DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI KULLANILARAK KALP HASTALIĞININ ÖNGÖRÜLMESİ

Ayşe ARI

Ekim, 2019
İZMİR

YAPAY SİNİR AĞLARI KULLANILARAK KALP HASTALIĞININ ÖNGÖRÜLMESİ

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Bilgisayar Bilimleri Anabilim Dalı

Ayşe ARI

Ekim, 2019
İZMİR

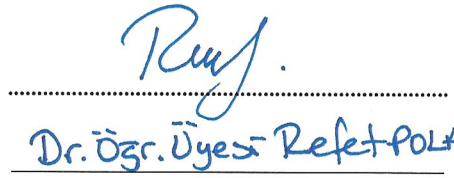
YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

AYŞE ARI, tarafından **DOÇ.DR. MURAT ERŞEN BERBERLER** yönetiminde hazırlanan "**YAPAY SİNİR AĞLARI KULLANILARAK KALP HASTALIĞININ ÖNGÖRÜLMESİ**" başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

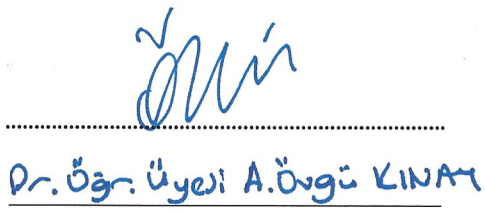
Doç.Dr. Murat Erşen BERBERLER

Danışman



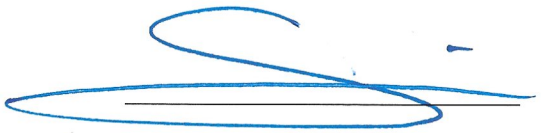
Dr. Öğr. Üyesi Refet POLAT

Jüri Üyesi



Dr. Öğr. Üyesi A. Övgü KINAY

Jüri Üyesi



Prof.Dr. Kadriye ERTEKİN
Müdür
Fen Bilimleri Enstitüsü

TEŞEKKÜR

Yüksek lisans eğitimim ve tez çalışmam boyunca önümde yeni ufukların açılmasına yardımcı olan, bilgi birikimi, deneyimi, sabrı ve anlayışı ile her zaman yanımda olan, modum düştüğü anda beni yüreklendiren saygıdeğer danışman hocam Doç. Dr. Murat Erşen BERBERLER'e desteği ve katkısı için teşekkür ederim.

Çalışmam süresince bana ayırdığı zaman ile bilgi ve görüşlerinden yararlandığım sevgili hocam Dr. Öğr. Üyesi Mete EMİNAĞAOĞLU'na değerli önerileri ve katkıları için teşekkür ederim.

Yüksek lisans eğitimim ve tez çalışmamın her aşamasında gösterdikleri anlayış ve ilgiyle her zaman yanımda olan, güler yüzlerini hiçbir zaman eksik etmeyen başta bölüm başkanım Prof. Dr. Efendi NASİBOĞLU olmak üzere tüm bölüm hocalarıma, araştırma görevlilerine, bölüm sekreteri ve çalışanlarına ayrı ayrı teşekkürü bir borç bilirim.

Yüksek lisans eğitimimde birlikte çalışmaktan ve öğrenmekten keyif aldığım, zorlukların üstesinden beraber geldiğimiz sevgili arkadaşlarım Çağla Efsun, Burak ve Mehmet'e teşekkür ederim.

Zorlu tez çalışma sürecim boyunca, her zaman olduğu gibi moral ve desteğini esirgemeyen biricik arkadaşım Tuğba ÖZDEMİR'e teşekkür ederim.

Son olarak, hayatımın her anında olduğu gibi yüksek lisans ve tez çalışmam boyunca da beni sabırla destekleyen ve moral veren, varlığı ile huzur bulduğum sevgili aileme sonsuz teşekkür ederim.

Ayşe ARI

YAPAY SİNİR AĞLARI KULLANILARAK KALP HASTALIĞININ ÖNGÖRÜLMESİ

ÖZ

Kalp hastalığı, günümüz sağlık sorunlarının başında gelmektedir. Hastalığın tedavisinde erken teşhisin önemi oldukça büyüktür. Yapay sinir ağları (YSA), birçok alanda olduğu gibi hastalık teşhisinde de kullanılan yapay zeka yöntemlerinden biridir. Bu çalışmada, kalp hastalığının teşhisi için YSA modeli önerilmiştir. En doğru teşhisi yapabilecek bir ağ geliştirmek için, ağın topolojisinin iyi seçilmesi gerekmektedir. Çalışmada ağın topolojisinin belirlenmesi için ağ mimarisi ve parametrelerin sürekli olarak değiştirilip, ağın eğitilip test edildiği program MATLAB dilinde yazılmıştır. Programdan elde edilen değerler üzerine yoğunlaşılıp en iyi doğruluk oranını verecek mimari yapısı ve parametreleri denenmiştir. Çalışmada, ayrıca tahmin ve sınıflandırma problemleri için YSA modelinin kullanıcının belirlediği parametrelerde eğitilip test edildiği esnek bir yazılım C#.NET programlama dilinde geliştirilmiştir. Yazılımda eğitilen ağın ağırlık değerleri saklanmakta ve bir sonraki eğitim başlangıç ağırlıkları, öğrendiği ağırlıklar kullanarak gerçekleştirildiğinde başarımın arttırıldığı gözlemlenmiştir. Ayrıca yazılım, eğitim esnasında öğrenme oranı ve momentum değerinin bozunum işlemine tabi tutulmasına olanak sağlamaktadır. Kalp hastalığı için belirlenen en uygun ağ topolojisi için, öğrenme oranına bozunum işlemi uygulanarak eğitilip test edildiğinde doğruluk oranının arttığı gözlemlenmiştir.

Anahtar kelimeler: Kalp hastalığı, yapay sinir ağları, tahmin, sınıflandırma

PREDICTION OF HEART DISEASE USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Heart disease is one of the leading daily health problems. Early diagnosis is great importance in the treatment of the disease. Artificial neural networks (ANN) is one of the artificial intelligence methods used in the disease diagnosis as in many areas. In this study, the ANN model is proposed for the diagnosis of heart disease. To develop a network that can make the most accurate diagnosis, the topology of the network should be chosen well. To determine the topology of the network in this study, the program in which the network architecture and parameters are continuously changed and the network is trained and tested, is written in MATLAB. The values obtained from the program have been focused on and the architectural structure and parameters that will give the best accuracy rate have been tried. In this study, also flexible software is developed in C# .NET programming language, where the ANN model for prediction and classification problems is trained and tested in user-defined parameters. The weight values of the network trained in software are stored, and it has been observed that performance is increased when the next training initial weights are performed using the weights learned. Also, the software allows the decay process for the learning rate and momentum value during training. For optimal network topology determined for heart disease, it has been observed that the accuracy rate increases when it is trained and tested by applying decay to the learning rate.

Keywords: Heart disease, artificial neural networks, prediction, classification

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

BÖLÜM BİR
GİRİŞ

1.1 Problemin Sunumu

Kalp hastalıkları kalp damar hastalıklarından olup ölüm riskinin en çok olduğu hastalıklardan biridir. Dünya Sağlık Örgütünün verilerine göre 2016 yılında gerçekleşen ölümlerin nedenlerinin başında kalp hastalığı gelmektedir. İskemik kalp hastalığı ve inme 2016 yılında 15,2 milyon ölümle sonuçlanan dünyanın en büyük sağlık sorunlarıdır. Bu hastalıklar son 15 yılda dünyanın önde gelen ölüm nedenleri olmayı sürdürmektedir World Health Organization (2018). Dahası Dünya Sağlık Örgütü'ne göre 2030 yılına kadar kalp hastalığı nedeniyle gerçekleşecek olan ölümlerin sayısının 23,6 milyona ulaşacağı öngörülmektedir Dangare ve Apte (2012). Bu nedenledir ki kalp hastalığı riskini azaltmak için önlem alınmalıdır. Hastalık teşhisi tıpta zor bir görevdir. Teşhis genellikle bir hastanın bulgu, semptom ve fizik muayenesine dayanır. Neredeyse tüm doktorlar bilgi ve deneyimlerine dayanarak kalp hastalığını öngörmektedir. Ancak doğru tanı için tek bir insan zekası yeterli olmayabilir. Hastalar tarafından verilen bilgiler gereksiz ve birbiriyle ilişkili semptomları içerebilir ve özellikle hastaların muzdarip olduğu şikayetler aynı kategoriden birçok hastalığı işaret edebilir. Bu gibi durumlar doktorların doğru teşhisi koymasını zorlaştırmaktadır. Son zamanlarda bilgisayar teknolojileri ve makine öğrenme teknikleri hastalıkların erken teşhisinde doktorlara yardımcı olmak için kullanılmaktadır. Yapay sinir ağları da hastalık teşhisinde sıklıkla kullanılan makine öğrenme tekniklerinden biridir.

1.2 Araştırmanın Önemi

Bu tez, kalp hastalarının verilerinin kullanılarak makine öğrenmesi yöntemlerinden biri olan yapay sinir ağları teknolojisi ile kalp hastalığının tahmini üzerine odaklanmaktadır. Hastalığın olup olmama durumu ve olması halinde hastalığın

derecesine göre sınıflandırılması üzerine çalışılmıştır. Hastalığın tahmini ve sınıflandırılması doktorların hastalarını izleme ve doğru kararlar almalarına yardımcı olması açısından önem taşımaktadır.

1.3 Araştırmanın Amacı

Kalp hastalığının tahmini için en doğru ağ yapısını ve paremetrelerini seçerek en yüksek doğrulukta tahmin yapmak çalışmanın ana amacıdır. Kalp hastalığını belirlemede yüksek doğruluk tahminleri almak doktorların teşhis süresince optimal doğruluk sağlamasına yardımcı olacaktır.

BÖLÜM İKİ
KALP HASTALIKLARI

Kalp hastalığı, kalbi ve kan damarlarını etkileyen çeşitli hastalıkların, durumların ve bozuklukların genel adıdır. Yaşam kalbin verimli çalışmasına bağlıdır. Kalbin çalışmasında oluşacak herhangi bir sorun doğrudan beyin, böbrek veya vücudun diğer bölümlerini etkileyecektir.

Her yıl 17 milyondan fazla insan kardiyovasküler hastalıktan (KVH) dolayı ölmektedir. Bu sayı tüm dünya ölümlerinin %31'ine tekabül etmektedir. Öncelikle kalp krizi ve felci olarak görülen bu hastalıkları tetikleyen faktörler; sağlıksız beslenme, tütün kullanımı, fiziksel hareketsizlik ve alkol kullanımıdır. Bunlar sırayla insanlarda yüksek tansiyon, yüksek kan şekeri, aşırı kilo ve obeziteye neden olup kalp sağlığı için tehdit oluşturmaktadırlar World Health Organization (bt).

2.1 Kalp Hastalıklarının Türleri

Kalp hastalıklarından bazıları aşağıdaki gibi sıralanabilir World Heart Federation (2017).

- **İskemik kalp rahatsızlığı:** Koroner arterlerin daralması ve dolayısıyla kalpteki kan akışının azalmasına neden olan kalp rahatsızlıklarıdır.
  - Anjina; yoğunlukla kalp kasında kan akışının azalması veya spazm nedeniyle göğüste meydana gelen ağrı olarak kendini gösterir. Kan, vücudumuzun etrafında oksijen taşır ve kalbi oksijenden yoksun bırakmanın ciddi sonuçları vardır. Anjina kalbi besleyen kan damarlarının daralması ve / veya tıkanmasından kaynaklanır. Anjina'nın tipik ağrısı göğüste olmakla birlikte genellikle sol kola, omuza veya çeneye yayılabilir. Anjina kadınlarda genellikle omuz ve sırt bölgelerinde, erkeklerde ise boğaz, boyun ve çene bölgelerinde daha çok hissedilir.

- Koroner arter hastalığı; Koroner arterler kalbi besleyen damarlara verilen isimdir. Hastalık bu damarlarda meydana gelen daralma veya tıkanma sonucunda oluşmaktadır. Kalp hastalığının en yaygın formlarından biri olmakla birlikte, kalp krizi ve anjinanın önde gelen nedenidir.
- Koroner kalp rahatsızlığı; atardamarlardaki hastalıkları ve neden olduğu komplikasyonları ifade eder.
- Kalp krizi; kan akışının yetersiz kalması nedeniyle, kalp kasının bir kısmının işlevini yitirmesi sonucu meydana gelmektedir. Kalp krizi sol kol, omuz veya çeneye de yayılabilen ciddi bir göğüs ağrısına neden olmaktadır. Şiddetli nefes darlığı, terleme ve solukluk yaygın ek semptomlarıdır.
- Ani ölüm; kalbin kan pompalama yeteneğinde ani bir kayıp olduğunda ortaya çıkar. Bu durum kalp krizi veya kalp ritmindeki ciddi anormallik nedeniyle olabilir.

• **Hipertansif kalp hastalığı:** Yüksek tansiyon nedeniyle kalp ve kan damarlarının aşırı yüklenmesi sonucunda oluşmaktadır.

- Anevrizma; kan damarlarının incelmesi nedeniyle balonlaşıp genişlemesidir. Damar duvarında zayıflama meydana geldiğinden yırtılma riski yüksektir.
- Periferik arter hastalığı; bacaklardaki kan damarlarının daralması ve / veya tıkanmasıdır. Yürürken bacaklarda ağrıya sebep olur. Periferik arter hastalığı olan birinin kangren olma riski oldukça yüksektir.

• **Romatizma kalp hastalığı:** Streptokok adı verilen bakterilerin neden olduğu romatizmal ateşin, kalbin kapaklarında hasar meydana getirmesidir.

- Kalp kapak hastalığı; kalbin kapak yapısının bozulup, işlevini yerine getirememesidir. Kapakçıklarda hastalık iki şekilde gelişebilir, kapağın daralması ve/veya yetmezliği (geriye doğru kan sızdırması).

- **İnflamatuar kalp hastalığı:** Kalp kasının (miyokardit), kalbi çevreleyen zarın (perikardit) ve kalbin iç astarının (endokardit) iltihaplanmasıdır.
  - Kardiyomiyopati; kalp kası hastalıkları anlamına gelir. Bazı kardiyomiyopati türleri genetiktir, diğerleri ise enfeksiyon veya diğer nedenlerden dolayı ortaya çıkar. En sık görülen kardiyomiyopati tiplerinden biri, kalbin genişlediği idiyopatik dilate kardiyomiyopatidir.
  - Perikardit; kalp zarı iltihaplanmasıdır. Perikardit genellikle aniden gelişir ve birkaç ay boyunca devam edebilir. Bazen perikardiyal tabakalar arasında aşırı miktarda sıvı oluşarak, perikardiyal efüzyona yol açabilmektedir.

2.2 Kalp Hastalıklarının Belirtileri

Kalp hastalığı belirtileri, hangi kalp hastalığının olduğuna bağlı olarak değişkenlik göstermektedir.

Genel olarak belirtiler aşağıdaki gibi listelenebilir.

- Kalp çarpıntısı
- Göğüs ağrısı, göğüste sıkışma, göğüs basıncı ve göğüs rahatsızlığı
- Nefes darlığı
- Baş dönmesi
- Bayılma (senkop)
- Düzensiz kalp atışı (taşikardi, bradikardi)
- Bacaklarda veya karında şişme
- Yorgunluk
- Uyuşukluk

2.3 Kalp Hastalıkları Risk Faktörleri

Kalp hastalığı oluşumunda etkili olan risk faktörleri aşağıdaki gibi sıralanabilir.

- **Yaş:** Yaşlanma ile birlikte oluşabilecek kalp kası incelmesi ya da kalınlaşması ve arterlerde daralma kalp hastalığı riskini arttırmaktadır.
- **Cinsiyet:** Kalp hastalığı kadınlara oranla erkeklerde daha fazla risk oluşturmaktadır. Kadınlarda ise risk oranı menopozdan sonra artmaktadır.
- **Aile Öyküsü:** Aile öyküsünde ebeveynlerden herhangi birinin kalp hastalığı geçirmiş olması, birey için risk faktörünü arttırmaktadır.
- **Sigara İçmek:** Nikotin damarlarda daralma oluşmasına, karbon monoksit ise damarların zarar görmesine neden olmaktadır. Bu sebepledir ki sigara içen insanların kalp hastalığına yakalanma riski içmeyenlere göre daha fazladır.
- **Kötü Beslenme:** Yağ, tuz, şeker ve kolesterolü yüksek gıda tüketimi kalp hastalığı oluşumunu arttırmaktadır.
- **Yüksek tansiyon:** Damarların sertleşmesine, kalınlaşmasına ve daralmasına neden olabilmektedir.
- **Kolesterol:** Kanda yüksek seviyelerde kolesterol plak ve ateroskleroz oluşma riskini arttırmaktadır.
- **Diyabet:** Diyabet kalp hastalığı riskini artırmaktadır.
- **Obezite:** Aşırı kilolu olmak, kalp hastalığına yakalanma riskini arttıran bir başka nedendir.
- **Fiziksel hareketsizlik:** Egzersiz eksikliği birçok kalp hastalığı ile ilişkili olmakla birlikte bazı risk faktörlerini beraberinde getirmektedir.
- **Stres:** Stres arterlerde hasara sebep olabileceği gibi diğer risk faktörlerini de kötüleştirebilmektedir.

- **Kötü hijyen:** Düzenli olarak el yıkamamak veya bakteriyel enfeksiyon oluşumunu önlemeye çalışmamak özellikle kalp rahatsızlığı olanlar için kalp krizi riskini arttırabilmektedir. Diş sağlığına dikkat etmemek de kalp hastalığına neden olabilmektedir.

2.4 Kalp Hastalıklarını Tahmin Etmek İçin Kullanılan Hesaplama Yöntemleri

Kalp hastalığının tanı ve tahmini için birçok yöntem kullanılmaktadır. Yapay sinir ağları da kullanılan yöntemlerden biridir. Kalp hastalığının tanı ve tahmini için yapılan çalışmalardan bazılarının yöntemlerine ve elde edilen sonuçlarına aşağıda yer verilmiştir.

Şahan ve diğer. (2005), yaptıkları çalışmada öznitelik ağırlıklı yapay bağışıklık sistemi yöntemini kullanarak UCI (University of California, Irvine)'den aldıkları Statlog kalp veri setinin 2'li sınıflandırma doğruluğunu 10 kat çapraz doğrulama yöntemi ile $\%82,59$ olarak elde etmişlerdir.

Bhatia ve diğer. (2008), yaptıkları çalışmada UCI'den almış olduğu 303 örnekli ve 5 sınıflı olan Cleveland veri setini kullanmışlardır. Veri setinde bulunan her sınıfta 13 öznitelik bulunmaktadır. Kalp hastalığı için genetik algoritma tabanlı sayısal olarak kodlanmış bir sınıflandırıcı geliştirmişlerdir. Bu sınıflandırıcı ile sadece gerekli ve önemli öznitelikleri baz alarak diğer veriler üzerinde eliminasyon yapmışlardır. Bu sayede hastalığın tahmini için gerekli öznitelik sayısını 13'ten 6 ya indirgemişlerdir. Veri setindeki 250 kaydı eğitim, geri kalan kısmı test için kullanmışlardır. 5'li sınıflandırıcıda elde ettikleri doğruluk oranı sadece 6 öznitelik kullandıklarında $\%72,55$ olarak hesaplanırken, 13 özniteliğin hepsini kullandıklarında $\%61,93$ olmuştur. Ayrıca 2'li sınıflamada (hasta ve hasta değil) önerdikleri teknik $\%90,57$ doğruluk oranına ulaşmıştır.

Özşen ve Güneş (2008), UCI'den aldıkları Statlog kalp veri setinin sınıflandırılması için basit bir yapay bağışıklık sistemi üzerinde öklid mesafesi, manhattan mesafesi ve

hibrid benzerlik ölçümü olan üç mesafe ölçütünü uygulamışlar ve 2'li sınıflandırmada $\%83,95$ doğruluk oranı elde etmişlerdir.

Özşen ve Güneş (2009), yaptıkları çalışmada genetik algoritmayı öznitelik ağırlıklı yapay bağışıklık sistemi ile birlikte kullanarak sınıflandırıcının hatasını minimize etmeyi amaçlamışlardır. Kullandıkları yöntem ile UCI'den aldıkları Statlog kalp veri seti üzerinde yaptıkları denemelerde 2'li sınıflandırmada $\%87,43$'lük doğruluk oranına ulaşmışlardır.

Das ve diğer. (2009), yaptıkları çalışmada UCI'den aldıkları Cleveland veri seti üzerinde SAS yazılımını kullanmışlardır. SAS yazılımı iki ana bileşenden oluşmaktadır. İlk bileşen veri işleme yaparken diğer bileşen analiz ve tanımlama üzerine yoğunlaşmaktadır. Üç farklı sinir ağı modelini paralel bir şekilde birleştirerek topluluk modelini oluşturmuşlardır. Modellerinde geri yayılım algoritmasını kullanmışlardır. Öğrenme algoritması olarak Levenberg – Marquardt (LM), ölçekli eşlenik gradyan ve Pola-Ribiere eşlenik gradyan algoritmalarını herbiri bir modele gelecek şekilde uygulayıp yaptıkları denemeler sonucunda 2'li sınıflandırmada $\%89,01$ sınıflandırma doğruluğuna ulaşmışlardır.

