T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY SİNİR AĞI KULLANARAK GÖĞÜS KANSERİ HASTALIĞININ TAHMİNİ

YÜKSEK LİSANS TEZİ

Mariya Kiknadze

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

MART 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY SİNİR AĞI KULLANARAK GÖĞÜS KANSERİ HASTALIĞININ TAHMİNİ

YÜKSEK LİSANS TEZİ

Mariya Kiknadze
(Y1613.010041)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi AHMET GÜRHANLI

MART 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

YÜKSEK LİSANS TEZ ONAY FORMU

Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı Y1613.010041 numaralı öğrencisi MARIYA KIKNADZE'nin **"Yapay Sinir Ağı Kullanarak Göğüs Kanseri Hastalığının Tahmini"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 24.02.2020 tarihli ve 2020/03 sayılı kararıyla oluşturulan jüri tarafından oybirliği/oyçokluğu ile Tezli Yüksek Lisans tezi 12.03.2020 tarihinde kabul edilmiştir.

	Unvan	Adı Soyadı	Üniversite	İmza
		ASIL ÜYELER		
Danışman	Dr. Öğr. Üyesi	Ahmet GÜRHANLI	İstanbul Aydın Üniversitesi	
1. Üye	Prof. Dr.	Ali GÜNEŞ	İstanbul Aydın Üniversitesi	
2. Üye	Doç. Dr.	Zeynep ORMAN	İstanbul Üniversitesi-Cerrahpaşa	
		YEDEK ÜYELER		
1. Üye	Dr. Öğr. Üyesi	Adem ÖZYAVAŞ	İstanbul Aydın Üniversitesi	
2. Üye	Dr. Öğr. Üyesi	Ali HAMİTOĞLU	İstanbul Sabahattin Zaim Üniversitesi	

ONAY

Prof. Dr. Ragıp Kutay KARACA
Enstitü Müdürü

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum **"YAPAY SİNİR AĞI KULLANARAK GÖĞÜS KANSERİ HASTALIĞININ TAHMİNİ"** adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşünecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. **(12/03/2020)**

Mariya KIKNADZE

ÖNSÖZ

Tez çalışmamı hazırladığım zorlu sürecin her aşamasında öncelikle, bana yol gösteren, her konuda yardımcı ve destek olan, her soruma sabırla yanıt veren ve tez çalışmasının son halini almasında eksik taraflarının çıkarılması ve giderilmesi kapsamında gayretlerini eksik etmeyen ve değerli zamanını harcayan tez danışmanım Sayın Dr. Öğr. Üyesi Ahmet GÜRHANLI'ya, başta Bilgisayar Mühendisliği Bölüm Başkanı Prof. Dr. Ali GÜNEŞ olmak üzere, İstanbul Aydın Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans öğretim üyelerine, tez çalışmamı yaparken maddi ve manevi desteğini eksik etmeyerek zamanını ayıran değerli arkadaşım Yunus Emre ARAÇ'a, çalışmam sürecince büyük ilgi ve fedakarlıklarıyla her zaman yanımda olan aileme, bana bu imkanı sağlayan ve desteklerini eksik etmeyen Filiz KOÇER ve Selahaddin KOÇER'e teşekkürlerimi sunmayı bir borç bilir, sonsuz minnettarlığımı sunarım.

MART 2020 Mariya Kiknadze

İÇİNDEKİLER

KISALTMALAR

YSA : Yapay Sinir Ağı

SGD : Stokastik gradyan inişi

NAG : Nesterov hızlandırılmış gradyan

LVQ : Linear Vector Quantization

ART : Adaptif Rezonans Teori

BC : Meme Kanseri

WDCB : Wisconsin Diagnostic Breast Cancer

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

YAPAY SİNİR AĞI KULLANARAK GÖĞÜS KANSERİ HASTALIĞININ TAHMİNİ

ÖZET

Günümüzde meme kanseri (breast cancer) dünyadaki en önemli kötü huylu hastalıklardan biridir. ABD'de meme kanseri, kadınlarda tüm onkolojik hastalıklar arasında birinci sırada yer alır ve akciğer kanserinden sonra onkolojide ölüm nedeninin ikincisidir. Meme kanserinin erken teşhisinde ve tedavisinde son zamanlarda elde edilen büyük başarılara rağmen, ilk aşamalarda teşhisi için yeni yaklaşımlar ve algoritmalar geliştirilmeye devam etmektedir. Meme kanseri, diğer kötü huylu hastalıklar gibi birçok sınıflandırmaya sahiptir. Histolojik, moleküler, fonksiyonel, TNM sınıflandırması bunlardan bazılarıdır. Çoğu kanser vakası hastalığın geç aşamalarında ancak teşhis edilebilir ve tedavi sıklıkla cevap vermez ve hasta kaybedilir. Bu sebepten meme kanserinin erken evrelerde teşhisi hayati önem taşır. Bu çalışmada sınıflandırma testi doğruluğunu, hassasiyet ve özgüllük değerlerini ölçerek sunmakta olan Wisconsin Meme Kanseri Teşhisi (WDBC) veri seti kullanılmaktadır. Uygulamada, veri seti eğitim aşaması için %70 ve test aşaması için %30 olarak bölünmüştür. Bu çalışma yapay sinir ağı kullanarak meme kanseri tahmininde optimizasyon algoritmalarının ve parametrelerin nasıl seçilmesi gerektiğini incelemekte ve farklı seçimlerinin nasıl sonuç verdiğini göstermektedir.

Anahtar Kelimeler: *Yapay Sinir Ağları; Meme Kanseri; Meme Kanseri Tahmini*

PREDICTION OF BREAST CANCER USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Breast cancer is one of the most important malignant diseases in the world. In the United States, breast cancer ranks first among all oncological diseases in women and is the second leading cause of cancer mortality after lung cancer. Despite recent great success in the early detection and treatment of breast cancer, new approaches and algorithms are still being developed for early diagnosis. Breast cancer has many classifications, like other malignant diseases: histological, molecular, functional, TNM classification. Most cases of cancer can be diagnosed in the later stages of the disease, and treatment is often not responding and the patient is lost. Therefore, early detection of breast cancer is vital. This study uses the UCI Breast Cancer Wisconsin (Diagnostic) Data Set (WDBC), which is presented by measuring test classification accuracy, sensitivity, and specificity values. The data set was divided into 70% for the training phase and 30% for the testing phase. This study demonstrates the importance of optimization algorithm selectiona and parameters in the diagnosis of Breast Cancer using Artificial Neural Networks and investigates how they should be chosen. The accuracy results of different optimization algorithms and parameter values are reported.

Keywords: *Artificial Neural Networks; Breast Cancer; Breast Cancer Diagnosis*

1. GİRİŞ

Makine öğrenimi şu anda birçok bilim ve üretim alanlarında kullanılmaktadır. Tıp da bu alan için bir istisna değildir. Makine öğrenimi sayesinde hastaları sınıflandırmak, en uygun tedavi yöntemini belirlemek, bir hastalığın süresini ve sonucunu tahmin etmek, komplikasyon riskini değerlendirmek, belirli bir hastalık tipinin en karakteristik sendromlarını bulmak gibi birçok görev çözülmüştür. Meme kanseri, normal glandüler hücrelerin kansere dönüşmesinden kaynaklanan bir hastalıktır. Dünyada, meme kanseri kadınlar arasında en yaygın kanser türüdür. Kadınlar arasında yaşam süresi boyunca 13 ile 90 yaş arası, 13 kişiden biri ya da 9 kişiden biri bu hastalığa yakalanmaktadırlar (Aleksandroviç, Ryazanov, 2016). Diğer birçok kanserde olduğu gibi, meme kanserinin erken teşhisi hayat kurtarabilir. Dolayısıyla meme kanserinin erken evrelerde kesin tanı koyulması hastanın yaşam kalitesini mümkün olan en iyi seviyede tutmak için çok önemlidir. Bununla birlikte, düzenli mamogramlar bile bu hastalığın zamanında teşhisini garanti etmez. ABD bilim adamları, göğüs yoğunluğunu otomatik olarak sınıflandırmak ve böylece meme kanserini tespit etmek için veri tabanlı yazılım geliştirmişler. Testler bu sistemin insan radyologları kadar doğru bir "teşhis uzmanı" olduğunu göstermiştir (Wolberg, Street, Mangasarian,1992). Bu algoritma, göğüsün yoğunluğunun net bir tanıya izin vermediği durumlarda doktorlara yardımcı olabilir. Makine öğrenme modellerinin uygulanması hastalık tahmini ve prognozu için daha sonradan hastaların tedavisini iyileştirmeyi amaçlayan kanser çalışmalarının ayrılmaz bir parçası haline geldi. İlgili meme kanseri çalışmalarından elde edilen iki veri seti, iyi performans gösteren ve veri kaybı olmayan uygun verileri ve grafiksel veritabanları kullanarak yatay ve dikey entegrasyona dayalı bir veri entegrasyonu yaklaşımı uygulanarak birleştirilir.

Donald Hebb'ın (1949) modern sinir ağları teorisini bulduğu bilinmektedir. Nörolog Hebb beynin nasıl öğrendiğini inceledi. Beynin çalışmasının en temel birimi sinir hücresi iki sinir hücresi birbiriyle nasıl ilişkilidir ve sinir ağları teorisini bu temele dayandırdı. Hebb'ın bu temele dayanarak fikir başlatıldı ve yüzlerce teoriğe sahip olmaktadır. Günümüzde gerçek hayatımızda kullanılan başarı oranı %99 olan birçok yapay sinir ağı (YSA) modeli vardır. Yapay sinir ağı ile makine öğrenmesi , görüntü işleme (Shi, He, 2010 ve Ramirez-Quintana, Chacon-Murguia, Chacon-Hinojo 2012), karakter tanımı, sınıflandırma, tahmin, kümeleme, ses işleme (Uncini,2003), veri filtreleme ve en uygun şekle sokma gibi birçok uygulama yapmak mümkündür. Bu alanlarda yapay sinir ağlarının tercih edilmesinin temel nedenlerinden biri, kullanılan algoritma ne olursa olsun her tür veri, öğrenme hatalarını en aza indirmek ve bu nedenle gerçekçi tahmin yapabilmektedir. Meme kanserinde klinik veri seti temelinde doğru tahmin yapılabilmesi için Yapay Sinir Ağı modelinin doğru optimizasyon algoritması ile uygulanması ve parametre aralıklarının doğru belirlenmesi kritik önem taşır.

2. LİTERATÜR ÇALIŞMASI

Bu konuyla ilgili birçok çalışma yapılmıştır. Aslında makine öğrenimi matematiksel istatistiklerin birleştiği yerde, optimizasyon yöntemleri ve klasik matematiksel disiplinler, aynı zamanda hesaplama verimliliği ve yeniden eğitim sorunları ile ilişkili kendi özellikleri vardır. Birçok endüktiv eğitim yöntemi klasik istatiksel yaklaşımlara alternatif olarak geliştirilmiştir. Yapılan çalışmalarda birçok yöntem, bilgilerin çıkarılmasıyla ve yapay veri analizi (Data Mining) ile yakından ilgilidir.

Xrulyov K.A.ve Ryazanov M.A. (2016) Azure Machine Learning ile meme kanseri tanısında teşhis için incelenen hastalar hakkındaki verilerin analizini kullanarak bir veb servisi geliştirmişler.

Fogel D. B., Wasson E.C., Boughton E.M. ve Porto V.W. (1997) hasta yaşına sahip sinir ağları ile radyoaktif özellikleri kullanarak meme kanseri tespiti için veri analizi çalışmasını yapmışlardır.

Revett K., Gorunescu F., Gorunescu M., El-Darzı E. ve Ene M.,(2005) ve Gorunescu M., Gorunescu F., ve Revett K.,(2007) ham kümeler ve muhtemel sinir ağları içeren hibrid bir modele dayanan bir meme kanseri tıbbi modeli için bir karar destek sistemi geliştirmişler.

Hsiao Y.H., Huang Y.L., Liang W.M., Kuo S.J. and Chen D.R., (2009) vasküler parametreler (harmonik ve harmonik olmayan 3D Dopplerografi) kullanarak iyi veya kötü huylu göğüs tümörlerinin belirlenmesi için bir MLP sınıflandırıcı analizi çalışmasını yapmışlardır.

E.Harwich, K.Laycock., (2018) ve JASON The MITRE Corporation (2017) "Birleşik Krallık"da ingiliz bilim adamları "Ulusal Sağlık Sisteminde Yapay Zeka" ve ABD' nin öndegelen amerikalı teknoloji bilim adamı Jason "Sağlık ve Sağlık Hizmetleri İçin Yapay Zeka" adlı 2017 yılında çalışdıkları bir rapor yayınladı. Her iki çalışmada Yapay Zeka kullanarak genel nüfuza yüksek nitelikli tıbbi bakım sağlanılması analiz olmuşdur. Kanser tanısı alanında Yapay Zeka kullanımı, Yapay Zekanın görevleri ve yöntemleri hakkında çalışma yapılmıştır.

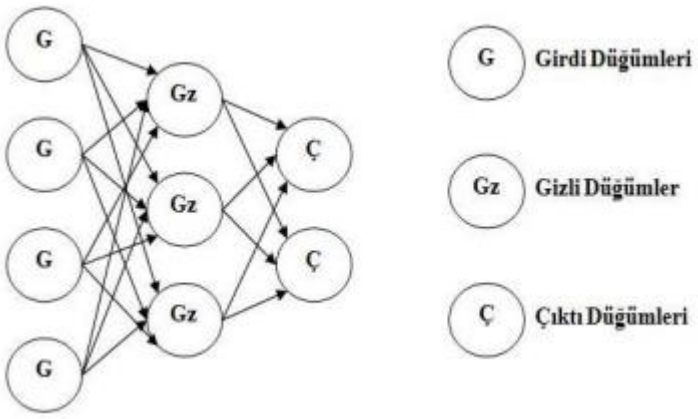
Mihaylov.I , Nisheva.M , and Vassilev.D (2019) doğru teşhis için makine öğrenme modellerini kullanarak meme kanserinde sağ kalım süresinin klinik verilere dayanarak öngörülmüsini sağlayan çalışma yapılmışdır. Bu çalışmada hastanın hayatda kalma süresinin, tümör evresini, tümör boyutunu ve yaşının tanısını orijinal olarak geliştirilen tümörle entegre klinik özellik olduğnu tahmin etmektedir. Çalışmada veri normalizasyonu ve sınıflandırmasının yanı sıra, uygulamalı makine öğrenimi yöntemi sağkalım süresi tahmininin doğruluğu açısından umut verici sonuçlar vermektedir. Bu çalışmada doğrusal destek vektör regresyonu, çekirdek ridge regresyonu, K en yakın komşu regresyonu, karar ağacı regresyonu ve kement regresyonu modelleri en doğru yaşam prognozu sonuçlarını elde etmişler. Aynı yöntemleri kullanarak meme kanseri verileri üzerindeki performansı için önerilen yaklaşım olarak Python tabanlı iş akışı geliştirmişler.

