



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

HASTALIK TEŞHİSİ İÇİN BİR YAPAY SİNİR AĞLARI YAZILIMININ TASARLANMASI VE GERÇEKLEŞTİRİLMESİ

Kadir TOK

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalını

Ocak-2017
KONYA
Her Hakkı Saklıdır

ÖZET

YÜKSEK LİSANS TEZİ

HASTALIK TEŞHİSİ İÇİN BİR YAPAY SİNİR AĞLARI YAZILIMININ TASARLANMASI VE GERÇEKLEŞTİRİLMESİ

Kadir TOK

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. İsmail SARITAŞ

2016, 94 Sayfa

Jüri
Doç. Dr. İsmail SARITAŞ
Doç. Dr. Harun UĞUZ
Yrd. Doç. Dr. Onur İNAN

Bu tez çalışmasında, tıbbi tanılamayı desteklemek amacıyla yapay sinir ağları yöntemi kullanılmıştır. C#.NET programlama dili ile görsel arayüze sahip olan bir yazılım gerçekleştirilmiştir. Yazılım ile, sonuçların başarı analizi yapılabilmekte ve performansı ölçülebilmektedir. Yazılım, yapay sinir ağlarıyla ilgili uygulamalarda ve akademik çalışmalarda kullanılabilecek şekilde tasarlanmıştır. Yapılan çalışmalarda, eğitim ve test veri seti olarak romatoid artrit hastası olan ve olmayan insanlardan alınmış veriler kullanılmıştır. Yazılım ile, dinamik biçimde belirlenen nöron sayısı, öğrenme hızı, momentum katsayısı ve iterasyon sayısına göre eğitim yapılabilmektedir. Ağın, 0 (sıfır) hataya, ortalama olarak 500 ve 1500 iterasyon aralığında ulaştığı tespit edilmiştir. Farklı işlemcili bilgisayarlarda, 1500 ve üzerinde gerçekleştirilen iterasyonlarda %82 ve %100 aralığında başarılar elde edilmiştir. Çalışmada, yapay sinir ağlarının önemi ve öğrenebilen yazılımların bilgisayar teknolojilerindeki avantajları incelenmiştir. Ayrıca, romatoid artrit hastalığı teşhisi için Backpropagation algoritması Matlab ortamında da incelenmiştir. Backpropagation ve Perceptron algoritması ile bulunan sonuçlar, performans açısından karşılaştırılmıştır. Backpropagation algoritması ile %82 doğruluk yüzdesi elde edilirken, Perceptron algoritması ile doğruluk yüzdesi %71 olarak bulunmuştur.

Anahtar Kelimeler: Yapay sinir ağları, hastalık teşhisi, yapay zeka.

ABSTRACT

MS THESIS

DESIGN AND IMPLEMENTATION OF AN ARTIFICIAL NEURAL NETWORK SOFTWARE TO DIAGNOSE DISEASE

Kadir TOK

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. İsmail SARITAŞ

2016, 94 Pages

Jury
Assoc. Prof. Dr. İsmail SARITAŞ
Assoc. Prof. Dr. Harun UĞUZ
Assist. Prof. Dr. Onur İNAN

In this thesis, method of artificial neural networks is used in order to support medical diagnostics. A software, which has a visual interface, has been designed with C#.NET programming language. With this software, success analysis of the results can be made and the performance of the results can be measured. This software was designed to be able to be used in academic studies and in applications which are related to artificial neural networks. In the studies, the datas which are received from the patients with rheumatoid arthritis and from the people who are not suffering from rheumatoid arthritis are used as training and test data sets . With the software, training takes place according to the desired number of neurons, learning rate, momentum coefficient and number of iterations which are determined dinamically. It was found that the network, on average, reaches the zero error rate, in the range of 500 and 1500 iterations. In the iterations which were performed 1500 times and more in computers with different processors, 82% to 100% success rate has been obtained. In this study, the importance of artificial neural networks and the advantages of the software, that can learn, in computer technologies have been examined. Also, backpropagation algorithm was examined in Matlab environment for the diagnosis of rheumatoid arthritis. The results found with the backpropagation algorithm and the perceptron algorithm have been compared in terms of performance. While %82 accuracy percentage is obtained with the Backpropagation algorithm, the accuracy percentage is found as %71 with Perceptron algorithm.

Keywords: Artificial neural networks, disease diagnosis, artificial intelligence.

ÖNSÖZ

Bu tez çalışmasında, bana her türlü maddi ve manevi destek veren aileme teşekkürü borç bilirim.

Tezin gerçekleşmesinde bana klavuzluk eden, tavsiyeleri ile daha doğruya erişmemi sağlayan, yapay sinir ağları alanında çalışmamda beni yüreklendiren ve aydınlatan, yardımlarını hiçbir zaman esirgemeyen tez danışmanım Sayın Doç. Dr. İsmail SARITAŞ'a teşekkür ederim.

Tezimdeki, romatoid artrit hastalığı ölçümleri yapılmış bireylerin verilerini temin etmemde yardımcı olan, tavsiye ve yardımlarıyla bana yol gösteren, Selçuk Üniversitesi Tıp Fakültesi'nde romatoloji bölümünden Sayın Doç. Dr. Sema YILMAZ'a teşekkür ederim.

Kadir TOK
KONYA-2017

İÇİNDEKİLER

ÖZET ..... **vii**

ABSTRACT ..... **viii**

ÖNSÖZ ..... **ix**

İÇİNDEKİLER ..... **x**

SİMGELER VE KISALTMALAR ..... **xii**

1. GİRİŞ **1**

2. YAPAY SİNİR AĞLARI **2**

2.1. Yapay Sinir Ağlarının Tanımı ve Genel Bilgiler 2
2.2. Yapay Sinir Ağlarının Yapısı 7
2.2.1. Biyolojik Sinir Hücresinin Yapısı 7
2.2.2. Yapay Sinir Hücresinin Yapısı 9
2.2.3. Yapay Sinir Ağının Yapısı 12
2.3. Yapay Sinir Ağlarının Sınıflandırılması 13
2.3.1. Yapılarına Göre Yapay Sinir Ağları 13
2.3.2. Öğrenme Zamanına Göre Yapay Sinir Ağları 14
2.3.3. Öğrenme Algoritmalarına Göre Yapay Sinir Ağları 14
2.4. Yapay Sinir Ağlarında Öğrenme Kuralları 15
2.5. Yapay Sinir Ağlarının Eğitilmesi ve Öğrenmesi 16
2.6. Yapay Sinir Ağlarının Uygulama Alanları 18
2.7. Yapay Sinir Ağlarının Avantaj ve Dezavantajları 19
2.7.1. Yapay Sinir Ağlarının Avantajları 19
2.7.2. Yapay Sinir Ağlarının Dezavantajları 21
2.7.3. Geleneksel Algoritmalar ile Yapay Sinir Ağlarının Karşılaştırılması 22
2.8. Bölüm Değerlendirmesi 22

3. İLGİLİ ÇALIŞMALAR **23**

3.1. SNNS 23
3.2. FANN 25
3.3. JOONE 27
3.4. Matlab Neural Network Toolbox (NNTOOL) 28
3.5. Bölüm Değerlendirmesi 29

4. KAYNAK ARAŞTIRMASI **31**

4.1. Yapay Siniri Ağları Yöntemleri ile Başağrısı Hastalıklarının Sınıflanadırılması 31
4.2. Troid Bezi Bozukluklarının Yapay Sinir Ağları ile Teşhisi 32

4.3. Ateroskleroz'un Tahmini İçin Bir Yapay Sinir Ağı 34
4.4. Yapay Sinir Ağları ile Ağız Kokusundan Diyabet Tanısı 36
4.5. Eşit Genişlikli Ayrıştırma ve Yapay Sinir Ağı Temelli Meme Kanseri Teşhisi . 38
4.6. Yapay Sinir Ağları Kullanılarak EMG Sinyallerinin Sınıflandırılması ve Neuropathy Hastalığının Teşhisi 40
4.7. Yapay Sinir Ağları ve Destek Vektör Makineleri ile Kemik Erimesinin Teşhisi 41
4.8. Yapay Sinir Ağları ile Kronik Obstrüktif Akciğer Hastalığının Sınıflandırılması 42
4.9. Bölüm Değerlendirmesi 44

5. MATERYAL VE METOT 45

5.1. C# Programlama Dili 45
5.2. .NET Framework 46
5.3. Microsoft Visual Studio 46
5.4. Bölüm Değerlendirmesi 47

6. YAZILIMIN TASARIMI VE GERÇEKLEŞTİRİMİ 48

6.1. Romatoid Artrit Hastalığının İncelenmesi 48
6.2. Yapay Sinir Ağında Kullanılan Veri Seti 49
6.3. Yapay Sinir Ağının Yapısı 49
6.4. Transfer Fonksiyonu 50
6.5. Öğrenme Algoritması 51
6.6. Geliştirilen Yazılımın Özellikleri 53
6.7. Bölüm Değerlendirmesi 55

7. SONUÇLAR 56

7.1. Matlab YSA Uygulaması ve Sonuçları 56
7.2. Gerçekleştirilen Yazılımın Uygulama Sonuçları 59
7.3. Genel Değerlendirme 65

KAYNAKLAR 66

EKLER 68

ÖZGEÇMİŞ 85

SİMGELER VE KISALTMALAR

YSA: Yapay Sinir Ağları

ANN: Artificial Neural Networks

SOM: Kendini Düzenleyen Haritalar

ART: Adaptif Rezonans Teorisi

RBF: Radial Tabanlı Fonksiyonlar

PNN: Probabilistik Sinir Ağları

GRNN: Genel Regresyon Sinir Ağları

VLSI: Çok Geniş Ölçekli Tümleşim

EEG: Elektroensefalografi

ECG: Elektrokardiyogram

MR: Manyetik Rezonans

FANN: Fast Artificial Neural Network Library

SNNS: Stuttgart Neural Network Simulator

JOONE: Java Object Oriented Neural Engine

NNTOOL: MATLAB Neural Network Toolbox

LVQ: Öğrenen Vektör Parçalama

TDNN: Time-delay Neural Networks

IDE: Integrated Development Environment

GUI: Graphical User Interface

MLP: Multi Layer Perceptron

1. GİRİŞ

Bilgisayarlar zamanla yaygınlaşarak, günlük hayatımızın temel bir unsuru olmaya başlamıştır. Hayatımızda çok fazla pay sahibi olan bilgisayarların, insanlara daha çok yardımcı olabilmesi için her gün çalışmalar yapılmaktadır. İşyerlerinde kayıtların bilgisayar ortamında tutulabilmesi, randevuların bilgisayar ortamı üzerinden takip edilebilmesi, ihtiyaç duyulan bilgilerin, gün geçtikçe yaygınlaşmakta olan internet ortamında aranıp bulunabilmesi gibi olumlu yönlerinden dolayı insanlar rutin ve kritik işlerini bilgisayar ortamı yardımıyla yapabilmektedirler (Doğan, 2003).

Bilişim teknolojisindeki büyük ilerlemelere rağmen bilgisayarlar henüz insanların günlük yaşamda rahatlıkla gerçekleştirdikleri işlemleri aynı rahatlıkla gerçekleştirememektedirler. Yalnızca kendisine daha önceden hazırlanıp verilen komut setlerini peşpeşe oldukça hızlı yapabilen bilgisayarlar, insanlar gibi düşünememektedirler. İnsan beyninin her çeşit probleme çözüm getirebilme kabiliyetini inceleyen bilim insanları, insan beynini bilgisayar ortamında modelleyerek analizi zor problemleri çözmeyi hedeflemişlerdir. Bunun için geliştirilen yapay sinir ağları, insan beynini taklit ettiği için alışılagelmiş programlama mantığıyla analiz edilemeyen sorunlara kabul edilebilir analizleri kısa sürede oluşturabilmektedir. Bu niteliği sayesinde YSA'lar oldukça çok alanda uygulanmış ve başarılı sonuçlar elde edilmiştir (Doğan, 2003).

Yapay sinir ağları alanında gerçekleştirilen çalışmalar gün geçtikçe artmakta ve yapay sinir ağlarının uygulama alanları gittikçe çoğalmaktadır. Hastalıkların teşhisinde yapay zeka yöntemleri ile beraber yapay sinir ağları da kullanılmaya başlamıştır. Hastalıkların teşhisinde tıp uzmanları, deneyimlerine, bildiklerine ve laboratuvar analizlerine dayanarak kanıya varabilmektedirler. Tıp doktorları ne kadar branşında uzman da olsa psikolojik durum, uyku, rahatsızlık vb sebeplerden dolayı yanlış tanılar koyabilmektedirler. Teşhis problemlerinde insan etkenininin devre dışı bırakılması için bilgisayar ortamında analiz edilebilmesi için günümüze kadar yurt içinde ve yurt dışında birtakım çalışmalar gerçekleştirilmiştir (Doğan ve ark., 2004).

İnsan beyninde yaklaşık 10^{11} sinir hücresinin bulunduğu bilinmektedir, bu sayının bilgisayar ortamında tasarlanması henüz mümkün değildir. Ancak karar hızı açısından insan beyni ile kıyaslanamasalar dahi, YSA'lar yapısallıkları ve hassas eşleştirmelerin muvaffakiyetle yapılmaları ile gittikçe daha çok uygulama alanı bulmaktadır (Tektaş ve ark., 2010).

2. YAPAY SİNİR AĞLARI

2.1. Yapay Sinir Ağlarının Tanımı ve Genel Bilgiler

Yapay sinir ağları, insan beyninin niteliklerinden biri olan öğrenme yöntemi ile yeni bilgiler üretebilme, yeni bilgiler meydana getirebilme ve bulabilme gibi kabiliyetleri, herhangi bir destek olmadan otomatik olarak yapabilmek için gerçekleştirilen bilgisayar sistemleridir. Yapay sinir ağları diğer bir ifadeyle, biyolojik sinir ağlarına benzemeye çalışan bilgisayar programlarıdır. İnsan beyninin fonksiyonel niteliklerine benzer biçimde;

- Öğrenme
- İlişkilendirme
- Sınıflandırma
- Genelleme
- Özellik belirleme
- Optimizasyon

gibi alanlarda başarılı bir biçimde gerçekleştirilmektedirler. Örneklerden buldukları bilgiler ile kendi tecrübelerini meydana getirir ve benzer alanlarda benzer kararlar verirler (Öztemel, 2003).

Yapay sinir ağları, farklı zeki nitelikleri barındırmasından ötürü birçok kullanım alanlarında kullanılmaktadır. Yapay sinir ağları, bir sisteme ait birtakım parametrelere bağlı olarak belirtilen girişler ve çıkışlar arasında ilişki oluşturabilme kabiliyetine sahiptir. Bu ilişkinin doğrusal biçimde olma zorunluluğu yoktur. Ayrıca yapay sinir ağları, çıkış değerleri belli olmayan sistem girişleri ile de uyumlu olarak çıkışlar sağlamakta, bu sayede oldukça kompleks sorunlara bile iyi çözümler getirebilmektedirler (Güven ve ark., 2008).

Yapay sinir ağlarının temel kullanım alanları tahmin, modelleme ve sınıflandırma olarak incelenebilir. Bunlar aşağıda izah edilmiştir (Elmas, 2003);

Sınıflandırma: Müşteri/Pazar tıbbi teşhis, profilleri, imza tetkikleri, ses tanıma, borçlanma/risk değerlendirmeleri, mal değeri, şekil tanıma, hücre tiplerinin sınıflandırılması.

Tahmin: İleriki satışlar, pazar performansı, üretim ihtiyacı, enerji ihtiyacı, ekonomik deliller, kimyasal reaksiyon ürünleri, tıbbi sonuçlar, hava tahminleri, çevresel riskler, at yarışları. Ayrıca, inşaat yapı maliyet tahminleri ve farklı maliyet tahminleri.

Modelleme: İşlem kontrolü, dinamik sistemler, kimyasal yapılar, kaynak kontrolü, işaret karşılaştırma, robot kontrolü ve farklı birçok uygulamalar (Yılmaz, 2012).

Yapay sinir ağları; insan beyninden ilham alarak, öğrenme işleminin sayısal olarak tasarlanması sonucu meydana gelmiştir. Bu sebepledir ki, bu alandaki çalışmalar öncelikle beyni meydana getiren biyolojik birimler olan nöronların tasarlanması ve bilgisayar sistemlerinde tatbik edilmesi ile başlamış, ve ilerleyen zamanlarda bilgisayar sistemlerinin ilerlemesine de paralel olarak birtakım konularda uygulanabilir olmuştur (Çayıroğlu, 2013).

İnsan beyninin çalışma ilkelerine benzemeye çalışarak çalışan bu sistemler, her ne kadar bilgisayar teknolojisi süratli bir ilerleme kaydetmiş, işlem hızları nano saniyeler derecesine indirgenmiş olsa da, insan beynini geçelim, ilkel bir canlının beyinsel fonksiyonları bile baz alındığında, böyle bir organizmanın yanında oldukça gelişmemiş kalmaktadır. Nano saniyeler bazındaki işlem süratleri ile YSA'lar, mili saniyeler derecesindeki işlem sürat ile işlem yapan insan beyninin fonksiyonlarının şu an oldukça uzağındadır (Tektaş ve ark., 2010).

Yapay sinir ağları hafızaya alma, öğrenme ve datalar arasındaki bağlantıyı meydana çıkarma yeteneğine sahiptirler. Başka bir deyişle, YSA'lar, alışılagelmiş bir durumda insanın gözlemleme ve düşünmeye yönelik doğal kabiliyetlerini isteyen sorunları analiz etmektedir. Bir insanın, gözlemleme ve düşünme kabiliyetlerini isteyen sorunlara yönelik çözümler getirebilmesinin temel nedeni ise insan beyninin yani insanın sahip olduğu deneyerek veya yaşayarak öğrenme kabiliyetidir (Anonim).

Yapay sinir ağlarının ilgi çekici ve faydalı birtakım niteliği vardır. En önemli niteliği öğrenebilme kabiliyetidir. Yapay sinir ağlarının öğrenme ilkelerini ve yapısını değiştirmeden yalnızca öğretim gereçlerini değiştirerek öğrenmesi sağlanabilmektedir (Elmas, 2003).

Yapay sinir ağlarının temel kendine özgü nitelikleri aşağıdaki gibi sıralanabilir (Öztemel, 2003):

Yapay sinir ağları makine öğrenmesi gerçekleştirirler: Yapay sinir ağlarının temel görevi makinaların öğrenmesini sağlamaktır. Olayları öğrenerek benzerlik gösteren olaylara karşı benzeyen tanılar ortaya koymaya çalışırlar.

Bilginin saklanması: Yapay sinir ağlarında bilgi, ağın bağlantılarının sayısal değerleri ile ölçülmekte ve bu bağlantılarda tutulmaktadır. Diğer programlardaki gibi datalar bir veri tabanında veya programın içinde bütünleşik değildir.

Yapay sinir ağları örnekleri kullanarak öğrenirler: Yapay sinir ağlarının olayları öğrenebilmesi için o konu ile alakalı örneklerin belirtilmesi gerekmektedir. Örnekleri kullanarak alakalı olay hakkında genelleme yapabilecek kabiliyete kavuşturulurlar. Örnekler ise meydana gelmiş olan olaylardır.

Yapay sinir ağlarının güvenle çalıştırılabilmesi için önce eğitilmeleri ve performanslarının test edilmesi gerekmektedir: Yapay sinir ağlarının eğitilmesi demek, var olan örneklerin sırayla ağa gösterilmesi ve ağın kendi sistemlerini çalıştırarak örnekteki olaylar arasındaki bağlantıları belirlemesidir. Herbir ağı eğitmek için elde bulunan örnekler iki farklı sete ayırılırlar. İlki ağı eğitmek için (eğitim seti) diğeri ise ağın başarımını test etmek için (test seti) kullanılır. Herbir ağ ilk olarak eğitim seti ile eğitilir. Ağ örneklerin tamamına doğru cevaplar vermeye başlayınca eğitim işi tamamlanmış varsayılır. Bu işlemden sonra ağın daha önce hiç görmediği test setindeki örnekler ağa verilerek ağın cevaplarına bakılır. Eğer ağ hiç görmediği örneklere başarılı bir doğrulukla cevap veriyor ise o zaman ağın başarımı iyi kabul edilir ve ağ kullanıma alınır. Eğer ağın performansı vasat olursa o zaman tekrar eğitmek veya yeni örneklerle eğitmek gibi bir yönteme başvurulur.

Kendi kendini organize etme ve öğrenebilme yetenekleri vardır: Yapay sinir ağlarının örnekler ile kendisine verilen yeni durumlara uyum sağlayabilmesi ve sürekli yeni olayları öğrenebilmesi olasıdır.

Eksik bilgi ile çalışabilmektedirler: Yapay sinir ağları, eğitildikten sonra eksik bilgiler ile de çalışabilir ve yeni gelen örneklerde eksik bilgiler olsa dahi sonuç verebilirler. Eksik bilgiler ile de çalışmaya devam ederler.

Hata toleransına sahiptirler: Yapay sinir ağlarının eksik bilgilerle çalışabilme kabiliyetleri hatalara karşı göz yumabilmelerini sağlamaktadır. Ağın bazı hücrelerinin özelliğini yitirmesi ve çalışamaz duruma düşmesi durumun ağ çalışmasına devam eder. Ağın bozulan hücrelerinin işlevlerinin önemine göre ağın performansında düşmeler meydana gelebilir.

Belirsiz, tam olmayan bilgileri işleyebilmektedirler: Yapay sinir ağlarının belirli olmayan verileri işleyebilme kabiliyetleri vardır. Olayları öğrendikten sonra

belirsizlik altında ağlar öğrendikleri olaylar ile alakalı bağlantıları kurarak karar verebilirler.

Görülmemiş örnekler hakkında bilgi üretebilir: Ağ kendisine gösterilen örneklerden genellemelerde bulunarak görmediği örnekler hakkında bilgiler üretebilirler.

Örüntü tamamlama gerçekleştirebilirler: Bazı şartlarda ağa eksik bilgileri bulunduran bir örüntü veya bir biçim verilir. Ağın bu eksik bilgileri bulması istenir. Örneğin yırtılmış bir resmin kime ait olduğunu tespit etmesi ve tam resmi vermesi gibi bir görev ağdan istenebilir.

Sadece nümerik bilgiler ile çalışabilmektedirler: Yapay sinir ağları yalnızca sayısal bilgiler ile çalışırlar. Sembolik ifadeler ile gösterilen bilgilerin sayısal gösterime dönüştürülmeleri gerekmektedir (Yılmaz, 2012).

Biyolojik sistemlerde öğrenme, nöronlar arasındaki sinaptik (synaptic) irtibatların ayarlanmasıyla olur. İnsanlar doğduktan sonra yaşayarak öğrenme sürecine dahil olurlar. Bu süreçte beyin daima bir gelişme göstermektedir. Yaşayıp tecrübe ettikçe sinaptik bağlantılar ayarlanır ve yeni bağlantılar oluşur. Bu sayede öğrenme gerçekleşir. Bu durum YSA için de geçerlidir. Öğrenme, eğitme yöntemiyle örnekler kullanarak olur; yani, gerçekleşme girdi ve çıktı datalarının işlenmesiyle, eğitme algoritmasının bu dataları kullanarak bağlantı ağırlıklarını (weights of the synapses) yaklaşma sağlanana kadar, defalarca ayarlamasıyla olur (Yıldırım ve ark., 2014).

YSA'lar, ağırlıklandırılmış bir biçimde birbirleriyle ilişkilendirilmiş birtakım proses biriminden (nöronlar) meydana gelen sayısal sistemlerdir. Bir proses birimi, esasında sıklıkla transfer fonksiyonu olarak bilinen bir denklemdir. Bu proses birimi, diğer nöronlardan sinyalleri alır; ve bunları biraraya getirir, dönüştürür ve matematiksel bir netice meydana getirir. Genelde, proses birimleri kabaca reel nöronlara karşılıktır ve bir ağ içerisinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır (Kurt ve ark., 2011).

Sinirsel (neural) hesaplamanın merkezinde dağıtılmış, adaptif ve doğrusal olmayan proses kavramları bulunmaktadır. YSA'lar, alışılagelmiş işlemcilerden farklı bir biçimde proses yürütmektedirler. Alışılagelmiş işlemcilerde, bir tek merkezi proses birimi her bir hareketi sırayla gerçekleştirir. YSA'lar ise her biri büyük bir sorunun bir kısmıyla ilgilenen, fazla sayıda basit proses birimlerinden meydana gelmektedir. En basit biçimde, bir proses birimi, bir girdiyi bir ağırlık kümesiyle ağırlıklandırır, doğrusal olmayacak bir biçimde dönüşümüne olanak tanır ve çıktı değeri meydana getirir. İlk

görüşte, proses birimlerinin çalışma biçimi yanıltıcı biçimde basittir. Sinirsel hesaplamanın gücü, toplam proses yükünü paylaşan proses birimlerinin birbiri arasındaki yoğun bağlantı yapısından kaynaklanmaktadır. Bu tarz mekanizmalarda back-propagation yöntemiyle daha olumlu öğrenme elde edilmektedir (Karasulu ve ark., 2008).

YSA'ların genelinde, benzer özelliği barındıran nöronlar katmanlar halinde yapılandırılırlar ve transfer fonksiyonları ortak zamanlı bir biçimde yürütülürler. Yaklaşık olarak tüm ağlar, veri alan nöronları ve çıktı üreten nöronları barındırırlar (Gemici, 2015).

YSA'nın temel öğesi olan sayısal fonksiyon, ağın mimarisi tarafından biçimlendirilir. Daha net bir biçimde tarif edecek olursak, fonksiyonun ana yapısını ağırlıkların büyüklüğü ve proses elemanlarının proses biçimi belirler. YSA'ların davranışları, yani giriş verisini çıkış verisine nasıl bağlantıladıkları, ilk olarak nöronların transfer fonksiyonlarından, nasıl birbiriyle ilişkilendirildiklerinden ve bu ilişkilerin ağırlıklarından etkilenir (Gemici, 2015).

Yapay sinir ağı; insan beyninin sinir hücrelerinden meydana gelmiş tabakalı ve paralel olan yapısının bütün işlevleriyle birlikte matematiksel dünyada gerçeklenmeye çalışılan tasarlanmasıdır. Matematiksel dünya ile belirtilmek istenen donanım ve yazılımdır. Başka bir deyişle yapay sinir ağı hem donanımsal olarak hemde yazılım ile tasarlanabilir. Bundan dolayı, yapay sinir ağları ilk elektronik devreler desteğiyle kurulmaya uğraşılmş fakat bu girişim kendini gitgide yazılım alanına bırakmıştır. Bu denli bir kısıtlanmanın sebebi; elektronik devrelerin esnek ve dinamik bir biçimde değiştirilememesi ve birbirlerinden farklı olan birimlerin biraraya getirilememesi olarak sunulmaktadır (Altınışık, 2012).

Yazılım desteğiyle daha rahat kurulabilen yapay sinir ağları, yine yazılımsal bir biçimde çalıştırılabilmesi de kolay olabilecek tasarılardır. Fakat elektronik devrelerle kurulan yapay sinir ağı tasarıları haliyle yazılımla kurulan modellerle karşılaştırıldığında daha süratli olarak neticeye ulaşabilecektir. Bu yüzden, yapay sinir ağları şimdilerde yazılımsal olarak kurulup, yürütülüp, test edilmekte ve gerekli tüm değişiklikler ve dinamik güncellemeler yapılmakta, ardından neticelere göre karara varılmaktadır. Eğer elde edilen neticelerin başarısı 99%'larla tarif ediliyorsa, o halde uygun görüldüğü takdirde model elektronik devreler üzerine aktarılmaya çalışmaktadır. Bu sayede yapay sinir ağı modelleri, gerçek hayata uygulanmak üzere fiziksel bir platform üzerinde kullanılabilir hale getirilmiş olmaktadır (Altınışık, 2012).

2.2. Yapay Sinir Ağlarının Yapısı

İnsan beyni göz önünde bulundurularak oluşturan yapay sinir ağlarının yapısını çözümleyebilmek için insan beyninin hem yapısını hem de çalışma şeklini anlamak gerekmektedir (Yılmaz, 2012).

Nöronlar insanın beyin gücünü oluşturan temel yapıtaşlarıdır. Biyolojik nöronlar kesin bir sonuç ortaya koyabilmek için diğer kaynaklardan önemli bilgileri alır ve bu bilgileri birleştirerek elde edilen sonuç üzerinde işlem yaparlar. İnsan beyninin genel olarak çalışması üç aşamadan oluşur:

- Bilgi girişi
- Sentezleme ve karşılaştırma
- Bilgi çıkışı ve eylem

Beynin bu işlevleri yerine getirebilmesini sağlayan ise temel yapı elemanı olan sinir hücreleri yani nöronlardır. Sinir hücreleri birbirleri ile ilişki halindedirler (Yılmaz, 2012).

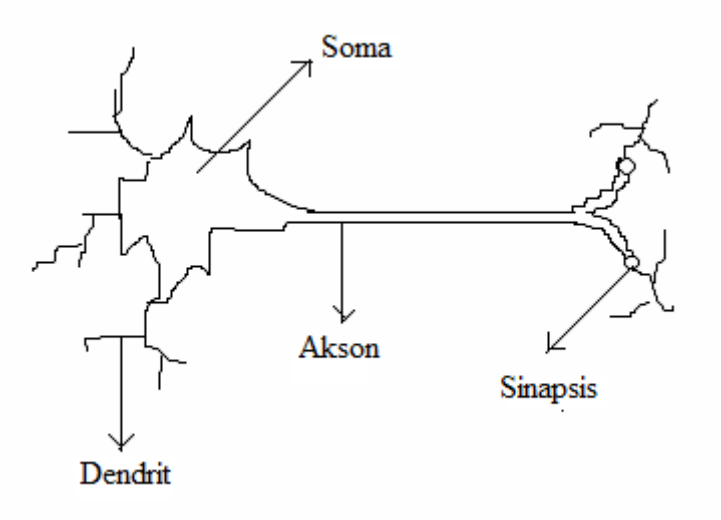
Yapay sinir ağının yapısını ve onu oluşturan elemanarı açıklayalım. Bu yapıyı anlayabilmek için öncelikle biyolojik sinir hücresinden bahsetmemiz gerekmektedir.

2.2.1. Biyolojik Sinir Hücresinin Yapısı

Tahminlere göre insan beyninde ki sinir hücresi 10 milyardır. Bu sinir hücrelerinin ise kendi aralarında kı bağlantı sayısı ise 60 trilyon olarak bilinmektedir.

Bu sinir hücrleri ise duyu organları aracılığıyla girdi bilgilerine ulaşırlar. Bir sonra ki aşama ise sinirlerin merkezi sinir sistemine ulaşmasıdır. Bunun içinde sinyallerin işlenip taşıyıcı sınırler tarafından diğer sinirlere aktarılmasıdır. Merkezi sinir sisteminin tepki sinyallerini üretebilmesi için işlenen sinyallerin yorumlanması gerekmektedir. Oluşan sinyallerin ilgili organa gönderilmesi gerekmektedir. Bu da tepki sinirleri aracılığıyla olur. Bu sayede duyu organlarından gelen bilgilere karşı tepki organlarına uygun işaretler sinir sistemi vasıtasıyla yollanır (Çayıroğlu, 2013).

Nöronlar biyolojik sinir sisteminin temel yapıtaşıdır ve bu nöronlar dört ana bölmden oluşur. Bu bölümler ise; dendrit, soma, akson, sinapsis.



