T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İÇİN WEB TABANLI BİR EĞİTİM YAZILIMI GELİŞTİRİLMESİ

Mehmet BİLEN

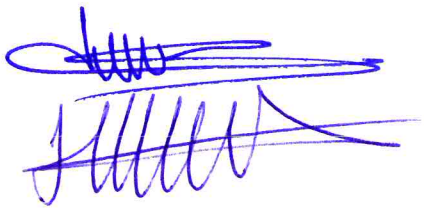
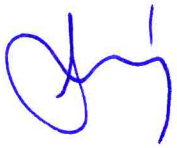
Danışman
Doç. Dr. Tuncay YİĞİT

II. Danışman
Yrd. Doç. Dr. Ali Hakan IŞIK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA - 2014

TEZ ONAYI

Mehmet BİLEN tarafından hazırlanan **"Yapay Sinir Ağları için Web Tabanlı bir Eğitim Yazılımı Geliştirilmesi"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Doç. Dr. Tuncay Yiğit** Süleyman Demirel Üniversitesi	
II. Danışman	**Yrd. Doç. Dr. Ali Hakan IŞIK** Mehmet Akif Ersoy Üniversitesi	
Jüri Üyesi	**Yrd. Doç. Dr. Göksel Aslan** Mehmet Akif Ersoy Üniversitesi	
Jüri Üyesi	**Yrd. Doç. Dr. İsmail Serkan ÜNCÜ** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Doç. Dr. Ömer DEPERLİOĞLU** Afyon Kocatepe Üniversitesi	

Enstitü Müdürü **Prof. Dr. Ahmet ŞAHİNER**

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Mehmet BİLEN

İÇİNDEKİLER

Sayfa

ÖZET

Yüksek Lisans Tezi

Yapay Sinir Ağları için Web Tabanlı bir Eğitim Yazılımı Geliştirilmesi

Mehmet BİLEN

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Tuncay YİĞİT

II. Danışman: Yrd. Doç. Dr. Ali Hakan IŞIK

Yapay sinir ağları doğrusal olmayan problemlerin çözümünde kullanılan önemli bir veri işleme algoritmasıdır. Matematiksel altyapısı ve içerdiği karmaşık denklemler, yapay sinir ağlarının geleneksel yaklaşımlar ile öğrenilmesini zorlaştırmaktadır. Bu zorlukların aşılmasında etkileşimli ve çoklu ortam destekli yazılımlar kullanmak gerekmektedir. Tez çalışmasında yapay sinir ağları için web tabanlı yazılımı etkili ve verimli öğrenme ortamı sağlamak için geliştirilmiştir. Bu etkileşimli araç aynı zamanda canlandırmalar ve metin tabanlı ders içerikleri ile zenginleştirilmiştir. Buna ek olarak, yapay sinir ağları parametrelerindeki değişimlerin etkileri grafiksel sonuçlar sayesinde doğrudan gözlemlenmiştir. Bu sayede kullanıcılar yapay sinir ağlarının çalışma mekanizmasını kolaylıkla anlayabilmektedir. Yazılım hiç bir ticari kütüphane kullanılmadan, nesne yönelimli programlama dili olan C# ile geliştirilmiştir. Diğer yazılımlarla ortak çalışması için XML, TXT ve CSV gibi dosya formatları desteklenmiştir. Yazılımın performansını değerlendirmek için BalanceandScale veri kümesi kullanılmıştır. 0,9918 doğruluk, 1 belirlilik ve 1 duyarlılık değerleri elde edilmiştir. Bu çalışma literatürdeki çalışmalarla kıyaslandığında görsellik, anlaşıla bilirlik ve etkileşim bakımından ilerleme sağlamıştır

Anahtar Kelimeler: yapay sinir ağları, eğitim yazılımı, c#

2014, 74 sayfa

ABSTRACT

M.Sc. Thesis

Development of Educational Software for Artificial Neural Networks

Mehmet BİLEN

Süleyman Demirel University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Tuncay YİĞİT

Co-Supervisor: Asst. Prof. Dr. Ali Hakan IŞIK

Artificial neural networks are important data processing algorithms which are used for solving nonlinear problems. Mathematical infrastructure and complex equations make artificial neural networks difficult to understand by classical approaches. Interactive and multimedia-enabled software should be used to overcome these difficulties. In this thesis, a web based educational software for artificial neural networks (ANNs) has been developed to provide effective and efficient learning environments. This interactive software is also enriched with animations and text-based course contents. In addition, the effects of changes of ANN parameters are directly observed through graphical results. In this way, users can easily understand the working mechanism of artificial neural networks. Without using any commercial libraries, software is developed with C#, which is an object-oriented programming language. File formats such as XML, TXT and CSV are supported to co-operate with other software. BalanceandScale data set is used to evaluate the performance of the software. 0.9918 accuracy, 1 specificity and 1 sensitivity values have been achieved. When this study is compared with previous studies, it became clear that there are improvements in the fields of visuality, understandability and interactivity.

Keywords: artificial neural networks, educational software

2014, 74 pages

TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Doç. Dr. Tuncay YİĞİT'E teşekkürlerimi sunarım. Literatür çalışmasından ulaştığım sonuçlara kadar tezimin her anında rehberlik eden yardımcı Danışman Hocam Yrd. Doç. Dr. Ali Hakan IŞIK'A teşekkür ederim.

Tezimin her aşamasında beni yalnız bırakmayan aileme sonsuz sevgi ve saygılarımı sunarım.

Mehmet BİLEN
ISPARTA, 2014

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

ΔA	Ağırlıktaki değişim miktarı
A	Ağırlık
ART	Adaptive Resonance Theory
B	Belirlilik
B	Bias, eşik değer
Ç	Çıkış
D	Duyarlılık
DN	Doğru negatif
DO	Doğruluk
DP	Doğru pozitif
E	Beklenen çıkış ile gerçekleşen çıkış arasındaki fark
G	Giriş
GRNN	Genelleştirilmiş Regresyon Ağları
LVQ	Learning Vector Quantization
MSE	Toplam kare hata
Mbps	Aktarım hızı (Mega bit per second)
Net	Bir sinirin net çıkış değeri
PPN	Olasılıksal ağlar
Sn	Saniye
SOM	Self Organization Maps
TH	Toplam hata
YN	Yanlış negatif
YP	Yanlış pozitif
YSA	Yapay sinir ağları
α	Devinirlik değeri (Momentum)
γ	Öğrenme katsayısı
δ	Her bir sinirin hatası

1. GİRİŞ

Yapay sinir ağları (YSA) hızlı bir şekilde yüksekokullarda ve üniversitelerdeki ders müfredatlarına girmeye başlamıştır. Fakat YSA'nın matematiksel yapısı ve karmaşık doğası nedeniyle mühendislik öğrencileri için dahi anlaşılması çok zor bir sistem ve algoritmadır. YSA'lar tasarım ve eğitim süreçlerinde değişiklik gösterdikleri için sınıf ortamında öğretme, kitaplar ve ders notları gibi geleneksel yöntemler YSA'ların çalışma mekanizmasının ve temellerinin öğretilmesinde tam anlamı ile yeterli olmamaktadır. Bu noktada bilgisayar benzetimleri gibi yazılımlar, YSA'nın çalışma mekanizmasının ve temellerinin öğrenciler tarafından anlaşılmasının kolay olduğu bir ortam sağlamaktadır.

Çalışmada kullanılan Yapay sinir ağları, yapay zekâ biliminin bir alt dalıdır, bu yüzden YSA'nın tam anlamı ile kavranabilmesi için yapay zekâ ve kavramlarını incelemek gerekmektedir.

1.1.Yapay Zekâ

Yapay zekâ kavramının daha kolay anlaşılabilmesi için öncelikle Zekâ kavramının tanımının bilinmesi gerekmektedir. Zekâ, insanın düşünmesi, akıl yürütmesi, gerçekleri anlama, kavrama, yargılayabilme, sonuç çıkarma, soyutlama, yeni durumlara uyum sağlama ve öğrenebilme yeteneklerinin tümüdür (Nabiyev, 2003).

Yapay zekâ insanın beyninin düşünme ve öğrenme işlemini nasıl yaptığını anlayarak, bu işlevleri taklit ederek problem çözmeye çalışan makineler geliştirmektir (Russel ve Norvig, 2003).

Günlük hayatta yaşamın birçok alanında teknolojiden yararlanılmaktadır. Ulaşım araçları, iletişim araçları, hesaplama sistemleri gibi birçok sistem, eğitim, askeri, sağlık ve bir sürü alanda hayatın vazgeçilmez bir parçası olmayı başarmıştır. Ancak yapay zekâ, matematiksel bir şekilde hesaplanamayan ve çözülmesi mümkün görülmeyen problemlerin sezgisel yöntemlerle

çözülebilmesi ve bu yeteneğin bilgisayarlara kazandırılabilmesi sürecidir (Öztemel, 2012).

Yapay zekâ terimi ilk olarak John McCarthy tarafından 1956 yılında ortaya atılmıştır ve ilk yapay zekâ programlama dili olan LISP'i geliştirilmiştir. Bir makinenin zeki olma kavramını ise Alan Turing ilk olarak kullanmıştır. Alan Turing yapmış olduğu bir test ile bir makinenin zeki olup olmamasının standardını belirlemiştir. Bu test bir insan, bir makine ve karar verici bir insandan oluşmaktadır. Karar verici farklı odalarda bulunan makine ve insan ile bir düzenek ile etkileşime geçerken hangisinin insan, hangisinin makine olduğunu bilmemektedir. Sorulan sorulara makine ve insan cevaplar vermektedir. Sonuç olarak karar verici verilen cevaplara bakarak hangisinin insan hangisinin makine olduğunu ayırt edemiyorsa, makine zeki olarak tanımlanabilmektedir (Turing, 1950). Bu tanımlamadan yola çıkarak örnek vermek gerekirse bir ses tanıma cihazı, insan sesini tanımlayıp kim olduğunu ayırt edebiliyorsa Turing Testine göre zekidir.

Günümüzde yapay zekâ çalışmaları birçok alt dalda devam etse de dört konu öne çıkmaktadır. Söz konusu olan konular aşağıda sunulmaktadır.

- Bulanık Mantık
- Uzman Sistemler
- Genetik Algoritma
- Yapay Sinir Ağları

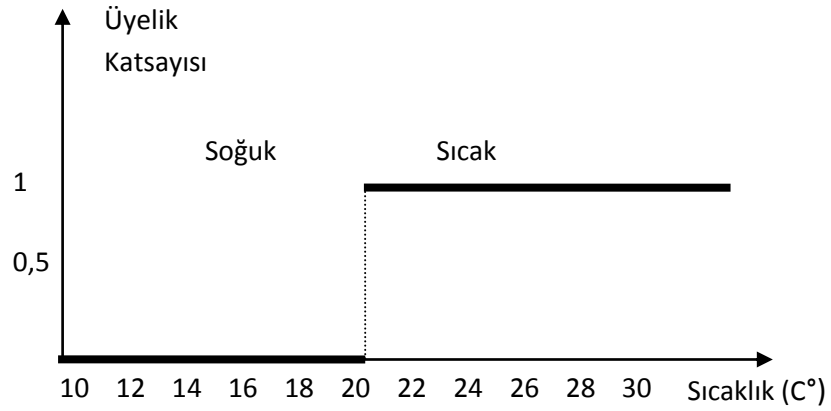
1.1.1. Bulanık mantık

Günlük hayatta karşılaştığımız birçok olay belirsiz koşullarda meydana gelir. İnsan kendi düşünce yapısı ile inceler ve yorumlar. Örneğin su soğuk denildiğinde, su günlük hayatta bilinen gerçek su kelimesi olarak anlaşılır ancak, soğuk kelimesi herkese göre aynı anlamı ifade etmemektedir. Soğuk bir iklimde yaşayan kişinin 15° için, sıcak bir iklimde yaşayan kişinin ise 35° için hissettikleri aynı olabilir, ya da ikisinin de 25° için hissettikleri ve algıladıkları

farklı olabilir. Bundan dolayı "soğuk" kelimesi altında belirsiz ve göreceli bir durum taşımaktadır. Kelimelerin ima ettikleri belirsizliklere bulanıklık denir (Şen, 2001). Bu noktada bilgisayarlara hava 30° olunca şunu yap demek yerine hava sıcaklığı normal olunca şunu yap denilebilmesi için, bu gibi durumlara karar verilebilmesi için Bulanık mantık kavramı geliştirilmiştir. Normal, Sıcak, Soğuk gibi kavramların anlaşılmasını sağlar (Öztemel, 2012).

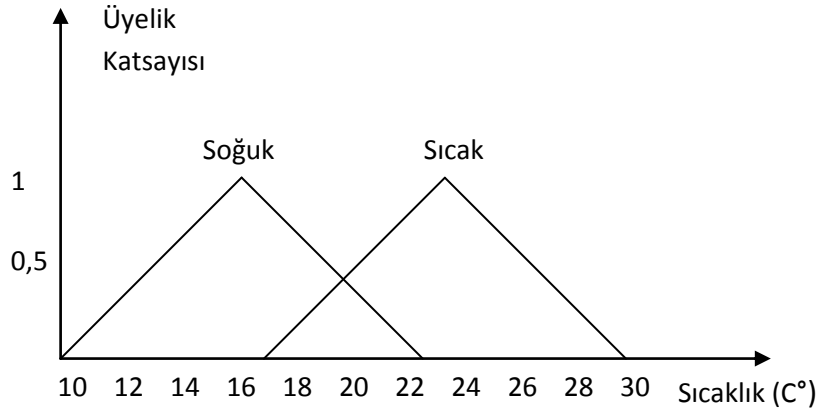
Geleneksel mantık da su ya soğuktur, ya sıcaktır. Bulanık mantık da ise bir kesinlik söz konusu değildir. Sıcak ve soğuk kararı verebilmemiz için eşik değerimiz eğer 20° ise; geleneksel mantığa göre 19.99°'nin soğuk olarak, 20.01'nin ise sıcak olarak kabul edilmesi gerekmektedir. Hâlbuki iki derece arasındaki fark bir insan tarafından fark edilemeyecek ve önemsenemeyecek kadar azdır. Buradan sıcak ve soğuk kavramları arasında geleneksel mantık ile ayrılabilecek bir durum söz konusu değildir ve çözüm için bulanık mantık kullanmak daha doğru olacaktır. Aynı örnek için 20.1° bulanık mantık ile değerlendirilmek istenirse %51 sıcak %49 soğuk sonucunu verecektir. Buradan yola çıkarak bulanık mantığın bir farklı tanımı ise, geleneksel ikilik sistemdeki mantığın tamamen doğru veya yanlış doğruluk değerleri arasında yer alan kısmi doğru veya kısmi yanlış kavramlarını da içine alacak şekilde genişletilmesi sonucu ortaya çıkan bir üst kümedir (Zimmerman, 1987).

Şekil 1.2.'de sıcaklık örneği için hangi sıcaklık değerinin hangi sınıfa ait olacağı ile ilgili geleneksel mantık için üyelik fonksiyonu gösterilmektedir. "Hava sıcak mı ?" sorusunun cevaplanması için 20° eşik değeri kabul edilmiştir. Bu durumda geleneksel mantığa göre 20° altında kalan bütün sıcaklıklar için sonuç 0 (yanlış) yani hava sıcak değil cevabı beklenmektedir. 20° üzeri için ise sonuç 1 (doğru) yani hava sıcak cevabı verilmektedir.



Şekil 1.1. Geleneksel mantık üyelik fonksiyonu

Aynı örneği bulanık mantık kullanarak tekrar çözümlemek istediğimiz de üyelik fonksiyonumuz Şekil 1.2.'deki gibi olacaktır. Bu durumda çeşitli sıcaklıklar için üyelik katsayıları hem geleneksel mantığa göre hem de bulanık mantığa göre Çizelge 1.1.'deki gibi hesaplanacaktır.

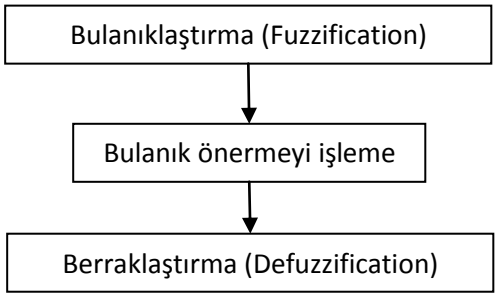


Şekil 1.2. Bulanık mantık üyelik fonksiyonu

Çizelge 1.1. Çeşitli sıcaklıkların hesaplanan üyelik katsayısı

Sıcaklık (C°)	Geleneksel Mantık		Bulanık Mantık	
	Sıcak	Soğuk	Sıcak	Soğuk
10	0	1	0	1
12	0	1	0,1	0,9
14	0	1	0,2	0,8
16	0	1	0,3	0,7
18	0	1	0,4	0,6
20	1	0	0,5	0,5
22	1	0	0,6	0,4
24	1	0	0,7	0,3
26	1	0	0,8	0,2
28	1	0	0,9	0,1
30	1	0	1	0

Bulanıklaştırma işlemlerini özetlediğimiz takdirde bulanık mantık süreci Şekil 1.3.'deki gibi gerçekleşmektedir.



Şekil 1.3. Bulanık mantık çalışma sürecinin gösterimi

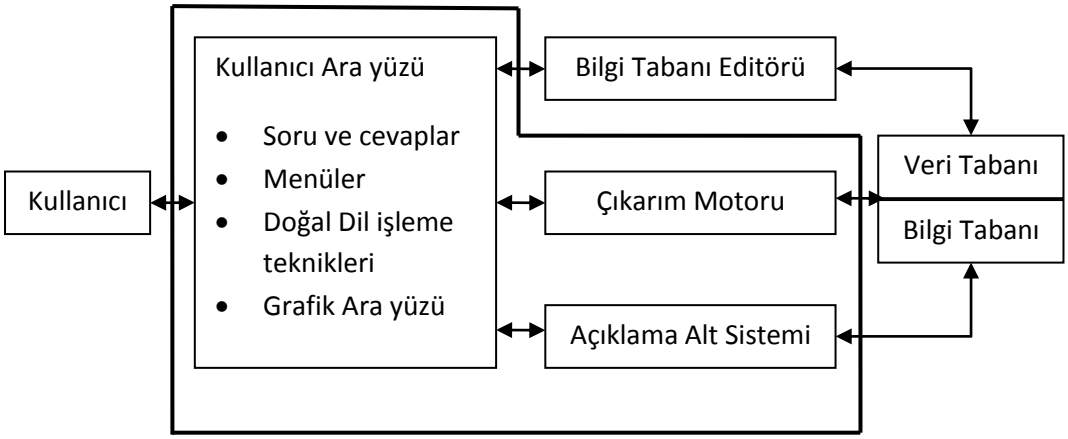
Çözülecek problemin incelenerek bulanık mantık ile ilgili önerme değişkenlerinin ve karar verme kurallarının belirlenmesi, üyelik fonksiyonlarının oluşturulma sürecine bulanıklaştırma, üyelik fonksiyonlarının üst üste konularak problemin çözüm alanının belirlenme işlemine bulanık önerme işlemi, son olarak bulunan çözüm alanında tek bir değer elde edilmesine ise berraklaştırma denir (Öztemel, 2012).

1.1.2. Uzman sistemler

Uzman sistemler, şimdiye kadar çözülmesi için uzmanlık gerektiren işlemlerin çözümünü sağlayan bilgisayar programlarıdır (Kastner ve Hong, 1984). Bir başka tanım uzman sistemi birçok alanda bir uzmanın bilgi, tecrübe ve becerisini kullanan sistem olarak açıklamaktadır (Kowali ve Janusz, 1986).

Diğer yapay zekâ alanları sıradan bir insanın çözdüğü problemleri gerçekleştirmeye çalışırken, uzman sistemler bir uzmanın çözdüğü problemi gerçekleştirmek için tasarlanırlar.

Basit bir uzman sistemin mimarisi Şekil 1.4.'de görüldüğü üzere Bilgi Tabanı, Çıkarım Motoru, Kullanıcı Ara yüzü, Veri Tabanı, Açıklama Alt Sistemi ve Bilgi Giriş Alt Sisteminden oluşmaktadır.



Şekil 1.4. Basit bir uzman sistemin mimarisi (Yavuz, 1995)

Bilgi tabanı; kayıt ve dosyalardan oluşan geleneksel veri tabanlarının aksine düzenli, düzensiz ve tecrübe ile kazanılmış bilgilerden oluşur. Bu bilgiler kuralları ifade eden bloklar halinde karakterize edilir ve kural tabanı olarak da adlandırılır (Frenzel, 1987). Problemin anlaşılması, hesaplanması ve çözümü

için gerekli olan bütün bilgileri içerir. Olaylar ve durumlar hakkında bilgi ve bunlar arasındaki mantıksal ilişkileri barındırır. Ayrıca standart çözüm ve karar alma modellerini de içerir (Tosyalı, 2008).

Çıkarım motoru; sonuç çıkarma, karar verme işlemlerini gerçekleştiren algoritmadır. Yani gelen bilgiden ilgili bir bilgi olup olmamasını inceler, elde edilen bilgiler sonucunda bir karar verme işlemi gerçekleştirmiş olur. İlgili problemlerin çözümünü gerçekleştiren sistemin beynidir. Bilginin nasıl kullanılacağına karar veren kısımdır.

Kullanıcı ara yüzü; daha önce bahsedilen bilgi tabanı, çıkarım motoru gibi birimler ile kullanıcı etkileşimini sağlayan ara yazılımdır. Kullanıcı bu yazılım üzerindeki kontrolleri kullanarak çeşitli sorulara cevap verir, sistem ise bu soruları bilgi tabanı üzerine işlenmiş bir şekilde yerleştirir. Daha sonra bu işlenen veriler tekrar kullanıcı ara yüzü ile kullanıcıya uygun bir formatta sunulur. Bu noktada bu işlemlerin kullanıcı ile etkileşimli bir şekilde gerçekleştirilebilmesi için soru cevaplar, menüler, doğal dil işleme teknikleri, grafik ara yüzü ve bunlar gibi birçok farklı yöntem kullanılır.

Veri tabanı; büyük miktardaki bilgileri depolamada geleneksel yöntem olarak bilinen dosya işleme sistemlerine alternatif olarak geliştirilen birbirileri ile ilişkide olan bilgilerin depolandığı alan olarak tanımlanmaktadır (Kilitçi vd. 2011). Uzman sistemler üzerinde kullanılan veri tabanları bir kaç noktada bu tanımdan ayrılmaktadır. Sistem de gerçekleşen her bir olayın kaydının tutulup, izlenebildiği, verilen her bir kararın, kullanıcı girişinin kayıt edilebildiği bir saklama bölgesidir (Yavuz, 1995).

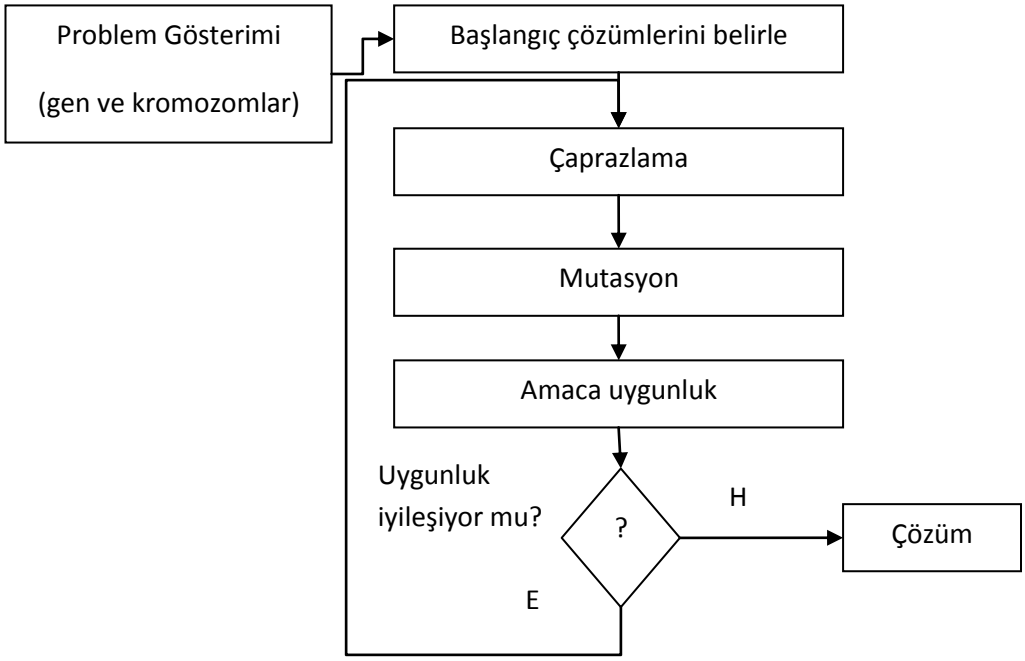
Açıklama alt sistemi; sistemin varmış olduğu sonuca hangi kriterler doğrultusunda vardığının kullanıcıya açıklandığı alt sistemdir. Bu kullanıcının sistemin verdiği kararın doğruluğu konusundaki şüpheleri gidermesi için çoğu zaman gerekli olan bir sistem olmasına rağmen bazı durumlarda kullanmak gereksiz ya da kullanılmaması gerekli olabilir. Bir araba tamircisi için gerçekleşen araba arızasının neden kaynaklandığını bilmesi gerekli

görülebilirken bir uçak pilotu için havada uzman sistemin verdiği direktiflerin sorgulanması hayati kayıplara yol açabilir.