3 YAPAY SİNİR AĞLARI (YSA)

3.1 Yapay Sinir Ağının Tanımı

Yapay sinir ağları, herhangi bir yardım almadan insan beyninin özelliklerinden biri olan eğitim yoluyla yeni bilgi alma, yeni bilgi oluşturma ve keşfetme yeteneği gibi becerileri otomatik olarak uygulamak için tasarlanmış bilgisayar sistemleridir. Diğer bir deyişle biyolojik sinir ağlarını simüle eden bilgisayar programlarıdır. Yapay Sinir Ağları zamansal bağlantı, paralel dağıtılmış işleme, sinirsel işleme, doğal zeka sistemleri ve buna makine öğrenme algoritmaları da dahil isimleri birlikte anılmakta ve bazen aralarındaki farklılıkları karşılaştırılmaktadır. Yapay Sinir Ağları geleneksel programcı becerileri gerektirmeyen kendi kendine öğrenme cihazıdır. YSA öğrenmenin yanı sıra ezber ve bilgi arasındaki ilişki yaratma yeteneğinede sahiptir (Elmas, 2007).

Yapay sinir ağlarının günümüzde birçok sorunu çözebilme yeteneği var. Tanımlarının birden fazla ortak özelliği var. Başlangıç olarak yapay sinir ağlar hiyerarşik olarak bağlanabilir ve paralel olarak çalışabilir hücrelerden oluşurlar. İşlem (proses) hücreleri olarak da adlandırılan bu hücreler, ilgili olmaları gerekir ve her bağlantı önemlidir çünkü her bağlantının bir değerinin olduğu kabul edilmiştir. (Öztemel, 2006). Yapay sinir ağı modeli aşağıdaki gibi basitçe gösterilmiştir.



Şekil 3.1: Yapay Sinir Ağı Modeli

3.2 Yapay **Sinir Ağlarının Genel Özellikleri ve Eksiklikleri**

YSA uygulanan ağ modeline göre çeşitli karakteristik özellikler gösterir ve bu özellikleri aşağıdaki gibidir.

- **YSA makine öğrenmesini yapar:** Yapay Sinir Ağları bilgisayar eğitimine odaklanır. Çalışılan olaylar arasındaki benzerliklerden faydalanır.
- **Yapay sinir ağları bilgiyi saklar:** Yapay sinir ağlarının iletişim değerleri, bilginin önemi tanımlar ve bilgiler bağlantılarda saklanır. Diğer programlar gibi veri veritabanı kullanılmazsa, bilgiler ağda saklanır.
- **Yapay sinir ağları yapılan çalışmaların örneklerini kullanarak öğrenirler:** Yapay sinir ağından olayları bilmek için bu olayla ilişkili örnekleri tanımlamanız gerekir. Bu örnekleri kullanarak olayla ilgili ağı özetleme yeteneğinin gelmesi sağlanır. Yapay bir sinir ağını örneksiz eğitmek imkansızdır.
- **Sınıflandırma ve örüntü ilişkilendirme yapabilirler:** Genel olarak ağlar çoğunun amacı kendilerine örnek olarak verilen şablonlar veya başkaları ile bağlılığıdır. Sınıflandırmada ise örnekleri kümeleyerek belirli sınıflara ayırır ve sonraki seçimde hangi sınıfa gireceğine karar vermesi amaçlanır.
- **Kendi kendini öğrenebilme ve organize etme kimi yetenekleri vardır**: YSA yeni durumlara uyarlanması, örneklerle gösterilmesi ve sürekli olarak, yeni olaylar öğrenebilmesi mümkündür.
- **Yapay sinir ağları eksik bilgi ile çalışır:** Yapay sinir ağları eğitimden sonra eksik bilgi ile çalışabilir ve yeni örneklerde bilgi olmamasına rağmen sonuç üretebilirler. Bu, ağ performansının düşmesine neden olacağı anlamına gelmemelidir, çünkü performansı eksik bilgilerin önemine bağlıdır. Hangi bilgiler önemlidirse ağ eğitim sırasında öğrenir. Kullanıcının bu konuda hiçbir fikri olmuyor, eğer performans düşerse eksik bilgilerin önemli olduğu düşünülür.
- **Yapay sinir ağlarında hata toleransı vardır:** Eksik bilgi ile çalışması için yapay sinir ağlarının hatalara toleranslı olmasını sağlar. Bazı hücreler ağda bozulmasına rağmen ağ çalışmaya devam ediyor. Bozuk hücre sorumluluk değeri performansın düşmesine neden olabilir.
- **Yapay sinir ağları dağıtık belleğe sahiptirler:** Yapay sinir ağlarında, bilgi ağa dağılmış oluyor. Hücre bağlantı değerleri ağ bilgilerini gösterir. Tüm ağ bilgi çalıştığı tüm olayı karakterize ettiği için ağa dağılmıştır.
- **Yapay sinir ağları sadece nümerik bilgilerle çalışa bilir:** Yapay sinir ağı sistemine dahil olan bilgiler numerik olmalıdır. Semboller veya görüntüler numerik olarak ifade edilirerek ağa eklenilmelidir. (Öztemel, 2006; Ergezer vd., 2003).

Yapay sinir ağlarındaki eksiklikler ise aşağıdaki gibidir.

1) YSA'lar donanıma bağlı ve ağ paralel işlemciler üzerinde çalışabilir. Modern makinelerin çoğu sırayla ve eşit olarak çalışırak bir kerede tek bir bilgiyi

işleyebilir. Seri makinelerde paralel işlemler gerçekleştirme zaman kaybı olabilir.

2) Bir probleme uygun ağ yapısını belirlemek için kullanılan genel teknik deneme yanılma yöntemine sahip olmak önemli bir dezavantajdır. Böylece çözümün çoğu iyi bir çözüm sağlama problemi olacağından yapay sinir ağları kabul edilebilir sonuç verir. Ancak, bu daha iyi bir çözümü garanti etmez.
3) Bir ağ oluştururken, parametre değerleri (proses eleman sayısı, öğrenme katsayısı vb.) belirlenmesinde kullanılan bir kuralın olmaması önemli bir dezavantajdır. Bu seçeneklerin kullanılabilirliği kullanıcının deneyimine bağlıdır. Bu parametre değerleri için belirli standartlar oluşturmak çok zordur, bu nedenle her sorun için ayrı ayrı değerlendirmeler yapılmalıdır.
4) Yapay sinir ağları yalnızca numerik değerlerle çalışma ekranda önemli bir kusuru temsil eder. Problem numerik temsiline dönüştürülmelidir. Uygun bir ekran motoru kurulamaması düşük etkili eğitim elde edilir, çünkü bu sorunun çözümünü önleyecektir.
5) Ağ eğitiminin ne kadar zaman alacağına dair herhangi bir karar yöntem tanımlanmamıştır. Belirli bir değerin altındaki hataları azaltmak için ağ eğitimi tamamlanması için yeterli kabul edilse bile, en iyi öğrenmenin gerçekleştiği anlamına gelmez.
6) En önemli dezavantaj, ağın davranışının açıklanamamasıdır. Karar verildiğinde bunun nasıl ve neden olduğunu öğrenebilirsiniz (Şen 2004; Öztemel 2006)

3.3 Yapay Sinir Ağlarının Çalışma Prensibi

Kendisine gösterilen girdi setine uygun yapay sinir ağları çıktı setini belirleyen mekanizmalardır. Ağ, bunları sağlamak için ilgili örnekleri kullanarak genelleme becerisini öğrenir ve kazanır. Eğitim genellikle bir ağ örneği olarak hizmet eder ve çıkış kümeleri ile yapılır. Ağın bu eğitim yoluyla katkılara cevap vermesi beklenen sonuçları çıkarmayı öğrenir. Bir genellemeye dayanarak, benzer bir girdiye yanıt olarak ağ gelebilecek sonuçları otomatik olarak belirleyebilir (Öztemel, 2006).

Yapay sinir ağlarının mimari bölümü ve fonksiyonel bölümü temel özelliklerini ele almaktadır. Mimari yapı ağ topolojisini tanımlar. Bu mimari yapı, ağdaki nöronların sayısını ve birbirleriyle olan bağlantılarını belirler. Ağ da çok sayıda nöron veya benzer özelliklere sahip diğer adlandırmalarla proses elemanları birbirine bağlayarak oluşur. Ağ öğrendiklerini öğrenme veri depolamak, birleştirmek, mevcut bilgileri yeni bilgilerle birleştirmek, yeni bilgileri karşılaştırır, sınıflandırır ve gerekirse sınıflandırma gelişimi ağın işlevsel bir özelliğidir (Kartalopoulos, 1996). Ağ topolojisini, ağırlıklandırma faktörlerini, aktivasyon parametrelerini ve başka ağ parametreleri uygun ağa öğrenme yöntemi kullanılarak be Yapay sinir ağları geleneksel işlemciler gibi çalışmaz. bir dizi

Geleneksel işlemcilerden farklı olarak yapay sinir ağları, seri sistem halinde olarak değil, her biri problemin bir kısmı ile ilgilenen çok sayıda basit 11 işlem öğesinin paralel çalışması ile çözülmektedir. Proses elemanları girdiği ağırlık ağı ile ağırlıklandırılır, doğrusal olmayan dönüşüm çıktı değeri sağlar ve üretir (Kalach, 2005). Matematiksel fonksiyon ağ mimarisi tarafından belirlenir. Üretilmiş çıkışlar belirli bir hata değerinin altına düşene kadar ağırlık değerlerini değiştirerek gerekli ağırlık değerleri elde edilir. Ağda gösterilen örnekler ve çıktılar arasındaki ilişki genelleme düzeyini ortaya çıkarır ve öğrenir. Çıkan ağın hata payı, beklenen sonuç ile karşılaştırılarak elde edilir. Bu hata payı ağ performansını belirlemektedir. Geri yayılım algoritması (backpropagation) ile, istenen değere kadar hata payı ağırlığı arttırmak için ağırlığı ayarlayabilirsiniz. Bu süreç optimum çözüme ulaşana kadar tekrarlanabilir (Yurtoğlu, 2005).

3.4 Yapay Sinir Ağlarının Tarihçesi

Yapay sinir ağlarının tarihçesi, insanların nörobiyoloji konusuna ilgisi ve elde etdikleri bilgileri bilgisayar bilimi uygulamaları ile başlar. 1970"li yıllardan sonra, araştırmalar önemli ilerlemeler göstermiş ve kaydedilmiştir.

1943'te ilk yapay sinir ağı bilimcisi Walter Pits ve bir nörolog olan Warren McCullough ile insan beyninin hesaplama yeteneklerinden esinlenen elektrik devreleri kullanarak başladı.

1949'da Hebb tarafından geliştirilen Davranış Organizasyonu adlı kitabında, çalıştığı konuyu incelemenin temel teorisi Hebbian eğitiminin bir kuralı olarak tanımlanmaktadır. Bu gelişmiş kuralın YSA bileşiklerinin sayısı değiştirilerek incelenebileceği belirtilmiştir (Öztemel, 2006).

1954'te Farley ve Clark "Random Networks" ve "Adaptive Response" terimlerini tanıttılar ve bu konsept 1958'de Rosenblatt ve 1961'de Cainiello tarafından geliştirildi (Kargı V, 2015).

Tanıma amacıyla tasarlanmış ve eğitilmiş tek çıkışlı tek katmanlı yapay sinir ağı 1958'de Rosenblatt tarafından geliştirilmiş ve "Perceptron" adını almıştır. Bu yapı daha da geliştirildi ve çok katmanlı sinir ağlarının altında yatan devrimci bir çalışma olarak kabul edildi (Yücesoy M., 2011).

1960'larda matematikçi Minsky ve Peypert, sensörlerin Perceptrons adlı kitaplarında doğrusal olmayan sorunlara bir çözüm sağlayamadığını ve yapay bir sinir ağının XOR problemini çözemediğini kanıtladılar. Bu nedenle, yapay sinir ağlarına olan güven azalmış ve yatırımlar azalmıştır.

Bu süreç 1982'ye kadar devam etti ve Hopfield çok katmanlı sensörler kullanılarak ayrı sensörlerle çözülemeyen XOR sorunlarını çözdü. Bu gelişme ile YSA'ya olan güven tekrar geri döndü.

1959'da Widrow ve Hoff tarafından tasarlanan Uyarlanabilir Doğrusal Nueron, mühendislik uygulamalarında YSA kullanmanın ilk adımı oldu. ADALINE modelinin çok katmanlı bir versiyonu olan MADALINE, 1970'lerin sonunda ortaya çıktı (Öztemel, 2006). MADALINE modeli daha sonra telefon hatlarındaki yankıyı ortadan kaldırmak için kullanıldı.

1986'da Rumelhart ve arkadaşları, yapay sinir ağlarında yaygın olarak kullanılan bir geri yayılım algoritması geliştirdiler. Geri yayılım algoritmasını kullanarak, tek katmanlı ağların çözemediği XOR sorunlarını çözebildiler.

1988'de Broomhead ve Lowe tarafından geliştirilen RDF (Radyal tabanlı fonksiyonlar) modeli, çok katmanlı sensörler için alternatif bir ağ haline gelmiştir. Çoğu zaman filtreleme ve veri sıkıştırma görevlerini çözmede kullanılır.

1990'da (Probabilistic Nueral Network PNN) Olasılık Sinir Ağı'nı geliştiren Spetch, daha gelişmiş bir radyal özellik haline geldi. 1991'de daha gelişmiş RNN'leri (Genelleştirilmiş Regresyon Ağları) geliştirdi.

Günümüzde, özellikle yapay sinir ağları, finans, tıp, fizik, ekonomi vb. eknoloji ve bilim alanında birçok sektöre girmeye başlamış ve uygulanmaya devam etmektedir.

3.5 Yapay Sinir Ağlarının Kullanım Alanları

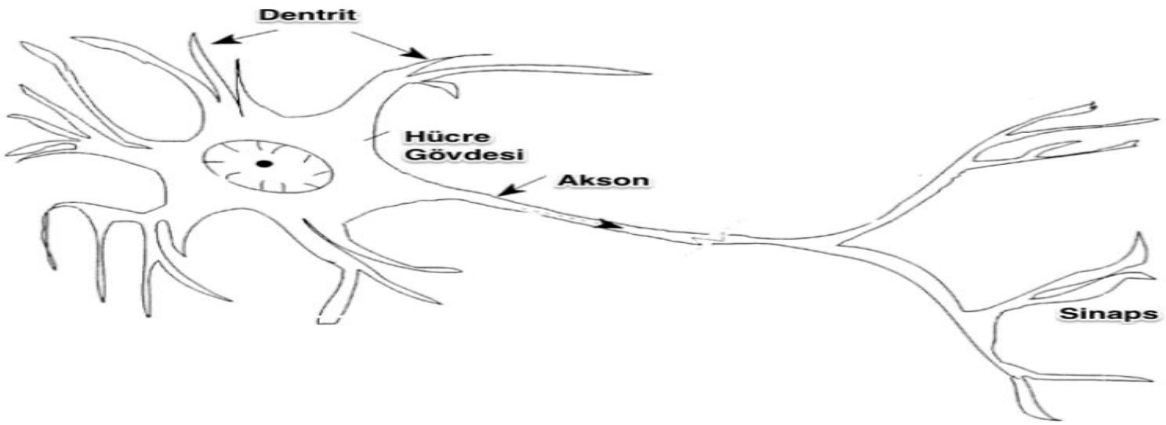
Yapay sinir ağı (YSA) uygulamaları temel olarak çoğu sınıflandırma, veri birleştirme, veri kavramlaştırması, veri süzülmesi, resim veya görüntü işleme sınıflarından birine girmektedir (Elmas, 2007).
Teorik uygulamalara ek olarak, günlük hayatta kullanılan finansal araç konularından mühendislik ve tıp bilimlerine kadar birçok uygulama hakkında bahs etme mümkündür. Veri madenciliği, optik karakter tanıma ve çek okuma, kredi kartı hilelerini saptama, bankalardan kredi isteyen müracaatları değerlendirme, ürünün pazardaki performansını tahmin etme,zeki araçlar ve robotlar için optimum rota belirleme, robot hareket mekanizmalarının kontrol edilmesi, güvenlik sistemlerinde konuşma ve parmak izi tanıma, mekanik parçalarının ömürlerinin ve kırılmalarının tahmin edilmesi, kalite kontrolü, iletişim kanallarındaki geçersiz ekoların filtrelenmesi, radar ve sonar sinyalleri sınıflandırılması, kan hücreleri reaksiyonları ve kan analizleri sınıflandırma, beyin modellenmesi çalışmaları bu uygulamaların bazılarıdır (Öztemel, 2006)

4. YAPAY SİNİR AĞLARININ YAPISI VE TEMEL ELEMANLARI

4.1 Biyolojik Sinir Hücresi

Yapay sinir ağlarını, biyolojik sinir hücrelerini ve bunu daha iyi anlamak için hücreler tarafından oluşturulan sinir ağları iyi anlaşılmalıdır.