Şekil 2.1. Biyolojik sinir hücresi (Fausett, 1994)

Dendrit: Dendritlerin amacı alınan sinyallerin nöronların çekirdeğine iletmektir. Dendritlerin görevi basit olarak bilinir ama günümüzde dendritler hakkında karmaşık bir yapıya sahip olduğuna dair söylemler bulunmaktdır. Hücrenin çekirdegi ile herbir dendrit arasinda farkli bir iletisim söz konusudur. Bu yüzden bazı dendritler etkileşim konusunda büyük bir paya sahiptir. Böylelikle diğer dendritler pasif kalır. Bu ise dışarıdan alınan sinyallerde seçicilik gibi önemli bir olgunun sinir hücresi tarafından gerçekleştirilmesi anlamını taşımaktadır.

Soma: Somalar dendritler tarafından gelen sinyalleri bir merkezde toplar. Somalar aslında biyolojik anlamda hücre çekirdeği (nükleus) olarak biliniyor. Çekirdek gelen sinyalleri aksona iletir. Bunun nedeni ise aksonun gelen sinyalleri diğer sinir hücrelerine gönderebilmesidir.

Akson: Aksonun görevi çekirdekten topladığı bütün sinyalleri diğer hücrelere iletmektir. Aksonun bir diğer görevi daha vardır. Bu da alınan sinyallerin ön hazırlık olmadan diğer hücrelere iletilmesine engel olur. Bunun sebebi ise birbiriyle bağlantısı olan sinapsise bilgilerini iletir.

Sinapsis: Sinapsis aksonlar tarafından gelen ilgileri dendritlere iletmeden önce ön hazırlıktan geçirir. Sinapsisin görevi çok büyük bir önem taşımaktadır. Çünkü ön işlem dediğimiz aşama gelen bütün sinyallerin belli bir eşik değerine denk getirilmesine göre değiştirilmesinden oluşur. Sinapsisin ön işlem ile gerçekleştirdiği görev çok önem taşımaktadır. Böylelikle toplam sinyal bulunduğu değerlerde olduğu gibi değil, belli bir eşik değerine getirilerek diğer sinir hücrelerine iletilir. Bu yönden alınan her sinyal ile dendrite gönderilen sinyallerle ilişki oluşturur. Buradan yola çıkılarak "öğrenme"

işleminin bu bilglerden yola çıkarsak öğrenme işleminin sinapsislerde gerçekleştirilen görevlerin fikri ortaya atılmış ve ortaya çıkan hipotez, gün geçtikçe yapay sinir ağı alanında bir teori haline dönüşmüştür. Günümüzde "öğrenme" yapay sinir ağı modelleri tarafından bu teoriye dayandırılır. Yani sinapsisler ve dendritler arasında yer alan ağırlık katsayılarının güncellenmesi olarak algılanmaktadır (Altınışık, 2012).

Yukarıda yer alan bilgilerde biyolojik olan sinir hücresinin elemanları anlatılmaktadır. Artık yapay sinir ağı hücre modeline inceleyebiliriz. Yapay sinir hücresi, biyolojik sinir hücreyle benzer esaslara dayandırılmaya çalışılmıştır (Fausett, 1994).

2.2.2. Yapay Sinir Hücresinin Yapısı

Yapay sinir hücreleri de biyolojik sinir hücrelerine benzeyen yapıdadır. Yapay nöronlar da kendi aralarında bağ oluşturarak yapay sinir ağlarını meydana getirirler. Biyolojik nöronlardaki gibi yapay nöronların da giriş sinyallerini aldıkları, bu sinyalleri biraraya getirip işledikleri ve çıkışları ilettikleri bölümleri vardır (Çayıroğlu, 2013).
Bir yapay sinir hücresi Şekil 2.2.'de görüldüğü gibi beş bölümden oluşmaktadır:

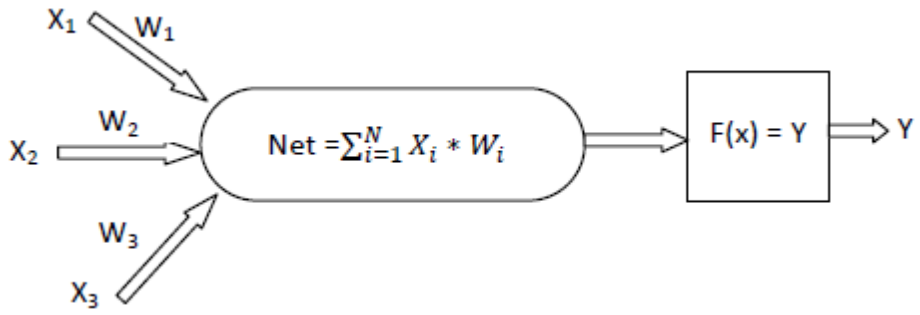
1. Girdiler

2. Ağırlıklar

3. Toplama Fonksiyonu (Birleştirme Fonksiyonu)

4. Aktivasyon fonksiyonu

5. Çıktılar



Şekil 2.2. Yapay sinir hücresinin yapısı (Çayıroğlu, 2013)

Girdiler: Girdiler nöronlara gelen verilerdir. Girdiler yapay sinir hücresine bir başka hücreden gelebileceği gibi doğrudan dış dünyadan da gelebilir. Bu girdilerden

gelen datalar biyolojik sinir hücrelerindeki gibi biraraya getirilmek için nöron çekirdeğine gönderilir.

Ağırlıklar: Yapay sinir hücresine gelen bilgiler girdiler üzerinden çekirdeğe gelmeden önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe gönderilir. Böylelikle girdilerin üretilecek çıktı üzerindeki tesiri konfigüre edilebilinmektedir. Bu ağırlıklar pozitif, negatif veya sıfır olabilir. Ağırlığı sıfır olan girdilerin çıktı üzerinde herhangi bir tesiri bulunmamaktadır.

Toplama Fonksiyonu (Birleştirme Fonksiyonu): Toplama fonksiyonu bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri biraraya getirerek o hücrenin net girdisini tespit eden bir işlevdir.

Bazı hallerde gelen girdilerin değeri önemli olmazken bazı durumlarda ise gelen girdilerin sayısı dikkate alınmaktadır. Bir problem için optimum toplama fonksiyonu tespit edilirken geliştirilmiş bir metot bulunmamaktadır. Genel olarak deneme yanılma yöntemiyle toplama fonksiyonu tespit edilmektedir. Bazen her hücrenin toplama fonksiyonunun benzer olması gerekli değildir. Bu konulara karar kılmak tasarımcının sorumluluğundadır. Çizelge 2.1.'de bazı toplama fonksiyonları gösterilmektedir.

Çizelge 2.1. Bazı toplama fonksiyonları (Çayıroğlu, 2013)

Toplam $Net = \sum_{i=1}^{N} X_i * W_i$	Ağırlık değerleri giriş değerleri ile çarpılır ve bulunan sonuçlar birbiriyle toplanarak Net giriş değeri hesaplanır.
Çarpım $Net = \prod_{i=1}^{N} X_i * W_i$	Ağırlık değerleri giriş değerleri ile çarpılır ve sonra bulunan sonuçlar birbiriyle çarpılarak Net giriş değeri hesaplanır.
Maximum $Net = Max(X_i * W_i)$	n adet giriş değeri arasından ağırlıklar giriş değeriyle çarpıldıktan sonra aralarından en büyüğü Net giriş değeri olarak varsayılır.
Minimum $Net = Min(X_i * W_i)$	n adet giriş değeri arasından ağırlıklar giriş değerleriyle çarpıldıktan sonra aralarından en küçüğü Net giriş değeri olarak varsayılır.
Çoğunluk $Net = \sum_{i=1}^{N} Sgn(X_i * W_i)$	n adet giriş değeri arasından giriş değerleriyle ağırlıklar çarpıldıktan sonra pozitif ve negatif olanların sayısı tespit edilir. Büyük olan sayı hücrenin Net giriş değeri olarak varsayılır.
Kumulatif Toplam $Net = Net(eski) + \sum_{i=1}^{N} X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan giriş değerleri eklenerek hücrenin Net giriş değeri bulunur.

Aktivasyon Fonksiyonu: Bu fonksiyon hücreye gelen net girdiyi ele alarak hücrenin bu girdiye yanıt olarak üreteceği çıktıyı belirler. Aktivasyon fonksiyonu genel olarak doğrusal olmayan bir fonksiyon olarak belirlenir. Yapay sinir ağlarının bir niteliği olan "doğrusal olmama" aktivasyon fonksiyonlarının doğrusal olmama niteliğinden gelmektedir. Aktivasyon fonksiyonu belirlenirken önemli olan bir diğer husus ise fonksiyonun türevinin rahat hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığından hesaplama hızının düşmemesi için türevi rahat hesaplanır bir fonksiyon seçilir. Günümüzde en yaygın olarak kullanılan "Çok katmanlı algılayıcı" modelinde genellikle aktivasyon fonksiyonu olarak "Sigmoid fonksiyonu" kullanılır. Çizelge 2.2.'de bazı aktivasyon fonksiyonları gösterilmiştir (Yayık, 2013).

Çizelge 2.2. Aktivasyon fonksiyonları (Çayıroğlu, 2013)

Doğrusal (Lineer) Aktivasyon Fonksiyonu	$\mathrm{F(Net)} = \mathrm{A} * \mathrm{Net}$ (A sabit bir sayı)	Doğrusal problemleri çözmek için aktivasyon fonksiyonu doğrusal bir fonksiyon olarak tercih edilebilir. Toplama fonksiyonunun neticesi, belli bir katsayı ile çarpılarak hücrenin çıkış değeri olarak bulunur.
Adım (Step) Aktivasyon Fonksiyonu	$\mathrm{F(Net)} = \begin{cases} 1 \; if \; Net > Eşik \; Değer \\ 0 \; if \; Net \leq Eşik \; Değer \end{cases}$	Gelen Net giriş değerlerinin belirlenen bir eşik değerin altında veya üstünde olmasına bağlı olarak hücrenin çıkış değeri 1 veya 0 olur.
Sigmoid Aktivasyon Fonksiyonu	$\mathrm{F(Net)} = \frac{1}{1 + e^{-Net}}$	Sigmoid aktivasyon fonksiyonu türevi alınabilir ve sürekli bir fonksiyondur. Doğrusal olmayışı sebebiyle yapay sinir ağır uygulamalarında en çok kullanılan fonksiyondur. Bu fonksiyon giriş değerlerinin her biri için 0 ve 1 aralığında bir değerler çıkarır.
Tanjant Hiperbolik Aktivasyon Fonksiyonu	$\mathrm{F(Net)} = \frac{e^{Net} + e^{-Net}}{e^{Net} - e^{-Net}}$	Tanjant hiperbolik fonksiyonu sigmoid fonksiyonuna benzer bir fonksiyondur. Sigmoid fonksiyonunda çıktı değerleri 0 ile 1 aralığında değişirken hiperbolik tanjant fonksiyonunun çıktı değerleri -1 ve 1 aralığında değerler almaktadır.
Eşik Değer Fonksiyonu	$\mathrm{F(Net)} = \begin{cases} 0 \; if \; Net \leq 0 \\ Net \; if \; 0 < Net < 1 \\ 1 \; if \; Net \geq 1 \end{cases}$	Gelen bilgilerin 0 dan küçük-eşit olduğunda 0 çıktısı, 1 den büyük-eşit olduğunda 1 çıktısı, 0 ve 1 aralığında olduğunda ise yine kendisini veren çıkış değerleri oluşturulabilir.
Sinüs Aktivasyon Fonksiyonu	$\mathrm{F(Net)} = \mathrm{Sin(Net)}$	Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği hallerde kullanılır.

Hücrenin Çıktısı: Aktivasyon fonksiyonundan çıkan değer, hücrenin çıkış değeri olmaktadır. Bu değer ya yapay sinir ağının çıktısı olarak dış dünyaya verilir ya da tekrardan ağın içinde kullanılabilir. Her hücrenin birden çok girdisi bulunmasına rağmen bir tek çıktısı olmaktadır. Bu çıktı istenilen miktarda hücreye bağlanabilir (Öztemel, 2003).

2.2.3. Yapay Sinir Ağının Yapısı

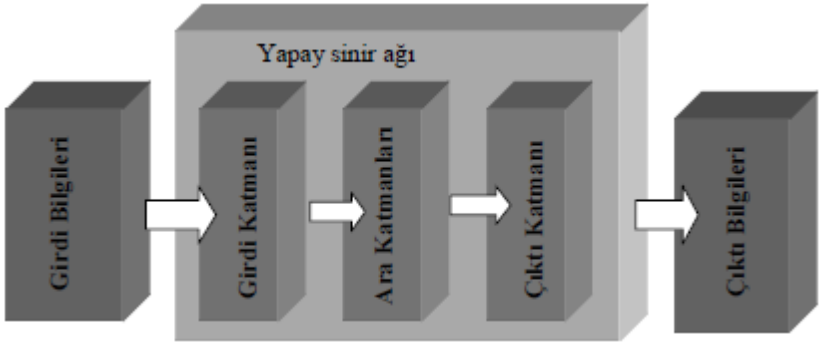
Yapay sinir ağları yapay sinir hücrelerinin birbirlerine ilişkilenmesiyle meydana gelen yapılardır. Yapay sinir ağları üç ana tabakada ele alınır; giriş katmanı, ara (gizli) katmanlar ve çıkış katmanı (Çayıroğlu, 2013).

Giriş Katmanı: Yapay sinir ağına dış dünyadan gelen girdilerin geldiği tabakadır. Bu tabakada dış dünyadan gelecek giriş sayısı kadar hücre bulunduğu halde genelde girdiler herhangi bir işleme maruz kalmadan alt tabakalara gönderilmektedir.

Ara (Gizli) Katman(lar): Giriş tabakasından çıkan veriler bu tabakaya gelir. Ara tabaka sayısı ağına göre değişebilir. Birtakım yapay sinir ağlarında ara tabaka bulunmadığı gibi birtakım yapay sinir ağlarında ise birden çok ara tabaka vardır. Ara tabakalardaki nöron sayıları giriş ve çıkış sayısına bağlı değildir. Birden çok ara tabaka bululan ağlarda ara tabakaların aralarındaki hücre sayıları da farklılık gösterebilir. Ara tabakaların ve bu tabakalardaki nöronların sayısının artış göstermesi hesaplama kompleksliğini ve süresini arttırdığı halde yapay sinir ağının daha kompleks problemlerin analizinde de kullanılabilmesine olanak tanır.

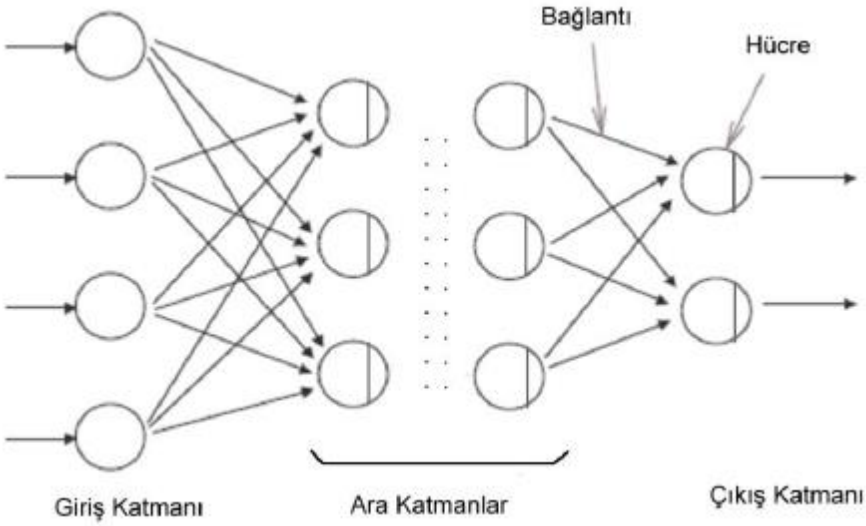
Çıkış Katmanı: Ara tabakalardan gelen verileri işleyerek ağın çıktılarını oluşturan tabakadır. Bu tabakada üretilen çıktılar dış dünyaya gönderilir. Geri beslemeli ağlarda bu tabakada üretilen çıktı kullanılarak ağın yeni ağırlık değerleri hesaplanır.

Şekil 2.3.'te yapay sinir ağında bulunan üç tabakanın birbiriyle olan bağlantısı gösterilmektedir.



Şekil 2.3. Yapay sinir ağı katmanlarının birbiri ile ilişkileri

Bu üç tabakanın her birisinde var olan işlem elemanları ve tabakalar arası ilişkiler şematik olarak Şekil 2.4.'te gösterilmektedir. Şekildeki yuvarlak elemanlar işlem elemanlarını temsil etmektedir. İşlem elemanlarını birbirine bağlayan çizgiler ise ağın ilişkilerini göstermektedir (Öztemel, 2003).



Şekil 2.4. Yapay sinir ağının yapısı (Çayıroğlu, 2013)

2.3. Yapay Sinir Ağlarının Sınıflandırılması

2.3.1. Yapılarına Göre Yapay Sinir Ağları

Yapay sinir ağları barındırdığı nöronların birbirlerine bağlanış biçimine göre ileri ve geri beslemeli olarak ikiye ayrılır (Çayıroğlu, 2013).

İleri Beslemeli Ağlar: İleri beslemeli ağlarda nöronlar girişten çıkışa doğru düzenli tabakalar biçimindedir. Bir tabakadan yalnızca kendinden sonraki tabakalara bağ vardır. Yapay sinir ağına gelen bilgiler giriş tabakasına daha sonra sırasıyla ara tabakadan ve çıkış tabakasından işlenerek geçer ve daha sonra dış dünyaya gönderilir.

Geri Beslemeli Ağlar: Geri beslemeli yapay sinir ağlarında ileri beslemeli olanların tersine bir hücrenin çıktısı yalnızca kendinden sonra gelen hücrenin tabakasına girdi olarak gönderilmez. Kendinden önceki tabakada veya kendi tabaksında var olan herhangi bir hücreye de girdi olarak bağlana bilmektedir.

Bu yapısı ile geri beslemeli yapay sinir ağlarında doğrusal biçimde olmayan dinamik biçimde bir davranış gözlenmektedir. Geri besleme özelliğini kazandıran ilişkilerin ilişkilenme şekline göre geri benzer yapay sinir ağıyla benzer olmayan davranışta ve yapıda geri beslemeli yapay sinir ağları elde edilebilmektedir (Başkurt ve ark., 2011).

2.3.2. Öğrenme Zamanına Göre Yapay Sinir Ağları

Yapay sinir ağları öğrenme zamanına göre statik ve dinamik öğrenme olmak üzere ikiye ayrılır (Çayıroğlu, 2013).

Statik Öğrenme: Statik öğrenme ilkesiyle çalışan yapay sinir ağları kullanmadan önce eğitim işlemine maruz kalmaktadır. Eğitimin tamamlanmasından sonra ağ istenilen biçimde kullanılabilmektedir. Fakat kullanım sırasında ağın üzerindeki ağırlıklarda hiçbir farklılık olmamaktadır.

Dinamik Öğrenme: Dinamik öğrenme kuralıysa yapay sinir ağlarının çalıştığı süre zarfında öğrenmesini önceden tahmin ederek modellenmiştir. Yapay sinir eğitim aşaması tamamlandıktan sonra da daha sonraki kullanımlarında çıkışların tasdik edilmesine göre ağırlıklarını farklılaştırarak çalışmaya devam etmektedir (Güney, 2009).

2.3.3. Öğrenme Algoritmalarına Göre Yapay Sinir Ağları

Yapay sinir ağlarının verilen girdilere göre çıktı sağlayabilmesinin yolu ağın öğrenmesidir. Öğrenme işleminin birden çok metotu vardır. Yapay sinir ağları öğrenme algoritmalarına göre danışmanlı, danışmansız ve takviyeli öğrenme olmak üzere üçe ayrılır (Çayıroğlu, 2013).

Danışmanlı Öğrenme: Danışmanlı öğrenmede bir öğretmene ihtiyaç bulunmaktadır. Öğretmen, bir veri alıştırma takımı veya ağın performansını ölçen bir gözlemci olabilmektedir. Danışmanlı öğrenmede, istenilen ağ çıkışının elde edilmesi için, çıkış hatasının azaltılmasında ağırlıkların adapte edilebilir hale getirilmesi gerekmektedir. Buna göre her giriş değeri için istenen çıkış sisteme gösterilir ve yapay sinir ağının giriş-çıkış bağlantısını oluşturana kadar kademe kademe ayarlanır. Danışmanlı öğrenmeye çok tabakalı perceptron, back-propagation, delta kuralı, Widrow-Hoff veya en küçük karelerin ortalaması ve adapte edilebilir doğrusal eleman anlamına gelen Adaline misal olarak gösterilebilir (Elmas, 2003).

Danışmanlı öğrenme esnasındaa ağa verilen giriş dataları için çıkış dataları da verilir. Ağ verilen girdiler için istenilen çıkışları meydana getirebilmek için kendi ağırlıklarını update eder. Ağın çıkışları ile beklenen çıkışlar arasındaki hata hesap edilerek ağın yeni ağırlıkları hata payına uygun olarak hazırlanır.

Hata payı hesaplanırken ağın beklenen çıkışları ile ağın bütün çıkışları arasındaki fark hesaplanır ve bu ayrıma uygun olarak her bir hücreye karşılık gelen hata

payı hesaplanır. Sonra her bir hücre kendine gelen ağırlıkları update eder (Çayıroğlu, 2013).

Danışmansız Öğrenme: Danışmansız öğrenmede öğretmen veya bir danışman, girişin ağın nerede iyi netice vereceğini veya hangi veri parçası grubuna dahil bulunduğunu söylemez. Burada dışarıdan herhangi bir müdahale bulunmamaktadır. Ağ datayı üyeleri birbirlerinin aynısı olan öbeklere yol gösterilmeden ayrıştırır. Danışmansız öğrenme danışmanlı öğrenmeye nazaran daha fazla süratlidir. Aynı zamanda matematik algoritmaları da daha kolaydır. Danışmansız öğrenmeye yarışmacı öğrenme, Hebbian öğrenme, Kohonen'in Grossberg öğrenme, öz örgütlemeli harita ağları vb. gibi öğrenme kuralları misal olarak gösterilebilir (Elmas, 2003).

Danışmasız öğrenmede ağa öğrenme esnasında yalnızca örnek girdiler verilmektedir. Beklenen çıktı bilgisi verilmemektedir. Girişte verilen bilgilere göre ağ her bir örneği kendi aralarında sınıflandıracak şekilde kendi ilkelerini meydana getirir. Ağ bağlantı ağırlıklarını benzer nitelikte olan dokuları ayırabilecek biçimde düzenleyerek öğrenme işlemini tamamlar.

Destekleyici Öğrenme: Bu öğrenme yaklaşımında ağın her epok neticesinde elde ettiği sonucun kötü veya iyi olup olmadığı hakkında bir bilgi verilir. Ağ bu bilgilere göre kendini tekrardan düzenler. Böylece ağ herhangi bir girdi dizisiyle hem netice elde ederek hem de öğrenerek çalışmaya devam eder.

2.4. Yapay Sinir Ağlarında Öğrenme Kuralları

YSA'da önemli öğrenme ilkelerinden bazıları aşağıdaki gibi örneklenerek anlatılmıştır (Elmas, 2003).

Hebb Kuralı: Şayet bir sinir başka bir sinirden girdi alırsa her ikiside çok aktif ise sinirler arasındaki kapsam güçlendirilir.

Hopfield Kuralı: Hebb ilkesi ile benzerlik bulundurmaktadır. Buna göre, şayet istenilen çıkış ve girişin her ikisi de pasif ya da her ikisi de aktif ise, bağlantı kapsamlarını öğrenme oranı kadar arttırır, aksi takdirde kapsamı öğrenme oranı kadar azaltır. Temel prensibi budur.

Delta Kuralı: En çok kullanılan ilkelerden birisi olan Delta ilkesi, Hebb ilkesinden daha ilerletilmişidir. Bu ilke bir sinirin istenilen çıkış değeri ile gerçek çıkışı arasındaki farkı düşürmek amacıyla giriş bağlantı kuvvetlerini devamlı olarak geliştirme prensibine dayanır. Bu ilke ağ hatasının karesini minimize etmek için bağlantı kapsamlarını değiştirir. Hata bir önceki tabakaya geri çoğaltılır. Bu tip ağ ileri besleme

ağ olarak adlandırılır, back-propagation ismini bu hata terimlerini toplama metotundan üretir.

Eğimli İniş Kuralı: Bu ilke delta ilkesine benzer çünkü transfer fonksiyonunun türevi bağlantı ağırlıklarında kullanılmadan önce, Delta hatasını gidermek için uygulanır. Bir ağın farklı tabakaları için öğrenme oranları, test işleminde çıkışa yakın olan tabakaların öğrenme oranından daha azdır. Giriş verilerinin güçlü bir tasarıdan çıkarılamadığı uygulamalarda, bu işlem ekstra önemlidir.

Kohonen Öğrenme Kuralı: Kohonen tarafından geliştirilen bu metotda biyolojik sinirlerdeki öğrenmeden ilham alınmıştır. Bu metotda sinirler öğrenmek amacıyla uygun ölçü ya da durumlarını update etmek için yarışırlar (Yılmaz, 2012).

2.5. Yapay Sinir Ağlarının Eğitilmesi ve Öğrenmesi

İnsan beyni, doğumdan sonraki olgunlaştığı zaman zarfında çevresinden duyu organlarıyla idrak ettiği davranışları yorumlar ve bu bilgileri diğer davranışlarında uygular. Yaşam sürecinde beyin gelişir ve deneyimlenir. Bundan böyle olaylar karşısında ne tepki vereceğini çoğu zaman bilmektedir. Ancak hiç karşılaşmadığı bir olay karşısında yine deneyimsiz olabilir. Yapay sinir ağlarının öğrenme zaman zarfında da dış dünyadan girişler alınır, aktivasyon fonksiyonundan geçirilerek bir tepki çıkışı oluşturulur. Bu çıkış yine deneyimle verilen çıkışla karşılaştırılarak hata tespit edilir (Çayıroğlu, 2013).

Farklı öğrenme algoritmalarıyla hata düşürülüp reel çıkışa yakınlaşılmaya gayret edilir. Bu çalışma sürecinde yenilenen yapay sinir ağının ağırlıklarıdır. Ağırlıklar her bir döngüde (cycle) yenilenerek hedefe ulaşılmaya gayret edilir. Hedefe ulaşmanın veya yakınlaşmanın ölçüsü de gene dışarıdan verilen bir değerdir. Şayet yapay sinir ağı verilen giriş-çıkış ikilisiyle hedefe ulaşmış ise ağırlık değerleri tutulur (Keleşoğlu ve ark., 2005).

Belli bir uygulamaya yönelik konfigüre edilen bir sinir ağı uygulanmadan önce eğitilmelidir. Bu evre, tecrübe yoluyla öğrenme niteliği için kritik değer taşımaktadır. Genelde, başlangıç ağırlıkları rastgele seçilir ve eğitme ya da diğer bir deyişle öğrenme işlemi bu işlemden sonra başlamaktadır (Yılmaz, 2012).

Yapay sinir ağlarında işlem ögelerinin ilişkilerinin ağırlık değerlerinin tespit edilmesi işlemine ağın eğitilmesi denir. Başlangıçta ağırlık değerleri rassal olarak atanır. Yapay sinir ağları kendilerine örnekler verildikçe bu ağırlık değerlerini değiştirirler. Hedef ağa verilen örnekler için doğru çıkışları oluşturacak ağırlık değerlerini

belirlemektir. Örnekler ağa tekrar tekrar verilere en doğru ağırlık değerleri bulunmaya gayret edilir. Ağın doğru ağırlık değerlerine varması örneklerin simgelediği olaya dair genellemeler yapabilme kabiliyetine ulaşması anlamına gelir. Bu genelleme kabiliyetine ulaşması işlemine ağın öğrenmesi denir. Ağırlıkların değerlerinin değişmesi belli ilkelere göre yürütülmektedir. Bu ilkelere öğrenme ilkeleri denir (Öztemel, 2003).

Ağırlıkların devamlı yenilenerek arzu edilen neticeye varılana kadar geçen süreye öğrenme ismi verilir. Yapay sinir ağı öğrendikten sonra daha önce gösterilmeyen girişler gösterilip, sinir ağı çıkışıyla gerçek çıkış yaklaşımı gözlemlenir. Şayet yeni gösterilen örneklere de doğru yaklaşıyorsa sinir ağı işi öğrenmiş anlamına gelir. Sinir ağına gösterilen örnek sayısı en uygun değerden daha çok ise sinir ağı işi öğrenmemiş, ezberlemiş anlamına gelir. Genel olarak elde bulunan örneklerin %80'i ağa verilip ağ eğitilir. Daha sonra geriye kalan %20'lik kısım ağa gösterilip ağın tutumu gözlemlenir. Bu sayede ağın testi yapılmış olmaktadır (Öztemel, 2003).

1. Örneklerin toplanması: Ağın öğrenmesi arzu edilen olay için daha önce elde edilmiş örneklerin bulunması aşamasıdır. Ağın eğitilmesi için örnekler toplandığı gibi (eğitim seti) ağın test edilmesi için de örneklerin (test seti) toplanması gerekir. Eğitim setindeki örnekler sırayla ağa verilerek ağın olayı öğrenmesi sağlanır. Ağ olayı öğrendikten sonra test setindeki örnekler ağa verilerek ağın performansı test edilir. Daha önce hiç ağa verilmeyen örnekler karşısındaki performansı ağın iyi öğrenip öğrenmediğini gösterir.

2. Ağın topolojik yapısının belirlenmesi: Öğrenilmesi arzu edilen olay için oluşturulacak olan ağın topolojik yapısı tespit edilir. Kaç tane girdi birimi, kaç tane ara tabaka, her ara tabakada kaç tane işlem elemanı, kaç tane çıktı elemanı olması gerektiği bu adımda tespit edilmektedir.

3. Öğrenme parametrelerinin belirlenmesi: Ağın öğrenme katsayısı, işlem elemanlarının aktivasyon ve toplama fonksiyonları, momentum katsayısı gibi değişkenler bu adımda tespit edilmektedir.

4. Ağırlıkların başlangıç değerlerinin atanması: İşlem elemanlarını birbirine bağlayan ağırlık değerlerinin ve eşik değer biriminin ağırlıklarının başlangıç değerlerinin atanması gerçekleştirilir. Başlangıçta genel olarak rassal değerler atanır. Daha sonra ağ elverişli değerleri öğrenme esnasında kendisi tespit eder.

5. Öğrenme setinden örneklerin seçilmesi ve ağa gösterilmesi: Ağın öğrenmeye başlaması ve öğrenme ilkesine uyumlu olarak ağırlıkları değiştirmesi için ağa örnekler belli bir sisteme göre verilir.

6. Öğrenme sırasında ileri hesaplamaların yapılması: Verilen girdi için ağın çıktı değerleri hesaplanır.

7. Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması: Ağın oluşturduğu hata değerleri bu aşamada hesaplanır.

8. Ağırlıkların değiştirilmesi: Geri hesaplama metotu kullanılarak oluşturulan hatanın düşürülmesi için ağırlıklar değiştirilir.

9. Öğrenmenin tamamlanması: İleri beslemeli sinir ağı öğrenmeyi bitirinceye, yani elde edilen çıktılar ile istenilen çıktılar arasındaki hatalar uygun seviye gelinceye kadar sürer (Çayıroğlu, 2013).