Srinivas ve diğer. (2010), yaptıkları çalışmada UCI 'den aldıkları veri seti ile bir bağımlılık arttırılmış Naive Bayes sınıflandırıcı ve Bayesian ağını kullanarak kalp krizlerini tahmin etmeye yarayacak bir sistem oluşturmuşlardır. 2'li sınıflandırmada en iyi doğruluk oranını Bayesian Ağı'nı kullanarak Weka'da $\%84$ olarak elde etmişlerdir.

Gudadhe ve diğer. (2010), UCI'den aldıkları Cleveland veri setindeki 200 kaydı eğitim geri kalanlarını ise test için kullanarak 2'li sınıflandırmada destek vektör makinesi algoritması ile doğruluk oranını $\%80,41$ olarak hesaplamışlardır. Diğer yandan veri setindeki 220 örneği eğitim, geri kalanını test için kullanarak 5'li sınıflandırmada yapay sinir ağını geri yayılım algoritması ile eğitmişler ve doğruluk oranını $\%97,5$ olarak elde etmişlerdir.

Khemphila ve Boonjing (2011), çalışmalarında çok katmanlı algılayıcı da geri yayılım algoritmasını kullanarak bir sınıflandırma sistemi önermişlerdir. UCI veri ambarından aldıkları 303 kayıtlı Cleveland veri setinin %60'ını eğitim %40'ını ise test için kullanmışlardır. 13 özniteliğe sahip sistemi öznitelik seçme algoritmasıyla 8 özniteliğe düşürmüşlerdir. 2'li sınıflandırmada 8 öznitelik kullanarak doğruluk oranını %80,99, 13 öznitelik kullanarak ise doğruluk oranını %80,13 olarak hesaplamışlardır.

Abdullah ve Rajalaxmi (2012), UCI 'den aldıkları Cleveland veri seti üzerinde rastgele orman (Random Forest) sınıflandırma algoritmasını kullanarak en iyi doğruluk oranını %63,33 olarak elde etmişlerdir.

Al-Milli (2013), yaptığı çalışmada UCI'den aldığı 303 kayıtlı veri setinin 116 kaydını eğitim seti, 50 kaydını test seti olmak üzere toplamda 166 kaydını kullanmışlardır. MATLAB'ta yapay sinir ağları simülasyonunu kullanarak 4'lü sınıflandırmalı sinir ağı eğitmiş ve test setinde %92 doğruluk oranına ulaşmıştır.

Deekshatulu ve diğer. (2013), kalp hastalıklarının sınıflandırılması için K en yakın komşuluk (KNN) algoritması ve genetik algoritma üzerinde çalışmışlar. Sınıflandırma işlemi için KNN makine öğrenme tekniğini genetik algoritma ile birleştiren yeni bir algoritma önermişlerdir. Önerdikleri algoritma ile çoklu kalp veri setlerinde ortalama %95,73 doğruluk oranı elde etmişlerdir.

Manju ve diğer. (2013), Yapay Sinir Ağı ve Genetik Algoritma kullanarak %89 doğruluk oranı elde etmişlerdir.

Patel ve diğer. (2013), Naive Bayes, karar ağacı algoritması ve kümeleme yoluyla sınıflandırma yöntemlerini kullanarak Weka'da kalp hastalığının tahmini üzerine çalışmalar yapmışlardır. Ayrıca, hangi özelliğin kalp rahatsızlıklarının teşhisine daha fazla katkıda bulunduğunu belirlemek için genetik algoritmayı kullanmışlardır. Yaptıkları çalışmalar neticesinde 6 özelliğin hastalığın belirlenmesinde daha etkili olduğu sonucuna ulaşmışlardır. Kullandıkları yöntemlerde 6 özelliğe indirgenmiş veri setini kullanmışlar ve en yüksek doğruluk oranını karar ağacı algoritması ile %99,2

olarak elde etmişlerdir.

Abushariah ve diğer. (2014), UCI'den aldıkları Cleveland veri seti üzerinde MATLAB'ta iki sistem önermişlerdir. İlk sistem YSA üzerindeki Çok Katmanlı Perceptron (ÇKP) yapısına dayanırken ikinci sistem Adaptif Nöro-Bulanık Çıkarım Sistemleri (ANFIS) yaklaşımına dayanmaktadır. Her iki sistemde Cleveland veri seti eğitim ve test için sırasıyla %80'e %20 oranla rastgele seçilerek ayrıştırılmıştır. Kalp hastalığı hastalığın olması ve olmaması durumuna göre 2'li sınıflandırılmıştır. Doğruluk oranı YSA ve ANFIS için sırasıyla $\%87,04$ ve $\%75,93$ olarak elde edilmiştir.

Juneja ve Dhingra (2014), kalp hastalığını tahmin etmek için birçok parametre ile bulanık mantık tabanlı bir hesaplama yaklaşımı uygulamışlardır. Hastalık tahmini için parametre olarak kan basıncı, kalp atışı, yaş, kolesterol, şeker ve sigara kullanımı özniteliklerini kullanmışlardır. Net değerleri bulanık değerlere dönüştürmüşler, girdi olarak bu değerleri ve o özelliğin kritiklik değerlerini de ele alarak üyelik fonksiyonları ile hastalık tahmini yapmışlardır.

Hasan ve diğer. (2017), UCI veri ambarından almış oldukları 303 kayıtlı Cleveland veri seti üzerinde MATLAB'ta kullandıkları çok katmanlı algılayıcı ve destek vektör makinesi algoritmaları ile 2'li sınıflamada sırasıyla %98 ve %96 doğruluk oranına ulaşmışlardır. Sınıfları şu şekilde belirlemişlerdir: Sıfır değeri hastalığın olmadığı, bir değeri kalp hastalığı olan, iki değeri anjina pektoris hastalığı olan, üç değeri kalp yetmezliği bulunan, dört değeri ise aritmi hastalığı olan hastalar için tanımlanmıştır. Çoklu sınıfa sahip sınıflandırıcıda ise çok katmanlı algılayıcı kullanarak %81 doğruluk oranına ulaşırken, destek vektör makineleri algoritması ile %60 doğruluk oranı elde etmişlerdir.

Liu ve diğer. (2017), yaptıkları çalışmada ReliefF ve Rough Set (RFRS) yöntemine dayalı bir hibrit sınıflandırma sistemi önermişlerdir. Önerilen sistem RFRS özellik seçim sistemi ve topluluk sınıflandırma sistemi olarak iki alt sistem içermektedir. İlk sistem; veri ayrıklaştırma, ReliefF algoritması kullanılarak özellik

çıkarma ve sezgisel Rough Set indirgeme algoritması ile özellik azaltma olmak üzere üç aşamadan oluşmaktadır. İkinci sistemde ise C4.5 sınıflandırıcısına dayalı bir topluluk sınıflandırıcısı önerilmiştir. Çalışmada UCI veri tabanından elde ettikleri Statlog veri setini kullanmışlardır. Veri setinde 270 örnek bulunmaktadır. Her bir örnek için 13 özellik mevcut olup hastalığın olma ve olmama durumu göz önüne alınarak 2 sınıf belirlenmiştir. Veri setini %70'e %30 oranla eğitim ve test seti olarak kullanıp maksimum $\%92,59$ doğruluk oranı elde etmişlerdir.

BÖLÜM ÜÇ
YAPAY SİNİR AĞLARI

Yapay Sinir Ağları (YSA) üzerine yapılan çalışmalar beynin çalışma mantığı ile geleneksel bilgisayarın çalışma mantığının tamamen farklı olduğunun anlaşılmasıyla başlamıştır. Beyin, karmaşıklığı oldukça yüksek, doğrusal olmayan ve paralel yapıda çalışan bilgi işleme sistemidir. Nöron olarak bilinen yapısal bileşenlerini düzenleyerek belirli hesaplamaları günümüz bilgisayarlarından daha hızlı yapabilme yeteneğine sahiptir. YSA beynin bu çalışma prensibinin modellenmesi için tasarlanmış bir makinadır Haykin (2009).

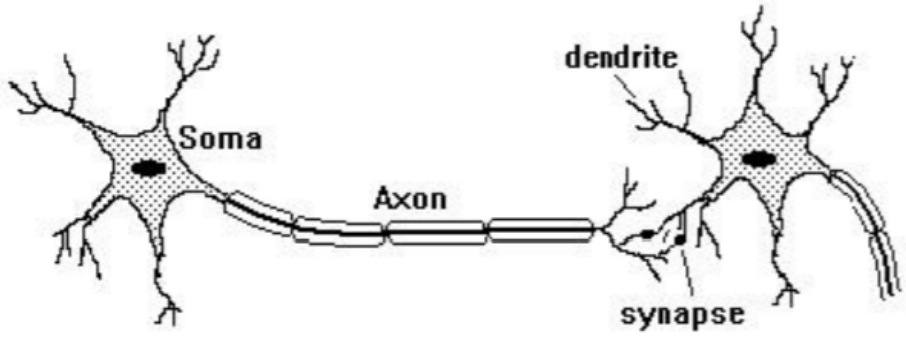
İnsanoğlu yaşamı boyunca edindiği tecrübeler neticesinde öğrenme kabiliyeti kazanmaktadır. Yaşanılan deneyimler ile sinir sistemindeki sinaptik bağlantılar sürekli gelişim ve değişim geçirir. İnsanlar ne kadar bilgi edinirlerse edinsin sinir hücreleri arasındaki bağlantıları gelişmemişse düşünme, muhakeme etme ve akıl yürütme becerileri gelişmemektedir, bu yüzden de eğitilmiş sayılmamaktadırlar. Tıpkı insanlardaki gibi YSA'larda nöronları arasındaki bağlantı ağırlıklarını edindiği deneyimler ile ayarlayarak öğrenme gerçekleştirir. Bu sayede ağ eğitilmiş olur. Eğitilen ağ yeni veya daha önce hiç karşılaşmadığı örneklere, önceki tecrübelerine dayanarak mantıklı cevap vererek genelleme yapabilmektedir.

YSA'lar sinirbilim, matematik, istatistik, fizik, bilgisayar bilimleri ve mühendisliği gibi birçok farklı disipline ait problemlerin çözümünde kullanılmaktadır. YSA öğrenme kabiliyetine sahip olması nedeniyle sinyal işleme, desen tanıma, tahmin, zaman serisi analizi ve kontrolü gibi farklı uygulamalarda yer bulmuştur.

3.1 Biyolojik Sinir Ağları

Biyolojik sinir ağları beynin çalışmasına olanak sağlayan en temel yapı taşlarıdır ve insanların çevresindekileri anlamlandırıp algılamasını sağlar. Milyonlarca sinir hücresi bir araya gelerek bir sinir ağını oluştururlar. Şekil 3.1'de görüleceği gibi bir

sinir hücresi hücre gövdesi (soma), akson, dentrit ve sinapslardan oluşmaktadır. Sinir hücreleri arasındaki elektrik sinyal geçişleri, sinapslar tarafından sağlanmaktadır. Elektirik sinyalleri, hücre gövdesi ve dentrit'lere iletilir. Dentritler aldıkları bu sinyalleri sinaps'lara göndererek diğer sinir hücrelerine aktarılmasını sağlarlar.



Şekil 3.1 Biyolojik sinir hücresi (Cilimkovic, 2019)

3.2 YSA'nın Tarihçesi

- 1943 yılında, matematikçi Walter Pitts ve nörofizyolog Warren McCulloch, nöronların nasıl çalışabileceği üzerine bir makale yayınlamışlardır. Beyindeki nöronların nasıl çalıştığını tanımlamak için insan beyninin çalışma tekniğinden esinlenip elektrik devreleri kullanarak bir sinir ağı oluşturmuşlardır.
- 1949'da Donald Hebb, "Organization of Behavior" adlı kitabında "Hebbian öğrenme" adlı temel teoriyi ele almıştır. Hebb kuralı sayesinde öğrenebilme ve uyum sağlayabilmenin sinir ağının bağlantı sayısı ile ilişkili olduğu saptanmıştır.
- 1951'de, ilk nöro bilgisayar üretilmiştir.
- 1954'te, IBM araştırmacıları olan Farley ve Clark YSA'lar için simülasyon çalışması yapmışlar ancak modellerini çalıştıramamışlardır.
- 1958'de Frank Rosenblatt, perceptron modelini geliştirmiştir. Bu model iki katmanlı, basit ekleme ve çıkarma işlemini kullanan bir desen tanıma algoritmasıdır.

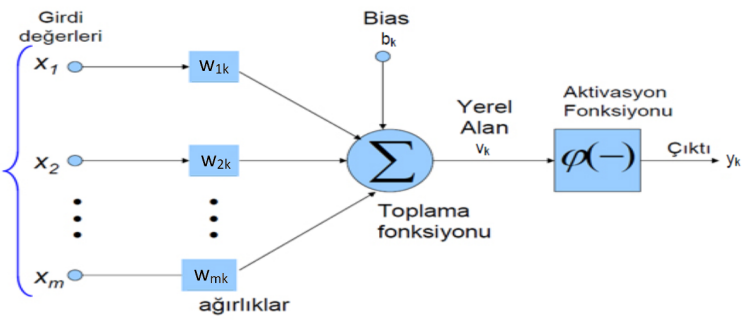
- 1959 yılında Stanford'ta Bernard Widrow ve lisansüstü öğrencisi olan Marcian Hoff "ADALINE" ve "MADALINE" adlı modelleri geliştirmişlerdir. Ayrıca, en küçük kareler algoritması olarak bilinen bir öğrenme algoritması da geliştirmişlerdir.
- 1965 yılında Nilssons, ilk makine öğrenme kitabı olan "Öğrenen Makineler" isimli kitabı yazmıştır. Bu kitapta yapay sinir ağı üzerine yapılan tüm çalışmalar bir araya getirilmiştir.
- 1969 'da Minsky ve Papert yaptıkları incelemeler sonucunda Perceptron'un çok sınırlı alanda kullanılabileceğini, bir çok problemin çözümünde yetersiz kaldığını yayınladıkları "Perceptrons" kitabıyla duyurmuşlardır. XOR probleminin çözümünü yapamadığını ispatlamışlardır. XOR probleminin çözümü için iki katmanlı ağların kullanılabileceği düşünmüşler ancak ağırlıklarının değiştirilmesi konusunda izleyebilecekleri bir yöntem önerememişlerdir.
- 1970 yılında Fukushima, görüntü işleme ve tanıma için Neocognitron modeli tanıtmıştır.
- 1972'de Kohonen ve Anderson, birbirinden bağımsız bir şekilde "çağrışımlı bellek" konusu üzerinde benzer bir ağ geliştirmişlerdir.
- 1982 yılına gelindiğinde Kohonen, "Kendi kendine Öğrenme Nitelik Haritaları (Self Organizing Feature Maps SOM)" isimli çalışma yayınlamıştır.
- 1982-1984 yılları arasında Hopfield, ağların matematiksel modellerini ortaya koymuştur. Yayınladığı çalışmalar ile YSA'ların genelleştirilebilme yeteneğinin varlığını ve çözülmesi oldukça zor problemler için çözüm üretebildiklerini kanıtlamıştır. Bunun en büyük kanıtı ise gezgin satıcı probleminin çözülmesidir.
- 1984 yılında Kohonen, danışmansız öğrenme stratejisini geliştirmiştir.
- 1986 yılında Rumelhart ve McClelland, ÇKA(Çok Katmanlı Algılayıcı)'lar için geriye yayılım öğrenme algoritmasını geliştirmişlerdir. Bu algoritma

günümüzde oldukça yaygın bir şekilde kullanılmaktadır.

- 1987'de, Amerikan Elektrik Elektronik Mühendisligi Enstitüsü tarafından uluslar arası ilk YSA konferansı 1800 den fazla katılımcı ile gerçekleştirilmiştir.

3.3 Yapay Sinir Hücresinin Yapısı

YSA insan beyninin kaba bir modeline dayanmaktadır. Nöron, bir YSA'da temel olan bilgi işleme birimidir. Nöronlar birbirlerine paralel yapıda ayarlamalı ağırlıklarla bağlanmaktadır.



Şekil 3.2 Doğrusal olmayan nöron modeli (Adıyaman, 2007)

Şekil 3.2'teki blok diyagram bir nöron modelini göstermektedir. Nöron modelinin beş temel elemanı vardır:

1. **Girdiler:** Dış dünyadan yapay sinir hücresine iletilen bilgilere denir. Bilgiler dış sistemden gelen veriler olabileceği gibi başka yapay sinir ağı hücrelerinin çıktı değerleri de olabilir. Bazen işlem elemanları geri besleme yöntemi ile kendileri de girdi oluşturabilir.

2. **Ağırlıklar:** Bir sinir hücresine genellikle bir çok girdi verisi ulaşır. Her girdinin kendine bağlı bir ağırlık değeri vardır. Her bir ağırlığın farklı sabit değerleri vardır. Ağırlıkların başlangıç değerleri genellikle rastgele seçilmekte ve pozitif veya negatif değer alabilmektedirler. Hücreye gelen bilginin etki ve

önem derecesi ağırlıklar ile belirlenir. Bu sayede bazı girdi değerleri diğerlerinden daha önemli olabilir ve böylece sinir hücresi üzerinde daha fazla etkili olabilir. Ağın topolojik yapısına veya öğrenme algoritmasına göre ağırlıkların değiştirilmesi yöntemleri farklılık göstermektedir.

3. **Toplama Fonksiyonu:** Bir hücreye gelen net girdiyi hesaplayan fonksiyondur. En yaygın kullanılan toplama fonksiyonu girdilerin ilgili ağırlıkları ile çarpılıp toplandığı ağırlıklı toplam fonksiyonudur ve 3.1 'deki gibi formülüze edilir.

$$net_k = \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

Hangi problemlerde hangi toplama fonksiyonunun kullanılacağına dair bir formül yoktur. Tüm işlem elemanları aynı toplama fonksiyonunu kullanabilecekleri gibi her bir işlem elemanı bağımsız olarak farklı bir toplama fonksiyonunu da kullanabilmektedir. Yaygın olarak kullanılan toplama fonksiyonlarından bazıları aşağıda verilmiştir:

- Minimum : $net = \min w_i x_i$
- Maksimum : $net = \max w_i x_i$
- Çoğunluk : $net = \sum_i sgn(w_i x_i)$
- Çarpım : $net = \prod w_i x_i$
- Kümülatif Toplam : $net = net_{eski} + \sum_i w_i x_i$

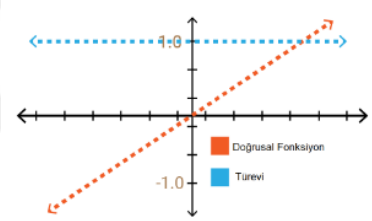
4. **Aktivasyon Fonksiyonu:** Her bir işlem elemanına gelen net girdinin işlenip girdiye karşılık gelen çıktı değerinin üretilmesini sağlayan fonksiyondur. Aktivasyon fonksiyonları ağın kullanıldığı probleme, ağın topolojisine veya kullanılan öğrenme algoritmasına göre değişiklik gösterebilir. Örneğin geri yayılım (backpropagation) öğrenme algoritması uygulanacak ağlar için, aktivasyon fonksiyonu türevlenebilir bir fonksiyon olmalıdır.

Nöron modelinin doğrusal olmama özelliği, doğrusal olmayan aktivasyon fonksiyonunun kullanılması ile ilişkilidir. Bu sayede girdi ve çıktı değerleri arasında daha kuvvetli nonlineer (doğrusal olmayan) ilişki kurulmuş olur.

Doğrusal olmayan fonksiyonlar hem pozitif hem de negatif değerler üretmektedirler ve sadece pozitif veya negatif değer üreten fonksiyonlara kıyasla ağın daha hızlı öğrenmesini sağlamaktadırlar. Yapay nöronlarda doğrusal olmayan aktivasyon fonksiyonu kullanılmazsa temsil edilen sistem doğrusal özellik gösterecektir. Bu tip yapay nöronlar doğrusal olmayan hesaplamaları yerine getiremeyeceğinden doğrusal olmayan sistemlerde kullanılamazlar. Doğrusal aktivasyon fonksiyonu kullanılan sinir modeli, Widrow-Hoff modeli olarak ta bilinmektedir.

Bazı aktivasyon fonksiyonları şunlardır.

- Doğrusal Fonksiyon : Doğrusal fonksiyon, doğrusal problemlerin çözümünde aktivasyon fonksiyonu olarak seçilebilir. Denklem 3.2'de gösterildiği gibi doğrusal fonksiyonun çıktısı girdisine eşittir. Şekil Şekil 3.3 doğrusal fonksiyonu ve türevini göstermektedir.