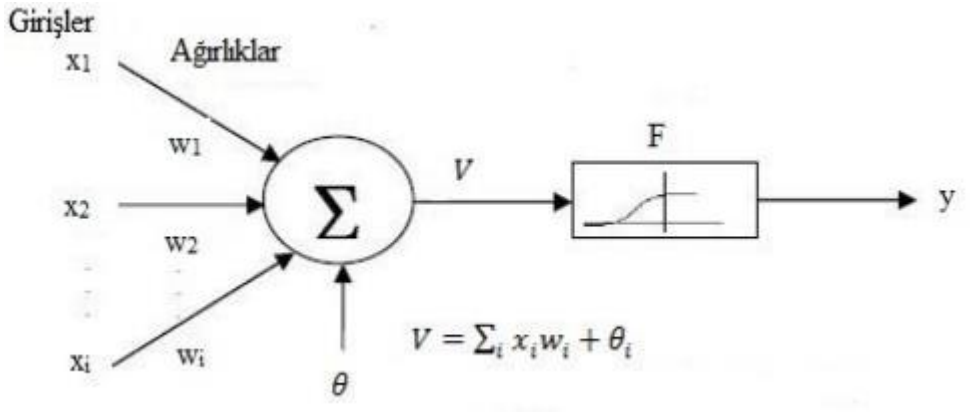
Biolojik sinir hücresi sinapsları, çekirdek, hücre gövdesi (soma), akson ve dendritlerden oluşmaktadır. Sinapslar, sinir hücreleri arasındaki ilişkiler nasıl olduğunu görebilirsiniz. Bunlar fiziksel bağlantılar değil, bir hücreden diğerine elektriktir sinyallerinin geçmesine izin veren yapılardır. Bu sinyaller somaya gider ve çekirdek onları işler. Sinir hücresi kendi elektrik sinyalini oluşturarak aksonunu yoluyla dendritlere gönderir. Bu sinyalleri dendritik sinapslara göndererek, diğer hücrelere iletilir. Bu fonksiyonda, milyarlarca sinir hücresi birleşerek sinir sistemi oluşur. Biyolojik hücrelerin bu özelliklerini kullanan yapay sinir ağları geliştirilebilir (Öztemel, 2006)



Şekil 4.1: Biyolojik Sinir Hücresi

4.2 Yapay Sinir Hücresi

Yapay sinir ağları yapısına göre bakıldığında biyolojik sinir ağlarına benzemektedir. Yapay sinir ağları nöronlarda kendi aralarında bağ kurarak paralel çalışan sistemdir. Bu yapay sinir ağları birbiriyle bağlantılı birçok düğümden oluşur ve veri girişi, veri işleme ve veri çıkışı şeklinde çalışmaktadır. Bir yapay sinir ağı girdi değerleri, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıkış değeri olarak bu bölümlerden oluşmaktadır (Elmas 2007).



Şekil 4.2: Yapay Sinir Ağı Hücresi

Burada x ve y giriş ve çıkış, toplama fonksiyonu ve θ değeri, F aktivasyon fonksiyonu, w ağırlıktır.

4.2.1 Giriş değerleri

Girdiler olarak da adlandırılan giriş değerleri, biolojik sinir ağında dentritler tarafından elde edilen bilgilerdir. Bu bilgi önceki sinirlerden veya dış dünyadan gelir. Genellikle girdiler yapay sinir ağını öğrenmek isteyen örnekler tarafından belirlenir.

4.2.2 Ağırlıklar

Ağırlıklar, girdilerin yapay sinir ağı üzerindeki etkisini belirler ve katsayılar olarak adlanır (Elmas, 2007). Her girişin kendi ağırlığı vardır yani, $x_1$girişi w_1 ağırlığına sahiptir. Ağırlık negatif ve pozitif olabilir. Ağa giriş derecesine bağlı olarak ağırlıklar sıfır olabildiği gibi ağırlıklar giriş değerlerinin ağa bağlanma dercelerine göre değişken ve ya sabit değer ala bilir.

4.2.3 Toplama fonksiyonu

Yapay sinir hücresindeki toplama işlevi bir hücreye gelen net giriş değerini hesaplar. Bunu yapa bilmek için değişik işlevler kullanılır. En çok kullanılanı ağırlık toplamını bulmaktır (Öztemel, 2006). Bu yöntemde gelen her girdi kendi ağırlığı ile çarpılır ve toplamı bulunur. Bu işlemin sonunda net giriş değerleri yani girdi bulunmaktadır. Girdi değerlerini X ve ağırlık değerlerini W olarak adlandıracak olursak N tane girdiyi bu fonksiyona göre aşağıdaki gibi tanımlayabiliriz.

$$= \sum_i^n X_i W_i \quad (4.1)$$

Toplama fonksiyonu yerine kullanılabilecek farklı fonksiyonlarda vardır. Bu fonksiyonlardan en sık kullanılanları aşağıdaki tablo 1 de yer almaktadır.

Çizelge 4.1: Toplama Fonksiyonları

Fonksiyon	Formül
Toplam	$= \sum_{k=1}^{N} X_k W_k$
Çarpım	$= \prod_{k=1}^{N} X_k W_k$
Maksimum	$= Max\ (X_k W_k)$
Minimum	$= Min\ (X_k W_k)$
Çoğunluk	$= \sum_{k=1}^{N} Sgn(X_k W_k)$
Kumilatif Toplam	$= Net(Eski) + \sum_{k=1}^{N} X_k W_k$

4.2.4 Aktivasyon (Transfer) fonksiyonu

Aktivasyon fonksiyonu hücreye gelen ağ giriş değerini işleyerek hücrenin bu girişe yanıt olarak üreteceği çıktıyı belirleyen bir işlevdir. Toplama fonksiyonunda olduğu gibi, aktivasyon fonksiyonu çıktıyı hesaplamak için kullanılır. Sorunu çözmek için en uygun fonksiyon tasarlanan çalışmanın denemeleri sonucunda tespit edilir. En çok kullanılan "çok katmanlı" bir sensör modelidir (Öztemel, 2006). Genellikle aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmaktadır. YSA'ların genel bir özelliği olan "Doğrusal Olmama" özelliğine göre eğer geri beslemeli bir yapay sinir ağı kullanılacak ise burada türevi alınabilecek bir aktivasyon fonksiyonu seçilmelidir. Bunun sebebi ise bu sinir ağında aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplama işleminin yavaşlamasını engellemektir.

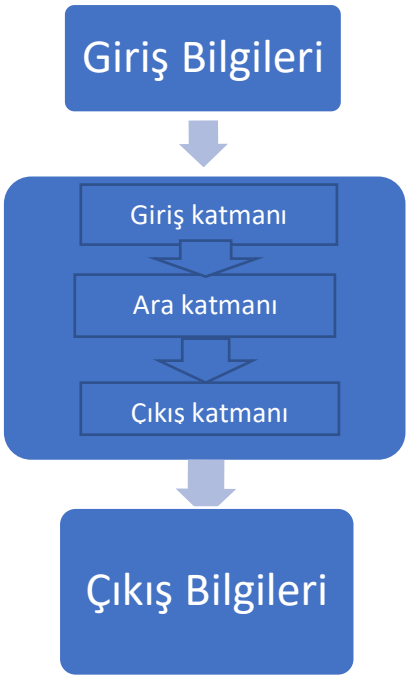
4.2.5 Çıkış değeri

Hücre çıkışı, transfer fonksiyonu tarafından belirlenen çıkış değeridir. Bu sonuç dış dünyaya veya başka bir hücreye gönderilir ya da tekrar olarak yeni bir hücreye aktarılarak bu süreç tekrarlanır. Hücre kendi sonucunu kendisi verir.

4.2.6 Yapay Sinir Ağının Yapısı

Yapay sinir hücreleri birleşerek YSA oluşturmaktadırlar. Genellikle hücreler üç katman halinde olarak ve her katman paralel birleşerek ağ yapısını oluşturmaktadır.

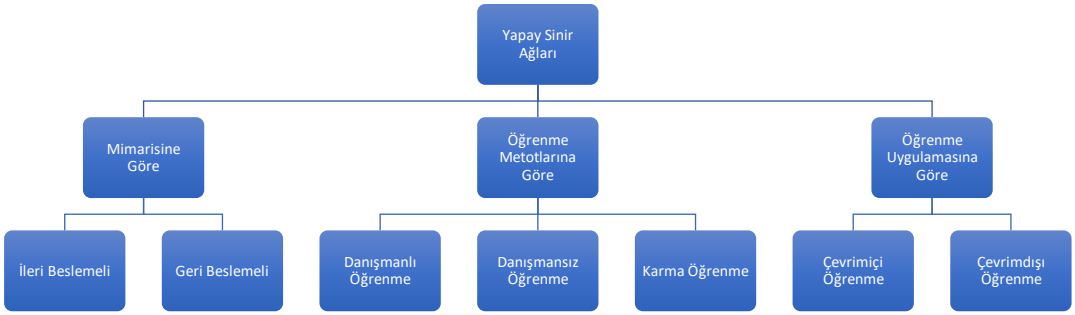
- **Giriş katmanı**: Ara katmanlar için dış dünyadan bilgi alarak taşınan yerdir.
- **Ara katmanlar**: Giriş katmanındaki bilgiler burada işlenir ve görüntülenir ve çıkış katmanına aktarılır. Yapay bir sinir ağının birkaç katmanı olabilir.
- **Çıktı katmanı**: Bu düzeyde, orta düzeydeki bilgiler işlenir ve sunularak girdi seti için üretilecek çıktı üretilir. Çıkış dış dünyaya aktarılır.



Şekil 4.3: Yapay sinir ağı katmanları

5. YAPAY SİNİR AĞLARININ SINIFLANDIRILMASI

Yapay sinir ağları birbirlerine işleyiş olarak benzer olsalar da herhangi bir işleyiş ve tasarım standardı bulunmamaktadır. Nöron ağırlıklarının düzenlenesi için yapılan hesaplama işlemi mimarisine, metotlarına ve uygulamasına göre YSA'lar üç ayrı kategoriye ayırabiliriz.



Şekil 5.1: Yapay Sinir Ağlarının Sınıflandırılması

5.1 Mimarisine Göre Yapay Sinir Ağları

Yapay sinir ağları nöronlarının birbirlerine bağlanış şekillerine göre ikiye ayrılmaktadır.

5.1.1 İleri beslemeli yapay sinir ağları

En tipik ileri beslemeli yapay sinir ağı modeli nöronların sıralı olarak bir araya getirilmesidir. Hücreler doğrudan ileri beslemeli YSA'larda katmanlanır ve bir katmandaki hücrelerin çıktıları bir sonraki katmana ağırlıklarla giriş olarak verilir. Bilgiler ortalama düzeyde işlenir ve bir çıktı ağı belirlenir. Bu yapı ileri beslemeli ağlar doğrusal olmayan bir statik işlev gerçekleştirir. İleri beslemeli 3 orta katmanında yeterli hücre olması koşuluyla çok katmanlı YSA, sürekli işlevi istenilen doğrulukla tahmin edilebilir (Fırat ve Güngör, 2004)

İleri beslemeli yapay sinir ağlarında, ağın genel davranışının doğrusal olmaması, giriş ve çıkış katmanları arasında gizli katmanlardaki nöronların doğrusal olmayan davranışı belirlemektedir. Giriş ve çıkış katmanlarındaki nöron sayısı, göz önüne

alınan problem tarafından belirlenir ancak gizli katmanlardaki nöron sayısını belirlemek için herhangi bir aritmetik yöntem kulanılmamaktadır. Yani, gizli katmandaki nöron sayısı deneme yanılma yoluyla belirlenmelidir (Efe ve Kaynak, 2000).

5.1.2 Geri beslemeli yapay sinir ağlar

Geri beslemeli YSA, çıktıdan elde edilen bilgiler ve bir önceki seviyenin orta seviyeleri ve bunları ara katmanlara veya girdilere yönlendirilen ve geri bildirim sağlayan yapay sinir ağlarıdır. Bu geri besleme sayesinde, bu ağlar ileri beslemeleri ağlarından daha dinamiktir. Geri beslemeli yapay sinir ağlarının bir döngüsü vardır. Bu döngü en az birini başlangıç hücresine yönlendirir. Çıkış hücresinin kendi fonksiyonu olmadığında bu tür yapılar zamanın fonksiyonlarını açıkça dikkate alınmasını sağlar. Hücre çıktısı da kendi başına bir fonksiyon olamaz, ancak bir değerin eksisi fonksiyon ola bilir. Geri beslemeli yapay sinir ağlarında, her bağlantıya bir gecikme atanır. Her gecikme çalışmaya başladığı ilk zamanının çok katmanlı bir türevidir ve döngü sınırındaki gecikmelerin toplamı sıfırdır. Zaman bozucu yinelemeli sinir ağları, hücresel fonksiyonların bir kombinasyonu ve bağlantılardaki gecikmeleri birleştiren doğrusal olmayan süreksiz zaman yinelemeli denklemler üzerinde çalışır (Dreyfus, 2005)

5.2 Öğrenme Metotlarına Göre Yapay Sinir Ağları

Yapay sinir ağlarında giriş verilerinden çıkış verilerinin üretilmesini sağlayan yöntem ağın öğrenmesidir. Bu öğrenme işlemi içinde birden fazla yöntem bulunmaktadır. Yapay sinir ağları öğrenme metotlarına göre üçe ayrılır.

