones(grup_buyuklugu)*1e10
        enii = [1e10]*grup_buyuklugu
        skorii = [-1]*grup_buyuklugu
        atkii = [np.zeros(imgs[0].shape)]*grup_buyuklugu
        for dis_adim in range(self.ADAM_OPT_SAY):
            print(enii)
            self.sess.run(self.init)
            batch = imgs[:grup_buyuklugu]
            batchlab = labs[:grup_buyuklugu]
```

```
        bestl2 = [1e10]*grup_buyuklugu
        bestskor = [-1]*grup_buyuklugu
        if self.repeat == True and dis_adim == self.ADAM_OPT_SAY-1:
            CONST = cevap_en_cok
        self.sess.run(self.setup, {self.ata_kanvas: batch,
                                   self.ata_cizdir_kanvas: batchlab,
                                   self.ata_const: CONST})
        prev = np.inf
        for iteras in range(self.ITERASYON_SAY):
            _, l, l2s, skorlar, nimg = self.sess.run([self.train, self.loss,
                                                      self.l2dist, self.output,
                                                      self.puha])
            for e,(l2,sc,ii) in enumerate(zip(l2s,skorlar,nimg)):
                if l2 < bestl2[e] and compare(sc, np.argmax(batchlab[e])):
                    bestl2[e] = l2
                    bestskor[e] = np.argmax(sc)
                if l2 < enii[e] and compare(sc, np.argmax(batchlab[e])):
                    enii[e] = l2
                    skorii[e] = np.argmax(sc)
                    atkii[e] = ii

        for e in range(grup_buyuklugu):
            if compare(bestskor[e], np.argmax(batchlab[e])) and bestskor[e] != -1:
                cevap_en_cok[e] = min(cevap_en_cok[e],CONST[e])
                if cevap_en_cok[e] < 1e9:
                    CONST[e] = (cevap_en_az[e] + cevap_en_cok[e])/2
            else:
                cevap_en_az[e] = max(cevap_en_az[e],CONST[e])
                if cevap_en_cok[e] < 1e9:
                    CONST[e] = (cevap_en_az[e] + cevap_en_cok[e])/2
                else:
                    CONST[e] *= 10
    enii = np.array(enii)
    return atkii
```

EK 4 Makine Öğrenme Terimlerinin İstatistik Disiplininde Karşılıkları

İSTATİSTİK	MAKİNE ÖĞRENME
Bağımlı değişken	Etiket, çıktı
Bağımsız değişken	Özellik, girdi
Tahmin	Öğrenme
Veri, data	Öğrenme verisi ile test verisi
Parametre tahmini	Tahmin
Parametre	Ağırlık
Regresyon, Sınıflama	Gözlem altında öğrenme paradigması
Kümeleme Analizi, Yoğunluk tahmini	Gözlem altında olmayan öğrenme paradigması
Süreç temelli dinamik istatistiksel yöntemler (Markov zincirleri vb.)	Pekiştirmeli öğrenme
Çok değişkenli, çoklu regresyon	Sinir ağları
Model	Hipotez
Parametre uzayı	Hipotez sınıfları
Hata terimleri	Tahmin hatası, kestirim hatası,test hatası
Örneklem	Öğrenme ile test verisi
Yığın	Karşılığı yok
(Modelde) Sabit terim	Yanlılık