T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI

YAPAY SİNİR AĞLARI İLE KREDİBİLİTE TESPİTİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Onur ŞANLI

Tez Danışmanı
Prof. Dr. Ali OKATAN

Ocak , 2008

T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Yönetim Bilişim Sistemleri Programı Yüksek Lisans öğrencisi Mehmet Tunç Onur Şanlı tarafından hazırlanan " **Yapay Sinir Ağları İle Kredibilite Tespiti** " adlı bu çalışma jürimizce Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi : 25.01.2008

(Jüri Üyesinin Ünvanı , Adı , Soyadı ve Kurumu) :	İmzası :
Jüri Üyesi: Prof.Dr.Ali OKATAN (Danışman-HÜ.Bil.Mühendisliği)	.....
Jüri Üyesi : Prof.Dr.Sami ERCAN (H.Ü.End.Mühendisliği)	.....
Jüri Üyesi : Yrd.Doç Dr.Yüksel BAL (H.Ü.Bil.Mühendisliği)	.....

Ç NDEK LER

BİRİNCİ BÖLÜM

ŞEKİLLER LİSTESİ

ÖNSÖZ

Bu tez çalışmamda yapay sinir ağlarını kullanarak bir bankanın kredibilitesi üzerinde durulmaya çalışılmıştır. Bu konunun oluşum aşaması, yapay sinir ağlarını çalıştığım bir alana yönlendirmek olmuştur. Kredibilitenin tespitine yönelik çalışmanın oluşturulması için ilk adım aldığım "Veri Analizi" dersinde kendi mesleğimle ilgili verileri analiz etmek olmuştur. Bu aşamadan sonra tezimin konusu oluşmaya başlamıştır. Daha sonra özel bir bankanın müşterilerine kredi verme verileriyle ilgili olarak Sayın Metin Şahin`e ait C# da hazırlanan yapay sinir ağları programında çalışmamı gerçekleştirdim.

Tezi hazırlama amacım; halen çalıştığım iş dalına ait olan işleri daha da kolaylaştırılmasına yardımcı olmak ve bu konuda yapılacak çalışmalara katıkıda bulunmaktır.

Tezimin düşünsel olarak hazırlanmasında emeği geçen ve gereken bilgi altyapısının oluşmasına yardımcı olan değerli hocam Sayın Prof. Dr. Ali OKATAN`a, tezimin matematiksel olarak ifadesini sağlayan C# oratamında yapay sinir ağları programını hazırlayan ve tezimin oluşumunda büyük katkısı olan Sayın Metin ŞAHİN`e ve tezimin oluşması için gereken verileri temin eden Sayın Engin Kender`e teşekkür ederim.

Onur ŞANLI

Ocak, 2008

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MÜHENDESLİK FAKÜLTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ
YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI İLE KREDİBİLİTE TESPİTİ

Hazırlayan
Onur ŞANLI

Tez Danışmanı
Prof. Dr. Ali OKATAN

Ocak , 2008

ÖZET

Girişte genel olarak yapay sinir ağları üzerinde durulmuştur. Yapay sinir ağları kavramından C# da uygulanan yapay sinir ağları programına geçilmiştir. Yapay sinir ağlarının kullanım amaçları, avantajları ve dezavantajlarından bahsedilmiştir. Bu aşamada yapay sinir ağları kavramının oluşumunu gerçekleştiren biyolojik sinir ağından söz edilmiştir. Yapay sinir ağlarının kullanıldığı alanlar ve uygulamaları söz konusu olmuştur. Yapay sinir ağlarının tarihsel gelişimi ve yapılan çalışmalar da dikkate alınmıştır. Ayrıca yapay
sinir ağları kavramını oluşturan matematiksel fonksiyonlar göz önüne alınmıştır. Adım adım bu fonksiyondan çıkan formüller ile program kodlaması için temel yapı meydana getirilmiştir.

Yapay sinir ağlarını oluşturan işlem elemanları, giriş katmanı, ara katman ve çıkış

katmanı ayrıntılı bir şekilde göz önüne alınmıştır. Bu aşamada, ağırlıklardan ve kullanılan transfer fonksiyonlarından bahsedilmiştir.C#'da kodlanan programın akış şeması verilmiştir. Programda kullanılan değişkenler ve matematiksel işlem yapısı da akış semasında yer almıştır.Akış şemasının ardından yazılmış olan programın detayı alınmıştır. Programın çalışması için gerekli olan bazı değişkenler program içinde tanımlanmış ve değerleri atanmıştır. Örneğin: ara katmandaki ve çıkış katmanındaki işlem elemanı sayısı gibi ve iterasyon sayısı gibi bazı değişkenler ise programın
çalışması aşamasında dışarıdan girilmiştir.

Daha önceden eğitim seti bilgileri bölümünün içinde yer alan giriş dosyası , çıkış dosyası ve sınama dosyası hazırlanmış ve dosya olarak programın içinde yer almıstır.Programda işlem elemanları üstündeki ağırlıklar ilk önce rasgele (random) alınmıştır.İterasyon sayısına kadar giriş ve çıkışlar birlikte değerlendirilip ağırlıklara son hali verilmistir. Ayrıca programın bu asamasından sonra belli periyotlarla olusan hata sayısal olarak gösterilip grafiği de verilmiştir. Tez çalışması boyunca kullanılan bilgisayarda iterasyon sayısı en fazla 14000 değeri almıştır. Bütün bu işlemler boyunca kullanılan program konsol ortamında çalıştırılmıştır.Eğitim seti bilgileri bölümünde yer alan çıkış dosyasının eğitilmesi sonucunda giriş dosyasındaki verilerin sonuç olarak " 1 rakamı kredi alır ya da 0 rakamı kredi almaz olarak sonuçlanacaktır.

UNIVERSITY OF HALIC
INSTITUE OF TECHNOLOGY SCIENCES
FACULTY OF ENGINEERING

FIXATION OF CREBILITY WITH ARTIFICIAL NEURAL NETWORKS
THESIS OF MASTER`S DEGREE

AUTHOR
ONUR ŞANLI

ADVISER OF THESIS
PROF.DR. ALI OKATAN

OCAK,2008

SUMMARY

At the entrence, the artificial neurals networks were discussed mostly,It passed through from concept of artificial neurals networks to program of artificial neurals networks which processed in c#.At the thesis,you can find advantages, disadvantages and purpose of atificial neurals networks.At this stage it explains concept biological neural network of artificial networks also.Next phases of thesis, it explains devolopment of historical and operations of artificial neural networks.And then contains mathematical functions.Step by step basic informations for computer program comes into existance with formulas .

This computer program wrote at C# programming language and wrote for fixation of credebility with artifical neural networks.Flowchart contains veriable values and mathematical operations.After flowchart it consists code of computer program.There are some values for running computer program in this section and some values are in the computer program.For example : counts of operation constiuent of input and output sections..And some values gives to computer programfrom outside during running.For example : count of step .The iteration number which was used in the program was 14000 at most , at the result of all theese information which was entered as a defult or manual way will finalised as “1 or 0” .Number 1 means that the person has suitable conditionsto take credit.Number 0 means ; the person hasn`t have a suitable conditions to take credit.

GİRİŞ vi

Günümüzde bankacılık sektörü özellikle son yıllarda artan globalleşmeyle birlikte daha da önem kazanmıştırGünümüzde teknoloji yapılan buluşlarla beraber ortaya çıkan yeni ürünleri kazandırmıştır.İnsan hayatını olumlu yönde geliştirmekle beraber alışıla gelmişin dışında yavaş giden hayatı ters yönde etkileyerek hızlandırmıştır.Bilgi teknolojisi daha önemli bir rol kazanmıştır.Hesaplanması zor olan ya da bir kural dahilinde çözülemeyen olgular, bilgisayar yardımı ile kolaylıkla çözülebilir hale gelmiştir.İşletmeler bilgi teknolojisi sayesinde artık daha da verimli hale gelmiştir.

Bankacılıkta ise her geçen gün artan rekabet koşulları bankaları daha da rekabetçi bir ortama taşımıştır.Son zamanlarda artan kredi talepleri bankaların sorumluluklarını dahada artırmıştır.Bankaların kredi verirken ön şart olarak koşullar sürmektedir.Örneğin krediye başvuru yapan kişilerin yaşı , mesleği, kazanç durumları ve diğer şartlarını daha detaylı incelenmektedir.Artan rekabetçi piyasa karşısında bankalar rakiplerinden daha da hızlı olmak zorundadır, hızlı olabilmeleri için ise daha sağlıklı ve daha güvenilir ortam olan bilgisayar ortamını tercih etmektedir.

Hazırladığım tezde; bankaların müşterilerine kredi verip , vermeme konusu hakkındadır.Bu yüzden çalışmamın konusunu yapay sinir ağlarını ile kredibilite konusu seçtim.

BÖLÜM 1.

1. YAPAY SİNİR AĞLARI

1. 1.Yapay Sinir Ağları Nedir ?

İnsanlığın doğayı araştırma ve taklit etme çabalarının en son ürünlerinden bir tanesi, yapay sinir ağları (YSA) teknolojisidir. YSA, basit biyolojik sinir sisteminin çalışma şekli simüle edilerek tasarlanan programlama yaklaşımıdır. Simüle edilen sinir hücreleri (nöronlar) içerirler ve bu nöronlar çeşitli şekillerde birbirlerine bağlanarak ağı oluştururlar. Bu ağlar öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahiptirler. Diğer bir ifadeyle, YSA'lar, normalde bir insanın düşünme ve gözlemlemeye yönelik doğal yeteneklerini gerektiren problemlere çözüm üretmektedir. Bir insanın, düşünme ve gözlemleme yeteneklerini gerektiren problemlere yönelik çözümler üretebilmesinin temel sebebi ise insan beyninin ve dolayısıyla insanın sahip olduğu yaşayarak veya deneyerek öğrenme yeteneğidir.

Biyolojik sistemlerde öğrenme, nöronlar arasındaki sinaptik (synaptic) bağlantıların ayarlanması ile olur. Yani, insanlar doğumlarından itibaren bir yaşayarak öğrenme süreci içerisine girerler. Bu süreç içinde beyin sürekli bir gelişme göstermektedir. Yaşayıp tecrübe ettikçe sinaptik bağlantılar ayarlanır ve hatta yeni bağlantılar oluşur. Bu sayede öğrenme gerçekleşir. Bu durum YSA için de geçerlidir. Öğrenme, eğitme yoluyla örnekler kullanarak olur; başka bir deyişle, gerçekleşme girdi/çıktı verilerinin işlenmesiyle, yani eğitme algoritmasının bu verileri kullanarak bağlantı ağırlıklarını (weights of the synapses) bir yakınsama sağlanana kadar, tekrar tekrar ayarlamasıyla olur.

YSA'lar, ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir. Bir işlem birimi, aslında sık sık transfer fonksiyonu olarak anılan bir denklemdir. Bu işlem birimi, diğer nöronlardan sinyalleri alır; bunları birleştirir, dönüştürür ve sayısal bir sonuç ortaya çıkartır. Genelde, işlem birimleri kabaca gerçek nöronlara karşılık gelirler ve bir ağ içinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır.