Bilgi giriş alt sistemi; bilgi tabanındaki kuralların kullanıcılar tarafından güncellenmesi anlamına gelen ana sistemden bağımsız çalışabilen bir alt sistemdir. Güncel konularda hizmet veren uzman sistemler için önemli bir özelliktir. Tıp alanında kullanılan bir uzman sistem için ilaç bilgilerinin sürekli güncellenmesi, teknoloji alanında çalışan bir uzman sistem için ise yeni teknolojilerin sisteme girilmesi gerekmektedir.

1.1.3. Genetik algoritma

Basit bit dizilerinin kullanılarak karmaşık yapıların kodlanabileceği bir algoritmadır (Holland, 1975). Canlıların seçilim sürecinden esinlenerek geliştirilen bu algoritmada önce bir başlangıç bit dizisi seçilir. Daha sonra bu dizi bir sürü kalıtım çaprazlama gibi seçme ve tekrar üretme işlemleriyle yeniden üretilir. İstenilen çözüm elde edilene kadar bu işlemler tekrar edilir. En son üretim yapılan sistemin en uygun çözüm kümesi olduğu kabul edilerek problem çözülmüş olur. Her zaman en uygun çözümlere ulaşılamasa da bu yöntem kullanılarak uygun çözümlere yakın sonuçlar elde edilmiş olur. Bu algoritmanın çalışma yapısındaki temel öğeler Şekil 1.5. üzerinde gösterilmiştir ve devamında açıklamaları yapılmıştır.



Şekil 1.5. Genetik algoritmanın öğeleri ve çalışma prensibi (Öztemel, 2012)

Gen; kendi içerisinde genetik bilgi taşıyan en küçük kalıtsal birime denir. Kromozom; kısmi bilgiler taşıyan bütün bir çözüm dizesinin bir araya gelmesi ile meydana gelir (Aktürk, 2013). Başlangıçta rastgele bir şekilde düzenlenen bu gen ve kromozomlar genetik algoritma kullanılarak çözüm kümesini oluşturacak şekilde evrilirler.

Çözüm için gerekli bilgileri üzerinde taşıyan kromozomların bir araya gelmesi ile oluşan yığınlara popülâsyon denilmektedir. Popülâsyondaki fazla kromozom sayısı çözüm süresini uzatmakta ancak az sayıda olduğu takdirde de çözüme ulaşılamamasına neden olabilmektedir.

Çaprazlama; belirli bir orana göre popülâsyon içerisindeki her iki kromozomdan farklı iki kromozom elde etme işlemine denir. Bir diğer deyişle çözüme katkı sağlamayan genlerin silinerek yerine çözüme katkı sağlayan genlerin elde edilerek değiştirilmesi süreci olarak ifade edilebilir.

Sadece çaprazlama ile sonuca ulaşmak çoğu zaman mümkün olmamaktadır. Bunun nedenlerinden biri çaprazlamanın sürekli aynı genleri kullanmasıdır. Kendine tekrar etmesini önlemek gelişmeyi sağlamak için mutasyon oranına göre genler üzerinde değişikliğe gidilmesi işlemine mutasyon olarak ifade edilmektedir. Böylece olmayan bir genin özelliklerine ulaşmak için tercih edilen bir yöntemdir ancak mutasyon oranının yüksek tutulması da büyük oranda değişiklik meydana getirip bozulmalara sebep olacağından uygun bir değer belirlenmesi gerekmektedir.

Uygunluk fonksiyonu her problem için özel olarak hazırlanan uygun çözüme ulaşılıp ulaşılmadığını kontrol eden fonksiyondur. Bazı problemlerde zaman çözülmesi gereken bir problem iken bazı problemlerde maliyeti azaltmak en uygun çözüm olabilir. Her iki durumda da azaltılabilen en uygun duruma gelene kadar çaprazlama ve mutasyon işlemleri devam etmektedir.

1.1.4. Yapay sinir ağları

YSA'lar tecrübe ile elde edilen bilgiyi alan, kaydeden ve kullanan fiziksel hücreli sistemlerdir (Sağıroğlu vd. 2003).

YSA, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgi türetebilme, yeni bilgi oluşturabilme ve keşfedebilme gibi yetenekleri yardım almadan doğrudan gerçekleştirmek amacı ile geliştirilen algoritmalardır(Öztemel, 2012).

Bir sinir ağı deneyimsel bilgi depolamak ve kullanıma hazır hale getirmek için doğal bir eğilimi olan basit işlem birimlerinden oluşan, paralel bir şekilde dağıtılmış bir işlemcidir(Haykin, 1999).

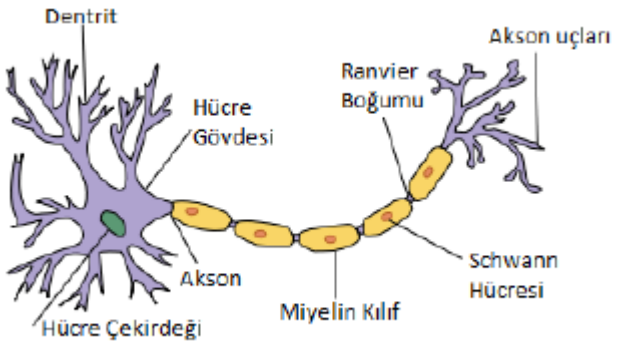
YSA bir biyolojik sinir ağlarını taklit eden bilgisayar programıdır. Bu özelliği sayesinde optimizasyon, sınıflandırma, tahmin, örüntü tanıma, bellek yönetimi ve kontrol gibi birçok problemi etkili ve kolay bir şekilde çözebilmektedir (Blanton, 1997).

YSA, olayların örneklerine bakmakta, onlardan ilgili olay hakkında genellemeler yapmakta, bilgi toplamakta ve daha sonra hiç görmediği örnekler ile karşılaşınca öğrendiği bilgileri kullanarak örnekler hakkında karar verebilmektedir (Keleşoğlu ve Fırat, 2006).

1.1.4.1. Biyolojik sinir ağı

Biyolojik bir sinir hücresi insan beyninin en temel parçası olmak ile beraber hatırlama, düşünme, daha önceki deneyimlerden yararlanma gibi birçok özelliğin gerçekleştirilmesini sağlamaktadır. İnsan beyninde yaklaşık olarak 10^{11}adet sinir hücresi bulunmaktadır. Ve her bir sinir hücresinin yaklaşık 10.000 adet komşu bağlantısı mevcuttur (Elmas, 2003).

Şekil 1.6.'da gösterildiği gibi bir sinir hücresi dentrit, gövde, çekirdek, miyelin kılıf schwann hücresi, ranvier boğumu, akson ve akson uçları gibi öğelerden oluşmaktadır.



Şekil 1.6. Biyolojik sinir hücresi (Tektaş ve Karataş, 2004)

Bir hücreden diğer bir hücreye veri akışı fiziksel olmayan sinaps adı verilen boşluklar üzerinden elektrik sinyalleri ile gerçekleşir. Hücre çekirdeği kendi sinyalini ürettikten sonra dentritler bunları diğer hücrelerin akson uçlarına iletir. Akson uçlarının her biri bir başka hücre dentriti ile birleşmektedir.

1.1.4.2. Tarihçe

Yapay sinir ağları ile ilgili çalışmalar 1950'li yıllarda başlamış ve tarihsel süreçte günümüze kadar hızlı bir gelişme göstermiştir. İlk yapay sinir ağı tasarımı Walter Pits ve Warren McCulloch tarafından 1943 yılında geliştirilmiştir. Bu model Elektrik devreleri hesaplama yapan basit bir sistemdir.1948 yılında Hebb "Wiener Cybernetcis" kitabında öğrenebilen ve uyum sağlayabilen modeller için temel teşkil eden Hebb kuralını ortaya koymuştur. Marvin Minsky bu kuramı kullanarak mikroskobik zekâ kavramını ortaya atmıştır ve uzman sistemlerin doğmasını sağlamıştır. 1957 yılında ise Frank Rosenblatt ilk öğrenebilen bilgisayar olan Algılayıcıyı (Perceptron) geliştirmiştir. Algılayıcı tek katmanlı, eğitilebilen ve tek çıkışa sahip bir ağ modelini kullanmaktadır (Elmas, 2003).

1959 yılında Bernand Widrow ve Marcian Hoff Stanford üniversitesinde yaptıkları çalışmada mühendislik uygulamaları için Uyarlamalı doğrusal sinir (Adaptive Linear Neuron) modelini geliştirmişlerdir. Bu uygulamada yapay sinir ağlarının öğrenme becerisi daha da gelişmiştir. Bu uygulama ile uzun mesafedeki telefon hatları arasındaki gürültü ve yankının yok edilmesi sağlanmış böylece günlük hayattaki problemlerde kullanılan ilk yapay sinir ağı olmuştur.

1960'lı yıllarda özel veya (XOR) problemi ile YSA'nın doğrusal olmayan problemleri çözemediği ispat edilmiştir. Böylece bu alanda yapılan çalışmalar 1972 yılında Kohonen ve Anderson'un Associative Memory konusunda yayınlar yapana kadar yavaşlamıştır. Kohonen daha sonra 1982 yılında Kendi Kendine Öğrenme nitelik Haritaları konusundaki çalışmasını yayınlamıştır. Aynı dönemde Fukushima şekil ve örüntü tanıma amaçlı kullanılan modeli NEOCOGNITRON'i tanıtmıştır.

Yapay sinir ağlarının genelleştirilebileceği ve zor problemlere çözüm üretebileceği 1982 ve 1984 yılları arasında Hopfield tarafından yapılan çalışmalarla gösterilmiştir ve gezgin satıcı problemini çözmüştür. Bunun sonucunda Boltzman Makinesinin temellerini oluşturmuştur. 1987 yılında ilk

yapay sinir ağları sempozyumu yapılmıştır ve gelişim sürecine önemli katkı sağlamıştır. 1988 yılında Broomhead ve Lowe radyan tabanlı fonksiyonlar modelini oluşturmuşlar ve filtreleme konusunda kayda değer sonuçlar elde etmişlerdir. Spect ise bu ağların gelişmiş şekli olan Olasılıksal ağlar (PPN) ve Genelleştirilmiş Regresyon Ağlarını (GRNN) geliştirmiştir(Öztemel, 2012).

YSA'ların günümüze kadar gelen süreçte yaşadığı gelişimi Çizelge 1.2.'de gösterilmektedir.

Çizelge 1.2 YSA'nın tarihsel süreçte ilerlemesi (Altıntaş, 2011)

Yıl	Çalışma
1880	İnsan beyninin yapısı ve fonksiyonları ile ilgili ilk yayının yazılması
1911	İnsan beyninin sinir hücrelerinden oluştuğu fikrinin benimsenmesi
1943	Yapay sinir hücrelerine dayanan hesaplama teorisinin ortaya atılması ve eşik değerli mantıksal devrelerin geliştirilmesi
1949	Öğrenme prosedürünün bilgisayarlar tarafından gerçekleştirilecek şekilde geliştirilmesi
1956-1962	ADALINE ve Widrow - Hoff öğrenme algoritmasının geliştirilmesi
1957-1969	Tek katmanlı algılayıcıların yetersizliklerinin ispatlanması
1965	İlk makine öğrenmesi kitabının yayınlanması
1967-1969	Bazı gelişmiş öğrenme algoritmalarının geliştirilmesi
1969	Tek katmanlı algılayıcıların yetersizliklerinin ispatlanması
1972	Korelâsyon matris belleğinin geliştirilmesi
1974	Geriye yayılım modelinin geliştirilmesi
1978	ART modelinin geliştirilmesi
1982	Çok katmanlı algılayıcıların geliştirilmesi
1984	Boltzman Makinesi'nin geliştirilmesi
1988	RBF modelinin geliştirilmesi
1991	GRNN modelinin geliştirilmesi

1.1.4.3 Uygulama alanları

Günümüzde YSA tahmin ve öngörü gibi birçok problemin çözümünde kullanılmaktadır.

İnsanlar canlı ve cansız tüm varlıklar hakkında detaylı ve sistematik bir bilgi elde etmek, karışıklığı gidermek için sınıflandırmaya ihtiyaç duymuşlardır. Sınıflandırılmayan nesnenin tanımlanması, değerlendirilmesi ve yargılanması mümkün değildir. Sınıflandırma nesneleri tanıdık hale sokar ve anlam, önem kazandırır (Anonim, 2007). Doğrusal denklemler kurularak çözülmesi zor olan sınıflandırma problemi yapay sinir ağları kullanılarak kabul edilebilir bir hata oranı ile çözülebilmektedir. Yapay sinir ağları kullanılarak ülkemizde çalışılan bazı sınıflandırma uygulamaları Çizelge 1.3'de gösterilmiştir. Çizelge 1.4'te mühendislik alanında, 1.5'de ise tıp alanında YSA ile gerçekleştirilen çalışmalar listelenmiştir.

Çizelge 1.3 YSA kullanılarak yapılan sınıflandırma çalışmaları

Yıl	Başlık	Yazar
2006	Biyoinformatikte Sınıflandırma Tekniği Olarak Yapay Sinir Ağlarının Kullanılması	ÖZYILMAZ, L.
2007	EMG İşaretlerinin Genetik Algoritmalar ve Çok Katmanlı Yapay Sinir Ağları ile Sınıflandırılması	YÜKSEL, A. & KORUKÜREK, M.
2007	Yapay Sinir Ağları ile Kemik Yoğunluğunun Sınıflandırılması	ÖZERDEM, M.S.
2008	Yapay Sinir Ağları ile Web İçeriklerini Sınıflandırma	GÜVEN, E. N. & ONUR, H. & SAĞIROĞLU, Ş.
2009	Yapay Sinir Ağları ile Adapazarı Killerinin Sınıflandırılmasında İstatistiksel Analiz	GÖKTEPE, F. & ARMAN, H. & DOĞAN, E. & SANDALCI, M.
2010	Yapay Sinir Ağı ve Görüntü İşleme Teknikleri Kullanarak Durum Buğdayının Camsılığının Belirlenmesi	BABALIK, A & BOTSALI, M.

Çizelge 1.4. Mühendislik alanındaki bazı YSA çalışmaları

Yıl	Başlık	Yazar
1997	Yapı analizi ve tasarımında yapay sinir ağları	DERE, Y.
2000	Depremde hasar görmüş betonarme yapılarda hasar düzeyinin belirlenmesinde bir yapay sinir ağları uygulaması	GÜÇSAV, A.R.
2002	C-130 uçaklarının performans hesaplarında yapay sinir ağlarının kullanılması	GÜLEÇ, K.
2003	Yapay sinir ağlarıyla 3-D bilgisayar görmesi	SOFU, A.B.

Çizelge 1.5. Tıp alanındaki bazı YSA çalışmaları

2004	Yapay Sinir Ağları ile tiroit hastalıklarının sınıflandırılması	ŞAHİN, C.
2005	Göze ait elektro fizyolojik sinyaller kullanılarak yapay sinir ağları destekli bazı göz hastalıklarının sınıflandırılması	GÜVEN, A.
2007	Yapay Sinir Ağlarının incelenmesi be sırt ağrısı olan bireyler üzerinde bir uygulaması	KARAKAYA, B.
2008	Yapay Sinir Ağları kullanılarak retinada hastalık teşhisi	YAĞMUR, F. D.

1.1.4.4. Yapısı

YSA birden çok yapay sinir hücresinin bir araya gelmesi ile oluşmaktadır. Bu yüzden bir yapay sinir hücresinin anlaşılması tüm ağın anlaşılmasına yardımcı olacaktır.

Girdiler, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıktılar olmak üzere bir yapay sinir hücresinin 5 ana öğesi vardır. Bu hücrelerin meydana getirdiği yapay sinir ağının öğeleri ise katmanlar olarak ayrılmıştır. Giriş, gizli(ara) ve çıkış katmanı olmak üzere 3 adettir.

1.1.4.4.1 Giriş değerleri

YSA'lar örneklerden öğrenen algoritmalardır. YSA'nın örneklerden öğrenebilmesi, kendi edindiği tecrübeleri tekrar geri bildirim olarak alabilmesi

veya bir başka hücreden veri alabilmesi için bunların sisteme girdi olarak verilmesi gerekmektedir. Her bir sinir hücresi girdi alabilme özelliğine sahiptir.

1.1.4.4.2. Ağırlıklar

Ağırlıklar yapay sinir ağının girişine verilen değerlerin hücreler üzerindeki etkisini belirleyen katsayılardır. Hücreler arasındaki bağlantıları temsil eden ağırlıklar bir hücreden diğer bir hücreye giden bilgi ve o bilginin etkisini göstermektedir.

1.1.4.4.3.Toplama fonksiyonu

Bir yapay sinir hücresine genellikle birden fazla hücreden veri gelmektedir. Bu gelen verilerden elde edilen net girdiyi hesaplamak için kullanılan fonksiyona toplama fonksiyonu denir. Genellikle ağırlıklar ile gelen bilgi çarpılarak toplanır, bu fonksiyona ağırlıklı toplam veya çarpım da denir. Çizelge 1.6'da bazı toplama fonksiyonu örnekleri gösterilmektedir.

Çizelge 1.6. Bazı toplama fonksiyonu örnekleri

Maksimum	Her bir girdi ile ilgili ağırlık çarpılarak elde edilen sonuçlar içerisinden en büyük olanı ilgili hücrenin girişi olarak kabul edilir.
Minimum	Her bir girdi ile ilgili ağırlık çarpılarak elde edilen sonuçlar içerisinden en küçük olanı ilgili hücrenin girişi olarak kabul edilir.
Çoğunluk	Her bir girdi ile ilgili ağırlık çarpılarak elde edilen sonuçlar pozitifler ve negatifler olmak üzere iki gruba ayrılır. Hangi grubun sayısı daha fazla ise o gruptaki mutlak değeri en yüksek değerin gerçek değeri ilgili hücrenin girişi olarak kabul edilir.
Kumilatif Toplam	Çarpım fonksiyonu ile hesaplanan girdiler bir önceki hesaplanan değerin üzerine eklenerek yeni giriş değeri hesaplanmış olur.

1.1.4.4.4. Aktivasyon fonksiyonu

Hücrenin girişi toplam fonksiyonu ile belirlendikten sonra hücrenin bu gelen bilgiyi işleyip bir tepki üretmesi gerekmektedir. Bu üretilen tepkinin hesaplanması için kullanılan fonksiyona harekete geçme fonksiyonu ya da

aktivasyon fonksiyonu denilmektedir. Çizelge 1.7'de yaygın olarak kullanılan aktivasyon fonksiyonları verilmiştir.

Çizelge 1.7 Aktivasyon fonksiyonları (Öztemel, 2012)

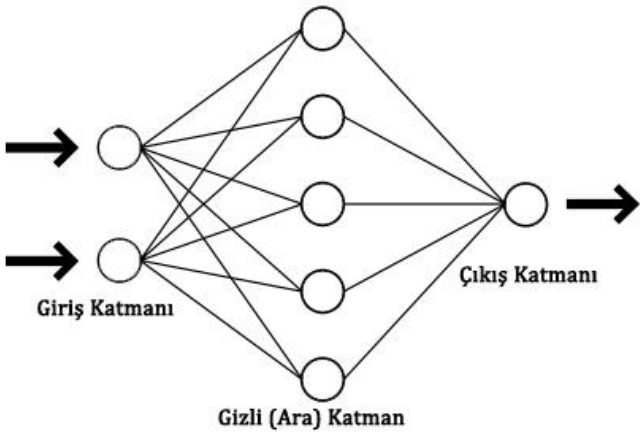
Aktivasyon Fonksiyonu	Açıklaması	İfadesi
Lineer (Doğrusal) fonksiyon	Gelen girdiler geldiği gibi hücrenin çıktısı olarak kabul edilmekte, hiç bir işlem yapılmamaktadır.	$F(NET) = NET$
Step fonksiyonu	Bu fonksiyonda çıktı ya 1'dir ya 0. Belirli bir eşik değer üzerinde ise 1 değil ise çıkışa 0 yansıtılır.	$F(NET) = \begin{cases} 1 \; eğer \; NET > eşik \; değer \\ 0 \; eğer \; NET < eşik \; değer \end{cases}$
Sinus fonksiyonu	Öğrenilmesi gereken olaylar eğer sinüs eğrisine benzer bir dağılım gösteriyor ise tercih edilen ve net değerinin sinüsü alınarak hesaplanan değerdir.	$F(NET) = SİN(NET)$
Eşik değer fonksiyonu	Gelen değerleri 0 ve 1 arasına çekmek için kullanılır. 0 ve 1 eşik değerdir.	$F(NET) = \begin{cases} 0 \; eğer \; NET \leq 0 \\ 1 \; eğer \; NET \geq 1 \\ NET \; eğer \; 0 < NET < 1 \end{cases}$
Sigmoid fonksiyonu	Çok katmanlı YSA modellerinde sıklıkla kullanılan aktivasyon fonksiyonudur.	$F(NET) = \frac{1}{1 + e^{-NET}}$
Hiperbolik tanjant fonksiyonu	Gelen net değerinin hiperbolik tanjant fonksiyonundan geçirilmesi ile elde edilir.	$F(NET) = (e^{NET} + e^{-NET}) / (e^{NET} - e^{-NET})$

1.1.4.4.5 Hücrenin çıktısı

Toplama fonksiyonu ile hesaplanan girdi, aktivasyon fonksiyonunda değerlendirilerek hücrenin tepkisi üretilmiş olur. Bu üretilen sonuç hücrenin çıktısı olarak kabul edilmektedir. Bir hücrenin birden çok girişi olabilir ancak tek bir çıktısı olmaktadır. Ama bu tek bir çıktı birçok hücreye giriş olarak ağırlıklarla aktarılabilir.

1.1.4.4.6 Katmanlar

Yapay sinir ağlarının genel yapısına baktığımızda sinirler sisteme tabakalar şeklinde yerleşmiştir. Her bir tabaka içinde bulunduğu sinir dizileri ile benzer davranışlar sergilenmektedir. Tabakaların amaçları kendisine gelen verilerin toplanıp bir aktivasyon fonksiyonundan geçirilerek ağırlıklandırılmış bağlantı dokularını oluşturmaktır. Bu tabakalar giriş katman, gizli katmanlar ve çıktı katmanı olmak üzere Şekil 1.7.'de görüldüğü gibi üç kısımda ele alınır.



Şekil 1.7 YSA'nın katman yapısı

Tüm bu katmanlar ele alındığında bir yapay sinir ağı yapısı; hücreler arası bağlantılar ile değerlerin iletildiği, bağlantıların belirli ağırlıklara sahip olduğu ve değerlerin bu ağırlıklarla işleme alındığı, hücrelerden çıktı fonksiyonu elde edilmesinde bir aktivasyon fonksiyonunun kullanıldığı bir model olarak ele alınabilir (Lee ve Park, 2001).

Giriş katmanı; dışarıdan gelen verilerin sisteme aktarıldığı kısımdır. Giriş olarak kullanılacak bütün parametreler sisteme buradan dâhil olmak zorundadır. Her probleme göre kullanılacak sinir sayısı değişse de en az bir girişin olması gerekmektedir. Burada veriler üzerinde herhangi bir işlem süreci gerçekleşmez sadece ağırlıklarla çarpılarak iç katmanlara iletilir.

