KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

Yapay Zeka İle Yazılım Efor Tahmini

Berna ŞEREF

OCAK 2015

Bilgisayar Mühendisliği Anabilim Dalında Berna ŞEREF tarafından hazırlanan YAPAY ZEKA İLE YAZILIM EFOR TAHMİNİ adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof. Dr. Hasan ERBAY
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Doç. Dr. Necaattin BARIŞÇI
Danışman

Jüri Üyeleri

Başkan : Prof. Dr. Hasan ERBAY _____

Üye (Danışman) : Doç. Dr. Necaattin BARIŞÇI _____

Üye : Yrd. Doç. Dr. Taner TOPAL _____

……/…../…….

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Doç. Dr. Erdem Kamil YILDIRIM
Fen Bilimleri Enstitüsü Müdürü

ÖZET

YAPAY ZEKA İLE YAZILIM EFOR TAHMİNİ

ŞEREF, Berna
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi
Danışman: Doç. Dr. Necaattin BARIŞÇI
Ocak 2015, 58 sayfa

Yazılım efor tahmini şirketler ve müşteriler için büyük bir önem arz etmektedir. Eforun alçak ya da yüksek tahmin edilmesi hem şirketleri hem de müşterileri olumsuz yönde etkilemektedir.

Bu çalışmada yazılım efor tahmini Çok Katmalı Algılayıcı Sinir Ağları ve Genetik Çok Katmanlı Algılayıcı Sinir Ağları kullanılarak tahmin edilmiştir. Veri seti olarak Desharnais veri seti kullanılmıştır. Her iki model için de 71 projeden oluşan aynı eğitim seti seçilmiştir.

Çok Katmalı Algılayıcı Sinir Ağları modeli için, 71 projenin 8'i onaylama ve test için kullanılmıştır. Geriye kalan projeler ise sistemi eğitmek için kullanılmıştır. Genetik Çok Katmanlı Algılayıcı Sinir Ağları modeli için, 71 projenin 10'u çapraz doğrulama, 3'ü ise sistemi test etmek için kullanılmıştır.

Eğitim setinde bulunmayan 10 farklı projenin efor değerleri tahmin edilmiştir. Daha sonra, Çok Katmalı Algılayıcı Sinir Ağları ve Genetik Çok Katmanlı Algılayıcı Sinir Ağları modellerinin tahmin performansları Ortalama Bağıl Hata ve Pred(25) performans değerlendirme kriterleri kullanılarak karşılaştırılmıştır. Sonuç olarak Genetik Çok Katmanlı Algılayıcı Sinir Ağları 'nın tahmin performansını geliştirdiği gözlenmiştir

Anahtar kelimeler: Yazılım Efor Tahmini, Ortalama Bağıl Hata, Pred(25), Çok Katmanlı Algılayıcı Sinir Ağları, Genetik Genetik Çok Katmanlı Algılayıcı Sinir Ağları

ABSTRACT

SOFTWARE EFFORT ESTIMATIN BY USING ARTIFICIAL INTELLIGENCE

ŞEREF, Berna

Kırıkkale University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering, M. Sc. Thesis

Supervisor: Assoc. Prof. Dr. Necaattin BARIŞÇI

January 2015, 58 pages

Software effort estimation has a big importance for companies and customers. Overestimating or underestimating of effort affects both of them in a negative way.

In this study, software effort estimation was predicted by using Multi Layer Perceptron Neural Networks and Genetic Multi Layer Perceptron Neural Networks. As a dataset, Desharnais dataset was used. For both of these neural networks, the same 71 projects were chosen with the aim of training the system.

For Multi Layer Perceptron Neural Networks model, 8 of 71 projects were used for validation and test the system. The other projects were used in order to train the system. For Genetic Multi Layer Perceptron Neural Networks, 10 of 71 projects are used for cross validation and 3 of the 71 projects are used for testing the system. The rest of the projects in the training set was used to train the system.

Efforts of 10 different projects which are not in the trainig set were predicted. Then, prediction performance of Multi Layer Perceptron Neural Networks and Genetic Multi Layer Perceptron Neural Networks models were compared by using Mean Magnitude of Relative Error and Pred(25) performance evaluation criterions. As a result, it was observed that prediction performance was improved when Genetic Multi Layer Perceptron Neural Networks were used.

Key Words: Software Effort Estimation, Mean Magnitude of Relative Error, Pred(25), Multi Layer Perceptron Neural Networks, Genetic Multi Layer Perceptron Neural Network

TEŞEKKÜR

Tezimin hazırlanması esnasında her türlü yardım ve desteğini esirgemeyen danışman hocam Sayın Doç. Dr. Necaattin BARIŞÇI'ya teşekkürlerimi sunarım.

Ayrıca, hoşgörü ve destekleriyle her zaman yanımda olan aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

ÇİZELGELER DİZİNİ

SİMGELER DİZİNİ

w	Ağırlık
O	Çıktı
φ	Eşik değeri
μ	Öğrenme katsayısı

1. GİRİŞ

Bilgi teknolojilerindeki gelişmelerle birlikte efor tahmini hem yazılım geliştiren kurumlar, hem de satın alan kişi ya da kuruluşlar için büyük bir önem kazanmıştır. Yanlış efor tahmini kaynakların fazla kullanılmasına neden olarak maliyeti artırabilmekte, proje teslim tarihlerini değiştirebilmekte ve bu durum hem yazılım geliştiren kurumlar, hem de satın alan kişi ya da kurumlar açısından büyük bir problem teşkil etmektedir.

Yazılım efor tahmini, bir projeyi geliştirebilmek için gerekli olan süredir. Süre ise saat olarak alınmaktadır. Bu sürenin doğru tahmin edilmesiyle daha doğru bir planlama sürecine girilmekte, doğru bir iş ayrışımı ve paylaşımı yapılmakta, kaynakların yönetimi daha verimli bir şekilde sağlanmakta, proje teslim tarihlerindeki değişiklikler ve tutarsızlıklar en aza indirilmektedir.

Yapay Sinir Ağları, insan beyninin yani insan biyolojik sinir sisteminin çalışma tekniği baz alınarak oluşturulan bir yöntemdir. İnsanlar, doğumlarından itibaren tüm yaşamlarını bir öğrenme eylemi içerisinde geçirirler. Düşünürler, gözlem yaparlar ve tecrübe ederler. Öğrenme insan sinir sisteminde sinaptik noktalarda gerçekleştiğinden, yeni öğrenmelerle ve yeni tecrübelerle yeni sinaptik bağlantılar oluşturulur ya da eski sinaptik bağlantılar güncellenir, daha güçlü hale getirilir. Böylece insanlar yeni bilgiler öğrenmiş olurlar ve eski öğrendikleri ile yeni öğrendiklerini mukayese edip, yeni düşünceler oluşturacak duruma gelirler. Yapay Sinir Ağları'nda da öğrenme sinaptik noktalarda ağırlıkların güncellenmesi şeklinde olur. Eğitim setindeki girdi ve çıktılar ağa gösterilir, eğitim algoritmasıyla ağırlıklar güncellenir ve ağırlıkları güncelleme işlemi seçilen durdurma kriteri sağlanana kadar devam eder. İlk geliştirilen Yapay Sinir Ağları bazı problemleri çözemezken, daha sonra geliştirilen Çok Katmanlı Algılayıcılar birçok problemi çözebilmektedir.

Genetik Algoritmalar, evrime dayalı algoritmalardan biridir. Buradaki amaç, en kaliteli, en uygun bireyi bulabilmektir. Bu birey de çözümü temsil etmektedir. Algoritmanın uygulanabilmesi için başlangıç popülasyonu gerekmektedir. Bu

popülasyon kullanılarak tabi seçme ve tekrar üreme işlemleri ile yeni popülasyonlar elde edilmektedir. Popülasyondaki her bir birey, yani her bir çözüm için uygunluk fonksiyonu aracılığıyla uygunluk değeri hesaplanmaktadır. Uygunluk değeri en yüksek olan bireylerin popülasyondaki diğer bireyler ile çoğalması sağlanmakta, yeni bireyler üretilmektedir. Uygunluk değeri düşük olan bireylerin çoğalma işlemi için seçilme olasılığı daha düşük olduğundan, bu bireyler bir süre sonra popülasyondan atılmaktadır. Sonuç olarak iyi olan özellikler nesiller boyunca korunmakta, uygun bireyler arasında yapılan genetik işlemlerle daha iyi özelliklere sahip bireyler elde edilmektedir.

Efor tahmini hem yazılım geliştiren kurumlar, hem de bu yazılımları satın alan kişi ya da kuruluşlar için büyük bir önem arz etmektedir. Daha doğru bir efor tahmini için birçok yöntem denenmiş, birçok çalışma yapılmıştır. Bu çalışmaların bazılarında yeni teknikler oluşturulmuş, bazılarında var olan teknikler hibrit bir biçimde kullanılarak performans artırılmaya çalışılmış, bazılarında ise performansı etkileyen faktörler üzerinde durularak yazılım efor tahmini doğruluk oranının geliştirilmesi amaçlanmıştır. Yapılan çalışmalarda hem eğitim hem de test setleri kullanılmış, tahminlerde bulunulmuş, performans değerlendirme kriterlerinden faydalanılarak gerçekleştirilen çalışmayla efor tahminlerinde gelişme sağlanıp sağlanmadığı gözlenmiştir.

2013 yılında Diaz ve arkadaşları [1] tarafından yapılan bir çalışmada, Takagi-Sugeno ve Mamdami bulanık modelleri kullanılarak efor tahmini yapılmış ve bu iki modelin performansları karşılaştırılmıştır. Bunun için 37 yazılımcı tarafından Kişisel Yazılım Süreci uygulamaları göz önünde bulundurularak 125 adet proje geliştirilmiştir. Girdi olarak "Yeni ve Değiştirilmiş Kaynak Kod" satırları, çıktı olarak da "Efor" değerleri kullanılmıştır. İki modelin performanslarının karşılaştırılması Ortalama Bağıl Hata ve Pred(25) değerleri kullanılarak yapılmıştır. Sonuç olarak, efor değeri 100 ve üzeri olan projelerde Takagi-Sugeno modelinin daha doğru tahminde bulunduğu gözlenmiştir.

2006 yılında Martin ve arkadaşları [2] tarafından yapılan bir çalışmada, Bulanık Mantık modeli ve Çoklu Doğrusal Regresyon modeli kullanılarak efor tahmini

yapılmış, bu modellerin performansları karşılaştırılmıştır. Veri seti olarak yazılımcılar tarafından Kişisel Yazılım Süreci uygulamaları göz önünde bulundurularak geliştirilen 94 adet program, performans değerlendirme kriteri olarak da Ortalama Bağıl Hata, Pred(25), Medyan Bağıl Hata ve Varyans Analizi değerleri kullanılmıştır. Sonuç olarak, küçük projeler için Bulanık Mantık modelinin daha iyi bir performans sergilediği gözlenmiştir.

2009 yılında Martin [3] tarafından yapılan bir çalışmada, Bulanık model ve İstatistiksel Regresyon modeli kullanılarak efor tahmini yapılmış, iki modelin tahmin doğruluk oranları karşılaştırılmıştır. Kişisel Yazılım Süreci uygulamaları göz önünde bulundurularak geliştirilen küçük programlar veri setini oluşturmuş, "Yeni ve Değiştirilmiş Kod" ile "Yeniden Kullanılan Kod" girdi olarak kullanılmıştır. Ortalama Hata Geometrili ve Varyans Analizi değerleri performans değerlendirme kriteri olarak seçilmiştir. Sonuç olarak, küçük programlar için "Yeni ve Değiştirilmiş Kod" ile "Yeniden Kullanılan Kod" girdi olarak kullanıldığında Bulanık modelin daha doğru tahminlerde bulunduğu gözlenmiştir.

2011 yılında Martin ve arkadaşları [4] tarafından yapılan bir çalışmada, efor tahmini için Genel Regresyon Sinir Ağı modeli ve İstatistiksel Regresyon modeli kullanılmıştır. Herbir model için Yazılım Kıyaslama Standartları Grubu yazılım projeleri deposundan veri seti seçilmiştir. Performans değerlendirme kriteri olarak Varyans Analizi değerleri kullanılmıştır. Sonuç olarak, endüstriyel alanlarda geliştirilen projeler için yapılacak olan efor tahminlerinde alternatif olarak Genel Regresyon Sinir Ağı 'nın kullanılabileceği gözlenmiştir.

2010 yılında Oliveira ve arkadaşları [5] tarafından yapılan bir çalışmada, efor tahmini doğruluk oranını artırmak amacıyla, uygun girdi öğelerinin belirlenmesinde ve regresyon için gerekli olan makine öğrenmesi teknikleri parametrelerinin belirlenmesinde Genetik Algoritma'dan faydalanılmıştır. Veri setleri olarak Desharnais, NASA, COCOMO, Albrecht, Kemerer, Koten ve Gray veri setleri kullanılmıştır. Sonuçları karşılaştırmak ve değerlendirmek için Ortalama Bağıl Hata ve Pred(25) değerleri hesaplanmıştır. Destek Vektör Regresyonu, Çok Katmanlı Algılayıcı Yapay Sinir Ağları ve Karar Ağaçları makine öğrenmesi tekniklerine

Genetik Algoritma uygulanmış, uygulama sonunda yazılım efor tahmini doğruluk oranında gelişme sağlanmıştır.

2009 yılında Muzaffar ve Ahmed [6] tarafından yapılan bir çalışmada, Bulanık Mantık kullanılarak yapılan yazılım efor tahminlerinde tahminlerin doğruluk derecesini etkileyen faktörler üzerinde durulmuş, bu faktörlerin tahminlere olan etkileri araştırılmıştır. Örneğin; Bulanık Mantık'ta kullanılan durulaştırma metodunun, üyelik fonksiyonlarının şeklinin ve yayılır hatanın etkileri araştırılmıştır. Değiştirilmiş yükseklik durulaştırılması, üçgen üyelik fonksiyonu ve bağıl hata kullanıldığında diğer bir yöntem olan yükseklik durulaştırılması, Gauss üyelik fonksiyonu ve normalleştirilmiş hata kullanılmasına kıyasla daha iyi sonuçlar elde edilmiştir. Tahmin performansı hem tip-1 hem de tip-2 Bulanık Mantık kullanılarak yapılmış, tip-2 Bulanık Mantık kullanılarak geliştirilen sistemin tip-1 Bulanık Mantık kullanılarak geliştirilen sisteme oranla belirsizlikleri daha iyi işleyebildiği gözlenmiştir.

2011 yılında Malhotra ve Jain [7] tarafından yapılan bir çalışmada, efor tahmini için Doğrusal Regresyon, Yapay Sinir Ağları, Karar Ağacı, Destek Vektör Makine ve Torbalama yöntemleri kullanılmış, Ortalama Bağıl Hata ve Pred(25) kriterlerine göre en iyi performansı Karar Ağacı'nın gösterdiği görülmüştür. Veri seti olarak "Promise" veri havuzu kullanılmış, 19 özellikten oluşan veri seti Korelasyon Tabanlı Özellik Seçim tekniği ile 10 özelliğe indirilmiştir.

2002 yılında Heiat [8] tarafından yapılan bir çalışmada, Çok Katmanlı Algılayıcı ve Radyan Tabanlı Fonksiyon Ağı Sinir Ağları'nın tahmin performansları, İstatistiksel Regresyon tahmin performansıyla karşılaştırılmış, karşılaştırma kriteri olarak Ortalama Bağıl Hata kullanılmıştır. Veri seti olarak 24 adet projeden oluşan IBM DP Servis Organizasyonu, 15 adet projeden oluşan Kemerer ve 28 adet projeden oluşan Hallmark olmak üzere üç farklı veri setinden faydalanılmış, iki ayrı deney yapılmıştır. İlk deneyde üçüncü nesil programlama dillerini içeren IBM ve Kemerer veri setlerindeki projeler kullanmıştır. İkinci deney veri seti ise hem üçüncü hem de dördüncü nesil programlama dillerini içeren Kemerer, IBM ve Hallmark veri setlerindeki projelerden oluşmuştur. Böylece ilk deney 32 adet eğitim setinden ve 7

adet test setinden oluşurken, ikinci deney 60 adet eğitim setinden ve 7 adet test setinden oluşmuştur. Deney sonunda, üçüncü nesil ve dördüncü nesil dilleri içeren veri setleri kullanıldığında Sinir Ağları'nın İstatistiksel Regresyon 'a göre daha iyi performans sergilediği ve bu konuda daha büyük veri setleri kullanılarak daha fazla çalışma yapılması gerektiği sonucuna varılmıştır.

2006 yılında Oliveira [9] tarafından yapılan bir çalışmada, Destek Vektör Regresyonu, Radyal Tabanlı Fonksiyon Yapay Sinir Ağı ve Lineer Regresyon kullanılarak tahminlerde bulunulmuş, performans değerlendirme kriteri olarak Ortalama Bağıl Hata değeri kullanılmıştır. Çalışmanın sonunda Destek Vektör Regresyonu'nun performansının Radyal Tabanlı Fonksiyon Yapay Sinir Ağı ve Lineer Regresyon performansına göre daha iyi sonuç verdiği görülmüştür.