Sinirsel (neural) hesaplamanın merkezinde dağıtılmış, adaptif ve doğrusal olmayan işlem kavramları vardır. YSA'lar, geleneksel işlemcilerden farklı şekilde işlem yapmaktadırlar. Geleneksel işlemcilerde, tek bir merkezi işlem birimi her hareketi sırasıyla gerçekleştirir. YSA'lar ise herbiri büyük bir problemin bir parçası ile ilgilenen, çok sayıda basit işlem birimlerinden oluşmaktadır. En basit şekilde, bir işlem birimi, bir girdiyi bir ağırlık kümesi ile

ağırlıklandırır, doğrusal olmayan bir şekilde dönüşümünü sağlar ve bir çıktı değeri oluşturur. İlk bakışta, işlem birimlerinin çalışma şekli yanıltıcı şekilde basittir. Sinirsel hesaplamanın gücü, toplam işlem yükünü paylaşan işlem birimlerinin birbirleri arasındaki yoğun bağlantı yapısından gelmektedir.

Çoğu YSA'da, benzer karakteristiğe sahip nöronlar tabakalar halinde yapılandırılırlar ve transfer fonksiyonları eş zamanlı olarak çalıştırılırlar. Hemen hemen tüm ağlar, veri alan nöronlara ve çıktı üreten nöronlara sahiptirler.

YSA'nın ana öğesi olan matematiksel fonksiyon, ağın mimarisi tarafından şekillendirilir. Daha açık bir şekilde ifade etmek gerekirse, fonksiyonun temel yapısını ağırlıkların büyüklüğü ve işlem elemanlarının işlem şekli belirler. YSA'ların davranışları, yani girdi veriyi çıktı veriye nasıl ilişkilendirdikleri, ilk olarak nöronların transfer fonksiyonlarından, nasıl birbirlerine bağlandıklarından ve bu bağlantıların ağırlıklarından etkilenir

Beynin üstün özellikleri, bilim adamlarini üzerinde çalismaya zorlamis ve beynin nörofiziksel yapisindan esinlenerek matematiksel modeli çikarilmaya çalisilmistir. Beynin bütün davranislarini tam olarak modelleyebilmek için fiziksel bilesenlerinin dogru olarak modellenmesi gerektigi düsüncesi ile çesitli yapay hücre ve ag modelleri gelistirilmistir. Böylece Yapay Sinir Aglari denen yeni ve günümüz bilgisayarlarinin algoritmik hesaplama yönteminden farkli bir bilim alani ortaya çikmistir. Yapay sinir aglari; yapisi, bilgi isleme yöntemindeki farklilik ve uygulama alanlari nedeniyle çesitli bilim dallarinin da kapsam alanina girmektedir.

Genel anlamda YSA, beynin bir islevi yerine getirme yöntemini modellemek için tasarlanan bir sistem olarak tanimlanabilir. YSA, yapay sinir hücrelerinin birbirleri ile çesitli sekillerde baglanmasindan olusur ve genellikle katmanlar seklinde düzenlenir. Donanim olarak elektronik devrelerle yada bilgisayarlarda yazilim olarak gerçeklenebilir. Beynin bilgi isleme yöntemine uygun olarak YSA, bir ögrenme sürecinden sonra bilgiyi toplama, hücreler arasindaki baglanti agirliklari ile bu bilgiyi saklama ve genelleme yetenegine sahip paralel dagilmis bir islemcidir. Ögrenme süreci, arzu edilen amaca ulasmak için YSA agirliklarinin yenilenmesini saglayan ögrenme algoritmalarini ihtiva eder.

1. 2. YAPAY SINIR AĞLARI ÖZELLİKLERİ:

YSA' nin hesaplama ve bilgi isleme gücünü, paralel dagilmis yapisindan, ögrenebilme ve genelleme yeteneginden aldigi söylenebilir. Genelleme, egitim yada ögrenme sürecinde karsilasilmayan girisler için de YSA' nin uygun tepkileri üretmesi olarak tanimlanir. Bu üstün özellikleri, YSA' nin karmasik problemleri çözebilme yetenegini gösterir. Günümüzde birçok bilim alaninda YSA, asagidaki özellikleri nedeniyle etkin olmus ve uygulama yeri bulmustur.

1.2.1. DOGRUSAL OLMAMA:

YSA' nin temel islem elemani olan hücre dogrusal degildir. Dolayisiyla hücrelerin birlesmesinden meydana gelen YSA da dogrusal degildir ve bu özellik bütün aga yayilmis durumdadir. Bu özelligi ile YSA, dogrusal olmayan karmasik problemlerin çözümünde en önemli araç olmustur.

1. 2. 1. ÖGRENME:

YSA' nin arzu edilen davranisi gösterebilmesi için amaca uygun olarak ayarlanmasi gerekir. Bu, hücreler arasinda dogru baglantilarin yapilmasi ve baglantilarin uygun agirliklara sahip olmasi gerektigini ifade eder. YSA' nin karmasik yapisi nedeniyle baglantilar ve agirliklar önceden ayarli olarak verilemez yada tasarlanamaz. Bu nedenle YSA, istenen davranisi gösterecek sekilde ilgilendigi problemden aldigi egitim örneklerini kullanarak problemi ögrenmelidir.

1. 2. 3. GENELLEME:

YSA, ilgilendigi problemi ögrendikten sonra egitim sirasinda karsilasmadigi test örnekleri için de arzu edilen tepkiyi üretebilir. Örnegin, karakter tanima amaciyla egitilmis bir YSA, bozuk karakter girislerinde de dogru karakterleri verebilir yada bir sistemin egitilmis YSA modeli, egitim sürecinde verilmeyen giris sinyalleri için de sistemle ayni davranisi gösterebilir.

1. 2. 4. UYARLANABILIRLIK:

YSA, ilgilendigi problemdeki degisikliklere göre agirliklarini ayarlar. Yani, belirli bir problemi çözmek amaciyla egitilen YSA, problemdeki degisimlere göre tekrar egitilebilir, degisimler devamli ise gerçek zamanda da egitime devam edilebilir. Bu özelligi ile YSA, uyarlamali örnek tanima, sinyal isleme, sistem tanilama ve denetim gibi alanlarda etkin olarak kullanilir.

1. 2. 5. HATA TOLERANSI:

YSA, çok sayida hücrenin çesitli sekillerde baglanmasindan olustugundan paralel dagilmis bir yapiya sahiptir ve agin sahip oldugu bilgi, agdaki bütün baglantilar üzerine dagilmis durumdadir. Bu nedenle, egitilmis bir YSA nin bazi baglantilarinin hatta bazi hücrelerinin etkisiz hale gelmesi, agin dogru bilgi üretmesini önemli ölçüde etkilemez. Bu nedenle, geleneksel yöntemlere göre hatayi tolere etme yetenekleri son derece yüksektir.

1. 2. 6. DONANIM VE HIZ:

YSA, paralel yapisi nedeniyle büyük ölçekli entegre devre (VLSI) teknolojisi ile gerçeklenebilir. Bu özellik, YSA nin hizli bilgi isleme yetenegini artirir ve gerçek zamanli uygulamalarda arzu edilir.

1. 2 . 7. ANALIZ VE TASARIM KOLAYLIGI:

YSA' nin temel islem elemani olan hücrenin yapisi ve modeli, bölüm 1.1'de açiklandigi gibi bütün YSA yapilarinda yaklasik aynidir. Dolayisiyla, YSA' nin farkli uygulama alanlarindaki yapilari da standart yapidaki bu hücrelerden olusacaktir. Bu nedenle, farkli uygulama alanlarinda kullanilan YSA' lari benzer ögrenme algoritmalarini ve teorilerini paylasabilirler. Bu özellik, problemlerin YSA ile çözümünde önemli bir kolaylik getirecektir.

1. 3 . YSA TARİH

YSA, beyindeki sinirlerin çalışmasını taklit ederek sistemlere öğrenme, genelleme yapma,hatırlama gibi yetenekler kazandırmayı amaçlayan bilgi işleme sistemidir.

insan beyninin ve düşünme yeteneğinin taklit edilmesi isteği sanıldığının aksine çok eski zamanlarda var olmuş bir istektir.İnsan beyni ve düşünebilme yeteneğine ilişkin ilk açıklayıcıteori geliştirme denemeleri Antik Yunan düşünürleri olan Plato (İ.Ö. 427-327) ve Aristoteles'e(İ.Ö. 384-322) kadar uzanmaktadır. Daha sonra ise Descartes (1596-1650) insanın düşünme yeteneğiyle ilgilenen 18. yüzyıl düşünürü olmuştur.

Beyinin üstün özellikleri, bilim adamlarını üzerinde çalışmaya zorlamış ve beynin nörofiziksel yapısından esinlenerek matematiksel modeli çıkarılmaya çalışılmıştır. Beynin bütün davranışlarını modelleyebilmek için fiziksel bile_enlerinin doğru olarak modellenmesi gerektiği düşüncesi ile çeşitli yapay hücre ve ağ modelleri geliştirilmiştir.

Böylece, Yapay Sinir Ağları denen günümüz bilgisayarlarının algoritmik hesaplama yöntemlerinden farklı bir bilim alanı ortaya çıkmıştır.

Genel anlamda YSA, beynin bir işlevini yerine getirme yöntemini modellemek için tasarlanan

bir sistem olarak tanımlanabilir. Bir YSA, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde başlanmasında oluşur. YSA'lar öğrenme algoritmaları ile öğrenme sürecinden geçtikten sonra, bilgiyi toplama, hücreler arasındaki bağlantı ağırlıkları ile bu bilgiyi saklama ve genelleme yeteneğine sahip olurlar. YSA lar yapılarına göre farklı öğrenme yaklaşımları kullanırlar.

Yapay sinir ağlarının dayandığı ilk hesaplama modelinin temelleri 1940'larýn başında araştırmalarına başlayan W.S. McCulloch ve W.A. Pitts'in, 1943 yılında yayınladıkları bir makaleyle atılmış olmuştur. Daha sonra 1954 yılında B.G. Farley ve W.A. Clark tarafından bir a_ içerisinde uyarılara tepki veren, uyarılara adapte olabilen model oluşturulmuştur. 1960 yılı ise ilk neural bilgisayarın ortaya çıkış yılıdır. 1963 yılında basit modellerin ilk eksiklikleri fark edilmiş, ancak başarılı sonuçların alınması 1970 ve 1980'lerde termodinamikteki teorik yapıların doğrusal olmayan ağların geliştirilmesinde kullanılmasına kadar gecikmiştir. 1985 yapay sinir ağlarının oldukça tanındığı, yoğun araştırmaların bağladığı yıl olmuştur.

1959'da, Stanford üniversitesinden Bernard Widrow , basit nöron benzeri elemanlara dayanan ve "adaline" (Adaptive Linear Neuron) olarak adlandırılan bir adaptif lineer elemanı geliştirmiştir. Adaline ve iki tabakalı biçimi olan "madaline" (Multiple Adaline); ses tanıma, karakter tanıma, hava tahmini ve adaptif kontrol gibi çok çeşitli uygulamalar için kullanılmıştır. Daha sonraları adaline, ayrık bir çıkış yerine sürekli bir çıkış üretmek için geliştirilmiştir. Widrow, telefon hatları üzerindeki ekoları elimine etmeye yarayan adaptif filtreleri geliştirmede, adaptif lineer eleman algoritmasını kullanmıştır. Bununla ilk defa YSA'lar gerçek bir probleme uygulanmıştır .Helsinki Teknik Üniversitesi'nden Teuvo Kohonen 1970'lerin ilk yıllarında adaptif öğrenme ve birleşik hafızalar üzerine temel çalışmalar yapmış ve bu olduğu çalışmaları ile danışmansız öğrenme metotlarının gelişmesine ışık tutmuştur.

Minsky ve Papert'in perseptron isimli kitaplarındA, YSA'nın temel olarak ilgi çekici konular olmadığını belirtmeleri bir çok araştırmacının bu alanda çalışmaktan vazgeçmelerine sebebiyet vermiştir. YSA konusunda çalışmaya devam eden Grossberg YSA modellerini yapılandırmak için nörolojik verinin kullanılması, algı ve hafıza için YSA tabanlı mekanizmaların önerilmesi, belirgin eşitliklerle bütünleşen bir sinaptik model için bir ilişkilendirici kural üzerinde çalışmıştır.