Gizli katman; girdi birimlerinin belirli işlemlerden geçirildiği aynı zamanda işlem katmanı ve ara katman olarak da isimlendirilir. Tek bir katmandan oluşabileceği gibi birden fazla katmandan da meydana gelebilir. Tercih edilen ağ modeline göre katmanın yapısı ve işlevi değişse de çalışma sistemi bir kara kutuya benzemektedir. Bunun nedeni çözülmesi istenen probleme uygun fonksiyon tanımlamasını bizim yerimize kendisinin yapmasıdır. Kara kutu tabiri bu sürecin detaylarını göremediğimiz için kullanılmaktadır.

Çıkış katmanı; giriş katmanı gibi en az bir elemandan oluşur ama giriş katmanındakinin aksine burada işlem gerçekleşir. Oluşacak çıktı ise tercih edilen aktivasyon fonksiyonuna göre şekillenir.

1.1.4.5. Yapay sinir ağlarının sınıflandırılması

YSA'ların sahip olduğu karmaşık yapıdan dolayı birden fazla başlık altında sınıflandırılmaları daha doğrudur.

1.1.4.5.1. Öğrenme yöntemi

YSA öğrenme yöntemine göre danışmanla ve danışmansız olmak üzere ikiye ayrılmaktadır.

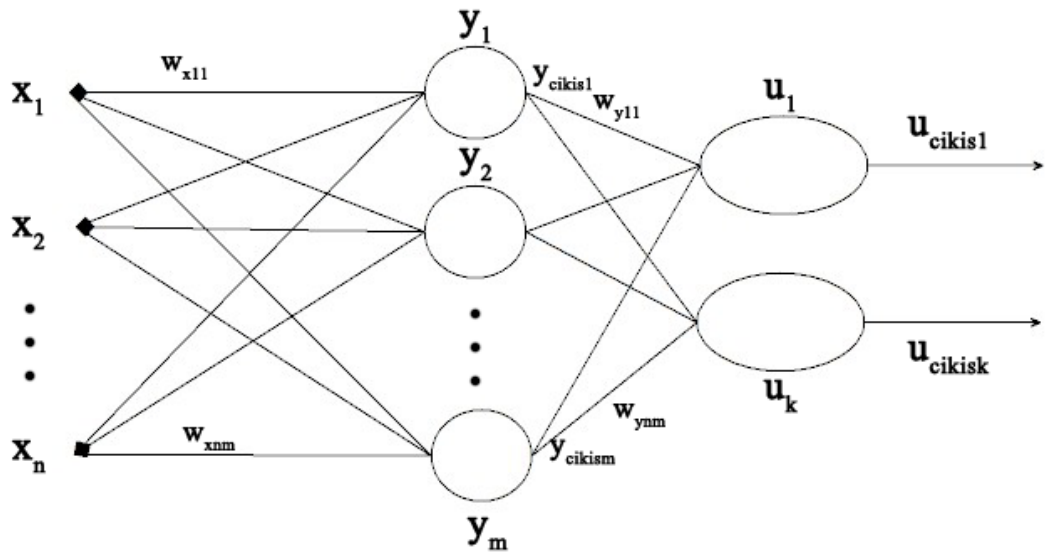
Danışmanla öğrenme; eğiticili öğrenme adı da verilen bu öğrenme yönteminde ağa verilen girişlerin sonucunda ağın üretmiş olduğu çıkışlar gerçek çıkışlar ile karşılaştırılır. Beklenen çıkış ve gerçekleşen çıkış arasındaki fark bulunarak hata hesaplanır ve bu hatanın minimum seviyeye indirilmesi için ağırlıkların güncellenmesi yolu ile öğrenme gerçekleştirilmiş olur. Geri yayılım ve delta kuralı danışmanla öğrenme kullanan algoritmalara örnek olarak gösterilebilir.

Danışmansız öğrenme; bu yöntem ağa verilen giriş değerlerinin sonucunda üretilen çıkış başka bir çıkış ile karşılaştırılmaz ağın sınıflandırma işlemini sonuçlar arasındaki farklılıklara bakarak yapması beklenir.

1.1.4.5.2. Veri iletim yönü

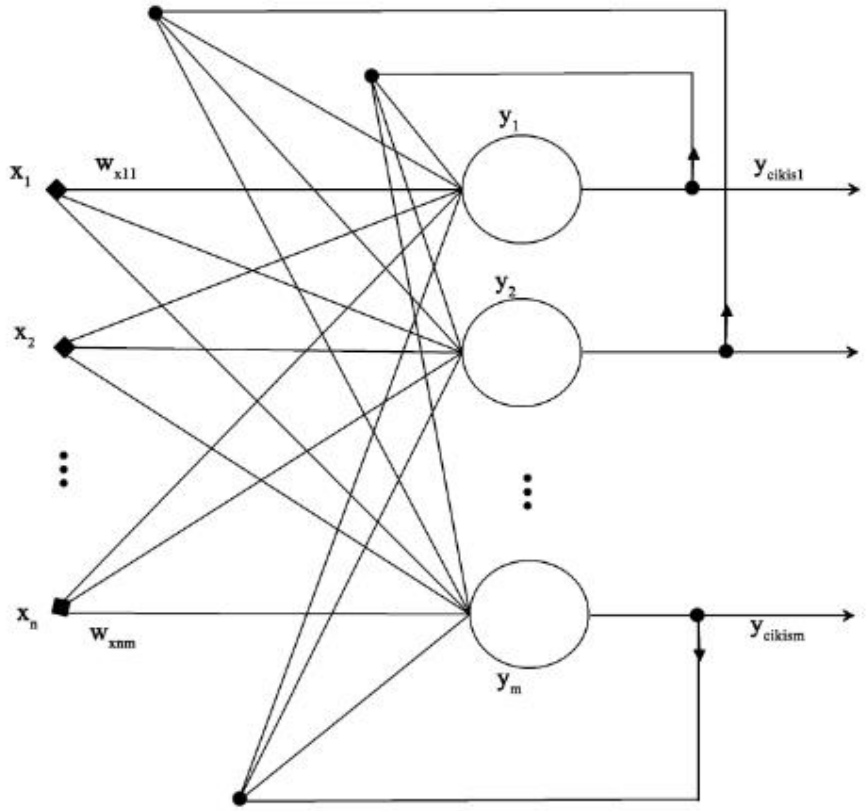
Ağın veriyi ilettiği yöne göre YSA'lar bir kategoriye ayrılması gerekirse ileri beslemeli ve geri beslemeli ağlar olmak üzere ikiye ayrılabilir.

İleri beslemeli ağlar; verinin sadece ileri yönde ilerlediği ağlara verilen isimdir. Öğrenme genelde geriye yayılarak gerçekleşse de bir hücrenin çıkışı kesinlikle kendisinden bir önceki katmanı beslememektedir. Çok katmanlı (Multilayer Perception - MLP) ve Öğrenmeli vektör nicelemesi (Learning Vector Quantization - LVQ) ağları ileri beslemeli ağlara örnek verilebilir. Şekil 1.8 'de ileri beslemeli ağın yapısı gösterilmektedir.



Şekil 1.8. İleri beslemeli YSA'nın yapısı

Geri beslemeli ağlar; verinin hem ileri hem de geri yönde olmak üzere iki yönde de ilerlediği ağlardır. Bir katmandaki çıktı kendisinden önceki katmanın veya ağın girişlerine aktarılıyor ise bu ağa geri beslemeli bir ağ denilebilir. Hopfield, Elman, Özyinelemeli haritalar (Self Organization Maps - SOM) ve Jordan ağları geri beslemeli ağlara örnek verilebilir. Şekil 1.9 'da geri beslemeli ağın örnek bir modeli gösterilmektedir.



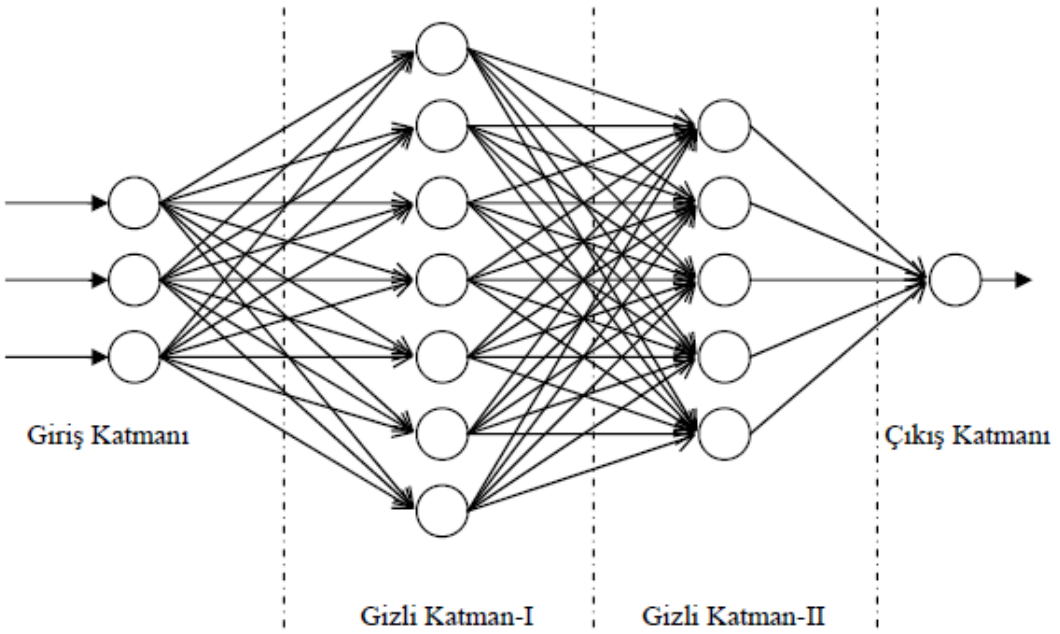
Şekil 1.9. Geri beslemeli YSA'nın yapısı

1.1.4.5.2. Mimari yapısı

YSA'ları mimari yapısına göre gruplamak için katman sayısına bakılması gerekmektedir. Tek katmanlı ağlar ve Çok katmanlı ağlar olmak üzere YSA'lar ikiye ayrılmaktadır.

Tek katmanlı ağlar; sadece girdi ve çıktı katmanından oluşan ağlardır. Arada ayrıca bir katman bulunmaz. Yaygın olarak algılayıcı modeli olarak isimlendirilirler.

Çok katmanlı ağlar; girdi ve çıkışlar arasında en az bir gizli katmanın olduğu ağlardır. Genelde karmaşık, doğrusal olmayan problemlerin çözümünde kullanılırlar. Şekil 1.10.'da iki adet gizli katman kullanılmış bir ağın modeli gösterilmektedir.



Şekil 1.10 Çok katmanlı bir YSA'nın yapısı (Bozik, 2011)

1.1.4.6. Özellikleri

Yapay sinir ağları hesaplama ve bilgi işleme gücünü paralel dağılmış yapısından, öğrenebilme ve genelleme yeteneğinden almaktadır. Bu üstün özellikler yapay sinir ağlarının karmaşık problemleri çözebilme yeteneğini göstermektedir.

1.1.4.6.1. Doğrusal olmama ve hız

Yapay sinir ağlarının temel işlemini yapan sinirler doğrusal değildir. Bununla beraber sinirlerin birleşmesi ile oluşan sistemde doğrusal değildir. Yapay sinir ağları, bu özelliği sayesinde doğrusal olmayan problemlerin çözümü için önemli ve vazgeçilmez bir araç olmuştur.

Geleneksel bilgi işlem süreçlerinde bilgi doğrusal bir düzen üzerinde işlenmeye çalışılır. Bunun en büyük dezavantajı hız sorunudur. Bilgisayarların bir problemi çözme sürecinde insan beyninden daha hızlı hareket ettiği kabul edilse de genel hıza bakıldığı zaman insan beyni bilgisayarlara göre çok daha hızlı çalışır. Yapay sinir ağları da bu noktada bir kez daha geleneksel sistemlerden

ayrılarak bütün ağa yayılarak işlemlerini paralel bir şekilde yürütür. Bu sayede tüm sistem bir diğer katmanı beklemeden, eş zamanlı olarak ve yüksek hızda çalışmaktadır.

1.1.4.6.2. Öğrenme yeteneği

Yapay sinir ağlarında öğrenme işlemi programlayarak gerçekleştirilmez. Verilen giriş ile bu girişe uygun gelen çıkış değerlerini kendisi öğrenir. Hangi giriş parametresinin çıkış değerini hangi ağırlıkta etkilediğini kendisi bulur. Elini sıcak sobaya korkusuzca değdiren bebek sobanın sıcak olduğu bilgisini öğrenmiş olur. Bu sayede geleneksel algoritmalarda daha önce çözümü tanımlanmayan problemler çözülemezken yapay sinir ağları problemin çözümü hakkında bilgilendirilmeden çözüme ulaşabilir. Bunun için gereken tek şey yapay sinir ağının girdilere ilişkin çıktılar ile deneyim yaşamasıdır.

1.1.4.6.3. Hata toleransı

Bir sistemde elemanlardan biri çalışmasını durdurduğunda sistem genelinde bir probleme neden olmaktadır. Bu durumda eleman kendinden beklenen görevi yerine getirememiş, beklenen çıktıları bir sonraki elemana ulaştıramamıştır. Yapay sinir ağları gibi doğrusal olmayan yapılarda görevler tüm ağ geneline dağıtılmış olduğundan dolayı herhangi bir birimdeki hata başlangıçta yanlış çıktılara sebep verse de ağ kendi için eksik olan elemanın görevini dağıtarak hatayı yüksek oranda telafi etmiş olacaktır.

YSA'nın bir diğer özelliği kesin olmayan verilerle çalışabilmesidir. Her adımda ikilik sistemi kullanan geleneksel yaklaşımlar bunun sonucunda kesin parametrelerle çalışmak zorunda kalır. Örneğin bir elma renginin kırmızı olması kesin bir parametredir ve bilgisayar için bir sıkıntı teşkil etmez ama gerçek hayatta kullandığımız birçok değer kesin değildir. Elmanın biraz kırmızı birazda yeşil olması bunun için örnek verilebilir. Daha çok yapay zekâ kuramlarından bulanık mantığın ilgilendiği bu konu yapay sinir ağlarının özellikleri içinde yer alır.

1.1.4.6.4. Uyarlana bilirlik ve genelleme

İnsan beyni bir alanda öğrendiği bilgiyi rahatlıkla başka bir alana aktarabilir. Örneğin bisiklet kullanmayı çok iyi bilen birisi motorlu bir bisiklete bindiğinde daha önceden sahip olduğu yetiyi yeni duruma göre uyarlayacak ve çok fazla zorlanmadan yeni tanıştığı aracı kullanmayı öğrenecektir. Yapay sinir ağlarında öğrenmenin gerçekleştiği ağırlıklar yeniden yapılandırılabilir. Böylece yapay sinir ağları tekrar eğitilerek problemlerde meydana gelen değişikliklere cevap verecek şekilde güncellenebilir. Değişken koşullara hızlı adapte olabilmesi sayesinde yapay sinir ağları sinyal ve görüntü işleme alanlarında diğer tekniklere nazaran daha verimli kullanılmaktadır.

1.1.4.6.5. Kullanım kolaylığı

Bir problem çözülmek istendiği zaman o probleme uygun çözüm yolu oluşturmak gerekmektedir. Problemin karmaşıklık derecesine göre denklemler ve formüllerde aynı oranda karmaşıklaşacaktır. Yapay sinir ağlarında kullanılan yapay sinir elemanı bütün ağlarda aynı şekilde kullanılır. Tüm yapay sinir ağlarının temel yapısı aynı olmasından dolayı bu ağlar basit tasarımlarla her problemin çözümü için kullanılabilirler. Sistem çözüm yolunu kendi bulacağı için kullanımı gayet kolaydır ve diğer tekniklerde oluşan karmaşıklıklar yapay sinir ağlarında meydana gelmez.

2. KAYNAK ÖZETLERİ

Tez çalışmasında YSA eğitimi için kullanılmak üzere hazırlanan yazılımlar detaylı bir şekilde incelenmiştir. Söz konusu çalışmaların eksikliklerini gidererek yeni özelliklere sahip bir yazılım geliştirilmiştir.

Nissan(1989), yapmış olduğu çalışmada yüksek öğrenimde yapay zekânın önemine birçok alanda vurgu yapmıştır. Yüksek öğrenim deki ders programlarına yapay zekânın eklenmesi, bilgisayarların desteği ile bu derslerdeki bilgilerin otomatik olarak sunulması kavramını tanımlamıştır.

Williams (1992),hazırladığı çalışmada 1987'de kurulan "The Artificial Intelligence Application to Learning Programme" adı ile bilinen programı anlatmış ve bu program kapsamında geliştirilen yapay zekâ eğitimi ve yapay zekâ ile eğitim yapma alanlarında geliştirilmiş on uygulamayı aktarmıştır.

Manic vd. (2002), internet üzerinden çalışan bir YSA benzetim yazılımı geliştirerek bunu çalışmalarında anlatmışlardır. Yazarlar geliştirilen sistemin piyasadaki diğer ticari ve ücretsiz araçlara sağlamış olduğu avantajları, eğitim için tasarlanmış olması gibi, kurulum gerektirmemesi, her bir kullanıcı türü için farklı kullanıcı seviyeleri olduğu gibi gerçekleştirmiş oldukları birçok özelliği ortaya koymuşlardır. Hazırlanan uygulamanın internet üzerinde çalışmasına vurgu yaparak istemcinin kaynaklarından tasarruf etmesini sağladıklarını da belirtmişlerdir. Hazırladıkları ve NeuroFuzz adını verdikleri uygulama kullanıcıya görevler vererek YSA'nın çalışma mekanizmasını öğretimini sağlamayı amaçlamaktadır. Kullanıcı YSA ağının benzetimini çeşitli çizelgeler ile takip edebilmekte ve sonuçları istediği takdirde çeşitli formatlarda kayıt edebilmekte veya çıktı olarak alabilmektedir. Bunun ile beraber kullanıcıların Yazarlar popüler herhangi bir web tarayıcısı ile erişilebilen uygulamanın ara yüzünü HTML, Javascript ve Perl ile YSA benzetimi için gerekli olan hesaplamaları ise C programlama dilini kullanarak gerçekleştirmiştir. Son olarak gelecekte benzetim yazılımını tamamen sunucu üzerine taşıyacaklarını ve böylece kullanıcı kendi sistem kaynaklarını kullanmak yerine sunucunun

kaynaklarını kullanabileceği, benzetimi başlatıp sistemden çıkış yapıp daha sonra istediği aralıklarla giriş yaparak kontrol edip sonuçları alabileceği bir tasarım yapmayı planlamaktadırlar. Çalışma içerisinde öğrencilerin YSA'nın temelleri hakkında yardım alabileceği bir belge sayfası da bulunmaktadır. YSA gibi karmaşık bir algoritma canlandırma tekniği gibi çoklu ortam desteği olmayan geleneksel yöntemler ile kullanıcılara aktarılmaya çalışılmıştır.

Rosello vd. (2003), çalışmalarında kolay kullanışlı bir ara yüz, keşfedici eğitim etkinliklerini destekleyen ve yüksek kullanışlılık özelliklerine sahip, İspanyadaki Vigo Üniversitesi Bilgisayar Mühendisliği öğrencileri üzerinde YSA öğretmek için kullanmak üzere bir YSA eğitim aracı üretmeyi amaçlamışlardır ve nesne yönelimli programlama dillerini kullanarak NeuroLab uygulamasını geliştirmişlerdir. Uygulamayı geliştirmeden önce bir YSA eğitim yazılımında bulunması gereken özellikleri bir bir tespit etmişlerdir. Kullanıcı dostu bir ara yüz, yüksek tekrar kullanılabilirlik, diğer programlara dâhil edilebilme ve ortak çalışabilme gibi tespit ettikleri gereksinimleri sunmuşlardır. Geliştirmiş oldukları sistemde öncelikle bu gereksinimleri sağlamışlardır. YSA'nın temelleri öğretmek için uygulama tek veya çift katmanlı YSA yapısı oluşturmaya izin vermekte, kullanıcılar katman sayısı, sinir sayısı, öğrenme algoritması, aktivasyon fonksiyonu gibi birçok parametreyi benzetim anında değiştirebilmektedir. Böylelikle her parametrenin sonuca olan etkisini grafiklerle desteklenmiş görsel bir ara yüz ile ayrı ayrı gözlemleyebilmektedir. Kullanıcılar aynı zamanda istedikleri test ve eğitim verisini seçebilmekte sonuçları çıktı olarak alabilmektedir. Çalışma İspanyadaki Vigo Üniversitesinde Bilgisayar Mühendisliği öğrencilerinin kullanımına sunulmuştur ve geri bildirimler alınmıştır. Katılımcı öğrencilerin %90.3'ü uygulamayı YSA öğrenmek için uygun görmüş ve %92.4 'ü programı kullanıcı dostu ve kullanması kolay bir program olarak nitelendirmiştir. Uygulama içerisinde, herhangi bir metin tabanlı ders içeriği veya canlandırma bulunmamaktadır. Öğrencilerin uygulamayı kullanabilmesi için YSA'nın temellerini ve çalışma mekanizması gibi konularda öncelikle ön bilgiye sahip olması gerekmektedir.

Venayagamoorty (2004), Java ile Amerika'daki Missouri-Roola Üniversitesinde sınıf ortamında temel YSA kavramlarının eğitilmesi için geliştirdikleri uygulamayı sunmuşlardır. Çalışmada birçok YSA modeli ve YSA eğitim algoritmasını incelenmiştir. YSA'ların eğitiminde kullanılan öğrenme döngüsü (epok), devinirlik (momentum), öğrenme katsayısı gibi parametrelerin ne işe yaradıkları, değiştirildiğinde ağın eğitimine olan etkisini uygulamalı olarak geliştirdikleri program üzerinde birçok örnek çözerek göstermişlerdir. Sonuç olarak, mühendislik gibi disiplinlerde geliştirmiş oldukları uygulama sayesinde YSA eğitiminin daha az çaba ve süre ile daha etkili bir şekilde gerçekleştirilebileceğini vurgulamışlardır. Çalışma öğreticilere ders ortamında anlatım yaparken yararlanabilecekleri bir materyal sunmuştur. Ancak öğrencilerin tek başına YSA'nın temel konuları ve çalışma prensibi hakkında bilgi sahip olup, deneyim kazanabilecekleri bir ortam sunmamaktadır. Uygulama kullanıcıların farklı problemleri çözebilmesi için gerekli olan ara yüzü sağlamamaktadır. YSA eğitimi ve test işlemleri sonucunda kullanıcıların detaylı sonuçlara erişemiyor olması YSA'nın temellerinin ve çalışma mekanizmasının tam anlamı ile kavranmasını zorlaştırmaktadır.

Bayındır ve Sesveren (2008), yapmış oldukları çalışmada YSA üzerinde yapılabilecek birçok işlem için kullanışlı ve etkili bir ara yüz tasarlamışlardır. İterasyon, devinirlik katsayısı, öğretme katsayısı, aktivasyon fonksiyonu gibi birçok değerin detaylı bir ara yüz ile değiştirilebilmesi ve değişikliklerin sonuca etkisinin eğitimi üzerinde durulmuştur. Bu etkinin izlenebilmesi için bir sürü grafik tasarlamışlardır. Çalışma aynı zamanda eğitim ve test verilerini içeri aktarma, sonuçları ise dışarı aktarıp kaydetme özelliklerini desteklemektedir. Çalışma içerisinde geliştirilen ara yüzün benzetim işlevi yapay sinir ağları konusunda yeterliliğe sahip 10 yüksek lisans öğrencisinin kullanımına verilmiştir ve bu öğrenciler ankete tabi tutulmuştur. Yazarlar anket sonuçları değerlendirildiğinde öğrencilerin çalışmaya ılımlı baktıkları, ara yüzün öğrenmeyi kolaylaştırdığı ve yapay zekâ öğretiminde benzetim kullanımının faydalı olacağı, ağın çalışma mekanizmasının incelenebilmesinin konunun kavranmasında son derece önemli olduğu çıkarımlarına ulaşmıştır. Yazarlar aynı zamanda bir YSA aracı hazırlarken karşılaşılan YSA yapısı, fonksiyon tipi,

öğrenme algoritması belirleme, performans düşmesi, başlangıç ağırlıklarının rastgele seçilmesinden dolayı aynı sonuçların üretilememesi gibi güçlükleri tespit etmiş ve aktarmışlardır. Hazırlanan çalışma eğitim amaçlı üretilmiş olmasına rağmen YSA hakkında ön bilgiye sahip kullanıcılara yönelik geliştirilmiştir. Uygulamanın modüler bir yapıya sahip olmaması bütün girdi ve çıktıların tek bir pencereden yönetilmesi uygulamanın çok karmaşık bir ara yüze sahip olmasına neden olmuştur.

