Koc. Üni. Fen Bil. Der., 3(1): (2020) 133-146

Kocaeli Üniversitesi

Kocaeli Üniversitesi Fen Bilimleri Dergisi

http://dergipark.gov.tr/koufbd

İleri Sürümlü Yapay Sinir Ağları Eğitim ve Geliştirme Aracı Tasarımı

Design of Training and Development Tool for Feedforward Artificial Neural Networks

Serhat YILMAZ[1], Sadettin Burak KILCI[2,*]

[1] *Elektronik ve Haberleşme Mühendisliği Bölümü, Mühendislik Fakültesi, Kocaeli Üniversitesi, Kocaeli, Türkiye,* ***Orcid:*** *0000-0001-9765-7225*
[2] *Elektronik ve Haberleşme Mühendisliği Bölümü, Mühendislik Fakültesi, Kocaeli Üniversitesi, Kocaeli, Türkiye,* ***Orcid:*** *0000-0002-6583-8379*

Araştırma Makalesi

Gönderilme Tarihi : *21/01/2020*
Kabul Tarihi : *30/06/2020*

Anahtar Kelimeler

Yapay Sinir Ağları
Mühendislik Eğitim Araçları
İleri Sürümlü Yapay Sinir Ağları
Geriye Yayılım Algoritması
Delphi® Programlama Dili

Özet

Bu çalışmada, elektronik ve yazılım üzerine çalışan öğrencilere yapay sinir ağlarının (YSA) çalışma prensipleri hakkındaki bilgilerini pekiştirmeye yönelik bir eğitim yazılımı hazırlanmıştır. YSA parametrelerinin işlevleri ve nasıl tanımlandığı arayüz (GUI) programında uygulamalı olarak gösterilmektedir. Program algoritması eğitim, sınama ve uygulama altyordamları olmak üzere üç aşamadan oluşmaktadır. YSA'na girilen Eğitim Çiftleri, istenen bir örüntüyü veya bilinen bir işlevi oluşturmaya yönelik giriş ve çıkış verilerini içermektedir. Buna karşılık YSA'nın her iterasyonda verdiği cevaplar gerek görsel gerekse sayısal olarak, istenen cevaba yaklaşılma aşamalarını öğrencilere göstererek ağın öğrenme sürecini pekiştirmelerini sağlamaktadır. Çalışmada bir örnek olarak sinüs işlevinin eğitim ve sınama aşamaları verilmiştir.

Research Paper

Received Date : *21/01/2020*
Accepted Date : *30/06/2020*

Keywords

Artificial Neural Networks
Engineering Educational Tools
Feedforward Artificial Neural Networks
Backpropagation Algorithm
Delphi Programming Language

Abstract

In this study, an educational software has been prepared to enhance the knowledge of the processing principles on artificial neural networks (ANNs) to students who are working on electronics and software. The functions of the ANN parameters and how they are defined are shown in the interface program as applied. The program algorithm consists of three stages: Training, Testing and Implementation Subroutines. Training Couples entered in the ANN include input and output data to create a desired pattern or a known function. On the other hand, the responses of the ANN in each iteration enable the network to reinforce the learning process by showing the students the stages of approaching the desired answers both visually and numerically. In the study, the education and testing stages of a sinus function are given as an example.

1. Giriş

Yapay sinir ağları, canlılardaki sinir sisteminin matematiksel modelinden yola çıkılarak geliştirilmiş, genelleme amaçlı, belirli bir öğrenme performansını gösterebilen hesaplama sistemleridir. Nöronlar, YSA'nın öğrenen ve bu doğrultuda genelleme yapan bölgesini oluştururlar[1]. YSA işlem birimleri olan nöronlar, birbirlerine bağlanarak paralel bir biçimde çalışırlar. Nöronun matematiksel modeli 1943'te Mc Culloch ve Walter Pitts tarafından önerilmiştir ve temel bir nöron ağının bile prensipte bir matematik ya da mantık problemini hesaplayabileceğini göstermişlerdir [2,3]. YSA, genelleme özelliği ile klasik yöntemlerdeki ara adım ihtiyaçlarını ortadan kaldırdığı için, saklanması gereken işlenmiş veri hacmini önemli ölçüde azaltır [4]. Böylece, eğitilmiş bir YSA, eksik bilgiler ışığında oldukça tutarlı tahminler yapabilir. Kullandıkları nöron aktivasyon işlevleri eğrisel olduğu için YSA doğrusal olmayan sistemleri modellemeye oldukça uygundur [5]. YSA'nın işlem birimleri olan nöronlar, ağın genelleme yeteneğini oluşturmak için düzenlenmiştir. İşlem birimi düğümleri arasındaki bağlantı ağırlıklarının, istenen çıkışı verecek şekilde ayarlanmasına öğrenme adı verilir [6].

* Sorumlu Yazar *(Corresponding Author)*: burakkilci1@gmail.com

Literatürde çeşitli YSA mimarileri yer almaktadır. En yaygın model, bu çalışmada da kullanılmış olan çok katmanlı ileri bildirimli algılayıcı(Multi Layered Perceptron-MLP) ağıdır. YSA, bulanık mantık, genetik algoritmalar, parçacık sürü optimizasyonu gibi yeni hesaplama teknikleri, işlemsel zekâ uygulamalarına temel oluşturmaktadır [11,12,13]. Günümüzde YSA çalışmaları, özellik eşleştirme kullanılarak yüzün kısıtlı bir parçasından kişi tanımadan [18], doğrusal olmayan sistemlerin uyarlanabilir kontrolüne [19], sağlık verilerindeki anomalilerin tespitine [20], yıllık en büyük su taşkınlarının tahminine [21] kadar pek çok alanda yaygın olarak kullanılmaktadır. YSA konusunda öğrenim gören öğrencilerin; temel algoritmaları ve işleyişini uygulamalar üzerinde gözlemleyip yorumlayabilmelerini sağlamak için literatürde eğitim amaçlı pek çok çalışma yapılmıştır. YSA eğitimi ve araştırması için hazırlanan internet tabanlı bir geliştirme aracı, doğrudan erişilebilir arayüzü aracılığıyla farklı ağlar oluşturma ve analizini yapabilme imkânı vermektedir [14]. Bu çalışmada öğrenciler, istedikleri şekilde eğitim ve test verileri hazırlayıp kullanabilmekte ve parametreleri ayarlayabilmektedir. Bu parametrelerin YSA eğitim ve geliştirme kalitesini nasıl etkilediği gözlenerek, farklı örüntülerin eğitilme aşamaları üzerinde benzetim yapabilmektedir. Kullanıcılar, farklı fonksiyonları ve bunların YSA çıkışı üzerindeki etkilerini deneyebilmektedir. Benzer şekilde bir gurup araştırmacı, GUI tabanlı kullanıcı dostu bir YSA modelleyicisi hazırlamışlardır [15]. Öğrenciler bu modelleyici aracılığıyla farklı YSA algoritmalarıyla deneyler yapıp kendi özgün modellerini oluşturabilmekte ve bunları anlaşılır bir şekilde takip edip yorumlayabilmektedir. Bir diğer çalışma gurubu, bir defada gerek yazılıma, gerek donanıma şeffaf bir şekilde erişilebilirlik sağlayabilen, web tabanlı olduğu için, indirme, kurulum veya herhangi bir kısıtlamaya gerek olmadan kolayca çalıştırılabilen çevrimiçi bir benzetim aracı hazırlamıştır [16]. Basit işlem birimlerini kolektif olarak kullanabilme imkânı vermesi, YSA'larını öğrenciler açısından ilgi çekici hale getirmektedir [17].

Bu çalışmada, Elektronik ve Bilgisayar Mühendisliği öğrencilerine, üzerinde çeşitli parametreleri deneyerek YSA'nın çalışma ilkelerini öğrenebilecekleri ve yeni geliştirmeler yapabilecekleri açık kaynak kodlu bir eğitim uygulaması tasarlanmıştır. Uygulama, Delphi® 6.0 programlama dilinde [22] öğrencilerin kolay anlayabileceği kullanıcı dostu ve Türkçe bir arayüzde hazırlanmıştır. Açık kaynak kodu, verilen şablon üzerinde kullanıcıların özgün yazılımlar geliştirebilmelerine ve parametrelerin, ağın performansına etkilerini görsel olarak takip edebilmelerine imkân verecektir.

2. İleri Sürümlü Yapay Sinir Ağları

YSA; eksik bilgi koşulları altında bile, bir işlevi kabul edilebilir bir doğrulukta kestirebilme kapasitesine sahiptir [5]. Son yıllarda YSA, doğrusal olmayan pek çok sistemin modellenmesinde yaygın olarak kullanılmaya başlanmıştır [23]. Sistemlerin girişleri ile çıkışları arasındaki doğrusal olmayan ilişki çıkarılabildiğinden YSA burada çok boyutlu bir interpolasyon işlevi görmektedir.

