ACTA INFOLOGICA – 2017 – 1 (2)

ACTA INFOLOGICA
ISSN: 2602-3563

Yapay Sinir Ağları ile Tahmin ve Sınıflandırma Problemlerinin Çözümü İçin Arayüz Tasarımı

Ayşe Arı, Murat Erşen Berberler

ÖZET

Yapay sinir ağları, sınıflandırma, modelleme ve tahmin gibi birçok günlük hayat probleminin çözümünde başarılı sonuç veren bir yöntemdir. Yapay sinir ağları nöronlar arasındaki bağlantı ağırlıklarını ayarlayarak öğrenme gerçekleştirir. Bu çalışmada tek katmanlı algılayıcılardan Adaline modeli ve çok katmanlı algılayıcı modelini içeren bir yazılım geliştirilmiştir. Bilindiği gibi tek katmanlı algılayıcılar lineer problemlere çözüm üretebilirken lineer olmayan problemlere çözüm üretememiş, lineer olmayan problemler yapay sinir ağları ile çok katmanlı algılayıcı modelinin geliştirilmesi ile çözülebilmişlerdir. Yazılımın tek katmanlı algılayıcı kısmında mantıksal AND ve OR problemleri doğrusal olarak sınıflandırabilirken, XOR problemini tek bir doğru ile sınıflandırılamadığı gözlemlenebilmektedir. Çok katmanlı algılayıcı kısmında ise, yapay sinir ağlarında yaygın olarak kullanılan geri yayılım algoritması ile tahmin ve sınıflandırma problemleri çözebilmektedir. Ağın katman sayısı, katmanlardaki nöron sayısı, iterasyon sayısı, öğrenme oranı, momentum katsayısı, aktivasyon fonksiyonu, normalizasyon yöntemi, başlangıç ağırlıkları gibi parametrelerin değiştirilerek ağın eğitilmesi sağlanabilmekte ve eğitilen ağ test edilerek ağın performans ölçümü yapılabilmektedir.

Anahtar Kelimeler: *Yapay sinir ağları, geri yayılım, adaline, sınıflandırma, tahmin*

ABSTRACT

Artificial neural networks are a successful method for solving many daily life problems such as classification, modeling and estimation. Artificial neural networks perform learning by adjusting the connection weights between the units. In this study, a software is developed which includes adaline model of single layer perceptron and multilayer perceptron model. As it is known, single layer perceptron model can produce solutions to linear problems, but can not nonlinear problems, they solved by multi-layer perceptron models with artificial neural networks. In the single layer part of the software, it can be observed that while logical AND and OR problems are linearly classified but the XOR problem can not be classified with a single line. In the multilayer perceptron part, it is possible to solve the approach and classification problems with the backpropagation algorithm which is widely used in artificial neural networks. The network can be trained by changing the parameters such as the number of layers in the network, the number of neurons in the layers, the number of iterations, learning rate, momentum constant, activation function, normalization method, initial weights, and network performance can be measured by testing the trained network.

Keywords: *Artificial neural networks, backpropagation, adaline, classification, prediction.*

Information of Author(s):

*** Ayşe Arı**
ayse.ari@ogr.deu.edu.tr
Dokuz Eylül Üniversitesi, Bilgisayar Bilimleri Bölümü

Murat Erşen Berberler
murat.berberler@deu.edu.tr
Dokuz Eylül Üniversitesi, Bilgisayar Bilimleri Bölümü

*** Contact Author**
Address: Dokuz Eylül Üniversitesi, Bilgisayar Bilimleri Bölümü, İzmir
Telephone Number: +90 232 3019514

Submit Date: 22.08.2017
Accept Date: 01.12.2017
Publish Date: 29.12.2017

İSTANBUL UNIVERSITY PRESS

ACTA INFOLOGICA
ISSN: 2602-3563

1. GİRİŞ

Günlük hayatımızda vazgeçilmez bir yere sahip olan bilgisayarın, artık insanlar gibi karar verme ve öğrenme gerçekleştirme yeteneklerini kazanması ile kullanım alanları oldukça genişlemiştir. Matematiksel olarak ifade edilemeyen ve insanlar tarafından çözülmesi zor olan problemlerin çözümünde yapay zekâ yöntemleri kullanılmaktadır. Yapay zekâ yöntemlerinin en temel özelliği, olaylara ve problemlere çözümler üretirken örneklerden, tecrübeden, benzetmelerden öğrenme gerçekleştirme ve karar verebilme yeteneklerinin olmasıdır.

Makine öğrenimi için en popüler yaklaşımlardan biri yapay sinir ağlarıdır**.** Yapay sinir ağları, geleneksel hesaplama yöntemleri ile çözülemeyen problemlerin çözümünde yaygın olarak kullanılmaktadır.

Öğrenme, genelleme, doğrusal olmama, hata toleransı, uyum, paralellik gibi üstünlüklere sahip olan yapay sinir ağları; görüntü ve sinyal işleme [1], hastalık tahmini [2], gibi tıbbi uygulamalarda; mühendislik [3], üretim, finans [4], optimizasyon, sınıflandırma gibi çok farklı uygulama alanlarında kullanılmaktadır.

Yapay sinir ağları için çeşitli programlama dillerinde birçok kütüphane ve araç geliştirilmiştir. Bunlardan yaygın olarak kullanılan ve genel kabul görmüş olanlardan birkaçı aşağıdaki gibidir [5].

- Stuttgart Neural Network Simulator (SNNS) yapay sinir ağları için Stuttgart Universitesi'nde IPVR enstitüsü tarafından geliştirilmiştir bir simülatördür. Sinir ağı uygulamaları ve araştırmaları için etkin ve esnek bir simülasyon ortamı oluşturmak amacıyla yapılmıştır.
- FANN (Fast Artificial Neural Network Library), çok katmanlı algılayıcıları kullanmayı sağlayan, C programlama dilinde yazılmış bir yapay sinir ağı kütüphanesidir.
- JOONE (Java Object Oriented Neural Engine), Java ortamında geliştirilmiş bir yapay sinir ağı kütüphanesidir.
- Matlab Neural Network Toolbox, yapay sinir ağlarının tasarlanmasını, gerçekleştirilmesini, görselleştirilmesini ve simülasyonunu sağlayan, Matlab için geliştirilmiş bir araçtır.

Bu çalışmada, C#.NET programlama dili kullanılarak YSA'nın öğrenme, genelleme, özelliklerinin kolayca görülebileceği, tahmin ve sınıflandırma problemlerinin uygulanabileceği görsel ve esnek bir yazılım geliştirilmiştir. Yazılımda tek katmanlı algılayıcılardan adaline modeli ve çok katmanlı algılayıcı modeli geri yayılım algoritması kullanılarak tasarlanmıştır. Yazılımın çok katmanlı algılayıcı bölümünde, ağın yapısı, katman ve katmanlardaki nöron sayısı, verilerin normalizasyon yöntemi, katmanlarda kullanılan aktivasyon fonksiyonu, öğrenme oranı, momentum katsayısı, iterasyon sayısı, eğitimi sonlandırma koşulu, başlangıç ağırlıklarının şeçimi kullanıcı tarafından belirlenmektedir. Eğitim esnasında oluşan hatanın grafiği gözlemlenebilmektedir. Test işleminden sonra ağın performansı ölçülebilmekte ve ayrıca sınıflandırma problemleri için karmaşıklık matrisi gözlemlenebilmektedir. Ağın yapısı, kullanılan parametreler, eğitim ve test verisi, eğitilen ağın bağlantı ağırlıkları, eşik değerleri ve test işleminden sonra ağın çıktıları, eğitim ve test işleminin hata ve süre değerleri bir klasör içerisinde (.txt) uzantılı dosyalar ile dizine kaydedilmektedir. Kaydedilen ağırlık ve eşik değerleri aynı ağ topolojisine sahip bir ağa başlangıç değerleri olarak verilip, eğitim ve test işlemi tamamlandığında başarım arttırılabilmektedir.

2. YÖNTEM: YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA), genel olarak insan beyninin ya da merkezi sinir sisteminin çalışma prensiplerini taklit eden bilgi işleme sistemidir [6]. Bu konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamıştır. Nöronlar bağlantılarla birbirine bağlanır ve her bağlantı, girdisinin gücünü veya diğer bir deyişle önemini ifade eden sayısal bir ağırlığa sahiptir. Ağırlıklar, YSA 'lardaki uzun süreli belleğin temel aracıdır. Bir sinir ağı, bu ağırlıkların tekrar tekrar ayarlanması yoluyla öğrenme gerçekleştirir [7].