Uğur ve Kınacı (2010), yapmış oldukları çalışmada YSA'nın Çok katmanlı algılayıcı, Özyinelemeli haritalar, Öğrenmeli vektör nicelemesi gibi modellerini ve ilgili eğitim algoritmaları içeren EasyLearnNN adını verdikleri uygulamayı, internet tabanlı olarak YSA eğitiminde kullanılmak üzere gerçekleştirmişlerdir. Çalışmada YSA'nın kullanıldığı alanlar, problem çözümlerinde YSA'ların sağladığı avantajlar üzerinde durulmuştur. Birçok farklı alanda birçok farklı araştırmacı, mühendis YSA'yı kullanıyor olmasına rağmen karmaşık yapısı yüzünden öğrenilmesi zor bir alan olduğu vurgulanmıştır. Bu yüzden YSA araçlarının kullanımının önemine dikkat çekmişlerdir. Birçok farklı YSA uygulaması, aracı ve kütüphanesinden bahsederek avantajlarını kendi uygulamalarında birleştirdiklerini belirtmişlerdir. Sistem nesne yönelimi yaklaşımlarla kodlanmıştır ve amaç eğitim olduğu için hız ve performans göz ardı edilmiştir. Ancak Uygulama içerisinde YSA için metin tabanlı ders içeriği bulunmamaktadır. Sınıf sınıf kodlama yapılmış, ağ modelindeki sinirler, katmanlar her biri farklı renkteki grafiklerle temsil edilmiş ve canlandırılmıştır. Sonuç olarak sistemin doğruluğu da kontrol edilmiştir ve yazarlar tarafından hesaplanan toplam hata 0,003773 olarak bulunmuştur. Ege üniversitesinde iki yıl boyunca uygulanan bu uygulama YSA dersini alan öğrenciler arasından seçilen ve YSA' ya ilgi duyan 30 kişiye uygulanmıştır. Öğrencilerin bu araca kolayca uyum sağladıkları, EasyLearnNN'i kullanarak YSA ile çalışmanın çok eğlenceli olduğunu söyledikleri belirtilmiştir. Yazarlar aynı zamanda bu uygulamanın YSA dersleri için yardımcı bir rehber olduğunu vurgulamaktadır.

Deperlioğlu ve Köse (2011), yapmış oldukları çalışmada geliştirmiş oldukları, kullanıcıların kendi YSA modellerini birleştirebileceği, birçok farklı öğrenme

algoritması kullanabileceği, etkileşimli ve görsel araçlarla donatılmış YSA için bir eğitim uygulaması sunmaktadırlar. Panellerle oluşturdukları uygulamada kullanıcılar her bir sinire tıklayarak ilgili sinirin çıkış değeri, aktivasyon fonksiyonu, bağlantılı olduğu ağırlık değerleri gibi bir sürü bilgi görülebilmektedir. Katman ekleme, sinir ekleme, ağ yapısını değiştirme, eğitim algoritmasını ve aktivasyon fonksiyonunu değiştirme özellikleri de mevcuttur. Nesne yönelimli yaklaşım ile kodladıkları uygulama web kullanılan veriler için XLS, CSV, DAT, XML gibi bir sürü dosya formatını desteklemekte, farklı programlar ile ortak çalışabilmektedir. Yazarlar sistemi Afyon Kocatepe Üniversitesinde yüksek lisans eğitimlerinde verilen derslerde kullanılmaktadır ve sistemin öğrencilerin ilgili YSA ve zeki sistem konularındaki bilgilerini geliştirmesini, etkili öğrenme tecrübesi sağlamasını tecrübe etmişlerdir. Çalışma web tabanlı olmadığı için uzaktan erişime imkân vermemektedir. Yazarlar gelecek çalışmalarda uygulamayı daha görsel ve etkileşimli hale getirerek sanal bir laboratuvar yapmak istediklerini belirtmişlerdir.

Literatürde yapılan çalışmaların çoğu YSA öğrenimin zor bir konu olması nedeni ile bir eğitim aracı geliştirmek amacı ile yapılmıştır. Buna rağmen geliştirilen çalışmaların bir kısmının karmaşık bir yapıya sahip olması, kullanmak için fazla önbilgi gerektirmesi, bazı çalışmaların öğrencilerin ilgisini azaltacak karmaşık bir yapıya sahip olması ve YSA öğretiminde kullanılacak bir içeriği sahip olmayışı tam anlamı ile gereksinimleri karşılayan bir yazılıma duyulan ihtiyacı gidermemektedir. Bundan dolayı yapılan çalışmaların avantajlarının bir araya getirilerek yeni bir çalışma içerisinde toplanması amaçlanmıştır.

3. YAPAY SİNİR AĞLARI İÇİN WEB TABANLI BİR EĞİTİM YAZILIMI GELİŞTİRİLMESİ

Yapay zekâ dersleri üniversitelerin bilgisayar mühendisliği, bilgisayar programcılığı gibi bölümlerinin müfredatında bulunmaktadır. Bu dersin işlenişi sırasında yararlanılan kitap, ders notu gibi kaynaklar YSA gibi karmaşık bir doğaya ve birçok matematiksel formüle sahip yapay zekâ alanlarının anlaşılmasında yetersiz kalmaktadır. Daha önce bahsedilen programların çoğu uygulama esnasında kullanılıyor olmasına rağmen tam olarak eğitim amacına hizmet etmemesi, çok karmaşık olması ve kullanmak için çok fazla ön bilgi gerektirmesi gibi nedenlerden dolayı bu alanda ihtiyacı karşılayamamaktadır. Bunun yanında bu programların çoğunun öğrencinin ilgisini azaltacak karmaşık bir yapıya sahip olması, YSA öğretiminde kullanılacak içeriğe sahip olmayışı ve YSA eğitimi için gereksiz birçok şeyi barındırması da kendi YSA eğitim yazılımımızı yapma kararı almamızda etkili olmuştur.

YSA, dinamik bir yapı olduğundan dolayı tam anlamı ile anlaşılabilmesi için öğrenci tarafından eğitim ve test sürecinin her adımında takip edilmesi gerekmektedir. Bu gibi süreçlerde öğrencilerin gerçekleşen adımları kavrayabilmesi için sürecin canlandırmalarla desteklenmesi gerekmektedir.

Tasarlayıp geliştirdiğimiz yazılımın web tabanlı olması da Yapay zekâ dersi içerisinde YSA konusunu alan veya internet bağlantısına sahip bilgisayar mühendisleri, araştırmacılar, istatistikçiler, matematikçiler ve YSA 'ya ilgi duyan kişilerin erişip kullanabilmesi için tercih edilmiştir. Böylelikle öğrenciler ders dışında da kendi başlarına bu yazılımı kullanıp YSA'nın temellerini öğrenebilecek, YSA'yı çalıştırıp çalışma mekanizmasını kavrayabilecektir.

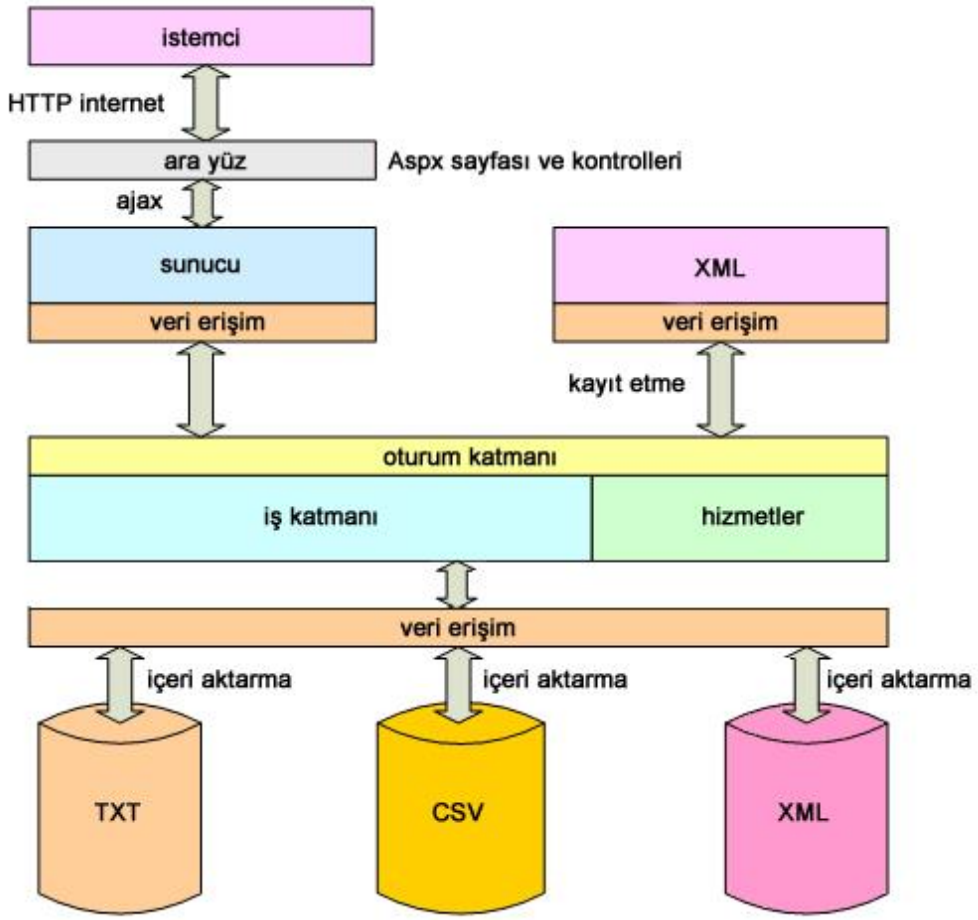
Kendi yazılımımızı gerçekleştirmemizin bir diğer nedeni de keşfedici öğrenmeyi destekleyen bir YSA eğitim yazılımına duyulan gereksinimdir. Geleneksel yöntemlerle YSA'nın temellerini anlatmak YSA'nın doğasına zaten başlı başına aykırı bir durumdur. Öğrencinin YSA'nın temellerini ve çalışma mekanizmasını

keşfederek öğrenmesi, ileride ihtiyaç duyduğu özgün problemlerin çözümünde kendi özgün YSA modellerini tasarlayabilmesine katkı sağlayacaktır.

Ayrıca hazırlanan sistemin nesne yönelimli yaklaşımla gerçekleştirilmesi ileride üzerine kolayca ekleme yapılmasına izin vermektedir. Yazılım yeni YSA modellerinin ve eğitim algoritmalarının eklenebileceği, güncellenebilecek bir yapıya sahiptir. Bu sistemi sürekli güncel ve kullanışlı tutacaktır. İleride ihtiyaç duyulması halinde mobil ortama da hızlı bir şekilde aktarılabilecek alt yapıya sahip olmuş olacaktır. Bununla beraber nesne yönelimli yaklaşım kullanarak yazılım geliştiren herhangi bir kişi hazırlamış olduğumuz alt yapıyı gerek web servisleri gerekse kütüphane olarak kendi sistemine dâhil ederek kendi uygulamasını yapabilecektir.

3.1. Sistemin Tasarlanması

Sistem tasarımında Visual Studio 2012 uygulama geliştirme ortamından yararlanarak ASP.NET platformu kullanılmıştır. Bu sayede güçlü ve etkili bir nesne yönelimli programlama dili olan C#'ı tercih edilebilmiştir. Buna ek olarak Ajax gibi web ortamında kullanılan güncel yaklaşımları da javascript dili ile birlikte projeye dâhil etme imkânı olmuştur. Şekil 3.1.'de geliştirilen sistemin mimarisi gösterilmektedir.

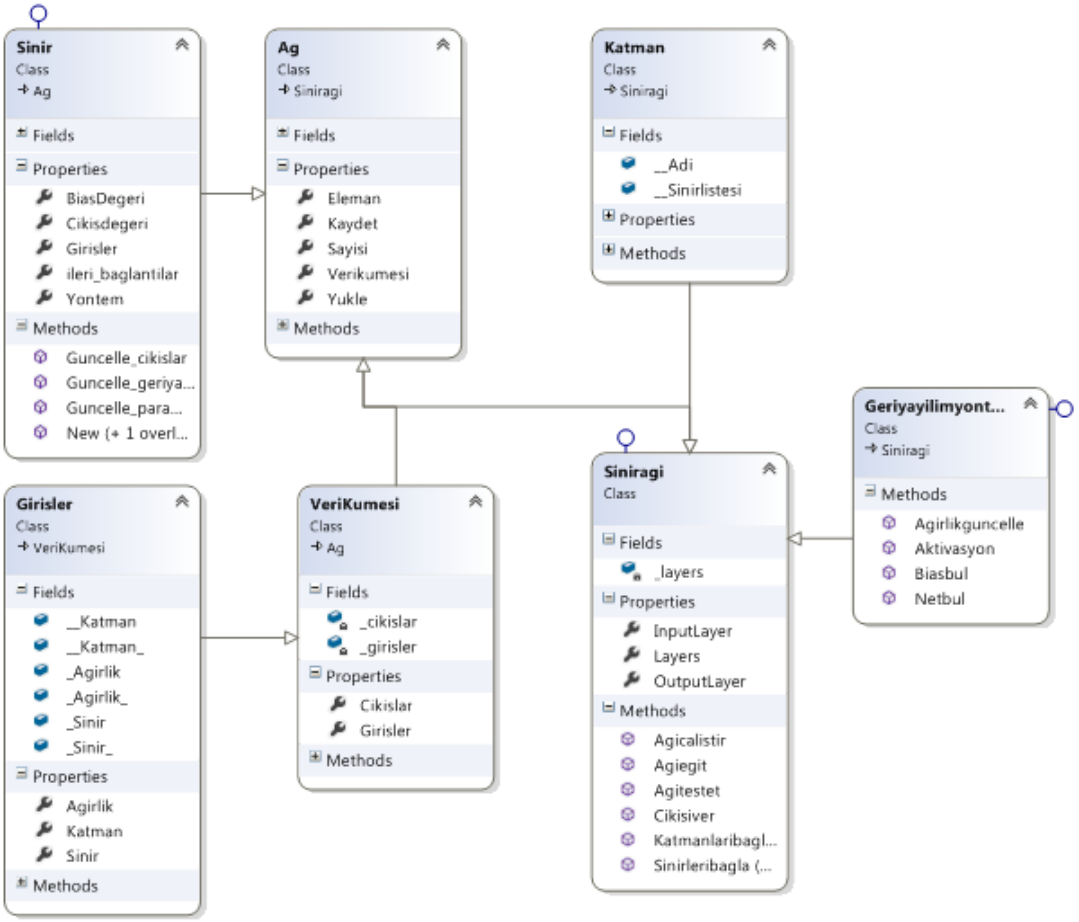


Şekil 3.1. Geliştirilen sistemin mimarisi

Görünüm kısmında ASP.NET kontrollerinin ve HTML kontrollerinin tutulduğu bir ASPX sayfası mevcuttur. İlgili kontroller kullanılarak bu sayfaya kullanıcılara gösterilmek istenen içerik ve benzetim için gerekli olan nesneler yerleştirilmiştir. Kullanıcının etkileşime geçmek istemesiyle birlikte, ilgili işlemin gerçekleşmesi ve sonucun kullanıcıya görsel kısımda geri bildirilebilmesi için kod içeren sunucu sayfasına ulaşmak gerekmektedir. Bu çözüm iki farklı şekilde gerçekleştirilebilmektedir. Web sayfasında kullanıcı herhangi bir etkileşime geçerse bütün sayfa sunucu tarafına gönderilir, sunucu işlemi yapar ve bütün sayfayı geriye gönderir. Bu durum hem çok büyük bir veri

transfer yüküne mal olur hem de sayfanın yenilenmesi ile sayfa üzerinde çalışılan modelin veya ilgili ayarların kaybolmasına yol açabilir, bu yüzden ikinci yöntem olan javascript'in kullanılmasıyla Ajax nesnelerinden yararlanılmıştır. Böylece bütün sayfayı yenilemeden sadece ilgili kontrollerden bilgiler alınarak sunucuya ajax üzerinden yollanmıştır ve sunucudan gelen değerler aynı şekilde ilgili nesneler üzerine yerleştirilmiştir. Kullanıcının sayfa yenilemesi yapmadan sadece istediği kontrollere ve içeriğe erişebilmesi sistemin kullanışlılığını ve görselliğini artırmıştır. Sistem internet üzerinden çalıştığı için veri trafiği önemli bir konudur, kullanmış olduğumuz yöntem veri trafiğini azalttığı için bu noktada önemli bir katkı sağlamıştır.

Geliştirilen sistemde en önemli kısım iş katmanıdır. YSA için uygun bir eğitim ve benzetim ortamı tasarlayabilmek için YSA'nın çalışma mekanizması, öğrenme algoritmaları ve diğer bir sürü gerekli özellikleri eksiksiz bir şekilde burada tanımlanan sınıflar içerisinde kodlanmıştır (Şekil 3.2.). Sistemin çekirdeğini oluşturan bu sınıflar her biri gerek geliştirilen ağın benzetimi gerekse sistemdeki diğer işlevler için ayrı ayrı hesaplamalar yapan görevlere sahiptir. Böylece bir sınıfın yapısında gerçekleştirilen bir değişiklik diğerini etkilememektedir.



Şekil 3.2. Geliştirilen sistemin sınıf şeması

YSA'nın en temel öğesi olmasından dolayı gerçekleştirdiğimiz "Sinir" sınıfı tanımlanan en temel sınıftır. Sistemde kullanılan her sinir için bu sınıftan bir tane türetilir ve ilgili sinirin bilgileri bu sınıf içerisindeki değişkenler üzerinde tutulur. Eğitim ve test veri kümeleri ilgili tasarıma göre “Girisler” ve “Verikumesi” sınıfları ile şekillendirilerek ağ oluşturulur ve “Ağ” sınıfı üzerinde tutulur. “Siniragi” sınıfı ağın eğitim ve test işlemlerini gerçekleştirmek için kullanılan sınıftır. Bu işlemleri gerçekleştirirken “Geriyayilimyontemi” sınıfının yapmış olduğu hesaplamaları da kullanmaktadır. İçeri veri aktarma, veri kaydetme, veri kümeleri üzerindeki işlemler gibi görevleri üstlenen "Verikumesi" sınıfı veri erişim katmanı üzerinde çalışmaktadır. İçerisinde tanımlanmış veri sağlayıcılar (provider) ile CSV, TXT ve XML dosya formatlarından veri okuyup yazabilmesini ve bu verileri istenilen biçime dönüştürüp sistem içerisinde kullanılabilmesini sağlamaktadır.

Bu sınıfların haricinde canlandırmaların ya da ders içeriklerinin gösterimi gibi sistemde birçok farkı işlevin gerçekleşmesi için gerekli işlevler hizmetler katmanında çalışan "Hizmetler" sınıfı içerisinde tanımlanmıştır.

3.2. Sistemin Gerçekleştirilmesi

Geliştirilen sistemde kullanıcının YSA ile ilgili işlemleri yapabilmesi için geri yayılım öğrenme algoritmasını kullanan, çok katmanlı bir YSA modeli tasarlanıp geliştirilmiştir. Geri yayılım algoritması, çok katmanlı ağlarda ağırlıkları öğrenmek için kullanılır. Hesaplama karmaşıklığı ağdaki ağırlık sayısına göre doğrusal olarak değiştiği için hesaplama yönünden etkilidir. Bu algoritmanın geliştirilmesiyle çok katmanlı ağlar en yaygın kullanılan YSA modeli haline gelmiştir (Yegnanarayana, 2009).

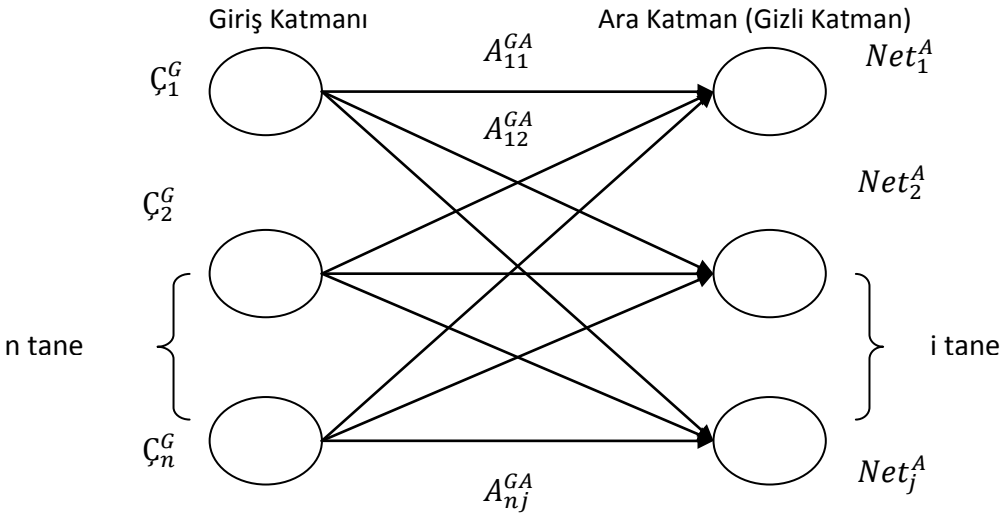
Giriş, gizli ve çıkış katmanından oluşan YSA modelinde ilk hesaplamalar girişleri ileri katmanlara yaymak için yapılmıştır. Giriş katmanındaki her bir sinirin değeri Denklem 3.1 'deki gibi direk çıkış kabul edilmiştir. Çünkü giriş katmanında hiç bir işlem yapılmaz, bu yüzden girişler direk giriş katmanının çıkışı olarak kabul edilmiştir. Şekil 3.3'de bu işlemin nasıl kodlandığı gösterilmektedir. giris_listesi giriş katmanındaki her bir siniri temsil eden sinir sınıflarını içerisinde tutmaktadır ve bunu bir döngü içerisinde döndürerek her bir veri kümesi örneği için giriş değerlerinin atamasını yapmaktadır. Çıkış listesinde tutulan çıkış sinirlerinin temsil edildiği sinir sınıflarına ise ağın gerçekleştirdiği hatayı hesaplarken karşılaştırma yapabilmek için beklenen gerçek çıkış değerleri atanmaktadır.

```
for (int donme_sayisi = 0; donme_sayisi < epok; donme_sayisi++) //epok  kadar tekrar eden bir döngü
{
    Epok_Suresi.Start();
    ortalama_hata = 0; //her bir epoktaki iterasyonların ortamalama hatasını tutan değişken
    foreach (ArrayList item in veri_listesi) //Her veri satırını döndüren döngü
    {
        #region Veri setinin girişini ve beklenen çıkışın nöronlara atanması
        for (int i = 0; i < item.Count - 1; i++) //giriş değerlerinin atanması
        {
            giris_listesi[i + 1].cikis = double.Parse(item[i].ToString().Replace('.', ',')); //i+1 dememizin nedeni biası atlamamız.

        }

        for (int i = 0; i < cikis_listesi.Count; i++) //beklenen çıkışların ikilik sisteme çevrilip nöronlara atanması
        {
            if (i.ToString() == item[item.Count - 1].ToString())
            {
                cikis_listesi[i].beklenen_cikis = 1;
            }
            else
            {
                cikis_listesi[i].beklenen_cikis = 0;
            }
        }
```

Şekil 3.3. Giriş değerlerinin giriş katmanındaki çıkışlara aktarılması ve çıkış katmanındaki beklenen değerlerin atanması

$$(Ç_i^G = Gi) \tag{3.1}$$

Burada Gi giriş katmanındaki her bir siniri, $Ç_i^G$ ise giriş katmanındaki her hangi bir sinirin çıktısını ifade etmektedir. Bu noktadan sonra bu çıkış değerlerinin bir sonraki katmana iletilmesi yani ileri yayma işleminin gerçekleşmesi gerekmektedir. Şekil 3.4.'de de görülen katmanlardaki sinirleri birbirine bağlayan ve öğrenmeyi sağlayan ağırlıkları kullanarak bir sonraki katman olan ara katmandaki her bir sinirin net değerini hesaplamak için Denklem 3.2 kullanılmıştır.