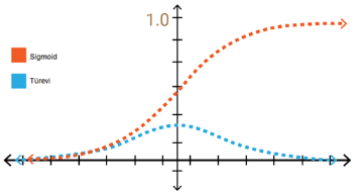


Şekil 3.3 Doğrusal fonksiyon (Kızrak, 2019)

Doğrusal fonksiyonun denklemi :

$$f(net) = net \tag{3.2}$$

- Sigmoid Fonksiyonu : Türevlenebilir, sürekli ve doğrusal olmayan bir fonksiyondur. Bu sebeplerden dolayı, en çok tercih edilen aktivasyon fonksiyonudur. Bu fonksiyon her girdi değerine karsılık [0,1] aralığında bir değer vermektedir. Şekil 3.4'de fonksiyon ve türevi gösterilmiştir.

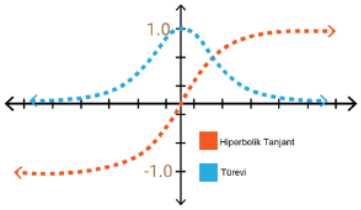


Şekil 3.4 Sigmoid fonsiyonu (Kızrak, 2019)

Sigmoid fonksiyonunun denklemi :

$$f(net) = \frac{1}{1 + e^{-net}} \tag{3.3}$$

- Hiperbolik Tanjant Fonksiyonu : Sigmoid fonksiyonunun farklı bir biçimidir. Bu fonksiyonda üretilen çıktı -1 ile +1 değerleri arasında bir değere karşılık gelir. Fonksiyonun türevi ve kendisi Şekil 3.5'te gösterilmiştir.



Şekil 3.5 Hiperbolik tanjant fonksiyonu (Kızrak, 2019)

Hiperbolik Tanjant Fonksiyonunun denklemi;

$$f(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}} \tag{3.4}$$

5. **Hücre çıktısı :** Aktivasyon fonksiyonun ürettiği sonuç hücre çıktısı olarak kabul edilir. Bu çıktı hücrenin ağdaki konumuna göre bağlı olduğu diğer hücrelere girdi olarak iletilir veya ağın ürettiği sonuç olarak değerlendirilir.

Bir yapay sinir hücresinin çalışmasını özetleyecek olursak;

Şekil 3.2'de gösterilen model dışardan ayrıca bir girdiye sahiptir. Aktivasyon fonksiyonunun net girdi değeri b_k ile gösterilen sapma (bias) değeri pozitif olduğunda

yükselmekte, negatif olduğunda ise düşmektedir.

Aşağıdaki iki eşitlik k nöronunun matematiksel ifadesidir.

$$u_k = \sum_{j=1}^{m} w_{jk} x_j \tag{3.5}$$

$$y_k = \phi(v_k) \tag{3.6}$$

Burada $x_1, x_2, ..., x_m$ girdi işaretlerini, $w_{1k}, w_{2k}, ..., w_{mk}$, k nöronunun sinaptik ağırlıklarını, u_k girdi işaretlerinden gelen ağırlıklı toplam değerini, b_k bias (sapma) terimini, $\phi(.)$ aktivasyon fonksiyonunu ve y_k nöronun çıktısını belirtir. 3.6 'te görülen v_k, aktivasyon fonksiyonunun net girdisidir. Bias kullanılması halinde $v_k = u_k + b_k$, kullanılmadığında ise $v_k = u_k$ olarak hesaplanır.

3.4 YSA'nın Özellikleri

YSA'nın paralel dağıtılmış yapısı ve genelleme yeteneği sayesinde hesaplama ve bilgi işleme yeteneği oldukça gelişmiştir. Genelleme, ağın öğrenme esnasında hiç karşılaşmadığı örneklere uygun çıktılar üretebilmek olarak tanımlanabilir. YSA bu üstün özellikleri ile karmaşık problemlerin çözebilmesini mümkün kılmaktadır.

YSA'nın başlıca özellikleri şöyle sıralanabilir.

1. **Doğrusal Olmama:** YSA'ların en küçük işlem elemanı olan hücre doğrusal bir yapıya sahip değildir. Bu nedenle bu hücrelerin birleşiminden oluşan YSA'lar da doğrusal değildir ve bu özellik tüm ağa yayılmıştır. YSA'nın bu özelliğinin karmaşık ve doğrusal olmayan problemlere çözüm üretmesindeki önemi büyüktür.

2. **Öğrenme:** Öğrenme ile kastedilen ilgili problemin girdi-çıktı ilişkisini en iyi verebilecek optimum ağırlık değerlerine ulaşılmasıdır. Karmaşık yapısı nedeniyle bağlantı ve ağırlık değerleri önceden ayarlanmış bir şekilde YSA'lara

verilememektedir. Bu nedenle YSA, üzerinde çalıştığı problemden aldığı eğitim verilerini kullanarak istenilen davranışı gösterecek şekilde problemi öğrenmelidir.

3. **Genelleme:** Güçlü paralel dağınık yapı ve öğrenme yeteneği sayesinde sinir ağı, öğrenme sürecine katılamayan girdiler için makul çıktılar üretebilir. Kusurlu veya eksik verileri işleyebilir ve hataya dayanıklı olma potansiyeline sahiptir. Pratik gürültülü verileri analiz ederken bu çok yararlıdır.

4. **Adaptasyon:** YSA üzerinde çalışılan sorunun değişimlerine göre ağırlık değerlerini günceller. Belirli bir çevrede çalışmak üzere eğitilmiş olan YSA çalışma ortamı koşullarında olabilecek küçük değişikliklere karşı tekrar eğitim alabilir. Eğer değişiklikler sürekli ise eğitim gerçek zamanlı olarak devam edebilir. Uyarlama özelliği nedeniyle YSA sinyal işleme, örnek tanıma ve denetim gibi alanlarda etkin bir şekilde kullanılır.

5. **Hata toleransı:** YSA'lar çeşitli şekillerde bağlanmış çok sayıda hücreden oluştuğundan, paralel dağılmış yapıya sahiptir ve ağda bulunan bilgi tek bir yerde toplanmak yerine ağdaki tüm bağlantılara dağılmış durumdadır. Dolayısıyla eğitimi tamamlanmış bir YSA'nın bağlantılarının bazılarında hatta bazı hücrelerinde hasar oluşması, ağın doğru çıktıyı üretmesini büyük ölçüde etkilemeyecektir. Bilginin tek bir yerde saklanmaktan ziyade tüm sisteme dağılmış olması YSA'nın hatayı tolere etme becerisinin, geleneksel yöntemler ile karşılaştırıldığında daha yüksek olduğunu göstermektedir.

6. **Paralel İşlem Yapma:** YSA paralel yapısı sayesinde büyük ölçekli entegre devre (VLSI) teknolojisi ile gerçeklenebilir. Bu özellik YSA'nın bilgi işleme yeteneğini hızlandırır ve işaret işleme, örnek tanıma, denetim, sistem kimliklendirme gibi gerçek zamanlı uygulamalar tarafından tercih edilir.

7. **Analiz ve Tasarım Kolaylığı:** YSA modeli ve işleme elemanının yapısı tüm problemler için neredeyse aynıdır. YSA'nın bu standart yapısı farklı uygulama alanlarında kullanılacaktır. Bu nedenle farklı uygulama alanlarında kullanılan YSA'lar benzer öğrenme algoritmalarını ve teorilerini paylaşabilirler. Bütün

YSA uygulamalarında hep aynı gösterimin kullanılması problemlerin YSA ile çözümünde önemli bir kolaylık getirmektedir.

3.5 YSA'nın Uygulama Alanları

YSA'nın insan zekasını taklit etmesi fikrinden yola çıkılarak her alana uygulanabileceği söylenebilir. YSA'nın son derece disiplinler arası bir alan olduğu günümüzde uygulama alanının genişliğine bakıldığında görülebilmektedir. YSA'nın temel kullanım alanları aşağıdaki şekilde belirtilebilir.

1. **Arıza analizi ve tespiti:** Bir sistemde veya cihazda oluşabilecek arızalar öğrenen bir yapay sinir ağı tarafından tahmin edilebilmektedir. YSA bu amaçla uçaklarda, elektrik makinelerinde ve entegre devrelerde kullanılmıştır.

2. **Finans:** Banka kredi değerlendirilmesi, ekonomik tahminler, risk analizleri, döviz kuru tahmini, bütçe kestirimi, performans analizi, hedef belirleme gibi finansal konularda uygulanmaktadır.

3. **Tıp alanında:** Kanserli hücre tespiti, tıbbi sinyal analizi, ilaç etkileri analizi, hastalık teşhisi ve tedavisi gibi alanlarda kullanılmaktadır.

4. **Savunma sanayi:** Hedef izleme ve seçme, silah otomasyonu, sensör, sonar ve radar sistemleri, görüntü ve sinyal işleme, askeri uçakların uçuş yörüngelerinin belirlenmesi, gürültüyü önleme gibi alanlarda etkin olarak kullanılmaktadır.

5. **Haberleşme:** Data ve görüntü karşılaştırma, gürültü sönümlendirilmesi, filtreleme, görüntü ve ses işleme, anahtarlama ve trafik yoğunluğu kontrolü, otomatik bilgi sunma servislerinde uygulama alanı bulmuştur.

6. **Üretim:** Üretim tasarımı ve analizi, sistem optimizasyonu, ürün tasarımı ve analizi, kalite kontrol ve analizi, yönetim ve planlama analizi, makina yıpranmalarının tespiti, iş çizelgelerinin hazırlanması v.b. alanlarda YSA uygulanmaktadır.

7. **Güvenlik:** Kredi kartı hile saptama, retina ve parmak izi tanıma, yüz eşleştirme alanlarında uygulanmaktadır.

8. **Otomasyon ve kontrol:** Robot sistem kontrolü, oto pilot otomasyonu, ulaşım araçlarında otomatik yol gösterme ve bulma, doğrusal olmayan sistemlerin kontrol ve modellemesi gibi uygulama alanlarında yer almıştır.

   Günlük hayatımızda YSA'ların pek farkına varamadığımız bir çok alanda kullanıldığı görülmektedir. Uygulama alanları gittikçe gelişmekte ve genişlemektedir.

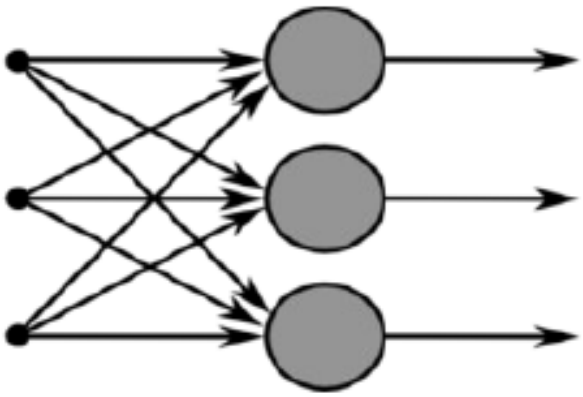
3.6 Ağ Yapıları

3.6.1 İleri Beslemeli Sinir Ağları(Feedforward Neural Networks)

İleri beslemeli YSA, akış yönünün sadece girişten çıkışa doğru tek yönlü olduğu ağlardır. Bu YSA modelinde bir katmana ancak önceki katmandan giriş olabilir. Yani bir nöronun çıktısı bir sonraki nöronun girdisi olarak kullanılabilir. YSA işlem elemanlarının oluşturduğu her bir gruba katman adı verilir. İşlem elemanları birbirlerine bağlanıp katmanları, bu katmanlar da ağ yapısını olusturur. İleri beslemeli YSA tek ve çok katmanlı olmak üzere ikiye ayrılır.

3.6.1.1 Tek Katmanlı İleri Beslemeli Ağlar

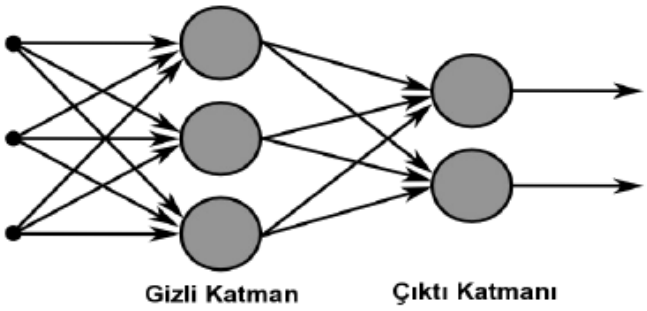
En basit ağ yapısı olarak bilinirler. Tek bir girdi katmanı vardır ve bu da doğrudan çıktı katmanına bağlıdır. Çıktı katmanında bulunan herbir nöron, girdi katmanında bulunan tüm nöronlarla bağlantılıdır. Girdi katmanında hiçbir hesaplama yapılmadığı için sadece çıktı katmanı hesaba katılarak bu türdeki ağlara tek katmanlı ağ denir. Şekil 3.6'da tek katmanlı ileri beslemeli ağ yapısı görülmektedir.



Şekil 3.6 Tek katmanlı ileri beslemeli sinir ağı (Wikibooks, 2019)

3.6.1.2 Çok Katmanlı İleri Beslemeli Ağlar

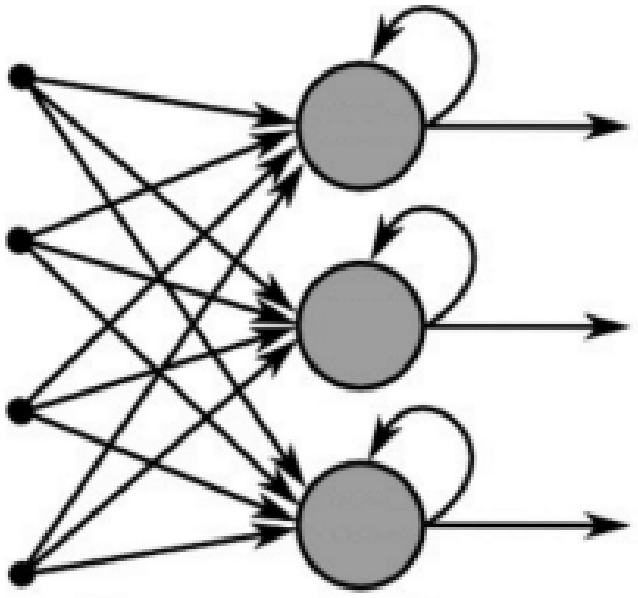
Bir önceki modelden farklı olarak bu tür ağlarda bir veya birden fazla gizli katman bulunur. Sinir ağı katmanlar halinde düzenlenir; ilk katman girdileri alır, son katman çıktıların üretildiği katmandır. Orta katmanların dış dünyayla hiçbir bağlantısı yoktur ve gizli katmanlar denir. Bir katmandaki her bir nöronun bir sonraki katmandaki her nörona bağlandığı ağlara *tümüyle bağlı ağ*, bazı nöronlar arası bağlantıların eksik olduğu ağlara ise *kısmen bağlı ağ* denilmektedir. Çok katmanlı ileri beslemeli ve tümüyle bağlı ağ Şekil 3.7 ’de gösterilmiştir.



Şekil 3.7 Çok katmanlı ileri beslemeli sinir ağı (Wikibooks, 2019)

3.6.2 Geri Dönüşümlü Sinir Ağları (Recurrent Neural Networks)

Bu tür ağları ileri beslemeli ağlardan ayıran özellik, en az bir tane geri besleme döngüsüne sahip olmasıdır. Örneğin bir geri dönüşümlü ağ tek bir katmandan oluşabilir ve bu katmandaki her sinir hücresinin çıktısı diğer hücrelere girdi olarak aktarılabilir. Bir sinir hücresi kendi çıktısını yine kendisine girdi olarak gönderebilir. Geri besleme döngülerinin olması ağın öğrenme yeteneklerini arttırır ve daha performanslı çalışmasını sağlar Haykin (1999). Şekil 3.8 'de örnek bir geri dönüşümlü ağ gösterilmiştir.



Şekil 3.8 Geri dönüşümlü sinir ağı (Jeans, 2019)

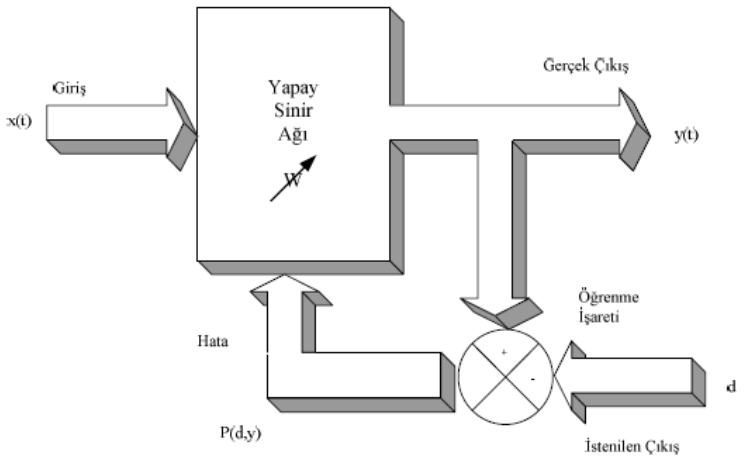
3.7 Öğrenme Stratejileri

Bir sinir ağı için birincil önemi olan özellik çevresinden öğrenmek ve öğrenme yoluyla performansını geliştirmektir. Genel olarak öğrenme deneyimle ortaya çıkan davranış değişiklikleridir. Sinir ağında öğrenme ağın problem için en doğru çıktıları üretmesini sağlayacak ağırlık değerlerinin bulunması olarak tanımlanabilir. YSA'da öğrenmeyi gerçekleştirecek sistem ve kullanılacak öğrenme algoritmasına göre genel

olarak üç öğrenme stratejisi kullanılmaktadır. Bunlar, danışmanlı, danışmansız ve takviyeli öğrenmedir.

3.7.1 Danışmanlı Öğrenme (Supervised Learning)

Danışmanlı öğrenmede YSA'ya gerçek çıktı değerleri verilir. Ağın çıktısı ve gerçek çıktı arasındaki fark hata olarak ele alınır. Elde edilen hata değerine göre nöronlar arası bağlantı ağırlıkları düzenlenerek en uygun çıktıyı elde etmek amaçlanmaktadır. Şekil 3.9 'da danışmanlı öğrenmenin blok diyagramı gösterilmiştir.



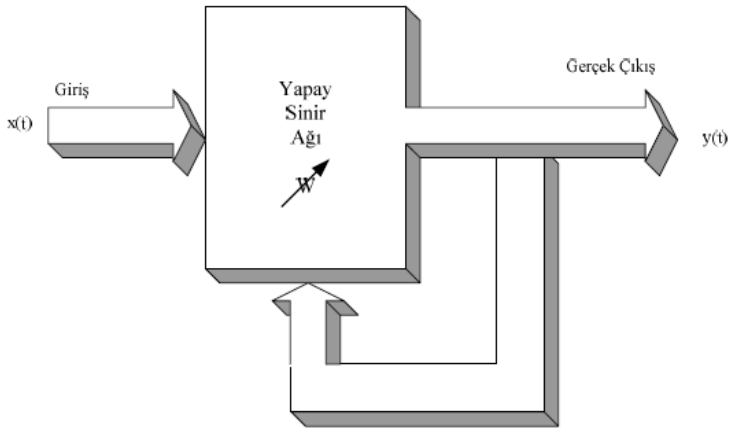
Şekil 3.9 Danışmanlı öğrenmenin blok diyagramı (Ayhan, 2016)

Widrow-Hoff'un geliştirmiş olduğu delta kuralı, McClelland ve Rumelhart tarafından geliştirilmiş olan genelleştirilmiş delta kuralı ve geriye yayılım (backpropagation) algoritması danışmanlı öğrenme için örnek olarak verilebilir.

3.7.2 Danışmansız Öğrenme (Unsupervised Learning)

Ağa sadece giriş bilgilerinin verildiği öğrenme stratejisidir. Ulaşılmak istenen hedef çıktılar ağa tanıtılmaz. Sistemin kendisine sunulan girdilerden elde edilen çıktı bilgisine göre ağın kuralları kendi kendisine öğrenmesi beklenir. Genellikle sınıflandırma problemlerinde kullanılan öğrenme stratejisidir. Sistemin öğrenme

işlemi tamamlandıktan sonra YSA çıktıları belli parametrelere göre sınıflandırmıştır ancak bu sınıfların ne anlama geldiğini bilmemektedir. Hangi kategorinin neyi gösterdiği kullanıcı tarafından belirlenmelidir. Bu işleme etiketlendirme ismi verilmektedir. Şekil 3.10'da danışmansız öğrenmenin blok diyagramı görülebilmektedir.

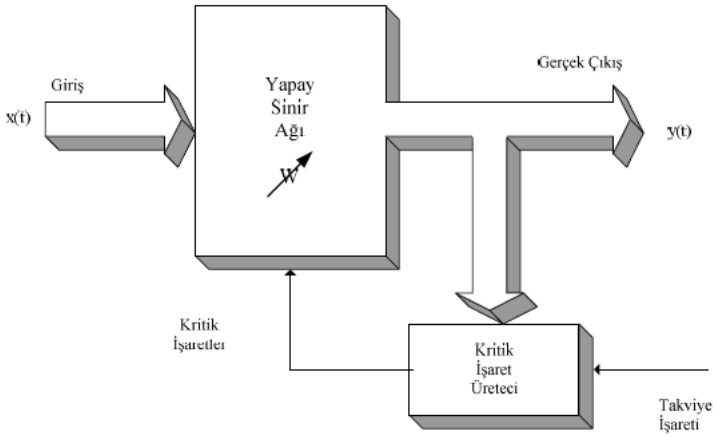


Şekil 3.10 Danışmansız öğrenmenin blok diyagramı (Ayhan, 2016)

Danışmansız öğrenmeye örnek olarak, Grosberg'in geliştirmiş olduğu ART ağları ile Kohonen'in geliştirdiği SOM öğrenme kuralı verilebilir.