5.2.1 Danışmanlı öğrenme yapay sinir ağları

Danışmanlı öğrenme YSA, sistemin olayı inceleyip öğrene bilmesi için bir danışmana yani öğreticiğe ihtiyacı var. Bu danışmanla sistem bir öğrenme olayıyla ilişkilendirilir ve örnekler bir giriş ve çıkış seti olarak verilir. Başka bir deyişle, hem girdi hem de bu girişlerin yerine oluşturulması gereken çıkışlar gösterilir. Böylece girdi ve çıktı olayları arasındaki ilişkiler incelenmiştir (Öztemel, 2006). Birçok uygulamada, ağa gerçek veriler uygulanmalıdır ve bu eğitim uzun sürebilir. Belirli bir sırayla girişler için bir sinir ağında istatistiksel doğruluk elde edildiğinde, öğrenme süreci tamamlanmış ve eğitim süreci bitmiş kabul edilir. Eğitim aşamasını tamamladıktan sonra, ağ başlangıçta, bulunan ağırlıkların değerinin sürekli olduğu varsayılarak sabit olarak alınır ve değiştirilmez. Bazı ağlar ağ yapımında çalışırken çok düşük oranda eğitime izin verir. Bu süreç ağların değişen koşullara uyum sağlamasına yardımcı olur.

5.2.2 Danışmansız öğrenme yapay sinir ağları

Sistemin, danışmansız öğrenme sürecinde danışmanlı öğrenmeden farklı olarak öğrenmesine yardımcı olan bir öğretici yoktur. Sistemde yalnızca girişler görüntülenir. Örnekler arasındaki parametreler kendi kendine öğrenmesi gerektirir. Bu genel olarak sınıflandırma problemleri için kullanılan yöntemdir. Ancak sistem eğitimi bitirdikten sonra sonuç ne anlama geldiğini gösteren etiketleme kullanıcı tarafından yapılmalıdır (Öztemel, 2006).

Danışman olmadan eğitim yaparken, ağ istenen harici verilerle değil, girilen bilgilerle çalışır. Bu tür eğitimde, gizli sinirler dışarıdan yardım almadan kendi kendini örgütlemek için kullanılır bir yol bulmalılar. Bu yaklaşımla, belirli bir girdi için önceden bilinebilir ağ için performansını ölçebilen bir çıkış sinyali sağlanmaz, yani ağ üzerinden öğrenir. Danışmansız öğrenmeye Grossberg öğretim kuralı Kohonen'in kendi kendini organize eden harita ağı, Hebbian öğrenme kuralı buna bir örnektir. Kohonen "in kendi kendini düzenleyen bir harita ağında, durum veya ölçümlerde güncelleme için yarışıyorlar. En yüksek çıktı işlenen sinir kazananı belirler ve komşuların bağlantı boyutlarını güncellemelerine izin verir ve bunu güncellemeler ayı şekilde devam ediyor (Elmas, 2007).

5.2.3 Karma öğrenme yapay sinir ağları

Destekleyici öğrenme yapay sinir ağlarında öğrenme yaklaşımı sırasında ağın her iterasyonu sonucunda elde ettiği sonucun iyi veya kötü olup olmadığına dair bilgi vermektedir. Bu bilgilere göre ağ kendini yeniden düzenler. Ağ bu sayede herhangi bir girdi dizisi ile hem öğrenmektedir hem de sonuç çıkararak işlemine devam etmektedir. Kısmen danışmanlı olsun veya kısmen danışmansız olarak öğrenen ağlar radyal yapay sinir ağları (RBN) ve olasılık tabanlı ağlar (PBN) bunlara örnektir gösterilebilir (Öztemel, 2006)

5.3 Öğrenme Uygulamasına Göre Yapay Sinir Ağları

Yapay sinir ağları öğrenme uygulamasına göre ikiye ayrılmaktadır.

5.3.1 Çevrimiçi öğrenme yapay sinir ağları

Çevrimiçi öğrenme, sistemi yapay sinir ağlarında kullanmadan önce eğitilir. Eğitimi tamamladıktan sonra, ağ istendiği gibi kullanılabilir. Yapay bir sinir ağının eğitimini tamamladıktan sonra, istenildiği şekilde kullanılabilir ve aynı anda ağda ağırlık değişiklikleri meydana gelmez.

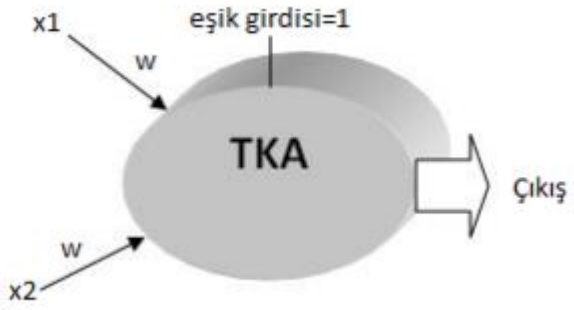
5.3.2 Çevrimdışı öğrenme yapay sinir ağı

Çevrimdışı öğrenme, yapay sinir ağında geliştirilir ve çalışma süresi boyunca ağın incelenmesini ön görerek yapılmıştır. Bu yapay sinir ağındaki eğitim sürecinin her bir çalışmada elde edilen sonucun onaylanması ile tamamlanmasından sonra, bu veriler ve sonuç, ağırlıktaki değişikliği etkileyerek sürece devam eder.

6. YAPAY SİNİR AĞLARINDA KULLANILAN MODELLER

6.1 Tek Katmanlı Algılayıcılar (TKA)

TKA yalnızca giriş ve çıkış katmanlarından oluşmaktadır. Her ağın bir veya daha fazla girişi ve çıkışı vardır. Toplam girdi çıktı birimleri, birimlerle ilişkilendirilirler. Her bağlantının bir ağırlığı ve ağdaki kendi işlemi vardır. Ayrıca boş öğeleri ve ağ çıkışını engelleyen bir eşik de vardır (Öztemel, 2006).



Şekil 6.1: İki girişi ve bir çıkışı olan TKA modeli

Ağ çıkışı, ağırlıklı giriş değerlerinin eşik değerleri ekleme sonucunda bulunur (Şekil 6.1). Bu giriş etkinleştirme fonksiyonu ağ üzerinden iletilir ve çıkışı elde edilmektedir. Bu fonksiyon aşağıdaki gibi ifade edilir.

$$Ç = f[\sum_i^n \omega_i \, x_i + \theta] \quad (6.1)$$

Tek katmanlı yapay sinir ağlarında 6.1 deki çıkış yani Ç fonksiyonumuz doğrusal fonksiyondur. Bu nedenle, ağa gönderilen örnekler iki sınıfa paylaştırılarak iki sınıfa ayrılır. Onu birbirlerinden doğru şekilde ayırarak bulmaya çalışıyorlar. Böylece, eşik değer fonksiyonu kullanılıyor. Ağ çıkışı 1 veya -1 değerini alır ve bu değerler yapay sinir ağındaki sınıfı temsil eder.

$$f(x) = \begin{cases} 1 \; eğer \; Ç < 0 \\ 1, aksitaktirde \end{cases} \quad (6.2)$$

6.1.1 Perseptron

1958 yılında Rosenblatt sınıflandırması bir bakış açısıyla yani şekil sınıflandırma amacıyla geliştirilmiştir (Rosenblatt, 1958). Bu algılayıcı basit bir modelidir ve bir sinir hücresinin birden fazla girişi kabul ederek bir çıktı ürettiği ilkesine dayanır. Ağ çıkışı, bir veya sıfırdan oluşan mantıksal bir değerdir (Öztemel, 2006).

Bu sınıflandırma modeli eğitilebilen bir yapay sinir hücresinden oluşur. Eğitilebilir dedikte, ağırlıkların değiştirilebilir olduğu anlamına gelir. Kayıtlar hücrede ve her kayıtta görüntülenir sete karşılık gelen çıkış seti ağda da görüntülenir ve hesaplanan çıkış değeri ağ eğitim kuralına göre görüntülenir. Çıktı olması gerektiği gibi değilse, ağırlık ve eşik değerleri değiştirilir. Bu değişikliğin nasıl yapılacağı, kullanılan eğitim kuralına bağlıdır. Girişlere karşılık gelen çıkış değerleri bir veya sıfırdan oluşur (Öztemel, 2006; Kasabov 1998). Algılayıcı eğitim kuralı aşağıdaki gibidir:

İlk olarak, ağ girişleri seti ve karşılık gelen istenilen çıkış görüntülenir. Giriş değeri $x_{1,}x_{2,}\dots,x_n$ gibi çeşitli değerler olabilir. Çıkış değeri 1 yada 0 değerlerinden birini alır. Perseptron algılayıcısına gelen net girişler aşağıdaki gibidir.

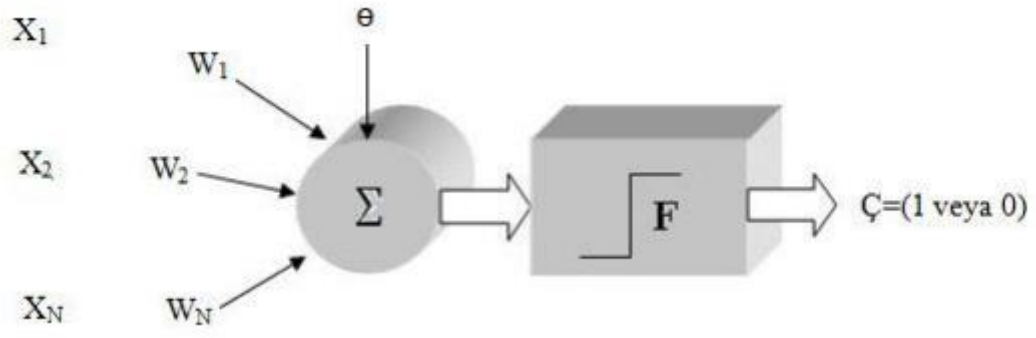
$$NET = \sum_i^n \omega_i x_i \qquad (6.3)$$

Perseptronun çıktısı hesaplanarak Net giriş değerinin eşik değerinden büyük yada küçük olmasına göre çıkış değeri 1 veya 0 değerlerinden birini alır.

$$Ç = \begin{cases} 1\ eğer NET > 0 \\ 0\ eğer NET \leq 0 \end{cases} \qquad (6.4)$$

6.1.2 ADALİNE model

Bu model 1959'da geliştirildi ve uyarlanabilir lineer elemente sahip (Adaptive Linear Element), bir ağın kısaltılmış biçimidir. Genellikle bir işlem öğesinden oluşan bir ağdır (Widrow ve Hoff, 1960). Bu ağ modeli en küçük araçların ortalama karekök (LMS, least mean square) yöntemidir. Öğrenme kuralına delta kuralı da denir. Bu kurala göre, çıkışın beklenen çıktı değerine göre ağ hatasını en aza indirmek için ağırlıklarını değiştiriyoruz (Öztemel, 2006).



Şekil 6.2: ADALİNE Modeli

Burada $x_1, x_2, \dots, x_n$ giriş değerleri, $\omega_1, \omega_2, \dots, \omega_n$ giriş değerlerine karşılık gelen ağırlıklar, θ çıkış değerinin sıfırdan farklı olması için kullanılan eşik değeridir. ADALİNE ağının öğrenme kuralı yapay sinir ağlarında genel öğrenme prensibine göre çalışır. Giriş parametrelerine göre çıktılar hesaplanır ve ağırlıklar çıkış parametrelerine göre değiştirilir. Burada net giriş aşağıdaki gibi hesaplanmaktadır:

$$NET = \sum_{i=1}^n \omega_i x_i + \theta \qquad (6.5)$$

Net değer sıfır ve sıfırdan büyükse çıkış değeri 1, aksi takdirde -1 değerini alır ve beklenen değer ile hatanın oluşturduğu değer arasındaki fark hatayı verir. Amaç bu hatayı en aza indirmektir ve sebepten ağa her seferinde farklı örnekler gösterilir ve ağırlıklar hatayı azaltacak şekilde ayarlanır. Zaman içinde hata olması gereken minimum değere düşer.

6.1.3 MADELİNE model

MADALINE ağları, birden çok ADALINE ağlarının bir araya gelmesidir ve genellikle iki tabakadan oluşurlar. Her katmanın farklı sayıda ADALINE birimi vardır. Ağ çıkışı 1 ve -1 değerleri ile görüntülenir. Her biri bir sınıfı temsil eder (Öztemel, 2006) ve MADALINE ağlarındaki öğrenme kuralı ADALINE ağına benzer. Son bölümde sadece VE ve VEYA sonlandırıcılar vardır. AND sonlandırıcısı durumunda MADALINE ağının çıkışı 1 olur, çünkü tüm ADALINE blokları 1 verir, aksi takdirde -1 değerini alır. Bir OR sonlandırıcısı varsa ADALINE bloklarından birinin 1 değeri vermesi yeterlidir, bu nedenle MADALINE ağının çıkışı 1'dir.

6.2 Çok Katmanlı Algılayıcılar (ÇKA)

Yapay sinir ağlarında girişler ve çıkışlar arasında doğrusal olmayan bir ilişki varsa, eğitim için çok katmanlı bir algılayıcı modeli gereklidir. XOR konusu bu modelin geliştirilmesinde büyük rol oynamıştır.

Çizelge 6.1: XOR problemi

Giriş 1	Giriş 2	Çıkış
0	0	0
0	1	1
1	0	1
1	1	0

1986'da Rumelhart ve arkadaşları doğrusal olmayan XOR probleminin çözümü için geliştirilen hata yayılım modeli veya geri yayılım ağı kullanarak çözümünü bulmuşlar. ÇKA model sayesinde, yapay sinir ağları önemli bir gelişme gösterdi ve bu model bugün tüm mühendislik sorunlarının çözümü haline geldi. Bu, özellikle sınıflandırma, tanıma ve genelleme gerektiren görevler için önemli bir çözüm yöntemi oldu. Çok katmanlı algılayıcı modelleri delta öğrenme kuralını kullanır. Ana amaç, ağın beklenen çıkış sinyali ile alınan çıkış sinyali arasındaki hatayı en aza indirmektir. Bu ise, hatayı tekrar ağa yayarak gerçekleşir (Öztemel, 2006).

6.2.1 Çok katmanlı algılayıcıların öğrenme kuralı

MCA ağları, bir danışma eğitim stratejisine uygun olarak çalışır. Eğitim sırasında, bu ağlar bu girişlerle eşleşmesi gereken giriş ve çıkışları gösterir. Çok katmanlı algılayıcı ağlarının öğrenme kuralı, delta öğrenme kuralının genel bir sürümüdür. Eğitimin gerçekleşmesi için bir eğitim seti ve örneklerden oluşan bir set gereklidir. Bu setler, girişlere karşılık gelen giriş ve çıkışları gösterir (Öztemel, 2006).

Genelleştirilmiş delta kuralı iki aşamadan oluşur: ağın çıkış gücünü hesaplama aşaması olan ileriye doğru hesaplama ve geriğe doğru hesaplama yani ağırlık değiştirme aşaması olarak ikiğe ayrılır.