2008 yılında Vinay ve arkadaşları [10] tarafından gerçekleştirilen bir çalışmada, efor tahminleri için Dalgacık Sinir Ağı tekniği önerilmiş, Dalgacık Sinir Ağı tahmin performansı ile Çok Katmanlı Algılayıcı, Radyal Tabanlı Fonksiyon Ağı, Çoklu Doğrusal Regresyon, Dinamik Gelişen Sinir-Bulanık Çıkarım Sistemi ve Destek Vektör Makinesi tahmin performansları Ortalama Bağıl Hata performans değerlendirme kriteri ile karşılaştırılmıştır. Veri seti olarak Kanada Finansı'na ait olan 24 adet projeden ve Uluslararası İş Makineleri veri işleme servisine ait olan 37 adet projeden faydalanılmıştır. Çalışmanın sonunda Dalgacık Sinir Ağı tahmin performansının diğerlerine göre daha iyi olduğu gözlenmiştir.

2008 yılında Park ve Baek [11] tarafından yapılan diğer bir çalışmada ise, Sinir Ağı'nın bağımsız değişkenleri değiştirilerek 3 adet deney yapılmıştır. İlk deneyde Sinir Ağı değişken olarak sadece fonksiyon noktalarını, ikinci deneyde fonksiyon noktaları değişkenlerini içermeyen 6 adet değişkeni, üçüncü deneyde ise hem fonksiyon noktalarını hem de 6 adet değişkeni değişken olarak almıştır. Veri seti olarak 1999 ve 2003 yılları arasında yapılan ve birçok sanayi kolunu da kapsayan 148 projeden faydalanılmış, performans değerlendirme kriteri olarak Ortalama Bağıl Hata değeri kullanılmıştır. Deney sonunda ise fonksiyon noktaları ve 6 adet değişkeni kullanan Sinir Ağı modelinin, diğer iki modele göre daha iyi performans gösterdiği görülmüştür.

2008 yılında De Barcelos [12] tarafından yapılan diğer bir çalışmada, efor tahmini için İleri Beslemeli Çok Katmanlı Algılayıcı Sinir Ağı ve İstatistiksel Regresyon kullanılmıştır ve performans karşılaştırması yapılmıştır. Veri seti olarak 1981 yılında yayınlanan ve 63 adet projeden oluşan COCOMO veri setinden faydalanılmıştır. 63 adet projeden 11 tanesi sistemi test etmek için kullanılırken, geriye kalan 52 tanesi eğitim için kullanılmıştır. Performans değerlendirme kriteri olarak Ortalama Bağıl Hata değeri seçilmiştir. Çalışma sonunda İleri Beslemeli Çok Katmanlı Algılayıcı Sinir Ağı ve İstatistiksel Regresyon tekniklerinin tahmin performanslarının birbirine çok yakın olup, rekabet halinde olduğu gözlenmiştir. Çalışma sonunda, Sinir Ağları ve Çoklu Regresyon tekniklerinin birleştirilmesi için yeni deneylerin yapılması önerilmiştir. Bu deneylerdeki amaç, tahmin modellerinin diğer veri setleri üzerinde test edilebilmesi ve ayarlanabilmesidir.

2011 yılında El-Sebakhy [13] tarafından yapılan bir çalışmada ise efor tahmini için Fonksiyonel Ağlar'ın kullanılması önerilmiştir. Performans değerlendirme kriteri olarak Ortalama Bağıl Hata değeri seçilmiş, Fonksiyonel Ağlar'ın tahmin performansı, standart Çok Katmanlı Algılayıcı ve Doğrusal Olmayan İstatistiksel Regresyon tahmin performansıyla karşılaştırılmıştır. Veri seti olarak 24 adet projeden oluşan IBM DP Servis Organizasyonu, 15 adet projeden oluşan Kemerer ve 28 adet projeden oluşan Hallmark olmak üzere üç farklı veri seti kullanılmıştır. Çalışma sonunda Fonksiyonel Ağlar'ın tahmin performansının standart Sinir Ağları ve Çoklu Regresyon'dan daha iyi olduğu gözlenmiştir.

Efor tahmini hatalarını en aza indirmek ve tahmin performanslarını artırmak amacıyla birçok çalışma yapılmış ve birçok yöntem denenmiştir. Bu yöntemlerin başında Yapay Zeka Teknikleri gelmektedir. Bu çalışmada ise efor tahmini Çok Katmanlı Algılayıcı ve Çok Katmalı Algılayıcı ile Genetik Algoritmanın hibrit olarak kullanılmasıyla yapılmış, tahminlerdeki hata oranının en aza indirilmesi amaçlanmıştır. Veri seti olarak Desharnais veri seti kullanılmış, performans değerlendirme kriteri olarak Ortalama Bağıl Hata ve Pred(25) değerleri seçilmiştir. Çok Katmanlı Algılayıcı ve Çok Katmalı Algılayıcı ile Genetik Algoritmanın hibrit olarak kullanılması metotlarının her ikisi için de aynı eğitim seti kullanılmış, aynı projelerin eforları tahmin ettirilmiştir.

Tezin ilk bölümünde, tez konusu, konunun önemi, kullanılan materyal ve yöntemlerden bahsedilmiş, literatür özetleri verilmiştir.

Tezin ikinci bölümünde ilk olarak çalışmanın uygulama kısmının nasıl yapıldığı anlatılmıştır. Kullanılan veri seti, tahminler için kullanılan metotlar ve performans değerlendirme kriteri olarak seçilen kriterler hakkında bilgi verilmiştir. Daha sonra Yapay Sinir Ağları, İlk Yapay Sinir Ağları, İlk Yapay Sinir Ağlarından olan Tek Katmanlı Algılayıcılar, Basit Algılayıcı Modeli hakkında bilgi verilmiştir. Ardından, Çok Katmanlı Algılayıcılar ve Genetik Algoritmalar konuları anlatılmış, kullanılan eğitme kuralı ve programlar hakkında açıklamalar yapılmıştır.

Tezin son bölümünde ise tez ile ilgili sonuçlar verilmiş, değerlendirmeler yapılmıştır. Çalışma sonunda tezde kullanılan her iki metot için elde edilen tahmin sonuçları, gerçek efor değerleri ve her bir proje için hesaplanan Bağıl Hata değerleri tablolar halinde verilmiştir. Daha sonra, her iki metot için hesaplanan Ortalama Bağıl Hata ve Pred(25) değerleri sergilenmiş, performans karşılaştırması yapılmıştır.

2. MATERYAL VE YÖNTEM

Bu çalışmada yazılım efor tahmini Çok Katmanlı Algılayıcı ve Çok Katmanlı Algılayıcı modelinin Genetik Algoritma ile hibrit bir biçimde kullanılması olmak üzere iki farklı yolla yapılmıştır. Veri seti olarak Desharnais [14] veri seti kullanılmış, 10 adet projenin efor tahmini hem Çok Katmanlı Algılayıcı ile hem de Çok Katmanlı Algılayıcı'nın üzerine Genetik Algoritma uygulanmasıyla tahmin edilmiştir.

Desharnais veri setinin sahip olduğu 12 adet özellikler orjinal haliyle "Project", "TeamExp", "ManagerExp", "YearEnd", "Lenght", "Transactions", "Entities", "PointsNonAdjust", "Adjustment", "PointsAjust", "Language" ve "Effort" olarak sıralanmaktadır. Bu özellikler "Proje", "Takım Tecrübesi", "Yönetici Tecrübesi", "Bitim Tarihi", "Uzunluk", "Temel Mantıksal İşlemlerin Sayısı", "Varlık Sayısı", "Ayarlanmamış Noktalar", "Ayarlar", "Ayarlanmış Noktalar", "Kullanılan Programlama Dili" ve "Efor" özelliklerine karşılık gelmektedir.

Desharnais veri seti "Numeric" ve "Nominal" değerlerden, yani "Sayısal" ve "Sembolik" değerlerden oluşmaktadır. 12 özellikten 11 tanesi sayısal değerlerden oluşurken, sadece "Kullanılan Programlama Dili" özelliği "1,2,3" gibi sembolik değerlerden oluşmuştur. Bu veri setinin özelliklerinden olan "Takım Tecrübesi" özelliği, projeyi hazırlayan grup elemanlarının tecrübesini göstermektedir. Bu tecrübe, yıl olarak ölçülmüştür. "Yönetici Tecrübesi" özelliği, yöneticinin tecrübesini ifade etmektedir. Bu tecrübe de yıl olarak ölçülmüştür. "Efor" özelliği ise insan saati cinsinden ölçülmüştür.

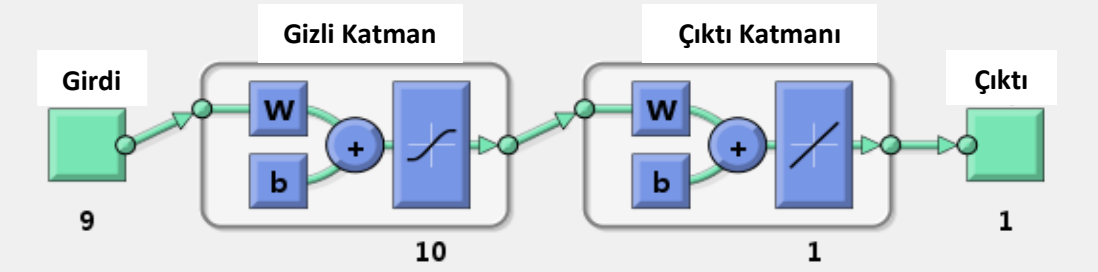
Desharnais veri setinde 38, 44, 66 ve 75 numaralı projelerde bazı eksik veriler bulunmaktadır. Bu veriler -1 ile doldurulmuştur. Bazı çalışmalarda veri eksikliği olan projeler kullanılmamakta ve 71 adet projeden faydalanılmakta iken, bazılarında ise tüm veri seti kullanılmaya devam etmektedir. Bu tez çalışmasında ise tüm veri setinden, yani 81 adet projeden yararlanılmıştır. 81 adet bu projenin bir kısmı eğitim seti olarak kullanılmıştır ve ağın eğitimi sağlanmıştır. Diğer bir kısmı da test seti

olarak kullanılmış, girdi değerleri ağa gösterilmiş, ağın bu girdi değerlerine göre makul tahminlerde bulunması beklenmiştir. Yapay Sinir Ağları eksik verilerle de çalışabilmekte, ağı eğitebilmekte ve gerçek değere çok yakın değerler üretebilmektedir. Bu da Yapay Sinir Ağları'nın en önemli ve en avantajlı özelliklerinden biridir. Eğitim seti ya da tahmin ettirilecek olan değerler bakımından sahip olunan verilerin her zaman tam ve tutarlı olamayacağı gerçeği göz önüne alındığında, Yapay Sinir Ağları en iyi tahmin üretebilen modeller arasında yer almaktadır.

Bu tez çalışmasında EK1'de verilen Desharnais veri setinden ilk 10 projenin efor değerleri tahmin ettirilirken, geriye kalan 71 proje eğitim seti olarak ayrılmıştır. Bu veri setinden "Takım Tecrübesi", "Yönetici Tecrübesi", "Uzunluk", "Temel Mantıksal İşlemlerin Sayısı", "Varlık Sayısı", "Ayarlanmamış Noktalar", "Ayarlar", "Ayarlanmış Noktalar", "Kullanılan Programlama Dili" ve "Efor" olmak üzere 10 adet özellik kullanılmıştır. Desharnais veri setinde her bir proje için verilen "Proje" ve "Bitim Tarihi" özellikleri performansı artırmak adına deneylerde kullanılmamıştır. Kullanılan özelliklerden "Takım Tecrübesi", "Yönetici Tecrübesi", "Uzunluk", "Temel Mantıksal İşlemlerin Sayısı", "Varlık Sayısı", "Ayarlanmamış Noktalar", "Ayarlar", "Ayarlanmış Noktalar", "Kullanılan Programlama Dili" özellikleri girdi olarak kullanılırken, "Efor" özelliği çıktı olarak kullanılmıştır. Sonuç olarak ağ, her bir örnek için 9'ar adet girdi ve 1'er çıktıdan oluşmuştur. İlk önce eğitim setinde bulunan girdiler ve çıktı değerleri ağa gösterilmiş, ağın girdi ve çıktılar arasındaki ilişkiyi çözebilmesi yani eğitilmesi sağlanmıştır. Eğitimden sonra, efor değerleri tahmin ettirilecek projelere ait olan girdi değerleri ağa gösterilmiştir ve ağın çıktı üretmesi sağlanmıştır.

Performans değerlendirme için gerçek efor değeri ile ağın tahmin ettiği efor değerinin mutlak farkının gerçek efor değerine bölünmesiyle Bağıl Hata bulunmuş, ardından Ortalama Bağıl Hata hesaplanmıştır. Bağıl Hata değeri 0.25'ten küçük ya da eşit olan projeler sayılıp toplam proje sayısına bölünmüş, Pred(25) değerleri hesaplanmıştır. 10'ar kez deneme yapılmış, bu denemeler sonucunda en iyi performansı gösteren deneyler karşılaştırma için seçilmiştir.

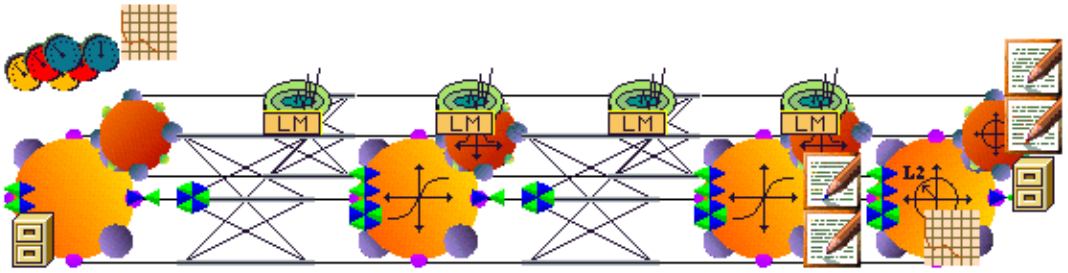
Çok Katmanlı Algılayıcı modelinde 63 adet proje eğitim seti için kullanılırken, 8 adet proje onaylama ve test etme için kullanılmıştır. Sistem 10 adet gizli nöron kullanılarak yapılandırılmış, Levenberg-Marquardt Geri Yayılım Algoritması kullanılarak 1000 iterasyonla eğitilmiştir. Oluşturulan ağın yapısı Şekil 2.1'de gösterilmiştir.



Şekil 2.1. Ağın Yapısı

Şekil 2.1'de de görüldüğü üzere, her bir proje için 9 adet girdi sisteme verilmiş, gizli katman ise 10 adet gizli nörondan oluşmuştur.

İkinci bir yöntem olan Çok Katmanlı Algılayıcı ile Genetik Algoritma'nın hibrit bir biçimde kullanılması modelinde ise 71 adet proje eğitim seti için kullanılırken, herbir projenin sahip olduğu 9'ar adet girdi değerlerinin hepsine Genetik Algoritma uygulanmış, ağ için gerekli olan en iyi girdi değerlerinin tespit edilmesi amaçlanmıştır. Eğitim setinde bulunan 10 adet proje çapraz doğrulama için kullanılırken, 3 adet proje de ağ performansını test etmek için kullanılmıştır. Gizli katman sayısı 1 olarak seçilmiştir. İşleme elemanları sayısı 2 olarak belirlenirken, bu elemanlara da Genetik Algoritma uygulanarak işleme elemanlarının optimize edilmesi amaçlanmıştır. Hem gizli katman, hem de çıktı katmanı için transfer fonksiyonu olarak TanhAxon, öğrenme kuralı olarak Levenberg-Marquardt kuralı seçilmiş, eğitim sağlanmıştır. Çapraz doğrulama Ortalama Bağıl Hata değerinde artma başladığında yani sistem fazla eğitildiğinde eğitim durdurulmuştur. Eğer bu değerde artma olmazsa, sistem 1000 iterasyon boyunca eğitilmiştir. Ağırlıkların güncellenmesi ise tüm eğitim seti eğitildikten sonra toplu halde yapılmıştır. Şekil 2.2'de belirtilen özelliklerle oluşturulan sistemin yapısı gösterilmektedir.



Şekil 2.2. Çok Katmanlı Algılayıcı ve Genetik Algoritma ile Oluşan Ağın Yapısı

Şekil 2.2'de gösterilen sistem kullanılarak eğitim sağlanmış, 50 adet kromozom ile 1000'er iterasyon sonucunda 100 adet yeni nesil oluşturulmuş, oluşturulan nesillerden en iyi performansa sahip olan değerlendirmeye alınmıştır.

Bu çalışmalarda kullanılan yöntemler ilerleyen sayfalarda açıklanmıştır.