2.6. Yapay Sinir Ağlarının Uygulama Alanları

Yapay sinir ağları temel olarak; tasarlama, tahmin ve sınıflandırma gibi birçok alanda kullanıyor. Yüksek başarılar elde etmiş çalışmalar incelendiğinde, yapay sinir ağlarının kesin olmayan, gürültülü, eksik, çok boyutlu, hata ihtimali yüksek sensor verilerinin olması, karmaşık, kusurlu ve problem çözmek için matemetiksel tasarıların ve sayısal değerlerin bulunmadığı, yalnıca örneklerin var olduğu durumlarda yaygın olarak kullanıldıkları gözlenmiştir. Bu yüzden geliştirilmiş ağlar çoğunlukla şu işlevleri gerçekleştirmektedirler (Öztemel, 2003);

Veri sıkıştırma, olası fonksiyon kestirimleri, sınıflandırma, ilişkilendirme veya, örüntü tanıma, örüntü eşleştirme, doğrusal olmayan sinyal işleme, sinyal filtreleme, doğrusal olmayan sistem modelleme, zaman serileri analizleri, optimizasyon, kontrol (Çayıroğlu, 2013).

YSA'lar pek çok sektörde değişik uygulama alanları bulmuştur. Bunlardan bazıları;

Uzay: Otomatik pilot uygulamaları, komponentlerin hata denetimleri, uçuş simülasyonları vs.

Otomotiv: Rehber, garanti, otomatik yol izleme, yol koşullarına göre sürüş analizi, garanti aktivite analizi vs.

Bankacılık: Müşteri analizi, kredi uygulamaları geliştirilmesi, kredi müraacat değerlendirilmesi, bütçe yatırım tahminleri vs.

Savunma: Hedef seçme, silah yönlendirme, sensör sonar sistemleri, radar, görüntü işleme, sinyal işleme vs.

Elektronik: Çip bozulma analizi, kod sırası öngörüsü, non-lineer modelleme vs.

Eğlence: Özel efektler, animasyonlar, pazarlama öngörüsü vs.

Finans: Pazar performans analizi, kıymet biçme, hedef belirleme, bütçe kestirimi vs.

Sigortacılık: Uygulama politikası geliştirme, ürün optimizasyonu, vs.

Üretim: Ürün dizaynı, üretim işlem kontrolü, dayanıklılık analizi, makina yıpranmalarının tespiti, iş çizelgeleri hazırlanması, kalite kontrolü vs.

Sağlık: EEG, EKG, MR, göğüs kanseri erken teşhis ve tedavisi, ilaç etkileri analizi, kalite artırımı, kalp krizi erken teşhis ve tedavisi, kan analizi sınıflandırma vs.

Petro kimya: Verim analizi, arama vs.

Robotik: Forklift robotları, yörünge kontrol, uzaktan kumandalı sistemler, görsel sistemler, optimum rota belirleme vs.

Dil: Yazı ve konuşma çevrimi, sözcük tanıma, dil tercüme vs.

Telekomünikasyon: Filtreleme, görüntü ve data karşılaştırma, ses ve görüntü işleme, eko ve gürültü sönümlendirilmesi, trafik yoğunluğunun kontrolü ve anahtarlama vs.

Güvenlik: Kredi kartı hileleri saptama, parmak izi tanıma, yüz eşleştirme, retina tarama vs.

Bu verilen örnekler arttırılabilir. Görmüş olduğumuz gibi YSA'lar günlük yaşantımızda farkında bile olmadığımız birçok alanda kullanılmaktadır. Hergün gün geçtikçe çalışma alanları çoğalmakta ve gelişmektedir (Çayıroğlu, 2013).

2.7. Yapay Sinir Ağlarının Avantaj ve Dezavantajları

2.7.1. Yapay Sinir Ağlarının Avantajları

YSA'lar, uygulanan ağ tasarımına göre değişik yapısal özellikler göstermelerine rağmen temel birkaç ortak niteliğe sahiptirler (Çayıroğlu, 2013).

1. YSA'lar birçok nörondan meydana gelir ve bu nöronlar aynı zamanlı olarak uyumlu çalışarak çözülmesi zor fonksiyonları yerine getirir. Başka bir söyleyişle de kompleks fonksiyonlar bir çok nöronun aynı zamanlı çalışması ile ortaya çıkar. Zaman içerisinde bu nöronlardan sıradan biri fonksiyonunu kaybetse bile sistem hala güvenilir bir şekilde çalışmasına ara vermeden devam edebilir.

2. Eğitim sırasında kullanılan sayısal bilgilerden, problemin nerdeyse tüm özellikleri toplaması ve böylelikle eğitim sırasında kullanılmayan bilgiler için de, mantıklı cevaplar üretebilmesidir.

3. Yapı üzerinde belli oranda dağılmış belli şekildeki non-lineer alt birimler, non lineer sorunların da çözümüne olanak tanımaktadır.

4. YSA'lar makina öğrenmesi yapabilirler. Yapay sinir ağlarının temel fonksiyonu hali hazırda bilgisayarın öğrenmesini sağlamaktır. Olayları öğrenerek karşılaşabileceğimiz benzer sorunlar karşısında daha mantıklı ve gerçekçi kararlar verebilirler.

5. Bilgi işleme metotları geleneksel programlamadan daha farklı ve karmaşıktır. Bu yüzden geleneksel programlamanın beraberinde getirdiği birçok olumsuzluklar yok edilebilir.

6. Veriler ağın bütününde saklanır. Geleneksel programlamada görüldüğü gibi bilgiler veri tabanları ya da dosyalarda belirli bir şekilde tutulmaz, bu bilgiler ağın tamamına dağıtılarak belirli değerler ile ölçülen ağ bağlantılarında tutulmaktadır. Hücrelerden bir kısmının görevini kaybetmesi, anlamlı bilginin yitirilmesine sebep olmaz.

7. Dağıtık hafızaya sahiptirler. YSA'larda tüm bilgi ağa dağılmış bir biçimde saklanır. Hücrelerin birbiri ile bağlantı ve ağırlık durumları, ağın bilgisini gösterir. Bu yüzden tek bir bağlantının tek başına anlamı bulunmamaktadır.

8. Örnekleri kullanarak kavrarlar. YSA'nın kavrayabilmesi için örneklerin belirlenmesi, bu örneklerin ise ağa aktarılarak istenen çıktılara uygun olarak ağın organize edilmesi gerekmektedir. Ağın başarısı, öne sürülen örnekler ile paraleldir, ağa bu olay her yönü ile aktarılamaz ise ağ da yanlış çıktılar ortaya çıkabilir.

9. Daha önce karşılaşılmamış örnekler hakkında bilgiler verebilirler. YSA'lar eğitimleri esnasında kendilerine gösterilen örneklerden genellemeler ortaya koyarlar ve bu genellemeler ile yeni örneklerle ilgili olarak yeni bilgiler verebilirler.

10. Algılamayı hedef alan olaylarda kullanılabilirler. YSA'ların başarı göstermiş oldukları alanlar, algılamayı hedef alan kullanım alanlarıdır. Bu alanlarda başarılarını ispatlamıştır.

11. Aynı zamanda örüntü (pattern) ilişkilendirme ve sınıflandırma gibi şeyler yapabilirler. YSA'lar kendilerine örnekler halinde gösterilen örüntüleri kendisiyle veya diğerleriyle bağlantılayabilir. Buna ek olarak kendisine gösterilen örneklerin kümelenmesi ile, bir sonraki verinin hangi kümeye ait olacağına karar kılınması hususunda kullanılabilirler.

12. Örüntü tamamlama da yapabilirler. Ağa eksik verilere sahip olan örüntüler verildiğinde eksik verilerin giderilmesi hususunda performansları iyidir.

13. Kendi kendilerine öğrenebilme ve organize olma (self organizing) kabiliyetlerine sahiptir. YSA'lar çevrimiçi bir biçimde öğrenebilirler ve kendilerini eğitebilme yeteniğine sahiptir.

14. Eksik veri ile çalışabilmektedirler. Geleneksel sistemlerin tam tersine YSA'lar belli bir eğitimden sonra veriler eksik bilgiye sahip olsa bile bilgi oluşturabilirler. Bu durum bir güç kaybı oluşturmaz, güç kaybı eksik bilginin değeriyle bağlantılıdır. Burada bilgilerin değeri eğitim esnasında öğrenilir.

15. Hata payına sahiptirler. YSA'ların eksik verilerle çalışabilmeleri ve bazı hücreleri arızalansa bile çalışabilmeleri, onları oluşabilecek hatalara karşı hoşgörülü yapar.

16. Bazı durumlarda dereceli bozulma (graceful degradation) sergilerler. Bir ağ, belli bir süreçte yavaş ve bağıl bir bozulmaya uğrar. Ağlar sorunların meydana geldiği anda hemen bozulmazlar.

2.7.2. Yapay Sinir Ağlarının Dezavantajları

YSA'ların, pek çok avantajının olduğu gibi bazı dezavantajları da vardır. Belli başlı dezavantajları (Çayıroğlu, 2013);

1. Donanım bağımlıdır. YSA'ların en önemli problemlerinden birisi donanım bağımsız olmamalarıdır. YSA'ların en önemli nitelikleri ve var olma sebeplerinden biri olan paralel işlem yapabilme kabiliyeti, paralel çalışan prosesör ile performans verir.

2. Elverişli ağ yapısının belirlenmesinde belirli bir ilke bulunmamaktadır. YSA'larda probleme göre elverişli ağ yapısının belirlenmesi için geliştirilmiş bir ilke bulunmamaktadır. Elverişli ağ yapısı tecrübe ve sınama yöntemi ile belirlenmektedir.

3. Ağın değişken değerlerinin tespit edilmesin de belirli bir ilke bulunmamaktadır. YSA'larda öğrenme hücre sayısı, katsayısı, katman sayısı gibi değişkenlerin tespit edilmesinde belirli bir ilke bulunmamaktadır. Bu değerlerin tespit edilmesi için belirli bir temel ölçüt bulunmamakla beraber her sorun için farklı bir ele alış biçimi söz konusu olabilmektedir.

4. Öğrenilecek problemin ağa verilmesi önemli bir sorundur. YSA'lar sayısal bilgiler ile çalışabilmektedirler. Problemler YSA'lara gösterilmeden önce sayısal değerlere çevirilmek mecburiyetindedirler. Burada belirlenecek gösterim düzeneği ağın performansını direkt olarak etkileyecektir. Bu da kullanıcının kabileyetine bağlıdır.

5. Ağın eğitiminin ne zaman tamamlanması gerektiğine dair belirli bir metot bulunmamaktadır. Ağın örnekler üzerindeki hatasının belirli bir değerin altına

indirgenmesi eğitimin bitirildiği anlamı taşımaktadır. Burada en uygun sonuçlar veren bir düzenek halen bulunmamaktadır ve YSA'ya dair gözlemlerin önemli bir boyutunu oluşturmaktadır.

6. Ağın tutumları açıklanamamaktadır. Bu problem YSA'ların en önemli sorunudur. YSA bir sorunu analiz ettiği zaman, bunun niçin ve ne şekilde olduğuna dair bir ipucu vermez. Bu hal ağa olan güveni azaltıcı bir etmendir.

2.7.3. Geleneksel Algoritmalar ile Yapay Sinir Ağlarının Karşılaştırılması

Çizelge 2.3.'te geleneksel algoritmalar ile yapay sinir ağlarının kıyaslanması gösterilmiştir (Çayıroğlu, 2013).

Çizelge 2.3. Geleneksel algoritmalar ile yapay sinir ağlarının kıyaslanması (Çayıroğlu, 2013)

Geleneksel Algoritmalar	**Yapay Sinir Ağları**
Çıkışlar, koyulan kuralların girişlerde uygulanmasıyla elde edilir.	Öğrenme sürecinde giriş-çıkış bilgileri verilerek, kurallar koyulur.
Hesaplama; merkezi, eş zamanlı ve ardışıldır.	Hesaplama; toplu, eş zamansız ve öğrenmeden sonra paraleldir.
Bellek paketlenmiş ve hazır bilgi depolanmıştır.	Bellek ayrılmış ve ağa yayılmıştır.
Hata toleransı yoktur.	Hata toleransı vardır.
Nispeten hızlıdır.	Yavaş ve donanıma bağımlıdır.
Bilgiler ve algoritmalar kesindir.	Deneyimden yararlanır.

2.8. Bölüm Değerlendirmesi

Bu bölümde yapay sinir ağlarının tanımından, yapısından, sınıflandırılmasından bahsedilmiştir. Ayrıca yapılardan bahsederken biyolojik ve sinir hücrelerinin de yapısı da ele alınmıştır. Yapay sinir ağlarında öğrenme kuralları, yapay sinir ağlarının eğitilmesi ve öğrenmesi, tarihçesi, uygulama alanları, avantaj ve dezavantajlarıda incelenmiştir. Son olarak geleneksel algoritmalar ve yapay sinir ağları karşılaştırılmıştır.

3. İLGİLİ ÇALIŞMALAR

Yapay sinir ağları için farklı programlama dillerinde birkaç kütüphane ve araç geliştirilmiştir. Bu kısımda, yaygın biçimde uygulanan ve genel olarak kabul gören; FANN, SNNS, JOONE ve Matlab Neural Network Toolbox tetkik edilmiştir (Kınacı, 2006).

3.1. SNNS

Yapay sinir ağları için Stuttgart Üniversitesi'nde IPVR enstitüsü aracılığıyla geliştirilmiş bir simülatördür Stuttgart Neural Network Simulator (SNNS). Yapay sinir ağının kullanımları ve incelemeleri için aktif ve farklılıklara elverişli bir simülasyon ortamı gerçekleşitirmek için oluşturulmuştur (Kınacı, 2006).

SNNS 4 temel bileşenden meydana gelir:

1. Simülasyon çekirdeği
2. Grafiksel arayüz
3. Toplu çalıştırma arayüzü
4. Ağ derleyicisi: snns2c

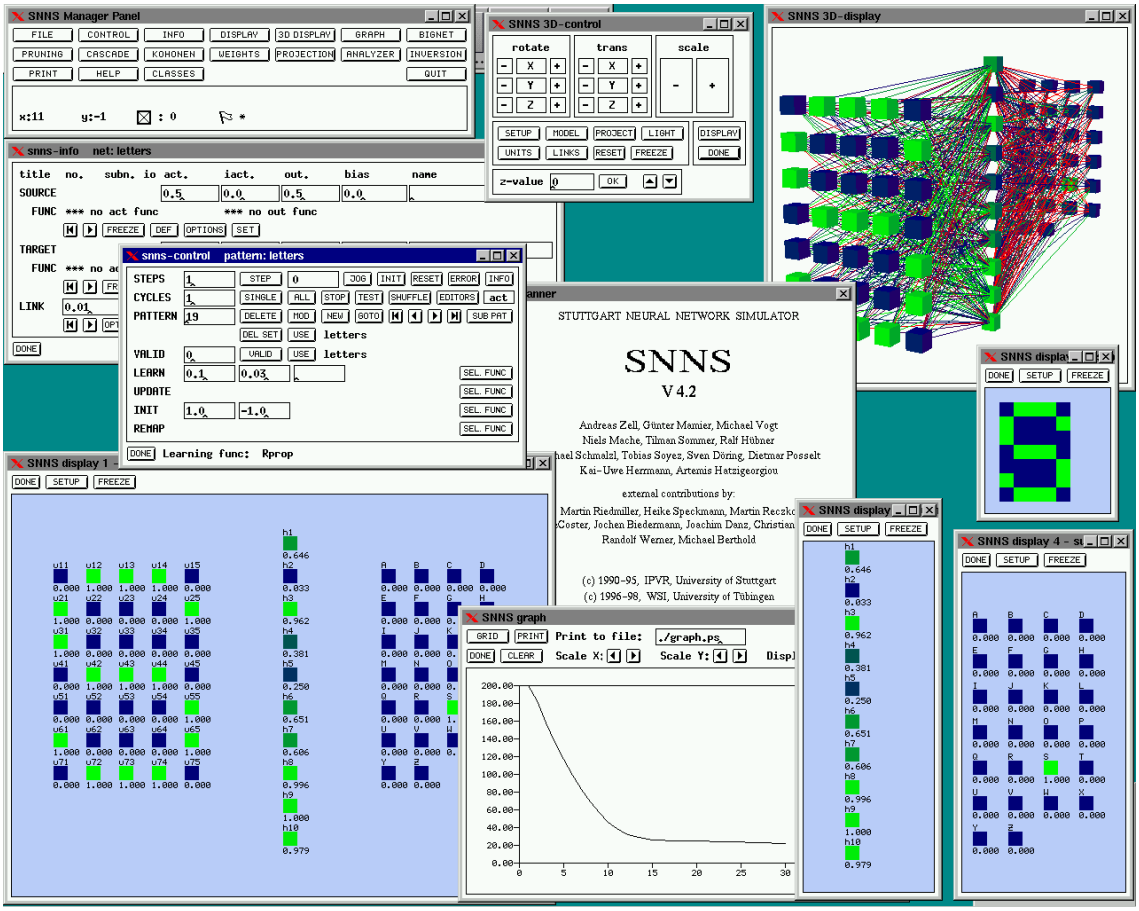
Simülasyon çekirdeği, sinir ağları yapısı ile ilgili olarak çalışır ve ilgili bütün işleri yapar. Ağ derleyicisi *snn2c* desteğiyle eğitilmiş ağlar C koduna dönüştürülerek programlarda gömülü çalışması sağlanmaktadır. Kullanıcı tanımlı olan çıktı fonksiyonlarının, aktivasyon fonksiyonlarının ve öğrenme algoritmalarının kolay C programları biçiminde yazılıp simülator çekirdeğine ilişkilendirilmeleri mümkün olmaktadır. SNNS'nin yardım sağladığı yapay sinir ağı yapıları ve öğrenme algoritmaları aşağıdaki gibidir (Kınacı, 2006):

- Momentumlu, Çevrim içi (on-line) ve Yığın Geriye Yayılma
- Quickprop, Counterpropagation, Rprop, Backpercolation 1
- ART1, ART2, ARTMAP, RBF
- Recurrent Cascade Correlation, Cascade Correlation
- Dinamik LVQ
- Quickprop through time, Backpropagation through time
- SOM
- TDNN (time-delay neural networks) back-propagation ile
- Elman ağları, Jordan ağları, Associative Memory

Grafiksel arayüz XGUI, çekirdek üzerinde çalışır. Sinir ağlarının simulasyon sırasından kontrol edilmelerine ve grafiksel gösterimine yardımcı olur. Grafiksel arayüz,

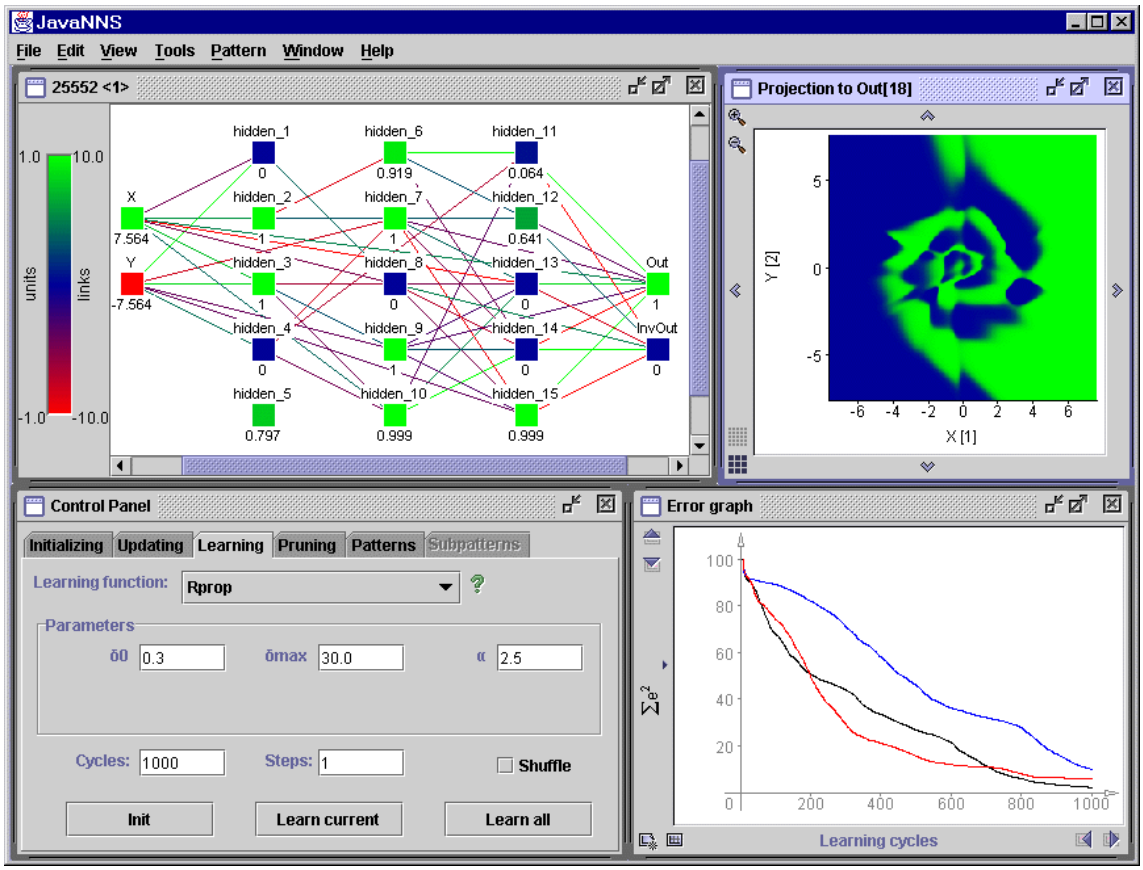
sinir ağlarının doğrudan yaratılmasını, değiştirilmesini ve görselleştirilmesini destekler. Kompleks ağlar süratli ve basit bir biçimde oluşturulabilir (Kınacı, 2006).

SNNS bütünüyle ANSI-C'de geliştirilmiştir. Simülasyon çekirdeği birtakım işletim sisteminde ve makinede yürütülebilmektedir. Grafiksel arayüz X11, XGUI pencere mekanizmasında yürütülmektedir. Şekil 3.1.'de XGUI gösterilmiştir.



Şekil 3.1. SNNS grafiksel arayüzü XGUI (Zell)

SNNS'nin grafiksel arayüzü daha sonraları Java ortamı ile tekrar yazılmıştır ve JavaNNS biçiminde adlandırılmıştır. JavaNNS direkt anlaşıldığı gibi bütünüyle simülatör ortamının Java'da oluşturulması değildir. JavaNNS'de simülatör çekirdeği tekrardan yazılmadan olduğu gibi kullanılmıştır. Şekil 3.2.'de JavaNNS gösterilmiştir.



Şekil 3.2. JavaNNS (Zell)

Bir örnekle anlatacak olursak, kolay bir karakter tanıma programı modellendiği düşünülürse, giriş olarak verilen bir karakterin resmini analiz edecek mekanizma için eğitilecek ağ, Latin alfabesinden meydana gelen bir eğitim veri kümesiyle eğitilmiş olsun. Şayet bu yazılım kullanıcı tarafından el yazısı için veya değişik bir harf kümesinin karakterleri için uygulanmak arzu edilirse bu halde SNNS içinden tekrar ağın eğitilmesi ve yazılıma ilave edilmesi gerekmektedir. Bu yapay sinir ağı bilinen alfabeler için genişletilip eğitilebilir. Fakat özellikle el yazısı tanıma kabiliyetine sahip bir yazılım arzu ediliyorsa mutlaka user kısmında eğitilebilir bir yazılıma ihtiyaç duyulacaktır. SNNS benzeri bir durumda elverişsiz kalmaktadır. Öğrenebilen yazılımlar geliştirmek amacıyla yazılım içerisinde bütünüyle çalışır bir yapay sinir ağı kütüphanesinin kullanılması zorunluluğu bulunmaktadır (Kınacı, 2006).

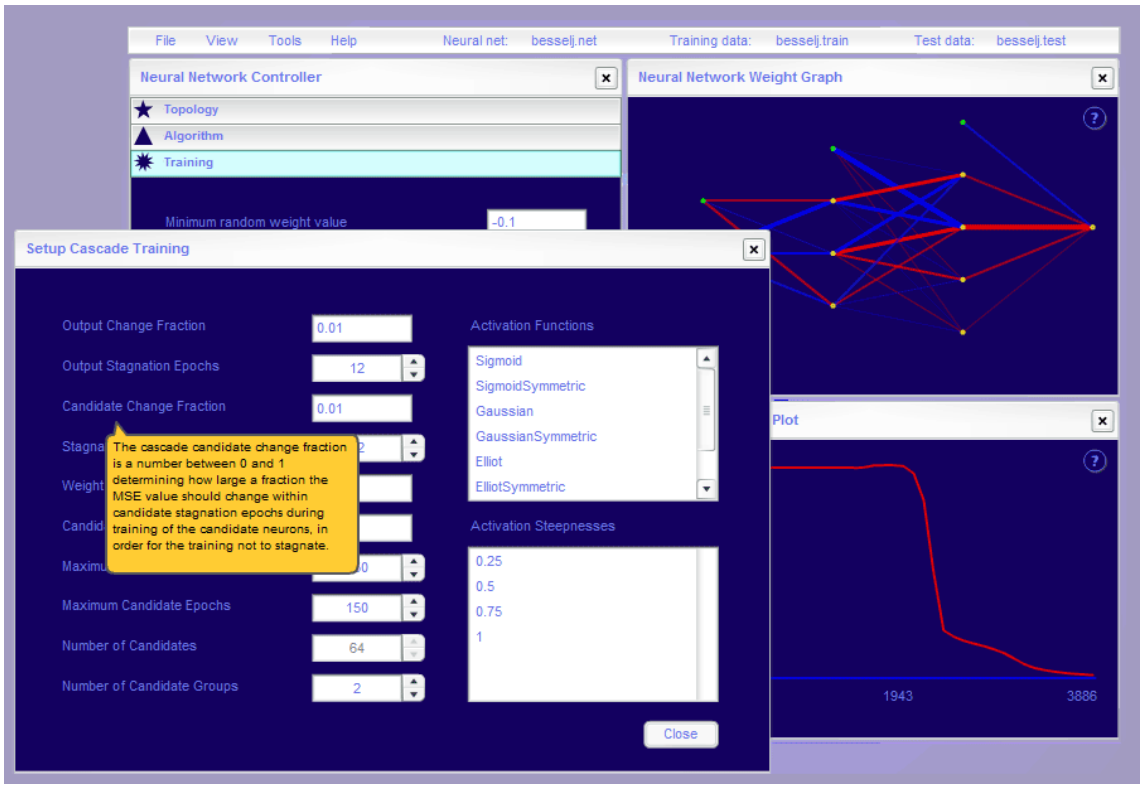
3.2. FANN

FANN (Fast Artificial Neural Network Library), çok tabakalı reseptörleri kullanmaya yardımcı olan, C programlama dilinde yazılmış bir yapay sinir ağı kütüphanesidir. Buna ek olarak birtakım programlama dillerinden, Perl, C++, Python

gibi, ulaşılabilmesine yardımcı olan bağlayıcı arabirimleri bulunmaktadır (Kınacı, 2006).

Kütüphanenin destek sağladığı öğrenme algoritmaları aşağıdadır:

- Back-propagation algoritması
- Arttırmalı back-propagation algoritması
- Quickprop
- RPROP
- Stack öğrenmeli back-propagation algoritması



Şekil 3.3. FANN (Anonim)

Yapay sinir ağlarının kullanılmasıyla alakalı sorunlardan bir tanesi uygulamadaki performans problemleri olmuştur. Bu yönden FANN, sinir ağlarının süratli çalışmasına olanak tanımak amacını göz önünde bulundurarak geliştirilmiştir. Destek sağladığı sinir ağı tasarıları ve öğrenme algoritmaları yönünden yalnızca feed-forward ağlarla Back-propagation algoritmalarına destek olması, fakat sınırlı sayıda problem çeşidi için kullanabilmesine imkan tanır. Şekil 3.3.'te FANN'ın grafik arayüzü gösterilmiştir (Kınacı, 2006).

3.3. JOONE

JOONE (Java Object Oriented Neural Engine), Java programlama dilinde geliştirilmiş yapay sinir ağı kütüphanesidir. Modüler bir mimariye sahiptir. Çekirdek bölümü görsel arayüzden hariç olarak geliştirilmiştir. Kod tasarımı açısından yeni sinir ağı modelleri veya yeni öğrenme algoritmaları eklenebilecek bir mimariye sahiptir. Genetik algoritmalar yardımıyla yapay sinir ağı iyileştirilmesi imkanlar dahilindedir. Joone çekirdek bölümü, çoklu kullanım (multi-threaded) niteliğine sahiptir. Buna ek olarak yapay sinir ağının eğitimi için dağıtık (distrubuted) biçimde çalışabilecek ortam da sunulmaktadır.