İleriye doğru hesaplama aşaması, örneği giriş katmanı tarafından ayarlanan eğitimde göstererek başlar. Giriş katmanında herhangi bir işlem yapılmaz. Gelen kayıtlar herhangi bir değişiklik yapılmadan ara katmana gönderilir. Yani, k giriş katmanında işlem elemanının çıktısı aşağıdaki gibidir.

$$Ç_k^i = G_k \qquad (6.6)$$

Ara katmana gelen net girişde aşağıda gösterildiği gibidir.

$$NET_j^a = \sum_{k=1}^{n} A_{kj} Ç_k^i \qquad (6.7)$$

Genelde transfer fonksiyonu olarak türevi alına bilecek bir fonksiyon olarak sigmoid fonksiyonu tercih edilir ve sigmoid fonksiyonu kullanıldığı halinde çıkış değeri aşağıdaki gibi hesaplanır.

$$Ç_j^a = \frac{1}{1+e^{-(NET_j^a + \beta_j^a)}} \qquad (6.8)$$

β_j ara katmanındaki j bir öğeyle ilişkili eşik değeri, ağırlığını gösterir. Bu eşik ünitesinin çıkışı sabittir ve 1'e eşittir. Ağırlık değerinin girilmesinin nedeni sigmoid fonksiyonunun yönünü belirlemektir. Ara ve çıkış katmanlarındaki hücreler için kendilerine gelen NET girişinin hesaplanması aynı şekilde devam eder. Çıktı katmanındaki çıktı değerleri alındığında, gelişmiş ağ hesaplama işlemi tamamlanır ve geriye doğru hesaplama aşamasında ağ tarafından üretilen çıktı verileri beklenen çıktı verileriyle $(B_1, B_2, \dots, B_N)$ karşılaştırılır. Fark bir hata olarak kabul edilir ve bu hata azaltılmalıdır. Bu nedenle, hata ağırlık değerlerine uzanır ve sonraki iterasyonlarda bu hata azaltılır.

6.2.2 Çok katmanlı algılayıcıların performansının ölçülmesi

ÇKA performansı yapay sinir ağlarının öğrenme yeteneğinin ölçülmesidir. Ağın kendisine gösterilen tüm örneklere doğru cevap vermesi, performansının iyi olduğu anlamına gelmez. Bu nedenle, eğitim ağlarının daha önce görmedikleri örneklere kıyasla beklenen sonuçları gösterip göstermediğini ölçülmülidir.

Bunu yapmak için hem eğitimde hem de test sırasında kullanılacak örnekler ağın eğitildiği problem üzerinden seçilir. Ağ üzerinde eğitim sırasında yalnızca eğitim setinden örnekler görüntülenir. Ağdaki eğitimi tamamladıktan sonra, örnekler ağda hiç görmediği bir test setinde gösterilir (Öztemel, 2006). Ağ performansı, görmediği örnekler için ürettiği cevaplarla ölçülür,

$$P = \frac{D}{T} \times 100 \qquad (6.9)$$

ile hesaplanır. Burada test setinden cevabın doğru verildiği örnek sayısını D, test setinde bulunan toplam örnek sayısını T ve performans katsayısını P gösterir. Performans seviyesi istenen seviyede veya kabul edilebilir bir değerde değilse, ağ eğitim setindeki tüm örnekler doğru verilmiş olsa bile iyi çalıştığı söylenemez. O zaman biraz daha eğitime devam etmemiz gerekebilir. Eğitim yinelemeleri artarsa ve üretkenlik hala iyileşmezse, örneklerin problem alanını iyi yansıtmadığı veya ağ parametreleri ile topolojinin yanlış seçildiği açıktır (Öztemel, 2006).

6.3 LVQ (Linear Vector Quantization) Modeli

LVQ ağı 1984 yılında Kohonen tarafından geliştirilen bir ağdır. Bir n-boyutlu vektörün bir vektör kümesi üzerine eşlenmesi, temel bir çalışma prensibidir. Bu ağı kullanarak, belirli sayıda vektör içeren bir vektör görüntülemek üzere tasarlanmıştır.

LVQ ağları genellikle sınıflandırma problemlerini çözmek için kullanılır. Çıkışlardan yalnızca biri 1 değerini, diğeri 0'ı alır. Çıkış 1 ise, bu, giriş parametresine karşılık gelen çıktının temsil ettiği sınıfa ait olduğu anlamına gelir. Eğitim sırasında, giriş verileri en yakın komşunun kuralına göre sınıflandırılır. Giriş vektörleri ve referans vektörler arasındaki en küçük mesafe aranır ve girdi vektörünün en kısa mesafede bulunan bir vektör grubuna ait olduğu varsayılır.

6.3.1 LVQ ağının öğrenme kuralı

LVQ ağının eğitim kuralı, Kohonen katmanındaki süreç öğelerinin birbiriyle rekabet etmesine dayanan ilkeye dayanır ve buna Kohonen eğitim kuralı denir (Öztemel, 2006). Rekabet, giriş vektörü ile ağırlık vektörü (referans) arasındaki Öklid mesafesinin hesaplanmasına dayanır. En küçük işlem elemanı rekabeti kazanır. i. işlem öğesine olan mesafe aşağıdaki gibi hesaplanmaktadır.

$$d_i = \|A_i - X\| = \sqrt{\sum_j (A_{ij} - x_j)^2} \qquad (6.10)$$

A referans vektörünü, x girdi vektörünü, d aradaki mesafedir. A_{ij} ağırlık vektörünü, x_j girdi vektörünü ve j değerlerini ifade etmektedir.

Bu mesafeleri ayrı ayrı hesapladıktan sonra, proses elemanının referans vektörü, giriş vektörüne en yakın rekabeti kazanır. Kazanan proses elemanı doğru sınıfın bir üyesiyse, ağırlıklar giriş vektörüne biraz daha yakındır. Bu örnek, ağa geri görüntülendiğinde aynı işlem öğesini kazanmak için yapılır ve ağırlıkların değişmesi aşağıdaki gibi hesaplanır.

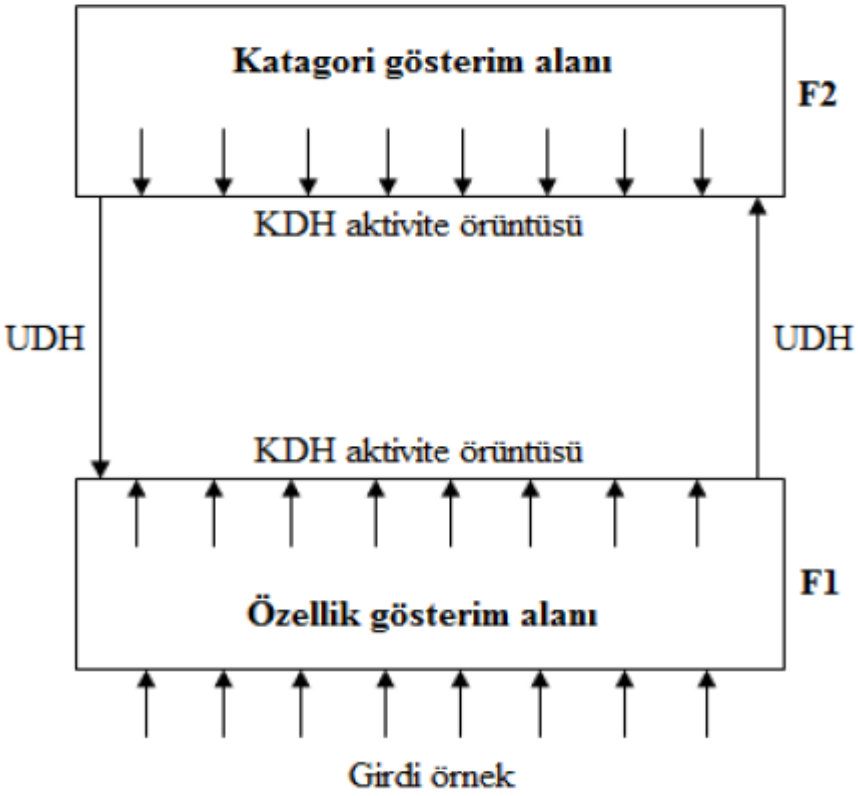
$$A_y = A_e + \lambda(x - A_e) \qquad (6.11)$$

λ zaman içerisinde 0 değerini alacak şekilde monoton öğrenme katsayısı olarak değiştirilir. Sonrasında LVQ modelinin uygulanması sonucu elde edilen çözümün iyileştirilmesi amaçlayarak Kohonen LVQ 2 ağını geliştirmiştir. Bu geliştirilmiş LVQ 2 ağında aynı anda iki ağırlık vektörünün ağırlıklarının değişmesi önerilmiştir.

6.4 ART (Adaptif Rezonans Teori) Modeli

ART ağları, Grossberg'in biyolojik beynin fonksiyonları üzerindeki çalışmaları sonucunda 1976'da ortaya çıktı (Grossberg, 1976). ART ağları, danışmansız eğitim temelinde çalışan ağlardır ve ART ağlarının ve danışmansız eğitim ile çalışan ağların geliştirilmesinde temel oluştururlar (Öztemel, 2006). ART ağları sınıflandırma amacı için tasarlanmışdır. LVQ ağları da sınıflandırma için kullanılmasına rağmen, ART ağları ve bu ağlar arasındaki fark, ağ üzerinde yapılması gereken herhangi bir sınıflandırma bilgisi olmadan ağın kendi başına çalışmasıdır. Ağ bunu doğru bilgileri tanımlayıp depolayarak yapar. Bilgiler kısa dönemli hafıza (KDH) ve Uzun dönemli hafıza (UDH) şeklinde depolanır. Kısa dönemli hafıza (KDH) bilgilerin geçici olarak yani zaman içerisinde yok olup yerine başka bilgilerin saklandığı hafıza türüdür. Uzun dönemli hafıza ise bilgilerin kolay unutulmadığı ve silinmesi için çok uzun zamana ihtiyaç gereken hafızadır.

Adaptif rezonans teorisi ağları tipik olarak iki katmandan oluşur. Bu katmanlar, F1 katmanı giriş özelliklerini ve F2 katmanı kategorileri (ayrılmış sınıflar) gösterir. Bu iki katman UDH ile birbirine bağlanır. Giriş bilgisi F1 katmanından alınır ve sınıflandırma F2 katmanında yapılır (Öztemel, 2006). ART ağlarında girdiler doğrudan sınıflandırılmaz yani ilk olarak F1 katmanının aktivasyonu, girdilerin özellikleri incelenerek belirlenir. UDH'deki bağlantı değerlerinden gelen bilgiler kategorilere ayrılır ve F2'ye gönderilir ve F2 seviyesindeki sınıflandırma ve F1 seviyesindeki sınıflandırma birbiriyle karşılaştırılır ve belirtilen sınıf eşleşirse örnek bu kategoride görüntülenir. Aksi takdirde, yeni bir sınıf oluşturulur veya kayıt sınıflandırılmaz.



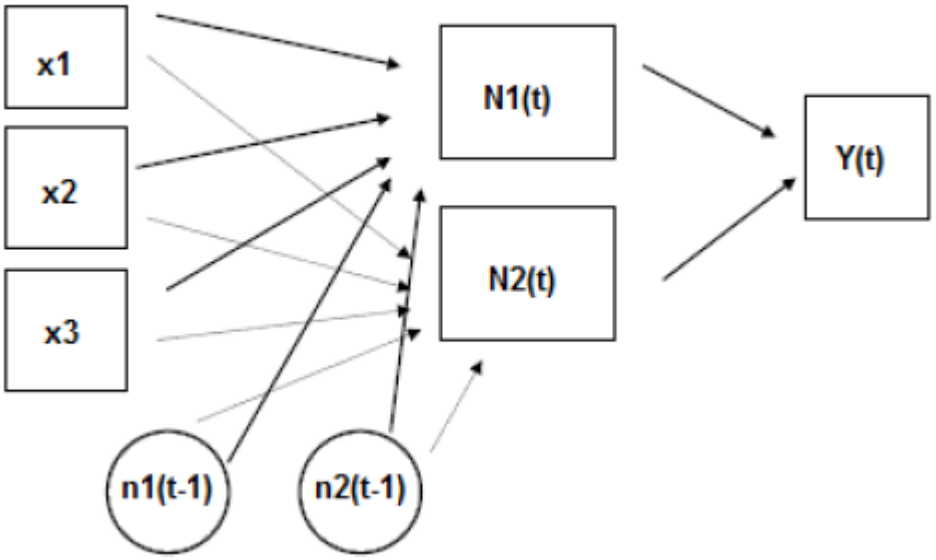
Şekil 6.3: Adaptif rezonans teori ağının yapısı

Adaptif rezonans teori ağları F1'den F2'ye (aşağıdan yukarıya) ve F1'den F2'ye (yukarıdan aşağıya) iki yönlü bilgi işler.

6.5 Elman Ağı

Elman ağı, dört tür işlem elemanına sahip yapay bir sinir ağıdır: giriş elemanları, ara katman elemanlar, çıkış elemanları ve içerik elemanları, çok katmanlı algılayıcı ağının öğrenme kuralına uygun prosesdir. Giriş katmanı, diğer ağlarda olduğu gibi dış dünyadan bilgi alır ve başka bir işlem gerçekleştirmez. Çıkış biriminin bilgi işleme işlevleri doğrusaldır. Ara katmanlar doğrusal veya doğrusal olmayan transfer fonksiyonlarına sahip olabilir. İçerik öğeleri, ara katman öğelerinin önceki etkinlik değerlerini hatırlatmak için kullanılır. Bu elemanlar adım gecikmesini içerir. Bir önceki yinelemede bir sonraki yinelemeye girdi olarak a ktivasyon değerleri içerirler (Öztemel, 2006).

Elman ağında herhangi bir zamanda, yani giriş verileri ve ara katmanda aktivite değerleri ağa giriş parametreleri olarak verilir. Ağ girişleri tanımlandıktan sonra, ağ doğrudan bağlantılı çok katmanlı bir algılayıcıya dönüşür. Bu girişler kullanılarak doğrudan ağın çıkışları belirlenir. Bu doğrudan hesaplamadan sonra, ağın ara katmanlarının aktivasyon değerleri, girdi olarak içerik elemanlarına geri gönderilir ve tekrarlamada kullanılması için orada saklanılır.



Şekil 6.4: Elman ağının şematik gösterimi

6.6 Hopfield Ağı

Hopfield ağı tek katmanlı ve yeniden kullanılabilir bir ağdır. İşlemin tüm öğeleri hem giriş hem de çıkış öğeleridir. Bağlantı değerlerini Hopfield ağı enerji fonksiyonu olarak saklar. Hopfield ağı kesikli ve sürekli olan iki türü vardır. Hopfield kesikli ağları ilişkilendirilebilir bellek olarak kullanılır. Hopfield sürekli ağları temel olarak karmaşık optimizasyon problemleri için kullanılır (Öztemel, 2006). Kesikli Hopfield ağı hücresinin iki ağı var ve hücre on (+1) yada hücre off (-1) olabilir. Kesikli Hopfield ağlarında signum, Sürekli Hopfield ağlarının sigmoid fonksiyonu kullanılır. Yani Sürekli Hopfeld ağları 0 ve 1 arasında sürekli çıktı değerleri alıyor (Öztemel, 2006)

7. YAPAY SİNİR AĞLARINDA KULLANILAN ALGORITMALAR

Meme kanserinde klinik veri seti temelinde doğru tahmin yapılabilmesi için Yapay Sinir Ağı modelinin doğru optimizasyon algoritması ile uygulanması ve parametre aralıklarının doğru belirlenmesi kritik önem taşır. Bu nedenle çalışmamızda aşağıdaki optimizasyon algoritmalarının nasıl sonuç verdiği araştırıldı.

7.1 Stokastik gradyan inişi (Stochastic Gradient Descent SGD)

Stokastik gradyan inişi (SGD) derin öğrenmede, nesnesel işlev genellikle eğitim veri setindeki her örnek için kayıp işlevlerinin ortalama değeridir. $F_i(x)$ n veri, indeks i ve parametre vektörü x ile eğitim verisi örneğinin bir kayıp fonksiyonu olduğunu varsayıyoruz, o zaman objektif fonksiyonumuz var.