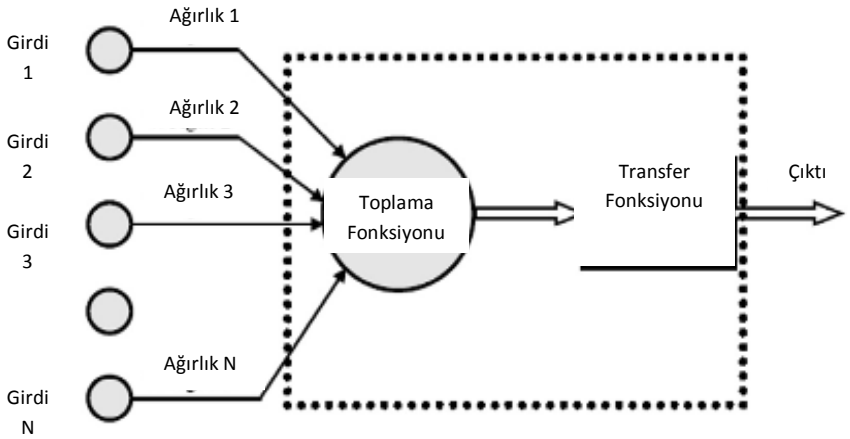
2.1. Yapay Sinir Ağları

Yapay Sinir Ağları, öğrenme yolu ile yeni bilgiler oluşturabilen bilgisayar sistemleridir. Yapay Sinir Ağları'nın yeni bilgiler oluşturabilmesi için ilk önce ilgili olayın örnekleri ile eğitilmesi gerekmektedir. Eğitimle birlikte Yapay Sinir Ağı, ilgili olaya benzer girdiler sisteme verildiğinde, sahip olduğu genelleme yeteneği ile uygun çıktı setlerini üretebilmektedir. Çünkü ağ, eğitimle birlikte sisteme verilen girdi ve çıktılar arasındaki yani örnekteki olaylar arasındaki ilişkiyi belirlemiştir. Belirlediği ilişki ile kendisine verilen yeni girdilerin çıktılarını tahmin edebilme yeteneğine ulaşmıştır. Var olan örneklerden bir bölümü eğitim seti olarak kullanılırken, diğer bir bölümü ise test seti olarak kullanılmaktadır. Buradaki amaç, eğitim seti yardımıyla örnekler arasındaki ilişkinin ağ tarafından belirlenmesini sağlamak ve test seti yardımıyla da bu ağın performansını ölçebilmektir. Test setinde bulunan örnekler ağa gösterilir ve ağın vereceği çıktı değerleri gözlenir. Eğer ağ, daha önce hiç görmediği örneklere makul çıktılar üretiyorsa, iyi bir performans gösteriyor demektir. Yalnız, ağ hiç görmediği örneklere karşı makul çıktılar üretemiyorsa, performansı kötü

demektir. Bu durumda sistem yeniden eğitilebilir. Ya da ikinci bir yol olarak eğitim seti değiştirilip, yeni bir eğitim seti ile sistem tekrardan eğitilip, test edilebilir [15].

Yapay Sinir Ağları, her katmanında birbiriyle bağlantılı olan ve nöron ya da düğüm olarak adlandırılan işleme elemanları içeren, genellikle girdi katmanı, gizli katmanlar ve çıktı katmanı olmak üzere 3 katmandan oluşan hesaplamalı bir zeka tekniğidir. Nöronlar birbirlerine iletişim bağlantıları ile bağlıdır. İletişim bağlantılarında iletişim ağırlıkları bulunmaktadır. Her bir nörona, birden çok girdi girişi olabilir ama tek bir çıktı sinyali üretilir. Yapay Sinir Ağı modelinin geliştirilmesini sağlamak ve performansını artırmak adına ağın işlenmesi eğitim, doğrulama ve test etme olmak üzere 3 aşamada yapılmaktadır [16].

Ağın işlenmesi aşamalarından biri olan eğitim aşamasında, verilen örnek olaylarla yani girdi değerleriyle ağın eğitimi sağlanmaktadır. Bu eğitim sayesinde ağ, kendisine gösterilen örnek olaylar arasındaki ilişkiyi belirleyerek, kendisine gösterilecek olan örneklere makul çıktılar üretebilecek hale gelmektedir. İkinci aşama olan doğrulama aşamasında eğitimin ne zaman durdurulması gerektiği test edilmektedir. Eğitimin yeterli oranda yapılmaması ağın örnek olaylar arasındaki ilişkiyi belirleyememesine neden olmaktadır. Sonuç olarak tahmin edilen çıktı değerleri makul çıktı değerleri olamamaktadır. Eğitimin fazla yapılması da tahmin performansını düşürmektedir. Test aşamasında ise en son elde edilen çözümün tahmin performansı test edilmektedir.



Şekil 2.3. Yapay Nöron

Şekil 2.3'te [16] yapay bir nöronun yapısı gösterilmiştir. Bu nöron yapısında toplam N adet girdi ve N adet ağırlık değeri bulunmaktadır. Çıktı değerinin hesaplanabilmesi için ilk önce hücreye giren net girdi değerinin hesaplanması gerekmektedir. Bu değerin hesaplanması işlemini Toplama Fonksiyonu yapmaktadır. En yaygın olarak kullanılan Toplama Fonksiyonları formüllerinden biri ağırlıklı toplamı bulmaktır. Ağırlıklı toplam bulma işleminde her girdi bağlı olduğu ağırlık değerleriyle çarpılmaktadır. Tüm çarpım değerleri elde edildikten sonra, bu değerler toplanmakta ve net girdi değerine ulaşılmaktadır. Daha sonra net girdi değerinin çıktı elde edilebilmesi için işlenmesi gerekmektedir. Net girdi, Transfer Fonksiyonu'nda ya da başka bir deyişle Aktivasyon Fonksiyonu'nda işlenmektedir. Bu amaçla kullanılan birçok Aktivasyon Fonksiyonu formülleri bulunmaktadır. Sonuç olarak, girilen net girdi değerine karşılık gelen çıktı değeri elde edilmiş olunur.

2.2. İlk Yapay Sinir Ağları

Yapay Sinir Ağları tarihsel gelişimi 1970 öncesi ve 1970 sonrası olarak ikiye ayrılmaktadır. 1970 öncesinde Yapay Sinir Ağları ile ilgili birçok çalışma yapılmıştır ve gelişmeler gözlenmiştir. Örneğin; 1943 yılında ilk temel hesap yapan nöron modeli tasarlanmıştır, 1950'lerde ilk nöro bilgisayarlar yapılmıştır, 1958'de nöron benzeri perceptron elemanları ortaya atılmıştır, 1960-1962 yıllarında Widrow-Hoff öğrenme algoritması geliştirilmiştir [17]. Yalnız, 1969 yılında XOR probleminin çözülememesi üzerine, çalışmalar durmuş ve bu konuya olan ilgi azalmıştır. 1970 sonrasında az sayıda araştırmacının çalışmalarını sürdürmesi ve XOR problemini çözmeleri üzerine bu konuya olan ilgi tekrardan artmış, yeni çalışmalar yapılmış, yeni modeller geliştirilmiştir.

İlk Yapay Sinir Ağları modellerinden olan Tek Katmanlı Algılayıcılar Modeli, Tek Katmanlı algılayıcılardan olan Perceptron modeli bu bölümde açıklanacaktır.

2.2.1. Tek Katmanlı Algılayıcılar

Tek Katmanlı Algılayıcılar girdi katmanı ve çıktı katmanı olmak üzere iki katmandan oluşmaktadır. Her bir girdi değeri kendi ağırlık değeriyle çarpılmaktadır ve bulunan bu çarpım değerleri toplanmaktadır. Ağın çıktısının 0 olma ihtimaline karşı, değeri 1 olan eşik değeri çarpım değerleri toplamına eklenmektedir. Daha sonra, elde edilen sonuç, aktivasyon fonksiyonundan geçirilmekte ve ağın çıktı değeri elde edilmektedir. Ağın çıktı değeri 2.1 Eşitliği'nde gösterildiği gibi hesaplanmaktadır [18].

$$\text{Çıkış} = f\left(\sum_{i=1}^{N} w_i x_i + \varphi\right) \tag{2.1}$$

2.1 Eşitliği'nde, N ifadesi toplam girdi sayısını, w_i ifadesi i'inci girdinin ağırlık değerini, x_i ifadesi i'inci girdinin değerini, φ ise eşik değerini göstermektedir.

Tek katmanlı Algılayıcı'da amaç, ağa gösterilen örneklerin iki grup halinde birbirlerinden ayrılmalarını sağlayabilmektir. Ayırma işlemi ise doğru parçası aracılığıyla yapılmaktadır. Örnekleri iki farklı gruba ayıran doğru ise eşik değer fonksiyonu aracılığı ile tespit edilmektedir. Ağın çıktısı 1 ya da -1'dir. Buradaki 1 ve -1 değerleri ağa gösterilen örneğin bulunabileceği sınıfları temsil etmektedir. Ağın çıktısı 2.2 Eşitliği'nde gösterildiği gibi hesaplanmaktadır.

$$f(g) = \begin{cases} 1, & \text{Çıktı} > 0 \\ -1, & aksi\ takdirde \end{cases} \tag{2.2}$$

2.2 Eşitliği'nde, ağın çıktısı 0'dan büyük olduğunda örnek 1 sınıfına konmuştur. 0'dan küçük olduğunda ise -1 sınıfına konmuştur. Ağın çıktısı 0 olduğunda ise örnek -1 sınıfına konmuştur. Yalnız, ağın çıktısının 0 olma durumunda o örneğin ait olacağı sınıf, tasarımcının kabulüne kalmış olup, farklılık gösterebilmektedir.

Ağlarda w değerlerinin yani ağırlık değerlerinin güncellenmesi şeklinde öğrenme gerçekleşmektedir. Buradaki öğrenmede amaç, örnekleri en iyi iki gruba yani en iyi

iki sınıfa ayıracak olan doğrunun tespit edilmesidir. Bu doğru ise ağırlık değerlerinin güncellenmesiyle en iyi halini almaktadır. Bazen eşik değerinin de güncellenmesi gerekmektedir. Ağırlık değerlerinin değiştirilmesiyle yani güncellenmesiyle doğrunun eğiminin değiştirilmesi, eşik değerinin değiştirilmesiyle de doğrunun sınıflar arasında kayabilmesi sağlanmaktadır. Örnekleri iki ayrı sınıfa ayıran doğrunun denklemi, x_1 ve x_2 girdi değerleri formülleri Eşitlik 2.3, 2.4 ve 2.5'te gösterilmiştir.

$$w_1.x_1 + w_2.x_2 + \varphi = 0 \tag{2.3}$$

$$x_1 = -\left(\frac{w_2}{w_1}\right)x_2 - \varphi/w_2 \tag{2.4}$$

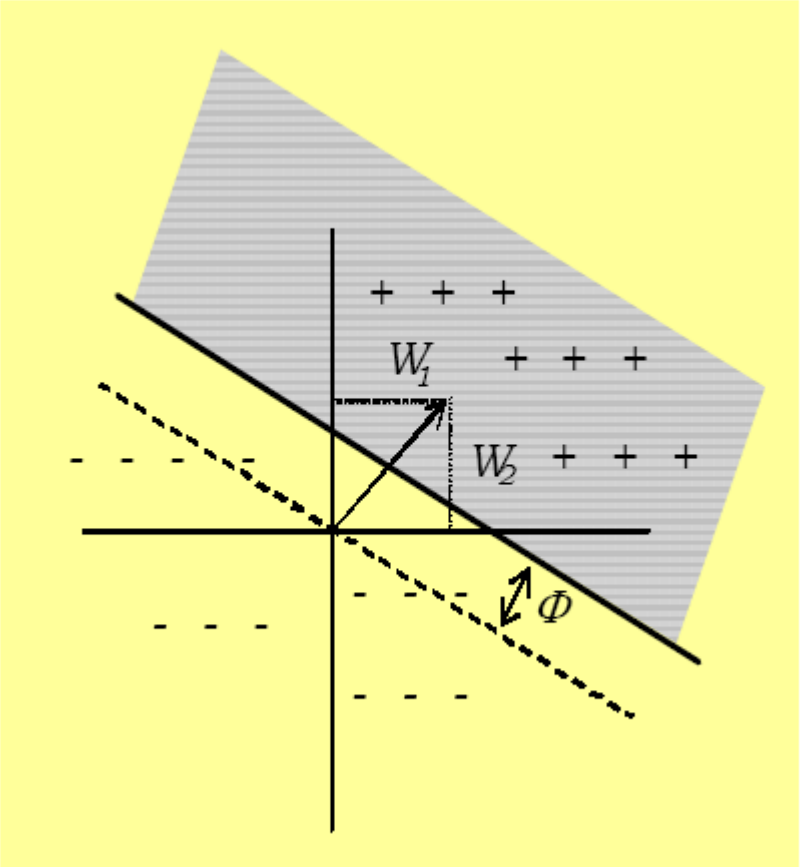
$$x_2 = -\left(\frac{w_1}{w_2}\right)x_1 - \varphi/w_1 \tag{2.5}$$

Ağa gösterilen örnekleri iki sınıfa ayıran doğrunun pozisyonunu en iyi bir şekilde ayarlayabilmek için ağırlık değerleri değiştirilmektedir. t zaman biriminde ağırlık değeri Δw kadar değiştirilirse, elde edilen yeni ağırlık değeri 2.6 Eşitliği'nde gösterildiği gibi bulunmaktadır.

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \tag{2.6}$$

t zaman biriminde eşik değeri Δt kadar değiştirilirse yeni eşik değeri 2.7 Eşitliği'nde gösterildiği gibi hesaplanmaktadır.

$$\varphi(t+1) = \varphi(t) + \Delta\varphi(t) \tag{2.7}$$



Şekil 2.4. Ağırlıkların ve Sınıf Ayıracı Olan Doğrunun Gösterimi

Şekil 2.4'te [18] ağırlıkların ve sınıfları birbirinden ayıran doğrunun gösterimi bulunmaktadır. Bu şekilde de gösterildiği üzere, doğru parçası yardımıyla iki farklı sınıf oluşturulmuştur.

2.2.2. Basit Algılayıcı Modeli

Basit Algılayıcı modelinde çıktı değerleri aktif ya da inaktif sinaps karşılığına gelen 0 ya da 1 değerleridir. Basit bir algılayıcı 2.8 ve 2.9 Eşitlikleri'nde gösterildiği gibi tanımlanmaktadır:

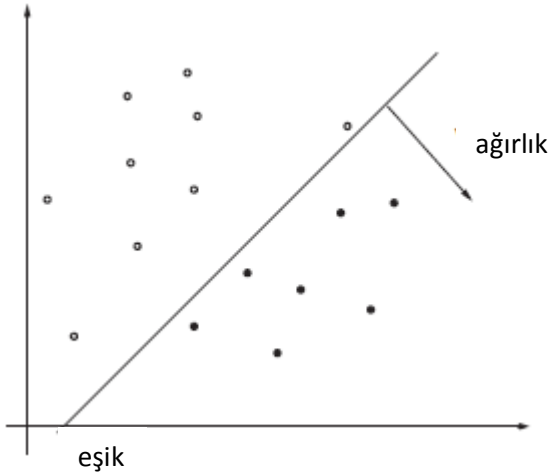
$$\mathbb{R}^n \ni x \rightarrow H(w^t x - \varphi) \in \mathbb{R} \tag{2.8}$$

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & aksi\ takdirde \end{cases} \quad (2.9)$$

2.8 Eşitliği'nde w ifadesi ağrılık değerini, φ ifadesi eşik değerini, $w^t x$ ifadesi w ve x vektörlerinin iç çarpımını, $H: \mathbb{R} \rightarrow \mathbb{R}$ ise lineer olmayan aktivasyon fonksiyonunu ifade etmektedir. 2.9 Eşitliği'nde de görüldüğü üzere, girdilerin ağırlıklı toplamı ve eşik değeri arasındaki fark 0'dan büyük ya da 0'a eşitse ağın çıktısı 1'dir. Aksi taktirde ağın çıktısı 0'dır [19].

Basit Algılayıcı, birbirlerinden lineer olarak ayrılabilen örnekleri ayırabilmekte, bir sınıflayıcı olarak görev yapmaktadır. Şekil 2.5'te [19] de görüldüğü gibi örnekler iki farklı sınıfa ayrılmış, w yani ağırlık ifadesi hiperdüzlemin yönünü gösterirken, φ eşik değeri ise offset noktasını göstermiştir.

Ağın çıktı üretmesi sonucunda, üretilen çıktı ile beklenen değer aynı olmayabilir. Beklenen çıktı 1 iken, ağ 0 değerini üretebilir. Ya da tam tersi de olabilir. Bu durumda ağın ağırlıklarının artırılması ya da azaltılması gerekmektedir. Ağın ağırlıklarının değiştirilmesi, ağın eğitiminin sağlanmasıdır. Beklenen değer 0 iken ağ 1 değerini üretiyorsa ağırlıkların azaltılması, beklenen değer 1 iken ağ 0 değerini üretiyorsa ağın ağırlıklarının arttırılması gerekmektedir. Ağırlıkların güncellenmesi Eşitlik 2.10 ve 2.11'de verilmiştir.