1982 yılında ilgi çeken bir başka gelişme, moleküller biyolojiden beyin kuramcılığına geçiş yapan bir model Caltech fizikçisi Hopfield tarafından sunulmuştur. Kendi adıyla anılan

bir ağ yapısı mevcuttur ve bir çok alana uygulanmıştır.

1987 yılında yapılan ilk yapay sinir ağları sempozyumundan sonra YSA uygulamaları yaygınlaşmıştır. Günümüzde, YSA'larla ilgili araştırmalar yapan çok sayıda bilim adamı ve araştırma grupları vardır. Farklı bilim ve ilgi alanlarında çalışan birçok araştırmacı, birçok yeni gelişmeleri sunmaya devam edeceklerdir.Takip eden bölümlerde biyolojik nörondan yapay nörona geçiş anlatılmaktadır.

- Hepner ve Ritter (1989) yapay sinir aglarinin yer örtüsünü siniflandirabilecegini gösterdi.
- Key ve digerleri (1989) dört yüzeyde sekiz bulut sinifi taninmasinda yapay sinir aglarinin üstün basarimini gösterdi.
- Hepner ve digerleri (1990) hata geriye yayma algoritmasinin az veri ile egitimde, süpervize diger siniflandiricilardan daha basarili oldugunu gösterdi.
- Ryan ve digerleri (1991) uydu fotograflarindan deniz kiyilarinin taninmasi uygulamasini yapti
- Lee (1990) ve Slawinsky (1991) yapay sinir aglari ile bulut tanima uygulamalari gerçeklestirdi.
- Liu ve Xiao (1991) standart hata geriye yayma algoritmasinin Thematic Mapper (TM) verisi için yakinsama yavasligi problemini bloke geriye yayilim (Blocked Backpropagation) kullanarak asti.
- Safavian ve Tenorio (1991) standart geriye yayilim algoritmasinin yer örtülerini MSS (Airborne Multi-spectral Scanner) görüntülerinden siniflayabilecegini buldular fakat SOM kadar iyi çalismadi.
- Omatu ve Yosida (1991) Landsat TM verilerini yapay sinir aglarinin Bayesian siniflandiricisindan daha iyi tanidigini deneysel olarak ispatladi.
- Kanellopoulos ve digerleri (1992) SPOT görüntüsünde yapay sinir aglarinin maksimum benzerlik siniflamasindan daha basarili oldugunu gösterdi.
- Boggess (1993) yapay sinir aglari ile uydu görüntüsünden yollarin taninmasini gerçeklestirdi.

- Dreyer (1993) optimize yapay sinir aglarinin basarili kullanimi ile yer örtüsünü 4

kategoride siniflandirdi.

- Dawson ve digerleri (1993) yapay sinir agi uygulamalarini gelistirerek deniz buzunun taninmasini sagladi.
- Bruzzone (1999) multitemporal ve çok kaynakli uzaktan algilama görüntülerinin siniflanmasina nöral-istatistiksel bir yaklasim getirdiler.
- Crosson (2002) hava tahmini ve tarimla ilgili uygulamalar için önemli olan yüzeye yakin toprak nemini mikrodalga uzaktan algilama sensörleri ile YSA tabanli bir model kullanarak kestirdi.

[1] Mehra Pankaj Wah W Benjamin, Artifisal Neural Networks Concepts and Theory, IEEE Computer Society Press, Washington, 1992, page 45

[2] CAUDILL, Maureen ‘‘Neural Network Primar Part 1’’ AI Expert, December 1987,pp47

1. 4. Yapay Sinir Ağları'nın Uygulama Alanları

Son yıllarda YSA' ları, özellikle günümüze kadar çözümü güç ve karmaşık olan yada ekonomik olmayan çok farklı alanlardaki problemlerin çözümüne uygulanmış ve genellikle başarılı sonuçlar alınabilmiştir. YSA' ları çok farklı alanlara uygulanabildiğinden bütün uygulama alanlarını burada sıralamak zor olmakla birlikte genel bir sınıflandırma ile YSA' nın uygulama alanları aşağıdaki gibi 6 grup içerisinde toplanabilir.

1. 4. 1 Arıza Analizi ve Tespiti

Bir sistemin, cihazın yada elemanın düzenli (doğru) çalışma şeklini öğrenen bir YSA yardımıyla bu sistemlerde meydana gelebilecek arızaların tanımlanma olanağı vardır. Bu amaçla YSA; elektrik makinelerinin, uçakların yada bileşenlerinin, entegre devrelerin v.s. arıza analizinde kullanılmıştır.

1. 4. 2Tıp Alanında

EEG ve ECG gibi tıbbi sinyallerin analizi, kanserli hücrelerin analizi, protez tasarımı, transplantasyon zamanlarının optimizasyonu ve hastanelerde giderlerin optimizasyonu v.s gibi uygulama yeri bulmuştur.

1. 4. 3. Savunma Sanayi

Silahların otomasyonu ve hedef izleme, nesneleri/görüntüleri ayırma ve tanıma, yeni algılayıcı tasarımı ve gürültü önleme v.s gibi alanlara uygulanmıştır.

1. 4. 4. Haberleşme

Görüntü ve veri sıkıştırma, otomatik bilgi sunma servisleri, konuşmaların gerçek zamanda çevirisi v.s gibi alanlarda uygulama örnekleri vardır.

1. 4. 5. Üretim

Üretim sistemlerinin optimizasyonu, ürün analizi ve tasarımı, ürünlerin (entegre, kağıt, kaynak v.s.) kalite analizi ve kontrolü, planlama ve yönetim analizi v.s. alanlarına uygulanmıştır.

1. 4. 6. Otomasyon ve Kontrol

Uçaklarda otomatik pilot sistemi otomasyonu, ulaşım araçlarında otomatik yol bulma/gösterme, robot sistemlerin kontrolü, doğrusal olmayan sistem modelleme ve kontrolü, elektrikli sürücü sistemlerin kontrolü v.s. gibi yaygın bir uygulama yeri bulmuştur.

[2] B. Dengiz, Sağlık Bilimlerinde Yapay Sinir A ları

1. 5. YAPAY HÜCRE MODELLERİ

Yapay sinir hücreleri, YSA' nın çalışmasına esas teşkil eden en küçük bilgi işleme birimidir. Geliştirilen hücre modellerinde bazı farklılıklar olmakla birlikte genel özellikleri ile bir yapay hücre modeli, şekil 1.1 de görüldüğü gibi girdiler, ağırlıklar, birleştirme fonksiyonu, aktivasyon (etkinleştirme) fonksiyonu ve çıktılar olmak üzere 5 bileşenden meydana gelir. Girdiler, diğer hücrelerden ya da dış ortamlardan hücreye giren bilgilerdir. Bilgiler, bağlantılar üzerindeki ağırlıklar üzerinden hücreye girer ve ağırlıklar, ilgili girişin hücre üzerindeki etkisini belirler. Birleştirme fonksiyonu, bir hücreye gelen net girdiyi hesaplayan bir fonksiyondur ve genellikle net girdi, girişlerin ilgili ağırlıkla çarpımlarının toplamıdır.

Birleştirme fonksiyonu, ağ yapısına göre maksimum alan, minimum alan ya da çarpım fonksiyonu olabilir. Aktivasyon fonksiyonu ise birleştirme fonksiyonundan elde edilen net girdiyi bir işlemden geçirerek hücre çıktısını belirleyen ve genellikle doğrusal olmayan bir fonksiyondur. Hücre modellerinde, net girdiyi artıran +1 değerli polarma girişi yada azaltan -1 değerli eşik girişi bulunabilir ve bu giriş de sabit değerli bir giriş olarak girdi vektörü (x_0), katsayısı ise (genellikle b ile gösterilir) ağırlık vektörü (W_0) içerisine alınabilir. Genel olarak hücre modelleri şekil 1.1 deki gibi olmakla birlikte gerçekleştirdiği işleve göre hücreler statik yada dinamik bir davranış gösterebilirler.

1. 5. 1. Statik Hücre Modeli

Ağırlıkların sabit olduğu ve hücrede geri besleme yada geciktirilmiş sinyaller kullanılmadığı dikkate alınırsa bu hücre statik bir işlevi gerçekleştireceğinden bu model, statik hücre modeli olarak söylenebilir. Sıfır indisler polarma girişini ve ağırlığını göstermek üzere (yada polarma girişi birim değerli olduğundan sadece polarma ağırlı b ile gösterilmek üzere) statik hücrenin matematiksel modeli Denklem 1.1 deki gibi yazılabilir.

$$v = \sum_{i=0}^{n} W_i x_i \quad yada \quad v = \sum_{i=1}^{n} W_i x_i + b \quad , \quad y = \varphi(v)$$

Burada; **W**- hücrenin ağırlıklar matrisini, **x**- hücrenin giriş vektörünü, **v**- hücrenin net girişini, **y**- hücre çıkışını ve **φ(.)**- hücrenin aktivasyon fonksiyonunu göstermektedir., **x** giriş vektörünün bileşenlerinin dış (geri beslemesiz) girişler olması durumunda hücrenin doğrusal olmayan statik bir işlevi gerçekleştireceği görülmektedir.

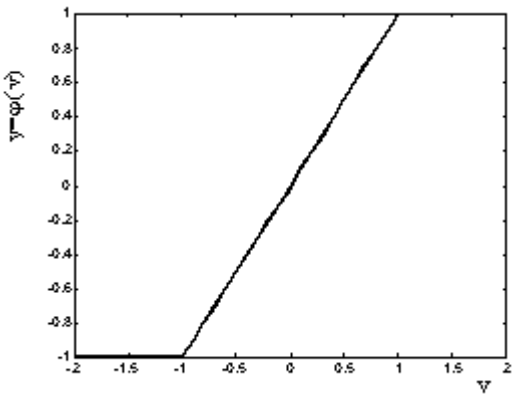
1. 5. 2. Aktivasyon Modelleri

Hücre modellerinde, hücrenin gerçekleştireceği işleve göre çeşitli tipte aktivasyon fonksiyonları kullanılabilir. Aktivasyon fonksiyonları sabit parametreli yada uyarlanabilir parametreli seçilebilir. Aşağıda, hücre modellerinde yaygın olarak kullanılan çeşitli aktivasyon fonksiyonları tanıtılmıştır.