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i\,(x) \qquad (7.1)$$

x'deki objektif fonksiyonun gradyanı şu şekilde hesaplanır:

$$\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) \qquad (7.2)$$

Degrade iniş kullanılırsa, bağımsız değişkenin her yinelemesi için hesaplama maliyeti, n ile doğrusal olarak büyüyen O (n) 'dir. Bu nedenle, modelin eğitim verilerinin örneği büyük olduğunda, her bir yineleme için degrade iniş maliyeti çok yüksek olacaktır.

Stokastik gradyan inişi (SGD) her bir yinelemenin hesaplama maliyetini azaltır. Stokastik gradyan inişin her yinelemesinde, rastgele veri örnekleri için i∈ {1, ..., n} indeksini eşit olarak seçeriz ve x'i güncellemek için $\nabla f_i(x)$ gradyanını hesaplarız:

$$x \leftarrow x - \eta \nabla f_i(x) \qquad (7.3)$$

Burada η öğrenme oranıdır. Her bir yineleme için hesaplama maliyetinin O (n) gradyan inişinden sabit O (1) 'e düştüğünü görebiliriz. Stokastik gradyan $\nabla f_i(x)$ 'nin ∇f (x) gradyanının tarafsız bir tahmini olduğu unutulmamalıdır (Ruder 2017).

$$E_i \nabla f_i(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla \mathrm{f(x)} \qquad (7.4)$$

Bu, ortalama olarak, stokastik gradyanın gradyanı iyi bir tahmini olduğu anlamına gelir.

7.2 Adagrad

s_t değişkenini, geçmiş gradyan varyansını aşağıdaki gibi biriktirmek için kullanırız.

$$\mathrm{g_t} = \partial_{\omega} \mathrm{l}\,(\mathrm{y_t}, \mathrm{f(x_t}, \omega)),$$

$$s_t = s_{t-1} + g_t^2,$$

$$\omega_t = \omega_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \cdot g_{t.} \qquad (7.5)$$

η öğrenme oranıdır ve ϵ, 0'a bölmememizi sağlayan bir katkı sabitidir. Son olarak, s_0 = 0 değerini başlatırız.

Momentumda olduğu gibi, her bir koordinat için bireysel öğrenme hızını dikkate almak için bu durumda yardımcı değişkeni izlememiz gerekir. Bu, Adagrad'ın maliyetini SGD'ye kıyasla önemli ölçüde artırmaz, çünkü ana maliyet genellikle l (yt, f (xt, w)) ve türevinin hesaplanmasından oluşur. Momentumda olduğu gibi, yardımcı bir değişkeni izlememiz gerekir, bu durumda koordinat başına bireysel bir öğrenme oranına izin vermek için Adagrad'ın $l\ (y_t, f(x_t, \omega))$, ve türevini hesaplaması olduğundan, Adagrad'ın SGD'ye göre maliyetini önemli ölçüde artırmaz.

7.3 RMSprop

RMSprop algoritması, hız planlamasını koordinat uyarlamalı öğrenme hızlarından ayırmaya izin veren basit bir düzeltme olarak kullanılmaktadır. Sorun, Adagrad'ın $g_t = s_t - 1 + g_t^2$ durum vektöründe g_t gradyanının karelerini biriktirmesidir. Sonuç olarak, algoritma yakınsadığı için, s_t normalleşme eksikliği nedeniyle, esasen doğrusal olarak, kısıtlamalar olmadan büyümeye devam eder(Ruder 2017). Bu sorunu çözmenin bir yolu s_t / t kullanmak olacaktır. Makul g_t dağıtımları için, bu yakınsama yapacaktır. Ne yazık ki, prosedür değerlerin tam yörüngesini hatırladığından, limitin davranışının önemli hale gelmesi çok uzun zaman alabilir. Bir alternatif, ortalama sızdıran değerini bazı parametreler için $s_t \leftarrow \gamma s_{t-1} + (1 - \gamma) g_t^2\ \gamma > 0$ ve diğer tüm parçaları değişmeden tutmak RMSprop verir.

$$s_t \leftarrow \gamma s_{t-1} + (1-\gamma) g_t^2$$

$$x_t \leftarrow x_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_t \qquad (7.6)$$

ϵ> 0 sabiti genellikle sıfır veya çok büyük adım boyutlarına bölünmememizi sağlamak için 10^{-6} olarak ayarlanır. Bu genişleme göz önüne alındığında, öğrenme hızını η koordinat başına uygulanan ölçeklemeden bağımsız olarak kontrol etmektedir.

7.4 Adadelta

Adadelta, AdaGrad'ın başka bir sürümüdür. Aralarındaki fark, öğrenme hızının koordinatlara uyarlanma miktarını azaltmasıdır. Ayrıca, geleneksel olarak bir öğrenme oranına sahip olmadığından değişim miktarını gelecekteki değişim için kalibrasyon olarak kullanır. Özetle, Adadelta iki durum değişkeni kullanır: s_t gradyanın ikinci momentinin ortalama sızıntısını depolamak için ve Δx_t ikinci değişiklik anının ortalama sızıntısını modelin kendisinde saklamak için.

$$s_t = p s_{t-1} + (1-p) g_t^2,$$

$$\boldsymbol{g'_t} = \sqrt{\frac{\Delta x_{t-1} + \epsilon}{s_t + \epsilon}} \odot \boldsymbol{g_t},$$

$$\boldsymbol{x_t} = \boldsymbol{x_{t-1}} - \boldsymbol{g'_t},$$

$$\Delta x_t = p \Delta x_{t-1} + (1-p) x_t^2. \qquad (7.7)$$

Bir öncekinden farkı, değişim oranının ortalama karesi ile gradyan ortalama ikinci momenti arasındaki ilişki alınarak hesaplanan değiştirilmiş bir gradyan g'_t ile güncellemeler gerçekleştirmemizdir. g'_t kullanımı yalnızca tanımlama kolaylığı içindir. Uygulamada, bu algoritmayı g'_t için ek geçici alan kullanmak zorunda kalmadan uygulayabiliriz. Daha önce olduğu gibi, η, önemsiz olmayan sayısal sonuçlar sağlayan, yani sıfır adım büyüklüğünden veya sonsuz varyanstan kaçınan bir parametredir. Genel olarak, bunu $\eta = 10^{-5}$ olarak ayarlanır.

7.5 Adam

Adam'ın temel bileşenlerinden biri, hem momentumun hem de gradyanın ikinci momentinin bir tahminini elde etmek için üstel sızdıran ortalamalar kullanmasıdır. Yani, durum değişkenlerini kullanır

$$v_t \leftarrow \beta_1 v_{t-1} + (1-\beta_2) g_t^2$$
$$s_t \leftarrow \beta_2 s_{t-1} + (1-\beta_2) g_t^2 \qquad (7.8)$$

Burada β1 ve β2 negatif olmayan ağırlık parametreleridir. Onlar için normal seçim: $\beta_1 = 0.9$ ve $\beta_2 = 0.999$. Yani, varyans tahmini momentum teriminden çok daha yavaş hareket eder. Eğer $v_0 = s_0 = 0$ değerini başlatırsak, başlangıçta daha düşük değerlere önemli bir önyargıya sahip oluruz. Bu, terimleri yeniden normalleştirmek için $\sum_{i=0}^{t} \beta^i = \frac{1-\beta^t}{1-\beta}$ kullanılarak çözülebilir. Buna göre, normalize edilmiş durum değişkenleri

$$\hat{v}_t = \frac{v_t}{1-\beta_1^t} \text{ ve } \hat{s}_t = \frac{s_t}{1-\beta_2^t} \qquad (7.9)$$

Şimdi güncelleme denklemlerini yazabiliriz ve lk olarak, gradyanı elde etmek için RMSProp'a çok benzer bir şekilde yeniden ölçeklendirilir

$$g'_t = \frac{\eta \widehat{v_t}}{\sqrt{\widehat{s_t}+\epsilon}}. \qquad (7.10)$$

RMSprop'tan farklı olarak, güncellememiz gradyanın kendisi yerine $\widehat{v_t}$ momentumunu kullanır. Dahası, yeniden ölçekleme $\frac{1}{\sqrt{\widehat{s_t}+\epsilon}}$ yerine $\frac{1}{\sqrt{\widehat{s_t}+\epsilon}}$ kullanarak gerçekleştiği için küçük bir kozmetik farkı vardır. Önceki pratikte pratikte biraz daha iyi çalışıyor, bu nedenle RMSProp'dan sapma genellikle sayısal kararlılık ve sadakat arasında iyi bir denge için $\epsilon = 10^{-6}$ seçeriz. Güncellemeleri hesaplamak için tüm parçalarımız var, bu biraz antiklimaktiktir ve formun basit bir güncellemesine sahibiz. Sonra, Adadelta ve RMSprop'ta gördüğümüz gibi parametreleri güncellemek için kullanırlar. Adam güncelleme kuralını verir:

$$\omega_{t+1} = \omega_t - \frac{\eta \widehat{v_t}}{\sqrt{\widehat{s_t}+\epsilon}} \qquad (7.11)$$

7.6 AdaMax

Adam güncelleme kuralındaki s_t faktörü gradyanı l_2 normuna göre ters orantılı olarak ölçeklendirir ve geçmiş gradyanları (v_{t-1}terimi üzerinden) ve geçerli gradyan $|g_t|'$dir(Ruder 2017).

$$s_t \leftarrow \beta_1 s_{t-1} + (1-\beta_2)|g_t|^2 \qquad (7.12)$$

Bunu Adam güncellemesine uyarlayarak AdaMax güncelleme kuralını elde ederiz.

$$\omega_{t+1} = \omega_t - \frac{\eta}{s_t}\widehat{v_t} \qquad (7.13)$$

Burada

$$S_t = \beta_2^{\infty} s_{t-1} + (1-\beta_2^{\infty})|g_t|^{\infty} = \max(\beta_2 s_{t-1,}|g_t|) \qquad (7.14)$$

7.7 Nadam

Nadam (Nesterov hızlandırmalı Uyarlanabilir Moment Tahmini) Adam ve NAG'yi birleştirir. İçinde Nesterov hızlandırılmış gradyan (NAG) Adam'a dahil etmek için, momentum terimini değiştirmemiz gerekiyor(Ruder 2017).

$$g_t = \nabla_{\omega_t} J(\omega_t)$$

$$m_t = \gamma m_{t-1} + \eta g_t$$

$$\omega_{t+1} = \omega_t - m_t \qquad (7.15)$$

Burada objektif fonksiyonumuz J, momentum bozulma terimi γ ve η adım boyutudur. Yukarıdaki üçüncü denklemin genişlenmesi şunu verir:

$$\omega_{t+1} = \omega_t - (\gamma m_{t-1} + \eta g_t) \qquad (7.16)$$

Momentum vektörü ve mevcut gradyan yönünde bir adım içerdiğini bir kez daha göstermektedir. Nesterov hızlandırılmış gradyanı güncelleyerek gradyan yönünde daha kesin bir adım atmamızı sağlar. Bu nedenle, yalnızca g_t de NAG değerini bulmak için geçerli parametreleri güncellemek için ileriye yönelik momentum vektörünü doğrudan uygulayarak:

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

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$\omega_{t+1} = \omega_t - \frac{\eta \widehat{m_t}}{\sqrt{\widehat{s_t}}+\epsilon} \qquad (7.17)$$

$\widehat{m_{t-1}}$ momentum vektörü akımın önyargı düzelmeli tahmini ile $\widehat{m_t}$ momentum vektörü Nadam güncelleme kuralını verir.

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{\widehat{s_t}}+\epsilon}\left(\beta_1 \widehat{m_t} + \frac{(1-\beta_1) g_t}{1-\beta_1^t}\right) \qquad (7.18)$$

8. MEME KANSERİ TAHMİNİNDE KULLANILAN VERİ SETİ

Yapay Sinir Ağını Kullanarak Meme kanseri Teşhisinde kullanılan veri seti Kaliforniya Üniversitesi (UCI) Makine Öğrenimi Deposundan alınmış meme kanseri (BC) veritabanıdır (Wolberg, Street, Mangasarian,1992). Makine öğrenme algoritmalarının deneysel analizi için kullanılabilecek birçok veri kümesiyle açık veri havuzu Madison'daki Wisconsin Üniversitesi Hastanesi'nden Dr. William H. Wolberg tarafından oluşturulmuş veri setidir. Bu çalışmada kullanılan veri kümesinde bulunan özelliklerden bazıları yarıçap, doku, çevre, yumuşaklık, kompaktlık, alan, içbükeylik, içbükey noktalar, her hücre çekirdeği için fraktal boyut, simetri ve veri seti için kullanılan UCI makine öğreniminde 569 örnek ve 32 fonksiyondan oluşan bir Wisconsin Diagnostic Breast Cancer (WDBC) veri setidir. WDBC veri setindeki 699 meme kanseri verisinin 458 tanesi iyi huylu (bening) ve 241 tanesi kötü huylu (malignant) kanser hücrelerinin örneklerini göstermektedir. Veri kümesinde iyi huylu kanser hücrelerinin dağılımı daha homojendir ve kötü huylu kanser hücrelerinde yapısal maligniteler bulunur. Veritabanında toplam 11 öznitelik ve değer aralıkları aşağıdaki tabloda gösterilmektedir.

Çizelge 8.1. Veri Setleri ve değer aralıkları

Veri seti	Değer aralığı
Sample code number	ID Numarası
Clump Thickness	1-10
Uniformity of Cell Size	1-10
Uniformity of Cell Shape	1-10
Marginal Adhesion	1-10
Single Epithelial Cell Size	1-10
Bare Nuclei	1-10
Bland Chromatin	1-10
Normal Nucleoli	1-10
Mitoses	1-10

Class	2 / 4

Meme kanseri teşhisi uygulaması tahmininde kullanılan özniteliklerin id numarasından sonraki on tanesi 1 ile 10 arasındaki özniteliklerdir. Son değerimiz ise sonuç kısmı olarak görülmektedir ve eğer iyi huylu ise 2, kötü huylu ise 4 değerini almaktadır.

Kullanılacak veri setinin ilk on verisi Tablo 2' de yer almaktadır.

Çizelge 8.2: Veri Setinde Yer Alan İlk On Veri

Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
1000025	5	1	1	1	2	1	3	1	1	2
1002945	5	4	4	5	7	10	3	2	1	2
1015425	3	1	1	1	2	2	3	1	1	2
1016277	6	8	8	1	3	4	3	7	1	2
1017023	4	1	1	3	2	1	3	1	1	2
1017122	8	10	10	8	7	10	9	7	1	4
1018099	1	1	1	1	2	10	3	1	1	2
1018561	2	1	2	1	2	1	3	1	1	2
1033078	2	1	1	1	2	1	1	1	5	2
1035283	1	1	1	1	1	1	3	1	1	2

Kullanılacak veri setinde verilerin id numarası sonuca herhangi bir değişiklik oluşturmayacağından dolayı bu alanı çıkarıyoruz ve iyi huylu mu kötü huylu mu olup olmadığı kısmı içinde yapay sinir ağı modelimiz iki çıkışa sahip olduğundan dolayı iki alan ekleyip bu durumu belirtmiş olmamız gerekmektedir.