Şekil 3.4. Giriş ve ara katman arasındaki ağırlık bağlantıları

$$Net_j^A = \sum_{k=1}^{n} A_{kj}^{GA}.\ Ç_k^G \quad (3.2)$$

Denklemde Net_j^Aile gösterilen değişken ara katmanda bulunan her bir j siniri için toplam net değerinin temsili için kullanılmaktadır. A_{kj}^{GA}giriş katmanından ara katmana doğru olan, ara katmandaki bir sinire gelen tüm ağırlıklar kastedilmektedir. Bu ağırlıklar $Ç_k^G$ ifadesi ile birlikte ilgili ağırlığın bağlı olmuş olduğu giriş katmanındaki sinirin çıkış değeriyle çarpılmakta ve tüm çarpımlar toplanarak ara katmandaki sinirin net değerini oluşturmaktadır. Ara katmandaki her bir sinir için hesaplanan net değerlerin çıkışa dönüşmesi için bir aktivasyon fonksiyonundan geçirilmesi gerektir. Burada sigmoid fonksiyonunu kullanılmıştır ve Denklem 3.3 'deki ifadeden yararlanılarak hesaplama yapılmıştır. Her bir sinire gelen net toplam değerlerinin ve bu toplamın sigmoid aktivasyon fonksiyonundan geçirilmesi için yapılan kodlama Şekil 3.5.'de gösterilmektedir. Ara katmandaki her bir siniri sağlayan bir döngü içerisinde gerçekleştirilen işlemde Linq to xml sorgulama dilinin yardımı ile ilgili sinire gelen bütün ağırlıklar tespit edilmekte ve o ağırlığın sahibi olan giriş sinirinin çıkışı ile çarpılmakta ardından tüm çarpımlar bir toplama metodu olan ".Sum" komutu ile toplanmaktadır. Böylece her bir ara katmandaki sinirin net değeri hesaplanabilmekte, C# dilinin kendi kütüphanesinde içerisindeki “Math" sınıfının yardımı ile hesaplanan net değeri kolayca sigmoid fonksiyonundan geçirilebilmektedir. Burada ara katmandaki sinirler için yapılan her bir işlem çıkış katmandaki sinirler içinde gerçekleştirilmektedir.

```
#region Sırayla ileri yaymanın yapılması

int sayac = 0; //Kaçıncı ara eleman olduğunu sayan değişken
//her bir ara katman nöronu için net ve transfer fonksiyonundan geçirilmesi
foreach (noron araeleman in ara_listesi)
{
    araeleman.net = (from p in agirlik_listesi
                     where p.alan == araeleman
                     select p).AsEnumerable().Sum(o => o.agirlik * o.gonderen.cikis);
    // net değer hesaplama giriş norönlarının agırlık ve çıkışların çarpılıp toplanması

    araeleman.cikis = 1 / (1 + Math.Exp(-araeleman.net));
    //aktivasyon fonk ile çıkışın hesaplanması

    sayac++;
}
```

Şekil 3.5. Net değerin hesaplanması ve aktivasyon fonksiyonu

$$Ç_j^A = \frac{1}{1+e^{-(Net_j^A+\beta_j^a)}} \tag{3.3}$$

Eşitlikte gözüken $Ç_j^A$ değişkeni ara katmandaki her bir j sinirinin çıkış değerini temsil etmektedir. Net_j^A daha önce Denklem 3.2 'de hesaplanan her bir j sinirinin net değeridir. β_j^a ise ilgili sinirin eşik değer ağırlığını temsil etmektedir. Bu değerin çıktısı sabit olup 1'e eşitlenmiştir. Çıkış katmanına doğru bu değerlerin aktarılıp net ve çıkış değerlerinin hesaplanması için Denklem 3.2 ve Denklem 3.3.'deki hesaplamaların aynısı çıkış katmanı için tekrar edilmiştir. Böylece son katmandaki sinirlerin çıkışları bulunmuştur. Sistemin yapmış olduğu hatayı hesaplamak için çıkış katmanındaki her sinirin hatasını Denklem 3.4 kullanılarak hesaplanmış ve toplam hata Denklem 3.5'deki eşitlikle bulunmuştur. Her bir çıkış siniri için hatanın hesaplanması Şekil 3.6.'de ve tüm çıkış sinirlerinden elde edilen ortalama kare hatası (MSE) Şekil 3.7.'de verilen kodlamalar ile bir kaç denklem birleştirerek tek seferde gerçekleştirilmiştir.

```
cikis_elemani.hata = cikis_elemani.cikis
                        * (1 - cikis_elemani.cikis)
                            * (cikis_elemani.beklenen_cikis - cikis_elemani.cikis);

//her bir çıkış nöronunun bireysel hatasının hesaplanması
```

Şekil 3.6. Her bir çıkış sinirinin bireysel hatasının hesaplanması

```
var toplam_hata = 0.5 * (from p in cikis_listesi
                         select p).AsEnumerable().
                  Sum(o => Math.Pow(o.cikis * (1 - o.cikis) * (o.beklenen_cikis - o.cikis), 2));

//MSE
```

Şekil 3.7. Ortalama kare hatasının bulunması

$$E_m = B_m - Ç_m^{Ç} \tag{3.4}$$

$$TH = \frac{1}{2}\sum_m E_m^2 \tag{3.5}$$

$E_m$yani çıkış katmanındaki her bir m sinirinin hatası, o sinirin beklenen çıkışı (B_m) ile gerçekleşen çıkışının ($Ç_m^{Ç}$) farkı alınarak hesaplanmıştır. Daha sonra ortalama kare hatayı (TH) bir diğer isimle MSE'yi bulmak için tüm çıkış sinirlerin hatalarının kareleri toplanıp yarısı alınmıştır. YSA ile bir problemin çözülmesi demek ilgili TH değerinin en aza inmesi demektir. Bunu gerçekleştirmek için ilgili sinirlerin aralarındaki ağırlıkları, hatayı geri yayarak tekrar hesaplamak gerekmektedir. Öncelikle ara katman ve çıkış katmanı arasındaki ağırlıkların değişim miktarını hesaplamak için Denklem 3.6 kullanılmıştır.

$$\Delta A_{jm}^{AÇ}(t) = \gamma \delta_m Ç_j^{A} + \alpha \Delta A_{jm}^{AÇ}(t-1) \tag{3.6}$$

Denklemde ara katmandaki her bir j sinirini çıkış katmandaki m sinirine bağlayan ağırlığın ($\Delta A_{jm}^{AÇ}(t)$) hesaplanması için önce öğrenme katsayısı γ , çıkış katmanındaki m sinirin hatası δ_m ve ara katmanındaki j sinirinin çıkış değeri çarpılmıştır. Daha sonra buna devinirlik katsayısı olan α ve ağırlığın bir önceki değerinin $\Delta A_{jm}^{AÇ}(t-1)$ çarpımını eklenerek ilgili ağırlıktaki değişim miktarı hesaplanmıştır. Öğrenme katsayısı ve devinirlik değeri kullanıcı tarafından belirlenebilmektedir. Çıkış katmanındaki m sinirinin hatası olan δ_m 'i hesaplarken ise Denklem 3.7'den yararlanılmıştır.

$$\delta_m = Ç_m^{Ç}(1 - Ç_m^{Ç})E_m \tag{3.7}$$

Ara katman ile giriş katmanındaki her bir ağırlığın değişim miktarı hesaplandıktan sonra ilgili ağırlığın t anındaki değerini Denklem 3.8 'den yararlanılarak hesaplanmıştır.

$$A_{jm}^{AÇ}(t) = A_{jm}^{AÇ}(t-1) \ + \Delta A_{jm}^{AÇ}(t) \tag{3.8}$$

Ara katman ile çıkış katmanı arasındaki ağırlıklardaki değişim ve ağırlıkların yeni değerleri hesaplandıktan sonra eşik değerlerin ağırlıkları üzerindeki değişimler de aynı yöntem kullanılarak Denklem 3.9'da görüldüğü gibi hesaplanmıştır.

$$.\Delta A_{jm}^{BÇ}(t) = \gamma \delta_m Ç_j^B + \alpha \Delta A_{jm}^{BÇ}(t-1) \tag{3.9}$$

Giriş katmanı ve ara katmandaki ağırlıkların yeni değerlerinin hesaplanması için Denklem 3.10 kullanılarak ilgili ağırlıklardaki değişim miktarı hesaplanmıştır.

$$\Delta A_{kj}^{GA}(t) = \gamma \delta_j^A Ç_k^G + \alpha \Delta\Delta A_{kj}^{GA}(t-1) \tag{3.10}$$

δ_j^A'yi ise Denklem 3.11'deki gibi hesaplanmıştır.

$$\delta_j^A = Ç_j^A (1 - Ç_j^A) \sum_m A_{jm}^{AÇ} \delta_m \tag{3.11}$$

Son olarak giriş katmanı ile ara katman arasındaki ağırlıkların t anındaki değeri yani kazanması gereken yeni değer ise Denklem 3.12'den yararlanılarak hesaplanmıştır. Her bir sinirin δ değeri ve her bir ağırlığın kazanması gereken yeni değeri değişim miktarları ile birlikte Şekil 3.8.' de ki kodlamalar ile gerçekleştirilmiştir.

```
foreach (noron noron_ara in ara_listesi)
{
    var hata_on_hesabi = (from p in agirlik_listesi
                          where p.gonderen == noron_ara
                          select p).AsEnumerable().Sum(o => o.alan.hata * o.agirlik);
    noron_ara.hata = noron_ara.cikis * (1 - noron_ara.cikis) * hata_on_hesabi;
}


#region Bağlantıların güncellenmesi


foreach (baglanti yeni_baglanti in agirlik_listesi) // bağlantıların güncellenerek eğitimin yapılması
{
    yeni_baglanti.agirlik += (alfa * yeni_baglanti.gonderen.cikis * yeni_baglanti.alan.hata)
        + (momentum * yeni_baglanti.agirlik_gecmis);

    yeni_baglanti.agirlik_gecmis = yeni_baglanti.agirlik;
}
#endregion
```

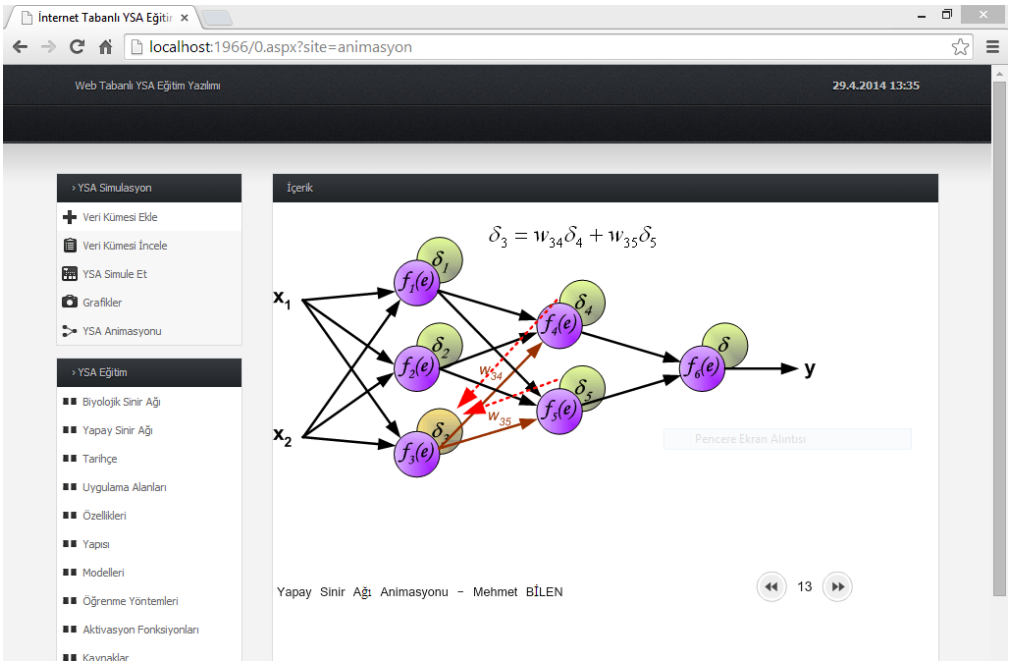
Şekil 3.8. δ ve ağırlık değişimlerinin hesaplanması ve güncellenmesi

$$A_{kj}^{GA}(t) = A_{kj}^{GA}(t-1) + \Delta A_{kj}^{GA}(t) \tag{3.12}$$

Tüm ağırlıklar güncellendikten sonra bir öğrenme döngüsü gerçekleştirilmiş olmaktadır. Hazırlanılan sistem kullanıcının verdiği öğrenme döngüsü sayısı kadar hesaplamaları tekrar ettirerek, tüm ağın eğitim işlemini gerçekleştirebilmektedir.

3.3. Geliştirilen Sistemin Özellikleri

İnternet ortamında YSA eğitimi için tasarlanan sistem, öğrencinin ilgisini kaybetmeyeceği basit bir yapıya sahip olması için kullanımı kolay ara yüze sahip bir web sayfası şeklinde hazırlanarak, içerisine paneller ve ilgili modüllere ulaşılabilmesi için linkler yerleştirilmiştir. Panellerin ve modüllerin sayfa üzerinde yerleşimi ve genel görünüm Şekil 3.9.'da gösterilmiştir. YSA Benzetim ve YSA Eğitim adı verilen iki adet panel içerisinde ilgili içeriğe ulaşabilecekleri modül isimleri listelenmiştir.



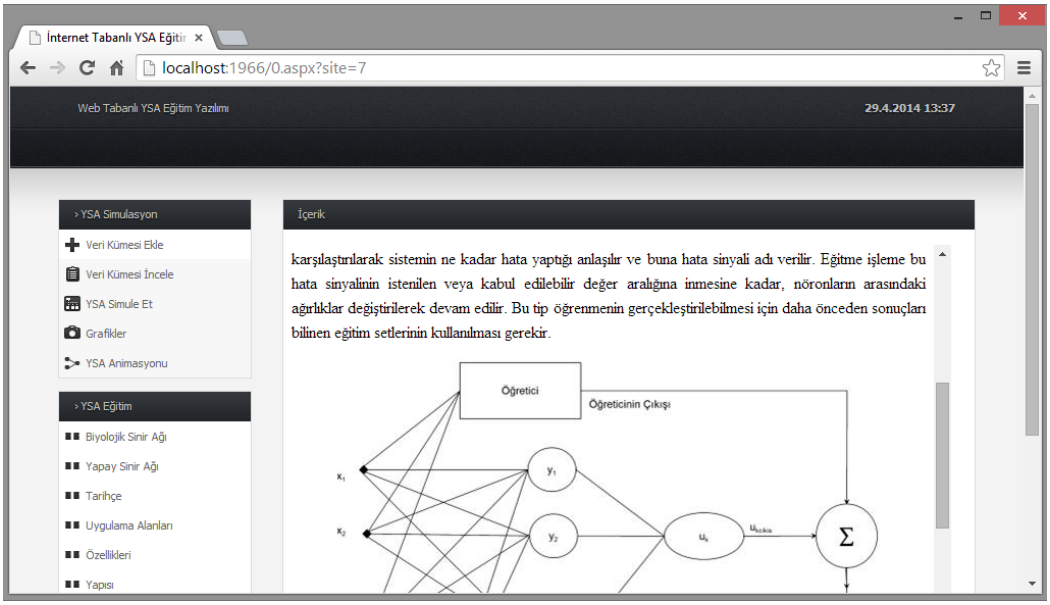
Şekil 3.9. Geliştirilen yazılımın genel görünümü

3.3.1. YSA eğitim paneli

Sayfasının sol altına yerleştirilen YSA Eğitim panelinde sistemin kullanan kişiye YSA'nın temellerinin aktarılabilmesi için gerekli ders notlarına ulaşılabilecek bir liste bulunmaktadır. Bu paneli kullanarak kullanıcılar ihtiyaç duydukları temel bilgiye hızlıca ulaşabilmektedir. Listenin en başında YSA'nın temel aldığı insan beyni ve davranışını anlatan biyolojik sinir ağları, daha sonra bundan esinlenerek oluşturulmuş basit bir Yapay Sinir Ağı'nın yapısına ait içeriklere ulaşılabilmektedir. Tarihçe alanında YSA'nın ilk tanıtıldığından günümüze kadar nasıl gelişim gösterdiği ve kimler tarafından hangi çalışmalarda kullanıldığı açıklanmaktadır. Uygulama alanlarında ise kullanıcılara tasarlayıp geliştirecekleri YSA modellerinde ve algoritmalarında fikir vermesi için şu an YSA'ların nerelerde, hangi alanlarda ne şekillerde kullanıldığını anlatan bir içerik sunulmaktadır. YSA'nın sahip olduğu doğrusal olmama, öğrenebilme yeteneği, hata toleransı, genelleme ve uyum sağlayabilmesi gibi özellikleri Özellikler modülüne tıklayarak açılan sayfada anlatılmaktadır. YSA'yı meydana getiren katmanlar, sinirler ve ağırlıklar ise Yapısı başlığı altında verilmektedir. MLP, SOM, Hopfield, LVQ, Boltzman Machine gibi yaygın olarak kullanılan

popüler YSA modelleri ise Modelleri başlığı altında kullanıcıya sunulmaktadır. Geri beslemeli ve ileri beslemeli YSA ağları ilgili başlıklarda ulaşılabilmektedir. Ağın çalışma yapısını önemli bir şekilde etkileyen çeşitli aktivasyon fonksiyonlarına ilgili linkten erişilebilmektedir.

Tüm ilgili içerikler, internet ve masaüstü yazılım geliştirme ortamlarında oluşturulan projeler için standartlaştırılmış bir dosya formatı olan XML ile tutulmaktadır. Böylece ilgili içerik istenilen her projeye dâhil edilerek farklı kullanıcılar ve yazılım geliştiriciler tarafından rahatlıkla kullanılabilmektedir. Şekil 3.10.'da örnek bir YSA'nın temellerini anlatan içeriğin sunulduğu sayfa gösterilmektedir.



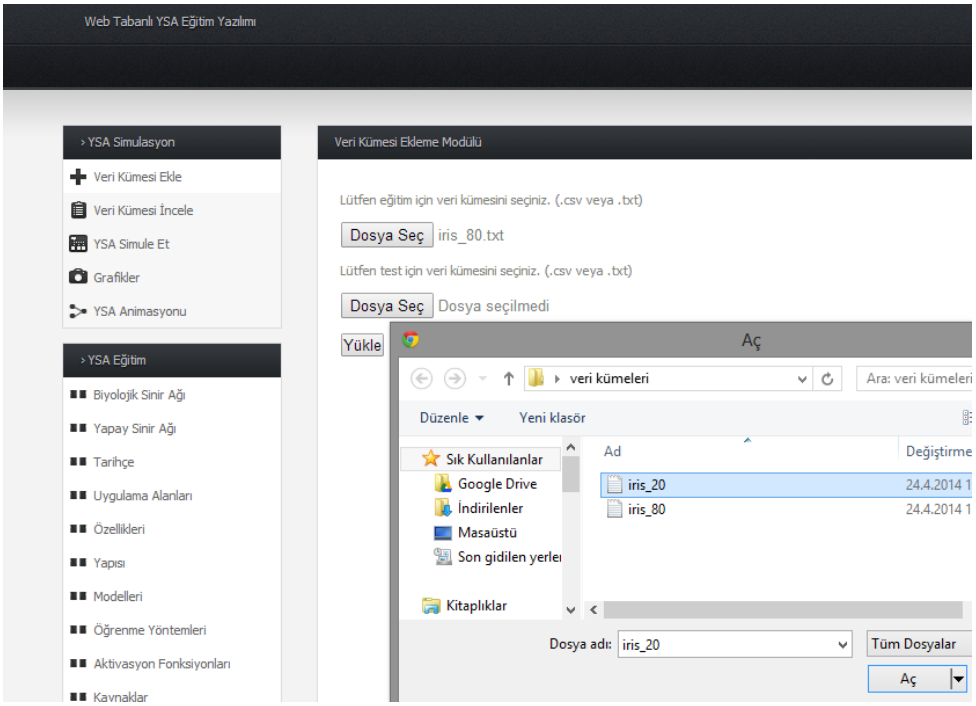
Şekil 3.10. Örnek bir içerik sayfası

3.3.2. YSA benzetim paneli

Bu panelde kullanıcılar bir veri kümesinin içeri aktarılmasından ağın eğitilip test sonuçlarının gösterilmesine kadar birçok işlem gerçekleştirebilmektedir. Bu işlemler karmaşıklığı önlemek ve modüler öğrenmeyi sağlamak üzere YSA'nın çalışma yapısı da göz önüne alınarak 5 farklı modüle ayrılmıştır.

3.3.2.1. Veri kümesi ekleme modülü

Ağın eğitilebilmesi için öncelikle bir eğitim veri kümesine ihtiyaç vardır. Ağın öğrenimi gerçekleştikten sonra, sistemin performansını test etmek için kullanılmak üzere test veri kümesine ihtiyaç duyulmaktadır. Şekil 3.11. 'de veri kümesi ekleme modülü içerisindeki ilgili veri kümelerinin yüklenebilmesi için geliştirilmiş dosya yükleme nesneleri ve veri kümelerini içeri aktarırken gözüken diyalog penceresi görülmektedir.



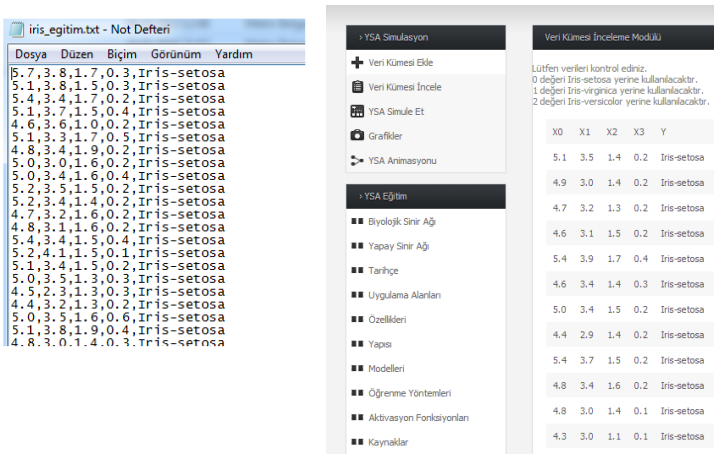
Şekil 3.11. Eğitim ve test veri kümesinin içeri aktarılması

Hazırlanan sistem veri kümelerinin hazırlanmasında "CSV" ve ".TXT" gibi en çok kullanılan formatlar desteklenmektedir. Bu sistemin kullanışlılığını artırmıştır ve kullanıcılar tarafından sıklıkla kullanılan diğer programlarla beraber çalışabilmesini sağlamıştır.

Eğitim ve test için istenilen veri kümeleri seçildikten sonra yükle butonu kullanılarak içeri aktarılma işlemi başlatılabilmektedir. İçeri aktarılan dosyalar sistemin kullanması için uygun hale getirilerek sunucuda saklanmaktadır.

3.3.2.2 Veri kümesi inceleme modülü

Bu modülde kullanıcılar bir önceki anlatılan veri kümesi ekleme modülünde ekledikleri veri kümelerinin sistem tarafından uygun hale getirilip düzenlenmiş halini inceleyebilmektedir. Dolayısıyla kullanıcıların bu modülü kullanabilmesi için öncelikle veri kümelerinin eklenmiş olması gerekmektedir, yoksa sistem otomatik olarak veri kümesi ekleme modülüne kullanıcıyı yönlendirecektir. Şekil 3.12.'de veri kümesinin içeri aktarılmadan önceki hali ve sistem tarafından düzenlenmiş hali gösterilmektedir.