Yapay sinir ağının genelleme yeteneği ağın topolojinin doğru seçilmesi ile birebir ilişkilidir. Ağ için en uygun mimari, problemi öğrenmek için yeterince büyük, genelleme yapabilmek için ise bir o kadar küçük olmalıdır. En uygun mimariden daha küçük bir ağ problemi iyi öğrenemez diğer taraftan daha büyük bir ağ ise eğitim verisini

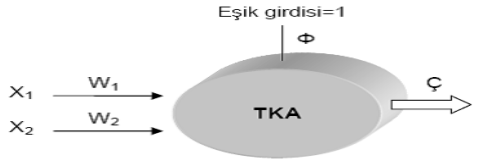
ACTA INFOLOGICA
ISSN: 2602-3563

aşırı öğrenir ki bu da ezberlemesine neden olduğundan genelleme yeteneği zayıf kalır. Ağın yapısının belirlenmesinde temel olarak büyüyen/yapıcı ve budama/yıkıcı olmak üzere iki açgözlü yaklaşım bulunmaktadır. Ağın yapısı küçük seçilip öğrenme sürecinde büyüyor ise büyüyen/yapıcı bir yaklaşım, aksine büyük seçilip öğrenme süresince küçülüyor ise budama/yıkıcı bir yaklaşım izlenmiştir [8].

YSA'lar genel olarak tek katmanlı algılayıcılar ve çok katmanlı algılayıcılar olarak ikiye ayrılmaktadır.

2.1. Tek Katmanlı Algılayıcılar

Tek katmanlı yapay sinir ağları doğrusal problemlerin çözümünde kullanılıp sadece girdi ve çıktı katmanından oluşmaktadır. Katmanların bir veya daha fazla nöronu bulunabilmektedir. Şekil 1'de basit bir tek katmanlı algılayıcı modeli gösterilmiştir.



Şekil 1. Tek Katmanlı Algılayıcı (TKA) Modeli

Eşik girdisi, bu tip ağlarda nöron elemanlarının değerlerinin ve de ağın çıktısının 0 olmasını önler. Değeri daima 1'dir. Ağın çıktısı Denklem 1'de gösterildiği gibi ağırlıklandırılmış girdi değerlerinin eşik değeri ile toplanması sonucu elde edilir.

$$Ç=f\left(\sum_{i=1}^{n} w_i x_i+\phi\right) \qquad (1)$$

Denklem 1'de x_i , $i = 1,2,\dots,n$ ile ağın girdileri, w_i , $i = 1,2,\dots,n$ ile bu girdilere karşılık gelen ağırlık değerleri, ϕ ile eşik değeri gösterilmiştir. Tek katmanlı algılayıcıda çıktı fonksiyonu doğrusaldır. Böylelikle, ağa gösterilen örnekler eşik değer fonksiyonu ile iki sınıf arasında paylaştırılarak iki sınıfı birbirinden ayıran doğru bulunmaya çalışılır. Ağın çıktısı 1 veya -1 değeri alır. Eşik değer fonksiyonu Denklem 2' de gösterilmiştir.

$$f(g) = \begin{cases} 1, & Ç > 0 \\ -1, & Ç \leq 0 \end{cases} \qquad (2)$$

Sınıf ayracı doğrusu Denklem 3'teki gibi tanımlanmaktadır.

$$W_1X_1 + W_2X_2 + \phi = 0 \qquad (3)$$

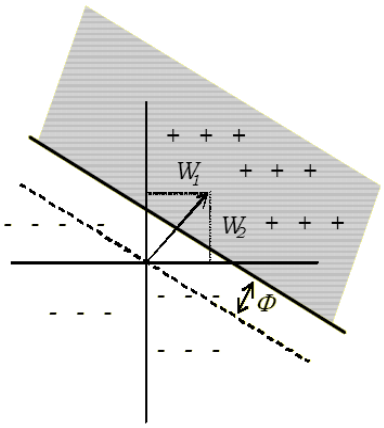
Buradan;

$$X_1 = -\frac{W_2}{W_1}X_2 - \frac{\phi}{W_1} \qquad (4)$$

ACTA INFOLOGICA
ISSN: 2602-3563

$$X_2 = -\frac{W_1}{W_2}X_1 - \frac{\phi}{W_2} \qquad (5)$$

olarak elde edilir. Denklem 4 ve 5 kullanılarak Şekil 2' de geometrik gösterimi verilen sınıf ayracı doğrusu çizilebilmektedir.



Şekil 2. Sınıf Ayıracı Doğrusu

Ağırlık değerleri, sınıf ayracı doğrusunun her iki grubu en iyi ayıracak şekilde belirlenmesi için her iterasyonda Denklem 6'daki formül ile değiştirilir.

$$W_i(t+1)=W_i(t)+\Delta W_i(t) \qquad (6)$$

Eşik değeri de sınıf ayıracı doğrusunu sınıflar arasında kaydırmak için her iterasyonda Denklem 7'deki formül ile güncellenir.

$$\phi(t+1)=\phi(t)+\Delta\phi(t) \qquad (7)$$

Tek katmanlı algılayıcılarda başlıca iki modelden söz edilebilir. Bunlar;

- Perceptron Modeli
- Adaline/Madaline Modeli

dir [9].

2.1.1. Basit Algılayıcı Modeli (Perceptron)

1958 yılında psikolog Frank Rosenblatt tarafından "zeki sistemlerin temel özelliklerinden bazılarını simüle etmek" amacıyla geliştirilen perceptron modeli, bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanır. Ağın çıktısı, girdi değerlerinin ağırlıklı toplamının bir eşik değer ile karşılaştırılması sonucu elde edilir. Toplam eşikten eşit veya büyük ise çıktı değeri 1, küçük ise 0 seçilir. Rosenblatt, algılayıcı eğitimi için desen tanıma problemlerini çözen bir öğrenme kuralı geliştirdi [10]. Bu kuralın, eğer problemi çözen ağırlıklar varsa, daima doğru ağırlıklara yakınsayacağını kanıtladı. Marvin Minsky ve Seymour Papert algılayıcılar üzerinde yaptıkları derin matematiksel incelemeler sonucunda, algılayıcıların çok sınırlı alanlarda kullanılabileceğini ve

ACTA INFOLOGICA
ISSN: 2602-3563

algılayıcının çözemeyeceği çok fazla problem sınıfı olduğunu yayınladıkları ”Perceptrons” kitabıyla kamuya göstermişlerdir [11]. Algılayıcıların çözemediği problemlere örnek olarak XOR problemini gösterebiliriz. Algılyıcıların bu sınırlaması 1980’lere gelindiğinde çok katmanlı perceptron modelinin geliştirilmesi ile giderilmiştir.

2.1.2 Adaline Modeli

Bernard Widrow, 1950’lerin sonlarında, Frank Rosenblatt’ın perceptronu geliştirdiği sırada, sinir ağları üzerine çalışmaya başlamıştır. 1960 yılında Widrow ve onun lisansüstü öğrencisi Marcian Hoff, ADALINE ağı ile En Küçük Kareler (Least Mean Square) algoritması olarak adlandırılan bir öğrenme kuralı geliştirdiler. Açık adı ’ADAptive LInear NEuron’ veya ’ADAptive LINear Element’ olan bu sinir hücresi modeli yapısal olarak algılayıcıdan farklı değildir. Ancak, algılayıcı aktivasyon fonksiyonu olarak eşik fonksiyonu kullanırken ADALINE doğrusal fonksiyon kullanır. Her iki modelde yalnızca doğrusal olarak ayrılabilen problemlere çözüm üretebilmektedir.

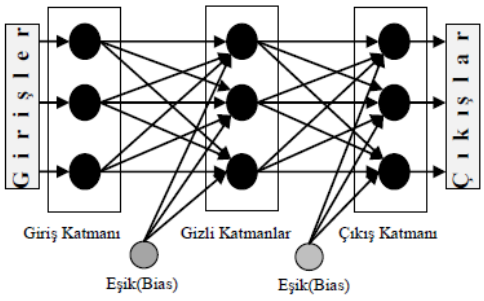
Widrow-Hoff kuralı da denilen En Küçük Kareler algoritması, perceptronun öğrenme kuralından daha güçlüdür. Perceptron öğrenme kuralı bir çözüme yakınsamayı garanti etse dahi, eğitim kalıplarının sınır çizgisine yakınlığından dolayı gürültüye duyarlı olabilir. En küçük kareler algoritması, ortalama karesel hatayı minimize ettiğinden eğitim kalıplarını sınır çizgisinden olabildiğince uzak tutmaya çalışır.

Widrow ve Hoff, birden fazla adaptif eleman içeren MADALINE yapay sinir ağı modellini de geliştirdiler.

2.2. Çok Katmanlı Algılayıcılar

Tek katmanlı algılayıcıların doğrusal olmayan problemlerin çözümünde başarısız olmasının üzerine geliştirilen çok katmanlı algılayıcılar (ÇKA), bilgi girişinin yapıldığı girdi katmanı, bir veya daha fazla gizli (ara) katman ve bir çıktı katmanından oluşmaktadır. ÇKA’da katmanlar arası ileri ve geri yayılım olarak adlandırılan geçişler bulunur. İleri yayılım safhasında, ağın çıktısı ve hata değeri hesaplanır. Geri yayılım safhasında ise hesaplanan hata değerinin minimize edilmesi için katmanlar arası bağlantı ağırlık değerleri güncellenir.