Son durumda tablo 2' de yer alan on tane verinin uygulama tarafından anlaşılacağı formatı tablo 3'teki gibidir.

Çizelge 8.3: YSA Modeli İçin Yeniden Düzenlenen Veri Setinin İlk On Verisi

5	1	1	1	2	1	3	1	1	1
5	4	4	5	7	10	3	2	1	1
3	1	1	1	2	2	3	1	1	1
6	8	8	1	3	4	3	7	1	1
4	1	1	3	2	1	3	1	1	1
8	10	10	8	7	10	9	7	1	0
1	1	1	1	2	10	3	1	1	1
2	1	2	1	2	1	3	1	1	1
2	1	1	1	2	1	1	1	5	1
1	1	1	1	1	1	3	1	1	1

9. KULLANILAN YAPAY SİNİR AĞI MODELİ

Bu çalışmada veri setinde hastalığı teşhis etmek için 9 öznitelik vardır. Bu sebepten giriş katmanında 9 nöron vardır ve tümörler iyi huylu veya kötü huylu olarak ayrıldığı için çıkış katmanında 2 nöron ve ara katmanda 10 nöron bulunmaktadır. Uygulamanın başlangıcında rastgele bias ve ağırlık değerleri oluşturulmuş ve uygulama süresinde bu değerler güncellenerek son değerler bulunmuştur. Uygulamada çıkış nöronlarında hatanın geri yayılması nedeniyle değerlendirmede hata olasılığı en aza indirilmiştir.

Aktivasyon fonksiyonu olarak, ayırt edilmesi kolay olduğu için sigmoid fonksiyonu tercih edilmiştir.

$$\boldsymbol{sigmoid(s)} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e^x}} \qquad (9.1)$$

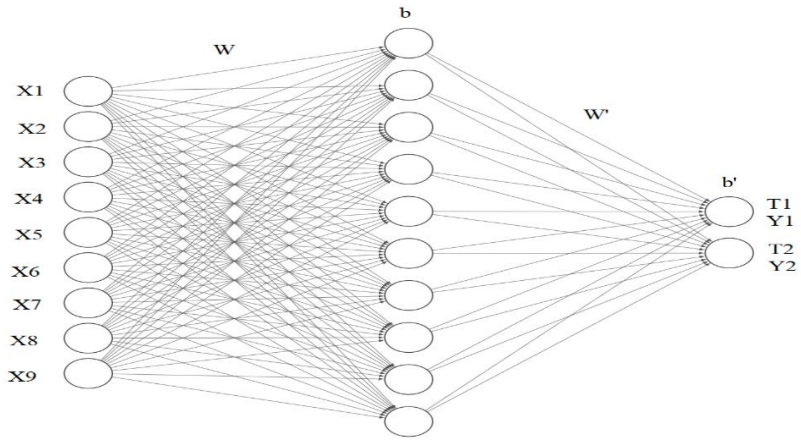
Aşağıdaki formüllere göre hesaplamalar yapılmıştır.

$$\boldsymbol{\theta_n} = \boldsymbol{f}\left(\sum_{\boldsymbol{i=1}}^{\boldsymbol{9}} \boldsymbol{x_i}.\boldsymbol{\omega_{in}} + \boldsymbol{b_n}\right), \boldsymbol{n} = \mathbf{1}, \dots, \mathbf{10}. \qquad (9.2)$$

Burada $\boldsymbol{b_n}$ ve $\boldsymbol{\omega_{in}}$ ara katmanın bias ve ağırlıklarını, $\boldsymbol{\theta_n}$ ara katmanın çıkışı anlamına gelmektedir.

$$\boldsymbol{Y_n} = \boldsymbol{f}\left(\sum_{\boldsymbol{i=1}}^{\boldsymbol{10}} \boldsymbol{\theta_i}.\boldsymbol{\omega'_{in}} + \boldsymbol{b'_n}\right), \boldsymbol{n} = \mathbf{1}, \mathbf{2}. \qquad (9.3)$$

Burada $\boldsymbol{Y_n}$ son katmanının çıkışını temsil etmektedir. $\boldsymbol{b_n}$ ve $\boldsymbol{\omega_{in}}$ise çıkış katmanı biası ve ağırlıklarıdır. Burada öğrenme oranı 0.5 olarak alınmıştır.



Şekil 9.1: Çalışmadaki Yapay Sinir Ağı Modeli

Aşağıdaki formüllerle çalışmadaki güncellemeler yapılmıştır.

$$\boldsymbol{\delta_i} = \left(\boldsymbol{T_J} - \boldsymbol{Y_j}\right).\boldsymbol{Y_J}.\left(\mathbf{1} - \boldsymbol{Y_j}\right)$$

$$\Delta W'_{ij} = (t+1) = 0.5.\delta_j.\theta_{i,j} = 1,2 \; ve \; i = 1,\dots,10.$$

$$W'^{yeni}_{ij} = W'^{eski}_{ij} + \Delta W'_{ij}(t+1)$$

$$\Delta W_{ij}(t+1) = 0,5.\theta_j.(1-\theta_j).\delta_{1,2}.W_{ij}.X_i, \qquad j = 1,\dots,10 \; ve \; i = 1,\dots,9.$$

$$W^{yeni}_{ij} = W^{eski}_{ij} + \Delta W_{ij}(t+1)$$

(9.4)

$$\Delta b'_{ij}(t+1) = 0,5.\delta_j, i = 1,2.$$

$$b'^{yeni}_{ij} = b'^{eski}_{ij} + \Delta b'_{ij}(t+1)$$

$$\Delta b_{ij}(t+1) = 0,5.\theta_j.(1-\theta_j).\delta_{1,2}.W_{j,i}, i = 1,\dots,10 \; ve \; j = 1,\dots,9.$$

$$b^{yeni}_{ij} = b^{eski}_{ij} + \Delta b_{ij}(t+1)$$

$$hata = \frac{1}{2}\sum_{i=1}^{2}(t_n - y_n)^2$$

Hedefdeki çıkış değerlerini formülde $t_{1,2}$, ifade etmektedir.

10. YAPAY SİNİR AĞINDA KULLANILAN OPTİMİZASYON ALGORİTMASININ SEÇİMİ VE PARAMETRELERİN AYARLANMASI

10.1 Optimizasyon Algoritması

Optimizasyon yöntemi olarak 'SGD' (Mei, 2018), 'RMSprop' (Teileman ve Hinton, 2012), 'Adagrad' (Duchi, Hazan, ve Singer, 2011), 'Adadelta' (Zeiler, 2012), 'Adam' (Diederik ve Ba, 2014), 'Adamax'(Diederik ve Ba, 2014) ve 'Nadam' (Dozat, 2016) algoritmalarından en iyi doğruluk oranı vereni tespit etmek için hepsi ile testler gerçekleştirildi. Tablo 4 bu algoritmaların kullandığı formülleri listelemektedir.

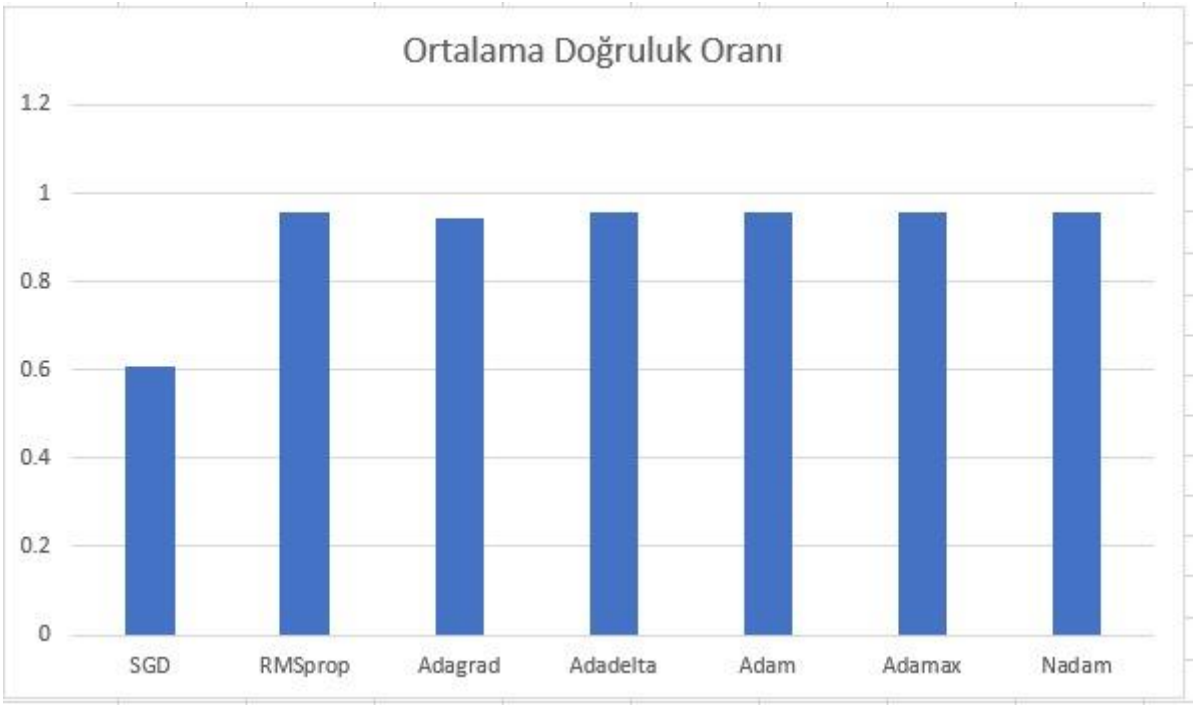
Çizelge 10.1: Optimizasyon Fonksiyonlarının Formülleri

Optimizasyon Algoritması	Formül
Stochastic Gradient Descent (SGD)	$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w),$
Root Mean Square Propagation (RMSprop)	$w := w - \frac{\eta}{\sqrt{v(w,t)}}\nabla Q_i(w)$
Adaptive Gradient Descent (Adagrad)	$G = \sum_{\tau=1}^{t} g_\tau g_\tau^{\mathsf{T}}$
Adaptive Learning Rate (Adadelta)	$\Delta x_t = -\frac{\eta}{\mathrm{RMS}[g]_t}\, g_t$
Adaptive Moment Estimation (Adam)	$w^{(t+1)} \leftarrow w^{(t)} - \eta\frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$
Adaptive Moment Estimation Maximum Adamax	$w_{t+1} = w_t - \frac{\alpha}{S_t}\cdot \hat{V}_t$

Nesterov-accelerated Adaptive Moment Estimation (Nadam)	$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t} + \epsilon}\left(\beta_1 \hat{V}_t + \frac{1-\beta_1}{1-\beta_1^t} \cdot \frac{\partial L}{\partial w_t}\right)$

Çizelge 10.2: Farklı optimizasyon yöntemleri için ortalama doğruluk oranı

Optimayzer	Ortalama doğruluk oranı
SGD	0.607
RMSprop	0.957
Adagrad	0.942
Adadelta	0.957
Adam	0.957
Adamax	0.957
Nadam	0.957



Şekil 10.1: Farklı optimizasyon yöntemleri için ortalama doğruluk oranı grafiği

Tablo 5 ve Şekil 2'deki grafikte görüldüğü üzere SGD ve Adagrad dışındakiler birbirine yakın doğruluk oranı vermektedirler ve bu uygulamada tercih edilebilirler. Birbirine yakın doğruluk değerler verseler de en iyi doğruluğu ADAM verdiği için bundan sonraki incelemeler ADAM algoritması kullanılarak yapıldı.

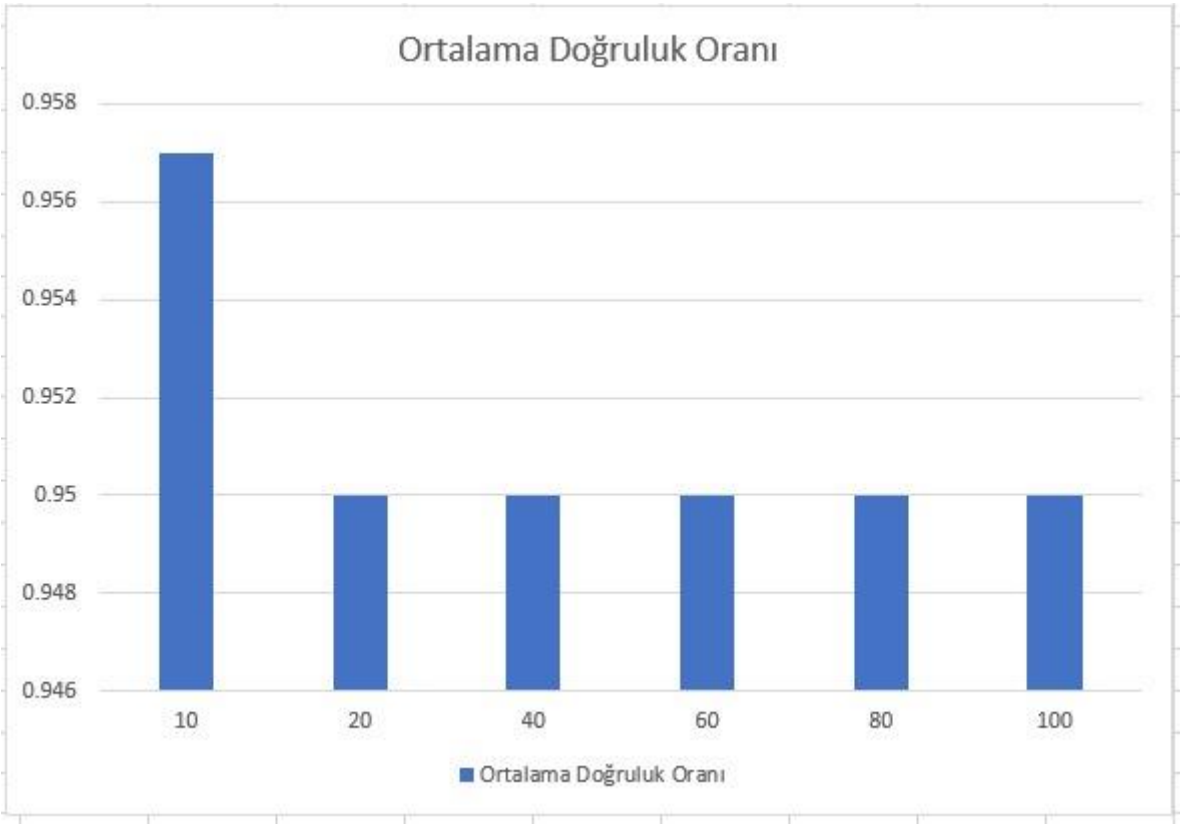
10.2 Batch Size

Grup boyutu (Batch Size) ağ üzerinden dağıtılacak örnek sayısını belirlemektedir. Örneğin, 1050 eğitim örneğiniz olduğunu ve batch_size değerini 100 olarak ayarlamak istediğinizi varsayalım. Algoritma ilk 100 örneği (1'den 100'e kadar) eğitim veri kümesinden alır ve ağı eğitir. Sonra ikinci 100 örneği (101'den 200'e kadar) alır ve ağı tekrar eğitir. Tüm örnekleri ağ üzerinden dağıtana kadar bu işlemi yapmaya devam edebiliriz. Son örnek kümesinde bir sorun ortaya çıkabilir. Örneğimizde, geriye kalan 100'e bölünmeyen 1050'yi kullandık. En basit çözüm, son 50 örneği alarak ağı eğitmektir.