2.1.Yapay Sinir Hücresi Davranışı

Algılayıcı olarak ta bilinen yapay sinir hücresi (işlem birimi), biyolojik sinir hücresinin temel davranışını taklit eden basit bir matematiksel işlem birimidir. Kendisine gelen işaretleri toplar ve belli bir eşik (thr) seviyesini geçince çıkış üretir. Bir yapay sinir hücresi, ağırlıklı girişleri hesaplar. Her biri_k girişini, ayarlanabilir w_k ağırlıklarını ve thr seviyesini göz önüne alarak toplar (Eş. 1).

$$T = \sum_{k=1}^{n} w_k i_k + thr \tag{1}$$

Bu ağırlıklı toplam, doğrusal olmayan bir işlevden geçirilir (Eş.2).

$$y = \varphi(T) = \frac{e^{\lambda T} - e^{-\lambda T}}{e^{\lambda T} + e^{-\lambda T}} = \tanh(\lambda T) \tag{2}$$

Burada φ; genelde türevlenebilir ve monoton artan bir aktarım işlevi, λ; üstel ifadenin katsayısıdır. [24].

2.2.YSA'nın Yapısı

Çok katmanlı bir algılayıcıda (Multi Layer Perceptron-MLP) ağ mimarisi katmanlar şeklinde düzenlenmiştir. Katmanlarda bulunan her bir işlem birimi, komşu katmandaki birimlerle belirli bir ağırlıkla (w) bağlantı kurar. Bu mimaride bilgi akışı önceki katmanlardan sonraki katmanlara doğrudur. Bu nedenle bütün bağlantılar ileri doğrudur [3].

2.3. Geriye Yayılım Algoritması

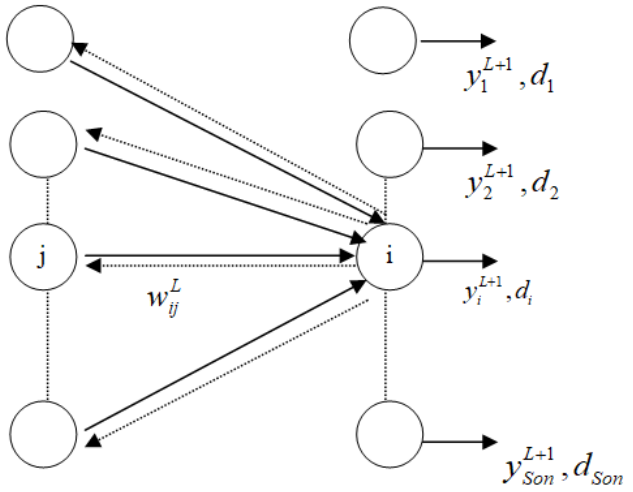
Genelleştirilmiş delta kuralı veya bilinen adıyla geriye yayılım algoritması diferansiyel zincir kuralına göre hata gradyen fonksiyonunu belirleyen bir yöntemdir [25]. Hata geriye yayma, çıkıştan ağrılıklara doğru zincir kuralına göre kısmi türevlerin alınmasıdır (Eş.8). Bu yönteme göre ileri yöndeki hesaplamalarda yapılan çıkış hatasının

karesel bir işlevi olan maliyet işlevinin (J_r) etkisi çıkış birimlerinden başlayarak katman katman geriye doğru yansıtılır [3]. $\{x(n), d(n)\}_{n=1}^{Son}$eğitim kümesi $x(n)$ giriş vektörü ve $d(n)$ istenen çıkış vektöründen oluşmaktadır. j. işlem biriminin çıkışı y_j ile temsil edilir.w_{ij} önceki katmandaki i. birimden sonraki katmandaki j. birime olan bağlantı ağırlığını temsil eder. Son katmandaki çıkışlar birden fazla ise çıkışların toplam hatası bir J_r maliyet işlevi ile temsil edilebilir (Eş.3). Çıkış katmanındaki hatanın geriye yayılımı Şekil 1'de verilmiştir. Burada L, ilgili işlem biriminin bulunduğu katman numarasını vermektedir [27];

y_i^{L+1}: L+1. katmandaki i. işlem biriminin çıkışı

w_{ij}^{L} : L+1. katmandaki i. işlem birimiyle k. katmandaki j. işlem birimini bağlayan ağırlık

T_i^{L+1}: L+1. katmandaki i. işlem biriminin toplam net girişi



Şekil 1. Çıkış katmanı için J_r'nin geriye doğru yayılımı [26].

3. İleri Sürümlü YSA ile Eğitim ve Geliştirme Aracı Tasarımı

Kuramsal konular, özellikle matematik içeren kavramlar, geleneksel olarak sınıf ortamında verilmektedir. Ancak mühendislik gibi kuramsal bilgilerin teknik uygulamalara dönüştürülmesi gereken meslek alanlarında etkileşimli eğitim yazılımları gibi araçların da öğrenim sürecine katılması kuramsal bilgilerin içselleştirilmesini kolaylaştırmaktadır.

Öğrenciler, etkileşimli grafik arayüzleri ile bir sistemdeki parametrelerin sonuçlara etkilerini anlık olarak görüp yorumlayabilmekte ve tasarıma müdahale edebilmektedir [27,28]. Bu kısımda öğrencilerin, verilen çeşitli giriş-çıkış giriş-çıkış örnekleri (örneğin bilinen bir fonksiyonun giriş-çıkış ilişkisi gibi) üzerinde eğitim, sınama ve uygulama algoritmalarını inceleyip geliştirebilecekleri bir YSA program altyapısı hazırlanmıştır. Algoritmalar, nesne yönelimli bir programlama dili olan Borland Delphi.6.0'da gerçeklenmiştir. Her bir yordamın dayandığı kuramlar ve ardından bunların arayüzde oluşturduğu görsel sonuçlar verilmiştir. Tasarlanan program, YSA Eğitim ve Geliştirme Aracı (YSA-EGA) olarak adlandırılmıştır.

3.1. YSA-EGA'nin Eğitim Aşamaları

Eğitim sürecinde, öncelikle YSA-EGA'nın eğitimi için programda gerekli olan sabitler, değişkenler, yordamlar ve işlevler tanımlanır. Sabitler, sırasıyla eğitimin kaç iterasyon süreceği, programın sorunsuz çalışabilmesi için hesaplama yapacağı gizli katman sayısının bir limitle sınırlandırılması gibi yapısal bilgileri içerir (Ek. A). YSA-EGA'nda ağın eğitimi için verilen giriş-çıkış çiftleri, programda eğitim çifti olarak adlandırılmıştır. Her iterasyon sonunda öğrenme adım büyüklüğü (η) yeniden ayarlanır. Maliyet fonksiyonu (J_r) bu ayarlamada belirleyici rol oynar. Bu fonksiyondaki değişim negatif çıkarsa, adım büyüklüğündeki değişme belirlenen gamma (γ) değeri kadar olacaktır. Ama fonksiyondaki değişim pozitif olursa, yani maliyet fonksiyonu artıyorsa, η deki değişme belirlenen beta (β) değerinin negatif değeri ile γ nın çarpımının değeri kadar olacaktır. Sonuç negatif geleceği için değişim negatif olacak ve böylece η küçülecektir [26].

Örnek çiftlere ait beklenen (d_i) ve gerçek (y_i) çıkış değerlerinin oluşturduğu hatalar toplanır. J_r istediğimiz tolerans değerine düşmemiş ise veya belirlenen iterasyon sayısına ulaşılmadı ise eğitime bir sonraki iterasyon ile devam edilir. Her iterasyon sonunda J_r, eski J_r ile güncellenir. Bu sayede bir önceki iterasyon hatasıyla şimdiki karşılaştırılarak eğitim sonunda elde edilen hatada artış ya da azalma olup olmadığı belirlenir. Buna göre eğitim adımının yönü ve büyüklüğü yeniden hesaplanır. Çok girişli çok çıkışlı bir sistem için o anki bir örnek eğitim çifti (giriş-çıkış çifti) vektörüne ait hatalardan J_r bulunur (Eş. 3) [29].

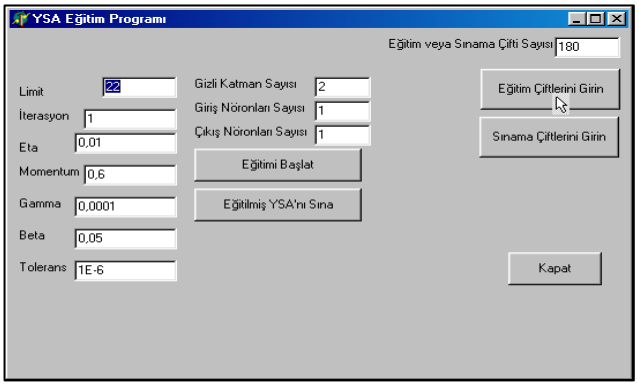
$$J_r = \frac{1}{2}\sum_{i=1}^{n_{k+1}} (d_i - y_i)^2 \tag{3}$$

Eğitim kümesine ait tüm örnek eğitim çiftlerinden elde ettiğimiz toplam hata ile "toplam küme hatası" belirlenir (Eş. 4).