Şekil 3’te ÇKA yapısı gösterilmiştir.



Şekil 3. ÇKA Modeli

ÇKA modeli doğrusal perceptrondaki en küçük kareler algoritmasının genelleştirilmesi olan geri yayılım (backpropagation) öğrenme algoritmasını kullanır.

Geri Yayılım Algoritması

Geri yayılım (backpropagation), algoritması, ağın çıktısının belirlendiği ileri besleme ve oluşan hatanın gradiyenti azaltacak şekilde geri yayılarak ağırlıkların güncellendiği geri besleme safhalarından oluşmaktadır.

İleri besleme safhasında, eğitim setinin girdileri ağın giriş katmanına sunulur. Giriş katmanı, bu girdileri alan nöronları içerir. Bu sebeple giriş katmanındaki nöron sayısının veri setindeki girdi değeri sayısıyla aynı olması gerekir. Giriş katmanındaki nöronlar girdi değerlerini doğrudan gizli katmana iletir. Gizli katmandaki her bir nöron, ağırlıklandırılmış girdi değerlerine eşik değeri de ekleyerek toplam değeri hesap eder ve bunları bir

ACTA INFOLOGICA
ISSN: 2602-3563

aktivasyon fonksiyonu ile işleyerek bir sonraki katmana veya doğrudan çıkış katmanına iletir. Katmanlar arasındaki ağırlıklar başlangıçta genellikle rasgele seçilir.

Çıkış katmanındaki, her bir nöronun net girdisi, ağırlıklandırılmış girdi değerlerine eşik değerin eklenmesiyle hesaplandıktan sonra, bu değer yine aktivasyon fonksiyonu ile işlenerek çıktı değerleri belirlenir. Ağın çıktı değerleri beklenen çıktı değerleri ile karşılaştırılarak hata değeri hesaplanır. Bu sebeple, çıkış katmanında bulunacak nöronların sayısı veri setindeki çıktı sayısı ile örtüşmelidir.

$\boldsymbol{j}$. çıktı hücresi için $\boldsymbol{n}$ 'inci eğitim verisi sonrası, $\boldsymbol{d_j(n)}$ beklenen değer olmak üzere hata şu şekilde tanımlanır;

$$e_j(n) = d_j(n) - y_j(n) \qquad (8)$$

Çıktı katmanındaki toplam hata Denklem 9 ile ifade edilir.

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \qquad (9)$$

C kümesi çıktı katmanındaki tüm nöronları içermektedir. Burada delta kuralındakine benzer bir yaklaşımla $E(n)$ en düşük hale getirilmeye çalışılır. Çıktı katman hücresine gelen girdiler toplamı Denklem 10 ile ifade edilir.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n) \qquad (10)$$

X=(x_1,...,x_n), j. nörona uygulanan m girdi değerini, w_j, x_i girdisinin ağırlığını ve f aktivasyon fonksiyonunu göstermektedir. w_{j0} sapma elemanını gösterir ve böylece $x_0 = +1$ olur.

Ağın çıktı hücrelerinin ürettiği sonuç Denklem 11'deki formül ile hesaplanır.

$$y_j(n) = f(v_j(n)) \qquad (11)$$

Ağın gradyeni hata fonksiyonunun ağırlıklara göre türevi alınarak bulunabilir. Zincir kuralına göre, gradyen aşağıdaki şekilde ifade edilebilir:

$$\frac{\partial \mathrm{E(n)}}{\partial \mathrm{w}_{ji}(\mathrm{n})} = \frac{\partial \mathrm{E(n)}}{\partial \mathrm{e_j(n)}} \frac{\partial \mathrm{e_j(n)}}{\partial \mathrm{y}_j(\mathrm{n})} \frac{\partial \mathrm{y_j(n)}}{\partial v_j(\mathrm{n})} \frac{\partial \mathrm{v_j(n)}}{\partial \mathrm{w}_{ji}(\mathrm{n})}$$

Tek tek türevler alınırsa,

ACTA INFOLOGICA
ISSN: 2602-3563

$$\frac{\partial \mathrm{E(n)}}{\partial \mathrm{W}_{ji}(\mathrm{n})} = -e_j(n) f\left(v_j(n)\right) x_i(n) \quad (12)$$

Ağırlık düzeltme miktarı, $\boldsymbol{\Delta W_{ji}(n)}$ delta kuralına göre uygulanır.

$$\Delta W_{ji}(n) = -\eta \frac{\partial \mathrm{E(n)}}{\partial \mathrm{W}_{ji}(\mathrm{n})} \quad (13)$$

$\boldsymbol{\eta}$ öğrenme oranıdır. Denklem 13 'deki $-$ işareti ağırlık uzayındaki dik inişi temsil eder. Böylece geriye yayılma algoritması için ağırlık düzeltme miktarı, Denklem 14' deki gibi ifade edilir.

$$\Delta W_{ji}(n) = \eta \delta_j(n) x_i(n) \quad (14)$$

Yerel gradyen $\boldsymbol{\delta_j(n)}$ ise şöyle tanımlanır.

$$\delta_j(n) = e_j(n) f'\left(v_j(n)\right) \quad (15)$$

Gizli katmanda bulunan herhangi bir j nöronu için, çıkış katmanındaki nöronlar gibi istenen çıktı değeri belirtilmemiştir. Bu nedende gizli bir j nöronunun hata değeri, o nöronun direk bağlı olduğu tüm nöronların hata değerinden geri dönük olarak etkilenecektir. Gizli katmandaki herhangi bir j nöronu için, yerel gradyen $\delta_j(n)$ şu şekilde tanımlanır.

$$\delta_j(n) = f'\left(v_j(n)\right) \sum_{j=0}^{l} \delta_j(n) w_{ji}(n) \quad (16)$$

1986 yılında Rumelhart, geriye yayılım algoritmasının ağırlık güncelleme denklemine, α momentum terimi ekleyerek, ağın yerel minimuma takılması olasılığını azaltmıştır. Momentum terimi eklendikten sonra ağırlık güncelleme denklemi aşağıdaki hali alır.

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \quad (17)$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) x_i(n) + \alpha \Delta w_{ji}(n) \quad (18)$$

Geri yayılım algoritmasında eğitim setinin hesaplamaya dahil edilmesinde kullanılan iki öğrenme yöntemi vardır. Bu yöntemler tekil (online) eğitim ve toplu (batch) eğitim. Tekil eğitimde ağırlıkların güncelleştirilmesi işlemi eğitim veri setindeki her bir örnek ağa uygulandığında oluşan hatanın geri yayılımı ile gerçekleştirilir. Toplu eğitimde ise eğitim veri setinin tamamı ağa uygulandıktan sonra elde edilen ortalama hatanın geri yayılımı ile ağırlıkların güncelleştirilmesi yapılır. Toplu eğitim paralelleştirilebilirken, tekil eğitim paralelleştirilememektedir [12].

Eğitim sürelerini azaltmak için sezgisel yaklaşım yöntemleri kullanılmaktadır. Geri yayılım algoritmasında yakınsamayı hızlandırmak ve ağın performansını iyileştirecek birkaç teknikten biri olan sezgisel yaklaşım teknikleri momentum katsayısı kullanılarak yapılır. Momentum katsayısı YSA'nın daha hızlı toparlanmasına

ACTA INFOLOGICA
ISSN: 2602-3563

yardım eden bir faktördür. Temel olarak daha önceki değişimin bir kısmını işlem gören değişime eklemeye dayanır. Momentum katsayısı ağın yerel gradiyentleri aşmasını sağladığı gibi aynı zamanda hatanın düşmesine de yardımcı olur [13].

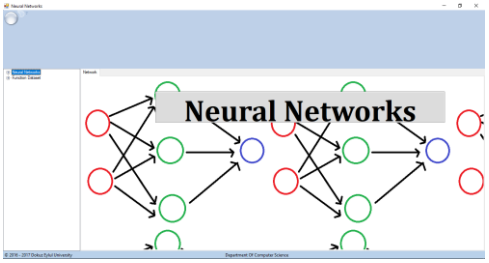
Öğrenme oranı (η), bir öğrenme prosedürünün hızı ve doğruluğu ile orantılı olup bunları kontrol eden bir sabittir. Öğrenme oranı, YSA'nın ağırlıklarının değişiminde kullanılmaktadır. Öğrenme oranı çok büyük seçilirse hata yüzeyinde geniş atlamalar meydana gelir, öğrenmenin gerçekleşeceği dar alanlar atlanabilir. Ayrıca, hata yüzeyi boyunca hareketler çok kontrolsüz olur. Çok küçük seçilmesi durumunda ise öğrenme süresi çok zaman alabilir. Tecrübeler $0.01 \leq \eta \leq 0.9$ aralığında seçilen öğrenme oranını iyi sonuçlar verdiğini göstermektedir. Büyük bir öğrenme oranı, başlangıçta iyi sonuçlara yol açar, ancak daha sonra yanlış sonuçlar verebilir. Daha küçük bir öğrenme oranı ile ise öğrenme daha zaman alıcıdır, ancak sonuç daha nettir. Böylece, öğrenme sürecinde öğrenme oranının başlangıçta büyük seçilerek zamanla (her iterasyonda veya birkaç iterasyonda bir) azaltılması gerekir [14].