Çekirdek motoru (core engine) şeklinde belirtilen bölümün destek sağladığı nitelşkler şunlardır (Kınacı, 2006):

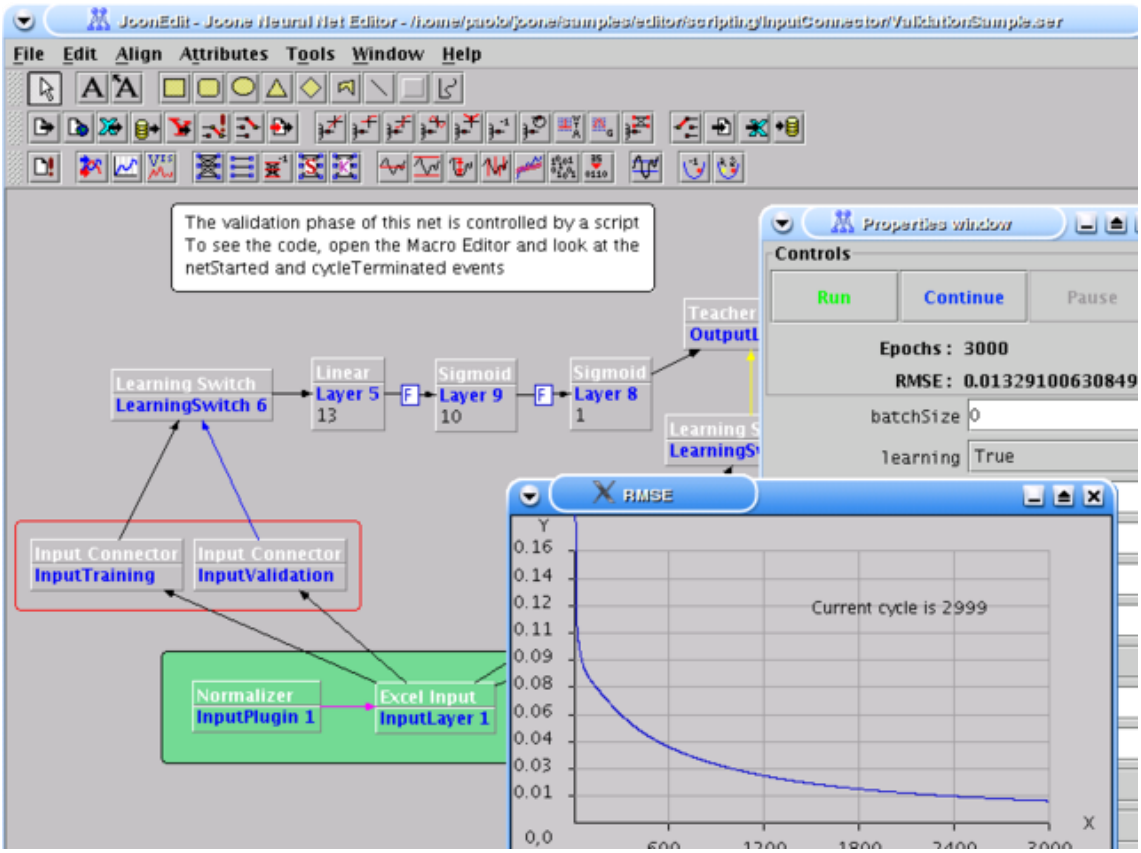
- Eğiticili öğrenme algoritmaları
- Back-propagation
- Stack back-propagation
- RPROP
- Kohonen SOM ve Principal Component Analysis
- Yapay sinir ağlarının serialization mantığıyla dosyada tutulabilmesi
- Veri okumak amacıyla girdi/çıktı bileşenleri
- Virgül ayırıcı kullanılmış ASCII basit .txt dosyaları
- Excel dosyaları
- Veritabanı ilişkilendirmesi
- Yapay sinir ağının öğrenmesini denetleyen ve ağ ile alakalı değişkenleri değiştirmeyi sağlayan bileşenler
- Giriş datasını ön işlemden (pre-processing) geçirmeye sağlayan uzantılar

JOONE çekirdeğini kullanan grafik temelli düzenleyici vardır. Düzenleyicinin grafiksel arayüzü Şekil 3.4.'teki gibidir. Bu araçla yapay sinir ağlarının oluşturulması, eğitilmesi ve görselleştirilmesi mümkün olmaktadır. Bu aracın nitelikleri şunlardır (Kınacı, 2006):

- Çekirdeğin destek olduğu her çeşit tasarı oluşturabilme
- Yapay sinir ağı elemanlarının değişkenlerinin değiştirilebilmesi
- Yapay sinir ağının dosyada tutulması ve dosyadan tekrar yüklenmesi
- Yapay sinir ağı eğitme
- Ön işlem (pre-processing) ve denetleme uzantılarının kullanılabilmesi
- Makro editörü

• Yapay sinir ağının çıkış değerlerinin görsel hale getirilmesi

• Yapay sinir ağının doğruluğunu test etme

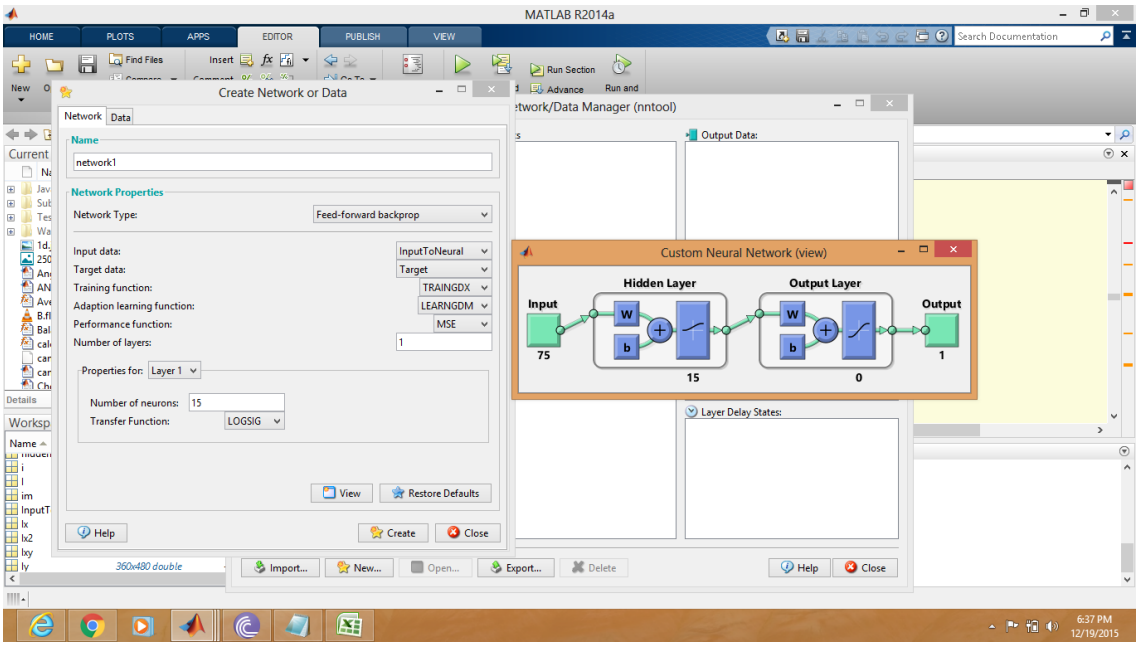


Şekil 3.4. JOONE grafiksel temelli düzenleyicisi (Pmarrone)

3.4. Matlab Neural Network Toolbox (NNTOOL)

Matlab, bir programlama dili ve bir matematiksel hesaplama ortamıdır. Fonksiyon ve veri çizimi, matris işlemleri, algoritma gerçekleştirimi gibi işlemleri yapabilmeye yardımcı olur. Araç kutusu (toolbox) denilen paketler ile kontrol tasarımı, görüntü işleme, ses işleme vb. gibi birçok alanda özgün çözümler sunar.

Matlab neural network toolbox, yapay sinir ağlarının modellenmesine, görselleştirilmesine, simulasyonuna ve gerçekleştirilmesine yardımcı olan, Matlab için geliştirilmiş bir araçtır. Genel olarak kabul edilmiş birçok yapay sinir ağı algoritması ve modeli için detaylı bir geliştirme ortamı sağlar. Şekil 3.5.'teki grafiksel kullanıcı arayüzü ile yapay sinir ağlarının modellenmesine ve test edilmesine yardımcı olur (Kınacı, 2006).



Şekil 3.5. Matlab Neural Network Toolbox kullanıcı (user) arabirimi

Danışmanlı (supervised) ve danışmansız (unsupervised) öğrenen yapay sinir ağı çeşitlerine destek olur. Danışmanlı ağlardan tabanlı, ileri beslemeli (feed-forward), radyal, dinamik ve LVQ ağlarına destek olur. İleri beslemeli (feed-forward) ağlar için geriye yayılma (Back-propagation) algoritması ve türevlerini, radyal tabanlı yapay sinir ağlarından genelleştirilmiş regresyon ve olasılıksal yapay sinir ağlarını (probabilistic neural network), dinamik ağlardan Elman, Hopfield, NARX, focused time-delay ve dağıtık zaman geciktiren (distributed time-delay) ağlara destek olur. Danışmanlı yapay sinir ağlarından rekabete dayalı ve SOM tipinde yapay sinir ağlarına destek olur. Girişlerin ilk, çıkışların son işlemden geçmesine olanak tanır (Kınacı, 2006).

Matlab, teknik programlama amacıyla kullanılan bir geliştirme ortamı olması sebebiyle yazılım geliştirme ortamı şeklinde değerlendirmek pek mümkün değildir. Programlama dili olarak günümüzde yazılım geliştirme dillerine göre kabiliyetleri epey kısıtlı bir dildir. Yapay sinir ağlarına destek olduğu halde bu aracın yazılım geliştirme esnasında kullanılması oldukça zordur. Buna ek olarak ücretli bir yazılım olması sebebiyle son kullanıcıya yönelik program geliştirmek oldukça maliyetlidir.

3.5. Bölüm Değerlendirmesi

Bu bölümde tezde gerçekleştirilen yapay sinir ağları yazılımı için destek sağlayabilecek ve örnek teşkil edebilecek farklı programlama dilleri ile geliştirilmiş olan yapay sinir ağları kütüphaneleri ve araçları incelenmiştir. Bunlar bu zamana kadar

kullanılmış hala yaygın olarak kullanılmakta olan ve genel olarak kabul görmüş olan SNNS, FANN, JOONE ve Matlab Neural Network Toolbox (NNTOOL)'tır.

4. KAYNAK ARAŞTIRMASI

4.1. Yapay Siniri Ağları Yöntemleri ile Başağrısı Hastalıklarının Sınıflanadırılması

Yapay Sinir Ağları (YSA) üzerine yapılan çalışmalar gittikçe yoğunlaşmakta ve yapay sinir ağlarının uygulama alanları gün geçtikçe artmaktadır. Hastalıkların teşhisinde yapay zeka yöntemlerinin yanında yapay sinir ağları da kullanılmaya başlamıştır. Başağrısı hastalıklarının teşhisinde doktorlar, bilgilerine, tecrübelerine ve laboratuvar sonuçlarına bakarak karar verebilmektedirler. Doktor ne kadar konusunda uzman da olsa bazen bazı noktalar gözden kaçmakta ve yanlış teşhisler konulabilmektedir. Teşhis problemlerinde insan faktörünü ortadan kaldırmak için bilgisayar ortamında çözüm üretilmesi için bugüne kadar yurt içinde ve yurt dışında birçok çalışma yapılmıştır (Doğan ve ark., 2004).

Başağrısı hastalıkları 13 ana başlık altında toplanmıştır. Bu hastalıklar kendi aralarında toplam 127 alt gruba ayrılmıştır. Bunun dışında başağrısı hastalıkları iki temel gruba ayrılabilir. İlk grup, nöroloji bilim dalını doğrudan ilgilendiren ve başka bir hastalık veya tıbbî durumla ilgili olmayan primer (birincil) başağrıları şeklinde isimlendirilen başağrılarıdır. Bu gruba giren hastalıklar migren, gerilim tipi başağrıları ve küme tipi başağrılarıdır. İkinci grup başağrıları ise darbeler, göz hastalıkları, kulak burun boğaz hastalıkları, boyun hastalıkları, beyin kanaması, beyin tümörü gibi tıbbî durumlar sebebiyle oluşan ve sekonder (ikincil) başağrıları şeklinde isimlendirilen başağrılarıdır (Doğan ve ark., 2004).

Bu bildiride en sık rastlanan başağrısı hastalıkları olan auralı migren, aurasız migren ve gerilim tipi başağrısı hastalıklarının teşhisi için yapay sinir ağları algoritmalarından perseptron, kendini düzenleyen ağlar ve geri yayılım algoritması ile birer çözüm önerilmiş ve bu çözümlerin verimlilikleri karşılaştırılmıştır.Test örneğindeki ortalama yanlış sayıları Şekil 4.1.'de verilmiştir (Doğan ve ark., 2004).

Algoritma	Maksimum Epok Sayıları		
	100	1000	10000
Kümelerin yokuş inmesi	7,8	3,9	2,3
Küm. mom. yokuş inm.	7,4	4,9	3,0
Değişken öğrenme	4,5	3,0	2,8
Mom. değişken öğr.	3,7	2,9	3,1
Esnek geri yayılım alg.	2,4	2,2	2,4
Fletcher-Reeves alg.	3,2	2,2	
Polak-Ribiére alg.	1,9	3,4	
Powell-Beale alg.	2,4	2,3	
Kad. çekim eğimi alg.	2,6	2,9	
BFGS algoritması	3,2	2,8	
Tek adım sekant alg.	2,4	2,2	
Levenberg-Marquardt	3,5		

Şekil 4.1. Test örneğindeki ortalama yanlış sayıları

Bu çalışmada başağrısı hastalıklarının teşhisi için yapay sinir ağları metotlarından perseptron, kendini düzenleyen ağlar ve geri yayılım algoritmaları denenmiştir. Geri yayılım algoritmaları arasında en hızlı çalışan algoritmanın Levenberg-Marquardt algoritması olduğu gözlenmiştir. 19 test örneğinden ortalama 3,5 örneği yanlış teşhis etmiş olmasına rağmen diğer geri yayılım algoritmalarıyla kıyaslandığında Levenberg-Marquardt algoritmasının tercih edilebilir bir algoritma olduğu tespit edilmiştir. Fakat geri yayılım algoritmalarının en verimli olanı bile perseptronun sağladığı hızı ve güvenirliği verememiştir. Perspteron, en basit yapay sinir ağı modeli olmasına rağmen başağrısı hastalıklarının teşhisinde çok etkili olmuştur. Bunda, belirtiler uzayının doğrusal olarak ayrılabilir olmasının payının büyük olduğu düşünülmektedir. Kendini düzenleyen ağlar ise perseptrona yakın başarı ve hız sağlamıştır (Doğan ve ark., 2004).

4.2. Troid Bezi Bozukluklarının Yapay Sinir Ağları ile Teşhisi

Son zamanlarda bileşim teknolojisindeki gelişmelere bağlı olarak bilgisayar destekli tıbbi karar verme uygulamaları ile ilgili çalışmalarda artış görülmektedir. Bu çalışmada, yapay sinir ağlarının troid bezi hastalığı verilerini kullanarak yapılan teşhislere yardımcı olabileceği gösterilmiştir. Troid bezi bozukluğu hastalığının risk faktörlerinden olan 4 parametre, yapay sinir ağlarının girişi olarak kullanılmıştır. Quasi-Newton, Lewenberg-Marquardt ve standart geri yayılım algoritmaları ile eğitilen çok katmanlı yapay sinir ağlarının sınıflama sonuçları karşılaştırılmıştır (Şahin ve ark., 2004).

Modern bilgi ve iletişim teknolojileri, sağlık hizmetlerinin niteliğini, etkinliğini ve verimliliğini büyük ölçüde arttırabilmekte, bu hizmetleri, standartlaştırılmış, ölçülebilir ve karşılaştırılabilir biçimde vermeyi sağlayabilmektedirler. Büyük bir hızla artan tıp bilgisi ve buna paralel olarak çoğalan ve gelişen ölçü ve görüntüleme yöntemleri, giderek otomatikleşen tıbbi test, analiz ve görüntüleme cihazları, bireyler ve hastalar için toplanılan tıbbi verileri de büyük bir hızla arttırmaktadır. Bilgi ve iletişim teknolojilerinin kullanımı ile hızlı ve etkin bir tedavi görmek, en iyi sağlık hizmetinin, yaşanılan yere en yakın ve en çabuk olarak verilmesi, sağlık-hastalık-tedavi bilgilerine kişilerin de kolaylıkla ulaşabilmesi mümkün olmaktadır. Bilgi ve iletişim teknolojilerinin en yoğun kullanıldığı alanların başında sağlık bilimleri ve tıp sektörü gelmektedir. Sağlık bilgi sistemleri konusunda dünya, standartlarla uyumlu, hastayı odak alan, internet tabanlı, elektronik iletişimi, sayısal kimlik belirleme yöntemlerini destekleyen, verileri yaşam boyu saklayan, kalite kriterlerinin ölçümünü, sağlık bilgilerinin güvenliğini ve kişisel gizliliğini, tıbbi kararlara desteği sağlayan bilgi sistemlerine yönelmektedir (Şahin ve ark., 2004).

Yapay sinir ağları ile medikal alanda teşhis ve tahmine dayalı çalışmalar gün geçtikçe artmaktadır. Özellikle bazı tıbbi problemlerde teşhisler doktor tecrübesine dayanmaktadır. Bu durumda doktorlara yardımcı olmak ve bazı hastalıkların teşhisini kolaylaştırmak amacıyla yapay sinir ağları ile çeşitli çözümler sunulmaktadır. Bu çalışmada troid bezi bozuklukları hastalığına ait veri kümesinden yararlanılarak hastalığın kişide yarattığı boyutlar sınıflandırılmıştır . Bu amaçla çalışmada, troid bezi fonksiyonları ile ilgili olarak Çukurova Üniversitesi Tıp Fakültesi Balcalı Hastanesi Endokrinoloji Anabilim dalındaki 502 troid bezi bozukluğu hastasından toplanan verilerin eğiticili yöntemlerle sınıflandırma konusu incelenecek ve sonuçlar karşılaştırılacaktır. Troid bezi fonksiyonlarını ve anatomik yapısının değerlendirmesi için yapılan 5 ayrı test sonucunda hastalar fonksiyonel olarak normal, hipotirodi ve hipertiroidi, anatomik olarak da nodüler guvatr, multi noduler guvatr, toksik multi noduler guvatr, toksik diffüz guvatr, basit diffüz guvatr ve toksik adenom şeklinde toplam 8 sınıfta toplanacaktı (Şahin ve ark., 2004).

Troid bezi fonksiyonlarını ve anatomisini değerlendirmek amacı ile hastalara uygulanan 5 test sırasıyla verilmiştir.

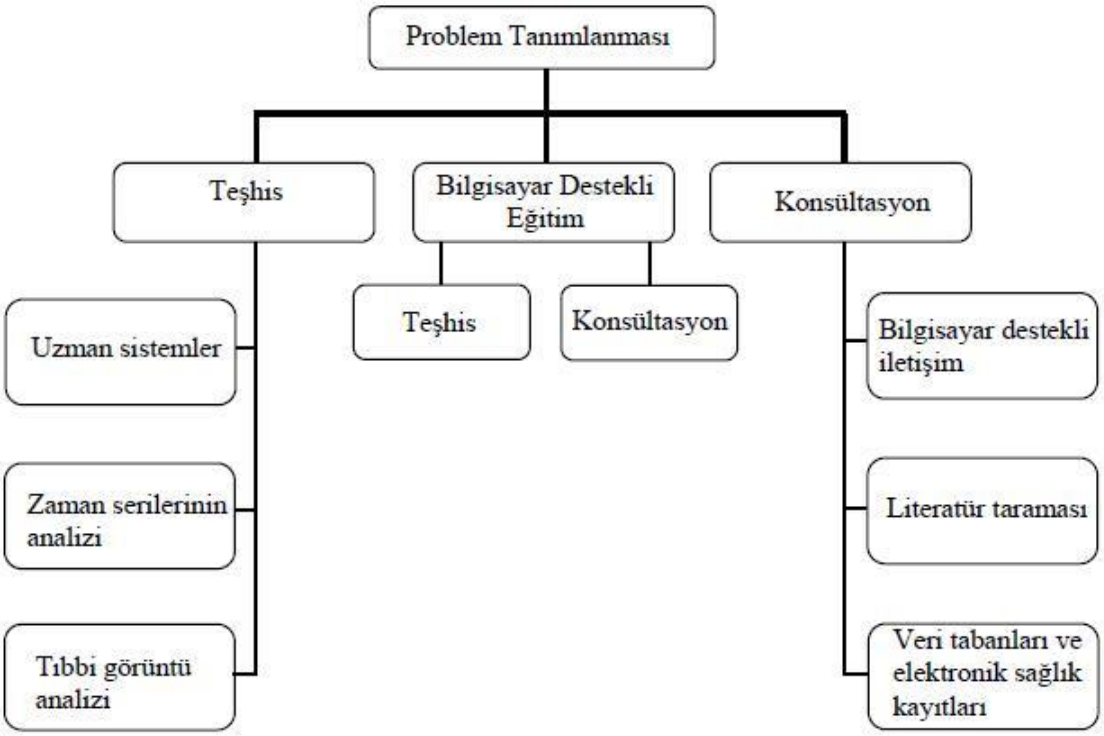
1-Isotopik gösterge metoduyla ölçülen toplam serum thyroxin

2-Radioimmuno deneyi ile ölçülen toplam serum triiodothyronine

3-Radioimmuno deneyi ile ölçülen basal thyroid-stimulating hormone (TSH)

4-Troid USG

5-Tiroid sintigrafisi



Şekil 4.2. Tıbbi karar destek sistemi

Tıbbi karar destek sistemi Şekil 4.2.'de gösterilmiştir. Bu çalışmada, çok katmanlı sinir ağı değişik algoritmalar ile Çukurova Üniversitesi Tıp Fakültesi Balcalı Hastanesi Endokrinoloji Anabilim dalından alınan troid bezi bozukluğu hastaları verileri üzerinde uygulanmıştır. Ağın eğitimi için kullanılan standart geri yayılım, Quasi-Newton, Lewenberg-Marquardt algoritmaları ile eğitim ve test verisi üzerinde elde edilen sonuçlar karşılaştırılmıştır. Karşılaştırma sonucunda çok katmanlı sinir ağında kullanılan 3 algoritma ile troid bezi hastalıkları verileri kullanarak yapılan hastalık teşhisine yardımcı olabileceği gösterilmiştir (Şahin ve ark., 2004).

4.3. Ateroskleroz'un Tahmini İçin Bir Yapay Sinir Ağı

Ateroskleroz'un, özellikle sanayi ülkelerinde mortalite ve morbiditenin önde gelen nedenlerinden olduğu belirlenmiştir. Ateroskleroz (ATS), yaşamın erken dönemlerinde arteryel duvarda yağlı çizgilenme olarak başlayan patolojik bir süreçtir. ATS, en çok koroner arter hastalığı (KAH) olarak ortaya çıkmaktadır. Batı toplumlarında KAH, en belirgin morbidite ve mortalite nedeni olmakla beraber son

yirmi yıl içinde, ATS'nin kardiyovasküler etkilerinin azalması nedeniyle morbidite ve mortalite oranları üçte bir oranında azalmıştır; ancak dünya nüfusunun yarısını oluşturan Asya toplumlarında tedrici bir artış göstermektedir. ATS, KAH gibi kalp hastalıklarının tahmin edilip gelişiminin engellenmesi çok önemlidir. Hastalıkların tahmin ve teşhisinde istatistik yöntemlerden biri olan yapay sinir ağları (YSA) kullanılmaktadır. YSA, elle çözüm imkanı vermeyen son derece karmaşık yapıya sahip, insan beynindeki sinir ağları gibi çalışarak en zor problemlerin çözümünü sağlayan ve değişken yapısına ilişkin herhangi bir kısıtlama getirmeksizin değişkenler arasındaki ilişkiyi ortaya koyan esnek bir yöntemdir. YSA, nöronlar olarak adlandırılan basit hesap hücrelerinin birbirleriyle olan bağlantılarını kullanarak insan beyninde olduğu gibi bilgiyi kaybetmeden hedefe doğru en az hata ile ulaşılmasını sağlar. ATS'nin tahmin ve teşhisinde YSA kullanılabilmekle beraber bu konuda yapılan çalışmalar da bulunmaktadır (Çolak, 2005).

Çalışmada YSA analizi iki aşamada gerçekleştirilmiştir. Birinci aşamada toplam verilerin %60'ı eğitim icin kullanılmıştır. Eğitim verilerine ait sonuçlara göre, eğitim verisinde YSA, ATS hastası olmayanların tamamını doğru olarak tahmin etmiş iken, ATS hastalarının %66'sını doğru olarak tahmin edebilmiştir. Toplam ayrımsama oranı ise %86.6 olmuştur. İkinci aşamada YSA, test için ayrılan veriler üzerinde uygulanmıştır. Bu analize ait bilgiler Şekil 4.3.'te bulunmaktadır. Test verisine ait sonuçlara göre YSA gibi ATS hastalarının %50'sini tahmin edebilmiş, ATS hastası olmayanların ise tamamını doğru olarak tahmin etmiştir. Toplam ayrımsama oranı ise %80 bulunmuştur (Çolak, 2005).

			Tahmin			
			ATS			
			Hasta	*Normal*	*Toplam*	*Yüzde*
Gözlem	ATS	Hasta	2	2	4	50
		Normal	0	6	6	100
		Toplam	2	8	10	80

ATS: Ateroskleroz

Şekil 4.3. Test Verisine Ait YSA Sınıflama Bilgileri

Çalışmada, incelenen olaya ait klinik parametrelerin zor elde edilmesi ile reoperasyonlar, kombine kapak-koroner bypass cerrahisi ve aort cerrahisi yapılan

hastaların çalışma dışı bırakılmasına ek olarak zaman ve maliyet gibi unsurların da etkisi nedeniyle küçük sayıda bir örnek üzerinde çalışılmıştır. Az sayıdaki örnek üzerinde yapılan analizde, YSA'nın test verilerinde toplam ayrımsama oranı %80, eğitim verisinde ise %86.6 olmuştur. Bu sonuçlar göz önüne alındığı zaman, YSA ATS'nin tahmin edilmesinde kullanılabilir. Ancak örnek sayısının az olması göz önünde bulundurulduğunda, daha tutarlı tahminler ve daha güvenilir sonuçlar elde edebilmek için örnek sayısının artırılması gereklidir. Yapılacak olan YSA çalışmalarında örnek sayısının olabildiğince büyük tutulması ile daha yararlı sonuçlar elde edilebilir. YSA ile ATS'nin teşhis ve tahmini yapılabilmekle beraber hekimlere yararlı bilgiler de sunulabilmektedir. Kotel'nikova ve ark., ATS'nin tahmin edilebilmesi için bir YSA oluşturmuşlardır. Sonuçta, YSA ile yapılan teşhisin doktor tarafından yapılan teşhisten 1.5-3 kat daha doğru olduğunu bildirmişlerdir. Rebrova ve ark., ATS'nin teşhisi için bir YSA geliştirmişler ve YSA'nın duyarlılığını % 97 olarak bulmuşlardır. George ve ark., koroner ATS'nin derecesini tahminlemek için bir YSA modeli elde etmişlerdir. Ulaştıkları sonuçlara dayanarak daha büyük bir kitlede YSA'nın kullanımı ile ATS'nin derecesinin invaziv olmayan bir şekilde tahmin edilebileceğini ifade etmişlerdir. Lapuerta ve ark., "the Cholesterol Lowering Atherosclerosis Study" adlı çalışmanın verilerini kullanarak bir YSA modeli oluşturmuşlardır. Klinik çıktıların tahmin edilmesinde YSA'nın performansının Cox regresyonundan daha iyi olduğunu bulmuşlardır. Bu ve sözü edilen diğer çalışmaların sonuçlarına dayanarak, YSA'nın ATS gibi kalp hastalıklarının tahmin edilmesinde umut verici bir yöntem olabileceği ifade edilebilir (Çolak, 2005).

Sonuç olarak, geliştirilen YSA'nın toplam ayrımsama oranı % 80 olarak bulunmuştur. YSA'nın Ateroskleroz hastalığının tahmin edilmesinde oldukça yararlı olacağı sonucuna varılabilir. Ancak örnek sayısının az olması göz önünde bulundurulduğunda, daha güvenilir sonuçlar elde edebilmek için örnek sayısının artırılması önerilebilir.

4.4. Yapay Sinir Ağları ile Ağız Kokusundan Diyabet Tanısı

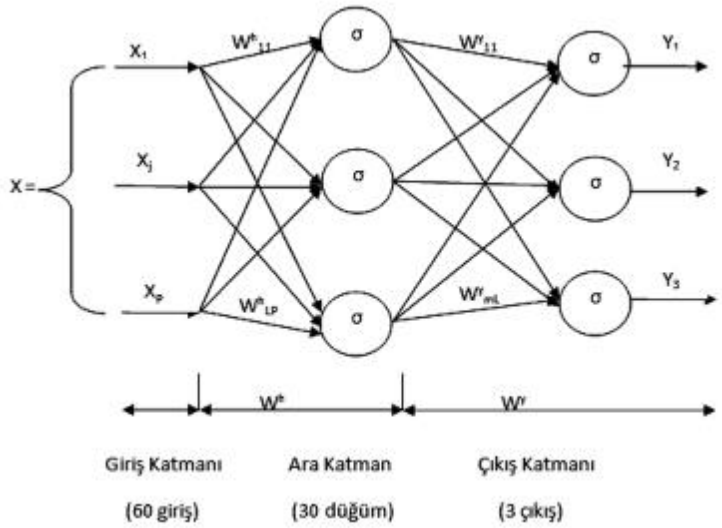
İnsan vücudunun ürettiği ter, kan, idrar ve gaita gibi parametreler, hastalıkların tanısı için kullanılabilmektedir. İnsan nefesinin de vücudun ürettiği diğer parametreler gibi vücut sağlığı hakkında önemli bilgiler taşıdığı bilinmektedir. Bu kapsamda insan nefesinden çıkan gazların incelenmesiyle bazı hastalıkların tanısı konulabilmektedir (Karlık ve Cemel, 2012).

Akciğerlerde kanın temizlenmesi esnasında kandaki gazlar alveoller vasıtasıyla nefese geçmektedir. Dolayısıyla nefeste vücutla ilgili pek çok parametre mevcuttur. Nefes yoluyla dışarı verilen gazlar çeşitli alkalik ve aromatik bileşenlerden oluşmaktadır. Bu bileşenlerin her biri hastalık bilgisini de ifade edebilmektedir. Sağlıklı bir insan nefesinde bulunan gazlar ve bunların oranları belirlenmiştir. Şayet insan hastalanırsa, hastalanma sebebi ile ilgili olarak nefeste bulunan gazların oranları değişmektedir. Bu da vücutta oluşan bir hastalığın belirtisidir ve nefesten bu gazların oranlarının değişimi tanımlanabilirse hastalığın tanısı konulabilir (Karlık ve Cemel, 2012).

İnsan nefesinde en çok görülen gazların sayısı 200 ile 400 arasında değişmektedir. Ayrıca nefeste algılanmış ve tanımlanmış gaz sayısı 3000'i geçmektedir. Elektronik Burun teknolojisinden yararlanılarak bu gazlar algılanabilmektedir. Son yıllarda elektronik burun kullanılarak kokudan hastalık tanısı çalışmaları olmuştur (Karlık ve Cemel, 2012).

Bu çalışmada, FIS firmasının SB-41 kodlu sensor dizisi ve diyabet hastalarından alınan koku verisi (aseton kokusu) kullanılarak değişik yapay sinir ağları ile sınıflandırma sonucu tip 1 ve tip 2 şeker hastalığı tanısı konulmuştur. SB-41 Sensör yardımı ile insan nefesindeki koku buharı moloküllerine ait ölçülen fiziksel değerler sayısal verilere dönüştürülmüştür. On saniyelik periyotlarla ölçümler yapılmış ve hasta başına 60 adet veri toplanmıştır. 'Yirmisi hastalıklı, 10'u sağlıklı toplam 30 kişiye ait verinin yarısı eğitme diğer yarısı da test için kullanılmıştır. Sayısal değerlere dönüştürülen bu veriler derlenip MatLab ortamında oluşturulan üç değişik Yapay Sinir Ağında (MLP, LVQ ve RBF) eğitilmiştir (Karlık ve Cemel, 2012).

Bu çalışmada, 60 girişli, üç çıkışlı, tek gizli katmanda 30 yapay sinir hücresinin kullanıldığı bir ağ yapısı tercih edilmiştir. Sınıflandırıcı yazılımı olarak MatLab kullanılmıştır. İlk katmanlarda kullanılan transfer fonksiyonu tanjant sigmoid, çıkış katmanında ise lineer'dir. Öğrenme algoritması, gradyent azaltım algoritması olan genelleştirilmiş delta kuralı (generalized delta rule learning) algoritmasıdır. Kullanılan çok katmanlı idrak (Multi Layered Perceptron-MLP) yapısı Şekil 4.4.'te verilmiştir (Karlık ve Cemel, 2012).



Şekil 4.4. Kullanılan MLP (Multi Layered Perceptron) Mimarisi.

Bu çalışmada, diyabet hastalığının, hastanın ağız kokusundan yararlanılarak tanılanmasında alternatif bir yöntem sunulmuştur. Çalışmanın başlangıç aşamasında hedeflenen ağız kokusuna sebebiyet veren pek çok hastalığın tanısını yapmak idi. Ancak, daha hassas ölçüm yapan çoklu sensör dizisine sahip bir Elektronik Burun temin etmek oldukça pahalı olması ve daha fazla sayıda değişik hastalıklara ait veri elde etmekteki zorluklardan ötürü, sadece iki tip diyabet hastalarına ait ağız kokuları üzerine çalışılmıştır. Hastalardan elde edilen veriler üç gruba ayrılır: İlk grup hastalar Tip 1 diyabet, ikinci grup hastalar Tip 2 diyabet, üçüncü grup ise hasta olmayanlar olarak sınıflandırılmıştır. Sınıflandırıcı olarak üç değişik tip (BP, RBF ve LVQ) YSA algoritması denenmiş ve RBF algoritmasının daha iyi sonuç verdiği gözlenmiştir (Karlık ve Cemel, 2012).