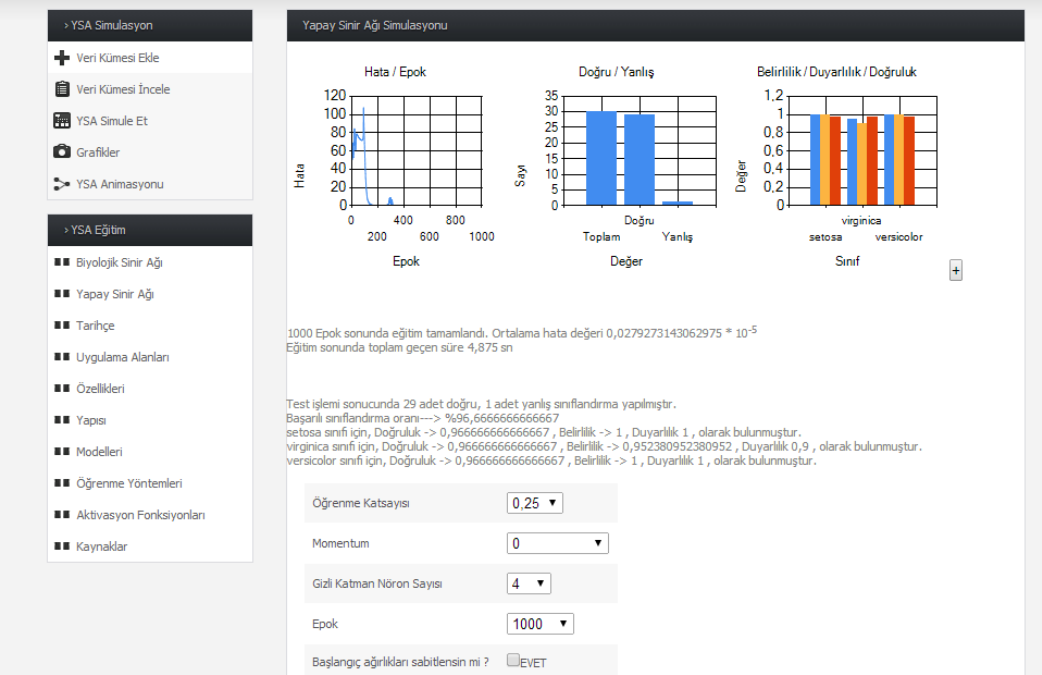
Şekil 3.12. Veri kümesinin içeri aktarılmadan önceki ve sonraki hali

Veri kümeleri dosya formatları içinde birçok farklı şekilde tutulabilir. Bunun bir örneği bazı ülkelerde ondalıklı sayıları ifade etmek için kullanılan "." karakteridir. Oysa ülkemizde ondalıklı sayıları ifade etmek için "," işareti kullanılır. Hatta ülkemizde kullanılan "," karakteri veri kümelerindeki her bir örneğin kendi içerisindeki değerlerini ayırmak için kullanılabilir. Eğer hiç bir işleme tabi tutulmadan dosyalar sistemde bu şekilde kullanılmaya çalışırsa karakter farklılıkları gibi nedenlerden dolayı sayı değerlerinde büyük farklılıklar meydana gelecektir ve sistem doğru çalışmayacaktır. Bundan dolayı sistem ilgili karakterleri ve diğer özellikleri ülkemizde kullanıldığı şekli ile dönüştürerek kullanmaktadır.

Eğitim için dâhil edilen veri kümeleri eğitim işleminin anlamlı olabilmesi için daha önceden sınıflara ayrılmış veri kümeleri olmak zorundadır. Böylelikle sistem hangi giriş değerlerinin hangi çıkışları verdiğini görerek giriş ve çıkış değerleri arasında bağlantı kuracaktır, yani öğrenecektir. Bu modülün gerçekleştirdiği işlemlerden biri de giriş ve beklenen çıkış değerleri, yani sınıf değerleri veri kümesi içerisinden tanımaktır. Sistemde her bir giriş değeri ilgili modül tarafından (Xn) simgesi ile temsil edilmek üzere ayarlanmaktadır. Beklenen çıkış değerleri ise (Y) ile temsil edilmektedir. Bu noktada bir diğer yapılması gereken şey ise beklenen çıkış değerlerinin yani sınıf isimlerinin sayısal değerlere dönüştürülmesidir. Çünkü YSA sistemleri sayısal ifadelerle çalışabilmektedir. Bu problemi çözmek için modül her bir farklı çıkış sınıfına sırası ile doğal sayılar kümesinden temsili değerler atamaktadır. Örneğin sistemin doğruluğu ve kararlılığını denemek için kullandığımız "Iris" veri kümesine ait üç farklı sınıf vardır. Bu sınıflar," Iris-Setosa", "Iris-Versicolor", "Iris-Virginica" 'dır. Modül bunları Şekil 3.12.'de görüldüğü gibi "Iris-Setosa=0", "Iris-Versicolor=1", "Iris-Virginica=2" olarak değiştirmektedir.

3.3.2.3. YSA benzetim modülü

YSA'nın yapısının kullanıcı tarafından anlaşılabilmesi ve çalışma mekanizmasının kavranabilmesi için en önemli modül YSA benzetim modülüdür. Burada kullanıcı bir YSA'nın eğitebilmesi için gerekli olan bazı parametreleri değiştirip ağın davranışında meydana gelen değişiklikleri görebilmektedir. Şekil 3.13.'de standart olarak seçili gelen parametrelerle eğitilmiş bir ağ ile birlikte, ilgili modülün genel bir görünümü gösterilmektedir.



Şekil 3.13. YSA Benzetim modülü genel görünümü

a) Grafik kontrolleri

Sayfanın üst kısmına yerleştirilmiş olan kullanıcının eğitim ve test adımlarını takip edebileceği üç farklı grafik kontrolü bulunmaktadır. "Hata / Öğrenme döngüsü" adı verilen ilk grafik seçili olan parametrelerle YSA'nın eğitilirken her bir öğrenme döngüsü sonucunda hata oranındaki değişmenin gösterilmesi için tasarlanmıştır. Böylelikle kullanıcı ağın eğitimi esnasında hatadaki değişimin karakteristik özelliklerine ulaşabilmekte ve ağın eğitimi için gerekli olan parametreleri değiştirerek grafik üzerinde sonuçların nasıl değiştiğini gözlemleyebilmektedir.

"Doğru / Yanlış " grafiği basit olarak test işlemi sonucunda başarılı bir şekilde tahmin edilen sınıfların, yanlış tahmin edilen sınıflara oranının izlenebilmesi için gerçekleştirilen sütun grafiklerinden oluşmaktadır. Aynı zamanda test kümesi içerisindeki toplam örnek sayısı da grafik içerisinde temsil edilmektedir. Grafikteki toplam örnek sayısını temsil eden sütun ile başarılı sınıflandırılan örnek sayılarının birbirine yakın çıkması başarılı bir eğitim ve test aşamasının,

ya da birbirine uzak çıkması ile başarısız bir eğitim ve test aşamasının gerçekleştiği anlamına gelebilmektedir.

Son grafik "Belirlilik / Duyarlılık / Doğruluk " adı verilen test işlemi sonucunda üretilen her bir çıkış sınıfı için ayrı ayrı performans değerlerinin gösterildiği bir grafiktir. Belirlilik mavi, duyarlılık sarı ve doğruluk turuncu olmak üzere grafikte her performans değeri farklı bir renge sahip sütunlar ile temsil edilmektedir. Böylelikle kullanıcının grafiği okuyabilmesinin ve anlayabilmesinin kolaylaştırılması sağlanmıştır. 0 ile 1 arasında değişen değerler alabilen bu performans değerleri, eğitilen ağın her bir sınıf için ayrı ayrı nasıl davranış sergilediğini göstermektedir. Geliştirilen sistemde eğitilen ağ, bir sınıf için ayırt edici olabilirken başka bir sınıf için ayırt edici olmayabilir ya da bir sınıf için başarılı sınıflandırma yaparken başka bir sınıf için yapamayabilir. Bu nokta da bu grafikten yararlanarak bu gibi durumların tespit ve analiz edilmesi sağlanabilir. Çizelge 3.1. ve Denklem 3.13, 3.14. ve 3.15. 'den yararlanarak belirlilik, duyarlılık ve doğruluk değerleri hesaplanmaktadır.

Çizelge 3.1. Performans analizi için gerekli hesaplamalar

		Gerçek Durum		
		Pozitif	Negatif	TOPLAM
Tahmin Edilen Durum	Pozitif	Doğru Pozitif(DP)	Yanlış Pozitif(YP)	DP+YP
	Negatif	Yanlış Negatif(YN)	Doğru Negatif(DN)	YN+DN
	TOPLAM	DP+YN	YP+DN	DP+YP+ YN+DN

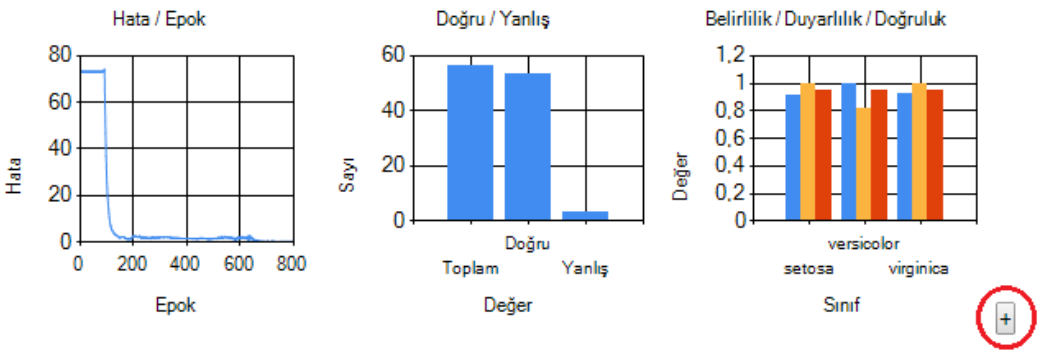
$$\text{Belirlilik} = P(B) = DN / (DN+YP) \quad (3.13)$$

$$\text{Duyarlılık} = P(D) = DP/(DP+YN) \quad (3.14)$$

$$\text{Doğruluk} = P(DO) = (DP+DN)/N \quad (3.15)$$

Performans analizi oluşturulurken örnek pozitif ise Doğru Pozitif (DP), örnek pozitif ancak negatif bulunmuşsa Yanlış Negatif (YN), örnek negatif ve negatif sınıflandırılmışsa Doğru Negatif (DN), örnek negatif ancak pozitif sınıflandırılmış ise Yanlış Pozitif (YP) olarak temsil edilmektedir.

Tamamı çalışma esnasında dinamik olarak oluşturulan grafikler, hemen hemen her popüler web tarayıcısının desteklediği ".png" resim formatında üretilmektedir. Kullanıcı istediği takdirde bu grafikleri indirebilmekte, hatta sistemde daha sonra göz gezdirmek üzere Şekil 3.14.'de de işaretlenmiş olan grafiğin hemen sağ tarafında ki "+" ile isimlendirilmiş "Buton" kontrolüne tıklayarak ağın detaylarının da içinde bulunduğu bir formatta saklayabilmektedir.

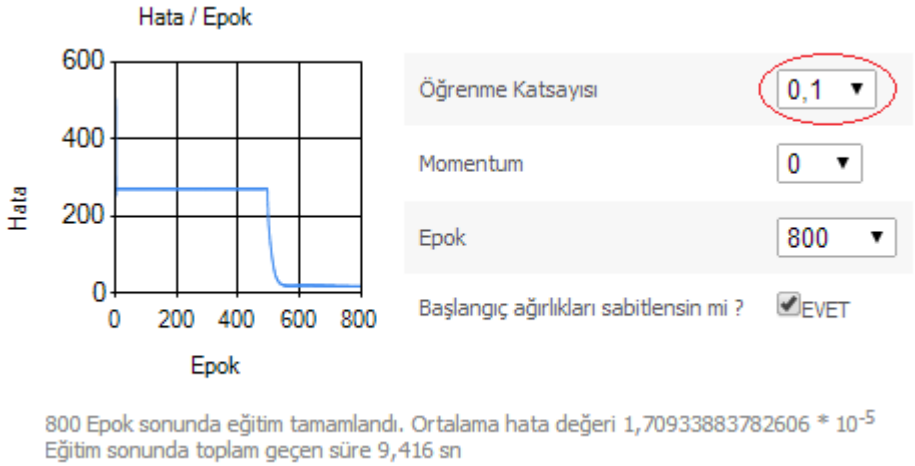


Şekil 3.14. Benzetim modülündeki grafikler ve kayıt için kullanılan buton

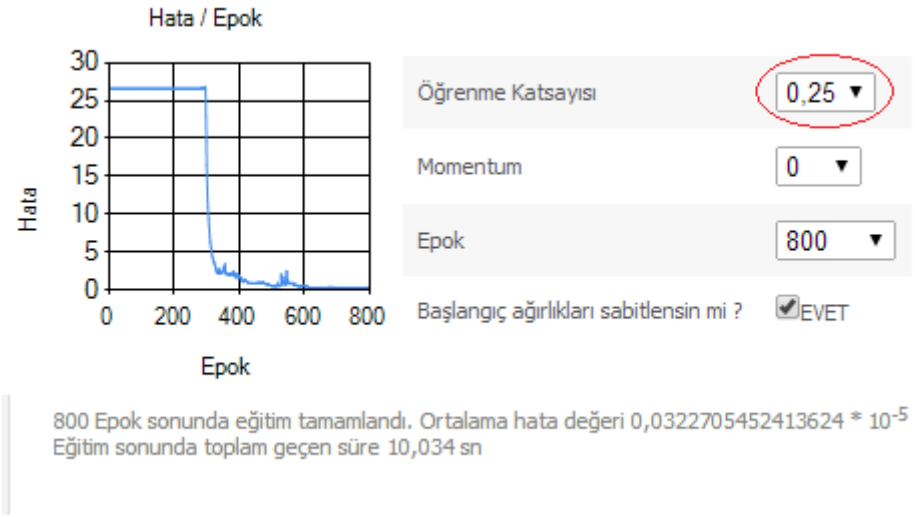
b) Eğitim adımları

Eğitim işlemi bir YSA'nın ağırlıklarının güncelleştirilmesi ile gerçekleşen bir olaydır, ancak burada ki esas problem bu ağırlıkların güncellenirken ilgili parametre değerlerinin etkilerinin önceden kestirilememesidir. YSA'nın dinamik yapısından dolayı bu ağırlıklarının ağa olan etkisi bir eğitimden diğerine de değişiklik gösterebilmektedir. YSA eğitimi verilirken, bu konuyu geleneksel yöntemlerle öğrencilere aktarmak hiç de kolay değildir. Ancak geliştirilen sistemi kullanarak kullanıcı, kaynak kodu değiştirmek yerine

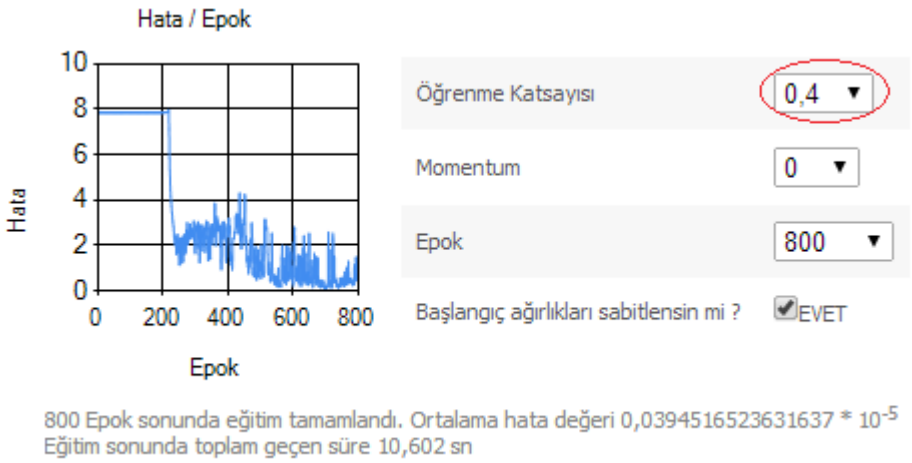
öğrenme katsayısı, devinirlik ve öğrenme döngüsü değerlerini modül içerisindeki "combobox" kontrollerini kullanarak istediği gibi değiştirebilmekte ve sayfanın yukarısındaki grafiklerden ağın eğitimine olan etkisini doğrudan izleyebilmektedir. Şekil 3.15., Şekil 3.16., Şekil 3.17. ve Şekil 3.18.'da "Iris" veri kümesi ile gerçekleştirilen ağ eğitim denemelerinde öğrenme katsayısının ağın eğitimine olan etkisi direkt gözlemlenebilmektedir. Denemelerde her öğrenme katsayısı için geliştirilen modül içerisindeki başlangıç ağırlıklarını sabitle seçeneği işaretlenerek aynı başlangıç ağırlıkları kullanılmıştır. Böylece farklı öğrenme katsayılarının karşılaştırmasında başka kriterlerin etkileri devre dışı bırakılmıştır.



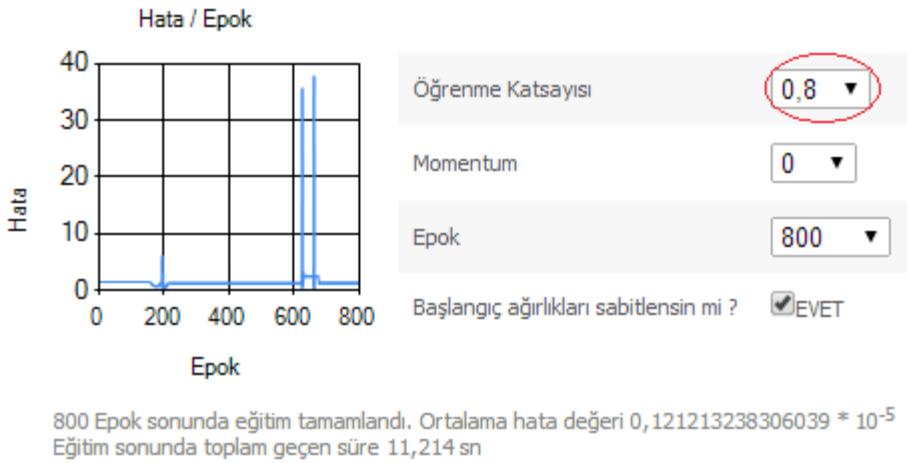
Şekil 3.15. Öğrenme katsayısı 0,1 iken Hata / Öğrenme döngüsü grafiği



Şekil 3.16. Öğrenme katsayısı 0,25 iken Hata / Öğrenme döngüsü grafiği



Şekil 3.17. Öğrenme katsayısı 0,4 iken Hata / Öğrenme döngüsü grafiği



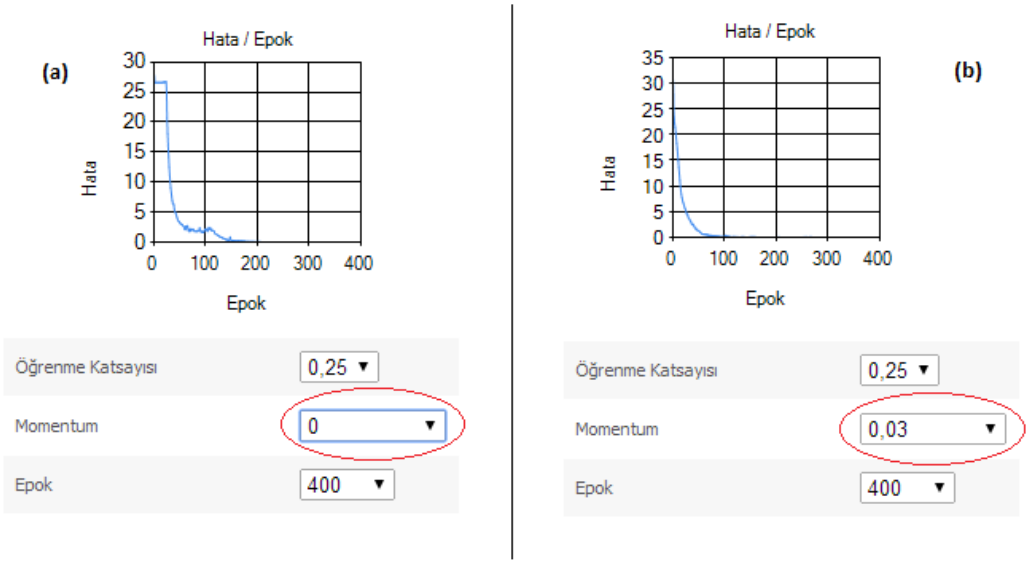
Şekil 3.18. Öğrenme katsayısı 0,8 iken Hata / Öğrenme döngüsü grafiği

Şekil 3.15.'de ki grafik incelendiğinde 0,1 olan öğrenme katsayısının öğrenme sürecini yüksek bir hata miktarından başlattığı, bir süre sabit kaldıktan sonra tam olarak istenilen ağı kararlı hale getirebilecek hata seviyelerine getiremeden sabitlendiği gözlemlenmektedir. Buna karşın 0,25 öğrenme katsayısı ile çalıştırılan ağın Şekil 3.16'da gösterilen grafikte, öğrenmeye bağlı olarak hatanın daha düşük bir değerle başladığı ve keskin bir ivme ile beklenen değerlere ulaştığı çıkarımı yapılabilmektedir. Şekil 3.17.'de ki grafikte ise düşük miktardaki hata seviyelerine rağmen gerçekleşen dalgalanmalar Şekil 3.18.'de daha belirgin bir şekilde fark edilebilmektedir. Sonuç olarak elde edilen hata ve süre değerlerinin gösterildiği Çizelge 3.2. incelendiğinde, farklı öğrenme katsayıları kullanarak ideal hata seviyelerinin sağlanabilmesi için birçok deneme yanılma yapılması gerekebilir. "Iris" veri kümesi ilen yapılan YSA eğitimi denemesinde 0,25 öğrenme katsayısı başarılı sonuçlar sağlamıştır. "Iris" veri kümesi çok fazla sayıda örnek içermediği için geçen süre çok az çıkmaktadır, ancak farklı öğrenme katsayılarının geçen süreye etkisi bariz bir şekilde gözlenmektedir. Bu gözlem kullanıcılara büyük eğitim kümeleri ile çalışırken de büyük bir avantaj sağlayacaktır. Çünkü ağı eğitmeye çalışırken gerçekleşen hata miktarı en önemli kriter değildir, toplam süreninde aynı derecede önem kazanabildiği durumlar olabilir. Kullanıcı YSA benzetim modülünü kullanarak çözmek istediği problem için en uygun öğrenme katsayısını bulabilmektedir.

Çizelge 3.2. Farklı öğrenme katsayıları sonrasında elde edilen hata ve toplam süre değerleri

Öğrenme Katsayısı	Toplam Hata Oranı (MSE) $*10^{-5}$	**Toplam Süre (sn)**	**Öğrenme döngüsü Sayısı**
0,1	1,709338	9,416	800
0,25	0,032270	10,034	800
0,4	0,039451	10,602	800
0,8	0,121213	11,214	800

Öğrenme katsayısı ile beraber ağın eğitimini etkileyen bir diğer parametre de devinirlik değeridir. Kullanıcılara hiç kaynak koda müdahale etmeden ağın eğitim parametrelerinin değiştirebilmesi imkânının verilmesi, devinirliğin kavranmasında da önemli rol oynamaktadır. Böylece kullanıcılar, Şekil 3.19.'de sonuçları verilen "Iris" veri kümesi ile denemesi yapılan farklı devinirlik değerlerine yakın sonuçlar bularak devinirliğin tam olarak işlevini kavrayabilecektir.