3.7.3 Takviyeli Öğrenme (Reinforcement Learning)

Takviyeli öğrenme algoritmasında istenilen çıktının bilinmesine gerek yoktur. Hedeflenen çıktı ağa verilmez fakat elde edilen çıktının girdiye göre iyiliğini değerlendiren bir kriter kullanılmaktadır. Şekil 3.11 takviyeli öğrenme algoritma yapısının blok diyagramını göstermektedir. Hinton ve Sejnowski tarafından optimizasyon problemlerini çözmek için geliştirilen Boltzmann kuralı ile genetik algoritma takviyeli öğrenme stratejisine örnek olarak verilebilir.



Şekil 3.11 Takviyeli öğrenmenin blok diyagramı (Ayhan, 2016)

3.8 Öğrenme Kuralları

YSA gibi sistemlerde öğrenme işlemi bazı kurallara göre gerçekleşmektedir. Bu kuralların bir kısmı çevrimiçi (on-line) çalışırken, bir kısmı ise çevrimdışı (off-line) çalışmaktadır.

3.8.1 Çevrimiçi (On-line) öğrenme

On-line diğer bir deyişle gerçek zamanlı öğrenmede sistemde bir öğrenme algoritması bulunmakta olup, bu yapıyla uyumlu yazılım ve donanım mevcuttur. Bu ağ çalışma modunda öğrenme süreklidir. Bu tip öğrenme daha karmaşık yapıların tasarımı için kullanılır ve maliyetleri yüksektir. Art ağının öğrenme kuralı ile Kohenen öğrenme kuralı çevrimiçi öğrenme gerçekleştirmektedirler.

3.8.2 Çevrimdışı (Off-line) öğrenme

Çoğunlukla ağlarda off-line öğrenme kullanılmaktadır. Bu yaklaşımda ilk olarak ağ eğitilir. Eğitim tamamlandıktan sonra elde edilen ağırlıklar on-line uygulamalarda kullanılabilir. Bu aşamada hiç bir öğrenme algoritması kullanılmaz sadece mevcut ağ

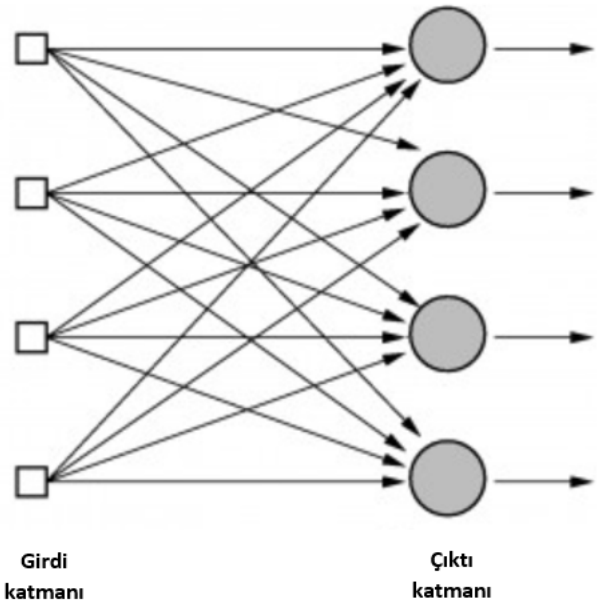
parametreleri kullanılarak ileri yönde bir hesaplama yapılarak çıktı hesaplanır. Örnek olarak aşağıdaki kurallar verilebilir.

- Hebb Öğrenme Kuralı
- Hopfield Öğrenme Kuralı
- Basit Algılayıcı (Perceptron) Öğrenme Kuralı
- Delta Ögrenme Kuralı (Windrow-Hoff Kuralı)

BÖLÜM DÖRT
YAPAY SİNİR AĞI MODELLERİ VE ÖĞRENME ALGORİTMALARI

4.1 Tek Katmanlı Algılayıcılar

Tek katmanlı algılayıcılarda bir girdi ve bir de çıktı katmanı bulunmaktadır. Katmanlarda bir veya daha fazla nöron yer alabilmektedir. $x = [x_1, x_2, ..., x_n]$ girdi katmanındaki n tane girdinin vektörünü, $y = [y_1, y_2, ..., y_k]$ çıktı katmanındaki k tane nöronun çıktı vektörünü göstermektedir. $w = [w_{1n}, w_{2n}, ..., w_{nk}]$ ise girdi katmanı ile çıktı katmanı arasındaki bağlantıların vektörünü göstermektedir. Burada w_{ij} ile ifade edilen girdi katmanındaki $i.$ node ile çıktı katmanındaki $j.$ node arasındaki bağlantının ağırlığıdır. Şekil 4.1'da TKA modeli görülebilir.



Şekil 4.1 Tek katmanlı algılayıcı modeli (Kurenkov, 2015)

Çıktı katmanındaki her bir nöronun çıktısı kendine gelen tüm girdi değerleri ve eşik değerin ağırlığının toplanarak bu toplam üzerine kendi aktivasyon fonksiyonunun uygulanması sonucu elde edilir. Çıkış vektörü Denklem 4.1'deki gibi ifade edilebilir.

$$Y_1 = F_1[w_{1x} + bB] \tag{4.1}$$

Burada F_1, k nörona sahip çıktı katmanının aktivasyon matrisidir ve bu katmanın net girdilerine bağlıdır.

$$F_1(S_1) = diag[f_1(S_1) + f_1(S_2) + ... + f_1(S_k)] \tag{4.2}$$

Burada k node'larının her birinin aktivasyon fonksiyonu eşit sayılmıştır.

$$f_{11} = f_{12} = ... = f_{1k} = f_1 \tag{4.3}$$

S_1 net vektörü $S_1 = [S_1, S_2, ..., S_k]^T$ çıktı katmanının toplam vektörüdür. Burada S_i çıktı katmanındaki $i.$ nöronda toplanan net ağırlığı belirtmektedir ve b_i çıktı katmanındaki $i.$ nöronun eşik değeri olmak üzere Denklem 4.4'deki gibi ifade edilmektedir.

$$S_i = \sum_{j=1}^{n} w_{ij}(t) + b_i \tag{4.4}$$

4.1.1 Basit Algılayıcı (Perceptron) Modeli

Frank Rosenblatt, "zeki sistemlerin temel özelliklerinden bazılarını simüle etmek" için Perceptronu geliştirmiştir. Perceptron modelinde çıktı katmanında bulunan bir nöron birden fazla girdi değerini alarak çıktı üretmektedir. Ağın ürettiği çıktı oluşturulurken, girdi değerleri kendi ağırlıkları ile çarpılıp toplanır ve bu toplamdan elde edilen sonuç eşik değer ile karşılaştırılarak ağın çıktısı belirlenir. Elde edilen toplam değer eşik değerden büyük veya eşit ise ağın çıktısı 1, küçük ise 0 seçilmektedir. Rosenblatt desen tanıma problemleri için de öğrenme kuralı geliştirmiştir Rosenblatt (1958). Çalışmasında kuralın çözümü verecek olan doğru ağırlık değerlerine yakınsayacağını kanıtlamıştır. Rosenblatt ayrıca, dijital bilgisayarlarda yapay sinir ağlarının simülasyonunu gerçekleştirmiştir Rosenblatt

(1961). Yaptıkları detaylı matematiksel incelemer sonucunda Marvin Minsky ve Seymour Papert algılayıcıların belli alanlarda kullanılabileceğini ve çözemeyeceği çok fazla problem sınıfının bulunduğunu yayınlamış oldukları ”Perceptrons” adlı kitapları ile kamuya göstermişlerdir Minsky ve Papert (1969). XOR problemi algılayıcılar tarafından çözülemeyecek örneklerden birisidir. Bu sorun 1980’lerde ÇKA’ın geliştirilmesi ile giderilmiştir.

4.1.2 Adaline Modeli

Frank Rosenblatt’ın Perceptronu geliştirdiği esnada YSA üzerine çalışmaya başlayan Bernard Widrow ve lisansüstü öğrencisi Marcian Hoff 1960 yılında ADALINE ağı ile En Küçük Kareler (EKK) algoritması olarak bilinen öğrenme kuralını geliştirdiler. Adaline ağının algılayıcılardan farkı aktivasyon fonksiyonu olarak eşik fonksiyonu yerine doğrusal fonksiyon kullanmasıdır. Her iki model de yalnızca doğrusal olarak ayrılabilen problemlere çözüm üretebilmektedir.

EKK algoritması gürültüye duyarlı olan Perceptron’dan daha güçlüdür. Ortalama karesel hatayı minimize eden EKK algoritmasının Perceptrona göre gürültüye duyarlılığı daha azdır.

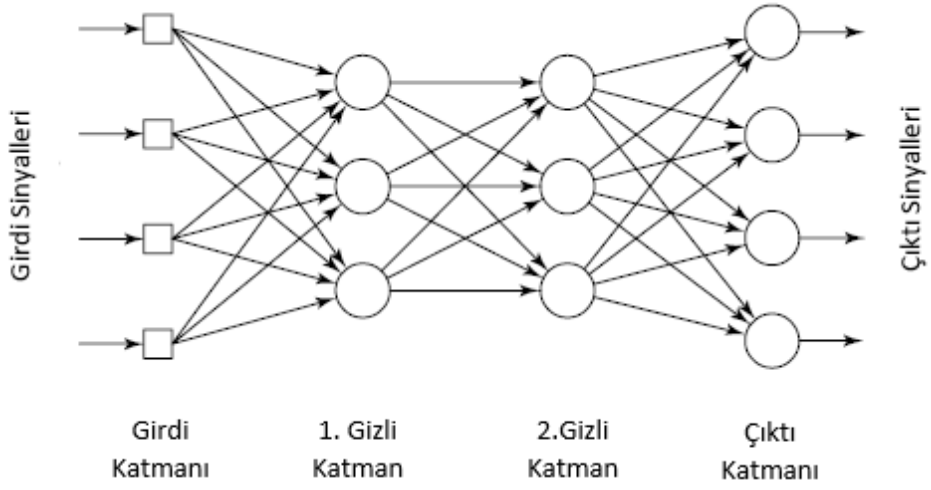
EKK algoritması kendisine Perceptrondan çok daha pratik kullanımlar bulmuştur. Özellikle dijital sinyal işleme alanında sıklıkla kullanılmıştır.

4.2 Çok Katmanlı Algılayıcılar

ÇKA modelinde bir girdi, bir veya birden fazla gizli katman ve bir adet te çıktı katmanı bulunmaktadır.

Çok katmanlı bir sinir ağındaki her katman kendi özel işlevine sahiptir. Girdi katmanı dış dünyadan gelen sinyalleri alır ve bunlar üzerinde hiç bir işlem yapmadan bir sonraki gizli katmandaki tüm nöronlara yeniden dağıtır. Gizli katmandaki nöronlar

özellikleri tespit eder, nöronların ağırlığı girdi kalıplarında gizli olan özellikleri temsil eder. Çıktı katmanı gizli katmanın çıkış sinyallerini işleyip tüm ağın çıktılarını belirler. Bir gizli katman girdi sinyallerinin herhangi bir sürekli fonksiyonunu temsil edebilirken, iki gizli katman girdi sinyallerinin süreksiz fonksiyonunu bile temsil edebilir Negnevitsky (2005). Şekil 4.2'de iki gizli katmana sahip YSA gösterilmiştir.



Şekil 4.2 Çok katmanlı algılayıcı modeli (Mohammadi, 2019)

Gizli katmanlar kendi çıktısını gizler. Gizli katmandaki nöronlar, ağın giriş / çıkış davranışıyla gözlenemez. Gizli katmanın istenen çıktısının ne olması gerektiğinin kesin bir yolu yoktur. Başka bir deyişle gizli katmanın istenen çıktısı katmanın kendisi tarafından belirlenir. Ticari YSA'lar üç ve bazen de dört katman içerir, bunlardan biri ya da ikisi gizli katmandır. Her katman 10 ila 1000 nöron içerebilir. Deneysel sinir ağları beş ya da altı katmana sahip olabilir, bunlar üç ya da dört gizli katmandan oluşur ve milyonlarca nöron kullanır. Ancak en pratik uygulamalarda yalnızca üç katman kullanır çünkü her ek katman üssel olarak artan hesaplama yükü getirir Negnevitsky (2005).

ÇKA'lar için yüzden fazla farklı öğrenme algoritması mevcuttur ancak en yaygın yöntem geri yayılımdır.

Tek katmanlı algılayıcılarda her girdi için yalnızca bir ağırlık ve bir çıktı değeri bulunur. Ancak çok katmanlı algılayıcılarda çok fazla ağırlık vardır ve her biri birden fazla çıktıya katkıda bulunur.

4.2.1 Geri Yayılım (Backpropagation) Algoritması

Frank Rosenblatt tarafından geliştirilen Perceptron öğrenme kuralı ile Bernard Widrow ve Marcian Hoff'un tasarladığı EKK algoritması yalnızca doğrusal olarak ayrılabilen sınıflandırma problemlerini çözebilmek gibi sınırlamalara sahiptirler. Bu sınırlamaların farkında olan Rosenblatt ve Widrow, doğrusal olmayan problemlerin çözülmesi için çok katmanlı ağların kullanılması gerektiğini önerdiler fakat ağlarını eğitmek için algoritmalarını genelleştiremediler.

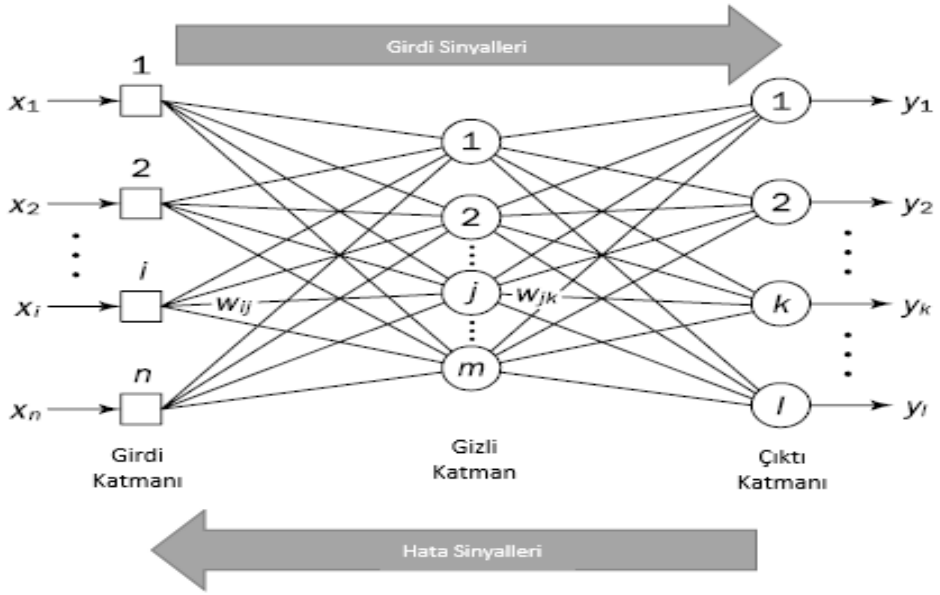
Çok katmanlı ağları eğitmek için geliştirilen ilk algoritma 1974 yılında Paul Werbos'un tezinde yer aldı Werbos (1974). Bu tez algoritmayı genel ağlar bağlamında özel bir durum olarak sundu ve tüm sinir ağları için yaygınlaştırılmadı. 1980'lerin ortalarına doğru, geri yayılım algoritması bağımsız olarak Geoffrey Hinton ve Ronald Williams, Williams ve Hinton (1986), David Parker, Parker (1985), ve Yann Le Cun, LeCun (1985) tarafından yeniden keşfedildi. Algoritma David Rumelhart ve James Mc-Clelland 'ın yönettiği Parallel Distributed Processing Group tarafından tasarlanan "Parallel Distributed Processing" kitabına dahil edilmesiyle popülerleşti Rumelhart ve Mcclelland (1986). Bu kitabın yayınlanması ile geri yayılım algoritması çok katmanlı algılayıcılarda yaygın bir şekilde kullanılmaya başladı.

En küçük kareler algoritmasının genellemesi olarak karşımıza çıkan geri yayılım algoritması çok katmanlı ağları eğitmek için kullanılır. En küçük kareler algoritmasında olduğu gibi geri yayılım algoritması da ortalama karesel hatayı indeks alan dik bir iniş algoritmasıdır. EKK ile geri yayılım arasındaki fark türevlerin hesaplanma şeklidir. Tek katmanlı doğrusal bir ağ için hata ağın ağırlıklarının doğrusal bir fonksiyonudur ve ağırlıklara göre türevleri kolayca hesaplanabilir. Doğrusal olmayan aktivasyon fonksiyonuna sahip çok katmanlı ağlarda ağın ağırlıkları ile hata değeri arasındaki ilişki daha karmaşıktır. Türev hesaplamak için zincir kuralı kullanılır.

Geri yayılım sinir ağında öğrenme algoritmasının ileri ve geri olmak üzere olmak üzere iki safhası vardır. İleri yayılım safhasında, eğitim setinde bulunan girdi

değerleri ağın girdi katmanına iletilir. Alınan girdi değerleri çıktı katmanından ağın çıktıları üretilene kadar katman katman yayılır. Ağın çıktıları ile gerçek çıktı değerleri karşılaştırılarak ağın hatası hesaplanır. Geri yayılım safhasında ise ağın hatası çıktı katmanından geriye doğru girdi katmanına kadar dağıtılır. Hata yayılırken ağırlıklar değiştirilir. Herhangi bir sinir ağında olduğu gibi geri yayılım ağında da nöronlar arasındaki bağlantılar nöronlar tarafından kullanılan aktivasyon fonksiyonu ve ağırlıkları ayarlama prosedürünü veren öğrenme algoritması ile belirlenir. Genellikle bir geri yayılım ağı üç veya dört katmandan oluşur. Katmanlar tamamen bağlıdır yani her katmanda bulunan nöronların her biri kendinden bir sonraki katmanda bulunan nöronların her biri ile bağlıdır.

Geri yayılım öğrenme kuralı için üç katmanlı yapı Şekil 4.3'de gösterilmiştir. i, j ve k indisleri sırasıyla girdi, gizli ve çıktı katmanlarındaki nöronları ifade eder. $x_1, x_2, ..., x_n$ ağ boyunca soldan sağa yayılan girdi sinyallerini, $e_1, e_2, ..., e_l$ ağ boyunca sağdan sola yayılan hata sinyallerini göstermektedir. w_{ij} girdi katmanındaki $i.$ nöron ile gizli katmandaki $j.$ nöron arasındaki bağlantının ağırlığını, w_{jk} gizli katmandaki $j.$ nöron ile çıktı katmanındaki $k.$ nöron arasındaki bağlantının ağırlık değerini göstermektedir.



Şekil 4.3 Geri yayılım sinir ağı modeli (Elmsley, 2019)

4.2.1.1 Çıktı Katmanında Bulunan Ağırlıkların Güncellenmesi

p. iterasyonda, çıktı katmanındaki k. nöronun çıktısındaki hata değeri, $d_k(p)$ gerçek çıktı değeri iken;

$$e_k(p) = d_k(p) - y_k(p) \tag{4.5}$$

olarak ifade edilir. k. çıktı nöronunun ani hata değeri, $\frac{1}{2}e_k(p)^2$ şeklinde ifade edilir. Çıktı katmanındaki tüm nöronların ani hata değerlerinin toplamı aşağıdaki gibi ifade edilir.

$$E(p) = \frac{1}{2}\sum_{k=1}^{l} e_k(p)^2 \tag{4.6}$$

N eğitim kümesinde yer alan tüm örnek sayısı olmak üzere ortalama karesel hata Denklem 4.7 ile bulunur.

$$E_{ort}(p) = \frac{1}{N}E(p) \tag{4.7}$$

$E_{ort}(p)$ eğitim seti için öğrenmenin bir ölçüsü olarak maliyet fonksiyonunu temsil eder. Geri yayılım algoritması ile maliyet fonksiyonunu minimum yapacak serbest parametrelerin ayarlanması amaçlanmaktadır. Dik iniş yöntemi kullanılarak öğrenme esnasında maliyet fonksiyonu minimize edilmektedir. Ağırlıkların değişim yönü $E(p)$'nin w_{jk} ağırlıklarına göre hesaplanan negatif gradyendir . Bu sayede hatayı azaltacak yönde ağırlıklar değiştirilebilmektedir.

Çıktı katmanındaki k. nöronun gelen girdiler toplamı

$$v_k(p) = \sum_{j=0}^{m} w_{jk}(p)y_j(p) \tag{4.8}$$

m toplam girdilerin sayısını (sapma(bias) hariç), $y_j(p)$ gizli katmandaki j. nöronun çıktısını, w_{0k} sapma elemanını gösterir; sapma elemanının girdisi her zaman 1 olduğundan $y_0 = +1$ olur. p. iterasyonda k nöronunun ürettiği çıktı,

$$y_k(p) = \phi_k(v_k(p)) \tag{4.9}$$

ϕ_k, $k.$ nörona uygulanan aktivasyon fonksiyonunu gösterir.