İleriye yönelik olarak, bu çalışma ağız kokusuna sebebiyet veren verem, akciğer kanseri vb. diğer hastalıkların da dâhil edilmesiyle genişletilerek çok daha verimli hale getirilebilir, tabiplerin daha hızlı tanı koymalarına yardımcı olunabilir (Karlık ve Cemel, 2012).

4.5. Eşit Genişlikli Ayrıştırma ve Yapay Sinir Ağı Temelli Meme Kanseri Teşhisi

Kadınlarda oldukça sık görülen meme kanseri son yıllarda önemli oranda artış göstermektedir ve dünyada kadın sağlığını tehdit eden önemli bir problem olmaya devam etmektedir. Türkiye' de kanser kontrol ve araştırma birliğinin son istatistiklerine göre kanser hastalığı, kalp ve damar hastalıklarından sonra en önemli ikinci ölüm

tehlikesidir. Bu kanser tipleri arasında meme kanseri yüzde otuzluk oranla ön sıralarda yer almaktadır. Meme kanserinin sebepleri tam olarak bilinemediğinden henüz meme kanserinin engellenebilmesi için kesin bir metot geliştirilememektedir. Bu yüzden erken teşhis hastalıktan kurtulma oranını önemli ölçüde etkilemektedir. Bu alanda kullanılan mamogram olarak adlandırılan insan göğsünden X-Ray ışınları kullanılarak elde edilen iki boyutlu görüntüler hastalığın teşhisinde anahtar rol oynamaktadır (İşeri ve ark., 2011).

Bu çalışmada kullanılan sinir ağı, 5 girişe sahip olup, yapılan farklı denemeler neticesinde gizli katmanda 10 adet sinir hücresine sahip modelin en iyi sonucu elde ettiği görülmüştür. Sinir ağının eğitiminde birden fazla gizli katman kullanılarak yapılan deneyler neticesinde, tek gizli katman kullanılarak ede edilen değerlerden daha iyi değerler elde edilememiştir. Neticesinde, 5 giriş hücresine, 10 gizli katman sinir hücresine ve 1 çıkış hücresine sahip sinir ağı modeli kullanılmıştır. Sinir ağı modelinin eğitimi için Levenberg Marquardt eğitim algoritması kullanılmıştır (İşeri ve ark., 2011).

Çalışmada kullanılan toplam 123 hastaya ait mamogram görüntüsünün %70' i eğitim, %15'i test ve %15'i ise doğrulama için kullanılmıştır. Tablo 1'de kullanılan yapay sinir ağı modelinin eğitim, test, doğrulama ve tüm veriye vermiş olduğu doğruluk değerleri *R* değeri baz alınarak verilmiştir (İşeri ve ark., 2011).

Çalışma neticesinde kullanılan farklı MLPNN modellerinden eğitim, test, doğrulama ve tüm veri kümesi için elde edilen R değerleri Şekil 4.5.'te verilmiştir.

MLPNN Model	Eğitim Verisi	Test Verisi	Doğrulama Verisi	Tüm Veri
5-5-1	0.98770	0.96616	0.96104	0.97822
5-10-1	0.99458	**0.97606**	0.88258	0.97089
5-20-1	0.96097	0.95654	0.94055	0.95578

Şekil 4.5. MLPNN Modellerinin R Değerleri

Bu çalışmada EWD olarak adlandırılan eşit genişlikli ayrıklaştırma yöntemi ve Çok Katmanlı Algılayıcı Ağları kullanılarak, mamogram görüntülerinin sınıflandırılması işlemi yapılmıştır. Mamogramlar, meme kanserine ilişkin ön teşhis aşamasında kullanılan görüntülerdir. Hızlı olarak çekilip elde edilmeleri ve hızlı olarak okunmaları, hastalığın teşhisine yönelik hızı da dolaylı olarak etkilemektedir. Mamogramlardan bilgisayar destekli sistemlerle bilgi çıkartımı ve sınıflandırma işlemleri için ağırlıklı olarak görüntüler içerisinde var olan kitleler veya

mikrokireçlenme noktalarına ait şekilsel özelliklerin elde edilmesi ve bu özellikler kullanılarak da farklı sınıflandırıcı yapılara giriş olarak verilmesi çalışılmaktadır. Bu makalede, farklı bir yaklaşımla, görüntü içerisindeki şekilsel özellikleri kullanmadan, tamamen görüntülerdeki gri seviyelerdeki değişim referans alınarak, EWD temelli bir özellik çıkartımı işlemi yapılmıştır. EWD yöntemi için kullanılan *K* değeri, özellik vektörümüzün boyutunu belirlerken, sınıflandırma işlemi için de farklı sonuçlar elde etmemizi sağlayabilir. Bu çalışmada *K*=5 olarak alınarak işlemler gerçekleştirilmiş ve sınıflandırma açısından yüksek doğruluk değerlerine ulaşılmıştır (İşeri ve ark., 2011).

4.6. Yapay Sinir Ağları Kullanılarak EMG Sinyallerinin Sınıflandırılması ve Neuropathy Hastalığının Teşhisi

Bu çalışmada 59 hastaya ait Abductor Pollicis Brevis (APB) kası First Dorsal Interosseous (FDI) ve Abductor Digiti Minimi (ADM) kaslarına ait EMG işaretlerine Hızlı Fourier Dönüşümü (HFD) analizi uygulanmıştır. Bu uygulama sonucu elde edilen HFD katsayıları yapay sinir ağının (YSA) geriyayılım algoritmasıyla eğitilmiştir ve teşhis amaçlı sınıflama gerçekleştirilmiştir. Bu sınıflama esnasında Eşlenik gradyan ve hızlıprop öğrenme algoritmaları eğitim esnasında kullanılmıştır. 500 öğrenme çevriminden (epoch) sonra, her iki öğrenme algoritmasına ait performans değerleri hesaplanarak, karşılaştırmaları yapılmıştır. Sonuç olarak, HFD katsayıları geri yayılımlı yapay sinir ağıyla eğitilmiş ve neuropathy ve normal EMG sinyalleri %97 doğrulukla sınıflandırılmıştır (Hardalaç ve Poyraz, 2002).

Bu çalışma gerçekleştirilirken 59 kişi üzerinde çalışılmıştır. EMG işaretleri alınan kişilerin 33'i normal ve 26 tanesi de neuropathy olarak tespit edilmiştir. YSA'nın eğitilmesi sonucunda eşlenik gradyan öğrenme tekniği normal hastaya ait EMG işaretlerini %96 doğrulukla ve neuropathy'li hastaya EMG işaretlerini de %97 doğrulukla sınıflamıştır. Diğer hızlıprop öğrenme tekniği ise normal hastaya ait EMG işaretlerini %92 doğrulukla ve neuropathy'li hastaya EMG işaretlerini de %97 doğrulukla sınıflamıştır. Her iki öğrenme tekniğinin öğrenme hızlarını incelemek için MSE eğrisine başvurulmuştur. Elde edilen sonuca göre yapay sinir ağları, ileri beslemeli mimari ile geri yayılım algoritması kullanılarak, eşlenik gradyan ve hızlıprop öğrenme tekniği aynı hızda ve eşlenik gradyan öğrenme tekniği hızlıprop öğrenme tekniğine göre daha az hatayla öğrendiği performans ve test sonuçlarında görülmüştür. YSA tahmin değerleri Şekil 4.6.'da verilmiştir (Hardalaç ve Poyraz, 2002).

EMG işaretinden elde edilen HFD katsayıları																	YSA çıkış değeri		Arzulanan çıkış değeri	
HFD 1	HFD 2	HFD 3	HFD 4	HFD 5	HFD 6	HFD 7	HFD 8	HFD 9	HFD 10	HFD 11	HFD 12	HFD 13	HFD 14	HFD 15	HFD 16	HFD 17	normal	neuropathy	normal	neuropathy
256	228	180	177	176	173	173	168	159	149	137	132	123	107	85	75	73	0,003	1,0012	0	1
256	228	180	179	176	177	172	164	160	150	139	129	122	105	78	70	69	0,004	1,0018	0	1
256	227	174	176	178	179	175	167	158	150	138	131	124	108	80	73	75	-0,001	1,0027	0	1
257	228	180	175	179	174	172	170	162	152	142	132	124	108	89	87	90	0,001	0,9998	0	1
257	228	180	175	179	174	172	170	162	152	142	132	124	108	89	87	90	0,001	0,9998	0	1
-15	18	23	10	5,8	2,6	-4	-14	-22	-23	-26	-34	-43	-54	-67	-79	-83	0,993	0,0025	1	0
-55	-28	-22	-30	-42	-43	-49	-59	-71	-81	-87	-93	-100	-107	-116	-123	-127	1,006	-0,007	1	0
-2	30	30	-6	-35	-64	-72	-81	-91	-98	-105	-120	-127	-134	-153	-161	-169	0,996	-0,006	1	0
-63	-28	-22	-33	-42	-48	-57	-61	-63	-64	-69	-79	-90	-104	-121	-126	-133	1,001	0,0017	1	0
-4	28	32	27	22	15	10	0,6	-12	-23	-29	-34	-45	-58	-72	-81	-89	0,988	0,0039	1	0

Şekil 4.6. YSA Tahmin Değerleri

4.7. Yapay Sinir Ağları ve Destek Vektör Makineleri ile Kemik Erimesinin Teşhisi

Bu tez çalışmasında, bahsedilen sorunlardan esinlenerek, bilgisayarlı tomografi görüntüleri üzerinden örüntü tanıma yöntemi ile kemik erimesi olan hasta dokuların sağlıklı dokulardan ayırt edilmesi amaçlanmıştır. Çalışmanın ana yapısında dalgacık dönüşümünden faydalanılarak öznitelikler elde edilmiştir. Temel bileşen analizi yardımıyla öznitelik sayısı azaltılmış, örüntü tanımlayarak hasta olabilecek kemik dokusu ile sağlıklı kemik dokusunu sınıflandırabilmek için yapay sinir ağları ve destek vektör makineleri yöntemlerinden faydalanılmıştır. Sınıflandırma sonuçlarının karşılaştırılması Şekil 5.7.'de verilmiştir (İstanbullu, 2013).

	DN	YN	DP	YP	Duyarlılık	Özgünlük	Doğruluk
YSA	51	9	52	8	0.85	0.86	%86
DVM	58	2	53	7	0.96	0.89	%92.5

Şekil 4.7. Sınıflandırma sonuçlarının karşılaştırılması

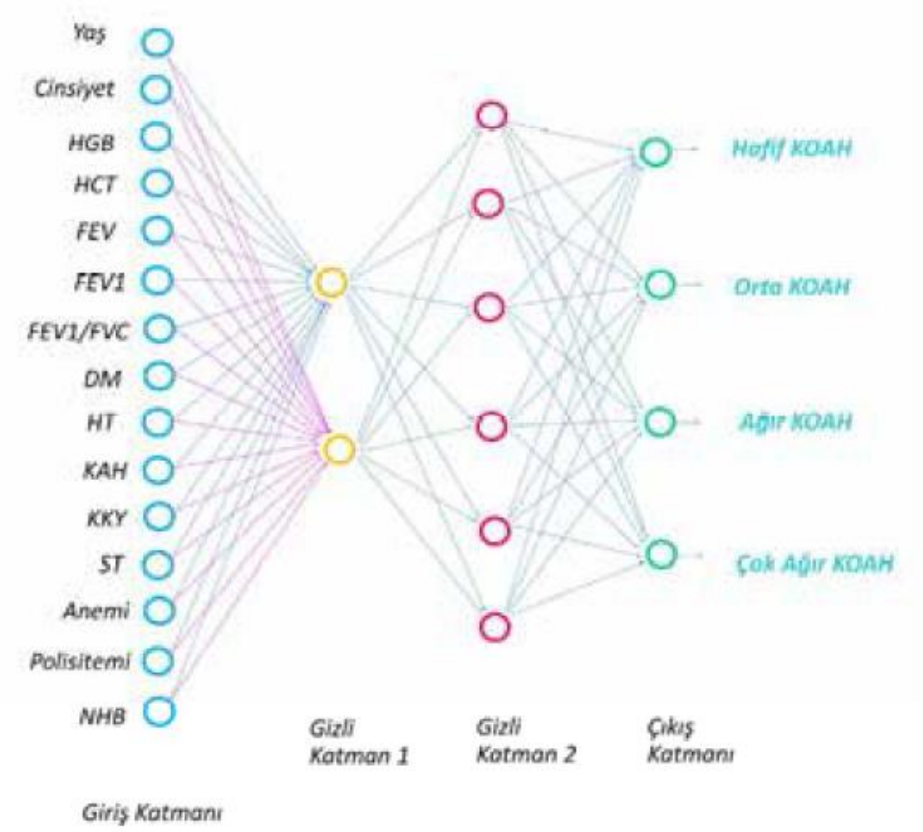
Sonuç olarak biyomedikal alanda yapılan çeşitli çalışmalarda sıkça kullanılan bu iki yöntemin başarısı karşılaştırılmış ve yapay sinir ağları ile yapılan sınıflandırmada %86 başarıyla erken dönem görüntülerine osteoporoz teşhisi konulabilmiştir. Destek vektör makineleri ile gerçekleştirilen sınıflandırmada ise doğruluk yüzdesi artırılarak %92.5 doğrulukta teşhis konulabilmektedir. Bu çalışmanın sonucunda alınmış olan başarılı sonuçlar kemik erimesi teşhisi için önemli bir katkı sağlamaktadır (İstanbullu, 2013).

4.8. Yapay Sinir Ağları ile Kronik Obstrüktif Akciğer Hastalığının Sınıflandırılması

Son yıllarda hastalıkların teşhisinde yeni ve yardımcı bazı sınıflandırıcı, karar verici yazılımlar ve araçların kullanılabilirliği üzerine çeşitli çalışmalar yapılmaktadır. Bu araçlardan birisi de Yapay Sinir Ağlarıdır (YSA). Teşhisin konulmasında yardımcı olabilecek, teşhis süresinin kısalması, zaman kazancı ve verimliliğin artışı gibi faydalar sağlayacak bu gibi sistemlerin medikal AR-GE alanında oldukça faydalı olacağı açıktır (Işık ve ark., 2015).

Bu çalışmada Kronik Obstrüktif Akciğer Hastalığının (KOAH) YSA ile sınıflandırılması amaçlanmıştır. Çalışmada kullanılan veriseti; 15 Değişken ile, 4 KOAH Hastalık Seviyesi (Hafif, Orta, Ağır, Çok ağır) için 507 Kişiye ait verileri içermektedir. Çalışma kapsamında Matlab Kod ile 2 arakatmanlı YSA yapısı 5 fold çapraz doğrulama (crossvalidation) yöntemi kullanılmış, test örneklerinde Ortalama Karesel Hata (MSE) ve Ortalama Mutlak Hata (MAE) değerleri hesaplanmıştır. Sonuçta hatalar MSE=0,00996, MAE=0,02478 şeklinde, doğruluk oranı ise % 99 olarak bulunmuştur. Sonuçlara bakıldığında 15 değişken ile oldukça yüksek bir doğruluk oranına ulaşıldığı görülmüştür. Dolayısıyla YSA'nın çalışmada kullanılan veri seti ile KOAH sınıflandırmasında kullanılabileceği sonucuna ulaşılmıştır. Bu sayede, KOAH seviyesinin hızlı ve yüksek başarım ile belirlenmesinde hekime yardımcı olabilecek bir sistem elde edilmiştir (Işık ve ark., 2015).

Bu çalışmada bulgulara göre, en iyi sonucu veren ağ yapısı 2 gizli katman (1. Arakatman 2 nöron; 2. Arakatman 6 nöron) şeklinde oluşmuştur. Ağın topolojisi 15 giriş, 2 arakatman ve 4 çıkış şeklindedir (Şekil 4.8.) (Işık ve ark., 2015).



Şekil 4.8. Oluşturulan YSA yapısı

Yapay sinir ağları içinde insan beynindekine benzer nöronlar bulunur. Paralel işlem yapan nöronlar ise katmanları oluşturur. Şekil 4.8.'de gösterilen dairelerden her biri nöronları, aynı renk nöronların oluşturduğu bölümler katmanları, oklar ise bağlantı noktalarını temsil etmektedir. Oluşan ağ yapısı incelendiğinde verisetinde bulunan 15 değişken giriş katmanını oluşturmakta, 2 gizli katmanda ağırlıklandırılarak işlem gören değişkenler, ağın çıkışında hafif, orta, ağır ve çok ağır olarak sınıflandırmayı gerçekleştirmektedir. Oluşturulan YSA yapısı (Şekil 5.8.) ile sonuçta bulunan hatalar; MSE=0,00996, MAE=0,02478 şeklindedir. Ortalama karesel hata baz alındığında sonuçta % 0,996'lık bir hata ile % 99 gibi yüksek bir doğruluk oranına ulaşılmıştır (Işık ve ark., 2015).

Çalışma	Hastalık	Metot	Doğruluk
Ashizawa ve diğ., 1999	İnterstisyel Akciğer Hastalığı	Üç - katmanlı, geri - yayınımlı bir algoritması olan ileri - beslemeli YSA	0,947 ROC eğrisinin altında kalan alan
Coppini ve diğ., 2007	KOAH hastalarında Amfizem	İleri beslemeli çok katmanlı YSA	% 90
Er, ve Temurtas, 2008	KOAH	İki farklı YSA	% 93,14- 96,08
Er ve diğ., 2009	KOAH	Radyal tabanlı fonksiyon YSA	% 90,20
Er ve diğ., 2009	KOAH ve Zatürre	YSA ve Yapay bağışıklık sistemleri	% 92, 14- 94,00 (4 Metod)
Er ve diğ., 2010	Göğüs hastalıkları	Çok katmanlı, olasılıksal, öğrenen vektör nicemlemesi ve genelleştirilmiş regresyon sinir ağları	% 88,24 – 92,16 (8 Metod & 6 Hastalık + Kontrol)
Er ve diğ., 2012	Göğüs hastalıkları	Yapay bağışıklık sistemi	% 93.84
Bhuvaneswari ve diğ., 2014	Akciğer hastalıkları	Füzyon bazlı özellik çıkarım tekniği; genetik algoritma; karar ağacı, k en yakın komşu (KNN), çok katmanlı perseptron YSA	% 90'ın üzerinde
Goryński ve diğ., 2014	Erken akciğer kanseri	YSA	En iyi skor için (ROC) eğrisinin altında kalan alan 0,90'ın üzerinde
Önerilen çalışma	KOAH	YSA	% 99

Şekil 4.9. Literatür Sonuçlar – Özet

Literatür çalışmaları Şekil 4.9.'da özet olarak verilmiştir. Tabloda KOAH ve bazı akciğer hastalıklarıyla ilgili sınıflandırmada kullanılan yapay zeka yöntemleri ve sonuçta elde edilen doğruluk oranları bulunmaktadır. Çalışmalarda başarılı sonuçlara ulaşıldığı görülmektedir (Şekil 4.9.) (Işık ve ark., 2015).

Bu çalışmada, literatürde yapılan çalışmalardan farklı olarak YSA kullanarak KOAH seviyelerinin belirleneceği temel bir sistem tasarımı üzerinde çalışılmıştır. Literatür verileri ve bizim elde ettiğimiz bulgular karşılaştırıldığında oldukça yüksek doğruluk oranlarına ulaşıldığı görülmektedir. Sonuçlara bakıldığında az değişken sayısı ile oldukça yüksek bir doğruluk oranına ulaşıldığı görülmüştür. Dolayısıyla YSA'nın çalışmada kullanılan veri seti ile KOAH sınıflandırmasında kullanılabileceği sonucuna ulaşılmıştır. Bu sayede, KOAH seviyesinin hızlı ve yüksek başarım ile belirlenmesi hastanın etkin tedavisinin başlamasında hekime yardımcı olacaktır (Işık ve ark., 2015).

4.9. Bölüm Değerlendirmesi

Bu bölümde geçmişte yapılmış yapay sinir ağlarının medikal alanda kullanılması ile ilgili çalışmalardan bazıları araştırılmıştır ve incelenmiştir. Bu araştırmalar ve incelemeler tez çalışmamda örnekler teşkil etmektedir ve yol göstermektedir.

5. MATERYAL VE METOT

Geliştirilen proje çalışmasında aşağıdaki yazılım araçları kullanılmıştır.

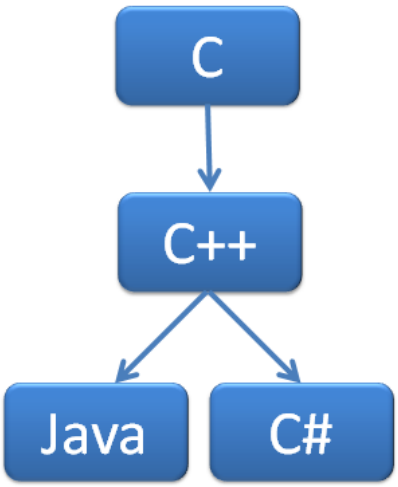
- C# .Net Programlama Dili
- Microsoft Visual Studio 2010

5.1. C# Programlama Dili

C# Programlama Dili, Microsoft'un geliştirtirdiği yeni nesil programlama dilidir. Aynı zamanda Microsoft tarafından geliştirilen .NET Teknolojisi'ne yönelik geliştirilmiş programlama dillerden biridir. Microsoft tarafından geliştirilmiş olsa da ECMA ve ISO standartları altına alınmıştır.

C#, 1990'ların sonlarında Microsoft'ta meydana çıktı ve Microsoft'un .NET stratejisinin bir parçası oldu. C# ilk kez 2000'in ortalarına doğru alfa sürümü ile piyasada yerini aldı. C#ın baş üreticisi, çeşitli başarılara imza atmış olan, dünyanın önde gelen dil uzmanlarından Anders Hejlsberg'dir (Aydın, 2013).

C#'ın aile ağacının yapısı Şekil 5.1.'de gösterildiği gibidir. C#'ın dedesi C'dir. C# söz dizimini, anahtar kelimelerin birçoğunu ve operatörlerini C'den almıştır. C#, C++ ile tanımlanan nesne modeli temel alınarak oluşturulmuş ve bu nesne modelini geliştirmiştir.



Şekil 5.1. C#'ın aile ağacı (Aydın, 2013)

C#'ın en önemli özelliklerinden bazıları, standart olarak bulunan ve yazılım bileşenleri için sunulan destek ile bağlantılı yapıda bulunmaktadır. Bundan dolayı, C# bileşen yönelimli bir dil (component-oriented language) olarak tanımlanır, çünkü C#, yazılım bileşenlerini yazmak için bütünleşik destek içermektedir (Aydın, 2013).

5.2. .NET Framework

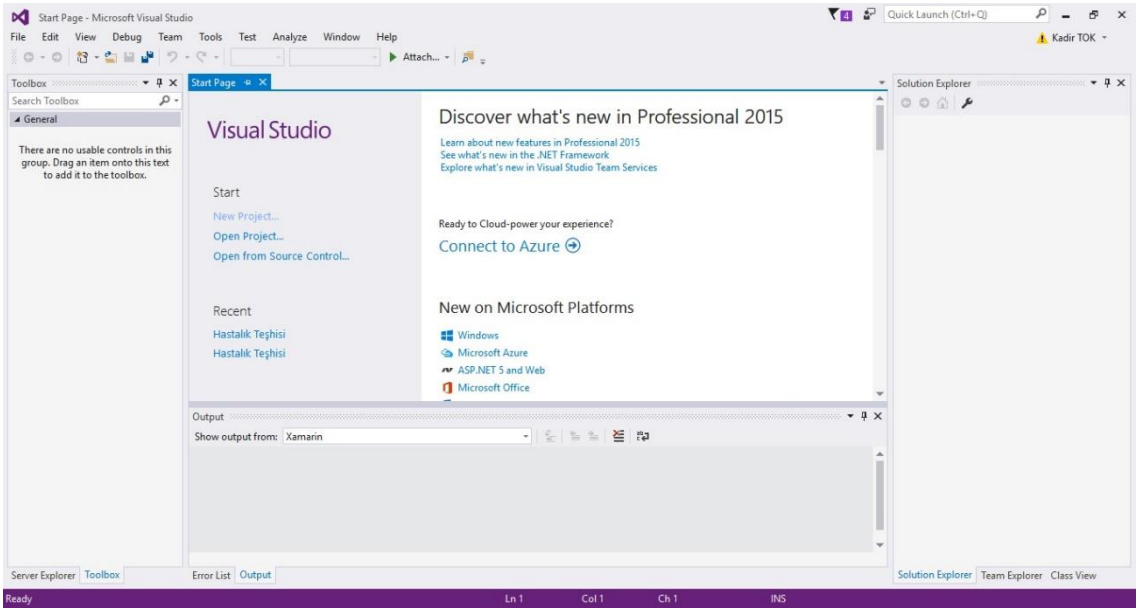
.NET Framework, bileşene yönelik uygulamaların geliştirilmesine destek veren bir yazılım ortamıdır. Değişiklik gösteren programlama dillerinin birlikte yürütülmesini olağan duruma getirebilen bir platformdur. Windows platformu için taşınabilirlik, güvenlik ve ortak bir programlama tasarısı oluşturmaya destek olur. .Net sınıf kütüphanesi de programlama için geliştirilmiştir. Bu kütüphane, programın yürütülme ortamına ulaşımına olanak tanır (Anonim).

5.3. Microsoft Visual Studio

Microsoft tarafınca geliştirilen bir tümleşik geliştirme ortamıdır (integrated development environment - IDE). Windows CE, .NET Compact Framework, Windows Mobile, Microsoft Windows, .NET Framework ve Microsoft Silverlight tarafınca yardım sağlanan bütün platformlar için yönetilen kodla beraber web siteleri, yerel kod ve Windows Forms uygulamaları, web servisleri ve web uygulamalarıyla beraber grafiksel ve konsol kullanıcı arabirimi uygulamaları geliştirmek amacıyla kullanılmaktadır (Aydın, 2013).

Visual Studio IntelliSense ile birlikte "code refactoring" destek olan bir kod editörü barındırmaktadır. Dahili hata ayıklayıcı, hem makine-seviyesinde hem de kaynak-seviyesinde çalışır. Bunların dışındaki bütünleşik araçlar, web tasarımcısı, GUI uygulamaları, veritabanı şema tasarımcısı ve sınıf tasarımcısı oluşturabilmek amacıyla bir form modelleyicisi bulundurur. Neredeyse her fonksiyonellik seviyesinde dahil olacak şekilde, kaynak denetleme sistemleri için destek (Subversion ve Visual SourceSafe gibi) sağlayan uzantıları kabul etmektedir (Aydın, 2013).

Visual Studio, farklı programlama dillerini destek sağlar, bu sayede hata ayıklayıcısının ve kod editörünün hemen hemen bütün programlama dillerini desteklemesine yardımcı olur. Entegre diller C# (Visual C# ile), VB.NET (Visual Basic .NET üzerinden), C/C++ (Görsel yoluyla C++) ve F# (Visual Studio 2010'dan sonra) bulundurmaktadır (Anonim). Microsoft Visual Studio programlama arayüzünün görüntüsü Şekil 5.2.'de gösterilmektedir.



Şekil 5.2. Microsoft Visual Studio tümleşik geliştirme ortamı

5.4. Bölüm Değerlendirmesi

Bu bölümde tez çalışmasında geliştirilen yapay sinir ağları yazılımının geliştirildiği C# programlama dili, yazılımın geliştirilmesinde kullanılan .NET Framework kütüphanesi ve yazılımın geliştirilmesinde kullanılan Microsoft Visual Studio bütünleşik geliştirme ortamı incelenmiştir.

6. YAZILIMIN TASARIMI VE GERÇEKLEŞTİRİMİ

6.1. Romatoid Artrit Hastalığının İncelenmesi

Romatoid artrit, el ve ayak eklemleri benzeri daha çok ufak eklemleri simetrik olarak tutan uzun süreli iltihaplı bir tür romatizmadır. Yalnızca ufak eklem tutulumu ile kalmayıp diz, kalça, omuz benzeri büyük eklemlerde de görülebilir. Dünyada rastlanan iltahaplı romatizmaların en başında gelir. Eklemlerde en büyük tahribata ve şekil bozukluklarına sebep olan romatizmalardan biridir (Anonim).

Romatiod artritin sebebi bilinmemektedir. Kalıtsal yatkınlık önemli olmakla birlikte bu tüm hastlarda geçerli değildir. Kalıtsal yatkınlığı bulunan her bireyde romatoid artrit gelişmediği gibi kalıtsal yatkınlığı bulunmadığı halde hastalık ortaya çıkabilmektedir. Kalıtsal yatkınlık yalnızca bu kişilerde görülme ihtimalinin daha çok bulunduğunu gösterir. Bugün için bilmediğimiz bedensel ve çevresel birtakım etmenler de hastalığın oluşmasında muhtemelen rol oynamaktadır veya tetikleyici olmaktadır. Sebebi hakkında günümüzde söylenebilecekleri kısaca özetlersek; romatoid artrit, kalıtsal yatkınlığı bulunan bireylerde bilinmeyen bazı çevresel ve bedensel faktörlerin de devreye girmesiyle meydana gelebilen bir rahatsızlıktır. Bazen hastalığı tetikleyen etmen büyük bir gerginlik, üzüntü, sıkıntı, mikrobik etmenler ve başka sebepler bulunabilmekte, kimi zaman ise hiçbir tetikleyici etken olmamaktadır. Bulaşıcı bir hastalık değildir (Anonim).

Hemen hemen her yaşta görülebilmekte ancak en sık olarak 20 ve 50 yaşları arasında başlar. Kadınlarda görülme sıklığı erkeklere göre 2-3 kat daha çoktur. Her ırkta ve her ülkede görülebilir. Görülme sıklığı bazı ırklarda daha çok, bazılarında daha az olabilir. Örnek olarak, Güney Afrika yerlilerinde bu oran 1/1000 iken, bazı Kızılderili toplumlarında 5/100'e kadar çıkmaktadır. Türkiye'nin de dahil olduğu yedi Avrupa ülkesinde son zamanlarda romatoid artrit görülme sıklığı araştırması devam etmektedir. Ülkemizdeki çalışma Türkiye Romatizma Araştırma ve Savaş Derneği(TRASD) şemsiyesi altında Akdeniz Üniversitesi, Fiziksel Tıp ve Rehabilitasyon Kliniği(FTR) önderliğinde ülke genelinde pek çok üniversite FTR kliniklerinin desteği ile tamamlanmış olup yakında bu hastalığın ülkemizdeki görülme sıklığı diğer 6 Avrupa ülkesi ile birlikte yayınlanacaktır. Antalya bölgesi tüm Türkiye'den göç alan bir bölge olarak ülkemizin bu açıdan genel durumunu yansıtabilir. Akdeniz Üniversitesi FTR-Romatoloji Kliniğinin 2004 yılında tamamladığı bir taramada Antalya bölgesindeki romatoid artrit görülme sıklığı 3,8/1000 (yaklaşık her 260 kişiden biri) olarak tespit edilmiştir (Anonim).