Öğrenme oranının zaman içerisinde azaltılması çürüme (decay) olarak adlandırılır.

3. BULGULAR: GELİŞTİRİLEN ARAYÜZÜN İNCELENMESİ

YSA 'lar tahmin, sınıflandırma, modelleme gibi birçok günlük hayat probleminin çözümünde kullanılan bir yaklaşımdır. Geliştirilen yazılımda tek katmanlı algılayıcılardan Adaline modeli, çok katmanlı algılayıcılar için geri yayılım algoritması ile birçok alandan tahmin ve sınıflandırma problemlerinin çözümü için YSA yapılabilecek esnek ve görsel bir arayüz tasarlanmıştır. Ayrıca yazılımda tek değişkenli ve iki değişkenli fonksiyonlar ile YSA'nın sınıflandırma kısmında kullanılmak üzere veri kümesi üretilebilecek bir arayüz de mevcuttur.

Şekil 4'de tasarlanan yazılımın giriş ekranı gösterilmiştir. Ekranın sol alt bölümünde bulunan menüden istenilen arayüze ulaşılabilmektedir. İstenilen arayüze çift tıklandıktan sonra, arayüz menünün sağına gömülürken, o bölümde yapabileceğimiz işlemler ile ilgi butonların bulunduğu panel ekranın üstünde konumlanmaktadır.



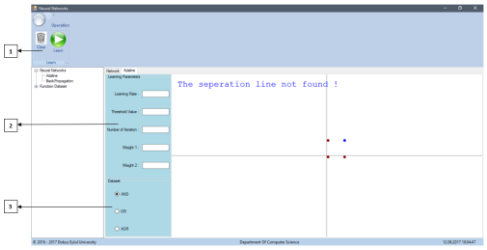
Şekil 4. Geliştirilen Arayüzün Giriş Ekranı

Bilindiği gibi tek katmanlı algılayıcılar ile doğrusal olarak ayrılabilen problemlerin çözümünde başarılı sonuçlar elde edilmektedir. Tasarlanan arayüzde mantıksal AND ve OR operatörünün adaline modeli ile doğrusal olarak ayrılabildiği gözlemlenebilirken, doğrusal olarak tek bir doğru ile ayrılamayan mantıksal XOR operatörüne çözüm üretemediği gözlenebilmektedir. Ayrıca kullanıcı ekrandan mouse'un sağ ve sol tuşunu kullanarak iki farklı sınıftan noktalar ekleyebilir ve iki sınıfın doğrusal olarak tek bir doğru ile ayrılıp ayrılamayacağını gözlemleyebilir.

Şekil 5'te menüden Adaline'a tıklandığında karşılaşılan ekran görüntüsü gösterilmiştir. Şekilde numaraldırılan bölümlerin yaptığı işlemler şu şekilde sıralanabilir.

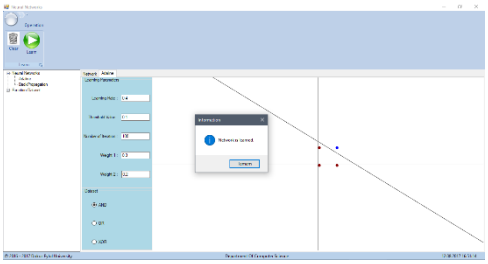
1. Ekranı temizlemeye yarayan "Clear" butonu ile öğrenme işlemini başlatan "Learn" butonunun bulunduğu paneldir.
2. Eğitimde kullanılacak öğrenme oranı, eşik değeri, iterasyon sayısı ve başlangıç ağırlık değerleri girildiği paneldir.
3. Veri setinin seçileceği paneldir.

ACTA INFOLOGICA
ISSN: 2602-3563



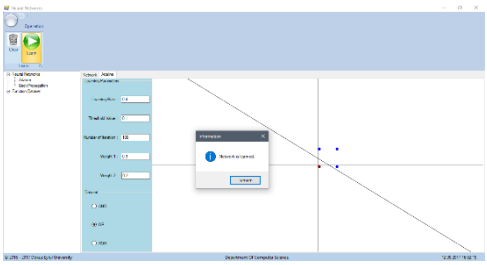
Şekil 5. Adaline Arayüzü

Şekil 6'da öğrenme oranı 0.4, eşik değeri 0.1, iterasyon sayısı 100, başlangıç ağırlıkları 0.3 ve 0.2 seçilerek eğitimi tamamlanmış mantıksal AND operatörü gösterilmiştir.



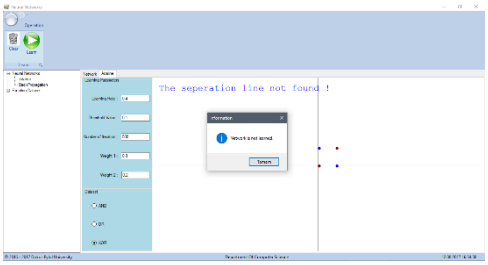
Şekil 6. Adaline Modeli ile Eğitilen AND Operatörü

Şekil 7'de öğrenme oranı 0.4, eşik değeri 0.1, iterasyon sayısı 100, başlangıç ağırlıkları 0.3 ve 0.2 seçilerek eğitimi tamamlanmış mantıksal OR operatörü gösterilmiştir.



Şekil 7. Adaline Modeli ile Eğitilen OR Operatörü

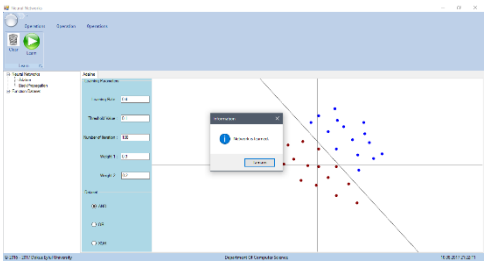
Şekil 8'de ise öğrenme oranı 0.4, eşik değeri 0.1, iterasyon sayısı 500, başlangıç ağırlıkları 0.3 ve 0.2 seçilerek eğitilen fakat 500 iterasyon tamamlanmasına rağmen doğrusal olarak ayrıştırılamayan mantıksal XOR operatörü gösterilmiştir.



Şekil 8. Adaline Modeli ile Eğitilemeyen XOR Operatörü

ACTA INFOLOGICA
ISSN: 2602-3563

Şekil 9'da ise mouse'un sağ ve sol tuşları ile eklenen noktaların, öğrenme oranı 0.4, eşik değeri 0.1, iterasyon sayısı 100, başlangıç ağırlıkları 0.3 ve 0.2 seçilerek iki farklı sınıfı ayıran doğru Adaline ağı ile eğitilen ağ ile çizilmiştir.



Şekil 9. Adaline Modeli ile Eğitilen iki farklı noktalar kümesi

Adaline sekmesi kapatılırken ekrandaki eğitim parametrelerini ve seçili veri setini (.ini) uzantılı dosyasına kaydeder. Bir sonraki çalıştırmada kaydedilen parametrelerle Adaline ekranı açılmaktadır.

Yazılımda Geliştirilen Backpropagation Arayüzü

Geliştirilen yazılım ile birçok alanda karşılaşılan tahmin ve sınıflandırma problemleri için YSA tanımlanabilmektedir.

Yazılımda çok katmanlı algılayıcılar için geri yayılım algoritması kullanılmıştır. Geri yayılım algoritmasının öğrenme kuralı genelleştirilmiş delta kuralı da denilen gradiyent azaltma öğrenme algoritmasıdır. Yazılımda momentum ile gradiyent azaltma öğrenme algoritması (gradient descent with momentum) kullanılmıştır.

Yazılımda verilerin işlenmesinde doğrusal (lineer), max-min ve z-score normalizasyon yöntemleri kullanılmıştır.

Lineer Normalizasyon

Bir veri setindeki verilerin en büyük değere bölünerek normalleştirilmesi işlemidir. Bu işlem sonucunda veriler [0,1] aralığında yer almaktadır.

Matematiksel olarak ifade edilmek istenirse $x_1, x_2, \ldots, x_n$, n adet veriyi göstersin. X_{max} $(\max = 1, 2, \ldots, n)$ veri setindeki en büyük değer , x_{normal_i} ; x_i $(i = 1,2, \ldots, n)$'nin normalize edilmiş hali olsun. O halde,

$$x_{normal_i} = \frac{x_i}{x_{max}} \qquad (19)$$

olarak tanımlanmaktadır.