Numunelerin gruplara bölerek kullanmanın faydaları şunlardır. Ağı daha az örnek kullanarak eğittiğiniz için, genel eğitim prosedürü daha az bellek gerektirir. Bu, tüm veri setinin makinenin belleğine sığmadığı durumlarda özellikle önemlidir. Ağlar genellikle mini paketlerle daha hızlı öğrenir. Bunun nedeni, her yayılmadan sonra ağırlığı güncellememizdir. Örneğimizde 11 paket dağıttık (10 tanesi 100 örnek ve 1 tanesi 50 örnek vardı) ve her birinin ardından ağımızın parametrelerini güncelledik. Tüm örnekleri dağıtım sırasında kullansaydık, ağ parametresi için sadece 1 güncelleme yapılırdı.

Çizelge 10.3: Batch Size için ortalama doğruluk oranı

Batch Size	Ortalama doğruluk oranı
10	0.957
20	0.950
40	0.950
60	0.950
80	0.950
100	0.950



Şekil 10.2: Batch Size için ortalama doğruluk oranı grafiği

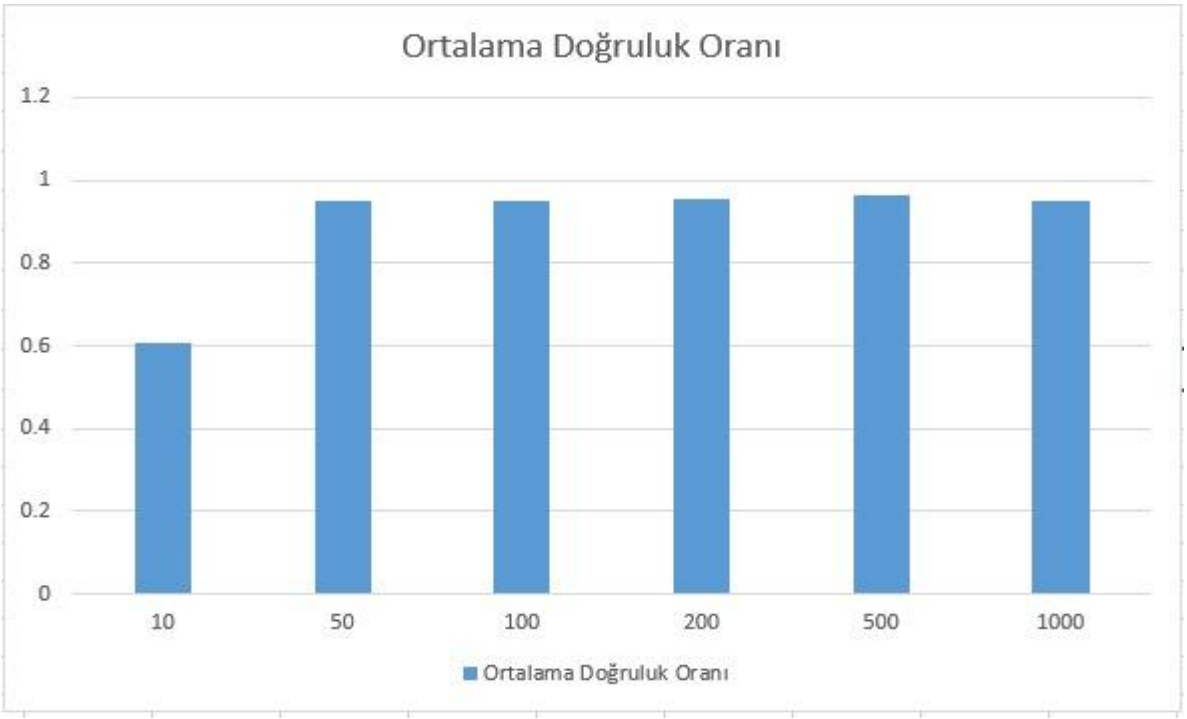
Tablo 6 ve Şekil 3'teki grafiğe göre en iyi sonuç batch size 10 olduğunda bulunuyor.

10.3 Epoch

Devir sayısı (Epoch) öğrenme algoritmasının tüm eğitim verileri kümesinde kaç kez çalışacağını belirleyen bir hiperparametredir. Bir epoch, eğitim veri setindeki her örneğin modelin dahili parametrelerini güncelleme fırsatı bulduğu anlamına gelir. Epoch bir veya daha fazla gruptan (batch) oluşur. Örneğin, yukarıda belirtildiği gibi, bir grup olduğu bir döneme grup gradyanı iniş (batch gradient descent) öğrenme algoritması denir. Her döngünün bir dizi eğitim verisinden geçtiği devir sayısı için bir for döngüsü düşünebiliriz. Bu döngü için, her bir numune grubu üzerinde yinelenen başka bir iç içe for döngüsü vardır, burada bir grup belirli sayıda "grup boyutu (batch size) " örneğine sahiptir. Devir sayısı geleneksel olarak büyüktür, genellikle yüzlerce veya binlercedir, bu da öğrenme algoritmasının model hatası en aza indirilene kadar çalışmasına izin verir. Literatürde devir sayısının örneklerini 10, 100, 500, 1000 ve daha büyük olarak örnekleri vardır. Tipik olarak, X ekseni boyunca evreleri Y ekseni üzerinde zaman ve hata veya model becerisi olarak gösteren çizgi grafikler oluşturulur ve bu grafiklere bazen öğrenme eğrileri denir. Bu grafikler, modelin yeniden eğitilip öğrenilmediğini, iyi anlaşılmadığını veya bir eğitim veri kümesi için uygun olup olmadığını teşhis etmeye yardımcı olabilir.

Çizelge 10.4: Epoch için ortalama doğruluk oranı

Epoch	Ortalama doğruluk oranı
10	0.607
50	0.950
100	0.950
200	0.957
500	0.964
1000	0.950



Şekil 10.3: Epoch için ortalama doğruluk oranı grafiği

Tablo 7 ve Şekil 4'te görüldüğü gibi epoch 500 olunca doğruluk oranı en iyi oluyor.

11. SONUÇ

Yapay sinir ağlarının doğrusal bir yapısı olmadığından dolayı, bu makalede bahsettiğimiz değişkenlerin doğru belirlenmesi daha fazla önem arz etmektedir. Bu parametre değerleri için net ve sabit bir değer verilemeyeceği gibi yapay sinir ağının eğitiminde kullanılan veri setinin türüne göre de değişiklik göstermektedir. Yaptığımız çalışmalarda optimizasyon algoritmasının ve parametre değerlerinin sonucu nasıl etkilediğini gördük. Başlangıç parametrelerimizi güncellediğimizde ortalama doğruluk oranı 0.607'den 0.96'ya yükselmiştir. WDBC veri setindeki öznitelikler ile yapay sinir ağları kullanarak meme kanseri tahmini için optimizasyon algoritması tercihi ve parametre ayarlarının aşağıdaki gibi olması, testlerimizde en iyi doğruluk oranını vermiştir.

- Optimizasyon Algoritması: ADAM
- Grup Boyutu (Batch Size): 10
- Devir Sayısı (Epochs): 500

KAYNAKLAR

Aleksandroviç X, K., Ryazanov M.A., (Barnaul 2016) http://elibrary.asu.ru/xmlui/bitstream/handle/asu/2682/vkr.pdf?sequence=1&isAllowed=y

David H. Staelin and Carl H. Staelin (2011) "Models for Neural Spike Computation and Cognition" ISBN 9781466472228 CreateSpace, Seattle, Washington. November 2011.

Dreyfus, G., (2005), Neural networks methodology and appplications, Springer-Verlag, Berlin, Heidelberg, 497 p.

E.Harwich, K.Laycock., (2018) Thinking on its own: AI in the NHS [Electronic resource] URL: http://www.reform.uk/publication/thinking-on-its-own-ai-in-the-nhs/ (application date17.05.2018).

Efe, M. ve Kaynak, O., (2000), Yapay sinir ağları ve uygulamaları, Boğaziçi Üniversitesi Yayınları, 148 s.

Elmas, Ç., (2007), Yapay zeka uygulamaları, Seçkin Yayıncılık San ve Tic Aş, 425 s**.**

Ergezer, H., Dikmen, M. ve Özdemir, E., (2003), Yapay sinir ağları ve tanıma sistemleri, Pivolka, 6, 14-17.

Fırat, M. ve Güngör, M., (2004), Askı madde konsantrasyonu ve miktarının yapay sinir ağları ile belirlenmesi, İMO Teknik Dergi, 3267-3282.

Fogel D. B., Wasson E.C., Boughton E.M. ve Porto V.W.,(1997) "A step toward computerassisted mammography using evolutionary programming and neural Networks", Cancer Letters, (1997), 119 (1), sayfa 93-97.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, (2016) "Deep networks with stochastic depth," in European Conference on Computer Vision. Springer, 2016, pp. 646–661.

Gorunescu M., Gorunescu F., ve Revett K.,(2007) "Investigating a Breast Cancer Dataset Using a Combined Approach: Probabilistic Neural Networks and Rough Sets", Proc. 3rd ACM International Conference on Intelligent Computing and Information Systems -ICICIS07, Cairo, Egypt, (2007), pp. 246-249.

Hsiao Y.H., Huang Y.L., Liang W.M., Kuo S.J. and Chen D.R., (2009) "Characterization of benign and malignant solid breast masses: harmonic versus nonharmonic 3D power Doppler imaging", Ultrasound Medicine & Biology, (2009), 35 (3), pp. 353-359**.**

Iliyan Mihaylov , Maria Nisheva , and Dimitar Vassilev (2019) "Application of Machine Learning Models for Survival Prognosis in Breast Cancer Studies" Published: 3 March 2019 file:///C:/Users/Admin/Downloads/information-10-00093.pdf

J. Duchi, E. Hazan, and Y. Singer, (2011) "Adaptive subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2121–2159, 2011.

JASON The MITRE Corporation (2017) Artificial Intelligence for Health and Health Care [Electronic resource]. URL: https://www.healthit.gov/sites/default/files/jsr-17-task-002_aiforhealthandhealthcare 12122017.pdf (application date17.05.2018).

Kalach, A.V., (2005), Application of neural networks in multitouch-sensitive systems for the detection of nitrohydrocarbons in the air, Sensors Journal, 5, 97-102.

Kalach, A.V., (2005), Application of neural networks in multitouch-sensitive systems for the detection of nitrohydrocarbons in the air, Sensors Journal, 5, 97-102.

Kargı, V. (2015). Yapay Sinir Ağ Modelleri ve Bir Tekstil Firmasında Uygulama, Ekin Yayınevi

Kartalapoulos, S.V., (1996), Understanding neural networks and fuzzy logic basic concepts and applications, IEEE Pres, 205 p.

Kasabov, N.K., (1998), Foundation of neural networks, fuzzy systems and knowledge engineering, The MIT Press, England, 550 p.

M. Ishii and A. Sato, (2017) "Layer-wise weight decay for deep neural networks," in Pacific-Rim Symposium on Image and Video Technology. Springer, 2017, pp. 276–289.

Öztemel, E., (2006), Yapay sinir ağları, Papatya Yayıncılık Eğitim Bilgisayar Sis. San. Ve Tic. A.Ş., 232 s.

Ramirez-Quintana J.A., Chacon-Murguia M. I. ve Chacon-Hinojos J. F.,(2012) "Artificial Neural Image Processing Applications: A Survey", Engineering Letters, (2012), 20 (1), sayfa 68-81.

Revett K., Gorunescu F., Gorunescu M., El-Darzı E. ve Ene M.,(2005) "A breast cancer diagnosis system: a combined approach using rough sets and probabilistic neural Networks", Computer as a tool Eurocon 2005, Belgrade, (2005), pp. 1124- 1127.

Rosenblatt, F., (1958), The perceptron: a probabilistic model for information stroge and organization in the brain, Pyschological Review, Volume 65, No 6, 386-408.

S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, (2018) "Don't decay the learning rate, increase the batch size," in International Conference on Learning Representations (ICLR), 2018.

Sebastian Ruder (2017) "An overview of gradient descent optimization algorithms" Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublinarxiv:1609.04747v2 [csLG] 15 June 2017

Shi Z. ve He L.,(2010) "Application of Neural Networks in Medical Image Processing", Proceedings of the Second International Symposium on Networking and Network Security (ISNNS '10) , China, (2010), sayfa. 2-4.

Şen, Z., (2004), Yapay sinir ağları ilkeleri, Su Vakfı yayınları, 183 s

Uncini A., (2003) "Audio signal processing by neural Networks", Neurocomputing, (2003), 55 (3-4), sayfa. 593 – 625.

William H Wolberg, W Nick Street, and Olvi L Mangasarian. (1992). Breast cancer Wisconsin (diagnostic) data set. UCI Machine Learning Repository [http://archive. ics. uci. edu/ml/] (1992)

Y. Nesterov, (1983) "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in Doklady AN USSR, vol. 269, 1983, pp. 543–547.

Yurtoğlu, H., (2005), Yapay sinir ağları metodolojisi ile öngörü modellemesi: bazı makroekonomik değişkenler için Türkiye örneği, Uzmanlık tezi, DPT Ekonomik Modeller ve Stratejik Araştırmalar Genel Müdürlüğü, 104 s.,

Yücesoy, M. (2011). Temizlik Sektöründe Yapay Sinir Ağları ile Talep Tahmini, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitisü, Endüstri Mühendisliği Bölümü, Yüksek Lisans Tezi.

Z. Huo and H. Huang, (2017) "Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization," in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

EKLER

Ek-1: Proje Kaynak Kodları

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

URL =""
dataset = pd.read_csv(URL)
dataset.head()

X = dataset.iloc[:,0:9].values
y = dataset.iloc[:,9:11].values

from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test = train_test_split(X,y, test_size = 0.2,
random_state = 0)

from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers

classifier = Sequential()
classifier.add(Dense(units = 5, kernel_initializer = 'uniform', activation =
'softmax', input_dim = 9))
classifier.add(Dense(units = 10, kernel_initializer = 'uniform', activation
= 'softmax'))
classifier.add(Dense(units = 2, kernel_initializer = 'uniform', activation =
'softmax'))
classifier.compile(optimizer = 'Adam', loss = 'categorical_crossentropy',
metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 10, epochs = 500)

y_pred = classifier.predict(X_test)
for x in y_pred:
    if (x[0] > 0.5):
        x[0] = 1
        x[1] = 0
    else:
        x[0] = 0
        x[1] = 1
print("Test verisi sonuclari: \n")
y_pred
print(y_pred)

dogruluk = accuracy_score(y_test, y_pred)
print("\n doğruluk Oranı: \n")
print(dogruluk)
```

ÖZ GEÇMİŞ

Ad -Soyad: Mariya Kiknadze

Doğum Tarihi ve Yeri: 1993, Azerbaycan

E-Posta: mariyamirabellla@gmail.com

ÖĞRENİM DURUMU:

- **Lisans:** 2014, Azerbaycan Devlet Pedoqoji Üniversitesi, Matamatik ve Bilgisayar
- **Yüksek Lisans:** Devam etmektedir, İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği

YABANCI DİLLER:

- **İngilizce**
- **Rusça**
- **Türkçe**
- **Azerbaycan dili (Ana dil)**