$$E = \sum_{i=0}^{M} J_r(i) \tag{4}$$

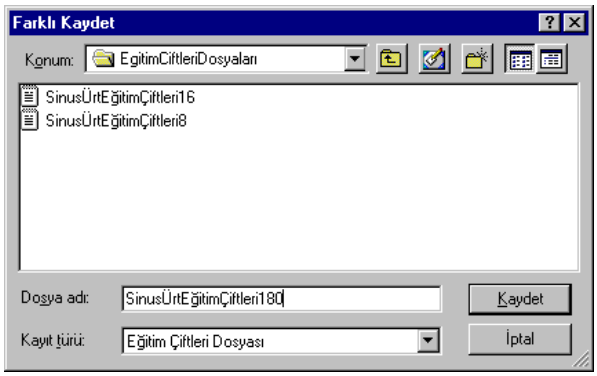
Burada M eğitim çifti sayısıdır. Küme hatası her iterasyonun sonunda eski küme hatasını güncellendikten

sonra sıfırlanır [29]. Katmanlarda işlem birimleri arasında bağlantı ağırlıkları Geriye Yayılım Algoritması ile güncellenir [26]. İleri yönde hesaplama; eğitim çiftleri kullanıcı tarafından bir dosyaya el ile girilebildiği gibi, benzetimi yapılacak işlev belli ise, program aracılığıyla istenen örnekleme aralığıyla otomatik olarak da oluşturulup dosyaya kaydedilebilir. Örneğin aşağıda bir sinüs işlevi, [0, 2π] aralığında otomatik örneklenip kaydedilmiştir (Şekil.2). Program arayüzünde "Eğitim Çiftlerini Girin" düğmesi ile ağı eğitecek eğitim çiftlerini oluşturan yordam etkinleştirilir.



Şekil 2. YSA-EGA Eğitim Programı ana menüsü üzerinde eğitim çiftlerinin oluşturulması.

Bu örnek için 0 ile 360º arasında 180 adet eğitim çifti örnek alınmış, üretilmiş ve bir dosyaya kaydedilmiştir (Şekil.3).



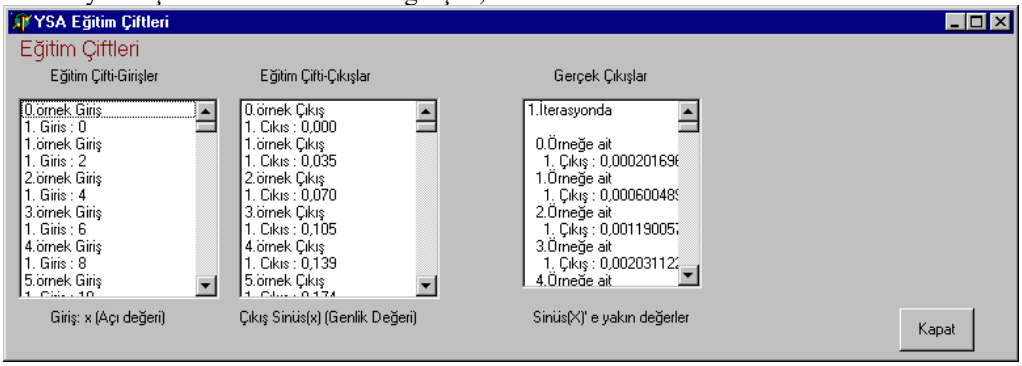
Şekil 3. Eğitim çiftlerinin oluşturulması ve kaydedilmesi.

Elimizde eğitim çiftleri bulunduğunda, YSA-EGA eğitim programını çalıştırarak ileri yönde hesaplamalara geçebiliriz. Ağ parametrelerini ileri yönde hesaplayan altyordama ait program kodları eklerde Şekil.A.1'de verilmiştir.

Bunun için algoritmada arayüze girilen "yapısal bilgiler", ağ parametreleri olarak YSA-EGA'na yüklenir. YSA'na ait yapısal bilgilerde; Eğitim Çiftlerinin tutulduğu dosyanın adı, gizli katman sayısı, giriş ve çıkış işlem birimlerinin kaçar adet olacağı, aynı şekilde her bir gizli katmanda kaçar adet işlem birimi olacağı gibi, önceden tanımladığımız sabitler bulunur. Ağla ilgili yapısal bilgiler Şekil 3.'teki menüde görüldüğü gibi programa girilir ve program başlatılır. "Eğitim Çiftlerini Yükleme" yordamında; ilk katmandaki (x[i j]) ile bunlara karşılık son katmanda istenen (d[i j]) eğitim çiftlerinin tutulduğu dosyadan sırayla hafızaya yüklenir. Ardından gizli katmanlarda kaçar adet işlem birimi olacağı sırayla girilir (Şekil 4).

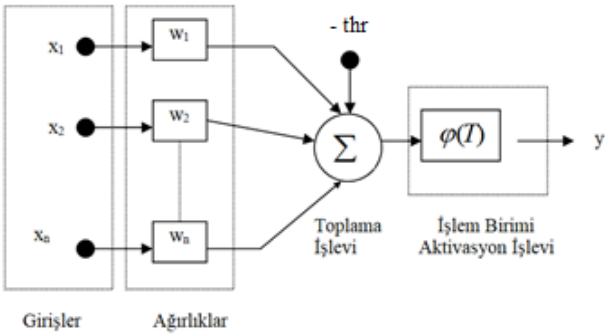
Şekil 4. Gizli Katman işlem birimi sayılarının programa girişi.

Eğitim çiftleri aynı sırayla "YSA Eğitim Çiftleri" Arayüzünde ekrana yazdırılır (Şekil.5) ve ağın eğitimi başlatılır. Her iterasyon için YSA'na verilen girişler, üretmesini istediğimiz çıkışlar ve ağın ürettiği gerçek çıkışlar görülmektedir.



Şekil 5. YSA-EGA'na girilen eğitim çiftleri ve ağın verdiği cevaplar.

Örnekte, ağa sinüs işlevi öğretilmektedir (Şekil.5 ve Şekil.9). Girişler açı ([0, 2π]), çıkışlar ise genliktir ([-1,1]). Eş.1'i Şekil 6 için yeniden yorumlayacak olursak; YSA girişleri (x_1, x_2…, xn), atanan ağırlıkları (w_1, w_2 …w_n) göz önüne alarak ileri yönde bir toplama işlevine gelir (Eş. 5), toplama sonucu bir aktivasyon işlevinden geçirilerek (Eş. 6) çıkış (y) hesaplanır (Şekil.8) [26,31]. Programlama sırasında eşik seviyesinin başlangıç değeri negatif *(-thr)* alınmıştır:



Şekil 6. Bir YSA işlem birimi [26].

$$T = w_1x_1 + w_2x_2 + ...w_nx_n - t = \left(\sum_{i=1}^{n} w_ix_i\right) - thr \quad (5)$$

$$y = \varphi(T) \quad (6)$$

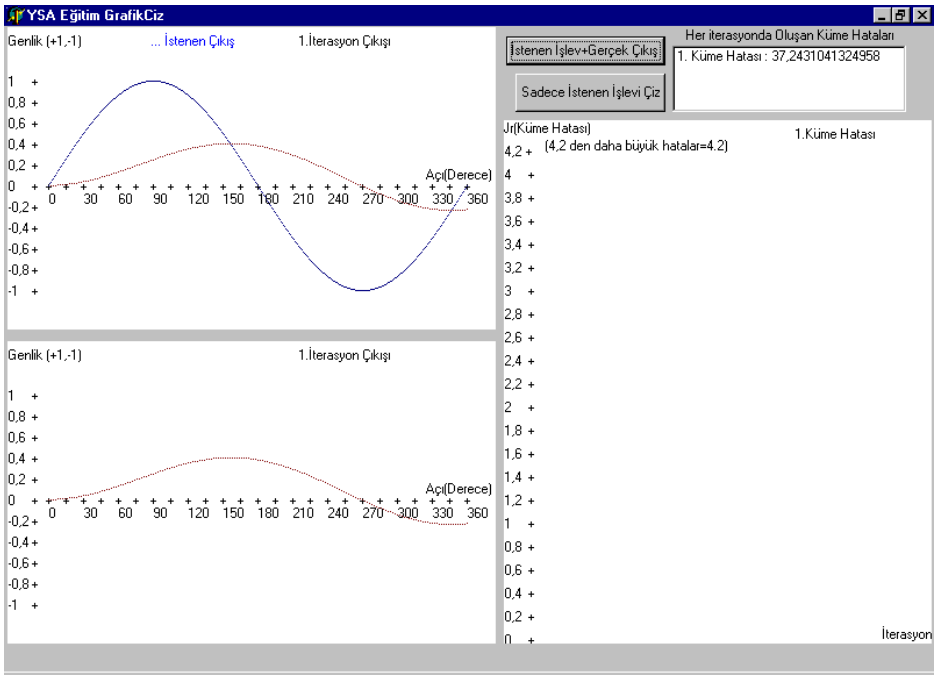
İşlem biriminin çıkışı, girişler, bağlantı ağırlıkları, eşik seviyesi ve doğrusal olmayan bir aktivasyon işlevinin bir işlevidir. Programda T ve y, sırasıyla Eş.5 ve Eş.6'dan hesaplanmış ve doğrusal olmayan aktivasyon işlevi olarak gizli katmanlarda tanh kullanılmıştır (Eş. 2). YSA'da, tüm işlem birimleri için ileri yönde hesaplamalar yapılarak, gerçek çıkışlar bulunur (Şekil.5). İleri yönde hesaplamalar tamamlanınca, çıkış katmanındaki çıkışlar kullanıcı arayüzüne sayısal olarak yazdırılır (Şekil.5) ve grafiği çizdirilir (Şekil.7). Hatanın geriye yayılımı ve ağırlık parametrelerinin güncellenmesi için sırasıyla çıkış ve gizli katmanlara ait yerel gradyenler (delta değerleri) hesaplanır [25].w değerindeki değişim yerel gradyen güncelleme kuralına göre yapılır (Eş. 7).