İklimsel özellikler romatoid artritin meydana gelmesinin ya da görülme sıklığını etkilememektedir. Hastalığa yakalanmada soğuk ya da sıcak bölgelerde oturuyor olmanın, kuru veya rutubetli ortamlarda bulunmanın etkisi yoktur. İklimsel faktörler muhtemelen hastalığın görülme sıklığını etkilememektedir ancak hastalığa bağlı şikayetlerin daha fazla ya da daha az duyulmasını etkileyebilir. Örneğin bölgesel sıcak uygulama şiş eklemlerdeki şikayetleri arttırabilirken soğuk uygulama, tam aksine azaltabilir (Anonim).

6.2. Yapay Sinir Ağında Kullanılan Veri Seti

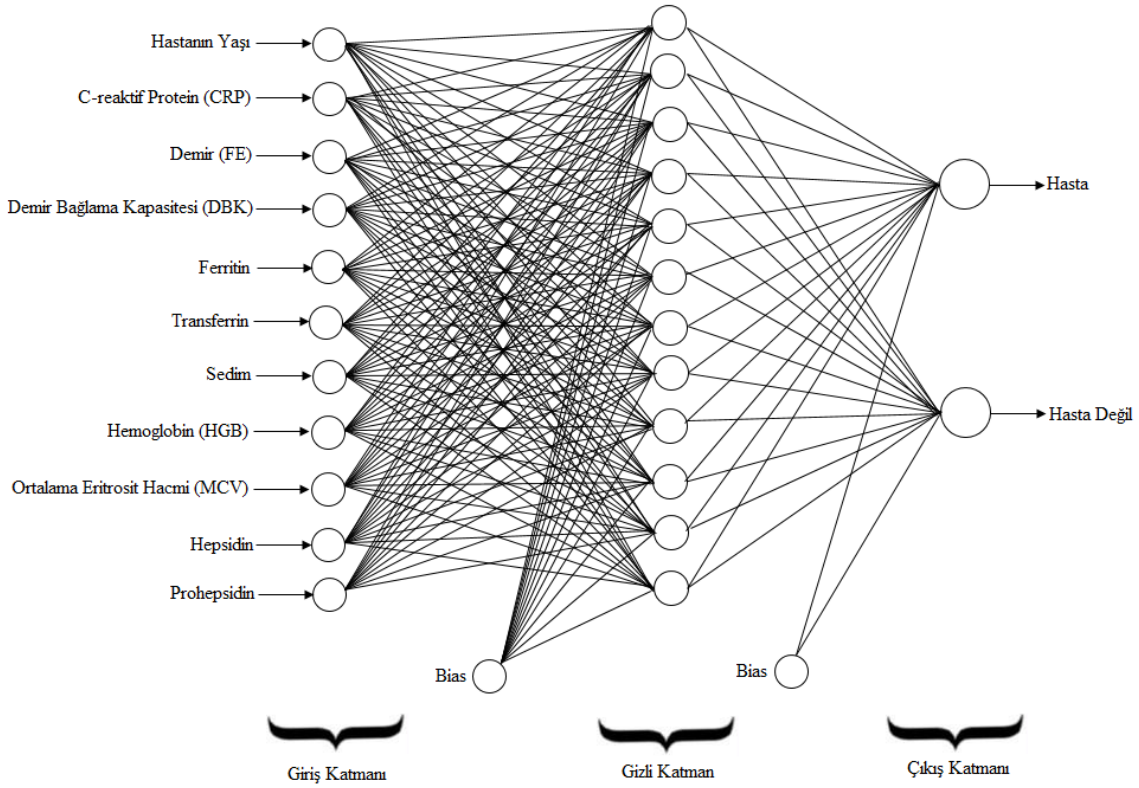
Bu çalışmada kullanılan veriler için Selçuk Üniversitesi Tıp Fakültesi Hastanesi'nden temin edilmiş olan romatoid artrit hastalığı veri kümesi kullanılmıştır. Bu veri kümesindeki datalar 20 romatoid artrit hastası ve 49 hasta olmayan toplamda 69 adet gerçek deneklerden alınmıştır. Veri kümesinde toplam 11 özelik için denekler üzerinde ölçümler yapılmıştır ve bu özellikler Çizelge 6.1.'de detaylı olarak verilmiştir.

Çizelge 6.1. Romatoid artrit hastalığı veri seti

Özellik No	Özellik Bilgisi
1	Hastanın yaşı
2	C-reaktif protein(CRP)
3	Demir(FE)
4	Demir bağlama kapasitesi(DBK)
5	Ferritin
6	Transferrin
7	Sedim
8	Hemoglobin(HGB)
9	Ortalama eritrosit hacmi(MCV)
10	Hepsidin
11	Prohepsidin

6.3. Yapay Sinir Ağının Yapısı

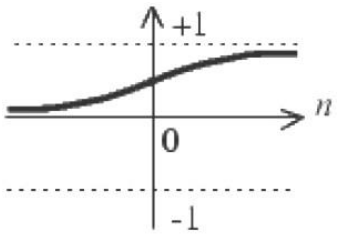
Geliştirilem yazılımda, dinamik giriş, dinamik gizli ve dinamik çıkış katmanlı bir ağ yapısı vardır. Nöron sayıları dinamik olarak kullanıcı tarafından belirlenebilmektedir. Gizli ve çıkış katmanındaki tüm nöronlar ile ilişkili Bias nöronuna sahip bir ağ yapısı kullanılmıştır. Şekil 6.1.'de programda kullanılan dinamik MLP yapısı gösterilmiştir.



Şekil 6.1. MLP yapısı

6.4. Transfer Fonksiyonu

Ağda ağırlıklar rastgele -1 ile +1 arasında atanabilir veya dosyadan daha önce kaydedilmiş olan ağırlıklar tekrar yüklenebilir. Tüm nöronlar için aktifleşme fonksiyonu olarak Şekil 6.2.'deki Sigmoid fonksiyonu kullanılmıştır.



Şekil 6.2. Sigmoid fonksiyonu a=logsig(n)

Sigmoid fonksiyonunda bulunan α değerinin hesaplanmasında denklem (6.1.) ve denklem (6.2.) kullanılmıştır. Kodlama kısmında bu görevi Şekil 6.3.'teki kod parçacığı yapmaktadır.

$$net_j = \sum_{i=1}^{n} X_i w_{ij} \tag{6.1.}$$

$$a = \frac{1}{1 + \exp(-net_j)} \tag{6.2.}$$

```
public void aktivasyonFonksiyon()
        {
            transferDeger = 1 / (1 + Math.Exp(-netDeger));
        }
```

Şekil 6.3. Aktivasyon fonksiyonu hesaplama kodu

6.5. Öğrenme Algoritması

Ağın eğitilmesinde hatayı geriye yayma (back propagation) algoritması kullanılmıştır. Her bir çıkış nöronu için hatanın hesaplanmasında denklem (6.3.) kullanılmaktadır. Kodlama kısmında ise bu görevi Şekil 6.4.'teki kod parçacığı yapmaktadır.

$$\delta_j = F_j\,(1 - F_j)(T_j - F_j) \tag{6.3.}$$

```
public void hesaplaHata(double hedefCikis)
        {
            hataDeger = transferDeger * (1 - transferDeger) * (hedefCikis -
transferDeger);
        }
```

Şekil 6.4. Çıkış nöronu hata hesaplama kodu

Her bir gizli katman nöronun hatasının hesaplanmasında denklem (6.4.) kullanılmaktadır. Kodlama kısmında ise Şekil 6.5.'teki kod parçacığı kullanılmıştır.

$$\delta_j = F_j\,(1 - F_j) \sum_{k} \delta_k\,\omega_{kj} \tag{6.4.}$$

```
public void hesaplaHata(CikisNoron[] cikisKatman)
        {
            double hata = 0;

            for (int i = 0; i < cikisKatman.Length; i++)
            {
                hata += cikisKatman[i].hataDeger * cikisBaglanti[i].agirlik;
            }

            hataDeger = transferDeger * (1 - transferDeger) * hata;
        }
```

Şekil 6.5. Gizli katman nöronu hata hesaplama kodu

Tüm ağılıkların güncellenmesinde denklem (6.5.) ve denklem (6.6.) kullanılmaktadır. Kodlama kısmında ise Şekil 6.6.'daki kod parçacığı kullanılmıştır.

$$\Delta\omega_{ji} = \alpha.\delta_j.F_i \tag{6.5.}$$

$$\omega_{ji}(t+1) = \omega_{ji}(t) + \Delta\omega_{ji} + m.\omega_{ji}(t) \tag{6.6.}$$

```
public void guncelleAgirlik(double transferDeger, double hataDeger, double egitmeOran,
double momentumOran)
        {
            agirlik = agirlik + egitmeOran * transferDeger * hataDeger + agirlik *
momentumOran;
        }
```

Şekil 6.6. Ağırlıkların update edilmesinde kullanılan kod

Ayrıca ağda öğrenme hızı ve momentum katsayıları kullanılmıştır. Ağın genel hatasının hesaplanmasında denklem (6.7.) kullanılmaktadır. Kodlama kısmında ise Şekil 6.7.'deki kod parçacığı kullanılmıştır.

$$E = (1/2)\sum_j (|T_j - F_j|) \tag{6.7.}$$

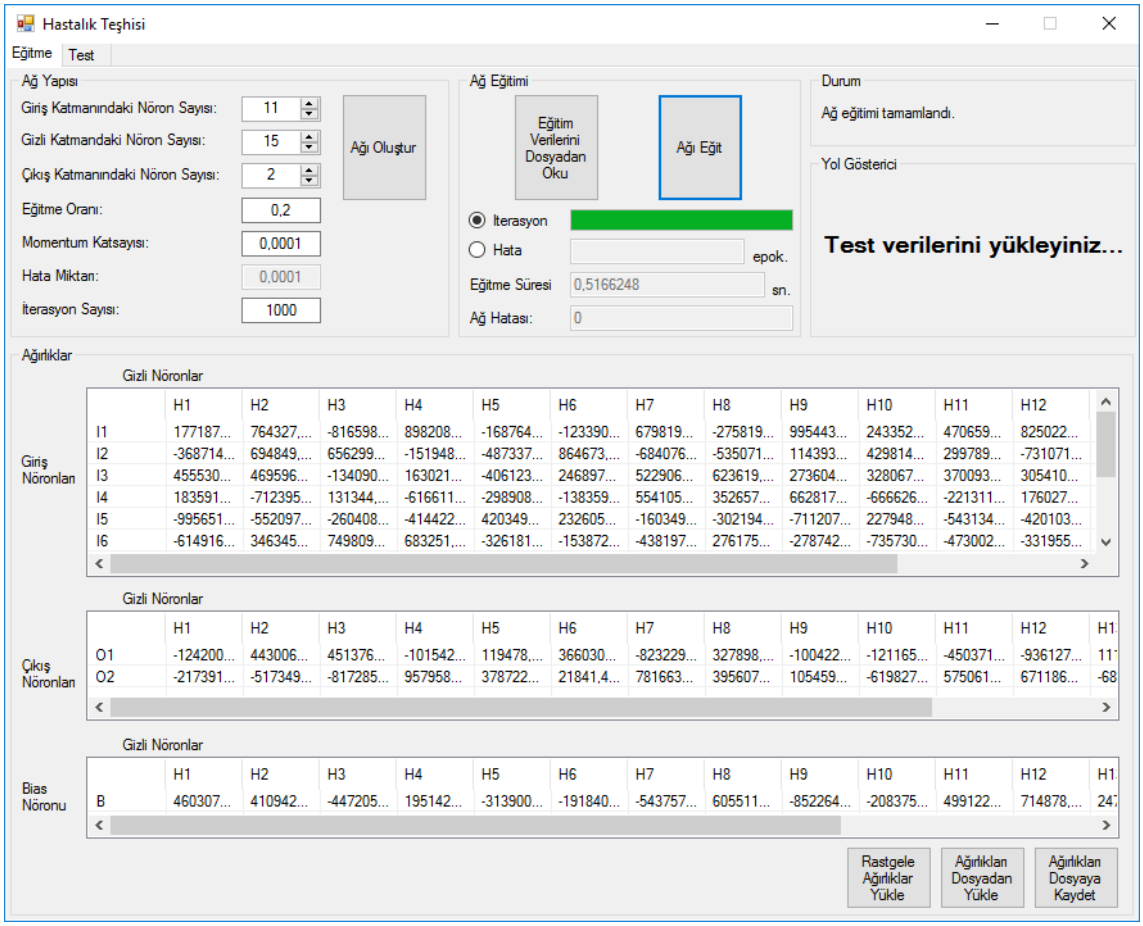
```
public void hesaplaAgHata(double[] cikisDegerler, double[] hedefDegerler)
        {
            double hata=0;

            if (cikisDegerler.Length == hedefDegerler.Length)
            {
                for (int i = 0; i < cikisDegerler.Length; i++)
                {
                    hata += Math.Abs(hedefDegerler[i] - cikisDegerler[i]);
                }
                agHata = hata / 2;
            }
        }
```

Şekil 6.7. Genel ağ hatasının hesaplanmasında kullanılan kod

6.6. Geliştirilen Yazılımın Özellikleri

C# programlama dilinde görsel bir uygulama geliştirilmiştir. Geliştirilen uygulama ile istenilen öğrenme hızı, istenilen momentum katsayısı, belirli bir epok iterasyon sayısına ve kabul edilebilir bir ağ hatasına kadar öğrenme süreci uygulanabilmektedir. Ayrıca başlangıç ağırlıkları rastgele atanabildiği gibi dışarıdan bir dosya ile istenilen değerlerde set edilebilmektedir. Eğitim sonunda ise genel ağ hatası ve eğitim süreci gözlenebilmektedir. Uygulamanın eğitme arayüzü Şekil 6.8.'deki gibidir.



Şekil 6.8. Yazılımın eğitme arayüzü

Projeyi çalıştırdığımızda ilk olarak Şekil 6.8.'deki eğitme arayüzü karşımıza çıkmaktadır. Ağ yapısı, ağ eğitimi, durum, yol gösterici ve ağırlıkların bulunduğu kısımlar bulunmaktadır.

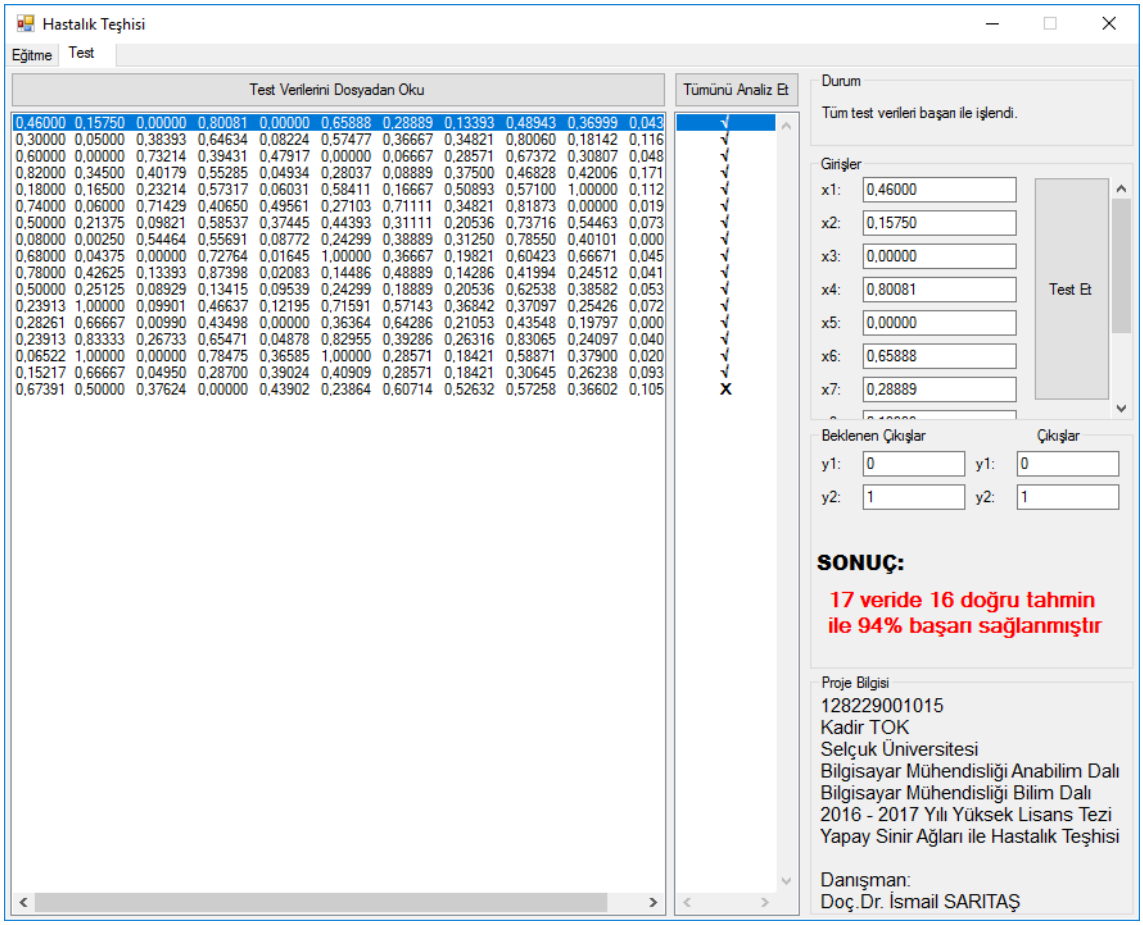
Ağ Yapısı: Ağın yapısının dinamik olarak belirlendiği bölümdür. Katmanlarda bulunan nöron sayıları, eğitme oranı, momentum katsayısı, hata miktarı ve iterasyon sayısının belirlendiği bölümdür.

Ağırlıklar: Ağırlıkların atandığı ve güncellenmiş ağırlıkların gözlemlendiği bölümdür. Bu kısımda ağırlıklar rastgele atanabilir veya dosyadan önceden belirlenmiş ağırlıklar yüklenebilir ve dosyaya son ağırlıklar kaydedilebilir.

Ağ Eğitimi: Ağ eğitiminin yapıldığı ve programın anlık kullanım işlemini gösteren bölümdür. Eğitim verileri dosyadan okunur sonrasında ağ eğitimi yapılır. Eğitme süresi ve ağdaki hata miktarının görüntülendiği bölümdür.

Durum: Gerçekleştirilen işlemlerin durumlarının gösterildiği bölümdür.

Yol Gösterici: Kullanıcılar için klavuzluk yapan yol gösterici bölümüdür.



Şekil 6.9. Yazılımın test arayüzü

Test arayüzü, eğitimi tamamlanan ağın test etmek için kullanıldığı Şekil 6.9.'daki kısımdır. Test verilerinin dosyadan yüklendiği ve test etmek için seçimler yapılabilen bir liste kutusunun bulunduğu bölümdür.

Girişler: Dinamik olarak oluşturulan ve eğitilen ağın test edildiği bölümdür. Herhangi bir test verisinin değerleri liste kutusundan seçilip ilgili alanlara aktarılarak

test edilir. Ayrıca “Tümünü Analiz Et” seçeneği ile tüm test verileri analiz edilerek ağın toplam başarısı yüzdesel olarak gösterilir.

Çıkışlar: Çıkışlar bölümünde, girişler bölümünde test edilen test verilerinin YSA çıkışları gösterilir. Çıkış sonuç olarak gösterilir.

Proje Bilgisi: Proje bilgisi bölümünde projeye ait bilgiler yer almaktadır.

6.7. Bölüm Değerlendirmesi

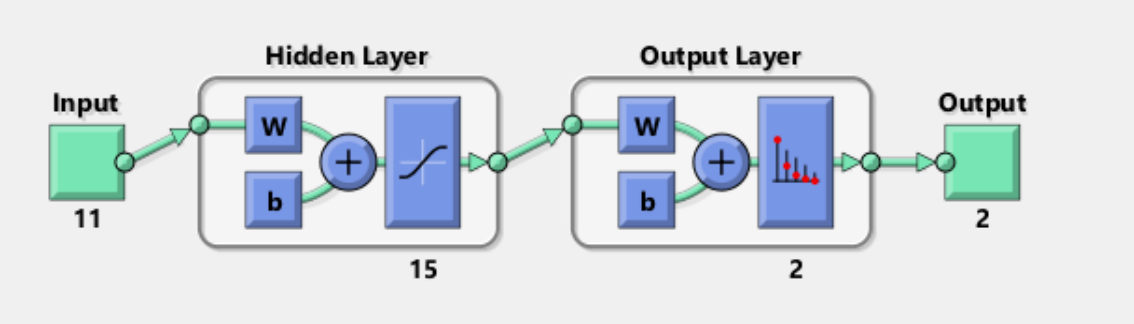
Bu bölümde romatiod artrit hastalığı araştırılmıştır ve incelenmiştir, yapay sinir ağında kullanılan veri seti gösterilmiştir, yapay sinir ağının MLP (Multi Layer Perceptron) yapısı gösterilmiştir, transfer fonksiyonu ve formülü belirtilmiştir, öğrenme algoritmasından, ağırlık güncelleme ve hata hesaplama formüllerinden bahsedilmiştir ve son olarak geliştirilen yazılımın özelliklerine değinilmiştir.

7. SONUÇLAR

7.1. Matlab YSA Uygulaması ve Sonuçları

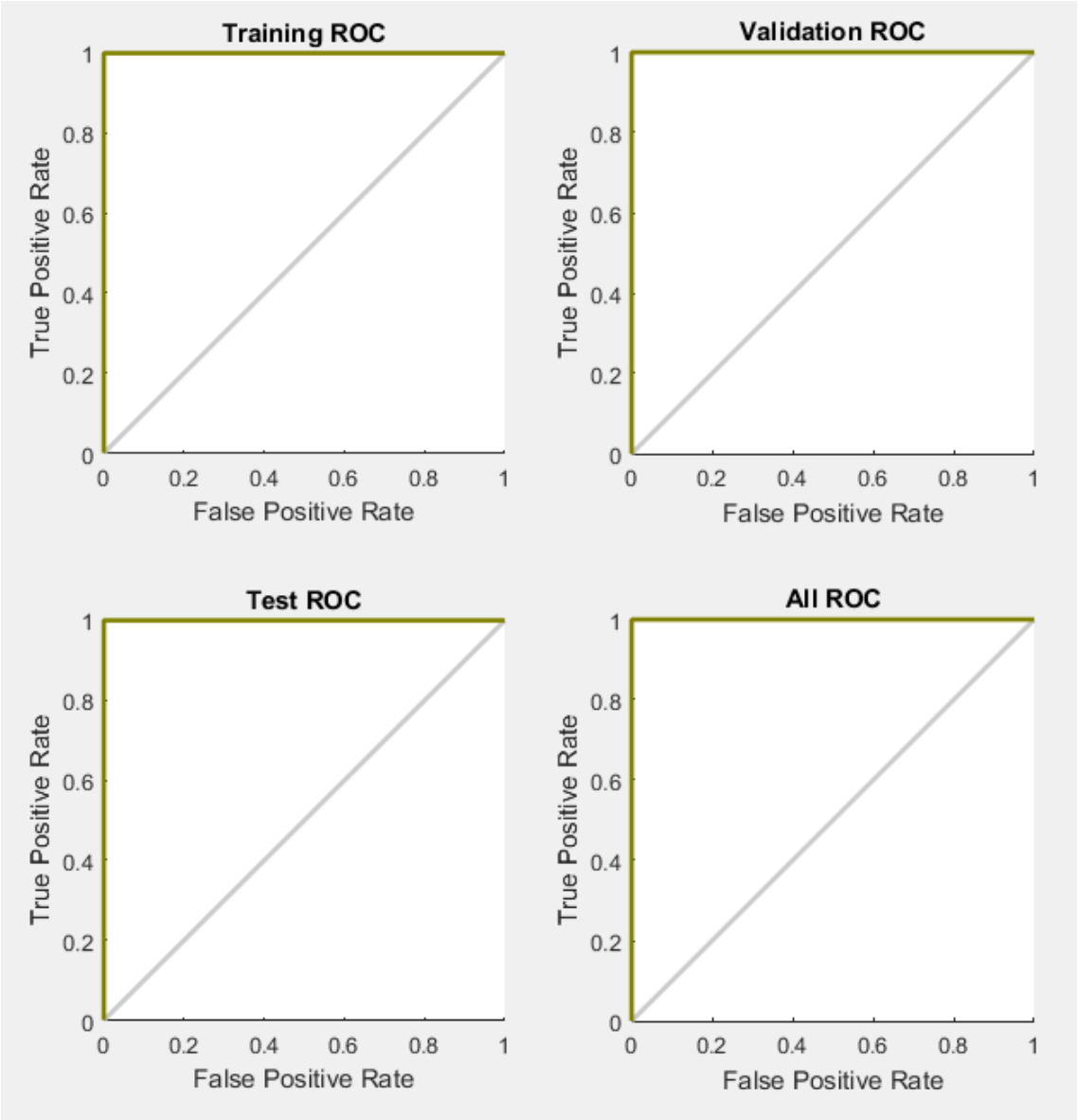
Çalışma kapsamında Matlab ortamında yapay sinir ağlarında romatoid artrit hastalığı teşhisi için sınıflandırma yapılmıştır. Öğrenme yöntemi olarak Back Propagation algoritması kullanılmıştır.

Çalışmada kullanılan veri kümesinde 20 adet romatoid artrit hastası ve 49 adet de romatoid artrit hastası olmayan deneklere ait toplamda 69 örnek vardır. Bu dataların %75'i YSA'nın eğitim aşamasında, geri kalan %25'lik kısmı ise test verisi olarak kullanılmıştır. Uygulamada giriş katmanında 11, gizli katmanda 15 ve çıkış katmanında 2 nöron kullanılmıştır. Bu ağ yapısı Şekil 7.1.'de gösterilmiştir.



Şekil 7.1. YSA Ağ Yapısı

Ağın eğitilmesi ve test sonuçları gerçekleştirilmiştir. Matlabta elde edilen eğitim, test ve YSA'nın genel çalışma karakteristik grafiği (ROC Analizi) Şekil 7.2.'de gösterilmiştir.



Şekil 7.2. ROC Analizi (Matlab)

Matlab ortamında “nprtool” ile elde edilen eğitim, test ve tüm veri karmaşıklık (confisuon) matrisi de Şekil 7.3.’te verilmiştir.



Şekil 7.3. Karmaşıklık (Confusion) Matrisi (Matlab)

Eğitimde kullanılan 16'sı sağlıklı 36'sı hasta olmak üzere toplam 52 adet verinin tamamı başarılı olarak sınıflandırılmıştır.

Doğrulamada kullanılan 1'i sağlıklı, 2'si hasta olmak üzere toplam 3 adet verinin tamamı başarılı olarak sınıflandırılmıştır.

Testte kullanılan 4'ü sağlıklı 10'u hasta olmak üzere toplam 14 adet verinin 1'i başarısız, 13'ü başarılı olarak sınıflandırılmıştır.

Ağın geneline baktığımızda kullanılan 21'i sağlıklı 48'i hasta olmak üzere toplam 69 verinin 1'i başarısız 68'i başarılı olarak sınıflandırılmıştır.

Analiz sonucunda doğru tespit yüzdesi Backpropagation yöntemi ile %94 olurken, yanlış tespit yüzdesi ise %6 olarak hesaplanmıştır. Ayrıca yapılan çalışma sınıflama tekniklerinden olan Perceptron teoremi ile de kıyaslanmıştır. Perceptron algoritması ile %71 doğruluk yüzdesi, %29 yanlışlık yüzdesi elde edilmiştir. Bu sonuçlara göre YSA ile daha başarılı performans sonuçlarına ulaşıldığı görülmüştür.

Test işlemi sonucunda performans analizini gerçekleştirmek için "Doğruluk Yüzdesi" ve "Yanlışlık Yüzdesi" kriterleri esas alınmıştır. Çalışmada 1000 epoch için başarılı sonuçlar elde edilmiştir. Her iki sınıflandırma yöntemi için 1000 epoch sonucunda elde edilen doğruluk ve yanlışlık yüzdeleri Çizelge 7.1.'de gösterilmiştir.

Çizelge 7.1. 1000 epoch için performans yüzdeleri

Algoritma	**Doğruluk Yüzdesi**	**Yanlışlık Yüzdesi**
Backpropagation	94	6
Perceptron	71	29

Bu çalışmada, romatoid artrit hastalığının teşhisinde yardımcı olacak bir YSA tabanlı sınıflandırma amaçlanmıştır. Sınıflandırma yöntemlerinin başarı değerlendirmesinde, "doğruluk yüzdesi" ve "yanlışlık yüzdesi" performans ölçümleri kullanılmıştır.

Veri kümesindeki test performansında Bacpropagation yöntemi ile %94 doğruluk yüzdesi elde edilirken, Perceptron yöntemi ile doğruluk yüzdesi %71 oranında hesaplanmıştır. Elde edilen sonuçlar, mevcut veri kümesi ile romatoid artrit hastalığının teşhisinde YSA sınıflandırıcıların eğitim ve test performansı açısından başarılı sonuçlar verdiği gözlenmiştir.

Çalışmada daha sonra sistem performansını yükseltmek için daha farklı önişleme teknikleri kullanılabilir ve farklı sınıflandırma yöntemleri ile bu çalışmanın performans değerlendirmesi yapılabilir.

7.2. Gerçekleştirilen Yazılımın Uygulama Sonuçları

Uygulamada farklı işlemci hızlarına sahip bilgisayarlarda birçok eğitim ve test denemeleri yapılmış olup birtakım çıktı örnekleri alınmıştır. 4 farklı eğitim ve test işlemleri Intel Core i5-6200U 2.3GHz olan işlemciye sahip bilgisayar üzerinde uygulanmıştır ve Çizelge 7.2.'de verilen sonuçlar elde edilmiştir.

Çizelge 7.2. Intel Core i5-6200U 2.3GHz sonuçları

Eğitme Oranı	Momentum Katsayısı	Hata/İterasyon Seçimi	Hata Miktarı	Ağ Hatası	İterasyon Sayısı	Öğrenme Süresi
0,2	0,0001	İterasyon	-	0	10000	4,75 sn.
0,2	0,0001	Hata	0	0	1075	0,66 sn.
0,7	0,0001	Hata	0	0	968	0,63 sn.
0,2	0,0001	İterasyon	-	0	1000	0,47 sn.

Aynı eğitim ve test işlemleri Intel Core i5 M430 2.27GHz olan işlemciye sahip bilgisayar üzerinde uygulanmıştır ve Çizelge 7.3.'te verilen sonuçlar elde edilmiştir.

Çizelge 7.3. Intel Core i5 M430 2.27GHz sonuçları

Eğitme Oranı	Momentum Katsayısı	Hata/İterasyon Seçimi	Hata Miktarı	Ağ Hatası	İterasyon Sayısı	Öğrenme Süresi
0,2	0,0001	İterasyon	-	0	10000	5,17 sn.
0,2	0,0001	Hata	0	0	1102	0,89 sn.
0,7	0,0001	Hata	0	0	935	0,66 sn.
0,2	0,0001	İterasyon	-	0	1000	0,52 sn.

Aynı eğitim ve test işlemleri Intel Core i5-4460 3.20GHz olan işlemciye sahip bilgisayar üzerinde uygulanmıştır ve Çizelge 7.4.'te verilen sonuçlar elde edilmiştir.