Max-Min Normalizasyon

Bu yöntemde, bir grup verinin içerisindeki en büyük ve en küçük değerler ele alınır. Diğer bütün veriler, bu değerlere göre normalleştirilir. Buradaki amaç en küçük değeri 0 ve en büyük değeri 1 olacak şekilde normalleştirmek ve diğer bütün verileri bu 0-1 aralığına yaymaktır.

Matematiksel olarak ifade edilmek istenirse

$x_1, x_2, \ldots, x_n$, n adet veriyi göstersin. x_{max} $(\max = 1, 2, \ldots, n)$ veri setindeki en büyük değer , x_{min} $(\min = 1, 2, \ldots, n)$ veri setindeki en küçük değer, x_{normal_i} ; x_i $(i = 1,2, \ldots, n)$'nin normalize edilmiş hali ise Denklem 20 ile hesaplanmaktadır.

$$x_{normal_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \qquad (20)$$

ACTA INFOLOGICA
ISSN: 2602-3563

Z-Score Normalizasyon

Bu yöntemde ise ortalama değer (mean value) ve standart sapma (standard deviation) değerleri göz önüne alınır. Matematiksel olarak ifade edilmek istenirse;
$x_1, x_2, \ldots, x_n$, n adet veriyi göstersin.
$\bar{x}$: n adet verinin ortalaması
σ: n adet verinin standat sapması
x_{normal_i} ; x_i $(i = 1,2, \ldots, n)'$nin normallize edilmiş hali ise Denklem 21 'daki formül ile elde edilir.

$$x_{normal_i} = \frac{x_i - \bar{x}}{\sigma} \qquad (21)$$

Çoklu sınıflandırma problemlerinde veriler normalleştirilirken çıktı değerleri sınıf sayısı boyutunda vektör ile temsil edilir. Vektörde verinin ait olduğu sınıf **1** diğerleri **0** olarak belirlenir. Örneğin, **5** sınıflı bir veri de **3**. Sınıfa ait verinin çıktı değerinin normalize edilmiş hali $[\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}]$ vektörü ile gösterilmektedir.

Geliştirilen yazılımda doğrusal(lineer), sigmoid, rasyonel sigmoid, Gaussian ve tanjant hiperbolik aktivasyon fonksiyonları kullanılabilmektedir. Kullanılan aktivasyon fonksiyonlarının matematiksel ifadeleri Tablo 1'de gösterilmiştir.

Tablo 1. Yazılımda kullanılan aktivasyon fonksiyonları

Fonksiyonun Adı	Formülü
Lineer	$\mathbf{y = x}$
Sigmoid	$\mathbf{y = \frac{1}{1 + e^{-x}}}$
Rasyonel Sigmoid	$\mathbf{y = \frac{x}{1 + \sqrt{1 + x^2}}}$
Gaussian	$\mathbf{y = e^{-x^2}}$
Tanjant Hiperbolik	$\mathbf{y = \frac{1 - e^{-2x}}{1 + e^{-2x}}}$

Yazılımda geri yayılım algoritması ile eğitim yapılırken öğrenme oranı ve momentum belirlenen çürüme oranı (s) ile her iterasyonda (o anki iterasyon sayısı (t) kullanılarak) çürütülebilir.

Bu işlemin matematiksel ifadesi öğrenme oranı (η) ve momentum (μ) için sırasıyla Denklem 22 ve 23 'de gösterilmiştir.

$$\eta_{yeni} = \frac{\eta_{eski}}{(1+st)} \qquad (22)$$

$$\mu_{yeni} = \frac{\mu_{eski}}{(1 + st)} \qquad (23)$$

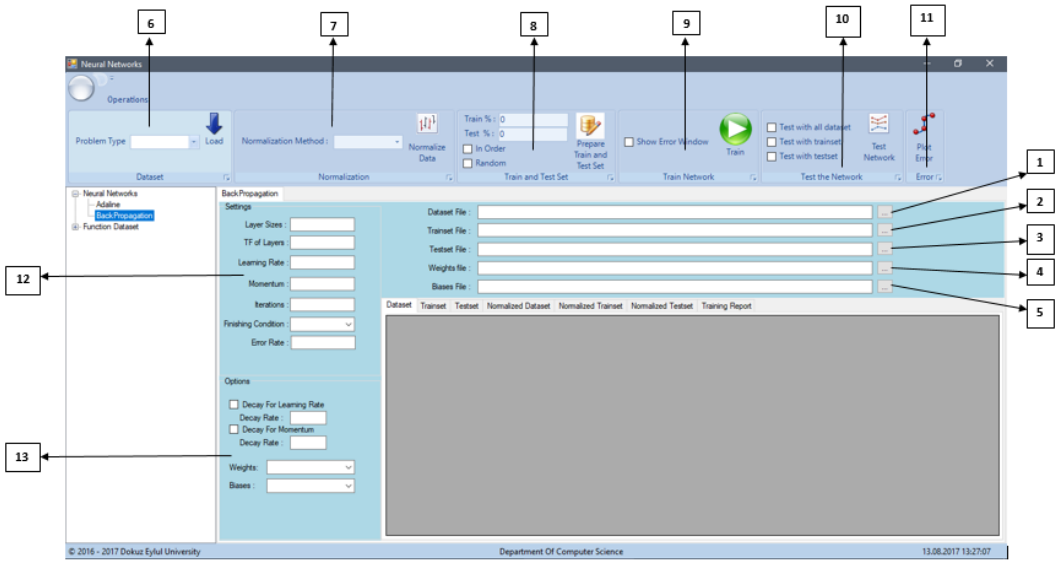
Yazılımda geliştirilen backpropagation arayüzünün tam ekran görüntüsü Şekil 10'da gösterilmiştir. Numarandırılan alanların işlevleri şu şekilde sıralanabilir.

ACTA INFOLOGICA
ISSN: 2602-3563

1. Veri setinin yükleneceği dosya konumunu açan butondur.
2. Tüm veri seti yerine eğitim ve test seti ayrı ayrı yükleneceği zaman eğitim setinin seçilmesi için dosya konumunu açan butondur
3. Tüm veri seti yerine eğitim ve test seti ayrı ayrı yükleneceği zaman test setinin seçilmesi için dosya konumunu açan butondur.
4. Başlangıçta katmanlardaki nöronlar arası bağlantı ağırlık değerleri dosyadan yüklenmek istendiğinde ilgili dosya konumunu açan butondur.
5. Girdi katmanı hariç katmanlardaki nöronların başlangıçtaki eşik değerleri dosyadan yüklenmek istendiğinde ilgili dosya konumunu açan butondur.
6. Uygulamanın yapılacağı problem tipinin seçilip verinin yüklendiği paneldir. Tahmin ve sınıflandırma olmak üzere iki tip problem seçimi yapılmaktadır. Tahmin problemleri için "Approximation", sınıflandırma problemleri için "Pattern Recognition" seçilmelidir. "Load" butonuna tıklanarak verinin yüklenmesi sağlanmaktadır.
7. Doğrusal, max-min ve z-score olmak üzere 3 tip normalizasyon seçeneği mevcuttur.
8. Eğitim ve test seti yüzdeleri belirlendikten sonra, eğitim ve test setinin sırayla veya rastgele oluşturulması tercihi yapılarak "Prepare Train and Test Set" butonuna tıklandığında eğitim ve test seti hazırlanmaktadır.
9. Ağın eğitiminin yapılacağı paneldir. Bu alan kullanılmadan önce 12 ve 13 ile numaralandırılmış alanlardaki ağın eğitim parametrelerinin girilmiş olması gerekmektedir. Bu alanda ağın eğitimi esnasında hatanın grafiğinin görüntülenmek istenip istenmediği seçimi yapılarak "Train" butonuna tıklanıp ağın eğitimi başlatılır.
10. Ağın test işleminin gerçekleştirildiği paneldir. Test işleminde kullanılacak veriseti seçimi yapıldıktan sonra "Test the Network" butonuna tıklanarak test işlemi başlatılır.
11. Eğitilen ağın hata grafiğinin görüntülenmesi için tıklanacak olan butonu barındıran paneldir.
12. YSA'nın eğitim işlemi tamamlanmadan önce ağın parametrelerinin belirlendiği paneldir. Bu parametreler;
    - Ağda kullanılacak katmanların ve bu katmanlardaki nöron sayılarının belirlenmesi. Katmanlardaki nöron sayıları aralarında birer boşluk bırakılarak "Layer Sizes" kutucuğuna girilmelidir. Birden fazla gizli katman eklenebilir. Örneğin "Layer Sizes " kutucuğuna girilen değer "2 3 4 1" ise, ilk değer girdi katmanı, son değerin çıktı katmanı ve aradaki değerler gizli katmanlardaki nöron sayılarını ifade eder. Bu örnekte; 2 nörona sahip girdi katmanı, 3 nörona sahip birinci gizli katman, 4 nörona sahip ikinci gizli katman ve 1 nörona sahip çıkış katmanı bulunmaktadır. Çıktı değerlerine göre program sınıf sayısını belirlemektedir.
    - Katmanlarda kullanılacak aktivasyon fonksiyonları belirlenmelidir. Linear, Sigmoid, Rasyonel Sigmoid, Gaussian ve Tanjant Hiperbolik aktivasyon fonksiyonları seçilebilmektedir. Katmanlarda kullanılmak istenen aktivasyon fonksiyonları baş harfleri ile aralarında birer boşluk kullanılarak "TF's of Layers" kutucuğuna girilmelidir. Girdi katmanında aktivasyon fonksiyonundan geçmeden direk gizli katmana iletildiği için girdi katmanı için N (none) harfi girilmelidir.
    - Ağın eğitilmesinde kullanılacak olan öğrenme oranının belirlenmesi. **0** ile **1** arasında olması önerilir.
    - Ağın eğitilmesinde kullanılacak olan momentum katsayısının belirlenmesi. **0** ile **1** arasında olması önerilir.
    - Ağın eğitiminde kullanılacak olan azami iterasyon sayısının belirlenmesi.
    - Eğitimi bitirme koşulunun belirlenmesi. Ağın eğitimi azami iterasyon sayısına ulaşıldığında, indirgenilmek istenilen minimum hata değerine ulaşıldığında ve azami iterasyon sayısı veya minimum hata değerinden biri sağlandığında durdurulmaktadır.
    - Ağın hatasının indirgenmek istenildiği minimum hata değerinin belirlenmesi.
13. Ağ eğitilirken öğrenme oranı veya momentum için çürüme (decay) yapılıp yapılmayacağının ve eğer yapılacaksa oranının belirlendiği paneldir. Bu panelde ayrıca ağın eğitiminde kullanılacak başlangıç