$$\Delta w = -\eta \nabla_w J_r \quad (7)$$

Burada ∇_w : w'ya göre kısmı türevdir. $\nabla_w J_r$ zincir kuralına göre tanımlanır (Eş. 8) [25].

$$\frac{\partial J_r}{\partial w_{ij}^L} = \frac{\partial J_r}{\partial y_i^{L+1}} \frac{\partial y_i^{L+1}}{\partial T_i^{L+1}} \frac{\partial T_i^{L+1}}{\partial w_{ij}^L} \quad (8)$$

Eğitim adımları boyunca, ağın cevabında meydana gelen iyileşme grafiklerle gösterilmiştir. Sol üstteki grafikte, ağın öğrenmesini istediğimiz işlev, tek başına ya da son eğitimde ağın verdiği gerçek çıkışlarla birlikte çizilebilmektedir. Bu tercih; "Sadece İstenen İşlevi Çiz" düğmesi veya "İstenen İşlev+Gerçek Çıkış" düğmesinden biri seçilerek sağlanmaktadır (Şekil.7).



Şekil 7. YSA-EGA eğitim grafikleri (1. İterasyon).

Eş. 8'teki parameter güncelleme denklemi; gerekli türev sadeleştirmeleri ve momentum düzeltme faktörleri eklenerek programda işler hale getirilmiştir (Eş. 9). Eşitliğin en sağında, programda kullanılan değişkenler verilmiştir.

$$\Delta w_{ij}^{L}(k+1)=\mu\Delta w_{ij}^{L}(k)+\eta\delta_i^{L+1}y_j^{L} = \text{MomentA+DIFF} \quad (9)$$

w değerleri güncellendikçe (Eş. 10) [3] YSA-EGA; daha gerçekçi sonuçlar bulmaya başlayacak ve işleve ait örüntü giderek orijinaline benzeyecektir (Şekil.10).

$$w_{ij}\ (k+1)=w_{ij}\ (k) + \Delta w_{ij}\ (k+1) \quad (10)$$

Parametre güncellenmesi ile ilgili altyordam eklerde Şekil.A.2'de verilmiştir. Sol altta ise gerçek çıkışlar eğitimin belirli adımlarında çizdirilerek ağın cevabında iyileşme olup olmadığı gözlemlenmektedir. Eşitlik 9'daki delta artımları çıkış ve gizli katmanlarda şu şekilde olacaktır: Çıkış katmanındaki yerel gradyenler, Eş. 11 ve Eş .12'deki delta artımları kullanılarak hesaplanır. Giriş katmanı için;

$$\delta_i^{L+1}\ \frac{-\partial J_r}{\partial T_i^{L+1}} \quad (11)$$

Çıkış katmanı için ;

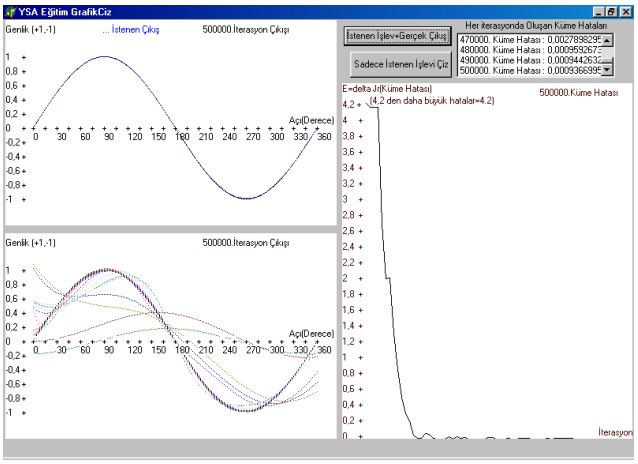
$$\delta_i^{L+1} = (d_i - y_i^{L+1})\phi'(T_i^{L+1}) \quad (12)$$

Gizli katman için ise parametre güncelleme Eş. 13 ile yapılmaktadır [26] ;

$$\delta_i^{L+1} = \left(-\sum_{h=1}^{n_{L+2}} \delta_h^{L+2}\, w_{\text{hi}}^{l+1} \right) \phi'(T_i^{L+1}) \quad (13)$$

Burada n; katmandaki işlem birimi sayısını ifade etmektedir. Delta (δ) artımlarının güncellenmesi ile ilgili altyordam eklerde Şekil A.3'te verilmiştir. Eşitlik 9'da kullanılarak tüm ağırlıklar güncellenir [25]. Delta değerlerinin Güncellenmiş ağırlıklar, istenen örüntüye gittikçe daha yakın çıkışlar sağlamaya başlar. Sağ tarafta, küme hatasının her iterasyonda ne kadar değiştiği görülmektedir. Ağırlık matrisi W en uygun değerlerine ulaşmaya başlayınca, hata da sıfıra oldukça yaklaşmaktadır (Şekil. 8).

Öğrenme süreci tamamlandıktan sonra veya küme hatası son iterasyon bitmeden istenen tolerans değerine ulaşırsa, eğitilmiş ağ parametreleri saklanmalıdır. Eğitilmiş ağ parametreleri, kullanıcının açtığında rahat biçimde okuyup bilgi alabilmesi için açıklamalı bir rapor şeklinde düzenlenmiş olarak bir dosyaya kaydedilir. Bu dosya sadece okuyucu için hazırlanmıştır. Daha sonra aynı parametreler, YSA-EGA denetim programı tarafından yüklenebilecek formatta, sıralı olarak ek bir dosyaya daha kaydedilir.



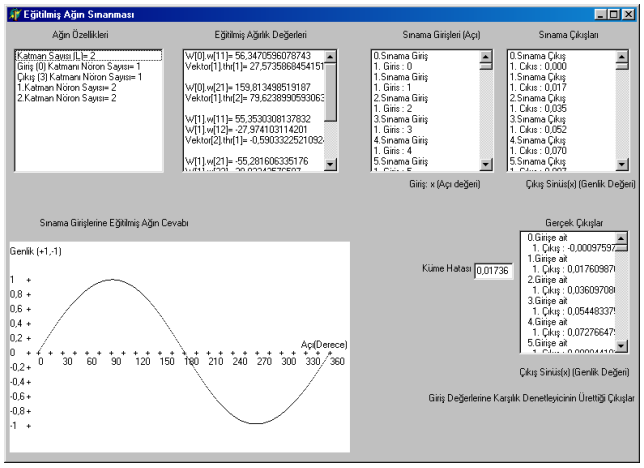
Şekil 8. YSA-EGA eğitim grafikleri (500 000. İterasyon).

3.2. Eğitilmiş YSA-EGA Parametrelerinin Sınanması

Eğitim sonunda YSA'nın başarımını gözlemlemek için test verileri ağa girilir. Bu test verileri, eğitim sırasında kulllanılan eğitim çiftleri dışında örneklerin bulunduğu, sınama çiftleri adını verdiğimiz bir dizi veridir.

Sınama çiftleri bir sistemden alınabildiği gibi program tarafından da üretilebilir. Bunun için, Şekil.2'deki "Sınama Çiftlerini Girin" düğmesi altında farklı kodlamalar geliştirilebilir. Bu kodlamalar, örüntüsü bulunmak istenen işleve ait verileri, işlevin kendinden üretmeye yönelik de olabilir. Bu örneğimizde; seçilen düğmenin altındaki altyordam, sinüs işlevine [0-2π] aralığında eşit aralıklı 360 farklı açı değeri girer. Buna karşılık, aynı sayıda genlik değeri elde eder ve bunları sınama çifti olarak dosyaya kaydeder. Sınama altyordamı şu şekilde çalıştırılır; daha önce eğitilmiş olan ağ parametrelerini yüklenir, istenen sınama verileri ile ağ test edilir (Şekil.2). Bu örnekte YSA, sınama verilerinin yarısını bilmemektedir. Ağın girişler için doğru çıkışları sağlayıp sağlayamadığını gözler. Öncelikle sınanacak YSA-EGA parametreleri, dosyadan yüklenir. Ardından önceden kaydedilmiş sınama çiftleri yüklenir ve arayüzde gösterilir (Şekil.21): Giriş Katmanı ve Çıkış Katmanındaki işlem birimlerinde sınama için kullanılacak değerler sırayla hafızaya yüklenir. Programın sınama arayüzünde, kullandığımız YSA'na ait yapısal bilgiler, eğitilmiş parametrelerin değerleri rapor edilir. Sınama girişleri, bu girişlere karşılık olması istenen çıkışlar, ağın gerçek çıkışları ve istenen çıkışlarla gerçek çıkışlar arasındaki toplam küme hatası (E) ekrana

yazdırılır. Burada E, YSA'nın nesnel bir başarım ölçütü olarak kullanılır. Eğitilmiş ağın sınama girişlerine YSA'nın gerçek cevabı grafiksel olarak çizdirilir (Şekil.9).