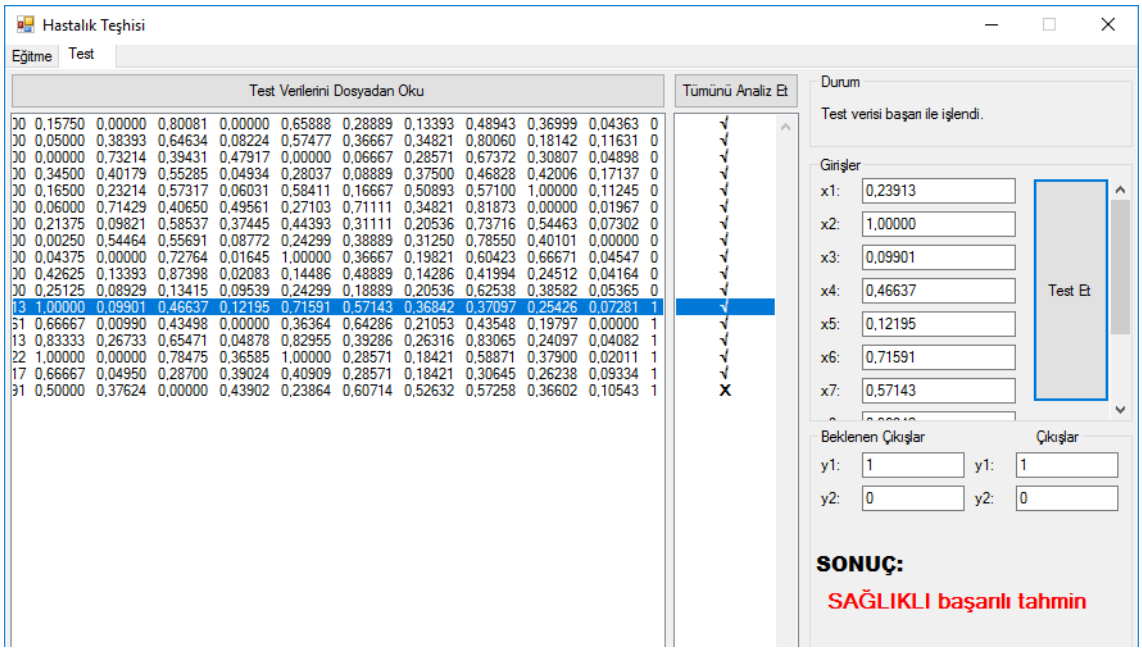
Çizelge 7.4. Intel Core i5-4460 3.20GHz sonuçları

Eğitme Oranı	Momentum Katsayısı	Hata/İterasyon Seçimi	Hata Miktarı	Ağ Hatası	İterasyon Sayısı	Öğrenme Süresi
0,2	0,0001	İterasyon	-	0	10000	3,18 sn.
0,2	0,0001	Hata	0	0	1106	0,42 sn.
0,7	0,0001	Hata	0	0	926	0,35 sn.
0,2	0,0001	İterasyon	-	0	1000	0,30 sn.

Ağın eğitilmesinde ve test edilmesinde yapılan çalışmalar sonucunda ağın 0 (sıfır) hataya ortalama 500-1500 epok sonrasında ulaştığı ancak yapılan test girdileri ardından ağın bu durumda tam olarak öğrenemediği belirlenmiştir. Bu durumun ise

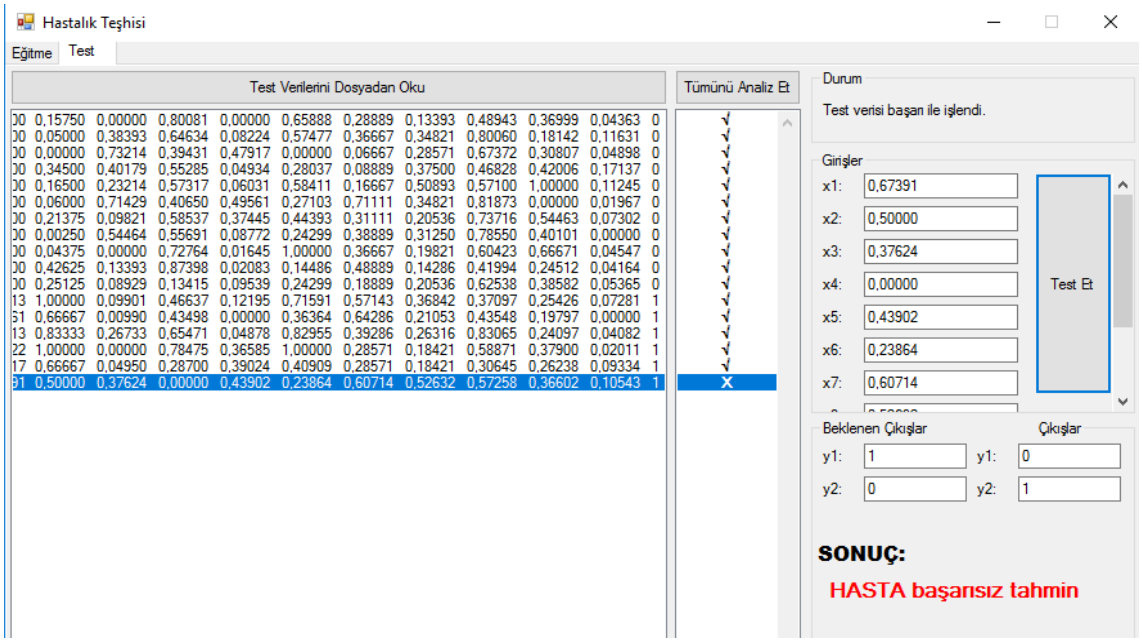
eğitim verilerinin büyük çoğunluğunun çok düzenli ve makul değerlerde olduğuna kanaat getirilmiştir. Ancak epok sayısının 1.000 ila 2.000 arasında uygulanması ile ağın %82 ve %100 aralığında öğrenme gerçekleştirdiği tespit edilmiştir.

Geliştirilen yazılımda test aşamasında elde edilen sonuçlar Şekil 7.4.'te ve Şekil 7.5.'te verilmiştir.



Şekil 7.4. Sağlıklı başarılı bir tahmin örneği

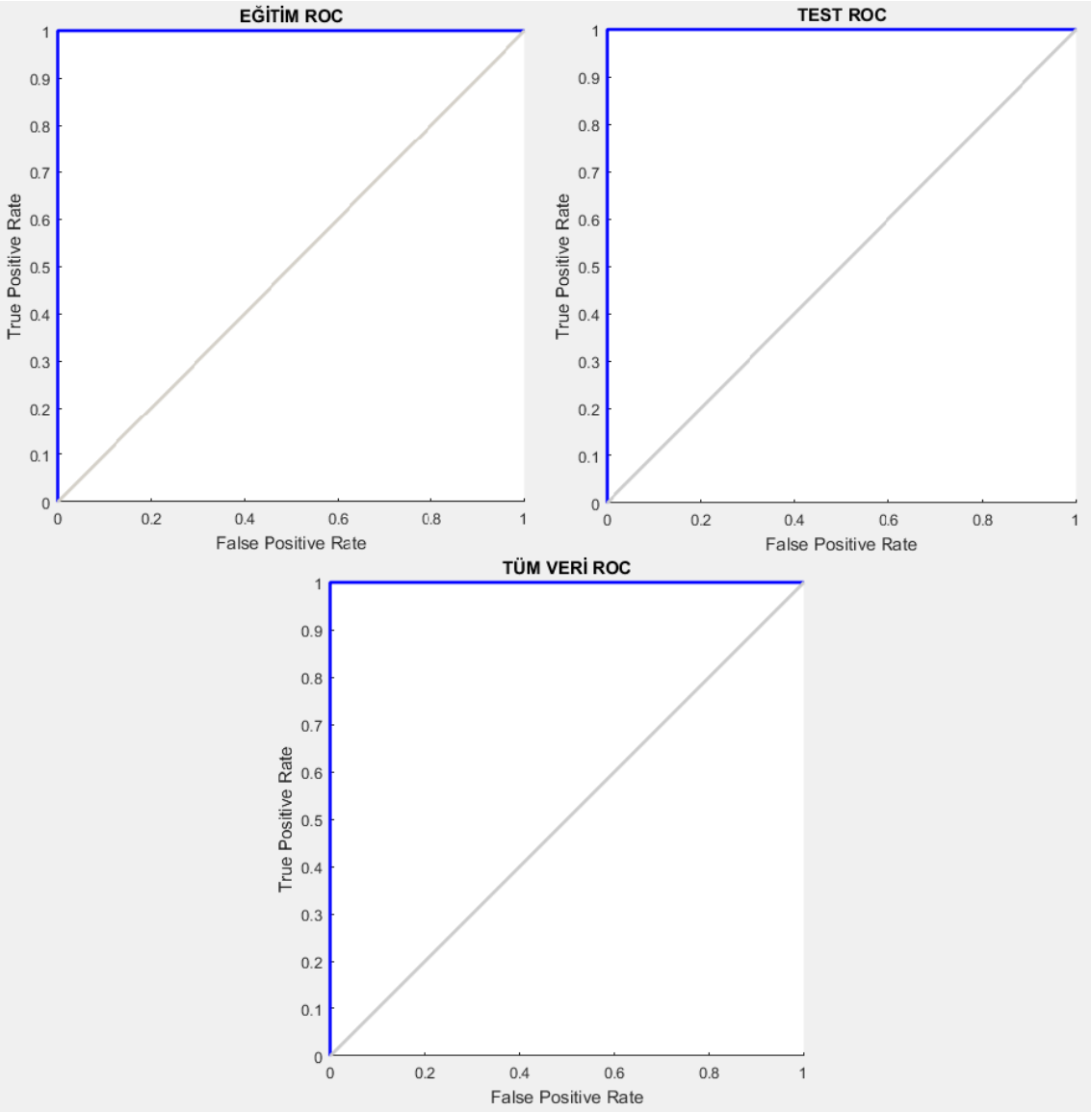
Şekil 7.4.'te görüldüğü gibi 12. Sıradaki sağlıklı test verisi test edilmiştir ve başarılı bir sonuç elde edilmiştir.



Şekil 7.5. Hasta başarısız bir tahmin örneği

Şekil 7.5.'te görüldüğü gibi 17. Sıradaki sağlıklı test verisi test edilmiştir ve başarısız bir sonuç elde edilmiştir.

Eğitim, test ve YSA'nın genel çalışma karakteristik grafiği Matlab kodlarıyla ROC Analizi grafikleri elde edilmiştir (Şekil 7.6).



Şekil 7.6. ROC Analizi

Karmaşıklık Testi

Bir sınıflandırma modelinin performans değerlendirmesi, model tarafından doğru veya yanlış olarak kestirimi yapılmış test kayıtlarının sayısına dayanır. Bu sayılar karmaşıklık matrisi olarak bilinen bir tablo üzerinde çizelgelendirilirler. İkili sınıflandırma problemi için oluşturulan karmaşıklık matrisi çizelgesi Şekil 7.7.'de görüldüğü gibi oluşturulmaktadır.

	Hasta	**Sağlıklı**	**Toplam**
Hasta	A	B	A+B
Sağlıklı	C	D	C+D
Toplam	A+C	B+D	A+B+C+D

Şekil 7.7. Karmaşıklık (Confusion) Matrisi Hesaplaması

$$Duyarlılık \quad A \tag{7.1.}$$

$$Özgüllük \quad D \tag{7.2.}$$

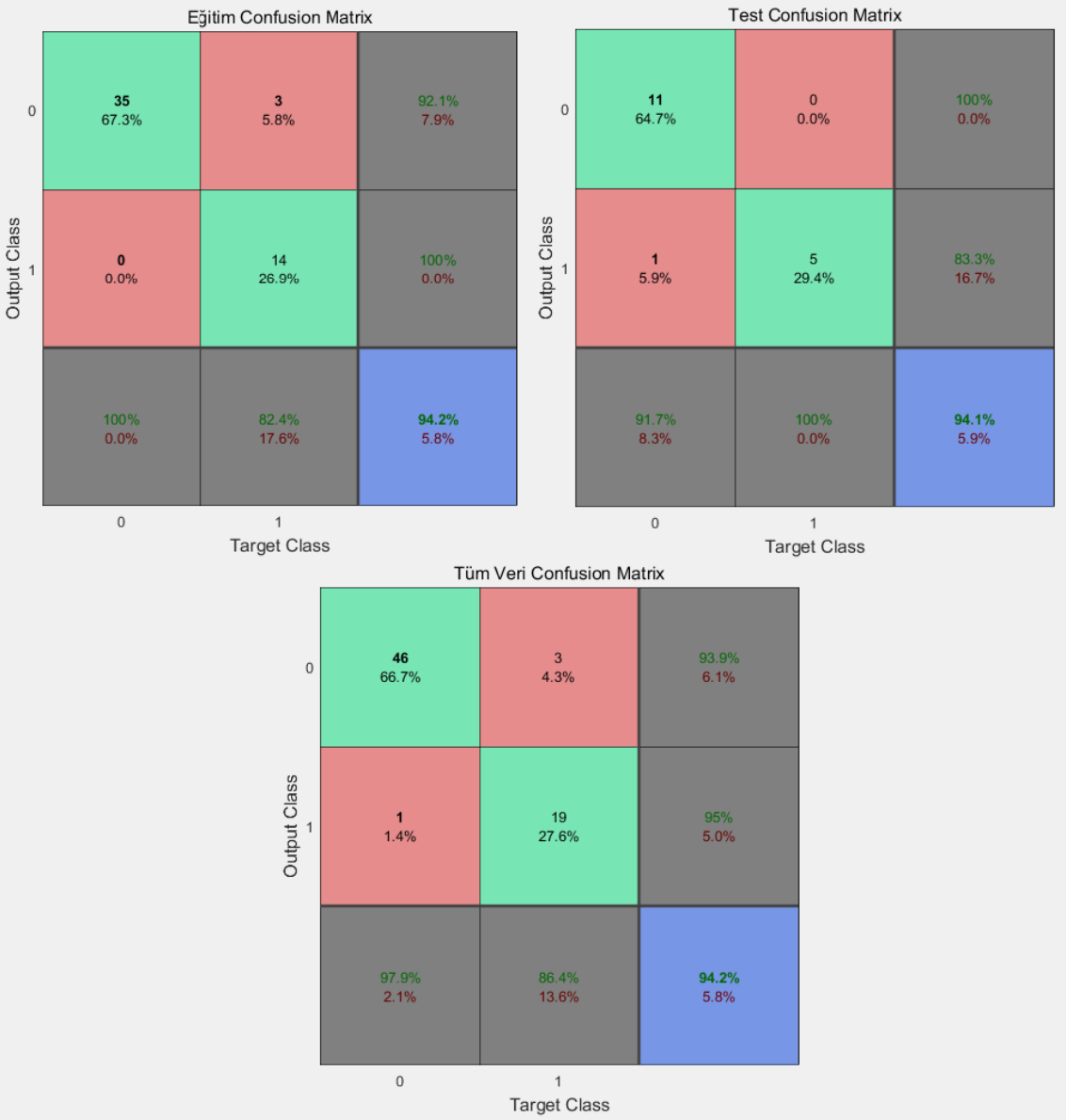
$$Doğruluk \quad A + D \tag{7.3.}$$

$$Tahmin\ Edilen\ Pozitif\ Değer \quad A \tag{7.4.}$$

$$Tahmin\ Edilen\ Negatif\ Değer \quad D \tag{7.5.}$$

Karmaşıklık (Confusion) Matrisi hesaplaması Şekil 7.7.'de verilmiştir. Bu çizelgeye göre Duyarlılık denklem (7.1.), Özgüllük denklem (7.2.), Doğruluk denklem (7.3.), Tahmin Edilen Pozitif Değer denklem (7.4) ve Tahmin Edilen Negatif Değer denklem (7.5.) ile hesaplanmaktadır.

Gerçekleştirdiğimiz yazılımda elde edilen eğitim, test ve tüm veri karmaşıklık (confisuon) matrisi oluşturulmuştur. Elde edilen sonuçlar Şekil 7.8.'de verilmiştir.



Şekil 7.8. Karmaşıklık Matrisi

Karmaşıklık testinde:

Eğitimde kullanılan 38'i sağlıklı 14'ü hasta olmak üzere toplam 52 adet verinin 3'ü başarısız, 49'u başarılı olarak sınıflandırılmıştır.

Testte kullanılan 11'i sağlıklı 6'sı hasta olmak üzere toplam 17 adet verinin 1'i başarısız, 16'sı başarılı olarak sınıflandırılmıştır.

Ağın geneline baktığımızda kullanılan 49'u sağlıklı 20'si hasta olmak üzere toplam 69 verinin 4'ü başarısız 65'i başarılı olarak sınıflandırılmıştır.

7.3. Genel Değerlendirme

Bu tezde esas olarak yapay sinir ağları yönteminin yazılımlarda kullanılması ve geliştirilmesi üzerinde durulmuştur. Görsel araçların genel olarak kullanımı basitleştirmesi ve süratli netice oluşturmaya yardımcı olması yönünden yazılım geliştirme sürecinde de yoğunlukla kullanılması, bu hususta inceleme ve geliştirme yapmaya yönlendirmiştir.

Geliştirilen ve gerçekleştirilen bu yazılım yapay sinir ağlarının topolojisine mümkün mertebede elverişli olacak biçimde hazırlanmıştır. Böylece teorik bilgiye sahip bir kullanıcının geliştirilen yazılımı kullanırken adaptasyon süreci de kısalmış olacaktır.

Yapay sinir ağlarının görsel ifadesinin basit ve anlaşılır olması sebebiyle, geliştirilen yazılım üzerinden yapay sinir ağlarının gerçekleştirilmesi amacıyla görsel konfigürasyon ve eğitim aracının oluşturulması olağan hale gelmiştir. Böylece hem süratli bir biçimde görsel ortamda gerçekleştirilen yazılım kullanılmaktadır. Hem de yazılımın kullanımının öğrenme ihtiyaçları minimize edilmiş olmaktadır.

Microsft Visual Studio yazılım geliştirme ortamının fonksiyonel nitelikler yönünden kuvvetli ve basitçe genişletilebilir olması, geliştirilen yazılımın bu ortama bütünleşik bir biçimde kullanılabilmesine yardımcı olmuştur. Yazılım geliştirme için ihtiyaç duyulan bütün araçlar aynı ortamda ve beraber çalışır olarak yazılım entegre geliştirme ortamı esasına elverişli olacak şekilde geliştirilmiştir.

Görsel aracın ve yazılımın geliştirilmesinde ilk olarak takip edilen yaklaşım yazılım geliştirilen zaman zarfında bütünleştirilmesini sağlamaktır. Ayrıca işlem hızı, hafıza kullanımı vb. performans alanları daha sonraki planda düşünülmüştür. Fakat elde edilen kıyaslamalar neticesinde, uygulama aşamasında başarı kriterleri yönünden kullanıma engel olacak durumların gözlenmediği söylenebilir.

Son yıllarda öğrenebilen yazılımların geliştirilmesi değer kazanırken, bu hedefe yardım edecek araçların da geliştirilmesi ve bir altyapı gerçekleştirilmesi gerekmektedir. Bu hedefe yönelik olarak incelendiğinde, yapılan bu çalışma, dinamik bir ağ gerçekleştirebilmesi ile yapay sinir ağlarının kullanılabileceği farklı konularda teşhis ve sınıflandırma yapılabilen bir araç olarak değerlendirilebilir.

KAYNAKLAR

Altınışık, A., 2012, Toprak İşlemede Traktör Çeki Performansının Yapay Sinir Ağları ile Analizi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Anonim, Romatoid Artrit | Türkiye Romatizma Araştırma ve Savaş Derneği, https://www.trasd.org.tr/hastalik/romatoid-artrit: [10 Haziran 2016].

Anonim, Microsoft Visual Studio - Vikipedi, https://tr.wikipedia.org/wiki/Microsoft_Visual_Studio: [3 Mayıs 2016].

Anonim, Microsoft Visual Studio Nedir? Ne Amaçla Kullanılır?, http://www.mshowto.org/microsoft-visual-studio-nedir-ne-amacla-kullanilir.html: [1 Nisan 2016].

Anonim, Fast Artificial Neural Network Library (FANN) - Fann Explorer, http://leenissen.dk/fann/gui.php: [6 Şubat 2016].

Anonim, Yapay Sinir Ağları - Vikipedi, https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1: [1 Mart 2016].

Aydın, A., 2013, Çoktan Seçmeli Ölçme Sonuçlarının Bilgisayar Yardımıyla Analizi.

Başkurt, M., Durmuş, M. C. ve Gümüşay, M. Ü., 2011, Yapay Sinir Ağları Yardımıyla Plaka Tanıma Sisteminin Oluşturulması ve CBS ile Bütünleştirilmesi.

Çayıroğlu, İ., 2013, IleriAlgoritmaAnalizi-5 Yapay Sinir Ağlari.

Çolak, C., 2005, Ateroskleroz'un Tahmini İçin Bir Yapay Sinir Ağı An Artificial Neural Network For The Prediction Ofatherosclerosis, *Ankara Üniversitesi Tıp Fakültesi Mecmuası*, 58 (04).

Doğan, M., 2003, Yapay Sinir Ağları Temelli Tibbî Teşhis Sistemi, *Yüksek Lisans Tezi, Anadolu Üniversitesi*.

Doğan, M., Babanlı, A. ve Telçeken, S., 2004, Yapay Sinir Ağları Yöntemleri ile Başağrısı Hastalıklarının Sınıflandırılması.

Elmas, Ç., 2003, Yapay Sinir Ağları:(Kuram, Mimari, Eğitim, Uygulama), Seçkin Yayıncılık, p.

Fausett, L., 1994, Fundamentals of Neural Networks: Architectures, Algorithms, and Applications, Prentice-Hall, Inc., p.

Gemici, M., 2015, Xml Tabanlı Uygulamaların Yapay Zeka Modeliyle Gerçek Zamanlı Analizi.

Güney, E., 2009, Kavramsal Radyo Sistemlerinde Yapay Sinir Ağına Dayalı Kanal Seçme Algoritması Tasarımı.

Güven, E. N., Onur, H. ve Sağıroğlu, Ş., 2008, Yapay Sinir Ağları ile Web İçeriklerini Sınıflandırma, *Bilgi Dünyası*, 9 (1), 158-178.

Hardalaç, F. ve Poyraz, M., 2002, Yapay Sinir Ağları Kullanılarak EMG Sinyallerinin Sınıflandırılması ve Neuropathy Kas Hastalığının Teşhisi, *Politeknik Dergisi*, 5 (1).

İstanbullu, M., 2013, Yapay Sinir Ağları Ve Destek Vektör Makineleri İle Kemik Erimesinin Teşhisi, *Fen Bilimleri Enstitüsü*.

İşeri, İ., Öz, C., Dağkurs, L. ve Kara, M., 2011, Eşit Genişlikli Ayrıştırma ve Yapay Sinir Ağı Temelli Meme Kanseri Teşhisi.

Işık, Ü., Güven, A. ve Büyükoğlan, H., 2015, Yapay Sinir Ağları ile Kronik Obstrüktif Akciğer Hastalığının Sınıflandırılması Chronic Obstructive Pulmonary Disease Classification with Artificial Neural Networks, *Vogue*, 15 (18).

Karasulu, B., Ballı, S., Körükoğlu, S. ve Aybars, U., 2008, Kutup Dengeleme Problemi İçin Yüksek Başarımlı Bir Optimizasyon Tekniği, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 14 (2), 175-183.

Karlık, B. ve Cemel, M. S., 2012, Yapay Sinir Ağları ile Ağız Kokusundan Diyabet Tanısı, *Turkiye Klinikleri Journal of Medical Sciences*, 32 (2), 331-336.

Keleşoğlu, Ö., Ekinci, C. E. ve Fırat, A., 2005, The Using of Artificial Neural Networks in Insulation Computations, *Sigma*, 3, 58-66.

Kınacı, A. C., 2006, Görsel Yazılım Geliştirme Ortamı ile Beraber Bir Yapay Sinir Ağı Kütüphanesi Tasarımı ve Gerçekleştirimi, *Ege Üniversitesi*.

Kurt, B., Nabiyev, V. V. ve Turhan, K., 2011, Yapay Sinir Ağları ile Down Sendromunun Tanınması.

Öztemel, E., 2003, Yapay Sinir Ağları.

Pmarrone, Joone - Freecode, http://freecode.com/projects/joone: [29 Şubat 2016].

Şahin, C., Oğulata, S. N., Kırım, S. ve Koçak, M., 2004, Troid Bezi Bozukluklarının Yapay Sinir Ağları ile Teşhisi.

Tektaş, M., Tektaş, N., Onat, N., Gökmen, G., Koçyiğit, G. ve Akıncı, T. Ç., 2010, Web Tabanlı Yapay Zeka Teknikleri Eğitim Simülatörlerinin Hazırlanması (PROJE NO: FEN-E-050608-138).

Yayık, A., 2013, Yapay Sinir Ağı ile Kriptoloji Uygulamaları.

Yıldırım, M. Z., Çavuşoğlu, A., Şen, B. ve Budak, İ., 2014, Yapay Sinir Ağları ile Ağ Üzerinde Saldırı Tespiti ve Paralel Optimizasyonu.

Yılmaz, İ., 2012, Osmanlı Dönemi Mimarlık Eserleri Restorasyon İnşaat Maliyetlerinin Yapay Zeka Yöntemleri ile Tahmini, *İstanbul Kültür Üniversitesi*.

Zell, A., University of Tübingen: SNNS, http://www.ra.cs.uni-tuebingen.de/software/snns/welcome_e.html: [13 Şubat 2016].

Zell, A., University of Tübingen: JavaNNS, http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/: [24 Şubat 2016].

EKLER

EK-1 Yazılım kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace Hastalık_Teşhisi
{
    public partial class AnaSayfa : Form
    {
        Ag ag;
        public int sayacTest = 0;
        public int lbc = 0;
        double[,] egitimGirisDegerler;
        double[,] egitimHedefDegerler;
        public TextBox[] tDiziG;
        public TextBox[] tDiziGKopya;
        public Label[] lDiziG;
        public TextBox[] tDiziC;
        public TextBox[] tDiziCKopya;
        public Label[] lDiziC;
        public AnaSayfa()
        {
            InitializeComponent();
        }
        private void bt_olusturAg_Click(object sender, EventArgs e)
        {
            ag = new Ag(Convert.ToInt32(nud_girisNoronlar.Value),
Convert.ToInt32(nud_gizliNorons.Value), Convert.ToInt32(nud_cikisNoronlar.Value),
Convert.ToDouble(tb_egitmeOran.Text), Convert.ToDouble(tb_momentumOran.Text),
Convert.ToDouble(tb_hedefAgHata.Text));
            if (listBox1.Items.Count != 0)
            {
                listBox1.DataSource = null;
                listBox1.Items.Clear();
            }
            if (lDiziG != null)
            {
                for (int i = 0; i < lDiziG.Length; i++)
                {
                    panel1.Controls.Remove(lDiziG[i]);
                }
            }
            if (tDiziG != null)
            {
                for (int i = 0; i < tDiziG.Length; i++)
                {
                    panel1.Controls.Remove(tDiziG[i]);
                }
            }
            if (tDiziGKopya != null)
            {
                for (int i = 0; i < tDiziGKopya.Length; i++)
                {
                    panel1.Controls.Remove(tDiziGKopya[i]);
```

```
            }
        }
        if (lDiziC != null)
        {
            for (int i = 0; i < lDiziC.Length; i++)
            {
                panel2.Controls.Remove(lDiziC[i]);
            }
        }
        if (tDiziC != null)
        {
            for (int i = 0; i < tDiziC.Length; i++)
            {
                panel2.Controls.Remove(tDiziC[i]);
            }
        }
        if (tDiziCKopya != null)
        {
            for (int i = 0; i < tDiziCKopya.Length; i++)
            {
                panel2.Controls.Remove(tDiziCKopya[i]);
            }
        }
        lDiziG = new Label[Convert.ToInt32(nud_girisNoronlar.Value)+1];
        for (int i = 1; i < lDiziG.Length; i++)
        {
            lDiziG[i] = new Label();
            lDiziG[i].Name = "lb_Giris" + i.ToString();
            lDiziG[i].Text = "x" + i.ToString() + ":";
            this.Controls.Add(lDiziG[i]);
            lDiziG[i].Top = -23 + (i * 26);
            lDiziG[i].Left = 6;
            lDiziG[i].Width = 33;
            lDiziG[i].Height = 13;
            lDiziG[i].Parent = panel1;
        }
        tDiziGKopya = new TextBox[Convert.ToInt32(nud_girisNoronlar.Value) +
Convert.ToInt32(nud_cikisNoronlar.Value) +1];
        for (int i = 1; i < tDiziGKopya.Length; i++)
        {
            tDiziGKopya[i] = new TextBox();
            tDiziGKopya[i].Name = "tb_Kopyax" + i.ToString();
            this.Controls.Add(tDiziGKopya[i]);
            tDiziGKopya[i].Top = -26 + (i * 26);
            tDiziGKopya[i].Left = 40;
            tDiziGKopya[i].Width = 120;
            tDiziGKopya[i].Height = 20;
            tDiziGKopya[i].Parent = panel1;
            tDiziGKopya[i].Visible = false;
        }
        tDiziG = new TextBox[Convert.ToInt32(nud_girisNoronlar.Value) + 1];
        for (int i = 1; i < tDiziG.Length; i++)
        {
            tDiziG[i] = new TextBox();
            tDiziG[i].Name = "tb_x" + i.ToString();
            this.Controls.Add(tDiziG[i]);
            tDiziG[i].Top = -26 + (i * 26);
            tDiziG[i].Left = 40;
            tDiziG[i].Width = 120;
            tDiziG[i].Height = 20;
            tDiziG[i].Parent = panel1;
        }
```

```
            lDiziC = new Label[Convert.ToInt32(nud_cikisNoronlar.Value) + 1];
            for (int i = 1; i < lDiziC.Length; i++)
            {
                lDiziC[i] = new Label();
                lDiziC[i].Name = "lb_Cikis" + i.ToString();
                lDiziC[i].Text = "y" + i.ToString() + ":";
                this.Controls.Add(lDiziC[i]);
                lDiziC[i].Top = -23 + (i * 26);
                lDiziC[i].Left = 6;
                lDiziC[i].Width = 33;
                lDiziC[i].Height = 13;
                lDiziC[i].Parent = panel2;
            }
            tDiziCKopya = new TextBox[Convert.ToInt32(nud_cikisNoronlar.Value) +
1];
            for (int i = 1; i < tDiziCKopya.Length; i++)
            {
                tDiziCKopya[i] = new TextBox();
                tDiziCKopya[i].Name = "tb_Kopyay" + i.ToString();
                this.Controls.Add(tDiziCKopya[i]);
                tDiziCKopya[i].Top = -26 + (i * 26);
                tDiziCKopya[i].Left = 40;
                tDiziCKopya[i].Width = 193;
                tDiziCKopya[i].Height = 20;
                tDiziCKopya[i].Parent = panel2;
                tDiziCKopya[i].Visible = false;
            }
            tDiziC = new TextBox[Convert.ToInt32(nud_cikisNoronlar.Value) + 1];
            for (int i = 1; i < tDiziC.Length; i++)
            {
                tDiziC[i] = new TextBox();
                tDiziC[i].Name = "tb_y" + i.ToString();
                this.Controls.Add(tDiziC[i]);
                tDiziC[i].Top = -26 + (i * 26);
                tDiziC[i].Left = 40;
                tDiziC[i].Width = 193;
                tDiziC[i].Height = 20;
                tDiziC[i].Parent = panel2;
            }
            if (lv_girisGizliAgirliklar.Items.Count!=0)
            {
                lv_girisGizliAgirliklar.Columns.Clear();
                lv_girisGizliAgirliklar.Items.Clear();
            }
                for (int i = 0; i < Convert.ToInt32(nud_gizliNorons.Value) + 1;
i++)
                {
                    lv_girisGizliAgirliklar.Columns.Add("H"+i.ToString());
                }
                lv_girisGizliAgirliklar.Columns[0].Text = "";

                for (int i = 1; i < Convert.ToInt32(nud_girisNoronlar.Value) + 1;
i++)
                {
                    lv_girisGizliAgirliklar.Items.Add(new ListViewItem(new
string[] { "I"+i.ToString(), "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
```