1. 5. 3. Doğrusal ve Doyumlu-doğrusal Aktivasyon Fonksiyonu

Doğrusal bir problemi çözmek amacıyla kullanılan doğrusal hücre ve YSA' da yada genellikle katmanlı YSA' nın çıkış katmanında kullanılan doğrusal fonksiyon, hücrenin net girdisini doğrudan hücre çıkışı olarak verir. Doğrusal aktivasyon fonksiyonu matematiksel olarak **y=v** şeklinde tanımlanabilir. Doyumlu doğrusal aktivasyon fonksiyonu ise aktif çalışma bölgesinde doğrusaldır ve hücrenin net girdisinin belirli bir değerinden sonra hücre çıkışını doyuma götürür. Doyumlu doğrusal aktivasyon fonksiyonunun matematiksel tanımı, Şekil 1.4 de ise grafiği görülmektedir.

$$y = \begin{Bmatrix} 1 & v > 1 \\ v & -1 < v < 1 \quad ise \\ -1 & v < -1 \end{Bmatrix} \qquad 1.2$$



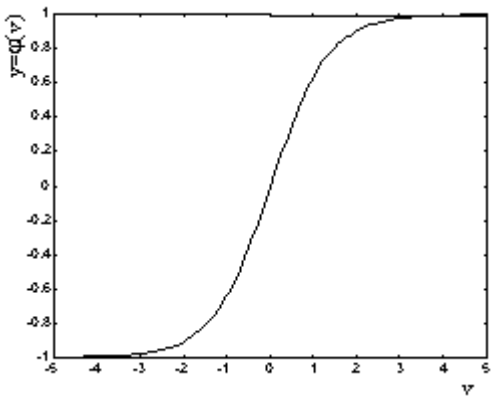
Şekil 1.1 *Doyumlu doğrusal aktivasyon fonksiyonu*

1. 5. 4. Sigmoid Aktivasyon Fonksiyonu

Şekil 1.1 de grafiği verilen çift yönlü sigmoid (tanh) fonksiyonu, türevi alınabilir, sürekli ve doğrusal olmayan bir fonksiyon olması nedeniyle doğrusal olmayan problemlerin çözümünde kullanılan YSA'larında tercih edilir. Çift yönlü sigmoid fonksiyonun tanımı denklem 1.3 de ve tek yönlü sigmoid fonksiyonunun matematiksel ifadesi ise denklem 1.4 de verilmiştir.

$$\varphi(v) = a\frac{1-e^{-bv}}{1+e^{-bv}} \qquad 1.3$$

$$\varphi(v) = a\frac{1}{1+e^{-bv}} \qquad 1.4$$



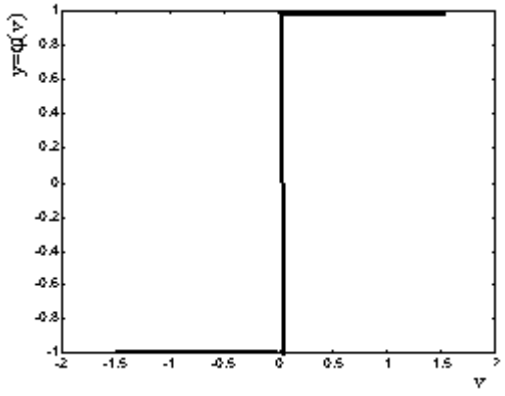
Şekil 1.3 *Sigmoid (tanh) aktivasyon fonksiyonu.*

Sigmoid fonksiyonlarında a ve b katsayıları genellikle birim olarak alınır ancak, YSA' nın eğitiminde öğrenme oranını hızlandırıcı etkilerinin olduğu ve en uygun değerleri ise a=1.716 , b=2/3 olarak belirlenmiştir. Ayrıca, a ve b katsayılarının YSA' nın eğitim sürecinde uyarlanmasıyla sabit katsayılı fonksiyona göre daha iyi bir performans elde edilebilmektedir.

1. 5. 5. Eşik Aktivasyon Fonksiyonu

McCulloch-Pitts modeli olarak bilinen eşik aktivasyon fonksiyonlu hücreler, mantıksal çıkış verir ve sınıflandırıcı ağlarda tercih edilir, şekil 1.2. Perceptron (Algılayıcı) olarak da söylenen eşik fonksiyonlu hücrelerin matematiksel modeli aşağıdaki gibi tanımlanabilir.

$$y = \begin{cases} 1 & v \geq 0 \\ -1 & v < 0 \end{cases} \quad \text{ise} \qquad 1.5$$

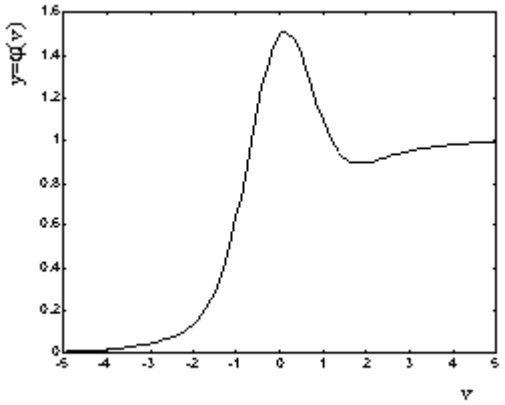


Şekil 1.4 *Eşik aktivasyon fonksiyonu*

1. 5. 6. Diğer Aktivasyon Fonksiyonları

Yukarıda verilen ve yaygın olarak kullanılan aktivasyon fonksiyonlarının dışında YSA' da çeşitli aktivasyon fonksiyonları kullanılmış ve aktivasyon fonksiyonlarına göre YSA' nın problemleri çözebilme performansları incelenmiştir.

$$y = \varphi(v) = a_1 e^{-b_1 v^2} + a_2 \cdot \frac{1}{1+e^{-b_2 v}} \quad 1.6$$



Şekil 1.5

a ve b katsayıları, YSA' nın eğitim sürecinde uyarlanmak üzere denklem 1.6 verilen aktivasyon fonksiyonuna sahip YSA' nın çeşitli problemlerin çözümünde etkin olduğu

belirlenmiştir. Farklı problemlerde a1, b1 ve a2, b2 katsayıları farklı değerlere yakınsayarak YSA' nın eğitimini daha etkin kılmaktadır. Denklem normal sigmoid fonksiyonlarına göre YSA' nın eğitimini hızlandırıcı etkisi olduğu belirlenen karesel sigmoid aktivasyon fonksiyonunun matematiksel ifadesi verilmiştir. Denklem ise, a katsayısına bağlı olarak tanımlanan belirli bir aralıkta doğrusal ve bu aralıkların dışında sigmoidal değişen uyarlanabilir parametreli aktivasyon fonksiyonununda çeşitli problemlerin YSA ile çözümünde etkin olduğu incelenmiştir. Uyarlanabilir a katsayısı 0-1 arasında değişmektedir.

$$y = \varphi(v) = \frac{1}{1+e^{-v}} \qquad 1.7$$

$$y = \varphi(v) = \begin{cases} v & |v| < a \quad \text{ise} \\ (\pm)\left[(1-a)\tanh(a+\frac{|v|-a}{1-a}\right] & |v| > a \end{cases} \qquad 1.8$$

1. 5. 6. Dinamik Hücre Modelleri

Şekil 1.3 de verilen yapay hücre modeli, x girişlerinden y çıkışlarına doğrusal olmayan statik bir dönüşümü gerçekleştirir. Örüntü tanıma ve sınıflandırma uygulamalarında statik hücre yada YSA modelleri uygun olmakla birlikte sistem modelleme ve denetimi gibi dinamik problemlerin çözümünde dinamik hücre yada YSA yapılarının kullanılması gereklidir. YSA yapıları çeşitli şekillerde elde edilebilir. Ancak, dinamik bir hücre genel olarak 2 şekilde oluşturulabilir.

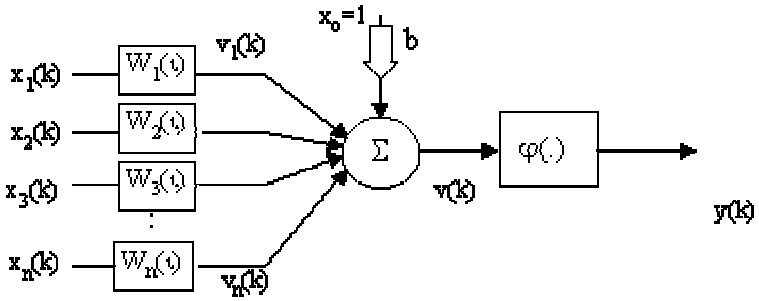
a-) Hücrenin ağırlıkları dinamik bir model (bir filtre) olarak seçilebilir.

b-) Hücrenin net girdisi dinamik bir modelden (bir filtre) geçirilebilir.

1. 5. 7. FIR Filtre Ağırlıklı Dinamik Hücre Modeli

Hücre ağırlıkları sabit seçilmek yerine bir filtre olarak modellenerek hücrenin dinamik davranışı sağlanabilir. Böylece, herhangi bir ağırlığın dinamik davranışı, zamanın bir fonksiyonu olan ani darbe cevabı ile tanımlanabilir. Her bir hücre ağırlığının FIR filtre olarak

modellendiği ayrık zamanlı hücre yapısı şekil 1.8'de verilmiştir.



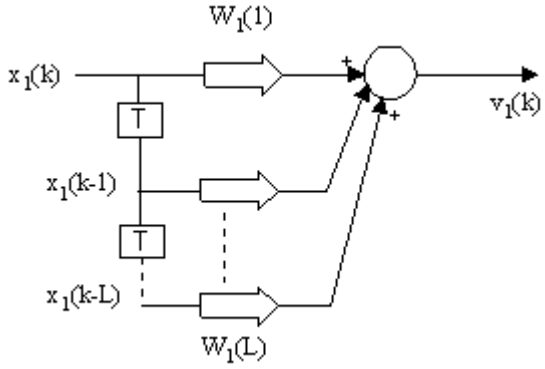
Şekil 1.6 *FIR filtre ağırlıklı dinamik hücre modeli*

Şekil 1.8 de verilen dinamik hücrenin matematiksel modeli aşağıdaki gibi yazılabilir.

$$v(k) = \sum_{i=1}^{n} \sum_{l=1}^{L} W_i(l).x_i(k-l) + b$$

1.9

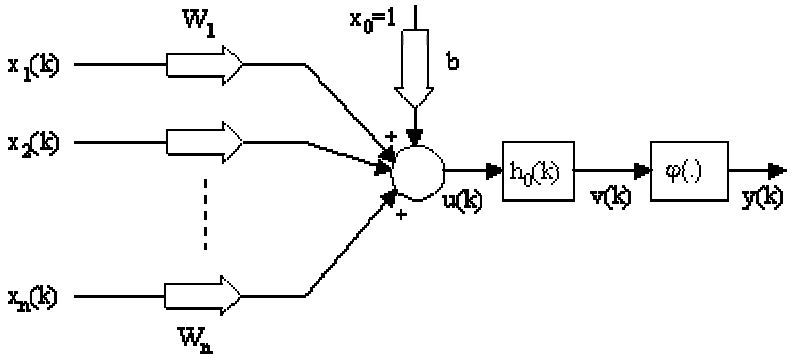
Burada L- filtrenin sonlu bellek (gecikme) sayısı olarak tanımlanır ve ağırlıkların gerçekleştirdiği işlevler, sadece birinci hücre girişi için ayrık zamanda şekil 1.9 da verilen blok şema ile gösterilebilir.



Şekil 1.7 *FIR filtre olarak tasarlanan ağırlıklar*

1. 5. 8. RC-Dinamik Hücre Modeli

Diğer bir dinamik hücre modeli, ağırlıkların dinamik bir model olarak seçilmesi yerine hücrenin net girdisinin doğrusal bir dinamik modelden (filtreden) geçirildiği hücre modelidir. Filtrenin seçimi farklı olabilmekle birlikte genellikle birinci dereceden bir filtre (RC filtre) kullanılır. Bu durumda filtrenin ani darbe cevabı $h_0(k)$ ya göre hücre modeli şekil 1.7'daki gibi çizilebilir.