Şekil 9. Eğitilmiş YSA'nın sınama sonuçları.

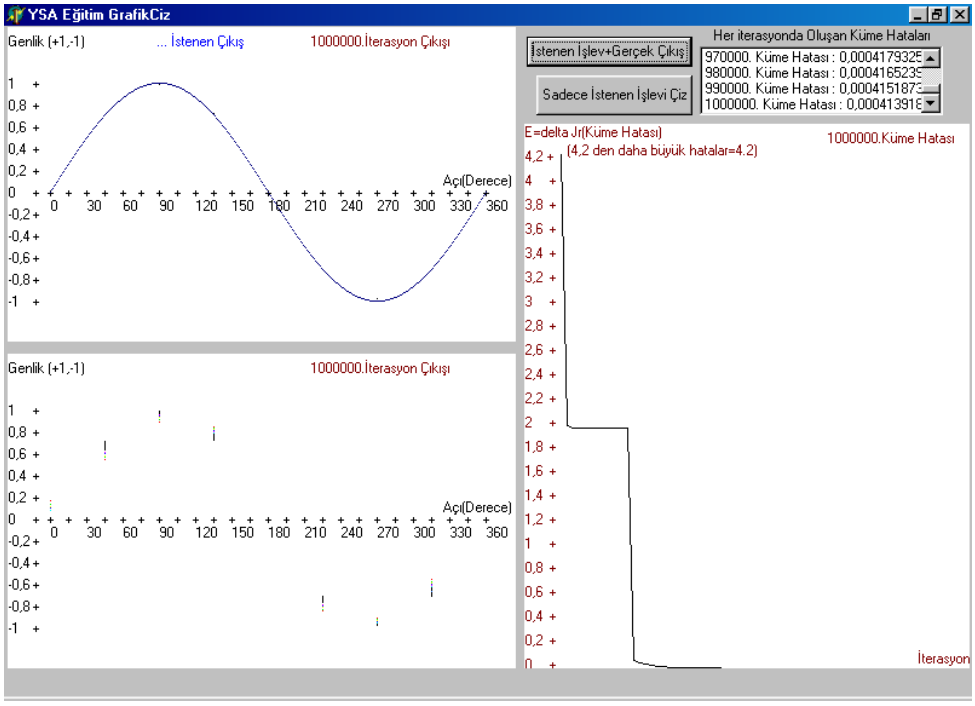
Sınama yordamında girdilerden çıkışlara doğru, ileri yönde hesaplama bir kez yapılır ve Eş.6'daki doğrusal olmayan aktivasyon işlevleri programda Eş. 14'deki biçimde ve gizli katmana ait yerel gradyenlerde kullanılmak üzere aktivasyon işlevlerinin türevleri Eş. 15'deki biçimde bir kez daha hesaplanır [26].

$$\varphi(T) = \tanh = \tanh\left(\left(\sum w_{ij} y_j\right) - \mathrm{thr}(i)\right) \quad (14)$$

$$\varphi'(T) = 1 - \left(\left(\sum w_{ij} y_j\right) - \mathrm{thr}(i)\right)^2 \quad (15)$$

3.3. YSA-EGA'nün Daha Az Eğitim Verisi ile Eğitimi ve Sınanması

Benzer bir eğitim ve sınama denemesini, eğiteceğimiz sisteme ait elimizde çok daha az sayıda veri ile yapacak olursak, ağın başarımı nasıl etkilenir? Bu kısımda, eğitim için 8 örnek eğitim çifti kullanılmıştır (Şekil.3). Bu durumda; sinüs işlevine ait bildiğimiz 8 noktaya ait açı ve genlik bilgileriyle ağ eğitilir. Eğitim sonunda 8 noktanın da istenen konuma geldiği görülmektedir. Nitekim, istenilen sinüs işlevi, YSA-EGA'nın eğitim sonunda bulduğu noktalardan geçmektedir (Şekil.10).

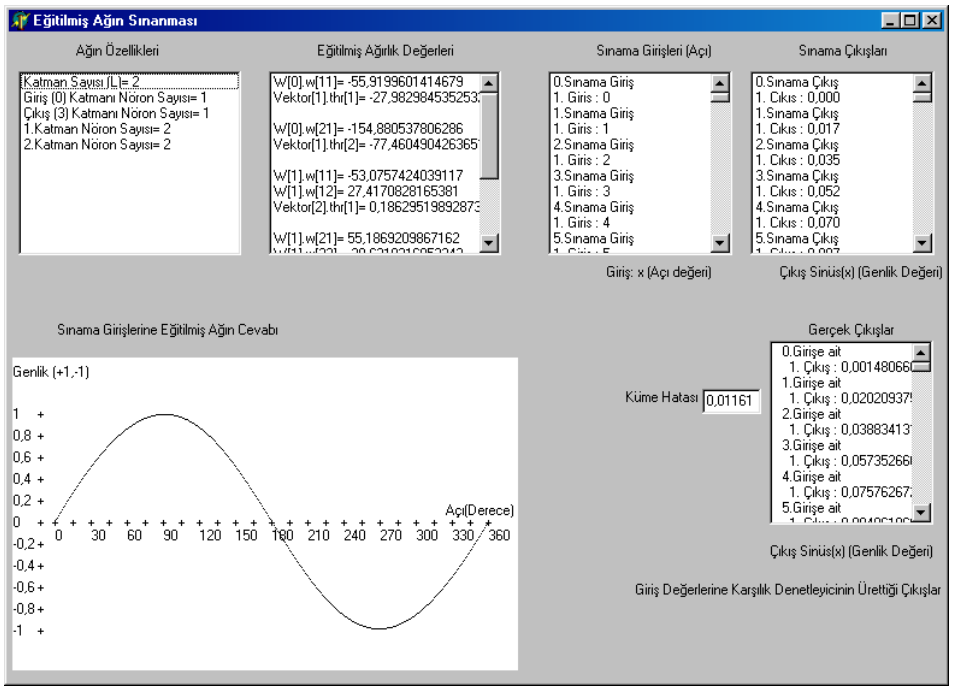


Şekil 10. YSA-EGA eğitim adımları.

Şekil.11'de, 8 örnekle eğitilmiş ağın 0 ile 360º arasındaki sınama girişlerine cevabı görülmektedir. Bu arayüz, ayrıca YSA'na ait özellikler, parametreler ve giriş çıkış değerlerini de raporlamaktadır. 8 örnekle yapılan eğitimde Küme Hatası 0.01161 çıkmıştır. Programın sınama arayüzü sınama girişlerine eğitilmiş ağın cevabını çizdirmektedir.

İleri sürümlü YSA'nda eğitim, bireysel veya toplumsal olabilir [26]. Bu çalışmada toplumsal küme hatası kullanılarak, tüm eğitim kümesinin toplam hatası en küçük hale getirilmeye çalışılmıştır. Kullanıcı Ek. A' daki en dış döngüyü kaldırarak, her eğitim çiftine göre bireysel hata hesaplayıp, ağ parametrelerini buna göre de güncelleyebilir. Böylece bireysel eğitim yöntemini de deneyebilir. Eşitlik.14'deki aktivasyon fonksiyonunu değiştirip, problem özgü, sigmoid veya sert geçişli gibi başka tip bir işlev kullanabilir. İleri sürümlü YSA'da ağırlıklar, L. katmana ait j. birim ile onu takip eden L+1. katmandaki i. birim arasında hesaplama yapmaktadır. Döngülerde değişiklik yapılarak, L. katman daki i. birim

ile kendi katmanındaki diğer birimler arasında da hesaplama yapılabilir. Programda döngüye ait sorgulanan indis değiştirilerek, L(k) ile (k+1) katmanlardaki işlem birimleri arasında bilgi akışı olabileceği gibi, L(k) 'dan L(k-1)'e geri yönde bir hesaplama ve bilgi akışı da gerçeklenebilir. İşlem birimi içerisindeki toplayıcıya ileri yönde gelen verilere ek olarak, geri bildirim verileri de dahil edilerek, dinamik yapıda bir YSA elde edilebilir. Gizli katmanlar, arayüzde 1 olarak seçip, yazılımda aktivasyon işlevini Gauss işlevi olarak değiştirerek, Gasuss Merkezcil YSA elde edilebilir.

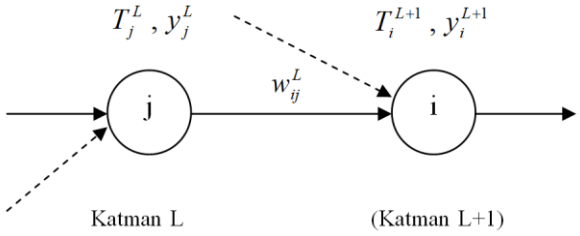


Şekil 11. 8 örnekle eğitilmiş YSA'nin sınama sonuçları.