Şekil 2.5. Eşik Değeri ve Ağırlık ile Oluşturulan Sınıflayıcı Görevindeki Algılayıcı

$$w(t+1) = w(t) - \mu x \tag{2.10}$$

$$w(t+1) = w(t) + \mu x \tag{2.11}$$

Eşitlik 2.10 ve 2.11'de bulunan $w(t+1)$ ifadeleri ağırlığın yeni değerini gösterirken, $w(t)$ ise ağırlığın eski değerini göstermektedir. x girdinin değerini gösterirken, μ ise değeri 2.12 Eşitliği'nde gösterilen aralıkta seçilen öğrenme katsayısını ifade etmektedir.

$$0 < \mu \leq 1 \tag{2.12}$$

2.10 Eşitliği ağın ürettiği değerin 1, beklenen değerin ise 0 olduğu durumdur. 2.11 Eşitliği ise üretilen değerin 0, beklenen değerin ise 1 olduğu durumu göstermektedir.

Basit Algılayıcılar lineer olarak ayrılabilen örnekleri sınıflandırabilirken, birbirine tek bir çıktı nöronu ile bağlanan iki girdi nöronu gerektiren ve çıktı kısmının sadece girdi kısımlarından birisi aktif olduğunda aktif olan XOR probleminin çözümünü gerçekleyememektedir [20].

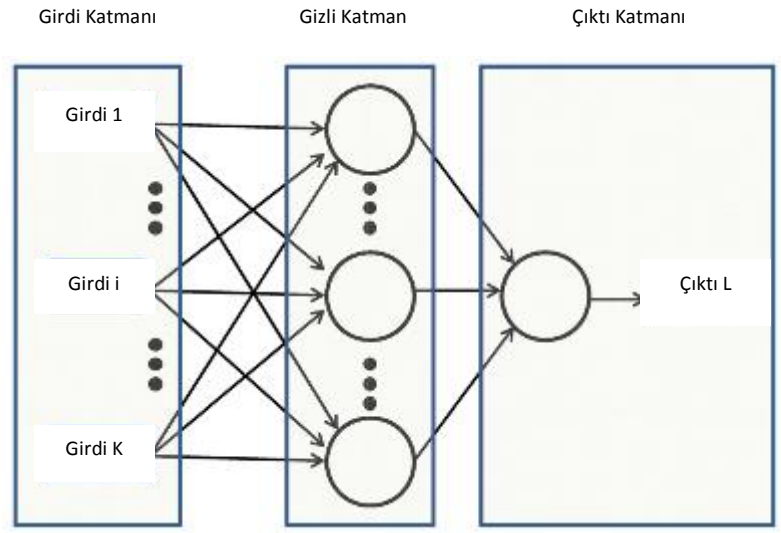
2.3. Çok Katmanlı Algılayıcılar

Minsky M. ve Papert S. [21] yaptıkları çalışma ile basit algılayıcı modelinin XOR problemini çözemediğini göstermiş, Yapay Sinir Ağları'nın doğrusal olmayan problemlerin çözümünü gerçekleyemediğini iddia etmiştir. Bu durum, Yapay Sinir Ağları'na olan ilgiyi azaltmıştır. Birçok kişi bu konuyla ilgili olan çalışmalarını bırakmış, sadece birkaç araştırmacı çalışmalarını sürdürmüştür. Rumelhart ve arkadaşları [22] tarafından Çok Katmanlı Algılayıcılar modelinin geliştirilmesiyle Yapay Sinir Ağları'na olan ilgi tekrar artmış, yapılan çalışmalar hız kazanmıştır [15].

Çok Katmanlı Algılayıcılar modelinde amaç, ağın ürettiği çıktı ile gerçek çıktı arasındaki farkın yani hatanın en aza indirilmesidir. Bu ise hatanın ağa yayılmasıyla

gerçekleştirilmektedir. Bu nedenle Çok Katmanlı Algılayıcılar modeline Geriye Yayma modeli ya da Hata Yayma modeli de denmektedir.

Çok Katmanlı Algılayıcılar modeli girdi katmanı, gizli katman (ara katman) ve çıktı katmanı olmak üzere üç adet katmandan oluşmaktadır. Gizli katman sayısı birden çok olabilmektedir. Girdi katmanının bilgi işleme gibi bir görevi olmayıp, dış dünyadan gelen girdi veya girdilerin gizli katmana iletilmesini sağlamakla görevlidir. Gizli katmanlar girdi katmanından gelen bilgileri ya da kendinden önceki ara katmandan gelen bilgileri işlemektedir. Çıktı katmanı ara katmandan gelen bilgileri işleyerek ağın çıktısını üretmektedir. Şekil 2.6'da [23] ileri beslemeli bir Çok Katmanlı Algılayıcı'nın yapısı gösterilmiştir.



Şekil 2.6. İleri Beslemeli Çok Katmanlı Algılayıcı Yapısı

Şekil 2.6'da görüldüğü gibi Çok Katmanlı Algılayıcı modeli girdi katmanı, gizli katman ve çıktı katmanından oluşmaktadır. Girdi katmanı, gizli katman ve çıktı katmanının hepsinde de birden çok işlem elemanı olabilmektedir. Şekil 2.6'da girdi katmanı ve gizli katman birden çok işlem elemanına sahipken, çıktı katmanı tek bir işlem elemanından oluşmaktadır. Girdi katmanındaki her bir işlem elemanı, bir sonrasındaki katmanda bulunan tüm işlem elemanlarıyla bağlı bir haldedir. Ara katmanda bulunan her bir işlem elemanı ise kendinden sonra yer alan katmandaki

tüm işlem elemanlarıyla bağlıdır. Çıktı katmanında bulunan tüm işlem elemanları ise kendinden önceki katmanda bulunan tüm işlem elemanlarıyla bağlı bir halde bulunmaktadır.

Çok Katmanlı Algılayıcılarda gizli nöron sayısı büyük bir öneme sahiptir. Gizli katmanda bulunan nöron sayısının arttırılması ya da azaltılması ağ performansını önemli bir derecede etkilemektedir. Fakat, ağda en iyi performansı veren gizli nöron sayısını bulmaya yarayan herhangi bir analitik metot bulunmamaktadır [23].

Çok Katmanlı Algılayıcılar öğretmenli öğrenme gerçekleştirmektedir. Tüm girdiler ve bu girdilere karşılık gelen çıktılar ağa gösterilmektedir. Ağın ise, kendisine gösterilen girdiler ve bu girdilerin çıktılarına bakarak, girdi ve çıktıların aralarındaki ilişkiyi öğrenip, genelleme yapabilmesi, kendisine gösterilen herhangi bir girdi değerine karşılık makul çıktılar üretebilmesi beklenmektedir. Çok Katmanlı Algılayıcı'nın öğrenme kuralı ileri besleme ve geriye yayılım olmak üzere iki kısımdan oluşmaktadır.

2.3.1 İleri Beslemeli Hesaplama

Sinyaller girdi katmanından geçmekte ve soldan sağa doğru ileri yönlü olarak yayılmaktadır. İleri beslemeli hesaplamada gizli katmana ait olan net girdi değeri, gizli katmanda bulunan bir düğüme ait olan çıktı değeri, çıktı katmanına ait olan net girdi değeri ve çıktı katmanında bulunan bir düğüme ait olan çıktı değeri hesaplamaları Eşitlik 2.13, 2.14, 2.15 ve 2.16'da verilmiştir [24].

$$n_j^{(H)} = \sum_i w_{ij}{}^{(H)} x_i \qquad (2.13)$$

2.13 Eşitliği'nde $n_j^{(H)}$ ile j gizli katmanına ait olan net girdi değeri ifade edilmiştir ve bu girdi değerini hesaplama formülü verilmiştir. $n_j^{(H)}$ gelen sinyallerin ağırlıklı toplamıdır. Bu eşitlikte $w_{ij}^{(H)}$ ile girdi katmanındaki i düğümü ile, gizli katmanda

bulunan j düğümü arasındaki bağın ağırlığı temsil edilmiştir. x_i ise girdi katmanında bulunan i düğümünden çıkan girdi sinyalidir.

$$H_j = f\left(n_j^{(H)}\right) = f\left(\sum_i {w_{ij}}^{(H)} x_i\right) \tag{2.14}$$

2.14 Eşitliği'nde gizli katmanda bulunan j düğümünün çıktısı hesaplanmıştır. $f(.)$ ise türevi alınabilen ve lineer olmayan aktivasyon fonksiyonudur.

$$n_k^{(O)} = \sum_j {w_{jk}}^{(O)} H_j = \sum_j {w_{jk}}^{(O)} . f\left(\sum_i {w_{ij}}^{(H)} x_i\right) \tag{2.15}$$

2.15 Eşitliği'nde $n_k^{(O)}$ ifadesi çıktı katmanına ait olan net girdi değerini göstermektedir. Bu girdi değeri gizli katman çıktılarının ağırlıklı toplamıdır. 2.15 Eşitliği'nde $w_{jk}^{(O)}$ ağırlığı, çıktı katmanındaki k düğümü ile gizli katmanda bulunan j düğümü arasındaki ağırlıktır.

2.16 Eşitliği'nde çıktı katmanında bulunan k düğümünün çıktısı hesaplanmıştır.

$$O_k = f\left(n_k^{(O)}\right) = f\left(\sum_j {w_{jk}}^{(O)} f\left(\sum_i {w_{ij}}^{(H)} x_i\right)\right) \tag{2.16}$$

2.16 Eşitliği'nde çıktının hesaplanmasıyla ileri beslemeli hesaplama işlemi sonra ermektedir. Çıktıların hesaplanması, çıktı katmanında bulunan tüm işlem elemanları için yapılmaktadır.

İleri beslemeli hesaplamada amaç, ağın ürettiği çıktının hesaplanabilmesidir. İlk önce gizli katmana giren net girdi değeri hesaplanmaktadır. Daha sonra net girdinin aktivasyon fonksiyonundan geçirilmesiyle, bu net girdiye sahip düğümün çıktısı bulunmaktadır. Bu çıktı değeri kendinden sonra bulunan katman için girdi niteliği taşımaktadır. Her bir düğüm için çıktı değerleri hesaplanmaktadır. Son olarak, bu çıktı değerleri çıktı katmanına girdi niteliğinde girmekte, ağırlıklı toplama yöntemi ile net girdi elde edilmekte ve net girdinin aktivasyon fonksiyonunda işlenmesiyle de

çıktı elde edilmektedir. Bu işlemler gizli katmanlar ve çıktı katmanında bulunan tüm işlem elemanları için tekrarlanmaktadır.

2.3.2. Geriye Yayılımlı Hesaplama

Geriye yayılımlı hesaplamada amaç, beklenen çıktı ile ileri beslemeli hesaplama sonucu ağın ürettiği çıktı arasındaki farkın yani hata değerinin ağırlıklara dağıtılarak hata değerinin düşürülmesini sağlamaktır. Çıktı katmanı ve gizli katman arasındaki ağırlıkların güncellenmesi Eşitlik 2.17'de verilmiştir.

$$\Delta w_{jm}(t) = \mu \delta_m O_j + \sigma \Delta w_{jm}(t-1) \tag{2.17}$$

2.17 Eşitliği'nde $\Delta w_{jm}(t)$ ile *t.* zaman aralığında ara katmanda bulunan *j.* işlem elemanı ile, çıktı katmanında bulunan *m.* işlem elemanı arasındaki ağırlık değerindeki değişimi gösterilmektedir. μ öğrenme kat sayısını, σ ise algılayıcının yerel bir optimum noktaya takılmasını önleyen momentum değerini ifade etmektedir. O_j ise gizli katmanda bulunan *j.* işlemin çıktısıdır. δ_m hata ifadesi 2.18 Eşitliği'nde hesaplanmıştır.

$$\delta_m = O_j(1 - O_j)E_m \tag{2.18}$$

2.18 Eşitliği'nde $E_m$ifadesi çıktı katmanındaki *m.* işlem elemanının hata değeridir.

Ağırlıklardaki değişim miktarı hesaplandıktan sonra yeni ağırlık değeri hesaplama formülü Eşitlik 2.19'de verilmiştir.

$$w_{jm}(t) = w_{jm}(t-1) + \Delta w_{jm}(t) \tag{2.19}$$

Girdi katmanı ile gizli katman arasındaki ağırlıkların güncellenmesi aşamaları ise Eşitlik 2.20, 2.21 ve 2.22'de verilmiştir.

$$\Delta w_{kj}(t) = \mu \delta_j O_k + \sigma \Delta w_{kj}(t-1) \quad (2.20)$$

2.20 Eşitliği'nde μ öğrenme katsayısı, δ_j hata terimi, σ momentum değeridir. δ_j hatasının hesaplanması Eşitlik 2.21'te verilmiştir.

$$\delta_j = O_j(1 - O_j) \sum_m \delta_m w_{jm} \quad (2.21)$$

2.21 Eşitliği'nde de görüldüğü gibi girdi katmanı ile ara katman arasındaki ağırlıkların değiştirilmesinde çıktı katmanındaki tüm işlem elemanlarının hatası hesaplamaya katılmaktadır.

Ağırlıkların yeni değerleri ise 2.22 Eşitliği'nde hesaplanmıştır.

$$w_{kj}(t) = w_{kj}(t-1) + \Delta w_{kj}(t) \quad (2.22)$$

2.4. Yapay Sinir Ağları Avantaj ve Dezavantajları

Yapay Sinir Ağları modelinin avantajları olduğu kadar dezavantajları da bulunmaktadır. Buna rağmen, Yapay Sinir Ağları, her problem için farklı şekillerde çözümler sunabilmekte, makul sonuçlar üretebilmekte ve finansal alandan mühendislik ve tıp alanına kadar birçok alanda kullanılabilmektedir. Ağların dezavantajlarını giderebilmek için Yapay Sinir Ağları modeli diğer modellerle hibrit olarak kullanılabilmekte, ağın yapısı daha doğru oluşturularak, performans arttırılmaya çalışılmaktadır.

Yapay Sinir Ağları avantajları şu şekilde sıralanabilmektedir [25,26]:

- Ağa örnek girdiler ve her bir girdiye ait olan çıktılar gösterilmektedir. Gösterilen girdi ve çıktı değerleri sayesinde ağ, girdi ve çıktılar arasındaki ilişkiyi çözüp genelleme yapabilecek duruma gelmektedir. Bu sayede ağ, kendisine gösterilen yeni girdi değerlerine karşılık makul çıktılar üretebilmektedir.

- Yapay Sinir Ağları hesaplamaları paralel bir şekilde yürütülebilmektedir. Bu işlem için özel donanım araçları üretilmektedir. Böylece hesaplama süresi azaltılmaktadır.
- Geleneksel teknolojilerin çözümünde zorlandığı ya da çözemediği problemler, Yapay Sinir Ağları aracılığı ile çözülebilmektedir. Bu problemlere örnek olarak örüntü tanıma ve tahmin problemleri verilebilmektedir.
- Risk yönetimi, veri doğrulama, satış tahmini gibi tahmin gerektiren birçok alanda kullanılabilmektedir.

Yapay Sinir Ağları dezavantajları şu şekilde sıralanabilmektedir [15]:

- Her probleme uygun ağ yapısının belirlenmesi gerekmektedir. Bu da genellikle deneme yanılma yöntemiyle yapılmaktadır. Bu durum da problemin çözülememesine ya da ağ performansının düşük olmasına neden olabilmektedir. Yapay Sinir Ağları ile elde edilen sonuçlar kabul edilebilir sonuçlardır ama en iyi sonuçlar olamayabilirler.
- Ağda bulunan öğrenme kat sayısı, her katmanda bulunması gereken nöron sayısı gibi parametreler kullanıcı tarafından belirlenmektedir. Bu parametrelerin doğru belirlenebilmesi, kullanıcı tecrübesine kalmıştır.
- Ağın eğitimin ne zaman sonlandırılması gerektiği bilgisini veren bir yöntem bulunmamaktadır. Elde edilen çözümler iyi çözümler olsa da en iyi çözümü garanti edememektedir.
- Ağın davranışları açıklanamamaktadır. Elde edilen sonuçların ağ tarafından nasıl ve neden üretildiği bilinememektedir.

2.5. Genetik Algoritma

Genetik Algoritma fikri J. Holland tarafından ileri sürülmüştür. Holland, 1975 yılında yaptığı çalışmada [27], karmaşık yapıların basit bit dizileri yardımıyla kodlanabileceğini göstermiştir. Yapılar, problemin çözümlerini temsil etmektedir. Genetik algoritmanın başlangıç popülasyonu olarak kullanacağı popülasyon, bu

çözümlerin bir kısmından oluşmaktadır. Daha sonra, temel genetik operatörler aracılığıyla yeni popülasyonlar yani yeni çözüm kümeleri oluşmaktadır. Bu işlem uygun bir şekilde yapıldığında ise çözüm popülasyonunun kalitesi artmaktadır [28].

2.5.1. Basit Bir Genetik Algoritmanın Yapısı

Basit bir Genetik Algoritma'nın yapısı çözümlerin temsil şekli, başlangıç popülasyonu oluşturma yöntemi, uygunluk değerlendirme fonksiyonu, kullanılan genetik operatörler ve kontrol parametreleri olmak üzere 5 elemandan oluşmaktadır.