Geriye yayılım algoritmasında ağırlıkların değişimi Widrow – Hoff kuralına benzer şekilde gerçekleştirilir. Hata fonksiyonuna ağın ağırlıklarına göre türev işlemi uygulanarak ağın gradyeni hesaplanabilmektedir. Gradyen zincir kuralı uygulanarak aşağıdaki gibi ifade edilir.

$$\frac{\partial E(p)}{\partial w_{jk}(p)} = \frac{\partial E(p)}{\partial e_k(p)} \frac{\partial e_k(p)}{\partial y_k(p)} \frac{\partial y_k(p)}{\partial v_k(p)} \frac{\partial v_k(p)}{\partial w_{jk}(p)} \tag{4.10}$$

Denklem 4.6 'da $e_k(p)$'ye göre her iki tarafın türevi alınırsa

$$\frac{\partial E(p)}{\partial e_k(p)} = e_k(p) \tag{4.11}$$

elde edilir. Denklem 4.5 'da $y_k(p)$'ye göre her iki tarafın türevi alınırsa

$$\frac{\partial e_k(p)}{\partial y_k(p)} = -1 \tag{4.12}$$

elde edilir. Denklem 4.9 'da $v_k(p)$'ye göre her iki tarafın türevi alınarak

$$\frac{\partial y_k(p)}{\partial v_k(p)} = \phi'(v_k(p)) \tag{4.13}$$

bulunur. Son olarak, denklem 4.8 'de $w_{jk}(p)$'ye göre her iki tarafın türevi alınırak

$$\frac{\partial v_k(p)}{\partial w_{jk}(p)} = y_k(p) \tag{4.14}$$

denklem 4.14 elde edilir.

$\frac{\partial E(p)}{\partial w_{jk}(p)}$ kısmı türevi 4.10, 4.11 ve 4.14 denklemleri kullanılarak 4.15'deki gibi bulunur.

$$\frac{\partial E(p)}{\partial w_{jk}(p)} = -e_k(p)\phi'(v_k(p))y_k(p) \tag{4.15}$$

Ağırlık değişim miktarı, Δw_{jk}, Denklem 4.16 ile gösterilir.

$$\Delta w_{jk}(p) = -\eta \frac{\partial E(p)}{\partial w_{jk}(p)} \tag{4.16}$$

η öğrenme oranıdır. Denklem 4.16'deki $-$ işareti ağırlık uzayındaki dik inişi temsil eder. Böylece ağırlık değişim miktarı denklem 4.15 ve 4.16 kullanılarak

$$\Delta w_{jk}(p) = \eta \delta_k(p) y_k(p) \tag{4.17}$$

4.17'teki gibi hesaplanır. Burada $\delta_k(p)$ yerel gradyen (hata bilgisi terimi olarak ta adlandırılır) şu şekilde tanımlanır:

$$\begin{aligned} \delta_k(p) &= -\frac{\partial E(p)}{\partial v_k(p)} \\ &= -\frac{\partial E(p)}{\partial e_k(p)} \frac{\partial e_k(p)}{\partial y_k(p)} \frac{\partial y_k(p)}{\partial v_k(p)} \\ &= e_k(p) \phi_k'(v_k(p)) \end{aligned}$$

Aktivasyon fonksiyonunun türevinin, nöronun hata değeri ile çarpımı, o nöronun yerel gradyenini verir Haykin (1999).

4.2.1.2 Gizli Katmanda Bulunan Ağırlıkların Güncellenmesi

Gizli katmanda bulunan nöronların çıktı katmanındakiler gibi gerçek çıktı değerleri yoktur. Bu sebeple gizli katmandaki nöronlar bağlı bulundukları tüm nöronların sahip olduğu hata değerinden geriye dönük bir şekilde etkilenecektir.

Gizli katmandaki herhangi bir j nöronu için $\delta_j(p)$ yerel gradiyeni şu şekilde yeniden tanımlanır:

$$\delta_j(p) = -\frac{\partial E(p)}{\partial y_j(p)} \frac{\partial y_j(p)}{\partial v_j(p)} \tag{4.18}$$

$$= -\frac{\partial E(p)}{\partial y_j(p)} \phi_j'(v_j(p)), \quad j \quad gizli \quad katmanda \tag{4.19}$$

ikinci satırda denklem 4.13 kullanılmıştır. $\frac{\partial E(p)}{\partial y_j(p)}$ türevini hesaplamak için aşağıdaki şekilde ilerleyebiliriz. Toplam hata denklemi 4.6'da $y_j(p)$'ye göre türev alınırsa

$$\frac{\partial E(p)}{\partial y_j(p)} = \sum_{k=1}^{l} e_k \frac{\partial e_k(p)}{\partial y_j(p)} \tag{4.20}$$

elde edilir. l çıktı katmanında bulunan nöron sayısını göstermektedir. $\frac{\partial E(p)}{\partial y_j(p)}$ türevini bulmak için zincir kuralını uygulayıp denklem 4.20'i yeniden yazarsak 4.21'deki denklem eşitliği elde ederiz.

$$\frac{\partial E(p)}{\partial y_j(p)} = \sum_{k=1}^{l} e_k(p) \frac{\partial e_k(p)}{\partial v_k(p)} \frac{\partial v_k(p)}{\partial y_j(p)} \tag{4.21}$$

Çıktı katmanındaki $k.$ nöronun hata değerini veren denklemi aşağıdaki gibi yeniden yazalım.

$$e_k(p) = d_k(p) - y_k(p) = d_k(p)\phi_k(v_k(p)) \tag{4.22}$$

Buradan,

$$\frac{\partial e_k(p)}{\partial v_k(p)} = -\phi_k'(v_k(p)) \tag{4.23}$$

elde ederiz. Çıktı katmanındaki $k.$ nöronun girdisini veren denklem 4.8'in $y_j(p)$'ye göre türevi alınırsa, denklem bulunur.

$$\frac{\partial v_k(p)}{\partial y_j(p)} = w_{jk}(p) \tag{4.24}$$

İstenen kısmi türev denklem 4.21, 4.23 ve 4.24 kullanarak

$$\frac{\partial E(p)}{\partial y_j(p)} = -\sum_{k=0}^{l} e_k(p)\phi_k'(v_k(p))w_{jk}(p) \tag{4.25}$$

$$= -\sum_{k=0}^{l} \delta_k(p) w_{jk}(p) \tag{4.26}$$

elde edilir. Son olarak, geri yayılım algoritmasının yerel gradyeni $\delta_j(p)$ denklem 4.19 ve denklem 4.26 kullanılarak bulunur.

$$\delta_j(p) = \phi_j'(v_j(p)) \sum_{k=0}^{l} \delta_k(p) w_{jk}(p) \tag{4.27}$$

Ağın yerel minimumlara takılmasını önlemek için Rumelhart, geri yayılım algoritmasında ağırlıkların değişim denklemine momentum terimini eklemiştir. Ağırlık değişim denklemi momentum teriminin eklenmesi ile aşağıdaki hali alır Haykin (1999).

$$w(p+1) = w(p) + \Delta w(p) \tag{4.28}$$

$$\Delta w_{jk}(p) = \eta \delta_k(p) y_k(p) + \alpha \Delta w_{jk}(p) \tag{4.29}$$

BÖLÜM BEŞ
ARAŞTIRMA METODOLOJİSİ

5.1 Geliştirilen Yazılımın İncelenmesi

Tez çalışmasında kullanılmak üzere C#.NET programlama dilinde bir yazılım geliştirilmiş ve yayınlanmıştır. Yazılım tahmin ve sınıflandırma problemleri için YSA oluşturulup test edilmesine olanak sağlamaktadır. Yazılımın ÇKA modelinde geri yayılım algoritması kullanılmaktadır. Ağ için gerekli tüm parametreler kullanıcı tarafından belirlenmektedir. Veri seti tamamen veya eğitim ve test setleri ayrıştırılarak (.txt) formatında yazılıma verilir. Veriler normalize edilmiş veya edilmemiş şekilde verilebilmektedir. Yazılımda doğrusal, max-min ve z-score olmak üzere 3 farklı normalizasyon seçeneği mevcuttur. Yazılımda aktivasyon fonksiyonu olarak doğrusal, sigmoid, gaussian, rasyonel sigmoid ve tanjant hiperbolik fonksiyonları kullanılabilir. Ağın yapısı katman sayısı ve katmanlardaki nöron sayısı, momentum katsayısı, öğrenme oranı, iterasyon sayısı, başlangıç ağırlıklarının şeçimi, eğitimi ve sonlandırma koşulu kullanıcı tarafından gerçekleştirilmektedir. Eğitim sırasında oluşan hatanın grafiği çizilmektedir. Test işleminin ardından karmaşıklık matrisi gözlemlenerek ağın performansı ölçülebilmekdir. Ağın yapısı ve kullanılan tüm parametreler, eğitilen ağın bağlantı ve eşik değerleri, ağın çıktıları, (.txt) uzantılı dosya formatında dizine kaydedilmektedir. Kaydedilmiş olan ağırlık ve eşik değerleri aynı topolojiye sahip bir başka ağa başlangıç değerleri olarak verilebilmekte ve başarımın arttırıldığı gözlenmektedir Arı ve Berberler (2017). Yazılımın kaynak koduna "`https://github.com/ayseari/NeuralNetworks.git`" linkinden erişilebilmektedir.

5.2 Veri Kümesi Havuzu

UCI Makine Öğrenme Havuzu makine öğrenme topluluklarının makine öğrenme algoritmalarının ampirik analizi için kullandığı veri tabanları, alan teorileri ve veri

jeneratörlerinin bir koleksiyonudur. Arşiv 1987'de David Aha ve UC Irvine'deki diğer lisansüstü öğrenciler tarafından bir ftp arşivi olarak oluşturulmuştur. O zamandan beri dünya çapında araştırmacılar, eğitimciler ve öğrenciler tarafından makine öğrenme veri setlerinin birincil kaynağı olarak oldukça yaygın bir şekilde kullanılmaktadır. 1000'den fazla atıf yapılması ve tüm bilgisayar bilimlerinde en çok kullanılan 100 "bildiri" arasında yer alması arşivin ne kadar etkin olduğunu göstermektedir. Web sitesinin şu anki versiyonu Arthur Asuncion ve David Newman tarafından 2007'de tasarlanmıştır ve bu proje Massachusetts Amherst Üniversitesi'ndeki Rexa.info ile işbirliği içindedir. Maddi desteğini Ulusal Bilim Vakfı'ndan almıştır UCI Machine Learning Repository (2017).

5.2.1 Veri Kümesi Bilgileri

UCI makine öğrenme havuzunda, kalp hastalıkları ile ilgili 4 veri tabanı bulunmaktadır. Verilere ait tüm nitelikler sayısal değerlidir. Veriler aşağıda belirten veri kümelerinde bulunmaktadırlar.

1. Cleveland Kliniği Vakfı (Cleveland Clinic Foundation) (cleveland.data)
2. Macaristan Kardiyoloji Enstitüsü, Budapeşte (Hungarian Institute of Cardiology, Budapest) (hungarian.data)
3. V.A. Tıp Merkezi, Long Beach, CA (long-beach-va.data)
4. Üniversite Hastanesi, Zürih, İsviçre (switzerland.data)

Her bir veritabanı aynı örnek biçimini kullanır. Veritabanlarında 76 tane özellik bulunmakta olup bunların sadece 14'ü bilimsel araştırmalarda yaygın bir şekilde kullanılmaktadır UCI Machine Learning Repository (2017).

Tablo 5.1 'de veri tabanlarında (kayıp veriler dahil) bulunan veri sayıları sınıflara göre gösterilmiştir. Sınıflar $0, 1, 2, 3, 4$ ile belirtilmiştir. 0 sınıfı hasta olmayan kişileri

ifade ederken $1, 2, 3, 4$ sınıfları hastalığın derecelerini göstermektedir. Yani 1.sınıfa giren bir hasta hastalığın 1.evresindedir şeklinde ifade edilebilir.

Tablo 5.1 Veri sayılarına göre kalp hastalığı veri kümeleri

Sınıflar	0	1	2	3	4	Total
Cleveland	164	55	36	35	13	303
Hungarian	188	37	26	28	15	294
Switzerland	8	48	32	30	5	123
Long Beach VA	51	56	41	42	10	200

Bu tez çalışmasında kayıp veriler hariç 297 örnek, 13 özellik ve 5 sınıf içeren Cleveland veri kümesi kullanılmıştır.

5.2.2 Veri Öznitelikleri

Cleveland veri seti kişinin kalp rahatsızlıklarıyla ilgili normal ve anormal kişilerin sınıflanması ile ilgilidir. Veri seti toplam 297 kayıttır. Veri seti bir kısmı eğitim ve bir kısmı test işleminde kullanılmak üzere ayrılmıştır. Veriler 13 öznitelikten (girdi) ve 5 sınıftan (çıktılar) oluşur.

Tablo 5.2'de hastalık teşhisinde kullanılan 13 özellik gösterilmiştir.

Tablo 5.2 Hastalık teşhisi için kullanılan özellikler

Özellik	**Açıklama**	**Aralık**
Yaş	Hasta yaşı	29-77
Cinsiyet	1=erkek 2=kadın	{0,1}
Göğüs Ağrısı Tipi	1: tipik anjina 2: atipik anjina 3: anjinal olmayan ağrı 4: asemptomatik	{1, 2, 3, 4}
Tansiyon	mmHg cinsinden	94-200
Kolesterol	mg / dl olarak	126-564
Kan Şekeri	(açlık kan şekeri> 120 mg / dl) (1 = doğru, 0 = yanlış)	{0,1}
Elektrokardiyografi Sonuçları	0: normal 1: ST-T dalga anormalliği 2: Olası veya kesin sol ventrikül hipertrofisi	{0, 1, 2}
Thalach	Maksimum Kalp Atış Hızı	71-202
Egzersize bağlı anjina	1 = evet, 0 = hayır	{0, 1, 2}
Oldpeak	egzersizle indirgenen ST depresyonu	0-6,2
Eğim	ST segmentinin egzersiz eğimi 0: yukarı 1: yatay 2: aşağı	{0, 1, 2}
CA	floroskopi ile renklendirilen büyük damarların sayısı	{0, 1, 2, 3 }
Thal	3 = normal 6 = sabit bozukluk 7 = geri çevrilebilir bozukluk	{3, 6, 7 }

5.3 Uygun Mimari Seçimi

Geri yayılım algoritması ile bir gizli katman kullanılarak gizli katmandaki node sayısı, iterasyon sayısı, öğrenme oranı ve momentumun artırıldığı program MATLAB'ta yazılmıştır. Bu programın yazılmasındaki amaç aynı eğitim seti ve test seti üzerinde momentum, öğrenme oranı, iterasyon sayısı ve gizli katmandaki node sayısı değişiminin test kümesinde elde edilen başarı oranına etkisinin gözlemlenmesi ve en yüksek doğruluk oranının elde edildiği mimari ve paremetrelerin belirlenmesidir.

Program iki farklı yöntemle çalıştırılmıştır.

- İlk yöntemde oluşturulan her bir mimari yapısı başlangıç ağırlıkları rastgele seçilerek 10'ar kez eğitilip test edilmiş ve bu 10 testte elde edilen ortalama doğruluk oranları hesaplanmıştır. Bu yöntem için geliştirilen kod EK-1'de verilmiştir.
- İkinci yöntemde ise oluşturulan her bir mimari yapısı için başlangıç ağırlıkları $0,5$ sabit değer vektörü seçilerek her bir yapı için birer kez eğitip test edilerek doğruluk oranları hesaplanmıştır. Bu yöntemin kodu EK-2'de bulunmaktadır.

Cleveland veri setinin %80'i eğitim seti, %20'si ise test seti olarak ikiye ayrılmıştır. Veriler max-min normalizasyon yöntemi ile normalize edilerek geri yayılım algoritması sigmoid aktivasyon ile eğitilmiştir. Her bir sınıfta %80'i eğitim, %20'si test verisi olarak ayrılan veri setindeki eğitim ve test veri sayılarının sınıflara göre dağılımı Tablo 5.3'te gösterilmiştir.

Gizli katmandaki node sayısı, iterasyon sayısı, öğrenme oranı, momentum değeri her bir eğitimde arttırılarak defalarca eğitilip test edilmiştir. Bu değerlerin başlangıç, bitiş ve artış değerleri Tablo 5.4'te gösterilmiştir.

Bu şekilde değerler sürekli arttırılarak 66150 kez eğitilip test edilmiştir.

Tablo 5.3 Eğitim ve test kümesinin sınıflara dağılımı

Sınıf	Eğitim kümesi	Test kümesi
1	128	32
2	43	11
3	28	7
4	28	7
5	10	3

Tablo 5.4 Eğitimde kullanılan değer aralıkları

Özellik	Başlangıç Değeri	Bitiş Değeri	Artış Miktarı
Gizli Katmandaki Node Sayısı	1	15	1
İterasyon Sayısı	500	5000	500
Öğrenme Oranı	0	1	$0,05$
Momentum Değeri	0	1	$0,05$

5.3.1 Başlangıç Ağırlıkları Rastgele Seçilerek Mimari Belirleme

Tablo 5.4'te gösterilen mimariler ile veri seti eğitilip test edilmiştir. Her bir mimari yapısı 10'ar kez eğitilip test edilmiş ve bu 10 testte elde edilen ortalama doğruluk oranı maksimum yüzde 65 olarak elde edilmiştir. Elde edilen en yüksek doğruluk oranı Tablo 5.5'de gösterilen mimari parametrelerinin farklı kombinasyonlarında seçilen yapılardan oluşmuştur.

Tablo 5.5 En iyi doğruluk oranının elde edildiği mimari için parametre aralıkları

Iterasyon Sayısı	Node Sayısı	Öğrenme Oranı	Momentum	Doğruluk Oranı
$[1500-5000]$	9	$[0-1]$	$[0-1]$	65

Tablo 5.5'de görüldüğü gibi gizli katmanda 9 node seçilerek uygun paremetre kombinasyonu seçildiğinde $\%65$ doğruluk oranı elde edilmektedir. Maliyeti en aza indirgemek için en yüksek doğruluk oranını veren en küçük YSA mimarisinde 1500 iterasyon sayısı seçilmiştir. Minimum maliyetli mimari paremetreleri tablo Tablo 5.6'da gösterilmiştir.

Gizli katmandaki node sayısının 9, iterasyon sayısının 1500, öğrenme oranının $0,5$ olarak seçilip momentumun $[0,1]$ aralığında $0,05$ artışla değiştiği mimarilerin her biri için elde edilen doğruluk oranı, eğitim seti hatası ve test seti hatalarının ortalama değerleri Tablo 5.7'de gösterilmiştir.

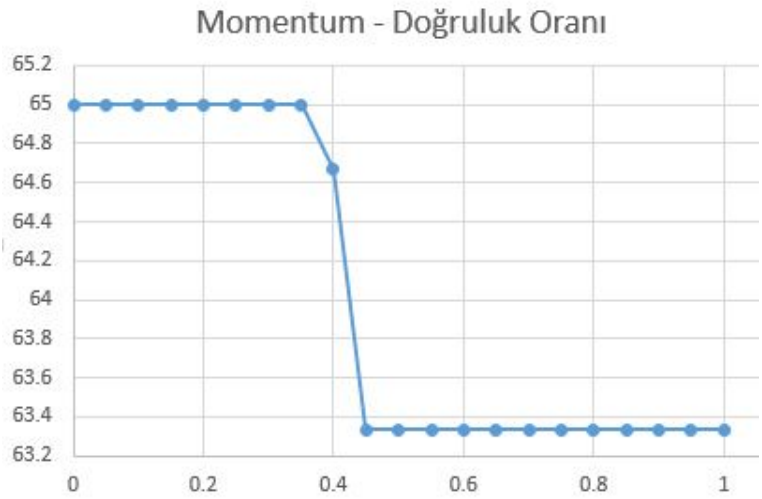
Tablo 5.6 En iyi doğruluk oranının elde edildiği minimal mimari için parametre aralıkları

Iterasyon Sayısı	Node Sayısı	Öğrenme Oranı	Momentum	Doğruluk Oranı
$[1500]$	9	$[0-1]$	$[0-1]$	65

Tablo 5.7 Doğruluk oranının momentum değerine göre değişim değerleri

Momentum	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
$0,00$	65	$2,24397$	$3,35532$
$0,05$	65	$2,24391$	$3,35509$
$0,10$	65	$2,24385$	$3,35486$
$0,15$	65	$2,24380$	$3,35463$
$0,20$	65	$2,24374$	$3,35440$
$0,25$	65	$2,24368$	$3,35417$
$0,30$	65	$2,24363$	$3,35394$
$0,35$	65	$2,24357$	$3,35372$
$0,40$	64.66667	$2,24352$	$3,35349$
$0,45$	63.33333	$2,24346$	$3,35326$
$0,50$	63.33333	$2,24341$	$3,35304$
$0,55$	63.33333	$2,24335$	$3,35282$
$0,60$	63.33333	$2,24329$	$3,35259$
$0,65$	63.33333	$2,24324$	$3,35237$
$0,70$	63.33333	$2,24318$	$3,35215$
$0,75$	63.33333	$2,24313$	$3,35192$
$0,80$	63.33333	$2,24307$	$3,35170$
$0,85$	63.33333	$2,24302$	$3,35147$
$0,90$	63.33333	$2,24297$	$3,35125$
$0,95$	63.33333	$2,24291$	$3,35102$
$1,00$	63.33333	$2,24286$	$3,35079$

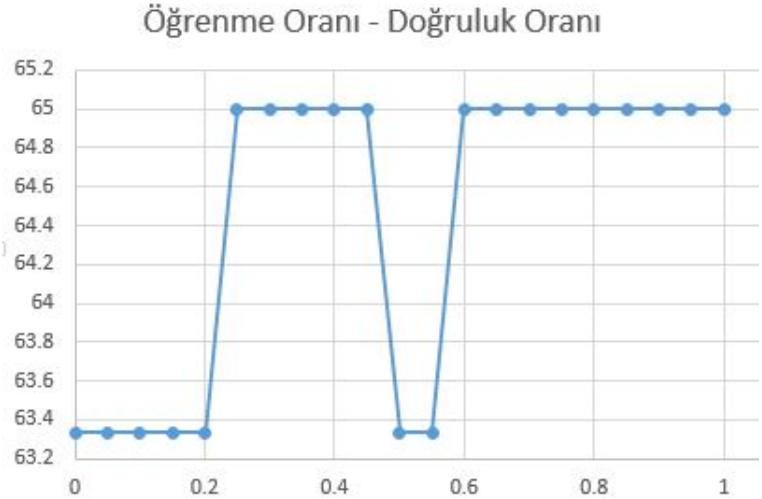
Şekil 5.1, Tablo 5.7'deki momentum değerinin ortalama doğruluk oranına değişimini göstermektedir.