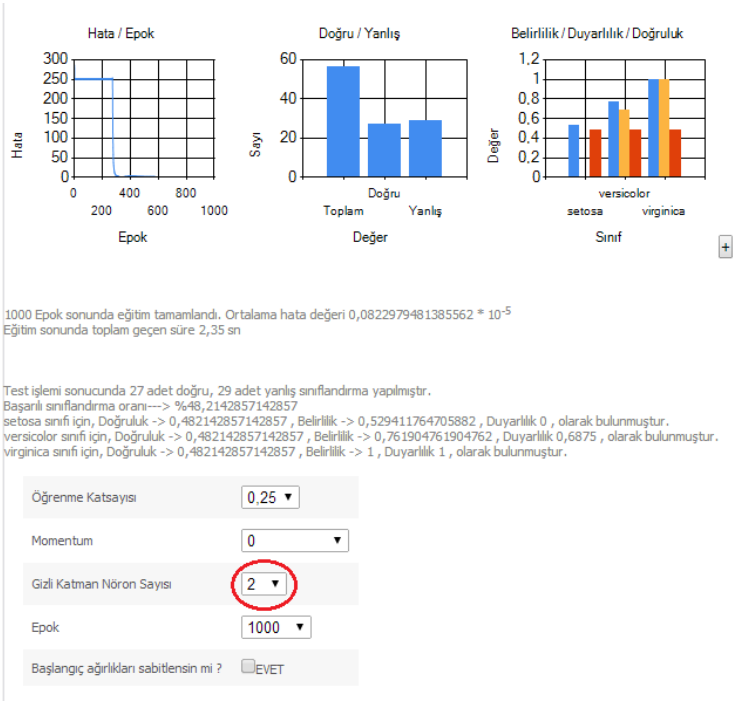


Şekil 3.19. Farklı devinirlik değerlerinin ağın eğitimi üzerindeki etkisi

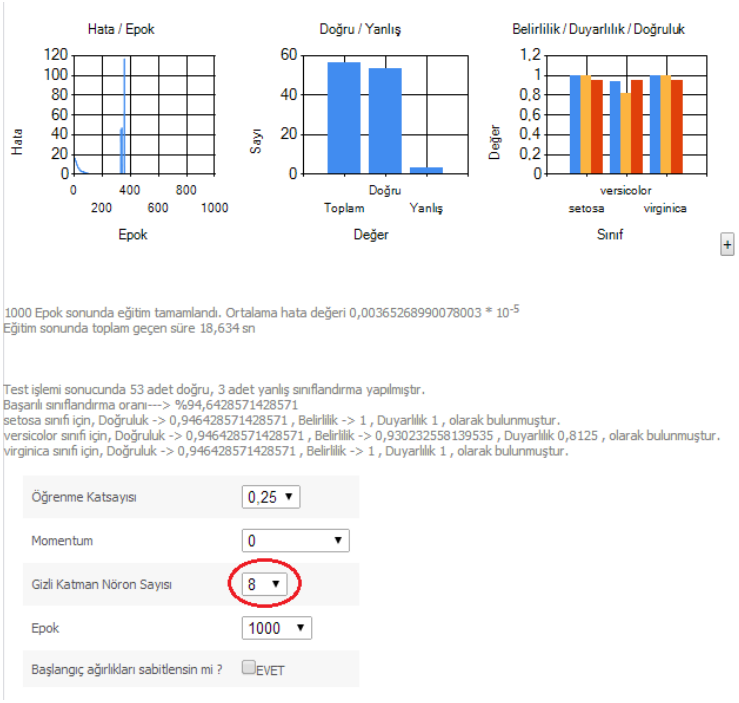
Şekil 3.19. (a)'da ve Şekil 3.19. (b)'de görüldüğü üzere devinirlik değerindeki çok az bir değişim dahi ağın eğitim sonucu için büyük önem taşımaktadır. Ancak

bu meydana getirdiği farklılığın öğrenciye aktarılması geleneksel yöntemlerle anlaşılması güç bir hale gelmektedir. Bu noktada geliştirilen sistem bu açığı gidererek, öğrenciye kendi devinirlik parametrelerini deneme imkânı vererek tecrübeye ve keşfetmeye dayalı öğrenmeyi kullanarak etkili ve kalıcı öğrenme sağlamaktadır.

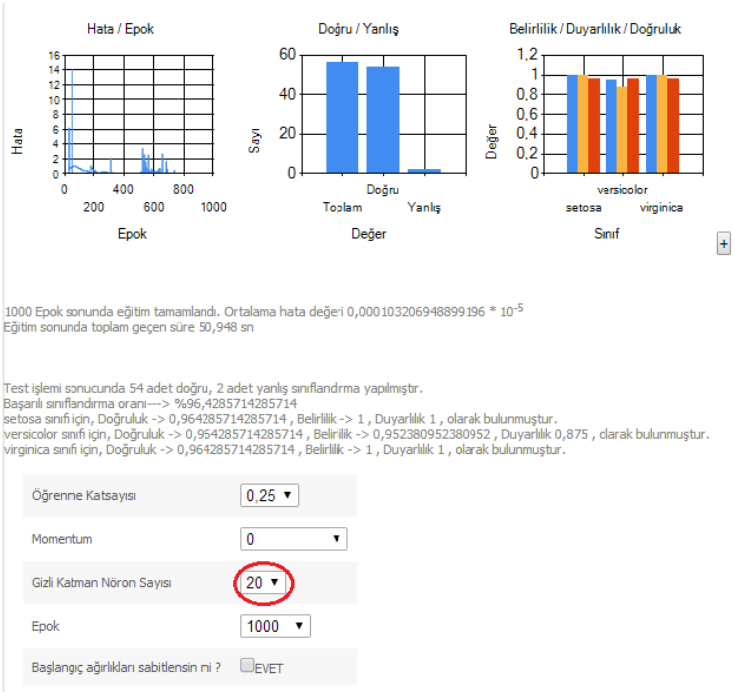
Yapay sinir ağının yapısında bulunan gizli katmandaki, diğer bir isimle ara katmandaki sinir sayısı ağın eğitim sürecini etkileyen en önemli faktörlerden biridir. Bu katmandaki sinir sayısının artırılması belli bir noktaya kadar sistemin doğruluğunu artırmaktadır. Bunun ile doğru orantılı olarak eklenen her bir sinir fazladan bir sürü hesap işlemi getirdiği için sistem performansını olumsuz yönde etkilemektedir. Şekil 3.20'de gizli katmanda 2 adet sinir kullanılması tercih edilmiştir ve ağın toplam hatası yüksek bir değer çıkmıştır. Eğitilen ağın test edilmesi sonucunda ise yaklaşık %48,21 değerinde başarısız kabul edilebilecek bir doğru sınıflandırma oranı ölçülmüştür. Aynı oranda performans değerleri de çok düşük seviyelerde gözlemlenmiştir. Ancak eğitim işlemi 2,35 saniye gibi çok hızlı bir sürede gerçekleşmiştir. Şekil 3.21.'de gizli katman sayısı 8 seçildiğinde ve Şekil 3.22.'de gizli katman sayısı 20 seçildiğinde gerçekleşen çıkışlar gözükmektedir. Gizli katmandaki sinir sayısı arasındaki fark çok olmasına rağmen sistemdeki başarı oranı çok düşük bir değer artış göstermiştir. Bu örnek için ara katmanda 8 sinir kullanılması, 20 sinirli eğitime göre daha hızlı olduğu için en uygun değer gibi gözükse de 20 sinir ile yapılan eğitim daha hassas sonuçlar vermiştir. Sonuçlar, ayrıntılı bir karşılaştırma yapılabilmesi için Çizelge 3.3.'de verilmiştir.



Şekil 3.20. Gizli katmada 2 sinir kullanımı



Şekil 3.21. Gizli Katmada 8 sinir kullanımı



Şekil 3.22. Gizli Katmada 20 sinir kullanımı

Çizelge 3.3 Gizli katmandaki farklı sinir sayılarının eğitim ve test süreçlerine etkisi

Gizli Katmandaki Sinir Sayısı	Hata Oranı (MSE) $*10^{-5}$	Başarılı Sınıflandırma Oranı	Toplam Süre (sn)
2	0,0822979	% 48,21	2,35
8	0,0036526	% 94,64	18,63
20	0,000103	% 96,42	50,94

Bir YSA'nın eğitim sürecini bitirme kararı yaygın olarak kullanılan iki yönteme göre verilir. Bu yöntemlerden biri ağın istenilen hata oranını yakalayabilmesi diğeri ise istenilen öğrenme döngüsü sonucunda hata ne olursa olsun eğitimin tamamlanmasıdır. Ağın ilgili hatayı yakalayabilmesi her zaman mümkün olmadığı için kontrolü kullanıcıya vermek adına biz ikinci yaklaşım tercih edilmiştir. Böylelikle kullanıcı örnek veriler için yapılan eğitim sürecini istediği kadar tekrar edip süreci şekillendirebilmektedir. Diğer bütün parametreler aynı kalarak öğrenme döngüsü değeri artırıldığında, ağ henüz toplam öğrenme döngüsü sayısına ulaşmadan istenilen hata değerlerine gelmiş olsa bile sistem ağ eğitimine devam edebilir. Bu devam sürecinde aslında eğitilmiş olan ağ

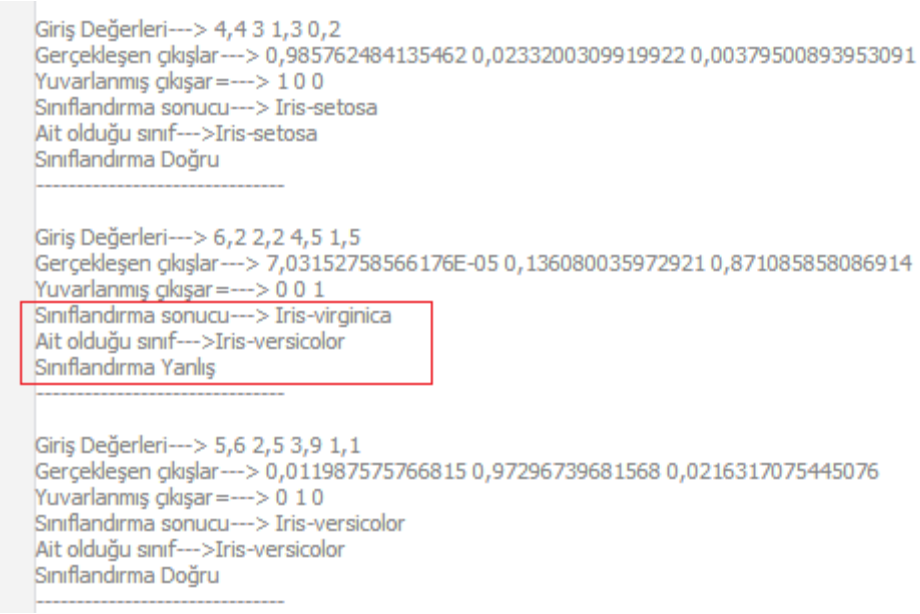
tekrar salınım göstererek istenemeyen ağırlık değerleri kazanabilir. Grafiksel olarak bu süreci takip etmek kullanıcılara eğitimin tam olarak hangi öğrenme döngüsünde gerçekleştiğini bulabilmeleri için imkân sağlamaktadır. Şekil 3.23.'de istenilen hata değerlerinin yakalanmış olmasına rağmen eğitimin tamamlanmamasının yani öğrenme döngüsü sayısının çok fazla tutularak ağın eğitilmeye çalışıldığın da meydana gelen Hata / Öğrenme döngüsü grafiğinde ki dalgalanmalar görülmektedir.

Şekil 3.23. Ağ eğitiminin 5000 öğrenme döngüsü boyunca çalıştırılması ile meydana gelen Hata / Öğrenme döngüsü grafiği

c) Test adımları

Geliştirilen sistemde kullanıcı, Test butonunu kullanarak daha önce eğitmiş olduğu ağı, istediği veri kümesi üzerinde test edebilmektedir. Test sonucunda kullanıcıya, daha önce bahsedilen grafikler çıktı olarak verildiği gibi her bir örneğin tek tek değerlendirme sonucu, bu sonuç doğrultusunda yapılan tahminin doğruluğu gibi bir sürü değer çıktı olarak verilmektedir. Kullanıcı bu bulgulara bakarak eğitilen ağın verimliliği, kararlılığı gibi birçok parametreyi değerlendirebilmektedir. Şekil 3.24.'de gerçekleştirilen bir test işlemi sonucunda "Iris" veri kümesi içerisinden sadece 3 örneğin değerlendirilmesinin sonucu verilmiştir. Şekil 3.24.'de işaretlenmiş olan ortadaki örneğin aslında "Versicolor" sınıfına ait olduğu halde sistem tarafından "Virginica" olarak

sınıflandırıldığı görülmektedir, bununla beraber diğer iki örnekte başarılı bir sınıflandırma yapıldığı gözükmektedir. Bu sonuçların yardımı ile kullanıcılar ağ hangi örnek için nasıl bir davranış sergilediğini detaylı bir şekilde inceleyebilmektedir.



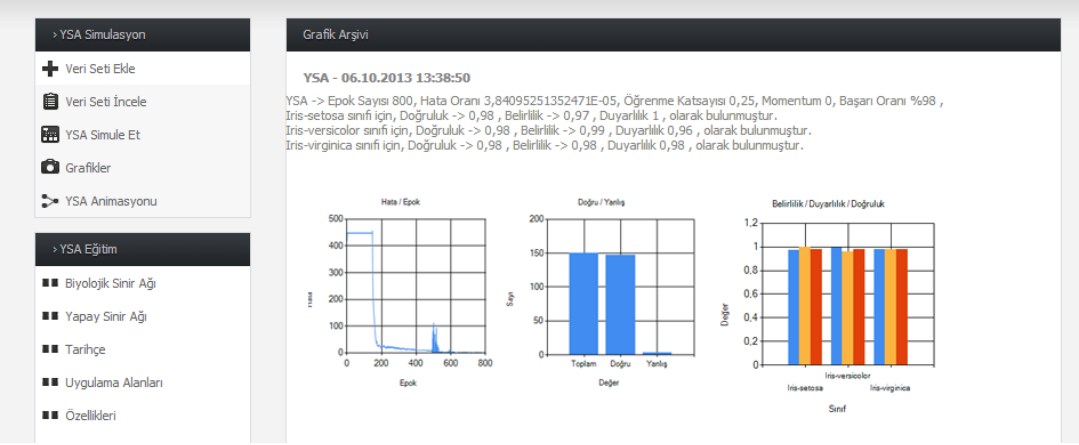
Şekil 3.24. Eğitilmiş bir YSA ile "Iris" veri kümesindeki 3 örnek için test işlemlerinin sonucu

Çıktılar içerisinde verilen giriş değeri, kullanıcı tarafından seçilen veri kümesi içerisindeki bir örneği temsil etmektedir. Bu örnek kendi içerisinde birden fazla parametre taşımaktadır. Gerçekleşen çıkışlar ise eğitimi yapılmış ağın giriş değerlerini çıkışa doğru yayarak elde etmiş olduğu sonuçlarıdır. Yuvarlanmış çıkışlar ise, gerçekleşen çıkışların en yakın oldukları tam sayıya yuvarlanması ile elde edilir, böylece en yakın oldukları sınıf hakkında karar verilebilmesi daha kolay hale gelmektedir. Yuvarlanmış çıkışlarda üç adet ikili sistemde sonuç vardır. Bunun hangi sınıfa ait olduğu hangi sıradaki çıkışın bir olduğu ile alakalıdır. Örneğin yuvarlanmış sonuçlar 1 0 0 şeklinde ise 0 indisine sahip "Setosa" sınıfına ait olduğu kararı verilmektedir. 0 1 0 yuvarlanmış çıkışları elde edilirse "Virginica", son olarak 0 0 1 sonuçları elde edilmiş ise 2 indisine sahip "Versicolor" olarak sınıflandırılma yapılmaktadır. Sınıflandırma sonucu ve ilgili örneğin gerçekte ait olduğu sınıf karşılaştırılarak doğruluk test edilmiş olur.

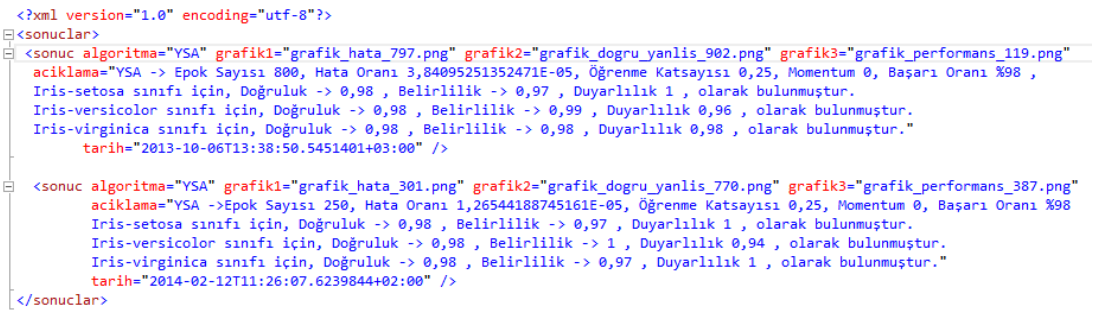
Kullanıcıya bütün örneklerden elde edilen bilgilerin değerlendirmesi ile oluşan, her sınıf için ayrı ayrı hesaplanmış performans analizi sonuçları da çıktı olarak gösterilmektedir. Bu sonuçlar aynı zamanda daha önce bahsedilen Belirlilik / Duyarlılık / Doğruluk ve Doğru / Yanlış grafiğinde gösterilmektedir.

3.3.2.4 Grafikler modülü

Kullanıcı geliştirilen sistemi kullanarak, kendi oluşturduğu veya hazır olarak temin ettiği birçok veri kümesini eğitip test edebilmektedir. Eğitim ve test işlemlerinin sonuçlarının kaydedilebilmesi ve daha sonra kullanıcının buna ulaşabilmesi gerekmektedir. Bu problemi çözmek için XML dosya formatı kullanılmıştır. Tüm grafik ve eğitim için kullanılan parametrelerin anlamlı bir şekilde saklanması ve istenildiği zaman tekrardan gösterilmesi Grafikler modülünde Şekil 3.25.'deki gibi sağlanmıştır ve Şekil 3.26.'da bu bilgilerin XML dosyası içerisinde nasıl tutulduğu gösterilmiştir. Böylelikle bu modül içerisinde, ağ eğitilirken kullanılan öğrenme döngüsü sayısı, öğrenme katsayısı, devinirlik, test sonucunda ki nihai hata oranı, başarı yüzdesi ve performans değerleri ilgili grafiklerle beraber kayıt altına alınıp tekrar kullanıcıya sunulabilmektedir.



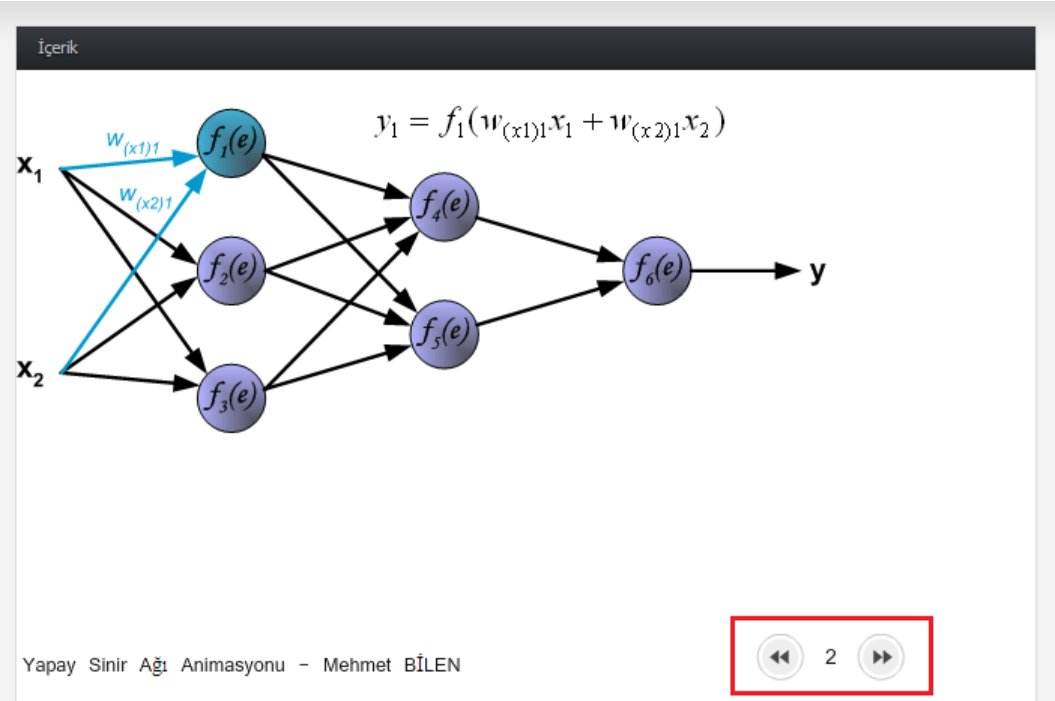
Şekil 3.25. Grafikler modülü genel görünümü



Şekil 3.26. Bir eğitim ve test sonucunun XML dosyasına kayıt edilmesi

3.3.2.5 YSA canlandırma modülü

Canlandırmalar, insanların karmaşık şeyleri anlayabilmesi için geliştirilen kullanışlı araçlardır. Etkili ve kalıcı öğrenmenin sağlanabilmesi ve keşfedici öğrenmenin desteklenmesi için canlandırma gibi araçlara mutlaka ihtiyaç duyulmaktadır. Bu canlandırmaların geliştirilerek karmaşık bir yapıya sahip olan YSA modellerinde eğitim süreçlerinin anlaşılması için kullanılması öğrenciler için büyük kolaylıklar sağlamıştır. Kullanıcı canlandırmayı, Şekil 27.'de işaretlenmiş kontroller sayesinde ileri geri hareket ettirebilmektedir. Böylelikle kullanıcı bir ağın eğitim işleminin baştan sona canlandırmaya dönüştürülmüş halini, adım adım görebilmekte, canlandırma üzerinde verilen her adıma ait hesaplamaları inceleyebilmekte ve YSA'nın çalışma mekanizmasını kolayca kavrayabilmektedir.



Şekil 3.27. Canlandırma modülünden bir kare

4. ARAŞTIRMA BULGULARI

Geliştirilen web tabanlı YSA eğitimi yazılımının gereksinimleri hangi ölçüde karşıladığının tespit edilmesi için yapılan ölçümlerin ve anketlerin sonuçları sırası ile kod ölçümleri, performans analizi ve anket uygulaması başlıkları altında paylaşılmıştır.

4.1. Kod Ölçümleri

Yazılım geliştirirken kullanılan Visual Studio 2012 uygulaması kendi içerisinde kod ölçümlerinin yapılabilmesi için bir araç bulundurmaktadır. Bu araç sayesinde Kod ölçümlerini hesapla (Generate Code Metrics) komutu verilerek geliştirilen uygulamanın sınıf bağıntıları, kalıtım derinliği, döngüsel karmaşıklık, kod bakım kolaylığı seviyesi ve kod satır sayısı değerleri hesaplanmıştır. Hesaplanan değerler Çizelge 4.1. 'de gösterilmektedir.

Çizelge 4.1. Geliştirilen uygulamanın kod ölçüm sonuçları

Sınıf bağıntıları	Kalıtım Derinliği	Döngüsel Karmaşıklık	Kod Bakım Kolaylığı Seviyesi	Kod Satır Sayısı
39	4	19	90	658

Sınıf bağıntıları; bir sınıfın bağlı olduğu sınıf sayılarını gösterir. Bu değerin yüksek olması durumu tanımlanan sınıfların bir birleri ile bağıntılarının fazla olması anlamına gelmektedir ki bu da kodun tekrar kullanılabilirliğini düşürmekte ve kod bakımını zorlaştırmaktadır. Bu yüzden uygulama geliştirme esnasında sınıf bağıntıları değerinin düşük tutulmasına önem gösterilmiştir. Kalıtım derinliği; türetilen sınıfların ilk tanımlanan sınıfa olan uzaklığını gösteren bir değerdir. Kalıtım derinliği için 4 değeri uygun bir değer olup daha yüksek çıkması durumunda metotların, sınıfların, alanların nerede tanımlandığını takip etmek zorlaşabilmektedir. Buda kod bakım kolaylığını düşürmektedir. Döngüsel karmaşıklık; birbirinden farklı kod parçalarının sayısıdır. Çizelge 4.2. 'den yararlanarak 19 değerinin orta riskli bir değer olduğu sonucuna varılmıştır. Sınıf üyeleri veya tipler seviyesinde kod bakımının

kolaylığına gösteren bu değişken 0-100 arasında bir değer alır. Çizelge 4.3'ten yararlanılarak 90 değeri ile yüksek bakım kolaylığına sahip bir kod elde edildiği sonucuna varılmıştır. Kod satır sayısı; geliştirilen uygulama içerisindeki işletilebilen kod sayısını göstermektedir.