Şekil 1.8 *RC- dinamik bir hücre modeli*

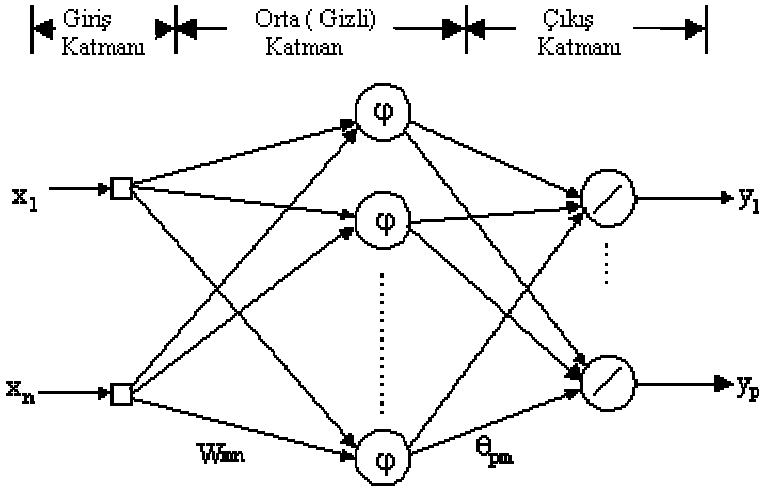
1. 6. YAPAY SİNİR AĞI YAPILARI

Yapay sinir ağları, hücrelerin birbirleri ile çeşitli şekillerde bağlanmalarından oluşur. Hücre çıkışları, ağırlıklar üzerinden *diğer hücrelere* ya da *kendisine* giriş olarak bağlanabilir ve bağlantılarda gecikme birimi de kullanılabilir. Hücrelerin bağlantı şekillerine, öğrenme kurallarına ve aktivasyon fonksiyonlarına göre çeşitli YSA yapıları geliştirilmiştir. Bu bölümde, çeşitli problemlerin çözümünde kullanılan ve kabul görmüş bazı YSA yapıları ayrıntısına girmeksizin genel özellikleri ile tanıtılacaktır.

1. 6. 1.İleri Beslemeli Yapay Sinir Ağları (İBYSA)

İleri beslemeli YSA' da, hücreler katmanlar şeklinde düzenlenir ve bir katmandaki hücrelerin çıkışları bir sonraki katmana ağırlıklar üzerinden giriş olarak verilir. Giriş katmanı, dış ortamlardan aldığı bilgileri hiçbir değişikliğe uğratmadan orta (gizli) katmandaki hücrelere iletir. Bilgi, orta ve çıkış katmanında işlenerek ağ çıkışı belirlenir. Bu yapısı ile ileri beslemeli ağlar doğrusal olmayan statik bir işlevi gerçekleştirir. İleri beslemeli 3 katmanlı YSA' nın, orta katmanında yeterli sayıda hücre olmak kaydıyla, herhangi bir sürekli fonksiyonu istenilen

doğrulukta yaklaştırabileceği gösterilmiştir. En çok bilinen geriye yayılım öğrenme algoritması, bu tip YSA ların eğitiminde etkin olarak kullanılmakta ve bazen bu ağlara geriye yayılım ağları da denmektedir. Şekil 1.8 de giriş, orta ve çıkış katmanı olmak üzere 3 katmanlı ileri beslemeli YSA yapısı verilmiştir.



Şekil 1.9 *İleri beslemeli 3 katmanlı YSA.*

İleri beslemeli 3 katmanlı ve *çıkış katmanı doğrusal* olan YSA' nın matematiksel modeli, **x**- giriş vektörünü, **o**- orta katman çıkış vektörün ü, **y**- ağ çıkış vektörünü göstermek üzere Denklem 1.10 daki gibi yazılabilir. x_0 ve o_0 girişleri, polarma girişleri olarak alınmıştır.

$$v_j = \sum_{i=0}^{n} W_{ji} x_i \;,\; o_j = \varphi(v_j) \qquad j = 1,2,\ldots\ldots m$$

$$y_k = \sum_{j=0}^{m} \theta_{kj} o_j \qquad k = 1,2,\ldots\ldots p \qquad 1.10$$

Herhangi bir problemi çözmek amacıyla kullanılan YSA da, katman sayısı ve orta katmandaki hücre sayısı gibi kesin belirlenememiş bilgilere rağmen nesne tanıma ve sinyal işleme gibi alanların yanı sıra ileri beslemeli YSA, sistemlerin tanılanması ve denetiminde de yaygın olarak kullanılmaktadır.

1.6.2Geri Beslemeli Yapay Sinir Ağları (GBYSA)

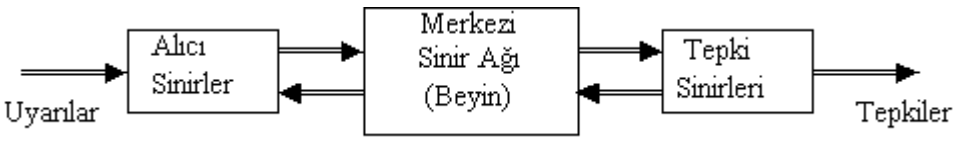
Geri beslemeli YSA' da, en az bir hücrenin çıkışı kendisine ya da diğer hücrelere giriş olarak verilir ve genellikle geri besleme bir geciktirme elemanı üzerinden yapılır. Geri besleme, bir katmandaki hücreler arasında olduğu gibi katmanlar arasındaki hücreler arasında da olabilir. Bu yapısı ile geri beslemeli YSA , doğrusal olmayan dinamik bir davranış gösterir. Dolayısıyla, geri beslemenin yapılış şekline göre farklı yapıda ve davranışta geri beslemeli YSA yapıları elde edilebilir. Bu nedenle, bu bölümde bazı geri beslemeli YSA yapılarında örnekler verilecektir.

BÖLÜM 2.

2. BİYOLOJİK ANLAMDA YAPAY SİNİR AĞLARI

2. 1. BİYOLOJİK SİNİR SİSTEMİ

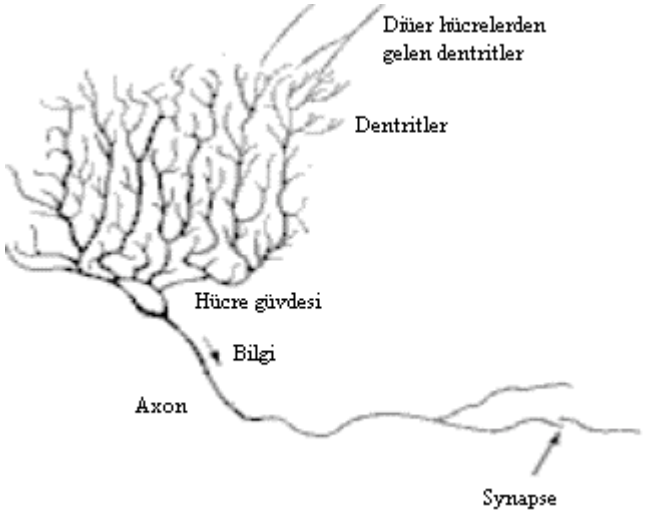
Biyolojik sinir sistemi, merkezinde sürekli olarak bilgiyi alan, yorumlayan ve uygun bir karar üreten beynin (merkezi sinir ağı) bulunduğu 3 katmanlı bir sistem olarak açıklanır. Alıcı sinirler (receptor) organizma içerisinden ya da dış ortamlardan algıladıkları uyarıları, beyine bilgi ileten elektriksel sinyallere dönüştürür. Tepki sinirleri (effector) ise, beyinin ürettiği elektriksel darbeleri organizma çıktısı olarak uygun tepkilere dönüştürür. Şekil 2.1 de bir sinir sisteminin blok gösterimi verilmiştir.



Şekil 2.1.*Biyolojik sinir sisteminin blok gösterimi*

Merkezi sinir ağında bilgiler, alıcı ve tepki sinirleri arasında ileri ve geri besleme yönünde değerlendirilerek uygun tepkiler üretilir. Bu yönüyle biyolojik sinir sistemi, kapalı çevrim denetim sisteminin karakteristiklerini taşır. Merkezi sinir sisteminin temel işlem elemanı, sinir hücresidir (nöron) ve insan beyninde yaklaşık 10 milyar sinir hücresi olduğu tahmin edilmektedir. Sinir hücresi; hücre gövdesi, dendriteler ve axonlar olmak üzere 3 bileşenden

meydana gelir. Dendriteler, diğer hücrelerden aldığı bilgileri hücre gövdesine bir ağaç yapısı şeklinde ince yollarla iletir. Axonlar ise elektriksel darbeler şeklindeki bilgiyi hücreden dışarı taşıyan daha uzun bir yoldur. Axonların bitimi, ince yollara ayrılabilir ve bu yollar, diğer hücreler için dendriteleri oluşturur. Şekil 2.2 de görüldüğü gibi axon-dendrite bağlantı elemanı synapse olarak söylenir.



Şekil 2.2.*Biyolojik Sinir Hücresi ve Bileşenleri.*

Synapse gelen ve dendriteler tarafından alınan bilgiler genellikle elektriksel darbelerdir ancak, synapsedeki kimyasal ileticilerden etkilenir. Belirli bir sürede bir hücreye gelen girişlerin değeri, belirli bir eşik değerine ulaştığında hücre bir tepki üretir. Hücrenin tepkisini artırıcı yöndeki girişler uyarıcı, azaltıcı yöndeki girişler ise önleyici girişler olarak söylenir ve bu etkiyi synapse belirler.

İnsan beyninin 10 milyar sinir hücresinden ve 60 trilyon synapse bağlantısından oluştuğu düşünülürse son derece karmaşık ve etkin bir yapı olduğu anlaşılır. Diğer taraftan bir sinir hücresinin tepki hızı, günümüz bilgisayarlarına göre oldukça yavaş olmakla birlikte duyusal bilgileri son derecede hızlı değerlendirebilmektedir. Bu nedenle insan beyni; öğrenme, birleştirme, uyarlama ve genelleştirme yeteneği nedeniyle son derece karmaşık, doğrusal olmayan ve paralel dağılmış bir bilgi işleme sistemi olarak tanımlanabilir.

BÖLÜM 3.

3. C# DA HAZIRLANAN PROGRAMIN HAZIRLANIŞ PROGRAMI

3.1. UYGULAMA PROGRAMI

Sayın Metin Şahin`e ait yapay sinir ağları programı, C# kullanılarak yapılmıştır. ”Delta yönetimi” ve “Sigmoid Aktivasyon Fonksiyonu” kullanılarak hazırlanmıştır.