```
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0" },
-1));
                }
            if (lv_cikisGizliAgirliklar.Items.Count!=0)
            {
                lv_cikisGizliAgirliklar.Columns.Clear();
                lv_cikisGizliAgirliklar.Items.Clear();
            }
                for (int i = 0; i < Convert.ToInt32(nud_gizliNorons.Value) + 1;
i++)
                {
                    lv_cikisGizliAgirliklar.Columns.Add("H"+i.ToString());
                }
                lv_cikisGizliAgirliklar.Columns[0].Text = "";
                for (int i = 1; i < Convert.ToInt32(nud_cikisNoronlar.Value) + 1;
i++)
                {
                    lv_cikisGizliAgirliklar.Items.Add(new ListViewItem(new
string[] { "O"+i.ToString(), "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0",
```

```
            bt_ataRastgeleAgirliklar.Enabled = true;
            bt_yukleDosyadanAgirliklar.Enabled = true;
            bt_kaydetAgirliklarDosyaya.Enabled = false;
            bt_okuEgitmeDegerlerDosyadan.Enabled = false;
            bt_egitAg.Enabled = false;
            bt_okuTestDegerlerDosyadan.Enabled = false;
            bt_testAg.Enabled = false;
            bt_tumunuTestAg.Enabled = false;
            tb_iteration.Text = "";
            tb_egitmeSure.Text = "";
            tb_agHata.Text = "";
            progB_Iterasyon.Value = 0;
            lb_durum.Text = "Ağ başarı ile oluşturuldu.";
            lb_durum1.Text = "Ağ başarı ile oluşturuldu.";
            lb_yolGoster.Text = "Ağırlıkları belirleyiniz...";
            lb_sonuc.Text = "";
        }
        private void bt_ataRastgeleAgirliklar_Click(object sender, EventArgs e)
        {
            ag.ataRastgeleAgirlik();
            doldurAgirliklarListeye();
            bt_kaydetAgirliklarDosyaya.Enabled = true;
            bt_okuEgitmeDegerlerDosyadan.Enabled = true;
            lb_durum.Text = "Rastgele ağırlıklar başarı ile atandı.";
            lb_durum1.Text = "Rastgele ağırlıklar başarı ile atandı.";
            lb_yolGoster.Text = "Eğitim verilerini yükleyiniz...";
        }
        public void doldurAgirliklarListeye()
        {
            for (int i = 0; i < ag.gizliKatman.Length; i++)
            {
                GizliNoron gizliNoron = ag.gizliKatman[i];
                for (int j = 0; j < gizliNoron.girisBaglanti.Length - 1; j++)
                {
                    lv_girisGizliAgirliklar.Items[j].SubItems[i + 1].Text =
gizliNoron.girisBaglanti[j].agirlik.ToString();
                }
                lv_biasGizliAgirliklar.Items[0].SubItems[i + 1].Text =
gizliNoron.girisBaglanti[gizliNoron.girisBaglanti.Length - 1].agirlik.ToString();
            }
            for (int i = 0; i < ag.cikisKatman.Length; i++)
            {
                CikisNoron cikisNoron = ag.cikisKatman[i];
                for (int j = 0; j < cikisNoron.girisBaglanti.Length - 1; j++)
                {
                    lv_cikisGizliAgirliklar.Items[i].SubItems[j + 1].Text =
cikisNoron.girisBaglanti[j].agirlik.ToString();
                }
                lv_biasGizliAgirliklar.Items[0].SubItems[ag.gizliKatman.Length +
i + 1].Text = cikisNoron.girisBaglanti[cikisNoron.girisBaglanti.Length -
1].agirlik.ToString();
            }
        }
        private void bt_testAg_Click(object sender, EventArgs e)
        {
            double[] girisDegerler = new double[ag.girisKatman.Length];
            for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value); i++)
            {
                girisDegerler[i] = Convert.ToDouble(tDiziG[i+1].Text);
            }
            double[] cikisDegerler = ag.calistirAg(girisDegerler);
            for (int i = 0; i < Convert.ToInt32(nud_cikisNoronlar.Value); i++)
```

```
            {
                tDiziC[i+1].Text = cikisDegerler[i].ToString();
                tDiziCKopya[i + 1].Text = cikisDegerler[i].ToString();
            }
            for (int z=Convert.ToInt32(nud_girisNoronlar.Value)+1;
z<=Convert.ToInt32(nud_girisNoronlar.Value)+Convert.ToInt32(nud_cikisNoronlar.Val
ue); z++)
            {
                for(int y=0; y<nud_cikisNoronlar.Value;y++)
                {
                    if ((cikisDegerler[y].ToString() == tDiziGKopya[z].Text) &&
(z - y == nud_girisNoronlar.Value + 1) && cikisDegerler[y] == 1)
                    {
                        lb_sonuc.Text = "TEŞHİS " + (y + 1) + " başarılı tahmin";
                    }
                    if ((cikisDegerler[y].ToString() != tDiziGKopya[z].Text) &&
(z - y == nud_girisNoronlar.Value + 1) && cikisDegerler[y] == 1)
                    {
                        lb_sonuc.Text = "TEŞHİS " + (y + 1) + " başarısız
tahmin";
                    }
                }
            }
            lb_durum.Text = "Test verisi başarı ile işlendi.";
            lb_durum1.Text = "Test verisi başarı ile işlendi.";
        }
        private void bt_egitAg_Click(object sender, EventArgs e)
        {
            bt_egitAg.Enabled = false;
            progB_Iterasyon.Maximum = Convert.ToInt32(tb_hedefIterasyon.Text);
            progB_Iterasyon.Value = 0;
            int index = 0;
            bool iterationContinue = true;
            DateTime timeStart = DateTime.Now;
            do
            {
                for (int i = 0; i < egitimGirisDegerler.GetLength(0); i++) // Bir
epok
                {
                    double[] girisDegerler = new double[ag.girisKatman.Length];
                    double[] hedefDegerler = new double[ag.cikisKatman.Length];
                    for (int j = 0; j < girisDegerler.Length; j++)
                    {
                        girisDegerler[j] = egitimGirisDegerler[i, j];
                    }
                    for (int j = 0; j < hedefDegerler.Length; j++)
                    {
                        hedefDegerler[j] = egitimHedefDegerler[i, j];
                    }
                    ag.egitAg(girisDegerler, hedefDegerler);
                }
                index++;
                if (rb_Iterasyon.Checked)
                {
                    progB_Iterasyon.Value++;
                    if (index == Convert.ToInt32(tb_hedefIterasyon.Text))
                    {
                        iterationContinue = false;
                    }
                }
                else if (rb_hataOran.Checked)
                {
```

```
                    progB_Iterasyon.Maximum = index;
                    progB_Iterasyon.Value++;
                    if (ag.agHata <= ag.hedefAgHata)
                    {
                        iterationContinue = false;
                        tb_iteration.Text = index.ToString();
                    }
                }
            } while (iterationContinue);
            TimeSpan timeSpan = DateTime.Now.Subtract(timeStart);
            tb_egitmeSure.Text = timeSpan.TotalSeconds.ToString();
            tb_agHata.Text = ag.agHata.ToString();
            doldurAgirliklarListeye();
            bt_okuTestDegerlerDosyadan.Enabled = true;
            lb_durum.Text = "Ağ eğitimi tamamlandı.";
            lb_durum1.Text = "Ağ eğitimi tamamlandı.";
            lb_yolGoster.Text = "Test verilerini yükleyiniz...";
        }
        private void bt_okuEgitmeDegerlerDosyadan_Click(object sender, EventArgs
e)
        {
            bt_okuEgitmeDegerlerDosyadan.Enabled = false;
            if (openFileDialog2.ShowDialog() != DialogResult.Cancel)
            {
                string[] lines = File.ReadAllLines(openFileDialog2.FileName);
                egitimGirisDegerler = new double[lines.Length,
ag.girisKatman.Length];
                egitimHedefDegerler = new double[lines.Length,
ag.cikisKatman.Length];
                for (int i = 0; i < lines.Length; i++)
                {
                    string[] temp = lines[i].Split('\t');
                    for (int j = 0; j < ag.girisKatman.Length; j++)
                    {
                        egitimGirisDegerler[i, j] = Convert.ToDouble(temp[j]);
                    }
                    for (int j = 0; j < ag.cikisKatman.Length; j++)
                    {
                        egitimHedefDegerler[i, j] = Convert.ToDouble(temp[j +
ag.girisKatman.Length]);
                    }
                }
                bt_egitAg.Enabled = true;
                lb_durum.Text = "Eğitme verileri dosyadan okundu. Eğitme için
hazır!";
                lb_durum1.Text = "Eğitme verileri dosyadan okundu. Eğitme için
hazır!";
                lb_yolGoster.Text = "Ağı eğitiniz...";
            }
        }
        private void bt_yukleDosyadanAgirliklar_Click(object sender, EventArgs e)
        {
            if (openFileDialog2.ShowDialog() != DialogResult.Cancel)
            {
                string[] lines = File.ReadAllLines(openFileDialog2.FileName);
                if (lines.Length == ag.girisKatman.Length + ag.cikisKatman.Length
+ 1)
                {
                    double[,] inputHiddenBaglantiWeights = new
double[ag.girisKatman.Length, ag.gizliKatman.Length];
                    double[,] outputHiddenBaglantiWeights = new
double[ag.cikisKatman.Length, ag.gizliKatman.Length];
```

```
                    double[] biasBaglantiWeights = new
double[ag.gizliKatman.Length + ag.cikisKatman.Length];
                    for (int i = 0; i < inputHiddenBaglantiWeights.GetLength(0);
i++)
                    {
                        string[] temp = lines[i].Split('\t');
                        for (int j = 0; j < temp.Length; j++)
                        {
                            inputHiddenBaglantiWeights[i, j] =
Convert.ToDouble(temp[j]);
                        }
                    }
                    for (int i = 0; i < outputHiddenBaglantiWeights.GetLength(0);
i++)
                    {
                        string[] temp = lines[i +
inputHiddenBaglantiWeights.GetLength(0)].Split('\t');
                        for (int j = 0; j < temp.Length; j++)
                        {
                            outputHiddenBaglantiWeights[i, j] =
Convert.ToDouble(temp[j]);
                        }
                    }
                    string[] biasTemp = lines[lines.Length - 1].Split('\t');
                    for (int j = 0; j < biasTemp.Length; j++)
                    {
                        biasBaglantiWeights[j] = Convert.ToDouble(biasTemp[j]);
                    }
                    ag.ataIceAktarilanAgirlik(inputHiddenBaglantiWeights,
outputHiddenBaglantiWeights, biasBaglantiWeights);
                    doldurAgirliklarListeye();
                    bt_kaydetAgirliklarDosyaya.Enabled = true;
                    bt_okuEgitmeDegerlerDosyadan.Enabled = true;
                    lb_durum.Text = "Ağırlıklar başarı ile içe aktarıldı.";
                    lb_durum1.Text = "Ağırlıklar başarı ile içe aktarıldı.";
                    lb_yolGoster.Text = "Eğitim verilerini yükleyiniz...";
                }
            }
        }
        private void bt_kaydetAgirliklarDosyaya_Click(object sender, EventArgs e)
        {
            double[,] girisGizliAgirliklar = new double[ag.girisKatman.Length,
ag.gizliKatman.Length];
            double[,] cikisGizliAgirliklar = new double[ag.cikisKatman.Length,
ag.gizliKatman.Length];
            double[] biasAgirliklar = new double[ag.gizliKatman.Length +
ag.cikisKatman.Length];
            for (int i = 0; i < ag.gizliKatman.Length; i++)
            {
                GizliNoron gizliNoron = ag.gizliKatman[i];
                for (int j = 0; j < gizliNoron.girisBaglanti.Length - 1; j++)
                {
                    girisGizliAgirliklar[j, i] =
gizliNoron.girisBaglanti[j].agirlik;
                }
                biasAgirliklar[i] =
gizliNoron.girisBaglanti[gizliNoron.girisBaglanti.Length - 1].agirlik;
            }
            for (int i = 0; i < ag.cikisKatman.Length; i++)
            {
                CikisNoron cikisNoron = ag.cikisKatman[i];
                for (int j = 0; j < cikisNoron.girisBaglanti.Length - 1; j++)
```

```
                {
                    cikisGizliAgirliklar[i, j] =
cikisNoron.girisBaglanti[j].agirlik;
                }
                biasAgirliklar[ag.gizliKatman.Length + i] =
cikisNoron.girisBaglanti[cikisNoron.girisBaglanti.Length - 1].agirlik;
            }
            string[] exportWeight = new string[ag.girisKatman.Length +
ag.cikisKatman.Length + 1];
            for (int i = 0; i < girisGizliAgirliklar.GetLength(0); i++)
            {
                for (int j = 0; j < girisGizliAgirliklar.GetLength(1); j++)
                {
                    exportWeight[i] += girisGizliAgirliklar[i, j];
                    if (j != (girisGizliAgirliklar.GetLength(1) - 1))
                        exportWeight[i] += "\t";
                }
            }
            for (int i = 0; i < cikisGizliAgirliklar.GetLength(0); i++)
            {
                for (int j = 0; j < cikisGizliAgirliklar.GetLength(1); j++)
                {
                    exportWeight[i + girisGizliAgirliklar.GetLength(0)] +=
cikisGizliAgirliklar[i, j];
                    if (j != (cikisGizliAgirliklar.GetLength(1) - 1))
                        exportWeight[i + girisGizliAgirliklar.GetLength(0)] +=
"\t";
                }
            }
            for (int i = 0; i < biasAgirliklar.Length; i++)
            {
                exportWeight[exportWeight.Length - 1] += biasAgirliklar[i];
                if (i != biasAgirliklar.Length - 1)
                    exportWeight[exportWeight.Length - 1] += "\t";
            }
            if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
                System.IO.File.WriteAllLines(saveFileDialog1.FileName,
exportWeight);
                lb_durum.Text = "Ağırlıklar başarı ile dışa aktarıldı.";
                lb_durum1.Text = "Ağırlıklar başarı ile dışa aktarıldı.";
            }
        }
        private void AnaSayfa_Load(object sender, EventArgs e)
        {
            tabControl1.SelectedIndex = 0;
            panel1.VerticalScroll.Value = 1;
            panel2.VerticalScroll.Value = 1;
            tb_hedefIterasyon.Enabled = true;
            tb_hedefAgHata.Enabled = false;
        }
        private void bt_okuTestDegerlerDosyadan_Click(object sender, EventArgs e)
        {
            bt_okuTestDegerlerDosyadan.Enabled = false;
            if (openFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
                string[] lines = File.ReadAllLines(openFileDialog1.FileName);
                listBox1.DataSource = lines;
                Clipboard.SetText(listBox1.SelectedItem.ToString());
                bt_testAg.Enabled = true;
                bt_tumunuTestAg.Enabled = true;
```

```
                lb_durum.Text = "Test verileri dosyadan okundu. Test için
hazır!";
                lb_durum1.Text = "Test verileri dosyadan okundu. Test için
hazır!";
                lb_yolGoster.Text = "Verileri test ediniz...";
            }
        }
        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (listBox1.SelectedItem!=null)
            {
                string testverisi = listBox1.SelectedItem.ToString();
                string[] test = testverisi.Split('\t');
                for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value);
i++)
                {
                    tDiziG[i + 1].Text = test[i];
                }
                for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value) +
Convert.ToInt32(nud_cikisNoronlar.Value); i++)
                {
                    tDiziGKopya[i + 1].Text = test[i];
                }
            }
        }
        private void bt_tumunuTestAg_Click(object sender, EventArgs e)
        {
            sayacTest = 0;
            lbc = listBox1.Items.Count;
            for (int j = 0; j < listBox1.Items.Count; j++)
            {
                string testverisi = listBox1.Items[j].ToString();
                string[] test = testverisi.Split('\t');
                for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value);
i++)
                {
                    tDiziG[i + 1].Text = test[i];
                }
                for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value) +
Convert.ToInt32(nud_cikisNoronlar.Value); i++)
                {
                    tDiziGKopya[i + 1].Text = test[i];
                }
                double[] girisDegerler = new double[ag.girisKatman.Length];
                for (int i = 0; i < Convert.ToInt32(nud_girisNoronlar.Value);
i++)
                {
                    girisDegerler[i] = Convert.ToDouble(tDiziG[i + 1].Text);
                }
                double[] cikisDegerler = ag.calistirAg(girisDegerler);
                for (int i = 0; i < Convert.ToInt32(nud_cikisNoronlar.Value);
i++)
                {
                    tDiziCKopya[i + 1].Text = cikisDegerler[i].ToString();
                }
                int icsayac = 0;
                for (int z = Convert.ToInt32(nud_girisNoronlar.Value) + 1; z <=
Convert.ToInt32(nud_girisNoronlar.Value) +
Convert.ToInt32(nud_cikisNoronlar.Value); z++)
                {
                    for (int y = 0; y < nud_cikisNoronlar.Value; y++)
                    {
```

```
                        if (cikisDegerler[y].ToString() == tDiziGKopya[z].Text &&
z-y==nud_girisNoronlar.Value+1)
                        {
                            icsayac = icsayac + 1;
                            if (icsayac == nud_cikisNoronlar.Value)
                                sayacTest = sayacTest + 1;
                        }
                    }
                }
            }
            double yuzde = 100*sayacTest/lbc;
            lb_durum.Text = "Tüm test verileri başarı ile işlendi.";
            lb_durum1.Text = "Tüm test verileri başarı ile işlendi.";
            lb_sonuc.Text = lbc.ToString()+" veride "+sayacTest.ToString()+"
doğru tahmin\nile "+yuzde.ToString()+"% başarı sağlanmıştır";
        }
        private void label26_Click(object sender, EventArgs e)
        {
        }
        private void rb_Iterasyon_CheckedChanged(object sender, EventArgs e)
        {
        }
        private void rb_hataOran_CheckedChanged(object sender, EventArgs e)
        {
        }
        private void rb_Iterasyon_MouseClick(object sender, MouseEventArgs e)
        {
            rb_Iterasyon.Checked = true;
            if (rb_hataOran.Checked == true)
            {
                rb_hataOran.Checked = false;
            }
            if (rb_Iterasyon.Checked == true)
            {
                tb_hedefIterasyon.Enabled = true;
                tb_hedefAgHata.Enabled = false;
            }
        }
        private void rb_hataOran_MouseClick(object sender, MouseEventArgs e)
        {
            rb_hataOran.Checked = true;
            if (rb_Iterasyon.Checked == true)
            {
                rb_Iterasyon.Checked = false;
            }
            else if(rb_hataOran.Checked==true)
            {
                tb_hedefAgHata.Enabled = true;
                tb_hedefIterasyon.Enabled = false;
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class Ag
    {
```

```
        public GirisNoron[] girisKatman;
        public GizliNoron[] gizliKatman;
        public CikisNoron[] cikisKatman;
        public BiasNoron biasNoron;
        double egitmeOran;
        double momentumOran;
        public double agHata;
        public double hedefAgHata;
        public Ag(int girisNoronlarSayi, int gizliNoronlarSayi, int
cikisNoronlarSayi, double egitmeOran, double momentumOran, double hedefAgHata)
        {
            girisKatman = new GirisNoron[girisNoronlarSayi];
            gizliKatman = new GizliNoron[gizliNoronlarSayi];
            cikisKatman = new CikisNoron[cikisNoronlarSayi];
            this.egitmeOran = egitmeOran;
            this.momentumOran = momentumOran;
            this.hedefAgHata = hedefAgHata;
            olusturNoron();
            olusturBaglanti();
            olusturAg();
        }
        public void olusturNoron()
        {
            for (int i = 0; i < girisKatman.Length; i++)
                girisKatman[i] = new GirisNoron();
            for (int i = 0; i < gizliKatman.Length; i++)
                gizliKatman[i] = new GizliNoron();
            for (int i = 0; i < cikisKatman.Length; i++)
                cikisKatman[i] = new CikisNoron();
            biasNoron = new BiasNoron();
        }
        public void olusturBaglanti()
        {
            foreach(GirisNoron girisNoron in girisKatman)
                girisNoron.cikisBaglanti=new Baglanti[gizliKatman.Length];
            foreach (GizliNoron gizliNoron in gizliKatman)
            {
                gizliNoron.girisBaglanti = new Baglanti[girisKatman.Length + 1];
//+1 Bias için.
                gizliNoron.cikisBaglanti = new Baglanti[cikisKatman.Length];
            }
            foreach (CikisNoron cikisNoron in cikisKatman)
                cikisNoron.girisBaglanti = new Baglanti[gizliKatman.Length + 1];
//+1 Bias için.
            biasNoron.cikisBaglanti = new Baglanti[gizliKatman.Length +
cikisKatman.Length];
        }
        public void olusturAg()
        {
            for (int i=0 ; i<girisKatman.Length;i++)
            {
                for (int j = 0; j < gizliKatman.Length; j++)
                {
                    gizliKatman[j].girisBaglanti[i] = new Baglanti();
                    girisKatman[i].cikisBaglanti[j] =
gizliKatman[j].girisBaglanti[i];
                }
            }
            for (int i = 0; i < gizliKatman.Length; i++)
            {
                for (int j = 0; j < cikisKatman.Length; j++)
                {
```

```
                    cikisKatman[j].girisBaglanti[i] = new Baglanti();
                    gizliKatman[i].cikisBaglanti[j] =
cikisKatman[j].girisBaglanti[i];
                }
            }
            for (int i = 0; i < gizliKatman.Length; i++)
            {
                gizliKatman[i].girisBaglanti[gizliKatman[i].girisBaglanti.Length
- 1] = new Baglanti();
                biasNoron.cikisBaglanti[i] =
gizliKatman[i].girisBaglanti[gizliKatman[i].girisBaglanti.Length - 1];
            }
            for (int i = 0; i < cikisKatman.Length; i++)
            {
                cikisKatman[i].girisBaglanti[cikisKatman[i].girisBaglanti.Length
- 1] = new Baglanti();
                biasNoron.cikisBaglanti[gizliKatman.Length + i] =
cikisKatman[i].girisBaglanti[cikisKatman[i].girisBaglanti.Length - 1];
            }
        }
        public void ataRastgeleAgirlik()
        {
            Random rastgele = new Random();
            foreach (GizliNoron gizliNoron in gizliKatman)
            {
                foreach (Baglanti baglanti in gizliNoron.girisBaglanti)
                {
                    if (rastgele.Next(-1, 1) < 0)
                        baglanti.agirlik = rastgele.NextDouble() * -1;
                    else
                        baglanti.agirlik = rastgele.NextDouble();
                }
            }
            foreach (CikisNoron cikisNoron in cikisKatman)
            {
                foreach (Baglanti baglanti in cikisNoron.girisBaglanti)
                {
                    if (rastgele.Next(-1, 1) < 0)
                        baglanti.agirlik = rastgele.NextDouble() * -1;
                    else
                        baglanti.agirlik = rastgele.NextDouble();
                }
            }
        }
        public void ataIceAktarilanAgirlik(double[,] girisGizliAgirliklar,
double[,] cikisGizliAgirliklar, double[] biasAgirliklar)
        {
            for (int i = 0; i < gizliKatman.Length; i++)
            {
                GizliNoron gizliNoron = gizliKatman[i];

                for (int j = 0; j < gizliNoron.girisBaglanti.Length - 1; j++)
                {
                    gizliNoron.girisBaglanti[j].agirlik = girisGizliAgirliklar[j,
i];
                }

                gizliNoron.girisBaglanti[gizliNoron.girisBaglanti.Length -
1].agirlik = biasAgirliklar[i];
            }
            for (int i = 0; i < cikisKatman.Length; i++)
            {
```

```
                CikisNoron cikisNoron = cikisKatman[i];

                for (int j = 0; j < cikisNoron.girisBaglanti.Length - 1; j++)
                {
                    cikisNoron.girisBaglanti[j].agirlik = cikisGizliAgirliklar[i,
j];
                }
                cikisNoron.girisBaglanti[cikisNoron.girisBaglanti.Length -
1].agirlik = biasAgirliklar[gizliKatman.Length + i];
            }
        }
        public double[] calistirAg(double[] girisDegerler)
        {
            if (girisDegerler.Length == girisKatman.Length)
            {
                for (int i = 0; i < girisDegerler.Length; i++)
                {
                    girisKatman[i].transferDeger = girisDegerler[i];
                }
                for (int i = 0; i < gizliKatman.Length; i++)
                {
                    double net = 0;
                    for (int j = 0; j < gizliKatman[i].girisBaglanti.Length - 1;
j++)
                    {
                        net += girisKatman[j].transferDeger *
gizliKatman[i].girisBaglanti[j].agirlik;
                    }
                    net += biasNoron.transferDeger *
gizliKatman[i].girisBaglanti[gizliKatman[i].girisBaglanti.Length - 1].agirlik;
                    gizliKatman[i].netDeger = net;
                    gizliKatman[i].hesaplaTransfer();
                }
                for (int i = 0; i < cikisKatman.Length; i++)
                {
                    double net = 0;
                    for (int j = 0; j < cikisKatman[i].girisBaglanti.Length - 1;
j++)
                    {
                         net += gizliKatman[j].transferDeger *
cikisKatman[i].girisBaglanti[j].agirlik;
                    }
                    net += biasNoron.transferDeger *
cikisKatman[i].girisBaglanti[cikisKatman[i].girisBaglanti.Length - 1].agirlik;
                    cikisKatman[i].netDeger = net;
                    cikisKatman[i].hesaplaTransfer();
                }
                double[] cikisDegerler = new double[cikisKatman.Length];

                for (int i = 0; i < cikisDegerler.Length; i++)
                {
                    cikisDegerler[i] = cikisKatman[i].transferDeger;
                }
                return cikisDegerler;
            }
            return null;
        }
        public void egitAg(double[] girisDegerler, double[] hedefDegerler)
        {
            double[] cikisDegerler = calistirAg(girisDegerler);
            hesaplaAgHata(cikisDegerler, hedefDegerler);
            for (int i = 0; i < cikisKatman.Length; i++)
```

```
            {
                cikisKatman[i].hesaplaHata(hedefDegerler[i]);
            }
            for (int i = 0; i < gizliKatman.Length; i++)
            {
                gizliKatman[i].hesaplaHata(cikisKatman);
            }
            for (int i = 0; i < cikisKatman.Length; i++)
            {
                for (int j = 0; j < cikisKatman[i].girisBaglanti.Length - 1; j++)
                {

cikisKatman[i].girisBaglanti[j].guncelleAgirlik(gizliKatman[j].transferDeger,
cikisKatman[i].hataDeger, egitmeOran, momentumOran);
                }
                cikisKatman[i].girisBaglanti[cikisKatman[i].girisBaglanti.Length
- 1].guncelleAgirlik(biasNoron.transferDeger, cikisKatman[i].hataDeger,
egitmeOran, momentumOran);
            }
            for (int i = 0; i < gizliKatman.Length; i++)
            {
                for (int j = 0; j < gizliKatman[i].girisBaglanti.Length - 1; j++)
                {

gizliKatman[i].girisBaglanti[j].guncelleAgirlik(girisKatman[j].transferDeger,
gizliKatman[i].hataDeger, egitmeOran, momentumOran);
                }
                gizliKatman[i].girisBaglanti[gizliKatman[i].girisBaglanti.Length
- 1].guncelleAgirlik(biasNoron.transferDeger, gizliKatman[i].hataDeger,
egitmeOran, momentumOran);
            }
        }
        public void hesaplaAgHata(double[] cikisDegerler, double[] hedefDegerler)
        {
            double hata=0;
            if (cikisDegerler.Length == hedefDegerler.Length)
            {
                for (int i = 0; i < cikisDegerler.Length; i++)
                {
                    hata += Math.Abs(hedefDegerler[i] - cikisDegerler[i]);
                }
                agHata = hata / 2;
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class Baglanti
    {
        public double agirlik = 0;
        public void guncelleAgirlik(double transferDeger, double hataDeger,
double egitmeOran, double momentumOran)
        {
            agirlik = agirlik + egitmeOran * transferDeger * hataDeger + agirlik
* momentumOran;
        }
```

```
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class BiasNoron
    {
        public Baglanti[] cikisBaglanti;
        public double netDeger = 1;
        public double transferDeger = 0;
        public BiasNoron()
        {
            aktivasyonFonksiyon();
        }
        public void aktivasyonFonksiyon()
        {
            transferDeger = 1 / (1 + Math.Exp(-netDeger));
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class CikisNoron
    {
        public Baglanti[] girisBaglanti;
        public double netDeger = 0;
        public double transferDeger = 0;
        public double hataDeger = 0;
        public void hesaplaTransfer()
        {
            aktivasyonFonksiyon();
        }
        public void aktivasyonFonksiyon()
        {
            transferDeger = 1 / (1 + Math.Exp(-netDeger));
        }
        public void hesaplaHata(double hedefCikis)
        {
            hataDeger = transferDeger * (1 - transferDeger) * (hedefCikis -
transferDeger);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class GirisNoron
    {
        public Baglanti[] cikisBaglanti;
```

```
        public double transferDeger = 0;
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Hastalık_Teşhisi
{
    public class GizliNoron
    {
        public Baglanti[] girisBaglanti;
        public Baglanti[] cikisBaglanti;
        public double netDeger = 0;
        public double transferDeger = 0;
        public double hataDeger = 0;
        public void hesaplaTransfer()
        {
            aktivasyonFonksiyon();
        }
        public void aktivasyonFonksiyon()
        {
            transferDeger = 1 / (1 + Math.Exp(-netDeger));
        }
        public void hesaplaHata(CikisNoron[] cikisKatman)
        {
            double hata = 0;
            for (int i = 0; i < cikisKatman.Length; i++)
            {
                hata += cikisKatman[i].hataDeger * cikisBaglanti[i].agirlik;
            }
            hataDeger = transferDeger * (1 - transferDeger) * hata;
        }
    }
}
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Kadir TOK
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Kayseri 03.05.1989
Telefon	**:**	+90 (506) 428 34 68
Faks	**:**	-
e-mail	**:**	kadirtok@selcuk.edu.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Fen Bilimleri – Aksaray Fen Lisesi, Merkez, Aksaray	**2006**
Üniversite	:	Bilgisayar Sistemleri Öğretmenliği – Selçuk Üniversitesi, Selçuklu, Konya	**2011**
		Bilgisayar Mühendisliği – Selçuk Üniversitesi, Selçuklu, Konya	**2015**

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2014-2015	Bilgi İşlem Daire Başkanlığı – Selçuk Üniversitesi, Selçuklu, Konya	Bilgisayar Mühendisi

UZMANLIK ALANI

Programlama Dilleri	:	C#, C/C++, Delphi, Java, ASP.NET, MVC, HTML
Veritabanı	:	Oracle, MySQL
Sistemler	:	Windows (XP, Vista, 7, 8, 10)
Paket Programlar	:	RAD Studio, Visual Studio, Dreamweaver

YABANCI DİLLER

İngilizce: Okuma: Orta Yazma: Orta Konuşma: Orta

YAYINLAR

Tok, K. ve Saritas, I., 2016, The Usage Of Artificial Neural Networks Method In The Diagnosis Of Rheumatoid Arthritis, *International Journal of Intelligent Systems and Applications in Engineering*, 4 (4), 119-121.