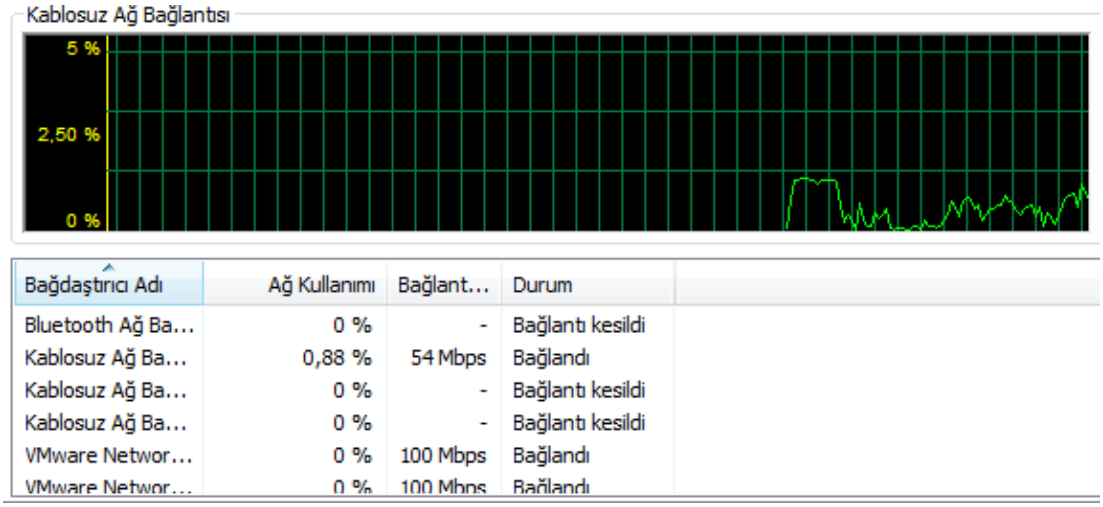
Çizelge 4.2. Döngüsel karmaşıklık değerleri ve riskleri (Karagedik, 2010)

Döngüsel Karmaşıklık	Yöntem	Risk
1-4	Basit bir yöntem	Düşük
5-10	Güzel yapılandırılmış ve kararlı bir yöntem	Düşük
11-20	Biraz daha karmaşık bir yöntem	Orta
21-50	Karmaşık yöntem uyarısı	Yüksek
>50	Kararsız ve hatalı bir yöntem	Çok Yüksek

Çizelge 4.3. Kod bakım kolaylığı seviyeleri (Karagedik, 2010)

Seviye	Aralık
Yüksek	20-100
Orta	10-19
Düşük	0-9

Geliştirilen uygulamanın windows işletim sistemi bileşenlerinden görev yöneticisi kullanılarak ölçülen ağ kullanım miktarı Şekil 4.1'de gösterilmektedir. Hız gereksinimi ağ trafiğinin en yoğun olduğu durumda 54 Mbps aktarım hızının sadece %0,88'ini kullanarak yaklaşık olarak 0,47 Mbps değerine ulaşmıştır. Bu değer ise Türkiye de ki ev kullanıcıların sahip olduğu 2 Mbps aktarım hızı değerinden çok düşük bir değerdir.



Şekil 4.1. Uygulama ağ trafik kullanımı

4.2. Performans Analizi

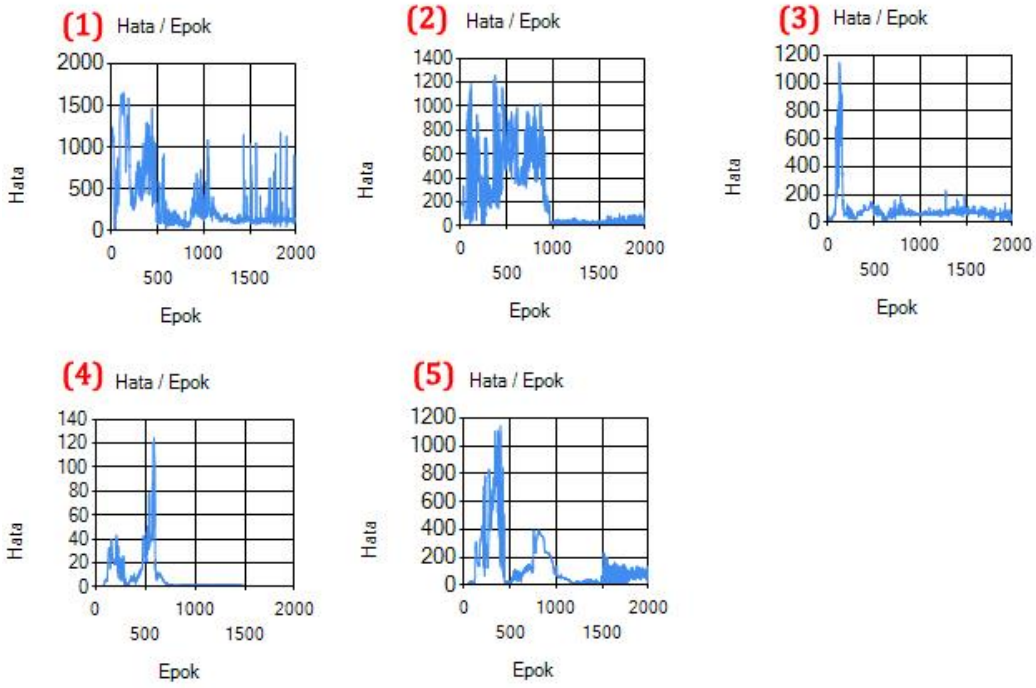
Geliştirilen sistem ile eğitilen ağların performans değerlerini değerlendirmek için "BalanceandScale" veri kümesi tercih edilmiştir (Bache & Lichman, 2013). Bu veri kümesi psikolojik tecrübe sonuçlarının modellenmesi ile oluşturulmuştur. Veri kümesi içerisindeki her bir örnek "Balanced (B)", "Right (R)" ve "Left (L)" sınıflarından birine aittir. Her bir örnek sol ağırlık, sol uzaklık, sağ ağırlık, sağ uzaklık değerleri ve sınıf parametresi ile birlikte toplam 5 adet değer barındırmaktadır. Veri kümesi içerisinde toplam 614 örneğin 47 tanesi "B" sınıfına, 283 tanesi "R" sınıfına, kalan 284 adet örnek ise "L" sınıfına aittir.

Seçilen veri kümesi içerisinden eğitim işlemleri için kullanmak üzere toplam örnek sayısının %80'i alınmıştır. Kalan %20 'lik kısım ise test için kullanılmıştır. Örneklerin seçilirken örneklerin ait oldukları sınıflar dikkate alınarak oran korunmuştur. Elde edilen eğitim veri kümesi içerisindeki sınıfların dağılımı Çizelge 4.4.'de listelenmektedir.

Çizelge 4.4. "BalanceandScale" veri kümesi içindeki eğitim ve test işlemleri için kullanılan örneklerin dağılımı

Sınıf	Veri Kümesi	Eğitim Veri Kümesi	Test Veri Kümesi
B	47	37	10
L	284	228	56
R	283	227	56

Eğitim veri kümesi ile yapılan eğitim ve test işlemlerinin sonucunda bulunan değerler Çizelge 4.5. ve Çizelge 4.6.'da listelenmiştir ve ilgili ağların eğitimi esnasında oluşturulan Hata / Öğrenme döngüsü grafikleri Şekil 4.2.'de gösterilmiştir.



Şekil 4.2 Eğitim işlemleri sonucunda oluşturulan Hata / Öğrenme döngüsü grafikleri

Çizelge 4.5 Eğitim ve test örnekleri için performans sonuçları

Ağ No	Doğruluk Tüm Sınıflar	Belirlilik			Duyarlılık		
		B	L	R	B	L	R
1	0,9262	0,9724	0,9672	0,9384	0,7	0,9642	0,9285
2	0,9590	0,9908	0,9841	0,9552	0,9000	0,9821	0,9464
3	0,9836	0,9910	1	0,9848	0,9000	1	0,9821
4	0,9918	1	0,9850	1	1	0,9821	1
5	0,9508	0,9649	0,9687	1	0,6	0,9642	1

Çizelge 4.6.Eğitim ve test örnekleri için diğer sonuçlar

Ağ No	Öğrenme Döngüsü	Öğrenme Katsayısı	Gizli Katman Sinir Sayısı	Doğru	Yanlış	Başarı	Ortalama Kare Hata MSE ($*10^{-5}$)	Toplam Süre (sn)
1	2000	0,30	3	113	9	%92,62	1,1783114	28,61
2	2000	0,30	5	117	5	%95,90	0,3276797	72,08
3	2000	0,30	10	120	2	%98,36	0,0285055	173,85
4	2000	0,30	20	121	1	%99,18	0,0203843	459,67
5	2000	0,20	20	116	6	%95,08	0,5657506	487,08

Yapılan eğitim ve test işlemleri sonucunda parametrelerdeki değişimlerin ağın eğitimine olan etkileri doğrudan gözlemlenmiştir. Gizli katmandaki sinir sayısının artması ile işlem kapasitesinin arttığı ve daha fazla işlem sayesinde belli bir noktaya kadar daha başarılı sonuçların elde edildiği, ters orantılı olarak eğitim esnasında harcanan toplam sürenin arttığı sonuçlarına ulaşılmıştır. Öğrenme katsayısının artışının ağın eğitimine ivme kazandırarak olumlu veya olumsuz yönde daha hızlı öğrenme gerçekleşmesini sağladığı da gözlemlenmiştir.

Çizelge 4.5. incelendiğinde "B" sınıfına ait performans değerlerinin işlemlerin çoğunda diğer sınıflardan daha düşük olduğu gözlemlenmiştir. "B" sınıfına ait örneklerin sayısı diğer sınıflara ait örneklerin sayısından daha az olduğu için eğitimi gerçekleştirilen ağların çoğu ilgili sınıf için yeterli tecrübe edinememiştir. Bunun sonucunda "B" sınıfına ait örneklerin bir kısmında başarısız sınıflandırma gözlemlenmiştir.

Ağ eğitimleri sonucunda 4. no'lu ağ en uygun değerlerin seçilmesi sonucunda yüksek bir başarılı sınıflandırma oranı ve performans değeri yakalayarak sadece 1 başarısız sınıflandırma gerçekleştirmiştir. Ağın daha önce görmediği örnekleri başarılı bir şekilde sınıflandırması ağın başarılı bir şekilde eğitildiğini kanıtlamıştır.

Her veri kümesi için en uygun öğrenme katsayısı, devinirlik değeri farklı olabilmekte, gizli katman sayısı değişebilmektedir. En uygun değerleri seçebilmek ise belli bir formül sonucunda değil tamamen deneme yanılma yolu ile sezgisel bir şekilde yapılmak zorundadır. Bu yüzden sistem kullanıcıların en uygun değerleri belirlemesine yardımcı olmaktadır.

Son olarak gizli katmandaki sinir sayısındaki artışın geçen süreyi büyük ölçüde uzattığı görülmüştür. Bununla birlikte öğrenme katsayısındaki değişikliğin toplam süre ile doğrudan bir ilişkisi olmadığı gözlemlenmiştir.

4.3. Anket Uygulaması

Geliştirilen uygulama Mehmet Akif Ersoy Üniversitesi Çavdır Meslek Yüksekokulunda iki dönem boyunca yapay zekâ derslerini alan öğrencilerin kullanıma sunulmuştur. Öğrencilere dönem sonlarında Çizelge 4.7'de gösterilen anket soruları uygulanmıştır ve öğrencilerden çok zayıf, zayıf, orta, iyi, çok iyi seçeneklerini işaretleyerek 1'den 5'e kadar puan vermeleri istenmiştir. 65 öğrencinin katıldığı anket istatistiksel olarak analiz edildiğinde ortalama 3,76 değeri ile öğrencilerin YSA konusunu açık bir şekilde YSA eğitim aracının yardımı ile anladıkları görülmektedir. Tüm anket sonucunda hesaplanan ortalama ve standart sapma değerleri Çizelge 4.8'de gösterilmektedir.

Çizelge 4.7 Anket soruları

Soru No	Anket Soruları
1	YSA eğitim aracını kullanarak istediğim bilgiye istediğim zaman ulaşabilirim.
2	YSA eğitim aracı öğrenme kaynaklarına ulaşırken bana zaman kazandırmaktadır.
3	YSA eğitim aracını sınavlara hazırlanmak için kullanırım.
4	YSA konusunu açık bir şekilde YSA eğitim aracının yardımı ile anladım
5	YSA eğitim aracı dersteki motivasyonumu artırdı.
6	YSA konusunu tekrardan değerlendirmek için YSA eğitim aracını kullanırım.
7	YSA eğitim aracı kolay ve kullanışlıdır.

Çizelge 4.8. Anket sonuçlarının istatistiksel analizi

Soru No	Ortalama	Standart Sapma
1	3,968750	0,951459
2	3,812500	0,826797
3	3,718750	0,819084
4	3,765625	0,914163
5	3,687500	0,899218
6	3,765625	0,896908
7	3,843750	0,887742

5. SONUÇ

Matematiksel doğası ve doğrusal olmayan karmaşık yapısından dolayı yapay sinir ağları eğitiminin geleneksel yöntemlerle yapılması, yapay sinir ağlarının anlaşılmasını zorlaştırmaktadır. Tez çalışmasında yapay sinir ağları eğitiminde kullanılabilecek, yapay sinir ağları ile ilgili her türlü eğitim, test, tasarım işlemlerinin benzetiminin yapılabildiği, sonuçların grafikler ile izlenebildiği, canlandırmalar ve içerikler ile desteklenmiş, internet üzerinden dünyanın herhangi bir yerinden erişilebilen, kullanımı kolay bir yazılım geliştirilmiştir. İnternet üzerinden yapay sinir ağlarına ilgi duyan herkesin ön bilgiye ihtiyaç duymadan yazılımı kullanabilecekleri bir ortam sağlanmıştır ve yapay sinir ağları eğitiminde keşfedici öğrenme desteklenmiştir.

Tez çalışmasında gerçek zamanlı olarak yapay sinir ağları eğitiminde kullanılan parametrelerin değiştirilebilmesi ve bu değişikliklerin grafiksel olarak gösterilmesi en uygun parametre seçiminin yapılmasını kolaylaşmıştır. Böylece yapılan ağ eğitimi ve test işlemlerinde 0,9918 doğruluk, 1 belirlilik ve 1 duyarlılık değerlerine ulaşılmıştır. Aynı zamanda parametrelerde yapılan değişikliğin ağın eğitimine etkisinin doğrudan gözlemlenebilmesi sağlanarak yapay sinir ağlarının çalışma mekanizmasının anlaşılması kolaylaştırılmıştır. Buna ek olarak yapay sinir ağlarının doğrusal olmayan karmaşık yapısının öğrenilmesi ve çalışma mekanizmasının anlaşılması grafiksel çıktıların yardımı ile takip edilmesi kolay bir hale getirilmiştir.

Yazılım, nesne yönelimli programlama dili olan C# ile Web 2.0 teknolojisi desteği ile geliştirilmesi, XML gibi popüler açık kaynak dosya sistemlerinin kullanılması ve Ajax nesnelerinin sayesinde diğer popüler yazılımlarla birlikte çalışabilen, gelecek çalışmalara alt yapı sağlayan, esnek, hızlı ve güçlü bir kod yapısı ile donatılmıştır. Yazılım üzerine kolayca ekleme yapılabilecek bir şekilde geliştirilmiştir. Yeni algoritmaların eklenerek, kolay bir şekilde güncelleme yapılabilecek bir yapıda tasarlanmıştır. Yazılımın ihtiyaç duyduğu bağlantı hızı optimize edilerek, Türkiye internet hizmet sağlayıcılarının sunmuş olduğu en düşük veri aktarım hızından daha düşük kaynak kullanması sağlanmıştır.

Tasarımdan bağımsız bir kodlama yapılarak mobil cihazlar gibi farklı platformlar için hazırlanacak uygulamalara, modüllerin hızlı bir şekilde aktarılabilmesi sağlanmıştır.

Geliştirilen yazılım iki dönem boyunca Mehmet Akif Ersoy Üniversitesi, Çavdır Meslek Yüksek Okulunda verilen yapay zekâ dersleri içerisinde kullanılmıştır. Dersi alan öğrencilere uygulanan anket sonucu değerlendirildiğinde, öğrencilerin büyük bir kısmı yapay sinir ağlarının temelleri ve çalışma prensibinin anlaşılması için bir eğitim yazılımının gerekli olduğunu ve hazırlanan yazılımın yapay sinir ağlarının temellerini ve çalışma prensibini etkili ve kolay bir şekilde öğrenmek için kullanışlı ve faydalı olduğunu belirtmiştir.

Yapılan çalışma literatürdeki benzer yazılımların avantajlı yönlerini birleştirerek tasarlanmıştır ve görsellik, anlaşıla bilirlik, işlevsellik bakımdan ilerleme sağlanmıştır. Geliştirilen yazılım var olan benzer yazılımların alternatifi ve destekleyicisidir.

Yapay sinir ağları eğitimini daha etkili ve verimli bir hale getirmek için geliştirilen uygulamanın sonraki sürümlerinde, farklı kullanıcı seviyeleri ile oturum açılabilmesi ve kullanıcıların öğrenme süreçlerinin takip edilebilmesi sağlanacaktır. Veri tabanı desteği eklenerek her bir kullanıcının elde ettiği sonuçların kaydının tutulması, bulguların değerlendirilmesi ve ödev takibinin yapılması için gerekli modüllerin eklenilmesi düşünülmektedir. Böylelikle yapay sinir ağları konusunda kullanıcıların deney yapabileceği, web tabanlı sistemlerin avantajlarını sonuna kadar kullanan, sanal bir laboratuvar yazılımının elde edilmesi planlanmaktadır.

KAYNAKÇA

Aktürk, F., 2013. Genetik Algoritma ile Büyükbaş Süt Hayvanlarında Süt Verimi Maksimizasyonu. Gazi Üniversitesi, Sosyal Bilimler Enstitüsü, Yüksek Lisans Tezi, 184s, Ankara.

Altıntaş, E., 2011. Yapay Sinir Ağları (Artifical Neural Networks). Erişim Tarihi 3.18.2012. http://www.yapayzeka.org/modules/wiwimod/index.php?page=ANN

Anonim, 2007. Metinlerin Sınıflandırılması. Erişim Tarihi 18.3.2012. http://www.turkceciler.com/metinlerin-siniflandirilmasi.html

Bache, K., Lichman, M., 2013. UCI Machine Learning Reposity Balance and Scale Data Set. Erişim Tarihi 09.01.2013. http://archive.ics.uci.edu/ml/datasets/Balance+Scale

Bayındır, R., Sesveren, Ö., 2008. YSA Tabanlı Sistemler için Görsel Bir Arayüz Tasarımı. Pamukkale Üniversitesi Mühendislik Fakültesi Mühendislik Bilimleri Dergisi, 14, 101-109.

Blanton, H., 1997. An Introduction to Neural Networks for Technicians, Engineers and Other non PhDs. 9-12 November, Proceedings of the 1997 Artificial Neural Networks in Engineering Conference. St. Louis. (CD-ROM).

Bozik, O., 2011. Ana Bileşenler Analizinde Dayalı Yapay Sinir Ağı Yaklaşımı ile Radarla Uçak Sınıflandırma. Hava Harp Okulu, Havacılık ve Uzay Enstitüsü, Yüksek Lisans Tezi, 134s, İstanbul.

Deperlioğlu, Ö., Köse, U., 2011. An Educational Tool For Artificial Neural Networks. Computers and Electrical Enginnering, 37, 392-402.

Elmas, Ç, 2003. Yapay Sinir Ağları Kuram, Mimari, Eğitim, Uygulama. Seçkin Yayıncılık, 190s, Ankara.

Frenzel, L. E., 1987. Understanding Expert Systems. Longman Higher Educations, 225p, London.

Haykin, S., 1999. Neural Networks: A Comprehensive Foundation. Prentica-Hall, 768p, New Jersey.

Holland, J. H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, 183p, Oxford.

Karagedik, Ö., 2010. Visual Studio 2008'de Kod Metrikleri ve Kod Analizi. Erişim Tarihi 29.05.2014. http://univera-ng.blogspot.com.tr/2010/03/visual-studio-2008de-kod-metrikleri-ve.html

Kastner, J. K., Hong, S. J., 1984. A Review Of Expert Systems. Europan Journal of Operational Research, 18(3), 285-292.

Keleşoğlu, Ö., Fırat, A. (2006). Tuğla Duvardaki ve Tesisattaki Isı Kaybının Yapay Sinir Ağı ile Belirlenmesi. Fırat Üniversitesi Fen ve Mühendislik Bilimleri Dergisi ,18(1), 133-141.

Kilitçi, A., Özdemir, S., Alp, S., 2011. Veri Tabanı Yönetim Sistemleri. Türkmen Kitabevi, 280s, İstanbul.

Kowali, K., Janusz, S., 1986. Knowledge Based Problem Solving. Prectice-Hall, 360p, New Jersey.

Lee, H., Park, J. C., 2001. Translating Natural Language Queries into Formal Language Queries with Combinatory Categorial Grammer. International Conference on Computer Proccessing of Oriental Languages, 14-16 May, Seoul , 41-46.

Manic, M., Wilamowski, B., Malinowski, A., 2002. Internet Based Neural Network Online Simulation Tool. Industrial Electronics Society, IEEE 2002 28th Annual Conference, 5-8 November, Sevilla, 2870-2874.

Nabiyev, V., 2003. Yapay Zeka. Seçkin Yayıncılık, 752s, Ankara.

Nissan, E., 1989. Artificial Intelligence in Higher Education AI for Education. Present Trends, as Sources for AI Concept of Knowledge-Presentation. Oosthoek, H.(Ed.), In Higher Education and New Technologies (67- 68). Pergamon, 558p, Mishawaka.

Öztemel, E. 2012. Yapay Sinir Ağları. Papatya Yayıncılık, 232s, İstanbul.

Rosello, E., Perez-Schofield, J. B., Dacosta, J., Perez-Cota, M., 2003. Neuro-Lab, A Highly Reusable Software-Based Enviroment to Teach Artifical Neural Networks. Computer Applications in Engineering Education, 11(2), 93-102.

Russel, S., Norvig, P., 2003. Artificial Intelligence, A Modern Approach. Prentice-Hall, 1109p, New Jersey.

Sağıroğlu, Ş., Beşdok, E.,Erler, M., 2003. Mühendislikte Yapay Zeka Uygulamaları - I Yapay Sinir Ağları. Ufuk Yayınevi, 426s, Kayseri.

Şen, Z. 2001. Bulanık Mantık ve Modelleme İlkeleri. Bilge Kültür Sanat, 172s, İstanbul.

Tektaş, A., Karataş, A., 2004. Yapay Sinir Ağları ve Finans Alanına Uygulanması, Hisse Senedi Fiyat Tahminlemesi. Atatürk Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 18 (3-4), 337-349.

Tosyalı, H., 2008. Uzman Sistemlerin Yasal Düzenlemelere Uygulanarak Akıllı Veri Tabanlarının Geliştirilmesi. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 70s, İstanbul.

Turing, A. M. 1950. Computing Machinery and Intelligence. Mind, 236 (59), 433-460.

Uğur, A., Kınacı, A., 2010. A web-based tool for teaching neural network concepts. Computers Applications in Engineering Education, 18 (3), 449-457.

Venayagamoorty, G. K., 2004. Teaching Neural Network Concepts and Their Learning Techniques. Proceedings of the 2004 American Society for Engineering Education Midwest Section Conference, 20 September-1 October. Pittsburg,(CD-ROM).

Williams, N., 1992. The Artifical Intelligence Applications to Learning Programme. Computers & Education (18), 101-107.

Yavuz, U., 1995. Uzman Sistemler ve Parametrik Hipotez Testleri Üzerine Bir Uygulama. Atatürk Üniversitesi, Sosyal Bilimler Enstitüsü, Doktora Tezi, 189s, Erzurum.

Yegnanarayana, B., 2006. Artificial Neural Networks. Prentice-Hall of India, 476p, New Delhi.

Zimmerman, H. J., 1987. Fuzzy Sets, Decisions Making and Expert Systems. Kluwer Academic Publishers, 335p, Boston.

ÖZGEÇMİŞ

Adı Soyadı :Mehmet BİLEN

Doğum Yeri ve Yılı : Uluborlu, 1986

Medeni Hali : Evli

Yabancı Dili : İngilizce

E-posta : mbilen@mehmetakif.edu.tr

Eğitim Durumu

Lise :Isparta, Anadolu Meslek Lisesi,

Lisans : SDÜ, Teknik Eğitim Fakültesi, Bilgisayar Sistemleri Öğr.

Mesleki Deneyim

Mehmet Akif Ersoy Üni., Çavdır Meslek Yüksek Okulu,
Öğretim Görevlisi

2012-...(halen)

Gümüşhane Üni., Kelkit Meslek Yüksek Okulu
Öğretim Görevlisi
2011-2012

Yayınları

Yiğit, T., Işık, A., H., Bilen, M., 2014. Web Based Educational Software for Artificial Neural Networks. International Conference on Education in Mathematics, Science and Technology, 16-18 May, Konya, 629-632.