Çözümlerin Temsil Şekli

Genetik Algoritmalarda çözümlerin temsil şekli olarak en çok kullanılan algoritmalardan biri ikili kodlanmış algoritmalardır. Yapılar 0 ve 1 ile ifade edilmektedir. Ayrıca sayısal problemler için ikili gösterimin yanı sıra gerçek rakamlar da kullanılabilmektedir. Gruplama problemlerinde ise çözümler tekrar eden semboller şeklinde gösterilmektedir [28,29].

Başlangıç Popülasyonu Oluşturma

Başlangıç popülasyonu rastgele sayı üreticisi yardımıyla oluşturulmaktadır.

Uygunluk Değerlendirme Fonksiyonu

Uygunluk değerlendirme fonksiyonunun seçimi büyük bir öneme sahiptir. Seçilen uygunluk fonksiyonu ile her bir birey için uygunluk değeri hesaplanmakta ve uyumluluk değeri yüksek olan bireyler gelecek nesillere aktarılmaktadır. Probleme uygun olmayan bir fonksiyonun seçilmesi, bireylerin yani çözümlerin uygunluk

değerlerinin doğru hesaplanamamasına, yanlış çözümler üretilmesine neden olmaktadır [30].

Genetik Operatörler

Genetik Algoritmalarda kullanılmak üzere çok sayıda genetik operatörler geliştirilmiştir. Bu operatörlerden tekrar üretme, çaprazlama ve mutasyon operatörleri en çok kullanılan operatörlerdendir. Nesiller bu operatörler aracılığıyla farklılaşmakta, yeni ve daha sağlıklı bireyler yani daha doğru çözümler elde edilmektedir.

Tekrar üreme operatörü, uygunluk değeri yüksek olan bireylerin sayısının artmasını yani popülasyonda baskın bir oranda bulunmasını sağlarken, uygunluk değeri düşük olan bireylerin ise sayıca azalarak popülasyondan kaybolmalarını sağlamaktadır.

Çaprazlama operatörü, önceden rastgele seçilmiş olan iki bireyden bu operatör yardımıyla yeni bireylerin üretilmesini sağlamaktadır. Oluşan yeni bireyler, eski bireylerden özellikler taşımaktadır. Amaç ise uygunluk derecesi yüksek olan bireyler elde etmektir. Çaprazlama sonunda hem seçilmiş olan yapıların hem de oluşan yapıların uygunluk değeri hesaplanmakta ve uygunluk değeri düşük olan yapılar popülasyondan atılmaktadır. Böylece popülasyon büyüklüğü korunmuş olmaktadır. Şekil 2.7'de Ebeveyn A, Ebeveyn B, A ve B Ebeveynlerinin çaprazlanması sonucu oluşan A ve B Çocukları verilmiştir. Çaprazlama sonucunda, eski bireylerden de özellikler taşıyan yeni bireyler oluşturulmuştur.

1	1	**0**	**1**	0	**0**	0	1	**0**	1

a:

0	**0**	**1**	0	1	**0**	0	**0**	1	**0**

b:

1	**0**	**1**	**1**	0	**0**	0	**0**	**0**	**0**

c:

1	**0**	**0**	**1**	1	**0**	0	**0**	**0**	**0**

d:

Şekil 2.7. Genetik Çaprazlama **(a:** Ebeveyn A, **b:** Ebeveyn B**, c:** Çocuk A, **d:** Çocuk B)

Şekil 2.7.a ve 2.7.b'de A ve B ebeveynleri verilmiştir. Bu bireylerin çaprazlanması sonucunda yeni bireyler oluşturulmuştur. A ve B ebeveynleri için çaprazlama noktaları rastgele seçilmiştir ve mavi renkle verilmiştir. A yeni bireyi için, B ebeveyninin çaprazlama noktasına kadar özellikler A ebeveyninden alınmıştır. Çaprazlama noktalarında ise B ebeveynindeki özellikler A yeni bireyine geçirilmiştir. B yeni bireyi içinse, A ebeveyninin çaprazlama noktasına kadar özellikler B ebeveyninden alınmıştır. A ebeveyninin çaprazlama noktasında ise A ebeveynindeki özellik, B yeni bireyine aktarılmıştır. Böylece yeni oluşan A ve B bireylerinin her ikisi de hem A hem de B ebeveynlerinden özellikler taşımaktadır.

Mutasyon operatörü, bulunamamış çözümlerin bulunmasına ya da daha önce popülasyondan atılmış iyi bireylerin tekrardan oluşturulmasına olanak sağlamaktadır.

Alt çözümlere takılınmasını engellemektedir. Mutasyon operatörü kullanılmayan bir algoritmada en iyi çözümün bulunması ancak popülasyon büyülüğünün yüksek tutulması halinde olmaktadır [31].

Mutasyon işleminde mutasyon değerine göre 1 değerleri 0 değerine, 0 değerleri ise 1 değerine çevrilmektedir. Şekil 2.8'de, Şekil 2.7.c'de verilen yeni oluşan A bireyinin mutasyona uğramış şekli ve Şekil 2.7.d'de verilen yeni oluşan B bireyinin mutasyona uğramış şekli verilmiştir.

1	**0**	**1**	**1**	0	**1**	**1**	**0**	**0**	**0**

a:

1	**0**	**0**	**0**	1	**0**	0	**0**	**0**	**0**

b:

Şekil 2.8. Mutasyon (**a:** Çocuk A Bireyinin Mutasyonu Sonucu Oluşan Birey, **b:** Çocuk B Bireyinin Mutasyonu Sonucu Oluşan Birey)

Şekil 2.8a ve Şekil 2.8b'de kırmızı renkle verilen bitler mutasyona uğramış bitlerdir. Şekil 2.8a'da 0 değeri mutasyon ile 1 değerine, Şekil 2.8b'de 1 değeri mutasyon ile 0 değerine çevrilmiştir.

Kontrol Parametreleri

Popülasyon büyüklüğü, çaprazlama oranı ve mutasyon oranı en önemli kontrol parametrelerinden biridir. Bu parametrelerin değerlerinin seçimi, algoritma performansını önemli derecede etkilemektedir.

Popülasyon büyüklüğünün küçük seçilmesi araştırma uzayının yetersiz örneklemesine ve iyi temsil edilememesine neden olarak, en iyi çözümü bulmak yerine alt sonuçlara ulaşılmasına neden olmaktadır. Popülasyon büyüklüğünün fazla büyük seçilmesi ise bir neslin gelişimin uzun sürede gerçekleşmesine neden olmaktadır. Bu nedenle popülasyon büyüklüğünün doğru seçilmesi gerekmektedir.

Çaprazlama oranının düşük seçilmesi, yeni bireylerin çok az sayıda yeni yapıdan oluşmasına neden olmakta ve çeşitliliği azaltmaktadır. Çaprazlama oranının yüksek seçilmesi ise araştırma uzayının çok hızlı bir şekilde araştırılmasına ve iyi çözümlerin üretilememesine neden olmaktadır [28].

Mutasyon oranının yüksek seçilmesi araştırma uzayının genişlemesine ve popülasyonun performansının düşmesine neden olacaktır. Mutasyon oranının düşük seçilmesi ise araştırma uzayının tamamen araştırılmasını engel olacak ve en iyi çözüm yerine iyi bir çözümün elde edilmesine neden olacaktır.

2.5.2. Genetik Algoritma Uygulamaları

Genetik Algoritmalar deneysel uygulamalarda, pratik uygulamalarda ve sınıflandırıcı sistemler uygulamalarında kullanılmaktadır. Genetik Algoritmalar, deneysel uygulamalara örnek olarak Gezgin Satıcı Problemi, Grafik Bölme Problemi ve Kör Knapsack Problemi'nde, pratik uygulamalara örnek olarak Nümerik Optimizasyon Problemleri, Çizelge Problemleri, Yerleşim Problemleri ve Görüntü İşleme uygulamalarında, sınıflandırıcı sistemler uygulamalarına örnek olarak da bir uzman sisteme ait olan bilgi tabanını oluşturan kuralları elde etme uygulamalarında kullanılmaktadır [28,32-33].

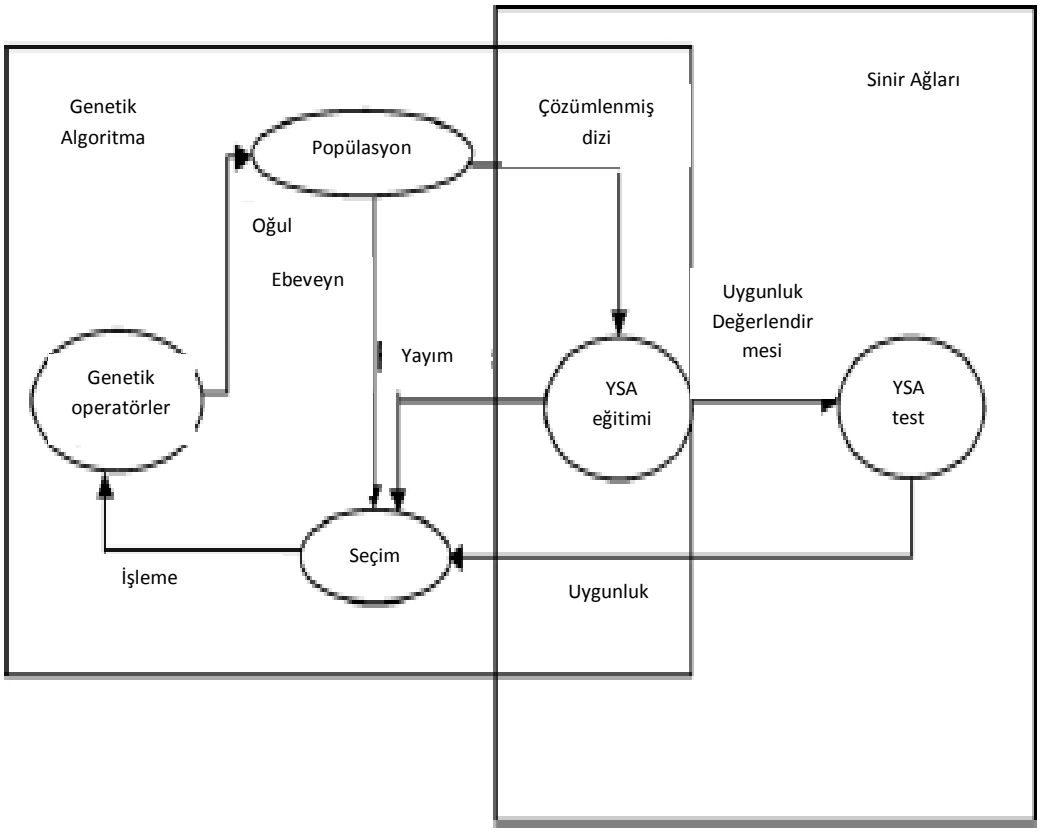
2.6. Genetik Yapay Sinir Ağları

Yapay Sinir Ağları dezavantajlarını en aza indirmek ve performansını artırmak amacıyla Yapay Sinir Ağları'nın bazı kısımlarına Genetik Algoritma uygulanmaktadır. Bu konuyla ilgili birçok çalışmalar yapılmıştır.

Hegazy ve arkadaşları [34], 1994 yılında Yapay Sinir Ağları parametrelerinin optimize edilmesi için Genetik Algoritma'nın kullanılmasını önermişlerdir. Van Rooij ve arkadaşları [35] ve Vonk ve arkadaşları [36] da yaptıkları çalışmayla Yapay Sinir Ağları mimarisinin ve ağırlıkların belirlenmesinde Genetik Algoritma'nın kullanılmasını önermişlerdir. 2001 yılında Huanga ve arkadaşları [37] ve 2006 yılında ise Chena ve Lina [38], daha doğru tahminlerin elde edilmesinde Yapay Sinir Ağları ve Genetik Algoritma'nın birleştirilmesinin çok etkili olduğunu göstermişlerdir [39].

Saemi ve arkadaşları [39], Taheri ve Mohebbi [40], Mohebbi ve arkadaşları [41] yaptıkları çalışmalarda Yapay Sinir Ağları ve Genetik Algoritmayı birleştirmişler, gizli katmanda bulunan nöron sayısının, momentum ve öğrenme oranlarının belirlenmesinde Genetik Algoritma kullanmışlardır. Tüm bu işlemleri ise NeuroSolutions programı aracılığı ile yapmışlardır. Bu şekilde ağ performansını artırma yoluna gitmişlerdir.

Şekil 2.9'da [39] genetik bir sinir ağının yapısı gösterilmiştir. Bu şekilde de görüldüğü gibi Genetik Algoritma ve Yapay Sinir Ağı bir arada kullanılmıştır. Popülasyondan seçim yapılmış, daha sonra seçilen üyelere genetik operatörler uygulanmıştır. Böylece yeni bireyler oluşmuştur. Bu bireyler popülasyona eklenmiştir. Daha sonra çözümleme yapılmış, çözümlenen bireyler Yapay Sinir Ağları ile eğitilmiştir. Ardından, eğitilen bireylerin uygunluk değerleri hesaplanmış, test edilmiş, uygunluk değeri yüksek olan bireyler tekrar seçilerek döngü devam ettirilmiştir.

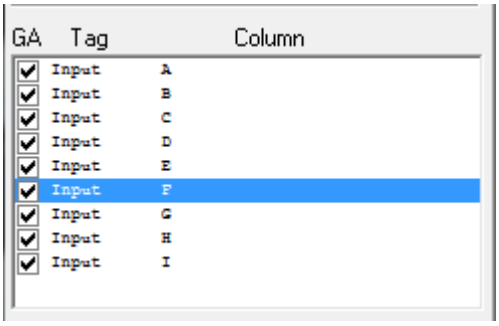


Şekil 2.9. Genetik Yapay Sinir Ağları

Bu tez çalışmasında tahminler Matlab programı araçları aracılığıyla Çok Katmanlı Algılayıcı kullanılması, daha sonra ise NeuroSolutions programı aracılığıyla Çok Katmanlı Algılayıcı ve Genetik Algoritma'nın birlikte kullanılması şeklinde iki yolla yapılmıştır. Tahminlerin doğruluk oranının arttırılmasında Yapay Sinir Ağları yapısının yani mimarisinin büyük bir önemi vardır. Örneğin; Yapay Sinir Ağları'nı oluşturan nöronların sayısının küçük tutulması, öğrenmeyi zorlaştırmakta ve öğrenme performansını düşürmektedir. Bu durum da ağın, yeni durumlar ya da örnekler karşısında makul değerler üretememesine neden olmaktadır. Ağın yapısında kullanılan nöron sayısının fazla olması da iyi bir durum değildir. Nöron sayısının fazla olması ağın eğitimi için geçen süre miktarını arttırabilmekte ve nöron sayısının arttırılmasıyla ağın ürettiği değerlerin doğruluk derecesinin artması beklenirken, tam tersi de olabilmektedir. Bu nedenle ağın yapısında kullanılan nöron sayısı büyük bir öneme sahiptir. Her problem ya da olay için gerekli olan nöron sayısı farklıdır ve

gerekli olan nöron sayısını hesaplayan bir formül de bulunmamaktadır. Yapay Sinir Ağları iyi birer çözüm sunmasına rağmen, en iyi çözümü garantileyememektedir. Yapay Sinir Ağları ile ulaşılan sonuçlar iyi olsa dahi en iyisi değildir. Örneğin; ağın yapısında bulunan bir bölüm değiştirildiğinde, ağ daha iyi sonuç üretebilmektedir. Ayrıca, ağa gösterilen girdi değerlerinden oluşturulan model için hangilerinin önemlilik değerinin fazla olduğu ve hangi girdi değerleriyle oluşturulan sistemin daha doğru sonuçlar üreteceği bilinememektedir. Bu nedenlerden dolayı bu tez çalışmasında, en iyi performansı gösteren nöron sayısının belirlenmesinde ve girdilerin seçilmesinde Genetik Algoritma'dan faydalanılmıştır.

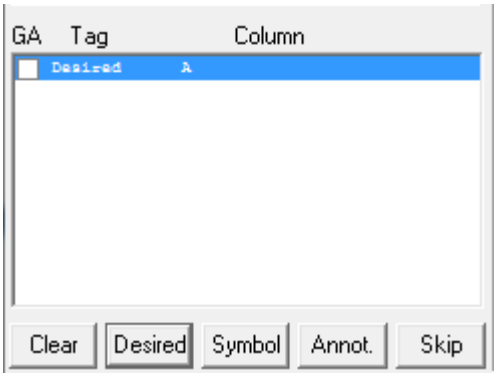
Bu tez çalışmasında, girdiler ve gizli katmanda bulunması gereken nöron sayıları genlerden oluşan kromozomlar şeklinde gösterilmiştir. Her kromozom, girdiler ve gizli katmanda bulunması gereken nöron sayılarını temsil eden genlerden oluşmuştur. Kromozomlar bir araya gelerek popülasyonu oluşturmuştur. Popülasyon büyüklüğü 50 olarak belirlenmiştir. 100 adet yeni nesil oluşturulmuştur. Tüm girdi değerlerine Şekil 2.10'da gösterildiği gibi genetik algoritma uygulanmış, eğitimin birden çok uygulanmasına dikkat edilmiştir.