ACTA INFOLOGICA
ISSN: 2602-3563

ağırlıkları ve eşik değerlerinin seçimi belirlenir. Bu seçimde ağırlıklar ve eşik değerleri (-1,1), (-0.5,0.5), (0,1) aralıklarında rastgele, normal dağılım gösteren veya dosyadan okutularak belirlenebilir.



Şekil 10. Geliştirilen Backpropagation Arayüzü

Geliştirilen yazılımda sınıflandırma problemleri için doğruluk (accuracy), duyarlılık (sensitivity) ve özgüllük (specificity) değeri sırasıyla Denklem 24, 25 ve 26 ‘ da verilen formülle hesaplanmaktadır.

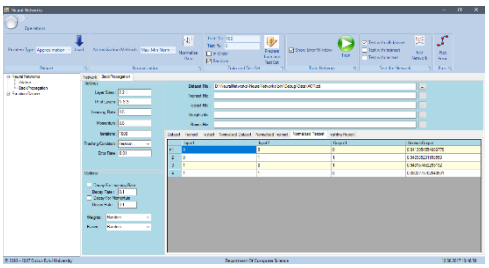
$$\text{Doğruluk}=\frac{\text{DP}+\text{DN}}{\text{DP}+\text{YN}+\text{YP}+\text{DN}} \qquad (24)$$

$$\text{Duyarlılık}=\frac{\text{DP}}{\text{DP}+\text{YN}} \qquad (25)$$

$$\text{Özgüllük}=\frac{\text{DN}}{\text{DN}+\text{YP}} \qquad (26)$$

DP: Doğru Pozitif
DN: Doğru Negatif
YP: Yanlış Pozitif
YN: Yanlış Negatif

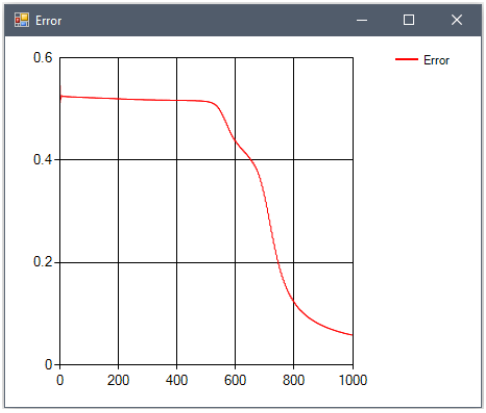
Şekil 11’de XOR mantıksal operatörünün geri yayılım algoritması ile eğitilmesi sonucunda elde edilen ekran görüntüsü verilmiştir. Eğitim setiyle test edilen ağın çıktılarının görüldüğü gibi gerçek çıktı ile arasındaki farkın (hata) yeterince küçük olduğu gözlenlenmektedir. Bu örnek ile ÇKA’nın doğrusal olmayan XOR operatörüne çözüm üretebildiği görülmektedir.



Şekil 11. XOR operatörünün ÇKA ile çözümü

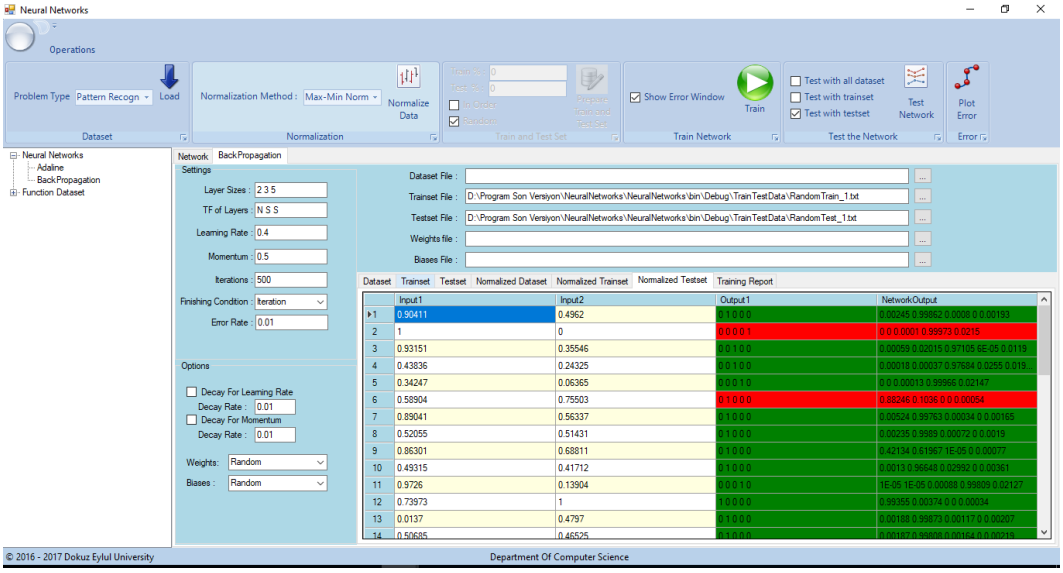
ACTA INFOLOGICA
ISSN: 2602-3563

Şekil 12'de eğitilen XOR operatörünün eğitim esnasındaki hata değerinin iterasyon sayısına göre grafiği gösterilmiştir.



Şekil 12. XOR operatörünün hata grafiği

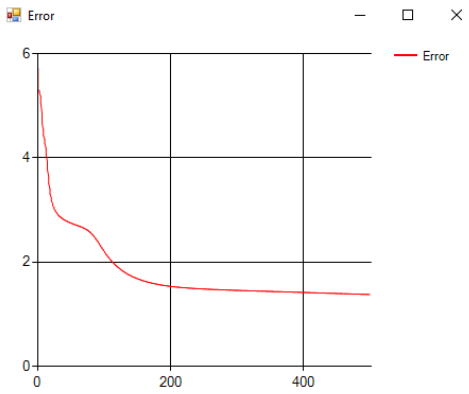
[-10,10] aralığında ele alınan xsinx fonksiyonun değer kümesinin 5 farklı sınıfa ayrılarak elde edilen sınıflandırma problemi için başlangıç ağırlıkları rastgele seçilen ağ, giriş katmanında 2 nöron, gizli katmanda 3 nöron ve çıktı katmanında 5 nöron (sınıf sayısı) ve gizli ve çıktı katmanlarında sigmoid aktivasyon fonksiyonu kullanılıp, öğrenme oranı 0.4, momentum katsayısı 0.5, iterasyon sayısı 500 seçilerek eğitilmiş ve daha önce hiç karşılaşmadığı test kümesi örnekleri ile test edilmiştir. Test edildikten sonra karşılaşılan ekran görüntüsü Şekil 13'te gösterilmiştir. 201 örnekten oluşan veri seti 158 örnek ile eğitilmiş, kalan 43 örnek ile test edilmiştir.