3.4. YSA-EGA'nün Uygulama Yordamı ve Arayüzü

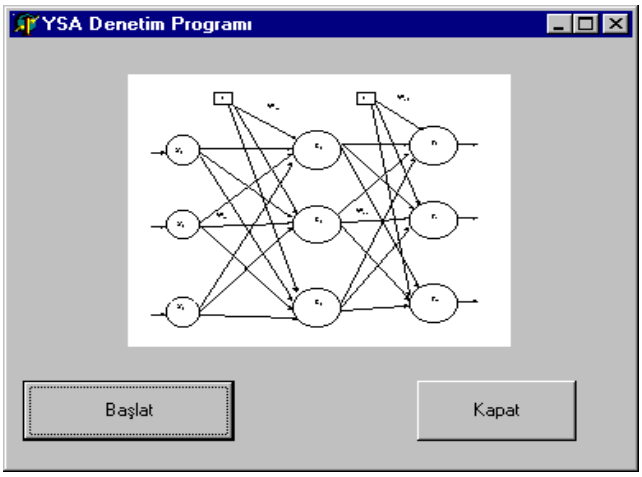
Programda Uygulama Yordamı, Sınama Yordamı ile aynı altyapıyı kullanmaktadır. Eğitilmiş YSA-EGA "Uygulama"da kullanılacak girdilere göre ileri yönde bir kez hesaplama yapmaktadır. Programda, ileri yönde veri akışı için tanımlanan değişkenler şu şekildedir:

Katmanların sırasını gösteren indis L olarak tanımlansın ve n adet gizli katmanımız bulunsun [3] (Bose ve Liang,1996). Burada Katman L+1'in i. işlem birimini, Katman L'nin j. işlem birimine birleştiren bağlantı ağırlığı w_{ij}^{L} olarak verelim. Katman L'nin j. işlem birimi çıkışı y_j^L'dir. Takip eden katman L+1'deki i. toplamı T_i^{L+1} ve çıkış şeklinde tanımlanır.



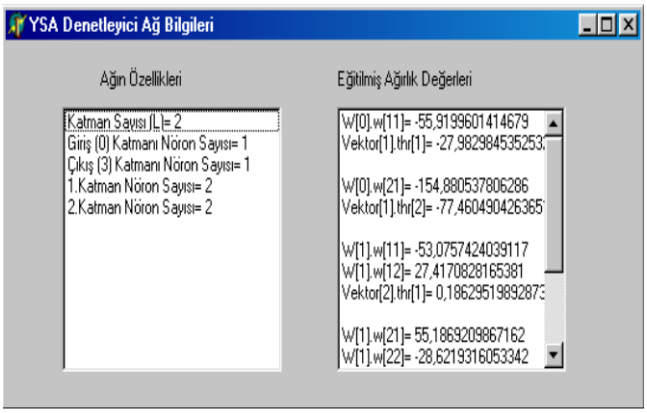
Şekil 12. Program için hazırlanan işlem birimleri ve içinde bulundukları ilgili katmanların sıralama indislerinin tanımlanması [3].

YSA-EGA'da geliştirilen Uygulama Yordamı, girişlere bağlı olarak eğitilmiş YSA'nın çıkışlarını denetlediği için programda "YSA Denetim Programı" arayüzü ile temsil edilmiştir (Şekil.13). Uygulama, doğrusal olmayan bağıntılar kurmayı gerektiren her çalışma için geliştirilebilir ve farklı isimler alabilir.



Şekil 13. YSA Denetleyici başlangıç menüsü.

"Başlat" düğmesi ile eğitilmiş ağ parametrelerinin bulunduğu dosya açılır. Dosyadan, katman sayısı, katmanlardaki işlem birimi sayıları gibi ağa ait yapısal bilgiler ve eğitilmiş w[i,j] ağırlık değerleri yüklenir (Şekil.14). Benzer şekilde her katmandaki işlem birimine ait bilgiler bir vektör elemanı olarak kayıt tipi [22] bir değişkende saklanır (Tablo 1).



Şekil 14. Kullanılan YSA denetleyicinin özellikleri.

Tablo 1. Kullanılan İşlem Birimi Bilgileri ve Programda Saklanma Biçimi (Informations Of the Neurons and Their Storage Formats in The Program)

Değişken	Açıklaması
Vektor	Bir katmandaki i. nörona ait bilgiler (i=[1.Limit])
y[i]:	Çıkış değeri
Delta[i]:	Delta değeri
thr[i]	Eşik değeri
ToplamJ[i]	İşlem birimine gelen girişlerin net toplamı

Okunan giriş değerlerine karşılık, ağın ileri yönde çıkış değerleri hesaplanır, denetleyicinin çıkışları ekrana yazdırılır ve grafik olarak çizdirilir. "İleri Yön Değerlerini Hesapla" altyordamında (Ek. A), her bir işlem birimi için önce Eş. 5'te verilen net toplam bulunur. Toplam değişkeni programda ToplamJ olarak tanımlanmıştır.

Tablo 2. "İleri Yön Değerlerini Hesapla" Altyordamında kullanılan değişkenler.

Değişken	Açıklaması
ToplamJ	Bir önceki katmandaki her bir işlem birimi için net toplam. Denklem.1'de genel formülü verilen toplam işlevi, programda $\left(\sum w_{ij} y_j\right) - \text{thr}(i))$ olarak geçmektedir.
W(L).w(ij)	Katman L'deki j. işlem birimi ile bir sonraki katmandaki i. işlem birimi arasındaki ağırlık değerlerinin (w(ij)) tümü, W(L) ağırlık matrisinde saklanır.

Tablo 2. (Devam) "İleri Yön Değerlerini Hesapla" Altyordamında kullanılan değişkenler.

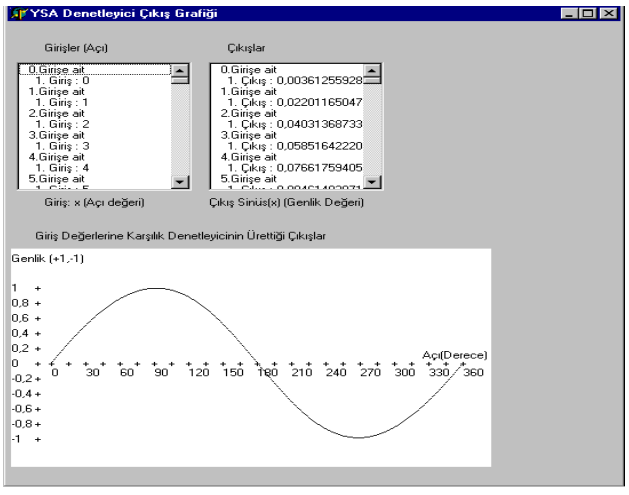
Değişken	Açıklaması
y_j	Bir önceki katmandaki ilgili işlem birimi çıkışı; Programda $\varphi(T)$ = Vektor[L].y[j] formülü ile hesaplanmaktadır.
T	Bir sonraki katmanın i. işlem birimine ait net toplam giriş. Programda Vektör[L+1].ToplamJ[i] formülü ile hesaplanmaktadır.

Programda belirlediğimiz W(L). w(ij) ağırlık değerleri (Eş. 10) bir önceki katmandaki y_j (Eş. 6) ile çarpılır. Bütün ağırlıklı çıkışlar toplanır. Eşik seviyesi de eklenerek bir sonraki katmanın i. işlem birimine ait net toplam giriş (Eş. 5) bulunur. Bu giriş, işlem biriminin bulunduğu katman gizli katman ise doğrusal olmayan geçiş işlevinden, son katman ise doğrusal geçiş işlevinden geçirilir (Tablo 3). Böylece çıkışlar hesaplanır.

Tablo 3. Programda geçiş işlevlerinin tanımlanması.

Değişken	Açıklaması
$\varphi(ToplamJ)$	Doğrusal olmayan geçiş işlevi: Programda; $\tanh\left(\left(\sum w_{ij} y_j\right) - \text{thr}(i)\right)$ formülü ile temsil edilmiştir.
$\varphi(ToplamJ)$	Doğrusal geçiş işlevi: Programda $=\left(\sum w_{ij} y_j\right) - \text{thr}(i)$ formülü ile temsil edilmiştir.