Programın içeriği ;

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;


namespace WindowsApplication1
{

  public partial class Form1 : Form
  {
    private int dd2 = 675, dd3, yy3, yy2, sonhd, sonhd1;
    private int[] sonh = new int[25];

    public Form1()
    {


      InitializeComponent();
      textBox1.Text = "C:\\meslek011.txt";
```

```
    textBox2.Text = "C:\\meslek021.txt";
    textBox3.Text = "C:\\meslek031.txt";
    textBox4.Text = "3";
    textBox5.Text = "72";
    textBox6.Text = "15";
    textBox7.Text = "1";
    textBox8.Text = "0,5";
    textBox9.Text = "0,8";

}

private void Form1_Load(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
string dosegigiris1 ="C:\\kriterb.txt";
if (textBox1.Text == "") textBox1.Text = dosegigiris1;
string dosegigiris = textBox1.Text;
string dosegicikis1 ="C:\\cikisb.txt";
if (textBox2.Text == "") textBox2.Text = dosegicikis1;
string dosegicikis = textBox2.Text;
string dosbilgiris1 ="C:\\kontrolb.txt";
if (textBox3.Text == "") textBox3.Text = dosbilgiris1;
string dosbilgiris = textBox3.Text;

FileStream fs = new FileStream(dosegigiris, FileMode.Open);
FileStream fs2 = new FileStream(dosegicikis, FileMode.Open);
FileStream fs3 = new FileStream(dosbilgiris, FileMode.Open);

StreamReader sr = new StreamReader(fs);
```

```
StreamReader sr2 = new StreamReader(fs2);
StreamReader sr3 = new StreamReader(fs3);

string Line;
string Line2;
string Line3;

int    xsu    = int.Parse(textBox4.Text);
int    xsa    = int.Parse(textBox5.Text);
int    akie   = int.Parse(textBox6.Text);
int    cies   = int.Parse(textBox7.Text);
double epsilon = double.Parse(textBox8.Text);
double alfa   = double.Parse(textBox9.Text);
int    adim   = int.Parse(textBox10.Text);
double hasab  = double.Parse(textBox11.Text);

// GİRİŞ İLE ARA KATMAN ARASINDAKİ AĞIRLIKLARIN ATANMASI
double [ , ]  wg = new double [akie,xsu];
for (int i=0 ; i<akie ; i++)
   for(int j=0 ;j<xsu ; j++)
   {
   TEKRAR1:
      double number = Program.getShapeNumber();
      double sayi = number / 100;
      if (sayi == 0) goto TEKRAR1;
      wg[i, j] = sayi;
   }
// ARA KATMANDAKİ AĞIRLIKLARIN ATANMASI
double[] wge = new double[akie];
for (int i = 0; i < akie; i++)
{
TEKRAR2:
   double number = Program.getShapeNumber();
```

```
            double sayi = number / 100;
            if (sayi == 0) goto TEKRAR2;
            wge[i] = sayi;
        }
        // ARA KATMAN İLE ÇIKTI KATMANI ARASINDAKİ AĞIRLIKLARIN ATANMASI
        double[ , ] wc = new double[cies,akie];
        for (int j = 0; j < cies; j++)
        {
            for (int i = 0; i < akie; i++)
            {
            TEKRAR3:
                double number = Program.getShapeNumber();
                double sayi = number / 100;
                if (sayi == 0) goto TEKRAR3;
                wc[j, i] = sayi;
            }
        }
        // ÇIKIŞ KATMANINDAKİ AĞIRLIKLARIN ATANMASI
        double[] wce1 = new double[cies];
        for (int i = 0; i < cies; i++)
        {
        TEKRAR4:
            double number = Program.getShapeNumber();
            double sayi = number / 100;
            if (sayi == 0) goto TEKRAR4;
            wce1[i] = sayi;
        }
        // GİRİŞLERİN VE ÇIKTILARIN OKUNMASI
        double[,] x = new double[xsa, xsu];
        double hasi1;
        double hasi2;
        double hasi3;
```

```
int xsatır = 0;
int xsutun = 0;
// EĞİTİM İÇİN GİRİŞ DOSYASININ OKUNMASI
char[] ayırıcı = { ' ' };
while ((Line = sr.ReadLine()) != null)
{
   xsutun = 0;
   //System.Console.WriteLine("satır = "+xsatır+"---"+Line);
   string[] sayılar = Line.Split(ayırıcı);
   foreach (string si in sayılar)
   {
      hasi1 = Convert.ToDouble(si);
      x[xsatır, xsutun] = hasi1;
      xsutun = xsutun + 1;
    //  System.Console.WriteLine("satır : " + xsatır + "---" + "sutun : " + xsutun);
   }
   xsatır = xsatır + 1;
}
fs.Close();
// EĞİTİM GİRİŞİ İÇİN OKUNAN MATRİSİN NORMALİZASYONU
double[] buyukx = new double[xsu];
for (int i11 = 0; i11 < xsu; i11++)
{
   buyukx[i11] = x[0, i11];
   for (int j11 = 1; j11 < xsa; j11++)
   {
      if (x[j11, i11] >= buyukx[i11]) buyukx[i11] = x[j11, i11];
   }
}
for (int q11 = 0; q11 < xsu; q11++)
{
   if (buyukx[q11] == 0) buyukx[q11] = 0.0000000000001;
   //System.Console.WriteLine("buyukx[" + q11 + "] :" + buyukx[q11]);
```

```
            //string ikk = System.Console.ReadLine();
        }
            //System.Console.WriteLine();
            //System.Console.WriteLine("         EĞİTİM İÇİN GİRİŞ DOSYASINI OKUDU");
        for (int z11 = 0; z11 < xsu; z11++)
        {
            for (int zj11 = 0; zj11 < xsa; zj11++)
            {
                x[zj11, z11] = x[zj11, z11] / buyukx[z11];
            //    System.Console.Write( x[zj11, z11]+" ");
                //string ikk11 = System.Console.ReadLine();
            }
            //System.Console.WriteLine();
            //System.Console.WriteLine();
        }

//        System.Console.WriteLine();
        // EĞİTİM İÇİN ÇIKIŞ DOSYASININ OKUNMASI
        double[,] dk = new double[xsa,cies];
        int dsatır = 0;
        int dsutun = 0;
        string ya2i;
        char[] ayırıcı2 = { ' ' };
        while ((Line2 = sr2.ReadLine()) != null)
        {
            dsutun = 0;
            //string[] sayılar2 = Line2.Split(ayırıcı2);
            //foreach (string ya2i in sayılar2)
            //{
                ya2i = Line2;
                hasi2 = Convert.ToDouble(ya2i);
                dk[dsatır, dsutun] = hasi2;
              //  dsutun = dsutun + 1;
```

```
    //}
    dsatır = dsatır + 1;
}
fs2.Close();

// System.Console.WriteLine("        EĞİTİM İÇİN ÇIKIŞ DOSYASINI OKUDU");
string[] harf = new string[2];

harf[0] = "KREDİ ALIR"; harf[1] = "KREDİ ALMAZ";

int t = 0;
double buyuk = 0;
int step = 0;
int adres = 0;
//int kont2 = 0;
int haadim = 0;
int skont = 0;
int oran = 0;
double fark = 0;
double toplam = 0;
double kare = 0;
double ilkb = 0;
double ilkd = 0;
double ilkg = 0;
double ilkc = 0;
double mse = 0;
int ii = 2;
int i2 = 1;
double top, top1 , top2;
int adim1 = adim + 1;

double[] hadez = new double[adim1];
double[] sondiz = new double[20];
```

```
    double[,] topwkj = new double[xsa,cies];
    double[,] net = new double[xsa,akie];
    double[,] ciktia = new double[xsa,akie];
    double[,] net1 = new double[xsa,cies];
    double[,] net11 = new double[xsa,cies];
   //  double[,] net11x = new double[60,cies];
    double[,] net11x = new double[72, cies];
    double[,] cikti = new double[xsa,cies];
    double[,] ros = new double[xsa,cies];
    double[,] dwc = new double[cies,akie];
    double [] dwce1 = new double [cies];
    double[,] roy = new double[xsa,akie];
    double[] dwge = new double[akie];
    double[,] dwg = new double[akie,xsu];
    double[,] ciktison = new double[xsa,cies];
    double[,] ciktisonx = new double[72,cies];
    double[,] netara = new double[xsa,akie];
    double[,] netarax = new double[72, akie];
    double[,] ciktiara = new double[xsa,akie];
    double[,] ciktiarax = new double[72, akie];

    oran = adim / 22;
ILKBAS:
    //if (mse < 0.0395) goto SONYAZ;
    if (t >= adim) goto SONYAZ;
    step=step+1;
    toplam = 0;
    for (int i1 = 0; i1 < xsa; i1++)
    {
       // ARA KATMANDAKİ NETLERİN HESAPLANMASI
       for (int k = 0; k < akie; k++)
       {
          net[i1,k] = 0;
```

```
    for (int j = 0; j < xsu; j++)
    {
       net[i1,k] = x[i1, j] * wg[k, j] + net[i1,k];
    }
    net[i1,k] = net[i1,k] + wge[k];
    ciktia[i1,k] = 1 / (1 + System.Math.Exp(-net[i1,k]));
}
// ÇIKTI KATMANINDAKİ NETLERİN HESAPLANMASI
for (int k1 = 0; k1 < cies; k1++)
{
    net1[i1,k1] = 0;
    for (int m = 0; m < akie; m++)
    {
       net1[i1,k1] = ciktia[i1,m] * wc[k1, m] + net1[i1,k1];
    }
    net1[i1,k1] = net1[i1,k1] + wce1[k1];
    cikti[i1,k1] = 1 / (1 + System.Math.Exp(-net1[i1,k1]));
}
// ITERASYONDAKİ KONTROL
    for (int ii2 = 0; ii2 < cies ; ii2++)
    {
      fark = dk[i1,ii2] - cikti[i1,ii2];
      kare = fark * fark;
      kare = System.Math.Sqrt(kare);
      toplam = toplam + kare;
    }

for (int e1 = 0; e1 < cies; e1++)
{
    ros[i1,e1] = (dk[i1, e1] - cikti[i1,e1]) * cikti[i1,e1] * (1 - cikti[i1,e1]);
    if (ii == 2) ilkb = 0;
    else
    {
```

```
            ilkb = dwce1[e1];
        }
        dwce1[e1] = epsilon * ros[i1,e1] + alfa * ilkb;
        top2 = wce1[e1] + dwce1[e1];
        wce1[e1] = top2;
    }
    for (int l1 = 0; l1 < cies; l1++)
    {
        for (int l = 0; l < akie; l++)
        {
            if (ii == 2) ilkd = 0;
            else
            {
                ilkd = dwc[l1, l];
            }
            dwc[l1, l] = epsilon * ros[i1,l1] * ciktia[i1,l] + alfa * ilkd;
            top = wc[l1, l] + dwc[l1, l];
            wc[l1, l] = top;
        }
    }
        for (int r1 = 0 ; r1 < cies ; r1++)
        {
         topwkj[i1,r1] = 0;
                for (int r = 0; r < akie; r++)
                {
                  topwkj[i1,r1] = topwkj[i1,r1] + wc[r1, r];
                }
        }
        for (int pp1 = 0; pp1 < cies; pp1++)
        {
            for (int p = 0; p < akie; p++)
            {
                roy[i1,p] = ciktia[i1,p] * (1 - ciktia[i1,p]) * ros[i1,pp1] * topwkj[i1,pp1];
```

```
                if (ii == 2) ilkg = 0;
                else
                {
                   ilkg = dwge[p];
                }
                dwge[p] = epsilon * roy[i1,p] + alfa * ilkg;
                wge[p] = dwge[p] + wge[p];
            }
        }
    for (int a = 0; a < akie; a++)
    {
        for (int b = 0; b < xsu; b++)
        {
            if (ii == 2) ilkc = 0;
            else
            {
                ilkc = dwg[a, b];
            }
            dwg[a, b] = epsilon * roy[i1,a] * x[i1, b] + alfa * ilkc;
            top1 = wg[a, b] + dwg[a, b];
            wg[a, b] = top1;
        }
    }
    ii = 3;
    i2 = i2 + 1;
    }
t = t + 1;
//mse = toplam / t;
mse = toplam / (cies * xsa);
hadez[t] = mse;
//Hatanın Belli Bir Değerden Küçük Olması Kontrolü
if (mse < hasab) goto SONYAZ;
//***eklendi***
```

```
    //if (skont == oran)
    //{
    //    hadez[haadim] = mse;
    //    skont = 0;
     //   haadim = haadim + 1;
     //   skont = skont + 1;
     //     goto YOL;
    //}
    //goto YOLCU;
//YOL:
  //  haadim = haadim + 1;
//YOLAR:
//    skont = skont + 1;
//    goto AMAC;
//    else
//    {
//YOLCU:
  //      if (skont == 0)
    //    {
      //      hadez[haadim] = mse;
        //    goto YOL;
//            haadim = haadim + 1;
//            skont = skont + 1;
            //goto ARAIKI;
       // }
        //goto YOLAR;
        //ARAIKI:

   //  }
   // HATALARI DİZİYE AKTARMA
   // if (skont == oran)
   // {
   //     hadez[haadim] = mse;
```