Şekil 13. Sınıflandırma Problemi Örneği

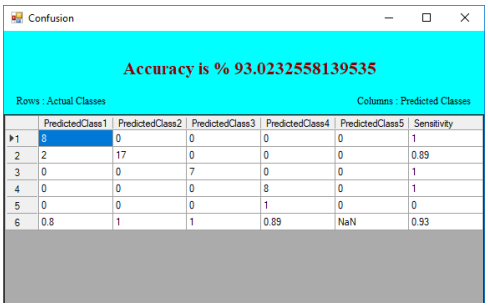
Ağ test edildikten sonra doğru sınıflandırılan örnekler yeşil, yanlış sınıflandırılan örnekler kırmızı ile renklendirilmiştir. Eğitim sırasında oluşan hatanın iterasyon sayısına göre değişim grafiği Şekil 14'te gösterilmiştir.

ACTA INFOLOGICA
ISSN: 2602-3563



Şekil 14. Sınıflandırma örneğinin hata grafiği

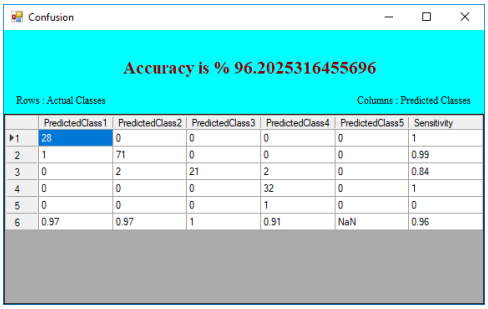
Ağın hiç karşılaşmadığı 43 adet test seti ile test edildikten sonra oluşan karmaşıklık matrisi Şekil 15'te verilmiştir. Karmaşıklık matrisinin satırları gerçek sınıfları, sütunları ise tahmini sınıf değerlerini göstermektedir. Sınıfların duyarlılık değerleri son sütunda gösterilirken özgüllük değerleri son satırda gösterilmiştir. Matrisin son satırı ve sütununun kesiştiği hücrede doğruluk oranı gösterilmiştir. Doğruluk oranı ayrıca matrisin üzerinde de belirtilmiştir.



	PredictedClass1	PredictedClass2	PredictedClass3	PredictedClass4	PredictedClass5	Sensitivity
1	8	0	0	0	0	1
2	2	17	0	0	0	0.89
3	0	0	7	0	0	1
4	0	0	0	8	0	1
5	0	0	0	1	0	0
6	0.8	1	1	0.89	NaN	0.93

Şekil 15. Rastgele Ağırlıklarla Eğitilip Test Seti ile Test edilen Sınıflandırma Örneğinin Karmaşıklık Matrisi

Aynı ağın 158 örnekten oluşan eğitim seti ile test edilmesi sonucu oluşan karmaşıklık matrisi Şekil 16'da gösterilmiştir.

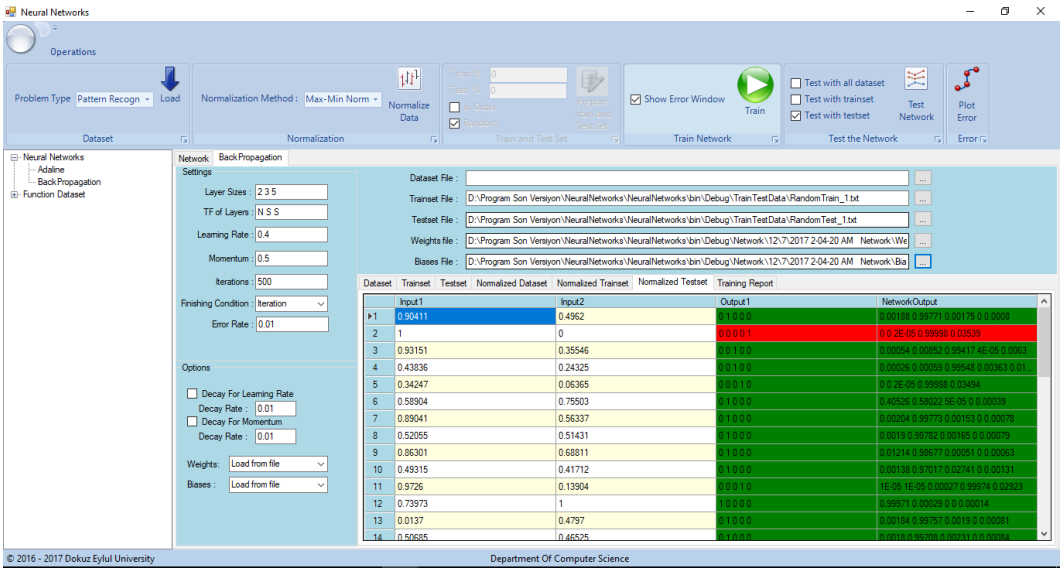


	PredictedClass1	PredictedClass2	PredictedClass3	PredictedClass4	PredictedClass5	Sensitivity
1	28	0	0	0	0	1
2	1	71	0	0	0	0.99
3	0	2	21	2	0	0.84
4	0	0	0	32	0	1
5	0	0	0	1	0	0
6	0.97	0.97	1	0.91	NaN	0.96

Şekil 16. Rastgele Ağırlıklarla Eğitilip Eğitim Seti ile Test Edilen Sınıflandırma Örneğinin Karmaşıklık Matrisi

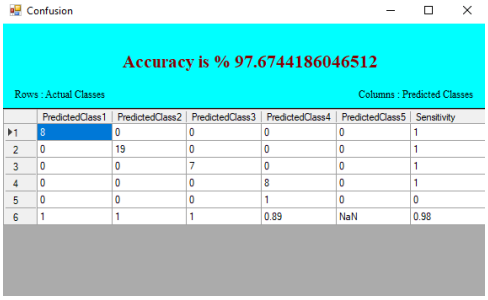
Ağ aynı parametreler ile yine aynı eğitim seti kullanılarak, başlangıç ağırlıkları ve eşik değerleri Şekil 17'de gösterildiği gibi bir önceki eğitimde elde edilen ağırlık ve eşik değerleri dosyadan seçilerek eğitilmiştir.

ACTA INFOLOGICA
ISSN: 2602-3563



Şekil 17. Öğrendiği Ağırlıklarla Eğitilip Test edilen Sınıflandırma Örneği

Öğrendiği ağırlıklar ve eşik değerleri ile eğitilip test edilen ağın karmaşıklık matrisi Şekil 18'de gösterilmiştir.

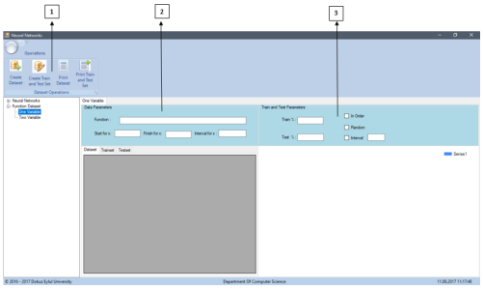


Şekil 18. Öğrendiği Ağırlıklarla Eğitilen Sınıflandırma Örneğinin Karmaşıklık Matrisi

Şekil 18'de görüldüğü gibi öğrendiği ağırlıklar ile eğitilen ağ, başlangıç ağırlıkları rastgele seçilerek eğitilen ağa göre daha başarılı bir sonuç vermektedir. Burada bağlantı sayısının eşleşmesi için ağın topolojinin aynı seçilmesine dikkat edilmesi gerekmektedir.

Tek Değişkenli ve İki Değişkenli Fonksiyonlar ile Veri Seti Oluşturulması

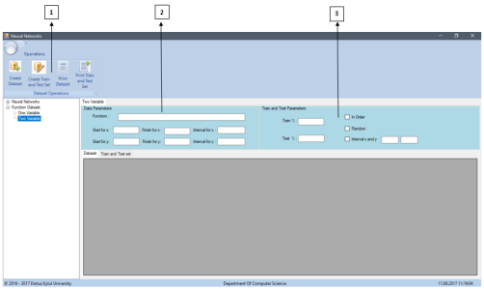
Yazılımın tek değişkenli ve iki değişkenli fonksiyon oluşturma arayüzlerine menüde "Function Dataset" 'ine tıklandıktan sonra açılan "One Variable" ve "Two Variable" sekmelerine tıklanarak ulaşılabilmektedir. Tek değişkenli ve iki değişkenli fonksiyonların değer kümesinin sıfıra eşit, sıfırdan küçük ve sıfırdan büyük değerlerine göre 3 sınıfa ayrılarak veri seti oluşturulur. Şekil 19'da tek değişkenli fonksiyon oluşturmak için geliştirilen arayüz gösterilmiştir.



Şekil 19. Tek değişkenli fonksiyon arayüzü

ACTA INFOLOGICA
ISSN: 2602-3563

Şekil 20'de iki değişkenli fonksiyon oluşturmak için geliştirilen arayüz gösterilmiştir



Şekil 20. İki değişkenli fonksiyon arayüzü

Numaralandırılmış alanların işlevleri şu şekildedir.