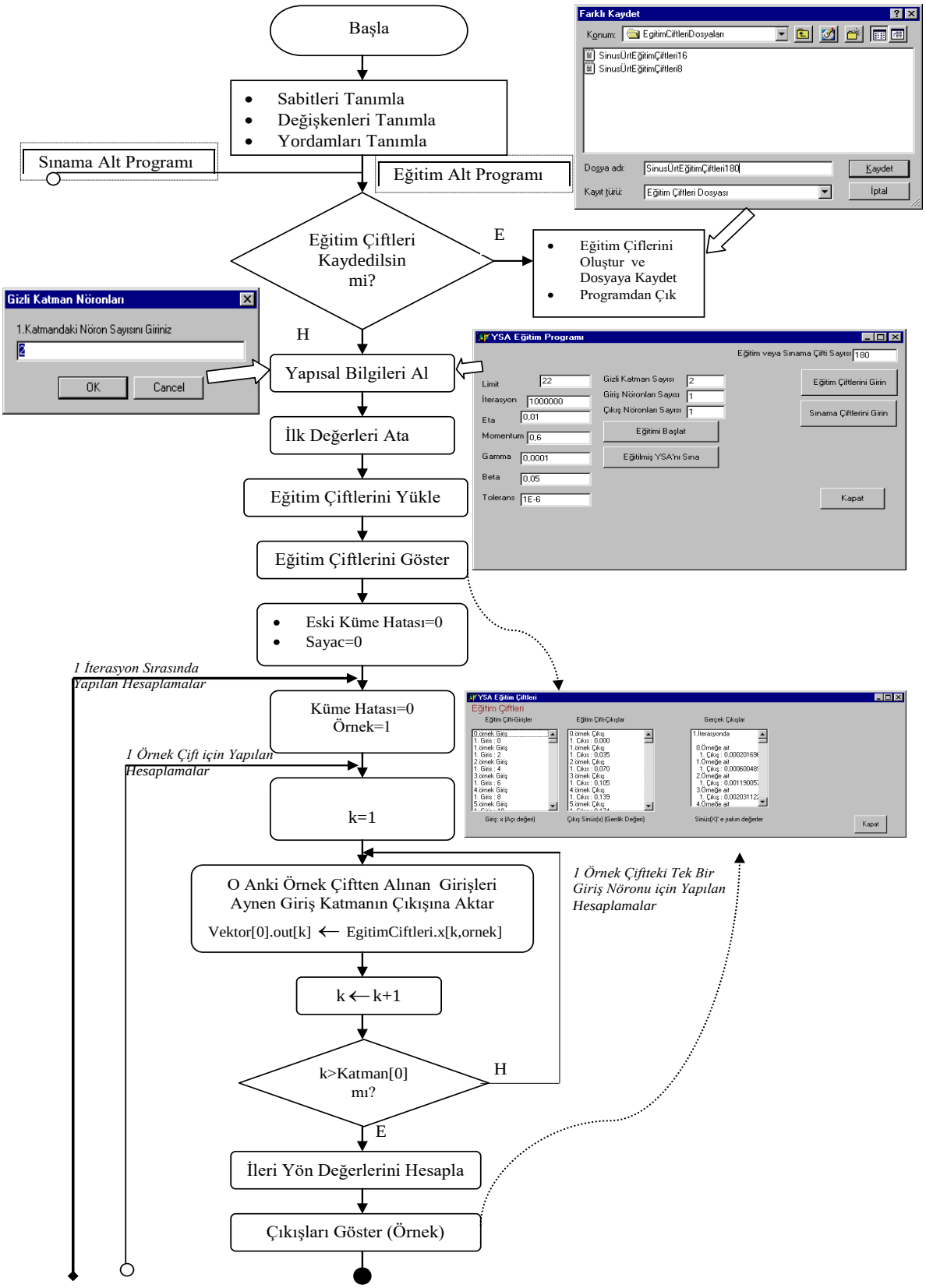
Son olarak "Çıkışları Göster" Altyordamı ile 'İleri yön değerlerini hesapla' altyordamında hesaplanan çıkışlardan son katmana ait çıkışlar (Vektor[L+1]. y[j]) ekrana yazdırılır. Grafik olarak çizdirilir. Aynı işlem, her eğitim çifti için tekrarlanır (Şekil.15).

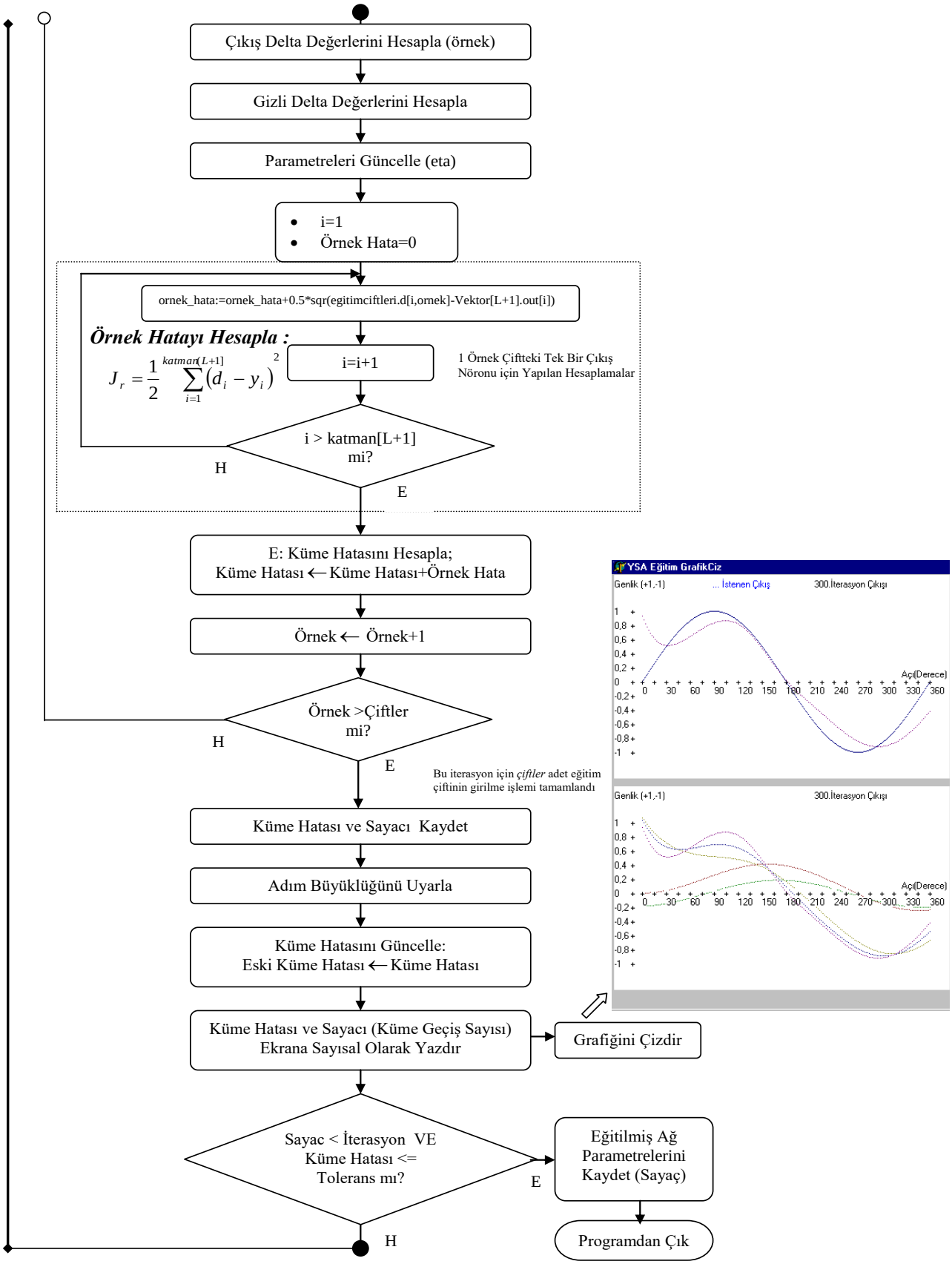


Şekil 15. Giriş değerlerine karşılık denetleyicinin ürettiği çıkışlar.

Bu örnekte açı girişlerine YSA-EGA'nin yanıtı gerçek bir bir sinüs işlevine oldukça benzediği görülmektedir.

Şekil.16'da YSA arayüz yazılımının (GUI) akış şeması verilmiştir.





Şekil.16. Eğitim altyordamının akış şeması.

Akış şeması bu çalışmada hazırlanan eğitim, sınama algoritmalarındaki aşamalarla geliştirilen arayüz arasındaki ile ilişkileri özetlemektedir. Kullanıcı, akış şemasında hangi aşamada arayüzün hangi kısmına erişildiği, arka planda YSA eğitiminin nasıl çalıştığı, hangi noktada hangi döngüler üzerinden eğitim çiftlerine, katmanlara erişildiği, parametre artım değerlerini hesaplandığı ve buradan ağırlık parametrelerinin güncellendiğini sıralı olarak görebildiğinden, tespit ve müdahale edebilecektir. Güncellenmiş parametreler üzerinden ileri yönde hesaplanan çıkış veya çıkışların grafik olarak arayüzde çizdirileceği aşama akış şemasında tarif edilmiştir. Bu aşamaların klavuzluğunda, kullanıcı müdahale edip, değiştirebileceği yerleri tespit edebilmektedir.

4. Sonuçlar

Bu çalışmada öğrenciler için açık kaynak kodlu bir YSA eğitimi ve program geliştirme aracı önerilmiştir. Tasarlanan YSA-EGA'nın, bir sisteme ait doğrusal olmayan giriş-çıkış bağıntılarını yaklaşık olarak çıkarsayabildiği gösterilmeye çalışılmıştır. Örnek olarak bir sinüs işlevinin örüntüsü, önce çok örnek, daha sonra az örnekle eğitilmiştir.

Çalışmada YSA-EGA'nın eğitim, sınama ve uygulama kiplerinde çalışması açıklanmış ve hazırlanan programın arkasındaki kuram verilmiştir. Örnek sinüs işlevinin YSA tarafından oldukça doğru bir biçimde öğrenildiği gerek görsel gerek sayısal sonuçlarda görülmektedir (Şekil.9-11). YSA-EGA'nın başarımında sayısal ölçüt olarak, tüm eğitim çiftleri kümesine ait toplumsal maliyet işlevi olan küme hataları (Eş.4) kaynak alınmıştır ve çalışmada yapılan her iki örnek eğitim çifti kümesi için başarımları karşılaştırılmıştır (Tablo.4).