Şekil 2.10. Tüm Girdi Değerlerine Genetik Algoritma Uygulanması

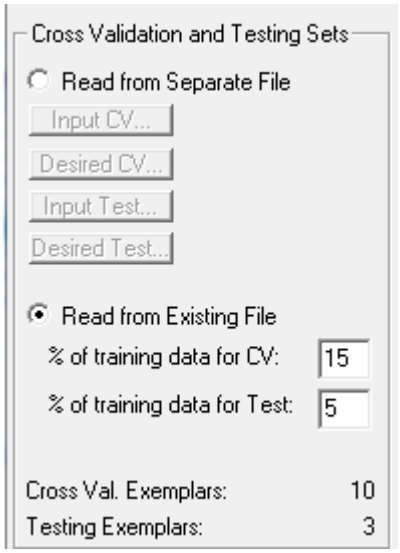
Şekil 2.10'da yazan GA ifadesi, Genetik Algoritma anlamında olup, Genetik Algoritma uygulanacak olan girdilerin seçiminde kullanılmaktadır. Şekil 2.10'da gösterildiği gibi, tüm girdi değerlerine Genetik Algoritma'nın uygulanması istenmiştir.

Ardından, Şekil 2.11'de gösterildiği gibi girdi değerlerine karşılık gelen çıktılar sisteme yüklenmiş, çıktı değerlerine Genetik Algoritma uygulanamayacağı için bu değerlere Genetik Algoritma uygulanmamıştır.



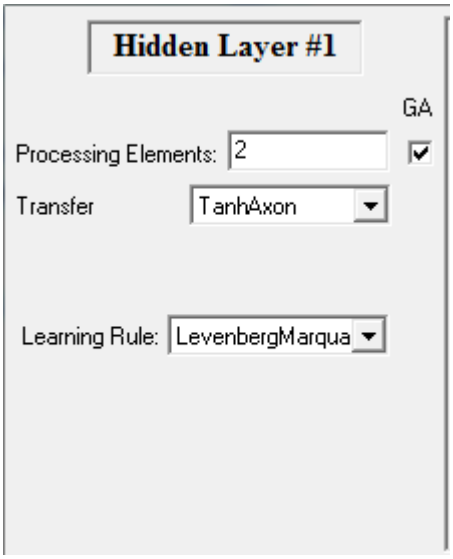
Şekil 2.11. Çıktı Değerlerinin Sisteme Yüklenmesi

Eğitim setinde yer alan projelerden rastgele %15'i çapraz doğrulama verisi, %5'i ise test için ayrılmıştır. Sonuç olarak 10 adet proje çapraz doğrulama, 3 adet proje de test için ayrılmıştır. Bu durum, Şekil 2.12'de gösterilmiştir.

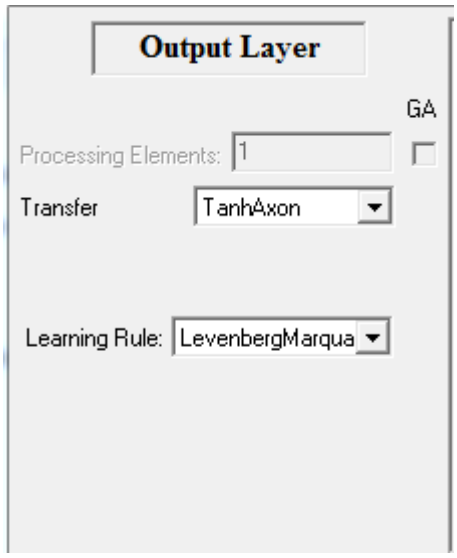


Şekil 2.12. Çapraz Doğrulama ve Test Verileri Sayısının Belirlenmesi

Gizli katman sayısı 1 olarak seçilmiştir. Gizli katmanda bulunan işleme elemanları sayısı 2 olarak seçilirken, bu elemanlara Genetik Algoritma uygulanmıştır. Hem gizli katman, hem de çıktı katmanı için transfer fonksiyonu olarak TanhAxon, öğrenme kuralı olarak Levenberg-Marquardt kuralı uygulanmıştır. Bu durumlar, Şekil 2.13 ve Şekil 2.14'de gösterilmiştir.



Şekil 2.13. Gizli Katman Özelliklerinin Seçimi



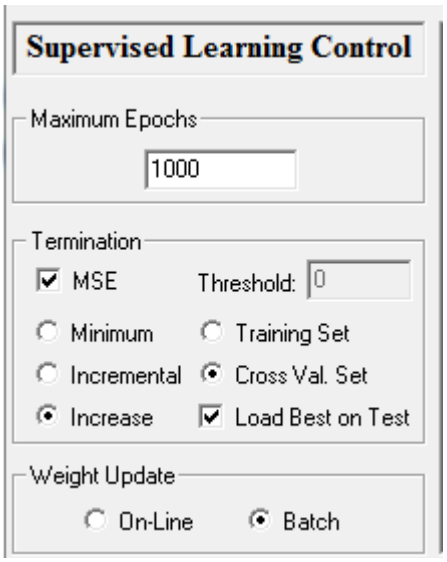
Şekil 2.14. Çıktı Katmanı Özelliklerinin Seçimi

Tekrar üreme, çaprazlama ve mutasyon operatörleri NeuroSolution tarafından otomatik olarak uygulanmıştır. Sistem eğitilmiş, yeni nesiller oluşmuştur. Eğitim normal koşullar altında 1000 iterasyon boyunca devam etmiştir. Yalnız, aşırı eğitimin önüne geçmek adına, çapraz doğrulama hata oranında artma olduğunda eğitimim durdurulması planlanmıştır. Aşırı eğitim, ağın yeni sonuçlar üretemeyip, eski sonuçları hatırlamasıdır. Bu hata oranı ise Eşitlik 2.23'te verilen ortalama kare hata değeri hesaplanarak bulunmuştur.

$$Ortalama\ Kare\ Hata = \sum_{i=1}^{K}(O_i - T_i)2\ /k \tag{2.23}$$

Eşitlik 2.23'te verilen O_i ifadesi eğitim setinde bulunan beklenen çıktı ya da çapraz doğrulama verisi, T_i eğitim veri setinden ağın çıktısı ya da çapraz doğrulama verisi, k ise veri sayısıdır [41].

Ağırlıkların güncellenmesi ise tüm eğitim seti ağa gösterildikten sonra yapılmıştır. Tüm öğrenme kontrolleri Şekil 2.15'te verilmiştir.



Şekil 2.15. Denetimli Öğrenme Kontrollerinin Belirlenmesi

2.7. Levenberg–Marquardt Algoritması

Levenberg–Marquardt algoritması, lineer olmayan en küçük kareler problemlerinin çözümünde kullanılmaktadır. Bu algoritma, minimizasyon yöntemlerinden olan dik iniş metodu ve Gauss-Newton metodunun birleşimidir. Levenberg–Marquardt algoritması, parametreler optimal değerlerden uzak olduğunda dik iniş algoritması gibi, yakın olduğunda ise Gauss-Newton metodu gibi davranmaktadır. Parametre güncellemeleri de bu iki metodun parametre güncelleme yöntemiyle yapılmaktadır [42].

Levenberg-Marquardt algoritması lineer olmayan fonksiyonların kareleri toplamı olarak olarak verilen bir $F(x)$ fonksiyonunun minimum değerini bulmayı sağlamaktadır. $F(x)$ fonksiyonu Eşitlik 2. 24'te verilmiştir [43].

$$F(x) = 1/2 \sum_{i=1}^{m} [f_i(x)]^2 \tag{2.24}$$

$f_i(x)$ 'in Jacobian değeri $J_i(x)$ ile gösterildiğinde, Levenberg-Marquardt algoritması verilen p çözümü yönünde Eşitlik 2.25'te arama yapmaktadır.

$$(J_k^T J_k + \lambda_k I) p_k = -J_k^T f_k \tag{2.25}$$

2.25 eşitliğinde λ_k değeri negatif olmayan skaler bir büyüklük, I ise kimlik matrisidir.

n adet eşitlik ve $x_1, x_2, \ldots . x_n$ şeklinde n adet veriden oluşan

$$y = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}$$

şeklinde bir dizi verilmektedir. Jacobian matrisi ise Eşitlik 2.26'da gösterildiği gibi hesaplanmaktadır [44].

$$J(x_1, x_2, \ldots . x_n) \equiv \begin{bmatrix} \partial y_1/\partial x_1 & \cdots & \partial y_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial y_n/\partial x_n & \cdots & \partial y_n/\partial x_n \end{bmatrix} \quad (2.26)$$

Kimlik matrisi ise tüm X vektörleri için $I(X) \equiv X$ şeklinde tanımlanmakta olup,

$$I \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

olarak tanımlanabilmektedir.

2.8. Performans Değerlendirme Kriterleri

Bu tez çalışmasında tahminler Çok Katmanlı Algılayıcı ve Çok Katmanlı Algılayıcı ile Genetik Algoritma'nın birlikte kullanılması şeklinde olmak üzere iki yolla yapılmıştır. Performans değerlendirme kriteri olarak Ortalama Bağıl Hata ve Pred(25) değerleri kullanılmıştır. Ortalama Bağıl Hata değerinin hesaplanabilmesi için ilk önce her bir proje için Eşitlik 2.27'de gösterildiği gibi hesaplanan Bağıl Hata değerleri hesaplanmış, ardından Eşitlik 2.28'de verildiği gibi hesaplanan Ortalama Bağıl Hata değeri bulunmuştur. Pred(25) değeri formülü ise Eşitlik 2.29'da verilmiştir. Her iki deney 10'ar kez yapılmış, en az hata değerini veren deney karşılaştırma için seçilmiştir.

$$Bağıl\ Hata_i = |\ Gerçek\ Efor_i - Tahmini\ Efor_i\ |\ /Gerçek\ Efor_i \quad (2.27)$$

2.27 Eşitliği'nde bulunan *i* ifadesi proje sırasını vermektedir. Her bir proje için Bağıl Hata değerleri hesaplanmıştır.

$$Ortalama\ Bağıl\ Hata = \frac{1}{N}\sum_{i=1}^{N} Bağıl\ Hata_i \quad (2.28)$$

2.28 Eşitliği'nde verilen N ifadesi toplam proje sayısıdır. i ise projenin sırasıdır. Herbir deney için Bağıl Hata değerleri kullanılarak Ortalama Bağıl Hata değerleri hesaplanmıştır.

$$Pred(A) = d/N \tag{2.29}$$

2.29 Eşitliği'nde verilen N toplam proje sayısını, d ise Bağıl Hata değeri $\%A$ değerine eşit ya da küçük olan proje sayısını göstermektedir. Bu tez çalışmasında A değeri, diğer birçok efor tahmini çalışmalarında seçildiği gibi *25* olarak seçilmiştir. Bazı çalışmalarda A değeri *30* olarak da seçilebilmektedir.

2.9. Kullanılan Programlar

Bu tez çalışmasında Weka 3.6.9, MatlabR2009b ve Neurosolutions 5 programlarından faydalanılmıştır.

Veri seti olarak EK1'de de gösterilen Desharnais veri seti kullanılmıştır. Bu veri seti, her biri 12 farklı özelliğe sahip olan ve toplamda 81 adet yazılım projesinden oluşan bir veri setidir. Veri seti, Kanadalı bir yazılım evi tarafından geliştirilmiştir [45].

Veri setini inceleme ve analiz etme işlemleri Weka programı yardımıyla yapılmıştır. Weka, makine öğrenme algoritmalarından oluşan bir koleksiyondur. Veri ön işleme, sınıflandırma, regresyon, kümeleme, görselleştirme gibi araçlar içermektedir [46]. Bu tez çalışmasında ise Desharnais veri seti Weka programı ile açılmıştır ve bu şekilde veri setinde bulunan verilere ulaşılmıştır. Ulaşılan veriler Matlab ve NeuroSolution programları aracılığıyla işlenmiştir.

Matlab aracılığıyla veri analizi yapılabilmekte, algoritmalar geliştirilebilmekte, modeller ve uygulamalar yapılabilmektedir. Sinyal işleme, test etme ve ölçme, görsel ve video işleme gibi birçok işlemde kullanılabilmektedir [47].

Neurosolution bir sinir ağları yazılım paketidir. Yapay zeka ve öğrenme algoritmaları uygulamalarını içeren bir ara yüze sahiptir. Küme analizleri, tahminler ve tıbbi sınıflandırma gibi daha birçok işlemde kullanılmaktadır [48].

3. ARAŞTIRMA BULGULARI

Bu tez çalışmasında Desharnais veri seti kullanılarak efor tahminlerinde bulunulmuştur. Desharnais veri seti EK 1'de verilmiştir.

Efor değerleri hesaplanan 10 adet projenin girdi değerleri Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Efor Değerleri Hesaplanan Projelerin Girdileri

No.	Takım Tecrübesi	Yönetici Tecrübesi	Uzunluk	İşlemler	Varlıklar	Ayrlanmamış noktalar	Ayarlar	Ayarlanmış noktalar	Dil
1	1.0	4,0	12,0	253,0	52,0	305,0	34,0	302,0	1
2	0.0	0,0	4,0	197,0	124,0	321,0	33,0	315,0	1
3	4.0	4,0	1,0	40,0	60,0	100,0	18,0	83,0	1
4	0.0	0,0	5,0	200,0	119,0	319,0	30,0	303,0	1
5	0.0	0,0	4,0	140,0	94,0	234,0	24,0	208,0	1
6	0.0	0,0	4,0	97,0	89,0	186,0	38,0	192,0	1
7	2.0	1,0	9,0	119,0	42,0	161,0	25,0	145,0	2
8	1.0	2,0	13,0	186,0	52,0	238,0	25,0	214,0	1
9	3.0	1,0	12,0	172,0	88,0	260,0	30,0	247,0	1
10	3.0	4,0	4,0	78,0	38,0	116,0	24,0	103,0	1

Çok Katmanlı Algılayıcı ve Genetik Çok Katmanlı Algılayıcı modellerinin her ikisi için de "Efor" özelliği çıktı, geriye kalan özellikler ise sisteme girdi olarak verilmiş, ağın eğitilmesi amaçlanmıştır. Ağın, kendisine gösterilen örneklere yani tüm girdi değerlerine ve bu girdi değerlerine karşılık gelen çıktı değerlerine bakarak bir genelleme yapabilmesi, sonuç çıkarabilmesi istenmiştir. Bu amaçla sistem, belirli bir iterasyon sayısınca ya da Çapraz Doğrulama Ortalama Bağıl Hata değerindeki değişime göre eğitilmiştir. Eğitilirken, iterasyon sayısı ve Çapraz Doğrulama Ortalama Bağıl Hata değerindeki değişim eğitimi durdurma kriterleri olarak seçilmiş,

fazla eğitimim önüne geçilmek istenmiştir. Sistemin fazla eğitilmesi üretilen tahminleri olumsuz yönde etkileyen bir faktördür. Sistem fazla eğitildiğinde tahmin performansı düşmektedir. Çünkü sistem, eski sonuçları ezberlemiştir ve kendisine sorulan sorulara yeni çözümler üretmek yerine, bulmuş olduğu eski sonuçları söylemektedir. Eğitimden sonra, Çizelge 3.1'de verilen ve 10 adet projeye ait olan özellikler ağa girdi olarak verilmiş, ağdan bu girdi değerlerine karşılık gelen "Efor" değerlerinin yani çıktı değerlerinin üretilmesi istenmiştir. Bu 10 adet projeye ait olan gerçek "Efor" değerleri Çizelge 3.2'de verilmiştir.