```
    //     haadim = haadim + 1;
    //     skont = 0;
    // }
    // else
    // {
    //     skont = skont + 1;
    // }
      //  AMAC:
    goto ILKBAS;
SONYAZ:
    //string sonsayi;
    string sonsayi = t.ToString();
    textBox12.Text = sonsayi;
    int car;
    int bol = (t / 20);
    for (int kq1 = 1; kq1 < 20; kq1++)
    {
        car = 19 * kq1;
        sondiz[kq1] = hadez[car];
    }
    //System.Console.WriteLine("kont2        : " + kont2);
    //System.Console.WriteLine();
    //System.Console.WriteLine("          Döngü Sayısı : " + t);
    //System.Console.WriteLine("Hatalar top. : " + mse);
    //System.Console.WriteLine(" AĞIRLIKLARIN GÖRÜNTÜLENMESİ  ");
    //System.Console.WriteLine("------------------------------");
    //for (int i = 0; i < akie; i++)
    //    for (int j = 0; j < xsu; j++)
    //        System.Console.WriteLine("wg[" + i + "," + j + "]  :  " + wg[i, j]);
    //for (int i = 0 ; i <akie ; i++)
    //    System.Console.WriteLine("wge["+i+"]   :   " +wge[i]);
    //for (int a1 = 0; a1 < cies; a1++)
    //{
```

```
//    for (int i = 0; i < akie; i++)
//    {
//        System.Console.WriteLine("wc[" + a1 + "," + i + "]    :   " + wc[a1, i]);
//    }
//}
//for (int a2 = 0; a2 < cies; a2++)
//{
//System.Console.WriteLine("wce1["+a2+"] : "+wce1[a2]);
//}
//  YENİ AĞIRLIKLAR İLE ARA KATMAN ÇIKIŞLARIN YENİDEN HESAPLANMASI
    for (int i3 = 0; i3 < xsa; i3++)
    {
      for (int k = 0; k < akie; k++)
      {
        netara[i3,k] = 0;
        for (int j = 0; j < xsu; j++)
        {
          netara[i3,k] = x[i3, j] * wg[k, j] + netara[i3,k];
        }
        netara[i3,k] = netara[i3,k] + wge[k];
        ciktiara[i3,k] = 1 / (1 + System.Math.Exp(-netara[i3,k]));
      }
      // YENİ AĞIRLIKLAR İLE ÇIKTI KATMANINDA Kİ ÇIKIŞIN YENİDEN HESAPLANMASI
      for (int mm1 = 0; mm1 < cies; mm1++)
      {
        net11[i3,mm1] = 0;
        for (int m = 0; m < akie; m++)
        {
          net11[i3,mm1] = ciktiara[i3,m] * wc[mm1, m] + net11[i3,mm1];
        }
        net11[i3,mm1] = net11[i3,mm1] + wce1[mm1];
```

```
            ciktison[i3,mm1] = 1 / (1 + System.Math.Exp(-net11[i3,mm1]));
        }
    }
        // System.Console.WriteLine
        //("------------------------------------------------------------");
    for (int i3 = 0; i3 < xsa; i3++)
    {
        //System.Console.Write("     "+"GİRİŞ DEĞERLERİ  : ");
        for (int i4 = 0; i4 < xsu; i4++)
        {
          //  System.Console.Write(x[i3, i4] + " ");
        }
        //System.Console.WriteLine();
        //System.Console.Write("     "+"İSTENEN DEĞERLER : ");
        for (int i6 = 0; i6 < cies; i6++)
        {
          // System.Console.Write(dk[i3, i6]+ " ") ;
        }
        //System.Console.WriteLine();
        //System.Console.Write("     "+"AĞIN ÇIKTILARI   : ");
        for (int i7 = 0; i7 < cies; i7++)
        {
          //  System.Console.Write(ciktison[i3,i7] + " ");
        }
        //System.Console.WriteLine();
        //System.Console.WriteLine
        //("------------------------------------------------------------");
    }
    // System.Console.WriteLine();
    // System.Console.Write("          Devam için  (ENTER)' a basın : ");
    // string kk1 = System.Console.ReadLine();
    // System.Console.WriteLine();
    //
```

```
System.Console.WriteLine("_____
_____");
      // System.Console.WriteLine("                    HATA BİLGİLERİ GÖRÜNTÜLEME
");
      //
System.Console.WriteLine("_____
_____");
       string hatayazılım;
       double aktar;
          for (int i3 = 1; i3 < 20; i3++)
       {
          aktar = sondiz[i3];
          hatayazılım = aktar.ToString();
          listBox1.Items.Add(hatayazılım);
        //   System.Console.WriteLine("              HATA " + i3 + " --> " + hadez[i3]);
       }

          // grafik oluşturma bölümü
       double buyh, kuch, yk, yb;
       yk = 90;
       yb = 390;
       buyh = sondiz[1];
       for (int re1 = 2; re1 < 20; re1++)
       {
          if (sondiz[re1] >= buyh) buyh = sondiz[re1];
       }
       kuch = sondiz[1];
       for (int rq1 = 2; rq1 < 20; rq1++)
       {
          if (sondiz[rq1] <= kuch) kuch = sondiz[rq1];
       }
       for (int r111 = 1; r111 < 20; r111++)
       {
```

```
        //sonh[r111]=(int)((yk-yb)/(kuch-buyh)*sondiz[r111]+yk-(yk-yb)/(kuch-buyh)*kuch);
        sonh[r111] = (int)(((yk - yb) / (buyh-kuch)) * sondiz[r111] + yb - ((yk - yb) / (buyh-kuch)) * kuch);
        // sonh[r111] = (int)((yk - yb) / (kuch - buyh) * sondiz[r111] + yb - ((yk - yb) / (kuch - buyh) * buyh));
        }

        int xsayısı = 675;

        Graphics g = this.CreateGraphics();
        g.DrawLine(new Pen(Brushes.Blue,3), 672, 86, 672, 390);
        g.DrawLine(new Pen(Brushes.Blue,3), 672, 390, 975, 390);

        dd2 = xsayısı;
        for (int d = 1; d <19; d++)
        {
           dd3 = dd2 + 15;
           yy3 = sonh[d];
           sonhd = sonh[d];
           //yy3=476 - yy3;
           yy2 = sonh[d + 1];
           sonhd1 = sonh[d + 1];
           //yy2 =476 - yy2;
           //g.DrawLine(Pens.Black, 675,90,975,390);
           g.DrawLine(new Pen(Brushes.Black,3), dd2,yy3,dd3,yy2);
           dd2 = dd2 + 15;
         }

        //g.Dispose();

//System.Console.WriteLine("_____
_____");
        //System.Console.Write("    Devam için  (ENTER)' a basın : ");
```

```
//string kk2 = System.Console.ReadLine();
//DEVADIM:
 //listBox2.Items.Add("birinci adım");
// DOSYADAN ALINAN GİRİŞ DEĞERLERİNE GÖRE ÇIKTI HESABI
double[,] xx = new double[80, xsu];
int x1satır = 0;
int x1sutun = 0;
char[] ayırıcı3 = { ' ' };
while ((Line3 = sr3.ReadLine()) != null)
{
  // listBox2.Items.Add(Line3);
   x1sutun = 0;
   string[] sayılar3 = Line3.Split(ayırıcı3);
   //System.Console.WriteLine(Line3);
   //string asd = System.Console.ReadLine();
   foreach (string ya3i in sayılar3)
   {
      hasi3 = Convert.ToDouble(ya3i);
      xx[x1satır, x1sutun] = hasi3;
      x1sutun = x1sutun + 1;
   }
   x1satır = x1satır + 1;
}
fs3.Close();
//listBox2.Items.Add("aaaaaaaaaaaaa");

for (int za11 = 0; za11 < xsu; za11++)
{
   for (int zja11 = 0; zja11 < xsa; zja11++)
   {
      xx[zja11, za11] = xx[zja11, za11] / buyukx[za11];
    //  System.Console.Write(xx[zja11, za11]+" ");
      //string ikk11 = System.Console.ReadLine();
```

```
            }
            //System.Console.WriteLine();
            //System.Console.WriteLine();
        }


        // System.Console.WriteLine();
        //  System.Console.WriteLine("                    TEST İÇİN GİRİŞ DEĞERLERİNİ OKUDU");
        xsa = x1satır;
        for (int i3 = 0; i3 < xsa; i3++)
        {
            for (int k = 0; k < akie; k++)
            {
                netarax[i3,k] = 0;
                for (int j = 0; j < xsu; j++)
                {
                    netarax[i3,k] = xx[i3, j] * wg[k, j] + netarax[i3,k];
                }
                netarax[i3,k] = netarax[i3,k] + wge[k];
                ciktiarax[i3,k] = 1 / (1 + System.Math.Exp(-netarax[i3,k]));
            }
            // YENİ AĞIRLIKLAR İLE ÇIKTI KATMANINDA Kİ ÇIKIŞIN YENİDEN HESAPLANMASI
            for (int mm1 = 0 ; mm1 < cies ; mm1++)
            {
            net11x[i3,mm1] = 0;
            for (int m = 0; m < akie; m++)
                {
                  net11x[i3,mm1] = ciktiarax[i3,m] * wc[mm1, m] + net11x[i3,mm1];
                }
            net11x[i3,mm1] = net11x[i3,mm1] + wce1[mm1];
            ciktisonx[i3,mm1] = 1 / (1 + System.Math.Exp(-net11x[i3,mm1]));
```

```
    }
}
//System.Console.WriteLine();
//System.Console.WriteLine("-----------------------------------------");
//System.Console.WriteLine("     VERİLEN GİRİŞLERE GÖRE ÇIKTI HESABI     ");
//System.Console.WriteLine("-----------------------------------------");
//for (int i3 = 0; i3 < xsa; i3++)
//{
//    System.Console.Write("     " + "GİRİŞ DEĞERLERİ  : ");
//    for (int i4 = 0; i4 < xsu; i4++)
//    {
//        System.Console.Write(x[i3, i4] + " ");
//    }
//    System.Console.WriteLine();
//    System.Console.Write("     " + "AĞIN ÇIKTISI     : ");
//    for (int i7 = 0; i7 < cies; i7++)
//    {
//        System.Console.Write(ciktison[i3,i7] + " ");
//    }
//    System.Console.WriteLine();
//    System.Console.WriteLine("-----------------------------------------");
//}
// BELİRLENMİŞ ARABANIN GÖRÜNTÜLENMESİ İŞLEMİ
//listBox2.Items.Add("zzzzzzzzzzzzzzzzzzz");
int xsyazi;
xsa = x1satır;
string harfyazı="";
string sayıyıyaz;
for (int i3 = 0; i3 < xsa; i3++)
{
    //buyuk = ciktisonx[i3, 0];
    //adres = 0;
    //for (int i7 = 1; i7 < cies; i7++)
```

```
            //{
            sayıyıyaz = ciktisonx[i3, 0].ToString();
           //  listBox2.Items.Add(sayıyıyaz);
            if (ciktisonx[i3, 0] > 0.5)
            {
               harfyazı = harf[0];
               //buyuk = ciktisonx[i3, i7];
               //adres = i7;
            }
            else
               harfyazı = harf[1];
            //}
            xsyazi = i3 + 1;
          //   System.Console.WriteLine();
          //
System.Console.WriteLine("_____________________________________________
_____________________");
          //   System.Console.WriteLine("              "+xsyazi+" .KARŞILIK GELEN HARF : " +
harf[adres]);
            //harfyazı = harf[adres];
            listBox2.Items.Add(harfyazı);
          //
System.Console.WriteLine("_____________________________________________
_____________________");
            //System.Console.WriteLine("Karşılık Gelen Harf Bulunamadı");
        }
        //CIK01:
        //System.Console.WriteLine(" Toplam iterasyon : " + t);