Şekil 5.1 Doğruluk oranının momentuma göre değişimi

Aynı eğitim ve test seti için gizli katmandaki node sayısının 9, iterasyon sayısının 1500, momentum değerinin $0,5$ olarak seçilip öğrenme oranının $[0,1]$ aralığında $0,05$ artışla değiştiği mimarilerin her biri için elde edilen doğruluk oranı, eğitim seti hatası ve test seti hatalarının ortalama değerleri Tablo 5.8'de gösterimiştir.

Doğruluk oranının öğrenme oranına göre değişim grafiği Şekil 5.2'da gösterilmiştir.



Şekil 5.2 Doğruluk oranının öğrenme oranına göre değişimi

Aynı eğitim ve test seti üzerinde 1500 iterasyon, $0,5$ öğrenme oranı ve $0,5$ momentum değeri ile gizli katmandaki düğüm sayısının $[1,15]$ aralığında

Tablo 5.8 Doğruluk oranının öğrenme oranına göre değişim değerleri

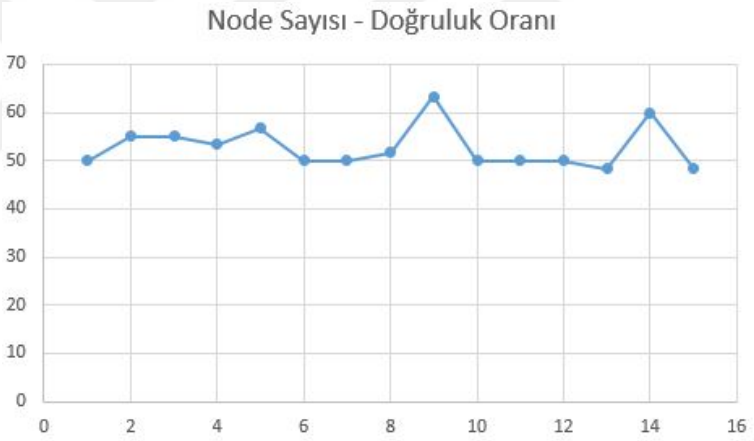
Öğrenme Oranı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
0,00	63,33333	2,24999	3,38758
0,05	63,33333	2,24991	3,38723
0,10	63,33333	12,24968	3,38611
0,15	63,33333	2,24929	3,38410
0,20	63,33333	2,24877	3,38108
0,25	65	2,24813	3,37703
0,30	65	2,24737	3,37216
0,35	65	2,24651	3,36703
0,40	65	2,24557	3,36222
0,45	65	2,24453	3,35768
0,50	63,33333	2,24341	3,35304
0,55	63,33333	2,24223	3,34753
0,60	65	2,18866	3,24736
0,65	65	2,18616	3,25526
0,70	65	2,18452	3,26777
0,75	65	2,18447	3,2682
0,80	65	2,18447	3,2682
0,85	65	2,18447	3,2682
0,90	65	2,18447	3,2682
0,95	65	2,18447	3,2682
1,00	65	2,18447	3,2682

değiştirilerek elde edilen mimarilerde ağ 10'er kez eğitilip test edildiğinde elde edilen ortalama doğruluk oranları, eğitim seti hataları ve test seti hataları Tablo 5.9'da gösterilmiştir.

Doğruluk oranının gizli katmandaki düğüm sayısına göre değişim grafiği Şekil 5.3'te gösterilmiştir.

Tablo 5.9 Doğruluk oranının gizli katmandaki düğüm sayısına göre değişim değerleri

Node Sayısı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
1	50	4, 96665	3, 00445
2	55	4, 74392	3, 20814
3	55	4, 41761	3, 27094
4	53, 33333	4, 05176	3, 22173
5	56, 66667	3, 38960	3, 49717
6	50	3, 42659	3, 76152
7	50	2, 87525	3, 75202
8	51, 66667	2, 44033	3, 68637
9	63, 33333	2, 24341	3, 35304
10	50	2, 13175	3, 89440
11	50	1, 74063	3, 78663
12	50	1, 74249	3, 82615
13	48, 33333	1, 42999	4, 00284
14	60	1, 66477	3, 47499
15	48, 33333	1, 23263	3, 78171



Şekil 5.3 Doğruluk oranının gizli katmandaki düğüm sayısına göre değişimi

Aynı eğitim ve test seti üzerinde gizli katmandaki düğüm sayısı 9, öğrenme oranı ve momentum değerlerinin $0, 5$ seçilip iterasyon sayısı $[500, 5000]$ aralığında 500'ler arttırılarak her bir mimari için 10'ar kez eğitilip test edilen ağın ortalama doğruluk oranı, eğitim ve test seti hata değerleri Tablo 5.10'da gösterilmiştir.

Doğruluk oranının iterasyon sayısına göre değişim grafiği Şekil 5.4'de gösterilmiştir.

Tablo 5.10 Doğruluk oranının iterasyon sayısına göre değişim değerleri

İterasyon Sayısı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
500	58, 33333	2, 63463	0, 00571
1000	60	2, 37698	0, 00572
1500	63, 33333	2, 24341	0, 00571
2000	65	2, 18447	0, 00567
2500	65	2, 18447	0, 00565
3000	65	2, 18447	0, 00568
3500	65	2, 18447	0, 00567
4000	65	2, 18447	0, 00570
4500	65	2, 18447	0, 00569
5000	65	2, 18447	0, 00576

Şekil 5.4 Doğruluk oranının iterasyon sayısına göre değişimi

5.3.2 Başlangıç Ağırlıkları Sabit Seçilerek Mimari Belirleme

Tablo 5.4 'te gösterilen mimariler ile veri seti eğitilip test edilmiştir. Her bir mimari yapısında başlangıç ağırlıkları $0,5$ sabit değer vektörü seçilerek her bir yapı için birer kez eğitilip test edilmiş ve doğruluk oranı maksimum yüzde 61.66667 olarak elde edilmiştir.

En yüksek doğruluk oranını veren mimari parametreleri Tablo 5.11'de gösterilmiştir.

Maliyeti en aza indirgemek için en yüksek doğruluk oranını veren en küçük YSA mimarisi gizli katmanda 5 node seçilerek elde edilmiştir. Minimum maliyetli mimari

Tablo 5.11 En iyi doğruluk oranını veren mimari için parametre aralıkları

Iterasyon Sayısı	Node Sayısı	Öğrenme Oranı	Momentum	Doğruluk Oranı
$[500-1500]$	$5, 6, 14$	$[0-1]$	$[0-1]$	61.66667

paremetreleri Tablo 5.12 ' de gösterilmiştir.

Tablo 5.12 En iyi doğruluk oranını veren minimal mimari için parametre aralıkları

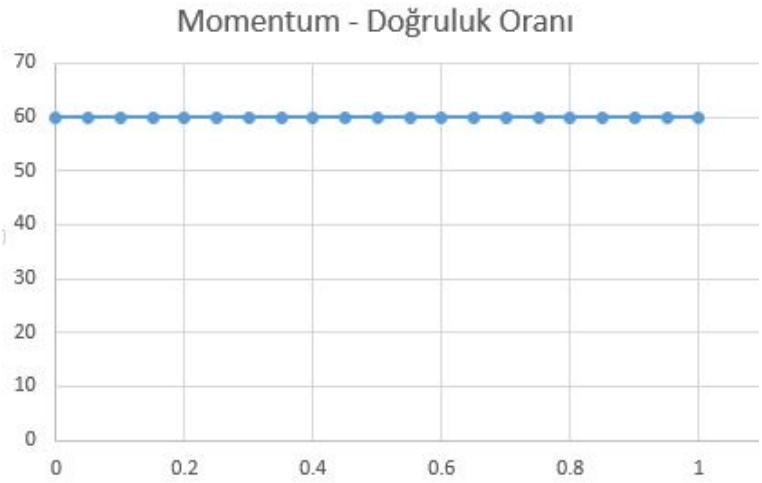
Iterasyon Sayısı	Node Sayısı	Öğrenme Oranı	Momentum	Doğruluk Oranı
$[500]$	5	$[0-1]$	$[0-1]$	61.66667

Gizli katmandaki node sayısının 5, iterasyon sayısının 500, öğrenme oranının $0,5$ olarak seçilip momentumun $[0,1]$ aralığında $0,05$ artışla değiştiği mimarilerin her biri için elde edilen doğruluk oranı, eğitim seti hatası ve test seti hatalarının ortalama değerleri Tablo 5.13'te gösterimiştir.

Tablo 5.13 Doğruluk oranının momentum değerine göre değişim değerleri

Momentum	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
$0,00$	60	$5,13702$	$0,02084$
$0,05$	60	$5,13675$	$0,02067$
$0,10$	60	$5,13648$	$0,02079$
$0,15$	60	$5,13621$	$0,02104$
$0,20$	60	$5,13594$	$0,02078$
$0,25$	60	$5,13568$	$0,02085$
$0,30$	60	$5,13542$	$0,02084$
$0,35$	60	$5,13516$	$0,02088$
$0,40$	60	$5,13490$	$0,02576$
$0,45$	60	$5,13464$	$0,02084$
$0,50$	60	$5,13439$	$0,02087$
$0,55$	60	$5,13414$	$0,02082$
$0,60$	60	$5,13389$	$0,02059$
$0,65$	60	$5,13364$	$0,02082$
$0,70$	60	$5,13340$	$0,02085$
$0,75$	60	$5,13315$	$0,02076$
$0,80$	60	$5,13291$	$0,02086$
$0,85$	60	$5,13267$	$0,02079$
$0,90$	60	$5,13243$	$0,02085$
$0,95$	60	$5,13219$	$0,02160$
$1,00$	60	$5,13196$	$0,02085$

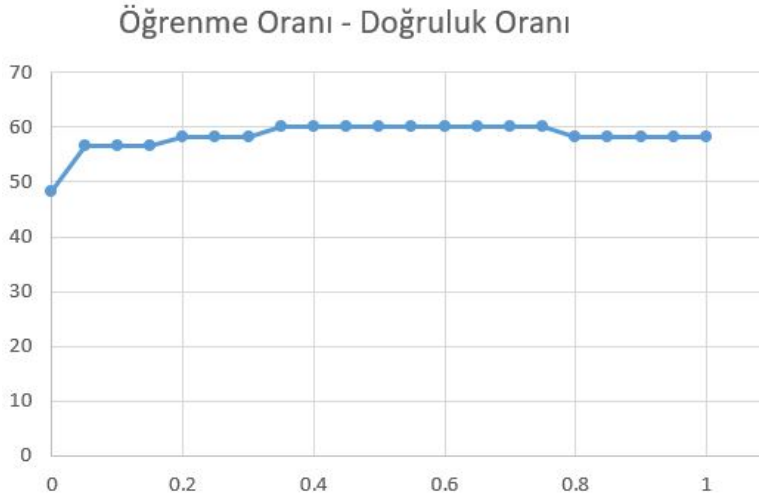
Şekil 5.5, Tablo 5.13'teki momentum değerinin ortalama doğruluk oranına değişimini göstermektedir.



Şekil 5.5 Doğruluk oranının momentuma göre değişimi

Aynı eğitim ve test seti için gizli katmandaki düğüm sayısının 5, iterasyon sayısının 500, momentum değerinin $0,5$ olarak seçilip öğrenme oranının $[0,1]$ aralığında $0,05$ artışla değiştiği mimarilerin her biri için elde edilen doğruluk oranı, eğitim seti hatası ve test seti hatalarının ortalama değerleri Tablo 5.14 gösterimiştir.

Doğruluk oranının öğrenme oranına göre değişim grafiği Şekil 5.6'de gösterilmiştir.



Şekil 5.6 Doğruluk oranının öğrenme oranına göre değişimi

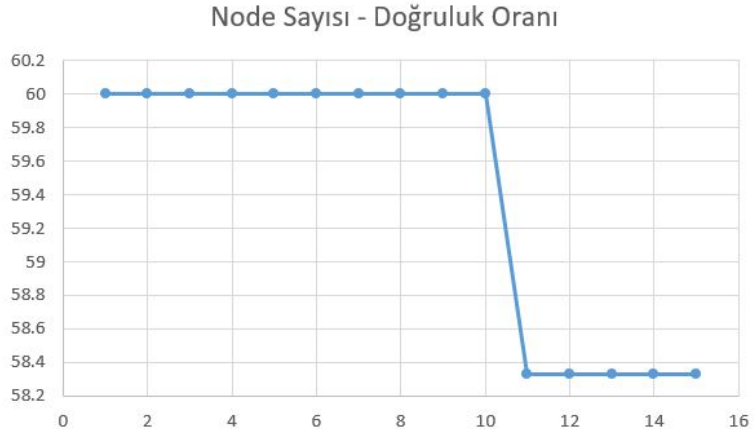
Aynı eğitim ve test seti üzerinde 500 iterasyon, $0,5$ öğrenme oranı ve $0,5$ momentum değeri ile gizli katmandaki düğüm sayısının $[1,15]$ aralığında değiştirilerek elde edilen

Tablo 5.14 Doğruluk oranının öğrenme oranına göre değişim değerleri

Öğrenme Oranı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
0,00	48,33333	12,45095	6,13830
0,05	56,66667	5,63019	2,88205
0,10	56,66667	5,31325	2,80547
0,15	56,66667	5,23151	2,80618
0,20	58,33333	5,19266	2,82489
0,25	58,33333	5,17173	2,83860
0,30	58,33333	5,15979	2,84625
0,35	60	5,15190	2,85050
0,40	60	5,14563	2,85317
0,45	60	5,13987	2,85649
0,50	60	5,13439	2,86216
0,55	60	5,12923	2,86980
0,60	60	5,12426	2,87898
0,65	60	5,11935	2,88945
0,70	60	5,11447	2,90041
0,75	60	5,09549	2,89829
0,80	58,33333	5,00335	2,94830
0,85	58,33333	4,97517	2,98418
0,90	58,33333	4,95097	2,99877
0,95	58,33333	4,92550	3,01131
1,00	58,33333	4,90382	3,02297

mimarilerde ağ 10'er kez eğitip test edildiğinde elde edilen ortalama doğruluk oranları, eğitim seti hataları ve test seti hataları Tablo 5.15'te gösterilmiştir.

Doğruluk oranının gizli katmandaki düğüm sayısına göre değişim grafiği Şekil 5.7'de gösterilmiştir.



Şekil 5.7 Doğruluk oranının gizli katmandaki düğüm sayısına göre değişimi

Tablo 5.15 Doğruluk oranının gizli katmandaki düğüm sayısına göre değişim değerleri

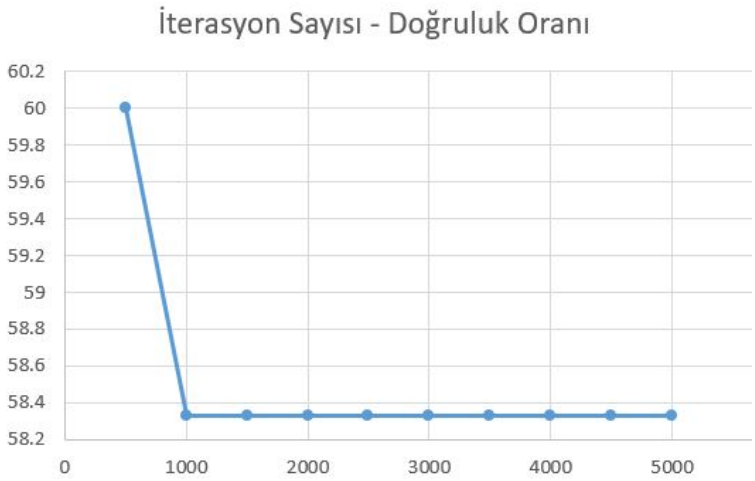
Node Sayısı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
1	60	5,10659	2,85310
2	60	5,11477	2,87769
3	60	5,12340	2,87009
4	60	5,12954	2,86543
5	60	5,13439	2,86216
6	60	5,13835	2,85988
7	60	5,14160	2,85845
8	60	5,14430	2,85758
9	60	5,14661	2,85699
10	60	5,14864	2,85651
11	58.33333	5,15047	2,85605
12	58.33333	5,15215	2,85556
13	58.33333	5,15371	2,85503
14	58.33333	5,15517	2,85446
15	58.33333	5,15656	2,85384

Aynı eğitim ve test seti üzerinde gizli katmandaki node sayısı 5, öğrenme oranı ve momentum değerlerinin $0,5$ seçilip iterasyon sayısı $[500, 5000]$ aralığında 500'ler arttırılarak her bir mimari için 10'ar kez eğitilip test edilen ağın ortalama doğruluk oranı, eğitim ve test seti hata değerleri Tablo 5.16'da gösterilmiştir.

Tablo 5.16 Doğruluk oranının iterasyon sayısına göre değişim değerleri

İterasyon Sayısı	Doğruluk Oranı	Eğitim Seti Hatası	Test Seti Hatası
500	60	5,13439	2,86216
1000	58,33333	4,80880	3,06445
1500	58,33333	4,73806	3,01255
2000	58,33333	4,71410	3,00430
2500	58,33333	4,70060	3,00152
3000	58,33333	4,69214	2,99939
3500	58,33333	4,68635	2,99706
4000	58,33333	4,68206	2,99460
4500	58,33333	4,67838	2,99202
5000	58,33333	4,67801	2,99190

Doğruluk oranının iterasyon sayısına göre değişim grafiği Şekil 5.8'de gösterilmiştir.



Şekil 5.8 Doğruluk oranının iterasyon sayısına göre değişimi

Görüldüğü gibi ilk yöntemde en yüksek başarım oranı gizli katmanda minimum 9 node seçilerek, ikinci yöntemde ise minimum 5 node seçilerek elde edilmiştir.

5.4 Hesaplama Denemeleri

Bölüm 5.3.1 ve 5.3.2'de elde edilen mimari parametreleri göz önüne alınarak MATLAB'ta ve geliştirilen programda gizli katmandaki node sayısı $[5, 9]$ aralığından seçilerek denemeler yapılmıştır. Deneler sonucunda en yüksek doğruluk oranı gizli katmanda 7 node seçilerek elde edilmiştir.

Denemeler veri setinin rastgele $\%80$'inin eğitim $\%20$'sinin test olarak ayrıldığı veri üzerinde yapılmıştır. Eğitim ve test setinde her bir sınıfta bulunan veri sayıları Tablo 5.17'de gösterilmiştir.

Tablo 5.17 Sınıflara göre eğitim ve test kümesinin dağılımı

Sınıf	Eğitim kümesi	Test kümesi
1	115	45
2	49	5
3	33	2
4	29	6
5	11	2

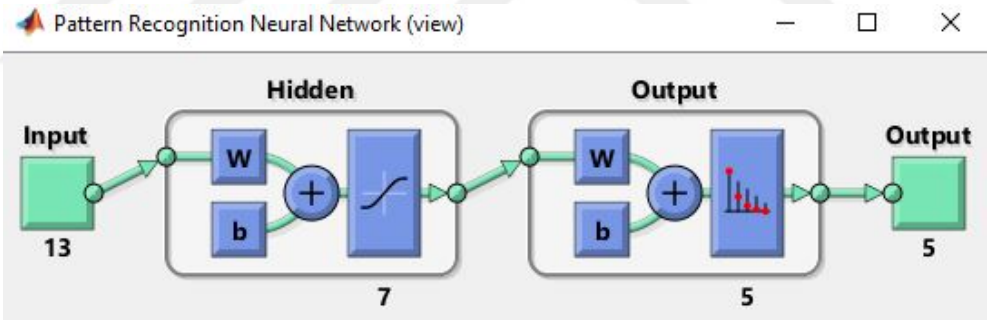
En yüksek doğruluk oranının elde edildiği mimari parametreleri Tablo 5.18'de gösterilmiştir.