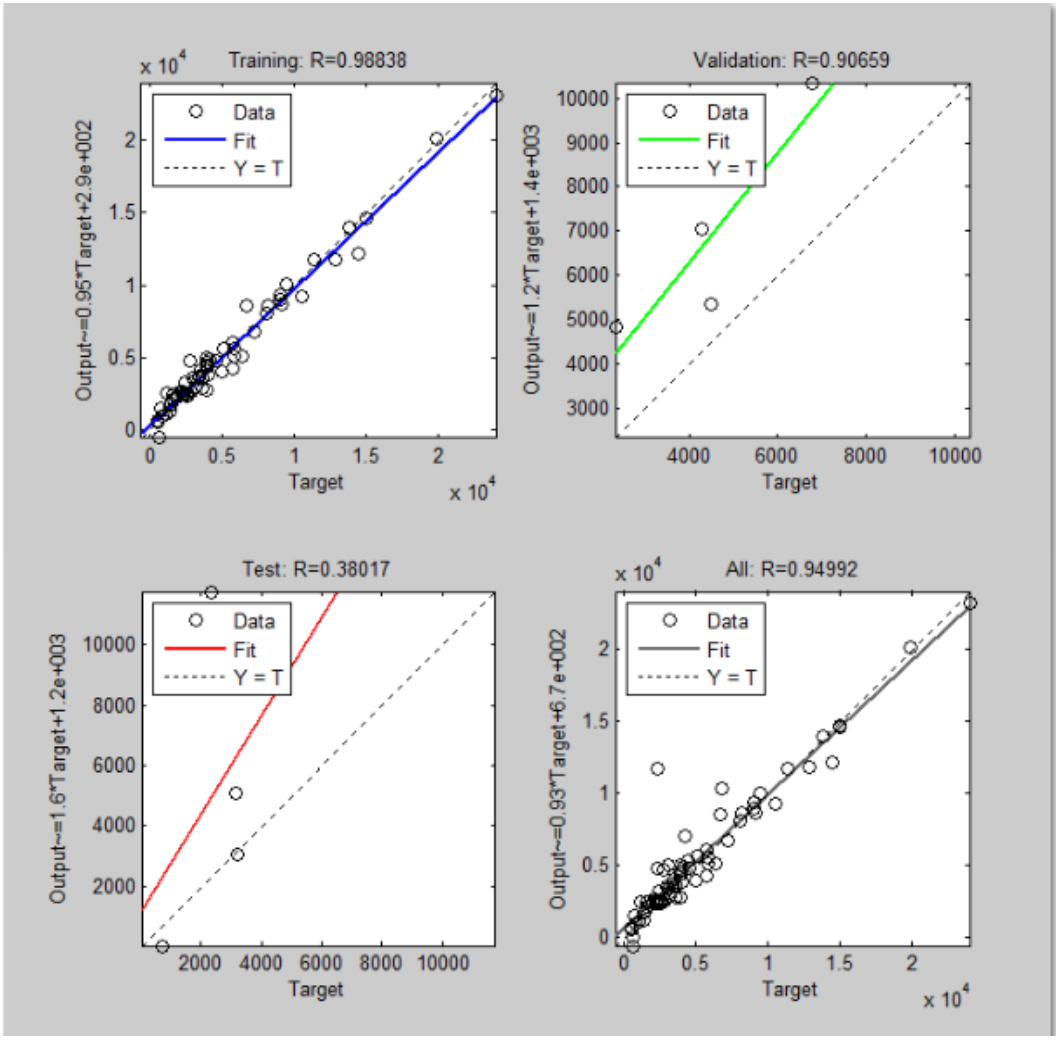
Çizelge 3.2. Gerçek Efor Değerleri

No	Efor
1	5152,0
2	5635,0
3	805,0
4	3829,0
5	2149,0
6	2821,0
7	2569,0
8	3913,0
9	7854,0
10	2422,0

2014 yılında Seref ve Barisci [49] tarafından yapılan çalışmada yazılım efor tahmini için veri seti olarak Desharnais ve NASA93 veri setleri kullanılmış, Çok Katmanlı Algılayıcı ve Adaptif Sinirsel Bulanık Çıkarım Sistemi ile tahminlerde bulunulmuştur. Adaptif Sinirsel Bulanık Çıkarım Sistemi performansının daha iyi olduğu gözlenmiştir. Bu tez çalışmasında ise Çok Katmanlı Algılayıcı modeline Genetik Algoritma eklenmesiyle performans artırılması yoluna gidilmiş, önemli derecede başarı sağlanmıştır.

3.1. Çok Katmanlı Algılayıcı ile Elde Edilen Sonuçlar

Çok Katmanlı Algılayıcı için eğitim algoritması olarak Levenberg-Marquart geri besleme algoritması kullanılmış, performans için Kare Hata Ortalaması alınmış, veri dağılımı ise rastgele yapılmıştır. Elde edilen regresyon grafikleri Şekil 3.1'de verilmiştir.



Şekil 3.1. Regresyon Grafikleri

10 adet projeye ait olan gerçek efor değerleri, ağın ürettiği tahmin değerleri, her bir projeye ait olan Bağıl Hata değerleri Çizelge 3.3'de verilmiştir. Ortalama Bağıl Hata;

$$Ortalama\ Bağıl\ Hata = \frac{1}{10} \times (0{,}04 + 0{,}44 + 0{,}90 + 0{,}12 + 0{,}51 + 0{,}08 + 0{,}61 + 0{,}11 + 0{,}48 + 0{,}26)$$
$$= 0{,}355$$

şeklinde bulunmuştur.

Bağıl Hata değeri *0,25* değerine eşit ya da bu değerden küçük olarak hesaplanan 4 adet proje bulunmaktadır. Bu nedenle, Pred(25) değeri;

$$Pred(25) = 4/10$$
$$= 0{,}40$$

şeklinde bulunmuştur. Bulunan Ortalama Bağıl Hata ve Pred(25) değerleri Çizelge 3.4'te de gösterilmiştir.

Çizelge 3.3. Çok Katmanlı Algılayıcı İle Ulaşılan Sonuçlar

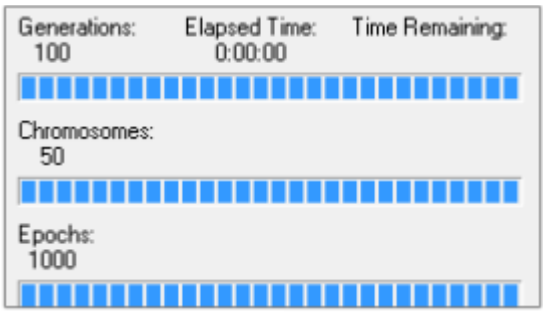
Gerçek Efor	Tahmin Edilen Efor	Bağıl Hata
5152	5376,6	0,04
5635	3175,6	0,44
805	1530,6	0,90
3829	3371,4	0,12
2149	3241,5	0,51
2821	3056,1	0,08
2569	1004,1	0,61
3913	4354,5	0,11
7854	4069,2	0,48
2422	1785,4	0,26

Çizelge 3.4. Performans Değerlendirme Kriterleri Sonuçları

Ortalama Bağıl Hata	Pred(25)
0,355	0,40

3.2. Genetik Çok Katmanlı Algılayıcı ile Elde Edilen Sonuçlar

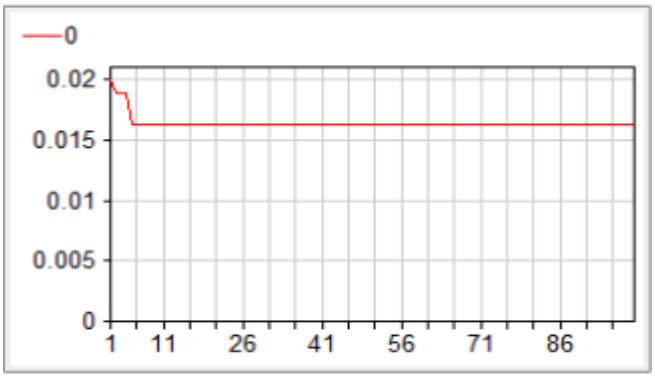
Çok Katmanlı Algılayıcı ve Genetik Algoritma'nın birlikte kullanılmasıyla tahmin hata oranlarının en aza indirilmesi ve performansın artırılması amaçlanmıştır. 50 adet kromozom belirlenmiş, 1000 iterasyon boyunca eğitim sağlanmış ve 100 adet yeni nesil oluşturulmuştur. Bu durum Şekil 3.2'de gösterilmiştir.



Şekil 3.2. Kullanılan Kromozom, İterasyon ve Nesil Sayıları

Şekil 3.2'de de gösterildiği gibi, nesil sayısı 100, kromozom sayısı 50, iterasyon sayısı 1000 olarak belirlenmiştir ve eğitim gerçekleştirilmiştir. 100. nesilden sonra, Neurosolution en iyi nesli belirlemiş ve belirlenen bu nesil kullanılarak sisteme yeni girdiler verilmiş, tahminler oluşturulmuştur. Daha sonra sistem tekrar eğitilmiş, sisteme girdiler gösterilerek, ağın tahminlerde bulunması sağlanmıştır ve tahminler kaydedilmiştir. Bu işlemler 10 kez tekrarlanmış, bulunan en iyi nesil ve en iyi tahminler Çok Katmanlı Algılayıcı ile karşılaştırma için seçilmiştir.

Deney sonucunda her nesil için en iyi uygunluk değerine sahip olan kontrol genetiği ortalama kare hata değeri şekli Şekil 3.3'te verilmiştir. Şekil 3.3'te x ekseni nesli, y ekseni ise ortalama kare hata değerlerini göstermiştir.



Şekil 3.3. En İyi Uygunluk Değerlerinin Ortalama Kare Hataları

Deney sonucunda elde edilen tahmin değerleri, gerçek efor değerleri ve her bir proje için hesaplanan Bağıl Hata değerleri Çizelge 3.5'te verilmiştir.

Çizelge 3.5. Genetik Çok Katmanlı Algılayıcı İle Ulaşılan Sonuçlar

Gerçek Efor	Tahmin Edilen Efor	Bağıl Hata
5152	5691,7	0,10
5635	3142,9	0,44
805	1569,9	0,95
3829	3079,2	0,20
2149	1806,0	0,16
2821	1633,2	0,42
2569	2103,3	0,18
3913	3603,3	0,08
7854	3450,9	0,56
2422	2093,3	0,14

Ortalama Bağıl Hata ve Pred(25) değerleri Çizelge 3.6'da verilmiştir.

Çizelge 3.6. Performans Değerlendirme Kriterleri Sonuçları

Ortalama Bağıl Hata	Pred(25)
0,323	0,60

Çizelge 3.6'da verilen Ortalama Bağıl Hata değeri;

$$Ortalama\ Bağıl\ Hata = \frac{1}{10}x(0{,}10 + 0{,}44 + 0{,}95 + 0{,}20 + 0{,}16 + 0{,}42 + 0{,}18 + 0{,}08 + 0{,}56 + 0{,}14)$$
$$= 0{,}323$$

olarak bulunmuştur.

Çizelge 3.6'da verilen Pred(25) değeri ise, Bağıl Hata değeri *0,25* 'e eşit ya da bu değerden daha küçük olan proje sayısı 6 olduğundan ve toplam proje sayısı *10* olduğundan, Pred(25) değeri;

$$Pred(25) = 6/10$$
$$= 0{,}60$$

olarak bulunmuştur.

4. SONUÇLAR VE TARTIŞMA

Bu tez çalışmasında yazılım efor tahmini Çok Katmanlı Algılayıcı ve Çok Katmanlı Algılayıcı ile Genetik Algoritma'nın hibrit bir biçimde kullanılması olmak üzere iki farklı yolla yapılmıştır. Veri seti olarak Desharnais ver seti kullanılmıştır. Her bir proje için verilen "Takım Tecrübesi", "Yönetici Tecrübesi", "Uzunluk", "Temel Mantıksal İşlemlerin Sayısı", "Varlık Sayısı", "Ayarlanmamış Noktalar", "Ayarlar", "Ayarlanmış Noktalar", "Kullanılan Programlama Dili" özellikleri girdi olarak kullanılırken, "Efor" özelliği çıktı olarak kullanılmıştır. Performans değerlendirme kriteri olarak Ortalama Bağıl Hata ve Pred(25) değerleri seçilmiştir.

Çok Katmanlı Algılayıcı için, eğitim seti için ayrılan 71 adet projeden 63 adet proje eğitim için kullanılırken, 8 adet proje onaylama ve test etme için kullanılmıştır. Sistem 10 adet gizli nöron kullanılarak oluşturulmuş, Levenberg-Marquardt Geri Yayılım Algoritması kullanılarak 1000 iterasyon sayısınca eğitilmiştir. Daha sonra, eğitim setinin dışında kalan 10 adet projenin eforları tahmin edilmiştir. Sistemin eğitilmesi 10 kez yapılmıştır.

Çok Katmanlı Algılayıcı ve Genetik Algoritma'nın birlikte kullanılması yönteminde, ağ için gerekli olan en iyi girdi değerlerinin belirlenmesinde ve işleme elemanlarının optimize edilmesinde Genetik Algoritma'dan yararlanılmıştır. Ağ için gerekli olan girdi değerleri ve işleme elemanları sayısı genlerle ifade edilirken, genler birleşerek kromozomları, kromozomlar birleşerek popülasyonu oluşturmuştur. Kromozom sayısı 50, nesil sayısı ise 100 olarak belirlenmiştir. Ağın eğitimi için 71 adet proje seçilmiş ve eğitim setinde yer alan 10 adet proje çapraz doğrulama için kullanılırken, 3 adet proje de ağ performansını test etmek için kullanılmıştır. Gizli katman sayısı 1 olarak seçilirken, işleme elemanları sayısı 2 olarak seçilmiş, işleme elemanlarına Genetik Algoritma uygulanmıştır. Gizli katman ve çıktı katmanının her ikisinde de transfer fonksiyonu olarak TanhAxon, öğrenme kuralı olarak Levenberg-Marquardt kuralı uygulanmış, ağ eğitilmiştir. Eğitimi durdurma kriteri olarak Çapraz Doğrulama Ortalama Kare Hata değeri ve iterasyon sayısı düşünülmüştür. Çapraz Doğrulama

Ortalama Kare Hata değerinde artma başladığında eğitim durdurulmuştur. Bu şart sağlanmadığında ise sistem 1000 iterasyon boyunca eğitilmiştir. Çapraz Doğrulama ile sistemin fazla eğitilmesinin önüne geçilmek istenmiştir. Eğitimden sonra eğitim setinin dışında tutulan 10 adet projeye ait olan girdiler ağa gösterilmiş, bu girdilere ait olan çıktı değerleri yani yazılım efor değerleri tahmin edilmiştir.

Çok Katmanlı Algılayıcı ile tahmin edilen efor değerleri Çizelge 3.3'te, Çok Katmanlı Algılayıcı ile Genetik Algoritma'nın birlikte kullanılmasıyla tahmin edilen efor değerleri Çizelge 3.5'te verilmiştir. Bu iki yönteme ait olan performans değerlendirme kriterleri sonuçları ise Çizelge 3.4 ve Çizelge 3.6'da verilmiştir.

Çok Katmanlı Algılayıcı ve Çok Katmanlı Algılayıcı'nın Genetik Algoritma ile hibrit bir şekilde kullanılması sonucu oluşan sinir ağlarının performans karşılaştırması özet haliyle Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Performans Sonuçları Özeti

Veri Seti	Çok Katmanlı Algılayıcı		Çok Katmanlı Algılayıcı ve Genetik Algoritma	
	Ortalama Bağıl Hata	Pred(25)	Ortalama Bağıl Hata	Pred(25)
Desharnais	0,355	0,40	0,323	0,60

Çizelge 4.1'de de görüldüğü gibi, Çok Katmanlı Algılayıcı'ya Genetik Algoritma eklenmesiyle efor tahmini performansında önemli derecede artış olmuştur. Ağın eğitimi için en iyi girdi değerlerinin tespitinde ve işlem elemanlarının optimize edilmesinde Genetik Algoritma'nın kullanılması, performans değerlendirme kriteri olarak seçilen Ortalama Bağıl Hata ve Pred(25) değerlerinin her ikisini de olumlu yönde etkilemiştir. Ortalama Bağıl Hata değeri Çok Katmanlı Algılayıcı modelinde 0,355 olarak bulunurken, bu değer Çok Katmanlı Algılayıcı'ya Genetik Algoritma eklendiğinde 0,323'e düşmüştür. Çok Katmanlı Algılayıcı modelinde 0,40 olarak

bulunan Pred(25) değeri, bu modele Genetik Algoritma eklendiğinde 0,60'a yükselmiştir. Genetik algoritma yardımıyla model için en iyi girdilerin ve işlem elemanları sayısının belirlenmesi ağın mimarisini ve tahmin etme yeteneğini güçlendirmiş, gerçek değerlere daha yakın sonuçların elde edilmesini sağlamıştır.

KAYNAKLAR

[1] Garcia-Diaz, N., Lopez-Martin, C., Chavoya, A., A comparative study of two fuzzy logic models for software development effort estimation. Procedia Technology 7 (2013), Volume 7, 305 – 314, 2013.

[2] Lopez-Martin, C., Yanez-Marquez, C., Gutierrez-Tornes, A., A Fuzzy Logic Model Based Upon Reused and New&Changed Code for Software Development Effort Estimation at Personal Level. Proceedings of the 15th International Conference on Computing (CIC'06), 0-7695-2708-6/06, 2006.

[3] Lopez-Martin, C., A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. Volume 11, Issue 1, pages 724-732, 2011.

[4] Lopez-Martin, C., Isaza, C., Chavoya, A., Software development effort prediction of industrial projects applying a general regression neural network. Empir Software Eng (2012), 17:738‑756, 2012.

[5] Oliveira, A. L.I., Braga, P. L., Lima, R.M.F., Cornélio, M. L., GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Volume 52, Issue 11, Pages 1155-1166, 2010.

[6] Muzaffar, Z., Ahmed, M. A., Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems. Information and Software Technology, Volume 52, Issue 1, Pages 92-109, 2010.

[7] Malhotra, R., Jain A., Software Effort Prediction using Statistical and Machine Learning Methods. International Journal of Advanced Computer Science and Applications, Vol. 2, No.1, Pages 145-152, 2011.

[8] Heiat A., Comparison of artificial neural network and regression models for estimating software development effort. J Inform Software Tech, Elsevier 44(15):911–922, 2002.

[9] Oliveira A. L. I., Estimation of software project effort with support vector regression. Neurocomputing, Elsevier 69:1749–1753, 2006.

[10] Vinay K. K. , Ravi V., Carr M., Raj K. N., Software development cost estimation using wavelet neural networks. J Syst Software, Elsevier 81:1853–1867, 2008.

[11] Park S., Baek S., An empirical validation of a neural network model for software effort estimation. J Expert Syst Appl, Elsevier 35:929–937, 2008.