1. Veri setlerinin oluşturulması ve dosyaya yazılması için gerekli butonların bulunduğu paneldir. 2 ve 3 ile numaralandırılan alanlar doldurulduktan sonra kullanılır.
2. Fonksiyonun yazıldığı, başlangıç, bitiş ve arttırım aralığının belirlendiği paneldir. "Function" kutucuğuna fonksiyon yazılır, tek değişkenli fonksiyon için değişken değeri sadece "x" notasyonu ile iki değişkenli fonksiyonda ise değişkenler sadece "x" ve "y" notasyonları ile ifade edilip değişkenler parantez içerisinde yazılmalıdır. Değişkenlerin tanım kümesindeki başlangıç ve bitiş aralığı ile arttırım aralığı belirlenmektedir.
3. Oluşturulmak istenen eğitim ve test kümelerinin yüzdeleri belirlenerek, kümelerin oluşturulmasının sırayla, rastgele veya değişkenlerin belli aralıkta alınarak yapılması seçeneklerinden biri belirlenmektedir.

Geliştirilen yazılım C# ortamında ribbon form kullanarak tasarlanmıştır. Programın düzgün çalışabilmesi için kullanılan bilgisayarın ondalık ayracının nokta (.) olması gerekmektedir. Programda kullanılacak veri kümelerinin (.txt) formatında, her satırda bir örnek ve her örneğin giriş değerleri ve çıktı değeri aralarında birer boşluk bırakılarak sütunlarda yer alacak şekilde hazırlanmalıdır. Değişkenlerde başlık satırı bulunmamalıdır. Değişken sayılarında herhangi bir kısıtlama yoktur. Son sütun çıkış değerini göstermelidir. Sınıflandırma problemi içinde çıkış değerleri tek sütunda verilmelidir. Program kendi içerisinde sınıf sayılarını belirleyerek çıktı değerlerini vektör şeklinde normalleştirir. Veriler normalleştirilmiş şekilde yüklenir ise ondalıklı veriler için yine nokta kullanılmalı ve sınıflandırma problemlerinin çıktı sütunu normalleştirilmemiş şekilde girilmelidir.

4. TARTIŞMA ve SONUÇ

Geliştirilen algoritmada backpropagation algoritması ile tahmin ve sınıflandırma problemlerine çözüm getirilmiştir. Arayüzün backpropagation bölümünde kullanıcıya sunulan imkanlar şöyle listelenebilir.

1. Geri yayılım algoritması ile doğrusal olmayan tahmin ve sınıflandırma problemlerine çözüm üretebilmektedir.
2. Çözülecek problem için veriler tüm veri seti veya eğitim ve test seti olarak ayrı ayrı yüklenebilmektedir. Yüklenen veriler (.txt) uzantılı dosyalara kaydedilir.
3. Veriler linear, max-min ve z-score (z-standartlaştırma) normalizasyon yöntemleri ile normalize edilebilmektedir.
4. Kullanıcı verileri normalize edilmiş şekilde de programa yükleyebilmektedir. Bu durumda normalizasyon bölümünde "None" ifadesi seçilmelidir. Normalize edilen eğitim ve test setleri (.txt) uzantılı dosyalara kaydedilir.
5. Eğer tüm veri seti yüklendi ise normalizasyon işleminin ardından eğitim ve test seti kullanıcının belirlediği yüzdeler ile sırayla veya rastgele oluşturulabilmektedir.
6. Kullanıcı birden fazla gizli katman ve bu katmanlarda istenildiği kadar nöron ekleyebilmektedir.

ACTA INFOLOGICA
ISSN: 2602-3563

7. Katmanlarda linear, sigmoid, rasyonel sigmoid, Gaussian ve tanjant hiperbolik aktivasyon fonksiyonu kullanılabilmektedir.
8. Ağın eğitilmesinde kullanılan başlangıç ağırlıkları ve eşik değerleri -1 ile 1 ,-0.5 ile 0.5, 0 ile 1 arasında rastgele, normal dağılım gösteren (yani standart sapması 0 varyansı 1 olan) değerler ile başlayabildiği gibi kullanıcı tarafından (.txt) uzantılı dosyadan da girilebilmektedir.
9. Eğitim sırasında momentum ve öğrenme oranı çürümesi (decay) yapılabilmektedir.
10. Ağ eğitildiği esnada hata grafiği opsiyona göre eğitim sırasında veya sonrasında görülebilmektedir.
11. Ağ eğitimi tamamlandıktan sonra, test seti için Şekil 10 'nun 10.bölümünde gösterildiği gibi test işleminde kullanılacak olan veri seti opsiyonel olarak tüm veri seti, eğitim seti veya test seti olarak belirlenebilmektedir. Sınıflandırma problemlerinde ağ test edildikten sonra karmaşıklık (confusion) matrisi, doğruluk, duyarlılık ve özgüllük değerleri görülebilmektedir.
12. Veriler eğitildikten sonra ağın yapısı, kullanılan parametreler, son durumdaki bağlantı ağırlıkları ve eşik değerleri (.txt) uzantılı dosyalara kaydedilir. Kaydedilen bağlantı ağırlıkları ve eşik değerleri, aynı problem için aynı topolojiye sahip ağ ile tekrar eğitilmek istendiğinde başlangıç değerleri olarak seçilerek başarımın arttığı gözlemlenmektedir.
13. Test aşaması tamamlandıktan sonra test verisi için gerçek çıktı değerleri ile ağın ürettiği çıktı değerleri, hata ve süre değerleri, sınıflandırma problemleri için karmaşıklık matrisi doğruluk, duyarlılık ve özgüllük değerleri (.txt) formatında dosyaya kaydedilmektedir.
14. Backpropagation arayüzü kapatılırken kullanılan parametre değerlere (.ini) dosyasına kaydedilir, ekran tekrar açıldığında kaydedilen son kaydedilen değerler ile birlikte arayüz açılmaktadır.

Yazılım nesne tabanlı yaklaşım ile geliştirildiğinden ileriki çalışmalar için geri yayılım algoritmasına farklı öğrenme fonksiyonları ve aktivasyon fonksiyonları kolaylıkla eklenebilir. ÇKA'da kullanılan farklı algoritmalar için arayüz oluşturulabilir.

Geliştirilen yazılım hali hazırda kullanılmakta olan birçok yapay sinir ağı kütüphanesine veya paket programlarının aksine ücretsiz olarak kullanıma sunulmuştur.

Bilimsel hesaplama paket programlarının içinden çağırılan yapay sinir ağı kütüphanelerini kullanabilmek için o paket programa ait notasyon bilgisine sahip olmak gerekmekte iken geliştirilen yazılımda tüm işlemler kullanıcı dostu arayüz ile gerçekleştirilmektedir.

Program Neural Networks linkinden indirilip kullanılabilir.

KAYNAKLAR

[1] A. S. Miller, B. H. Blott, and T. K. Hames, "Review of neural network applications in medical imaging and signal processing," *Med. Biol. Eng. Comput.*, vol. 30, no. 5, pp. 449–464, 1992.

[2] R. Das, I. Turkoglu, and A. Sengur, "Expert Systems with Applications Effective diagnosis of heart disease through neural networks ensembles," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7675–7680, 2009.

[3] M. A. Shahin, M. B. Jaksa, and H. R. Maier, "Artificial Neural Network Applications in Geotechnical Engineering," *Aust. Geomech.*, vol. 36, no. 1, pp. 49–62, 2001.

[4] P. D. McNelis, *Neural Networks in Finance: Gaining Predictive Edge in the Market*. Elsevier Inc., 2005.

[5] A. C. Kınacı, "Görsel yazilim geliştirme ortami ile beraber bir yapay sinir ağı kütüphanesi tasarimi ve gerçekleştirimi," Ege Üniversitesi, 2006.

[6] J. A. Freeman and D. M. Skapura, *Neural Networks Algorithms , Applications and Programming Techniques*. New York, USA: Addison-Wesley Publishing Company, 1991.

ACTA INFOLOGICA
ISSN: 2602-3563

[7] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson Education, 2005.

[8] O. ARAN, O. T. YILDIZ, and E. ALPAYDIN, "an Incremental Framework Based on Cross-Validation for Estimating the Architecture of a Multilayer Perceptron," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 2, pp. 159–190, 2009.

[9] E. Öztemel, *Yapay Sinir Ağları*. İstanbul: Papatya Yayıncılık Eğitim, 2012.

[10] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in …," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[11] M. Minsky and S. Papert, *Perceptrons*. 1969.

[12] S. Haykin, *Neural Networks and Learning Machines (Vol.3.)*, Pearson. Upper Saddle River, NJ, USA, 2009.

[13] R. Bayındır and Ö. Sesveren, "Ysa Tabanlı Sistemler İçin Görsel Bir Arayüz Tasarımı," *MÜHENDİSLİK BİLİMLERİ DERGİSİ*, vol. 14, no. 1, pp. 101–109, 2008.

[14] D. Kriesel, *A Brief Introduction to Neural Networks*. 2007.