Tablo 5.18 Optimal mimari için parametre aralıkları

Iterasyon Sayısı	Node Sayısı	Öğrenme Oranı	Momentum
$[100]$	7	$0,8$	$0,4$

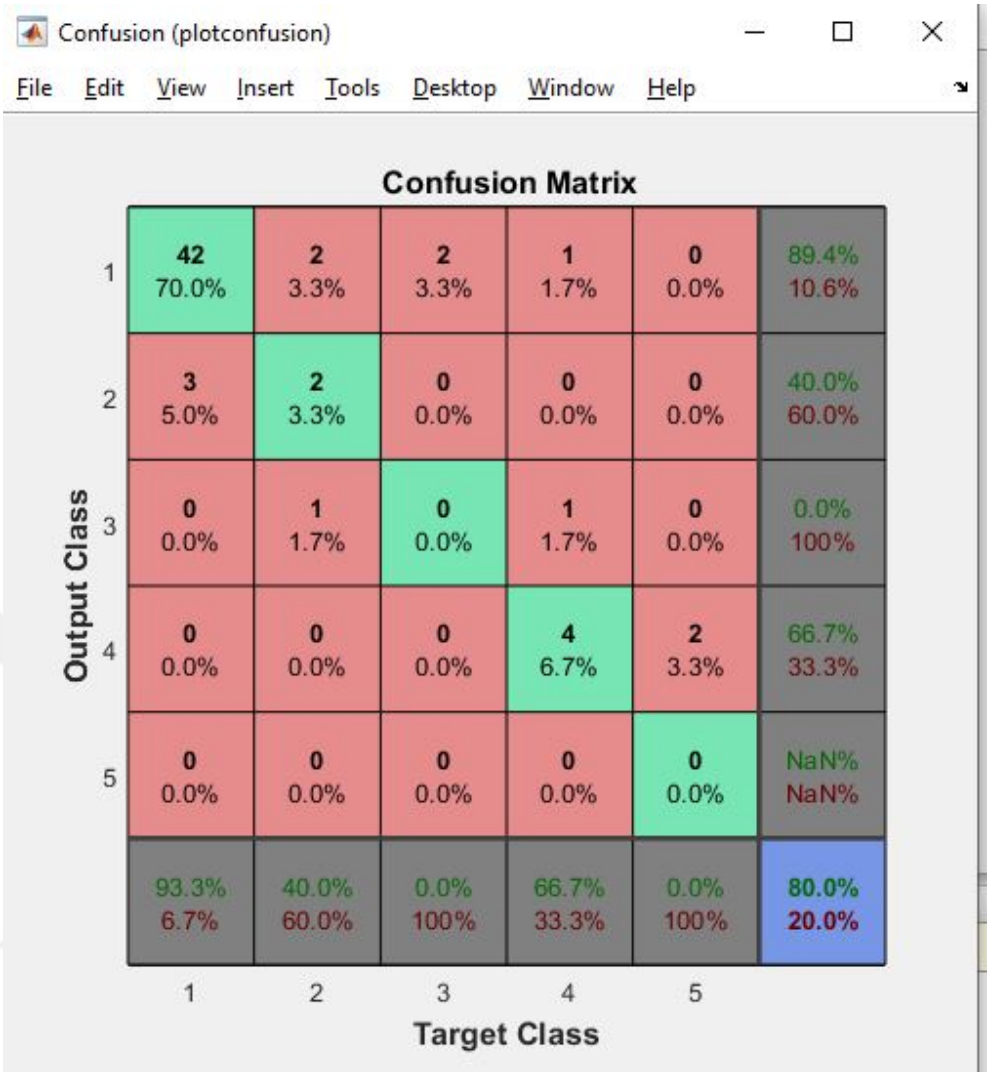
Tablo 5.18'deki mimari parametreleri ile oluşturulan ağ tablo Tablo 5.17'de eğitim ve test setininin sınıflara göre dağılımının gösterildiği Cleveland veri setinde eğitilip test edildiğinde en yüksek doğruluk oranı MATLAB'ta $\%80$ Bölüm 5.1'de geliştirilen programda ise en yüksek doğruluk oranı $\%85$ olarak bulunmuştur. İterasyon Sayısı olabildiğince küçük tutulmuş ve 100 iterasyonda bu başarım elde edilmiştir.

Her iki programda doğruluk oranının farklı çıkmasında başlangıç ağırlıklarının rastgele seçilmesinin etkisi büyüktür.
MATLAB'ta oluşturulan ağın yapısı Şekil 5.9'de gösterilmiştir.



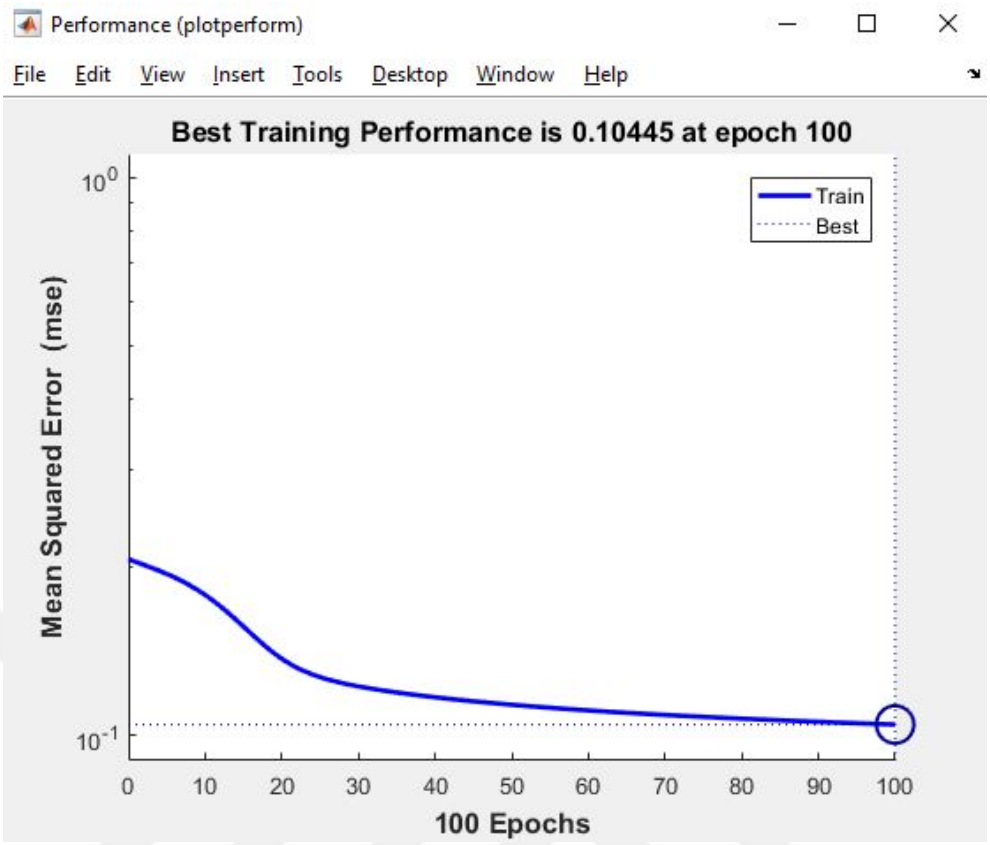
Şekil 5.9 MATLAB'ta oluşturulan ağın yapısı

Şekil 5.10'te MATLAB'ta elde edilen karmaşıklık matrisi görülmektedir.



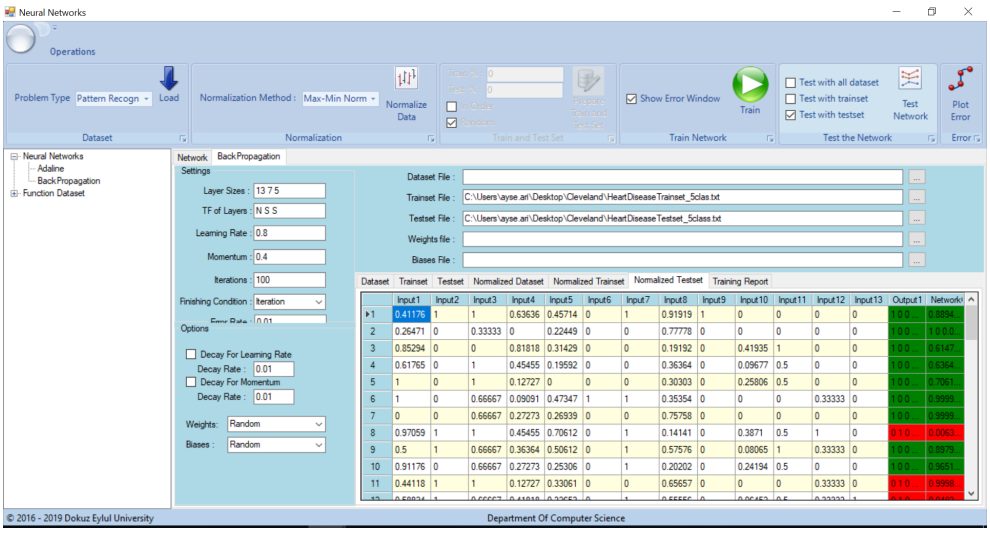
Şekil 5.10 Karmaşıklık matrisi

Şekil 5.11 ise hatanın iterasyon sayısına göre değişimini göstermektedir.



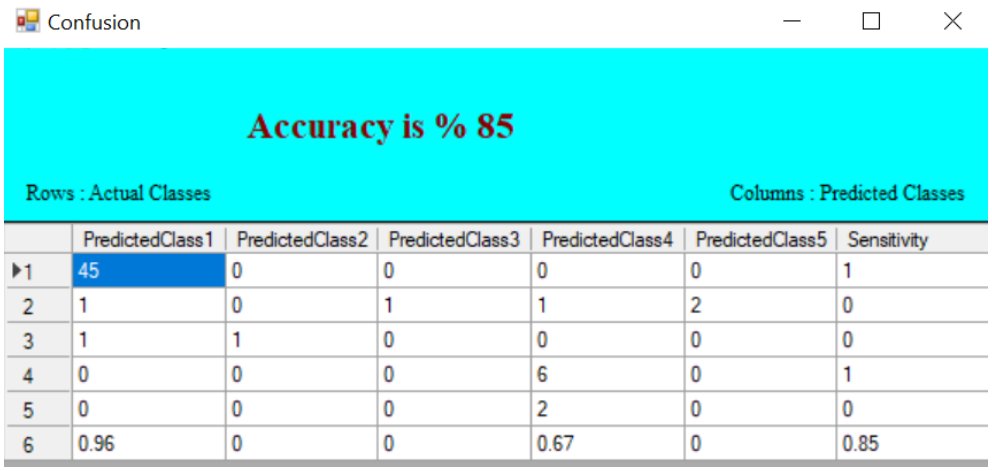
Şekil 5.11 Hatanın iterasyon sayısına göre değişimi

Bölüm 5.1'de geliştirilen yazılım ile oluşturulan ağın yapısı Şekil 5.12'de gösterilmiştir.



Şekil 5.12 Geliştirilen yazılımla oluşturulan ağın yapısı

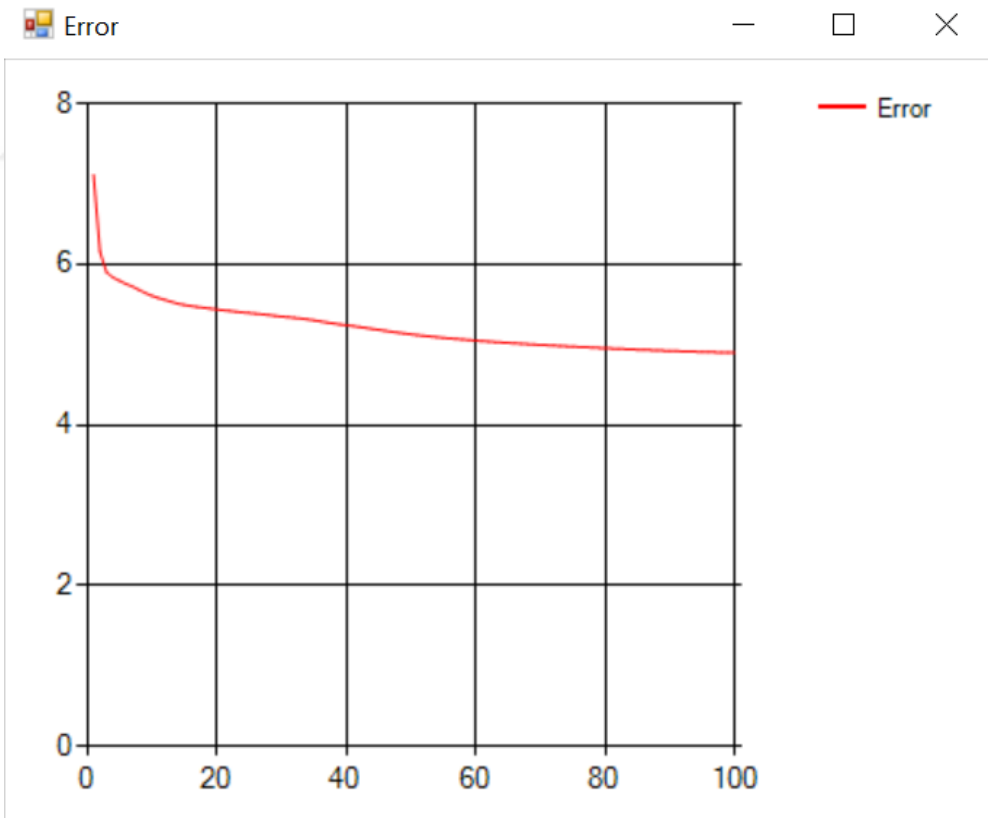
Şekil 5.13 geliştirilen yazılımla elde edilen karmaşıklık matrisi görülmektedir.



	PredictedClass1	PredictedClass2	PredictedClass3	PredictedClass4	PredictedClass5	Sensitivity
1	45	0	0	0	0	1
2	1	0	1	1	2	0
3	1	1	0	0	0	0
4	0	0	0	6	0	1
5	0	0	0	2	0	0
6	0.96	0	0	0.67	0	0.85

Şekil 5.13 Geliştirilen yazılımla elde edilen karmaşıklık matrisi

Şekil 5.14 grafik ise hatanın iterasyon sayısına göre değişimini göstermektedir.

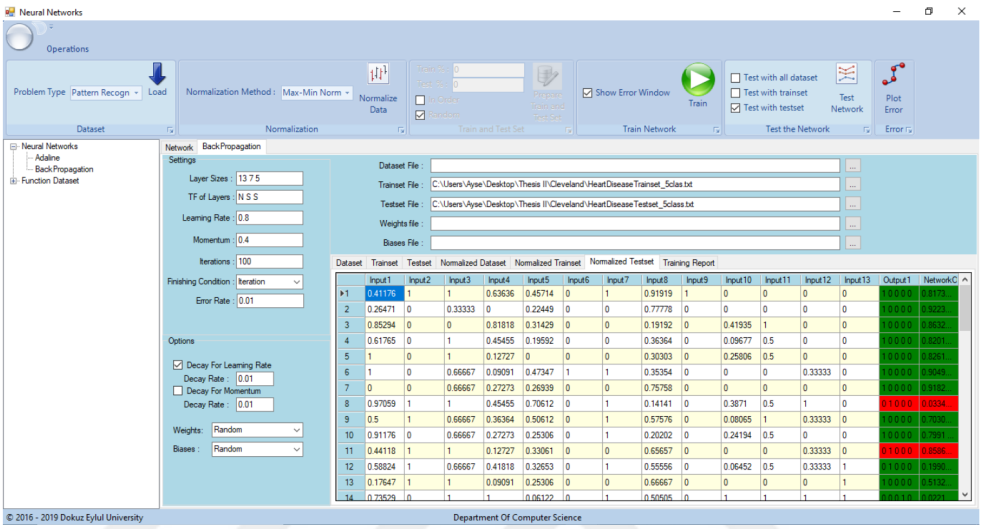


Şekil 5.14 Hatanın iterasyon sayısına göre değişimi

Geliştirilen yazılımda öğrenme oranı veya momentum değerinin ağ eğitilirken

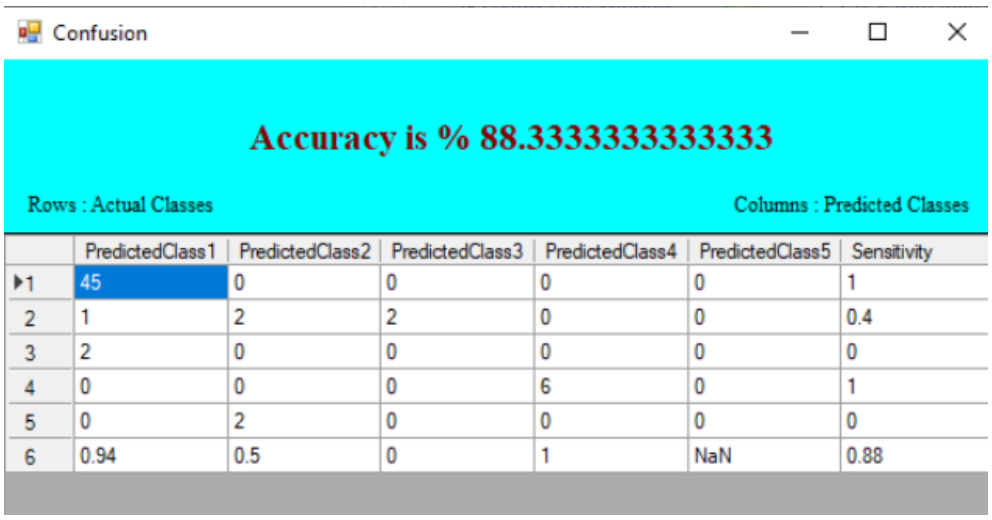
bozunum (decay) özelliği bulunmaktaydı. Programın bu özelliği için öğrenme oranı üzerine bozunum oranı $0,01$ seçilerek aynı parametrelerle tekrar eğitilip test edildiğinde doğruluk oranında artış gözlemlenmiş ve $\%88,33$ doğruluk oranına ulaşılmıştır.

Öğrenme oranına bozunum işleminin uygulandığı ağın yapısı Şekil 5.15'te verilmiştir.



Şekil 5.15 Öğrenme oranına bozunum işlemi uygulanan ağın yapısı

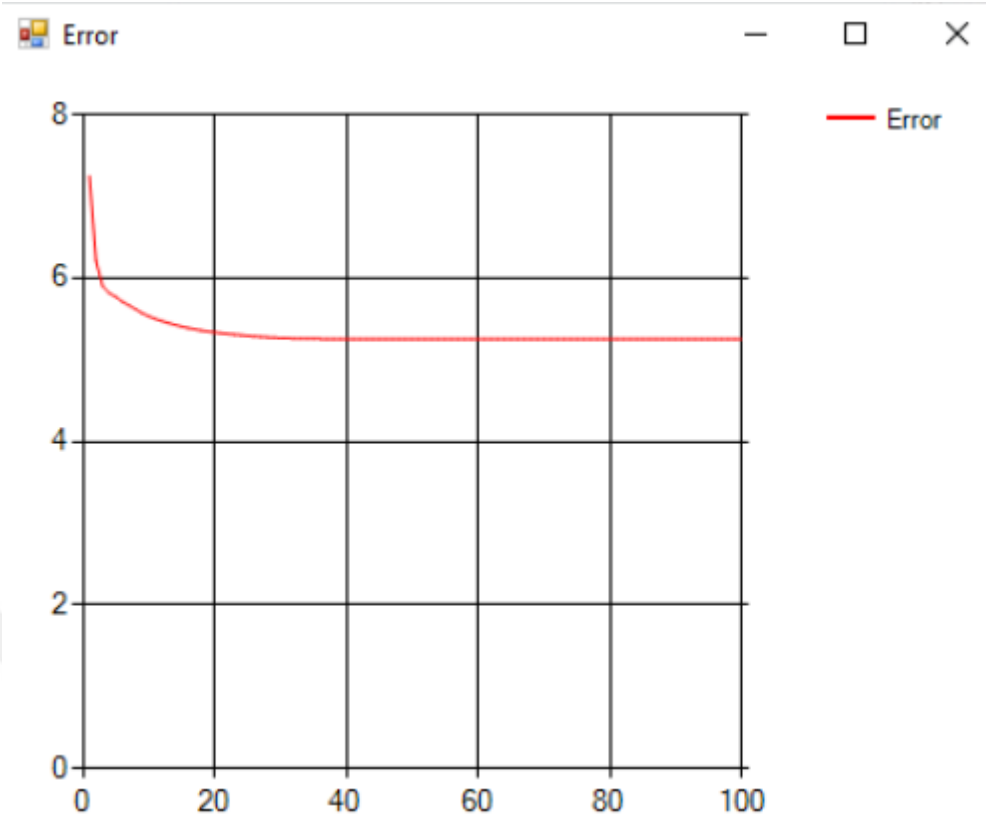
Karmaşıklık matrisi Şekil 5.16'da görülebilmektedir.