[12] De Barcelos T. I. F., Simies da Silva J. D., Sant Anna N., An investigation of artificial neural networks based prediction systems in software project management. J Syst Software, Elsevier 81:356–367, 2008.

[13] El-Sebakhy E. A., Functional networks as a novel data mining paradigm in forecasting software development efforts. J Expert Syst Appl, Elsevier 38:2187–2194, 2011.

[14] Sayyad Shirabad, J. and Menzies, T.J. (2005) The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada . Available: http://promise.site.uottawa.ca/SERepository

[15] Öztemel E., Yapay Sinir Ağları. Papatya Yayuncılık.

[16] Yaici, W., Entchev, E., Performance prediction of a solar thermal energy system using artificial neural networks. Applied Thermal Engineering, Volume 73, Issue 1, , Pages 1346–1357, 2014.

[17] Yıldırım T., Yapay Sinir Ağları ve Nöral Hesaplama. http://www.google.com.tr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&cad=rja&uact=8&sqi=2&ved=0CFEQFjAJ&url=http%3A%2F%2Fwww.yildiz.edu.tr%2F~nicoskun%2FYSA_genel%2520bilgi.ppt&ei=N1NeVOSKH5T7avqQgsgM&usg=AFQjCNENFK_fLXL-52kNe7E_IjBuj6fB1A&sig2=yVTmEiFKl_sBlfoHdFplUQ&bvm=bv.79189006,d.d2s (Erişim tarihi: 08.11.2014)

[18] Uğurlu B., İlk Yapay Sinir Ağları. http://www.google.com.tr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=0CDkQFjAF&url=http%3A%2F%2Fmembers.comu.edu.tr%2Fboraugurlu%2Fcourses%2Fai%2FPresentation8.ppt&ei=CstfVNHuGo_gaP7RgIAB&usg=AFQjCNEd-yMzZi0ZzTNRN3XZf-bv6-M7aQ&sig2=0Te224VMuJXzr7_fs_U9lw&bvm=bv.79189006,d.d2s (Erişim tarihi: 14.11.2014)

[19] Hammer B., Neural Networks. Chapter 15, Academic Press Library in Signal Processing, Volume 1, Pages 817-855, 2014.

[20] Feng, J. F., Tirozzi B., Learning in a Higher-Order Simple Perceptron. Mathematical and Computer Modelling 30 (1999) 217-223, 1999.

[21] Minsky M., Paapert S., Perceptrons, The MIT Press, 1969.

[22] Rumelhart D. E., Hinton, G.E., Williams, R. J., Learning representations by backpropogation errors. Nature, vol.323, pp:533-536, 1986.

[23] Khadem, A., Zadeh, G. A. H., Estimation of direct nonlinear effective connectivity using information theory and multilayer perceptron. Journal of Neuroscience Methods, Volume 229, Pages 53-67, 2014.

[24] Huang, C. J., Clustered defect detection of high quality chips using self-supervised multilayer perceptron. Expert Systems with Applications, Volume 33, Issue 4, Pages 996-1003, 2007.

[25] Dalkıran, İ., Danışman, Kenan., Artificial neural network based chaotic generator for cryptology. Turk J Elec Eng & Comp Sci, Vol.18, No.2, 2010.

[26] Stergiou, C., What is a Neural Network? http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/cs11/article1.html (Erişim tarihi: 05.12.2014)

[27] Holland, J. H., Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.

[28] Karaboğa, D., Yapay Zeka Optimizasyon Algoritmaları. Nobel Yayın Dağıtım.

[29] Oliveira, R. A., Junior, M. F. M., Menezes, R. F. A., Application of genetic algorithm for optimization on projects of public illumination. Electric Power Systems Research, Volume 117, Pages 84-93, 2014.

[30] Yang, X. S., Chapter 5 - Genetic Algorithms. Nature-Inspired Optimization Algorithms, Pages 77-87, 2014.

[31] De Jong, K. A., Analysis of the Behavior of a Class of Genetic Adaptive Systems,Ph.D. Dissertation, Department of Computer and Communication Science,University of Michigan, Ann Arbor, MI, 1975.

[32] Reeves, C.R., Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, Maidenhead, UK, 1995.

[33] Davis, L., Handbook of Genetic Algorithms, Van Nonstrand Reinhold. New York, NY, 1991.

[34] Hegazy, T., Moselhi, O., Fazio, P., Developing practical neural network applications using back-propagation. Microcomput. Civ.Eng. 9, 145–159, 1994.

[35] Van Rooij, A. J. F., Jain, L.C., Johnson, R.P., Neural Network Training Using Genetic Algorithms. World Scientific Publishing Co. Pvt. Ltd, Singapore, 1996.

[36] Vonk, E., Jain, L.C., Johnson, R.P., Automatic Generation of Neural Network Architecture Using Evolutionary Computation. World Scientific Publishing Co. Pvt. Ltd, Singapore, 1997.

[37] Huanga, Y., Gedeonb, T., Wongc, P., An integrated neural-fuzzy- genetic algorithm usinghyper-surface membership functionstopredict permeability in petroleum reservoirs. J. Pet. Sci. Eng. 14, 15–21, 2001.

[38] Chena, C., Lina, L., A committee machine with empirical formulas for permeability prediction. Comput. Geosci. 32, 485–496, 2006.

[39] Saemi, A., Ahmadi, M., Varjani, A. Y., Design of neural networks using genetic algorithm for the permeability estimation of the reservoir. Journal of Petroleum Science and Engineering, Volume 59, Issues 1–2, Pages 97-105, 2007.

[40] Taheri, M., Mohebbi, A., Design of artificial neural networks using a genetic algorithm to predict collection efficiency in venturi scrubbers. Journal of Hazardous Materials, Volume 157, Issue 1, Pages 122-129, 2008.

[41] Mohebbi, A., Taheri, M., Soltani, A., A neural network for predicting saturated liquid density using genetic algorithm for pure and mixed refrigerants. International Journal of Refrigeration, Volume 31, Issue 8, Pages 1317-1327, 2008.

[42] Henri P. Gavin, http://people.duke.edu/~hpgavin/ce281/lm.pdf (Erişim tarihi: 10.01.2015)

[43] Anonim, http://mathworld.wolfram.com/Levenberg-MarquardtMethod.html (Erişim tarihi: 10.01.2015)

[44] Simon, C. P., Blume, L. E., Mathematics for Economists. New York: W. W. Norton, 1994.

[45] Nassif, A. B., Capretz, L. F., Ho, D., Analyzing the Non-Functional Requirements in the Desharnais Dataset for Software Effort Estimation. http://arxiv.org/abs/1405.1131 (Erişim tarihi: 15.12.2014)

[46] Machine Learning Group at the University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/ (Erişim tarihi: 18.12.2014)

[47] Anonim, http://www.mathworks.com/products/matlab/ (Erişim tarihi: 18.12.2014)

[48] Anonim, http://www.neurosolutions.com/neurosolutions/ (Erişim tarihi: 20.12.2014)

[49] Seref, B., Barisci, N., Software Effort Estimation using Multilayer Perceptron and Adaptive Neuro Fuzzy Inference System. 2014 3rd Journal Conference on Innovation, Management and Technology, vol. 5, no. 5, pp. 374-377, 2014.

EKLER

EK 1. Desharnais Veri Seti

No.	Project Numeric	TeamExp Numeric	ManagerExp Numeric	YearEnd Numeric	Length Numeric	Transactions Numeric	Entities Numeric	PointsNonAdjust Numeric	Adjustment Numeric	PointsAjust Numeric	Language Nominal	**Effort** Numeric
1	1.0	1.0	4.0	85.0	12.0	253.0	52.0	305.0	34.0	302.0	1	5152.0
2	2.0	0.0	0.0	86.0	4.0	197.0	124.0	321.0	33.0	315.0	1	5635.0
3	3.0	4.0	4.0	85.0	1.0	40.0	60.0	100.0	18.0	83.0	1	805.0
4	4.0	0.0	0.0	86.0	5.0	200.0	119.0	319.0	30.0	303.0	1	3829.0
5	5.0	0.0	0.0	86.0	4.0	140.0	94.0	234.0	24.0	208.0	1	2149.0
6	6.0	0.0	0.0	86.0	4.0	97.0	89.0	186.0	38.0	192.0	1	2821.0
7	7.0	2.0	1.0	85.0	9.0	119.0	42.0	161.0	25.0	145.0	2	2569.0
8	8.0	1.0	2.0	83.0	13.0	186.0	52.0	238.0	25.0	214.0	1	3913.0
9	9.0	3.0	1.0	85.0	12.0	172.0	88.0	260.0	30.0	247.0	1	7854.0
10	10.0	3.0	4.0	83.0	4.0	78.0	38.0	116.0	24.0	103.0	1	2422.0
11	11.0	4.0	1.0	84.0	21.0	167.0	99.0	266.0	24.0	237.0	1	4067.0
12	12.0	2.0	1.0	84.0	17.0	146.0	112.0	258.0	40.0	271.0	1	9051.0
13	13.0	1.0	1.0	84.0	3.0	33.0	72.0	105.0	19.0	88.0	1	2282.0
14	14.0	3.0	4.0	85.0	8.0	162.0	61.0	223.0	32.0	216.0	1	4172.0
15	15.0	4.0	4.0	85.0	9.0	223.0	121.0	344.0	28.0	320.0	1	4977.0
16	16.0	3.0	2.0	85.0	8.0	119.0	48.0	167.0	26.0	152.0	2	1617.0
17	17.0	4.0	3.0	85.0	8.0	57.0	43.0	100.0	43.0	108.0	1	3192.0
18	18.0	4.0	4.0	86.0	14.0	68.0	316.0	384.0	20.0	326.0	2	3437.0
19	19.0	3.0	4.0	87.0	14.0	9.0	386.0	395.0	21.0	340.0	2	4494.0
20	20.0	4.0	2.0	86.0	5.0	58.0	34.0	92.0	29.0	86.0	1	840.0
21	21.0	4.0	4.0	86.0	12.0	318.0	269.0	587.0	34.0	581.0	2	14973.0
22	22.0	2.0	4.0	85.0	18.0	88.0	170.0	258.0	34.0	255.0	1	5180.0
23	23.0	2.0	4.0	86.0	5.0	306.0	132.0	438.0	37.0	447.0	1	5775.0
24	24.0	4.0	1.0	87.0	20.0	304.0	78.0	382.0	39.0	397.0	1	10577.0
25	25.0	1.0	4.0	86.0	8.0	89.0	200.0	289.0	33.0	283.0	1	3983.0
26	26.0	4.0	1.0	85.0	14.0	86.0	230.0	316.0	33.0	310.0	1	3164.0
27	27.0	2.0	0.0	86.0	6.0	71.0	235.0	306.0	37.0	312.0	1	3542.0
28	28.0	3.0	1.0	85.0	14.0	148.0	324.0	472.0	39.0	491.0	1	4277.0
29	29.0	4.0	4.0	85.0	16.0	116.0	170.0	286.0	27.0	263.0	1	7252.0
30	30.0	4.0	1.0	85.0	14.0	175.0	277.0	452.0	37.0	461.0	1	3948.0

31	31.0	4.0	3.0	86.0	6.0	79.0	128.0	207.0	27.0	190.0	1	3927.0
32	32.0	1.0	1.0	86.0	9.0	145.0	38.0	183.0	27.0	168.0	3	710.0
33	33.0	4.0	4.0	87.0	9.0	174.0	78.0	252.0	41.0	267.0	3	2429.0
34	34.0	1.0	1.0	85.0	5.0	194.0	91.0	285.0	35.0	285.0	1	6405.0
35	35.0	2.0	2.0	88.0	3.0	126.0	49.0	175.0	38.0	180.0	3	651.0
36	36.0	1.0	3.0	86.0	17.0	317.0	119.0	436.0	34.0	432.0	2	9135.0
37	37.0	2.0	4.0	87.0	11.0	289.0	88.0	377.0	28.0	351.0	3	1435.0
38	38.0	-1.0	-1.0	87.0	8.0	260.0	144.0	404.0	24.0	360.0	1	5922.0
39	39.0	1.0	4.0	88.0	4.0	158.0	59.0	217.0	18.0	180.0	3	847.0
40	40.0	3.0	3.0	88.0	16.0	302.0	145.0	447.0	52.0	523.0	2	8050.0
41	41.0	1.0	1.0	87.0	9.0	451.0	48.0	499.0	28.0	464.0	1	4620.0
42	42.0	2.0	4.0	87.0	34.0	661.0	132.0	793.0	23.0	698.0	3	2352.0
43	43.0	1.0	1.0	88.0	10.0	64.0	54.0	118.0	25.0	106.0	1	2174.0
44	44.0	-1.0	4.0	86.0	39.0	284.0	230.0	514.0	50.0	591.0	1	19894.0
45	45.0	2.0	1.0	86.0	18.0	182.0	126.0	308.0	35.0	308.0	1	6699.0
46	46.0	2.0	3.0	87.0	27.0	173.0	332.0	505.0	19.0	424.0	1	14987.0
47	47.0	2.0	2.0	88.0	9.0	252.0	7.0	259.0	28.0	241.0	1	4004.0
48	48.0	4.0	3.0	85.0	11.0	131.0	180.0	311.0	51.0	361.0	1	12824.0
49	49.0	2.0	3.0	85.0	8.0	106.0	39.0	145.0	6.0	103.0	1	2331.0
50	50.0	3.0	3.0	85.0	9.0	96.0	108.0	204.0	29.0	192.0	1	5817.0
51	51.0	2.0	3.0	85.0	7.0	116.0	72.0	188.0	18.0	156.0	1	2989.0
52	52.0	3.0	3.0	85.0	6.0	86.0	49.0	135.0	32.0	131.0	1	3136.0
53	53.0	2.0	3.0	85.0	17.0	221.0	121.0	342.0	35.0	342.0	1	14434.0
54	54.0	1.0	1.0	87.0	12.0	61.0	96.0	157.0	18.0	130.0	1	2583.0
55	55.0	1.0	3.0	86.0	12.0	132.0	89.0	221.0	5.0	155.0	2	3647.0
56	56.0	3.0	7.0	86.0	13.0	45.0	387.0	432.0	16.0	350.0	2	8232.0
57	57.0	1.0	1.0	86.0	12.0	55.0	112.0	167.0	12.0	129.0	2	3276.0
58	58.0	1.0	4.0	87.0	8.0	124.0	52.0	176.0	14.0	139.0	2	2723.0
59	59.0	3.0	3.0	87.0	5.0	120.0	126.0	246.0	15.0	197.0	2	3472.0
60	60.0	1.0	2.0	87.0	6.0	47.0	32.0	79.0	14.0	62.0	2	1575.0
61	61.0	1.0	1.0	86.0	12.0	126.0	107.0	233.0	23.0	205.0	2	2926.0
62	62.0	3.0	2.0	86.0	6.0	101.0	45.0	146.0	15.0	117.0	2	1876.0
63	63.0	1.0	1.0	86.0	5.0	78.0	99.0	177.0	14.0	140.0	1	2520.0

64	64.0	4.0	7.0	86.0	13.0	69.0	74.0	143.0	14.0	113.0	1	1603.0
65	65.0	1.0	3.0	86.0	8.0	194.0	97.0	291.0	35.0	291.0	2	3626.0
66	66.0	2.0	-1.0	87.0	10.0	224.0	110.0	334.0	28.0	309.0	2	6783.0
67	67.0	2.0	4.0	87.0	15.0	323.0	184.0	507.0	35.0	507.0	2	11361.0
68	68.0	1.0	3.0	86.0	6.0	42.0	31.0	73.0	27.0	67.0	2	1267.0
69	69.0	1.0	2.0	87.0	5.0	74.0	43.0	117.0	25.0	105.0	2	2548.0
70	70.0	3.0	4.0	87.0	10.0	101.0	57.0	158.0	9.0	117.0	2	1155.0
71	71.0	0.0	4.0	86.0	6.0	97.0	42.0	139.0	6.0	99.0	3	546.0
72	72.0	2.0	3.0	84.0	13.0	134.0	77.0	211.0	13.0	165.0	2	2275.0
73	73.0	4.0	5.0	86.0	26.0	482.0	227.0	709.0	26.0	645.0	2	9100.0
74	74.0	0.0	2.0	84.0	6.0	213.0	73.0	286.0	6.0	203.0	3	595.0
75	75.0	0.0	-1.0	84.0	22.0	139.0	143.0	282.0	22.0	245.0	2	3941.0
76	76.0	2.0	3.0	86.0	24.0	473.0	182.0	655.0	40.0	688.0	2	13860.0
77	77.0	4.0	4.0	85.0	12.0	229.0	169.0	398.0	39.0	414.0	3	1400.0
78	78.0	4.0	3.0	83.0	12.0	227.0	73.0	300.0	34.0	297.0	1	2800.0
79	79.0	4.0	4.0	82.0	24.0	395.0	193.0	588.0	40.0	617.0	1	9520.0
80	80.0	4.0	3.0	86.0	12.0	469.0	176.0	645.0	43.0	697.0	3	5880.0
81	81.0	4.0	4.0	85.0	36.0	886.0	241.0	1127.0	34.0	1116.0	1	23940.0