Tablo.4 Eğitim ve sınama sonuçları.

	Eğitim Çifti	Sınama Çifti	Gizli Katman	Her Gizli Katmandaki İşlem Birimi	İterasyon Sayısı	Küme Hatası	
						Eğitim	Sınama
1	180	360	2	2	500.000	0.0009366995	0.01736
2	8	360	2	2	1.000.000	0.000413918	0.01161

Sinüs gibi eksponansiyel artış ve azalışlar gösteren örüntülerde öğrenme başarımı yüksek olmaktadır. İlk eğitim, 180 veri ile yapılmıştır. Küme hatası 500.000 iterasyondan sonra değişmemiş, 0.0009366995 civarında kalmıştır. Tüm eğitim kümesine ait toplam maliyet işlevi, kümeyi oluşturan eğitim çiftleri fazla olduğu ve her birinin hatası kümeye katıldığı için görece daha yüksek çıkmıştır. 8 örnekle yapılan ikinci eğitimde öğrenme 1.000.000 iterasyona kadar sürmüş ve küme hatası 0.000413918 çıkmıştır. Çok ve az örnekle eğitilen iki YSA, 360 veri ile sınanmıştır. Az örnekle eğitilen YSA, buna rağmen sınamada daha düşük küme hatası (0.01161) oluşturmuştur (Tablo.4). Önerilen YSA-EGA, düzenli değişim gösteren örüntülerde çok az veri ile oldukça yüksek doğrulukta çıkarsama yapabilmektedir. Geliştirilen yazılım altyapısı, eğitim ve üzerinde çeşitli uygulamalar geliştirmeye uygun olarak hazırlanmıştır. Bu çalışmada olduğu gibi öğrencilerin kolay adapte olabileceği etkileşimli Türkçe Arayüzler hazırlanabilir. Benzer şekilde uluslar arası çalışmalar için yabancı dilde farklı YSA Arayüzleri geliştirilebilir.

YSA algoritmalarının açık kaynak kodu üzerinde gerçeklenmesi, yeni tasarımların önünü açacaktır. Bu çalışma ile ilgili kaynak kodları internet ortamında paylaşılmıştır [32]. Bu çalışma, YSA'nın eğitim, sınama ve uygulama aşamalarının nasıl gerçeklenebileceği ile ilgili çalışır durumda yordamlar ve işlevler sunmakta ve açıklamaktadır. Açık kaynak kodlu olduğu için, kullananlar, esinlendikleri aşamalara kendi bilgilerini katarak çok farklı çalışmalar gerçekleştirebilir.

Kaynaklar

[1] Majors M., Stori J., Cho D., 1994. Neural network control of automotive fuel-injection systems. IEEE Trans Control Syst, 14(3), 31–36.

[2] McCulloch W.S., Pitts W., 1943. A Logical Calculus of the Ideas Imminent in Nervous Activity. The Bulletin of Mathematical Biophysics, 5, 115-133.

[3] Bose N.K., Liang P., 1996. Neural Network Fundamentals with Graphs, Algorithms and Applications, Mc Graw Hills Series in Electrical and Computer Engineering.

[4] Karri V, Ho TN., 2009. Predictive models for emission of hydrogen powered car using various artificial intelligent tools. Neural Comput & Applic 18, 469–476.

[5] Kişi Ö., 2004. River flow modelling using artificial neural networks. Journal of Hydrologic Engineering, 9(1), 60-63.

[6] Arora N., 2009. Regulating air-fuel balance in combustion engines using adaptive learning in neural network. In: Proceedings of the international conference on methods and models incomputer science, Delhi, India, pp 1–6, Aralık 2009.

[7] Haehn D., Tompkin J., Pfister H., 2019. Evaluating 'Graphical Perception' with CNNs. IEEE Transactions on Visualization and Computer Graphics, 25(1),641-650.

[8] Jaafar K., Ismail N., Tajjudin M., Adnan R., Rahiman, M.H.F., 2016. Hidden Neuron Variation in Multi-layer Perceptron for Flood Water Level Prediction at Kusial Station. 2016 IEEE 12th International Colloquium on Signal Processing & its Applications (CSPA2016), Melaka, Malaysia, 4-6 Mart 2016.

[9] Patil K., Jadhav N., 2017. Multi-Layer Perceptron Classifier and Paillier Encryption Scheme for Friend Recommendation System. Third International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India.

[10] Syed M.A., Kumara Y.S., 2015. Web Service classification using Multi-Layer Perceptron optimized with Tabu search. 2015 IEEE International Advance Computing Conference (IACC), pp. 290-294, Bangalore, India.

[11] Puhan P.S., Behera S., 2017. Application of Soft Computing Methods to Detect Fault in A.C Motor. International Conference on Advances in Computing, Communication and Control (ICAC3), pp. 1-5, Mumbai, India.

[12] Kumari U., 2017. Soft computing applications: A perspective view, 2nd International Conference on Communication and Electronics Systems (ICCES), pp. 787-789, Coimbatore, India.

[13] Rosenberg L., Baltaxe D., Pescetelli N., 2016. Crowds vs swarms, a comparison of intelligence, Swarm/Human Blended Intelligence Workshop (SHBI), pp. 1-4, Cleveland, Ohio, USA.

[14] Corbett F.D., Card H.C., 1998. Java Tools for Research and Education in Artificial Neural Networks, IEEE Canadian Conference on Electrical and Computer Engineering, Vol.1, pp. 417-420, Ontario, Canada.

[15] Nasr G.E., Joun C., Zaatar W.A., 2004. GUI Based Artificial Neural Network Simulator. 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL, pp.135-138, Belgrade, Serbia and Montenegro.

[16] Manic M., Wilamowski B., Malinowski A., 2002. Internet Based Neural Network Online Simulation Tool. IECON 02 IEEE 2002 28th Annual Conference of the Industrial Electronics Society, Volume: 4, pp.2870-2874, Sevilla, Spain.

[17] Ramirez M.R., Brar P.S., 1993. Novel Uses of Neural Networks, IEEE FIE 93 Frontiers in Education Conference, pp.710-713, Washington, DC, USA.

[18] He L., Li H., Zhang Q., Sun Z., 2019. Dynamic Feature Matching for Partial Face Recognition. IEEE Transactions on Image Processing, 28(2), 791-802.

[19] Liu X., Wang S., Zhang W., Li Q., 2018. Research of Nonlinear Adaptive Control Based on BP Neural Network. 10th International Conference on Modeling, Identification and Control (ICMIC), pp.2-4, Guiyang, China.

[20] Han N,. Gao S., Li J.,Zhang X., Guo J., 2018. Anomaly Detection in Health Data Based on Deep Learning. Proceedings of IC-NIDC, the IEEE International Conference on Network Infrastructure and Digital Content, pp.188-192, Guiyang, China.

[21] Shaozhong Z., Juqin Y., 2010. Flux and Level Prediction based on An Wavelet Neural Network Flood Model. 3rd International Symposium on Knowledge Acquisition and Modeling, pp. 67-70, Wuhan, China.

[22]https://admhelp.microfocus.com/uft/en/14.51/UFT_Help/Subsystems/FunctionReference/Subsystems/OMRHelp/Content/Delphi/DELPHIDOCSLib_P.html, (Erişim Tarihi: 11 Kasım 2018).

[23] Hsu K.L., Gupta H.V., Sorooshian S., 1995. Artifical neural network modeling of the rainfall-runoff process. Water resource research, 31(10), 2517-2530.

[24] Kosko B., Neural Networks and Fuzzy Systems, Prentice-Hall International Editions, 1992.

[25] Moon T.K., Stirling W.C., Mathematical Methods and Algorithms for Signal Processing, Prentice Hall Inc., 2000.

[26] Efe M.O., Kaynak O., 2000. Yapay Sinir Ağları ve Uygulamaları. Boğaziçi Üniversitesi.

[27] Johansson M., Gafvert M., Astrom K.J., 1998. Interactive tools for education in automatic control. IEEE Control Systems, 18(3), 33-40.

[28] Shiakolas P.S., Piyabongkam D., 2003. Development of a Real-Time Digital Control System With a

Hardware-in-the-Loop Magnetic Levitation Device for Reinforcement of Controls Education. IEEE Transactions on Education, 46(1), 79-87.

[29] Aşkın D., İskender İ., Mamızadeh A., 2011. Farklı Yapay sinir ağları yöntemlerini kullanarak kuru tip transformatör sargısının termal analizi. Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi, 26(4), 905-913.

[30] Kulkarni A.D., 2001. Computer vision and fuzzy-neural systems, Prentice Hall, NJ, USA.

[31] Bulut M., Başoğlu B., 2017. Kısa Dönem Elektrik Talep Tahminleri İçin Yapay Sinir Ağları ve Uzman Sistemler Tabanlı Hibrid Tahmin Sistemi Geliştirilmesi. Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi, 32(2), 575-583.

[32] https://avesis.kocaeli.edu.tr/serhaty/dokumanlar, (Erişim Tarihi: Mayıs 2020).