	PredictedClass1	PredictedClass2	PredictedClass3	PredictedClass4	PredictedClass5	Sensitivity
▶1	45	0	0	0	0	1
2	1	2	2	0	0	0.4
3	2	0	0	0	0	0
4	0	0	0	6	0	1
5	0	2	0	0	0	0
6	0.94	0.5	0	1	NaN	0.88

Şekil 5.16 Öğrenme oranına bozunum işlemi uygulanan ağın karmaşıklık matrisi

Oluşan hatanın grafiği ise Şekil 5.17'de gösterilmiştir.



Şekil 5.17 Hatanın iterasyon sayısına göre değişimi

BÖLÜM ALTI
SONUÇLAR

Bu çalışmada kalp hastalığının teşhisinde kullanılmak üzere bir YSA mimarisi önerilmiştir. YSA'nın eğitiminde geri yayılım algoritması kullanılmıştır. Bilindiği gibi YSA'larda problem için en iyi doğruluk oranını verecek ağ mimarisinin yapısını belirlemek için herhangi bir yöntem yoktur. En uygun mimari, problemi öğrenmesi açısından yeterince büyük, genelleme yapabilmesi için ise bir o kadar küçük seçilmelidir. Ağın mimarisi çok küçük seçildiği durumda ağ iyi bir öğrenme gerçekleştiremeyecektir, diğer yandan çok büyük seçilen bir ağ eğitim verisini aşırı öğrenmiş olacaktır ki bu durum ağın ezberlemesine sebep olacağından genelleme yeteneği zayıf kalacaktır. Tüm bu sebepler göz önüne alındığında çalışmada ilk olarak kalp hastalığının teşhisinde kullanılacak ağ için en uygun parametreler ile mimari belirlenmesi üzerine çalışılmıştır. Bu bağlamda MATLAB'ta, geri yayılım algoritması ile bir gizli katmana sahip gizli katmandaki node sayısı, iterasyon sayısı, öğrenme oranı ve momentumun ilk olarak küçük seçilip sonra sürekli olarak arttırıldığı bir program yazılmıştır. Oluşan her ağ yapısında veri seti eğitilip test edilmiştir. En yüksek doğruluk oranının elde edildiği parametreler göz önüne alınarak onlar üzerinde detaylı analiz yapılmıştır. En yüksek doğruluk oranı gizli katmanda 7 node seçilerek oluşturulan ağ topolojisinde elde edilmiştir. Veri setinin %80'i eğitim %20'si test seti olarak ayrılmıştır. MATLAB'ta eğitilip test edilen ağda 5'li sınıflandırmada en iyi %80 doğruluk oranı elde edilmiştir.

Diğer yandan tez çalışmasında kullanılmak üzere C#.NET programlama dilinde bir yazılım geliştirilmiştir. Geliştirilen yazılımın ÇKA kısmında tahmin ve sınıflandırma problemleri için YSA geri yayılım algoritması ile eğitilip test edilebilmektedir. Yazılımda ağın yapısı, gizli katman sayısı, katmanlardaki node sayısı, katmanlarda kullanılan aktivasyon fonksiyonu, verilerinin normalizasyon yöntemi, momentum katsayısı, öğrenme oranı, iterasyon sayısı, başlangıç ağırlıkları, eğitimi sonlandırma yöntemi gibi birçok özellik kullanıcılar tarafından belirlenmektedir. Geliştirilen yazılımda elde edilen en uygun ağ topolojisi ile aynı eğitim ve test seti kullanılarak

bir YSA eğitilip test edilmiş ve 5'li sınıflandırmada en iyi %85 doğruluk oranı elde edilmiştir. Yazılımda, öğrenme oranı ve/veya momentum değeri eğitim esnasında bozunum (decay) işlemine tabi tutulabilmektedir. Yazılımın bu özelliği de kullanılarak, aynı eğitim ve test kümesinde test edildiğinde 5'li sınıflandırmada %88,33 doğruluk oranına ulaşılmıştır.

Sonuç olarak bu tezde tüm fonksiyonların ve parametrelerin kullanımının kullancının kontrolünde olduğu bir yazılım üretilmiş olup yüksek bir tahmin oranına ulaşılmıştır. İleride bu yazılımın geliştirilip daha çok veriyle eğitilerek tıp dünyasının kullanımına sunulması amaçlanmaktadır.

KAYNAKLAR

Abdullah, A. S. ve Rajalaxmi, R. R. (2012). A data mining model for predicting the coronary heart disease using random forest classifier. *IJCA Proceedings on International Conference in Recent Trends in Computational Methods, Communication and Controls (ICON3C 2012)*, *ICON3C*(3), 22-25.

Abushariah, M. A., Alqudah, A. A., Adwan, O. Y. ve Yousef, R. M. (2014). Automatic heart disease diagnosis system based on artificial neural network (ann) and adaptive neuro-fuzzy inference systems (anfis) approaches. *Journal of Software Engineering and Applications*, *7*(12), 1055.

Adıyaman, F. (2007). *Talep tahmininde yapay sinir ağlarının kullanılması*. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, İstanbul.

Al-Milli, N. (2013). Backpropagation neural network for prediction of heart disease. *Journal of Theoretical and Applied Information Technology*, *56*(1), 131–135.

Angadi, S. ve Naravani, M. M. (2014). Predictiong heart attack using nbc, k-nn and id3. *International Journal of Computer Science and Engineering*, *2*(7), 6–12.

Anooj, P. (2012). Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules. *Journal of King Saud University-Computer and Information Sciences*, *24*(1), 27–40.

Arı, A. ve Berberler, M. E. (2017). Yapay sinir ağları ile tahmin ve sınıflandırma problemlerinin çözümü için arayüz tasarımı. *Acta Infologica*, *1*(2), 55–73.

Awang, M. K. ve Siraj, F. (2013). Utilization of an artificial neural network in the prediction of heart disease. *International Journal of Bio-science and Bio-technology*, *5*(4), 159–166.

Ayhan, B. (2016). *Yapay sinir ağları*. 11 Eylül 2019, https://slideplayer.biz.tr/slide/10200379/.

Bahrami, B. ve Shirvani, M. H. (2015). Prediction and diagnosis of heart disease by data mining techniques. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, *2*(2), 164–168.

Bhatia, S., Prakash, P. ve Pillai, G. (2008). Svm based decision support system for heart disease classification with integer-coded genetic algorithm to select critical features. *International Association of Engineers, Proceedings of the World Congress on Engineering and Computer Science*, 34–38.

Cilimkovic, M. (2019). *Neural networks and back propagation algorithm*. 11 Eylül 2019, https://www.semanticscholar.org/paper/Neural-Networks-and-Back-Propagation-Algorithm-Cilimkovic/df2cd0a34aeebec6b13ccdc51f845e622b781254.

Dangare, C. S. ve Apte, S. S. (2012). A data mining approach for prediction of heart disease using neural networks. *International Journal of Computer Engineering and Technology (IJCET)*, *3*(3), 30–40.

Das, R., Turkoglu, I. ve Sengur, A. (2009). Effective diagnosis of heart disease through neural networks ensembles. *Expert Systems with Applications*, *36*(4), 7675–7680.

Deekshatulu, B., Chandra, P. ve diğer. (2013). Classification of heart disease using k-nearest neighbor and genetic algorithm. *Procedia Technology*, *10*, 85–94.

Durairaj, M. ve Sivagowry, S. (2014). A pragmatic approach of preprocessing the data set for heart disease prediction. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(11), 6457–6465.

Elmsley, A. (2019). *Training to train: artificial neural networks, part II*. 10 Eylül 2019, https://medium.com/the-sound-of-ai/training-to-train-artificial-neural-networks-part-ii-b1b4efd944be.

Ghumbre, S., Patil, C. ve Ghatol, A. (2011). Heart disease diagnosis using support vector machine. *International Conference on Computer Science and Information Technology (ICCSIT)*, 84–88.

Gudadhe, M., Wankhade, K. ve Dongre, S. (2010). Decision support system for heart disease based on support vector machine and artificial neural network. *International Conference on Computer and Communication Technology (ICCCT)*, 741–745.

Hagan, M. T., Demuth, H. B. ve Beale, M. H. (1997). *Neural network design*. Boston MA, USA: PWS Publishing Co.

Hardalaç, F. ve Poyraz, M. (2002). Yapay sinir ağları kullanılarak emg sinyallerinin sınıflandırılması ve neuropathy kas hastalığının teşhisi. *Politeknik Dergisi*, *5*(1), 75–81.

Hasan, T. T., Jasim, M. H. ve Hashim, I. A. (2017). Heart disease diagnosis system based on mmulti-layer perceptron neural network and support vector machine. *International Journal of Current Engineering and Technology*, *7*(5), 1842–1853.

Haykin, S. (1999). *Neural networks: A comprehensive foundation second edition*. India: Prentice Hall.

Haykin, S. S. (2009). *Neural networks and learning machines*, volume 3. NJ, USA: Pearson Upper Saddle River.

Jabbar, M. A., Deekshatulu, B. ve Chandra, P. (2013). Classification of heart disease using artificial neural network and feature subset selection. *Global Journal of Computer Science and Technology Neural & Artificial Intelligence*, *13*(3), 5–14.

Jeans, N. (2019). *How i classified images with recurrent neural networks*. 11 Eylül 2019, https://medium.com/@nathaliejeans7/how-i-classified-images-with-recurrent-neural-networks-28eb4b57fc79.

Juneja, U. ve Dhingra, D. (2014). Multi parametric approach using fuzzification on heart disease analysis. *International Journal of Engineering Sciences & Research Technology*, *3*(5), 492–497.

Khemphila, A. ve Boonjing, V. (2011). Heart disease classification using neural network and feature selection. *2011 21st International Conference on Systems Engineering*, *3*, 406–409.

Kurenkov, A. (2015). *A 'Brief' history of neural nets and deep learning*. 11 Eylül 2019, https://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/.

Kızrak, A. (2019). *Derin öğrenme için aktivasyon fonksiyonlarının karşılaştırılması*. 5 Eylül 2019, https://medium.com/@ayyucekizrak/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). *Proceedings of Cognitiva*, 599–604.

Liu, X., Wang, X., Su, Q., Zhang, M., Zhu, Y., Wang, Q. ve Wang, Q. (2017). A hybrid classification system for heart disease diagnosis based on the rfrs method. *Computational and Mathematical Methods in Medicine*, *2017*, 1–11.

Manju, T., Priya, K. ve Chitra, R. (2013). Heart disease prediction system using weight optimized neural network. *International Journal Of Computer Science and Management Research*, *2*(5), 2391–2397.

Meng, N., Zhang, P., Li, J., He, J. ve Zhu, J. (2018). Prediction of coronary heart disease using routine blood tests. *arXiv preprint arXiv:1809.09553*, 1–10.

Minsky, M. ve Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT press.

Mohammadi, M. (2019). *Yapay sinir ağları*. 10 Eylül 2019, https://www.researchgate.net/figure/Structure-of-an-autoencoder-network_fig6_321768441.

Mythili, T., Mukherji, D., Padalia, N. ve Naidu, A. (2013). A heart disease prediction model using svm-decision trees-logistic regression (sdl). *International Journal of Computer Applications*, *68*(16), 11-15.

Negnevitsky, M. (2005). *Artificial intelligence: A guide to intelligent systems*. London: Pearson Education.

Özşen, S. ve Güneş, S. (2008). Effect of feature-type in selecting distance measure for an artificial immune system as a pattern recognizer. *Digital Signal Processing*, *18*(4), 635–645.

Özşen, S. ve Güneş, S. (2009). Attribute weighting via genetic algorithms for attribute weighted artificial immune system (awais) and its application to heart disease and liver disorders problems. *Expert Systems with Applications*, *36*(1), 386–392.

Parker, D. B. (1985). *Learning-logic: Casting the Cortex of the Human Brain in Silicon*. Technical report, Massachusetts Institute of Technology, Center for Computational Research in Economics and Management Science.

Patel, S. B., Yadav, P. K. ve Shukla, D. (2013). Predict the diagnosis of heart disease patients using classification mining techniques. *IOSR Journal of Agriculture and Veterinary Science (IOSR-JAVS)*, *4*(2), 61–64.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.

Rosenblatt, F. (1961). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. New York: Cornell Aeronautical Laboratory, Inc.

Rumelhart, D. E. ve Mcclelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations (Parallel distributed processing)*. Cambridge, Massachusetts:MIT Press.

Şahan, S., Polat, K., Kodaz, H. ve Güneş, S. (2005). The medical applications of attribute weighted artificial immune system (awais): Diagnosis of heart and diabetes diseases. *International Conference on Artificial Immune Systems*, *3627*, 456–468.

Saxena, K., Sharma, R. ve diğer. (2016). Efficient heart disease prediction system. *Procedia Computer Science*, *85*, 962–969.

Srinivas, K., Rani, B. K. ve Govrdhan, A. (2010). Applications of data mining techniques in healthcare and prediction of heart attacks. *International Journal on Computer Science and Engineering (IJCSE)*, *2*(02), 250–255.

Sundar, N. A., Latha, P. P. ve Chandra, M. R. (2012). Performance analysis of classification data mining techniques over heart disease database. *IJESAT International Journal of Engineering Science & Advanced technology*, *2*(3), 470–478.

UCI Machine Learning Repository, U. (1988). *Heart disease data set*. 9 Eylül 2019, https://archive.ics.uci.edu/ml/datasets/heart+Disease.

Uyar, K. ve İlhan, A. (2017). Diagnosis of heart disease using genetic algorithm based trained recurrent fuzzy neural networks. *Procedia Computer Science*, *120*, 588–593.

Waghulde, N. P. ve Patil, N. P. (2014). Genetic neural approach for heart disease prediction. *International Journal of Advanced Computer Research*, *4*(3), 778.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Doktora Tezi, Harvard Üniversitesi, Cambridge, Massachusetts.

Widrow, B. ve Hoff, M. E. (1960). *Adaptive switching circuits*. Technical report, Stanford Univ Ca Stanford Electronics Labs.

Wikibooks (2019). *Artificial neural networks/feed-forward networks*. 9 Eylül 2019, https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Feed-Forward_Networks.

Williams, D. ve Hinton, G. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–538.

World Health Organization, W. H. O. (2018). *The top 10 causes of death*. 16 Aralık 2018, http://www.who.int/mediacentre/factsheets/fs310/en/.

World Health Organization, W. H. O. (b.t). *Cardiovascular disease*. 5 Eylül 2019, https://www.who.int/cardiovascular_diseases/world-heart-day/en/.

World Heart Federation, W. H. F. (2017). *Different heart diseases*. 5 Eylül 2019, https://www.world-heart-federation.org/resources/different-heart-diseases/.

Ziasabounchi, N. ve Askerzade, I. (2014). Anfis based classification model for heart disease prediction. *International Journal of Electrical & Computer Sciences IJECS-IJENS*, *14*(02), 7–12.

Zriqat, I. A., Altamimi, A. M. ve Azzeh, M. (2016). A comparative study for predicting heart diseases using data mining classification methods. *International Journal of Computer Science and Information Security*, *14*(12), 868–879.

EKLER

EK 1: Başlangıç ağırlıkları rastgele seçilerek mimari belirleme MATLAB kodu

```
clear all;
clc;

inputs = [train inputs in here] ;
targets = [train targets in here];
TestInputs = [test inputs in here];
TestTargets = [test targets in here];

fid=fopen('sonuc.txt','a+');
fprintf(fid,'HiddenLayerNode Iteration LearningRate Momentum
AvarageTrainError MaxTrainError MinTrainError AvarageTrainTime
 MaxTrainTime MinTrainTime AvarageTestError MaxTestError
 MinTestError AvarageTestTime MaxTestTime MinTestTime
 AvarageAccuracy MaxAccuracy MinAccuracy\n');
fclose(fid);

for i=1:15
hiddenLayerSize = [i];
net = patternnet(hiddenLayerSize);
net = configure(net, inputs,targets);

net.divideFcn = 'dividetrain';
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 100/100;



net.trainFcn = 'traingdm';

net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';

net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate',
```

```
'ploterrhist', ... ,
'plotregression', 'plotfit'};

for l=500:500:5000
iteration=l;
net.trainParam.epochs = iteration;
for j=1:0.5:11
learningrate=(j-1)/10;
net.trainParam.lr=learningrate;
for k=1:0.5:11
momentum=(k-1)/10;
net.trainParam.mc=momentum;

net.trainParam.showWindow = false;
net.trainParam.showCommandLine = false;
x=[];
for p=1:10

tic
[net,tr] = train(net,inputs,targets);
trainTime=toc;

Trainoutputs = net(inputs);
TrainErrors = gsubtract(targets,Trainoutputs);

sum=0.0;
for m=1:(size(TrainErrors,2))
for n=1:size(TrainErrors,1)
sum =sum + abs(TrainErrors(n,m)^(2));
end
end
trainError=1/2*sqrt(sum);

tic
outputs=sim(net,TestInputs);
testTime=toc;
errors = gsubtract(TestTargets,outputs);
```

```
performance = perform(net,TestTargets,outputs);

sum=0.0;
for m=1:(size(errors,2))
for n=1:size(errors,1)
sum =sum + abs(errors(n,m)^(2));
end
end
testError=1/2*sqrt(sum);

[c,cm,ind,per] = confusion(TestTargets,outputs);
accuracy=(1-c)*100;

x(p,1)=i;
x(p,2)=iteration;
x(p,3)=learningrate;
x(p,4)=momentum;
x(p,5)=trainError;
x(p,6)=trainTime;
x(p,7)=testError;
x(p,8)=testTime;
x(p,9)=accuracy;

end

y=[x(1,1) x(1,2) x(1,3) x(1,4) mean(x(:,5)) max(x(:,5))
min(x(:,5)) mean(x(:,6)) max(x(:,6)) min(x(:,6)) mean(x(:,7))
max(x(:,7)) min(x(:,7)) mean(x(:,8)) max(x(:,8)) min(x(:,8))
mean(x(:,9)) max(x(:,9)) min(x(:,9)) ]

fid=fopen('sonuc.txt','a+');
fprintf(fid,'%8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f
 %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f
 %8.5f\n',y);
fclose(fid);
end
end
```

```
end
end
```

EK 2: Başlangıç ağırlıkları sabit seçilerek mimari belirleme MATLAB kodu

```
clear all;
clc;

inputs = [train inputs in here] ;
targets = [train targets in here];
TestInputs = [test inputs in here];
TestTargets = [test targets in here];

fid=fopen('sonuc.txt','a+');
fprintf(fid,'HiddenLayerNode Iteration LearningRate Momentum
 TrainError TrainTime TestError TestTime Accuracy\n');
fclose(fid);

for i=1:15
hiddenLayerSize = [i];
net = patternnet(hiddenLayerSize);
w=[];
sizee=13*i+i*5+5+i;
for h=1:1:sizee
w(h,1)=0.5;
end
net = configure(net, inputs,targets);
net = setwb(net, w);

net.divideFcn = 'dividetrain';
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 100/100;

net.trainFcn = 'traingdm';

net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';

net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate',
```

```
'ploterrhist', ... ,'plotregression', 'plotfit'};

for l=500:500:5000
iteration=l;
net.trainParam.epochs = iteration;
for j=1:0.5:11
learningrate=(j-1)/10;
net.trainParam.lr=learningrate;
for k=1:0.5:11
momentum=(k-1)/10;
net.trainParam.mc=momentum;

net.trainParam.showWindow = false;
net.trainParam.showCommandLine = false;
x=[];


tic
[net,tr] = train(net,inputs,targets);
trainTime=toc;

Trainoutputs = net(inputs);
TrainErrors = gsubtract(targets,Trainoutputs);

sum=0.0;
for m=1:(size(TrainErrors,2))
for n=1:size(TrainErrors,1)
sum =sum + abs(TrainErrors(n,m)^(2));
end
end
trainError=1/2*sqrt(sum);


tic
outputs=sim(net,TestInputs);
testTime=toc;
errors = gsubtract(TestTargets,outputs);
```

```
performance = perform(net,TestTargets,outputs);

sum=0.0;
for m=1:(size(errors,2))
for n=1:size(errors,1)
sum =sum + abs(errors(n,m)^(2));
end
end
testError=1/2*sqrt(sum);

[c,cm,ind,per] = confusion(TestTargets,outputs);
accuracy=(1-c)*100;

x(1,1)=i;
x(1,2)=iteration;
x(1,3)=learningrate;
x(1,4)=momentum;
x(1,5)=trainError;
x(1,6)=trainTime;
x(1,7)=testError;
x(1,8)=testTime;
x(1,9)=accuracy;


y=[x(1,1) x(1,2) x(1,3) x(1,4) x(1,5) x(1,6) x(1,7) x(1,8)
x(1,9) ]

fid=fopen('sonuc.txt','a+');
fprintf(fid,'%8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f %8.5f
 %8.5f\n',y);
fclose(fid);
end
end
end
end
```