        //System.Console.Write("      Programdan çıkmak için (ENTER): ");
        //string kk = System.Console.ReadLine();
        //if (kk == "") goto CIKIS;
        //else
```

```
    //    goto DEVADIM;
// CIKIS:
 //    System.Console.WriteLine();
 //    System.Console.WriteLine("      *****  PROGRAMDAN  ÇIKIŞ  *****   ");
 }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
      // textBox1.Text = "C:\\dosya01.txt";
    }

   private void textBox2_TextChanged(object sender, EventArgs e)
    {
     // textBox2.Text = "C:\\dosya02.txt";
    }

   private void textBox3_TextChanged(object sender, EventArgs e)
    {
      //textBox3.Text = "C:\\dosya03.txt";
    }

    private void button3_Click(object sender, EventArgs e)
     {
        listBox1.Items.Clear();
        listBox2.Items.Clear();
        textBox12.Text = "      ";
        //*************** EKLENEN BÖLÜM

        Graphics g = this.CreateGraphics();

        dd2 = 675;

        for (int d = 1; d < 19; d++)
```

```
    {
        dd3 = dd2 + 15;
        yy3 = sonh[d];
        //yy3=476 - yy3;
        yy2 = sonh[d + 1];
        //yy2 =476 - yy2;
        //g.DrawLine(Pens.Black, 675,90,975,390);
        g.DrawLine(new Pen(Brushes.Yellow, 3), dd2, yy3, dd3, yy2);
        dd2 = dd2 + 15;
    }
     g.DrawLine(new Pen(Brushes.Blue, 3), 672, 86, 672, 390);
     g.DrawLine(new Pen(Brushes.Blue, 3), 672, 390, 975, 390);

    //************** EKLENEN BÖLÜMÜN SONU

}

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    dd2 = 675;
    g.DrawLine(new Pen(Brushes.Blue,3), 672, 86, 672, 390);
    g.DrawLine(new Pen(Brushes.Blue,3), 672, 390, 975, 390);

    for (int d = 1; d < 19; d++)
    {
        dd3 = dd2 + 15;
        yy3 = sonh[d];
        //yy3=476 - yy3;
        yy2 = sonh[d + 1];
        //yy2 =476 - yy2;
        //g.DrawLine(Pens.Black, 675,90,975,390);
```

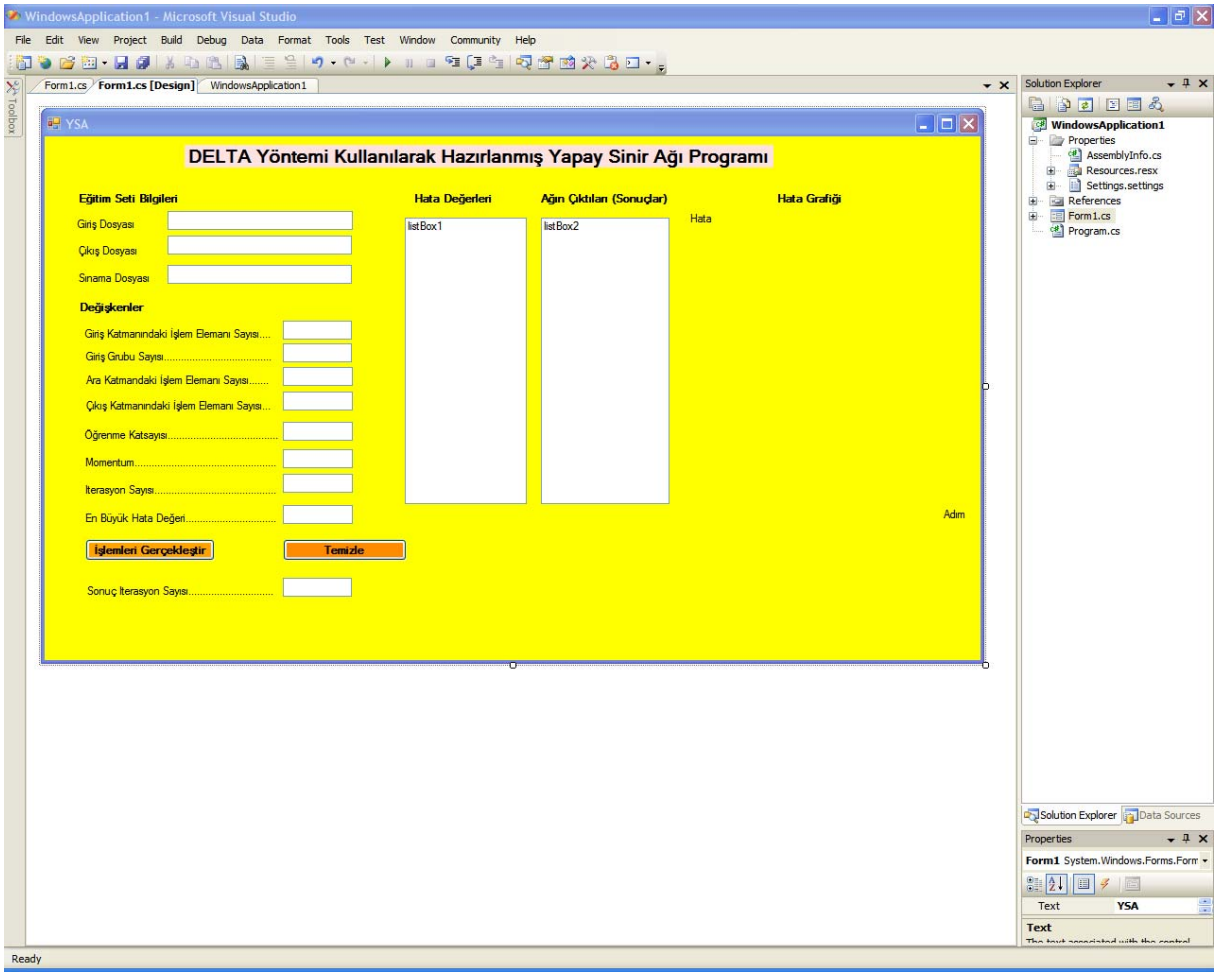
```
            g.DrawLine(new Pen(Brushes.Black,3), dd2, yy3, dd3, yy2);
            dd2 = dd2 + 15;
        }

    }
}
```

BÖLÜM 4.

4. PROGRAMIN HAZIRLANIŞ AŞAMALARI

4. 1. Programın çalıştırılmadan önceki dizaynı:


WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools Test Window Community Help
Form1.cs Form1.cs [Design] WindowsApplication1
YSA
DELTA Yöntemi Kullanılarak Hazırlanmış Yapay Sinir Ağı Programı
Eğitim Seti Bilgileri
Giriş Dosyası
Çıkış Dosyası
Sınama Dosyası
Değişkenler
Giriş Katmanındaki İşlem Elemanı Sayısı....
Giriş Grubu Sayısı.....
Ara Katmandaki İşlem Elemanı Sayısı.....
Çıkış Katmanındaki İşlem Elemanı Sayısı...
Öğrenme Katsayısı.....
Momentum.....
İterasyon Sayısı.....
En Büyük Hata Değeri.....
İşlemleri Gerçekleştir
Temizle
Sonuç İterasyon Sayısı.....
Hata Değerleri
listBox1
Ağın Çıktıları (Sonuçlar)
listBox2
Hata Grafiği
Hata
Adım
Solution Explorer
WindowsApplication1
Properties
AssemblyInfo.cs
Resources.resx
Settings.settings
References
Form1.cs
Program.cs
Solution Explorer Data Sources
Properties
Form1 System.Windows.Forms.Form
Text YSA
Text
Ready

5. BÖLÜM

5. Programın çalıştırıldıktan sonraki aşamaları ;

Eğitim seti bilgileri : Hata grafiğindeki verileri içerir.

5.1.**Giriş dosyası** : meslek011.txt uzantısındaki verileri içerir.

Meslek011.txt : Özel bir bankadan alınan gerçek kişi kredi başvuru verilerini içerir.Veri`de kişiler yaş,meslek grupları ve maaşlarına göre sınıflandırıldı.
Meslek grupları: 1 numara : İşçiler
2 numara: Muhasebe elemanları
3 numara: Satış temsilcileri
4 numara: Avukatlar
5 numara: Mimarlar
6 numara: Askeri personeller
7 numara: Emekliler
oluşturmaktadır.

5.1.2 Text içeriği;

25 1 750
27 1 750
26 1 750
29 1 750
30 1 750
24 1 750
33 1 750
24 2 1200
28 2 1200
26 2 1200
25 2 1200
29 2 1200
32 2 1200
35 2 1200
41 2 1200

24 3 650
23 3 730
26 3 730
21 3 730
22 3 730
28 3 680
22 3 680
23 3 680
23 3 700
25 3 700
45 4 3500
47 4 2100
39 4 1500
34 5 2800
38 5 3100
36 5 3500
43 5 2300
26 6 900
38 6 900
21 7 480
23 7 450
25 7 450
27 7 450

5.2 Çıkış dosyası: Meslek 021.txt uzantısındaki verileri içerir.Meslek 021 texti : Hesaplama sonucunda "0 ve 1" değerlerini belirtir."0" rakamı; kredi almazı ifade eder, "1" rakamı ise; kredi alırı ifade eder.

Grafikte ağın çıktıları (Sonuçlar) kısmında yer alır .Sınama dosyası: Meslek 031.txt uzantısını içerir.Meslek 031.texti: Daha önce hiç girilmemiş veriler girilerek, ağın eğitilip, eğitilmediği sonucunda; verilerdeki kişilerin kredi alıp alamayacağını gösterir.

Değişkenler:

- Giriş katmanlarındaki işlem eleman sayısı: 3,

- Giriş grubu sayısı: 72,
- Ara katmanlardaki işlem sayısı: 15 ,
- Çıkış katmanlarındaki işlem sayısı: 1,
- Öğrenme sayısı: 0.5,
- Momentum: 0.8.

Programda hazır olarak verilmiştir.

İterasyon Sayısı: Matematiksel bir döngüdür. Programda hazır olarak verilmemiştir. Program değerler girildikten sonra iterasyon sayısı kadar ağı eğitecektir.

12000, 13000 ve 14000 olarak denenecektir.En büyük hata değeri : Ağın eğitiminin tamamlanması için kullanılır. Hesaplanan hata, en büyük hata değerinden büyük ise, iterasyon sayısı kadar ağın eğitimine devam edilir.En büyük hata değeri grafikte hazır olarak verilmemiştir ve 0,00001, 0,00002 , 0,00003 olarak denenecektir.

BÖLÜM 6.

6. SONUÇ VE ÖNERİLER

Yapay sinir ağları kavramı kullanılarak bu tez çalışması hazırlanmıştır.Bir akış yapısı göz önüne alındığında ;

1) C# da yazılmış olan program çalıştırılır.
(Microsoft visual studio 2005`de)…

Programın çalışması aşamasında ağırlıklar rastgele üretilir.Ağ eğitilinceye kadar ağırlıklar hesaplanır ve belirlenmiş iterasyon sayısına ulaşıldığında programın çalışmasının ilk aşaması tamamlanmış olur.

2) Programın çalışmasını sağlayacak olan giriş dosyası , çıkış dosyası ve sınama dosyası verileri daha önceden hazırlanmalıdır.Giriş dosyasında ; özel bir bankadan elde edilen veriler kullanılmıştır,çıkış dosyasında sonuçlara ulaşabilmek için 1 yada 0 ile kodlama yapılmıştır.Sınama dosyasında ağın eğitilip ,eğitilmediğinin kontrolünü yapılır.

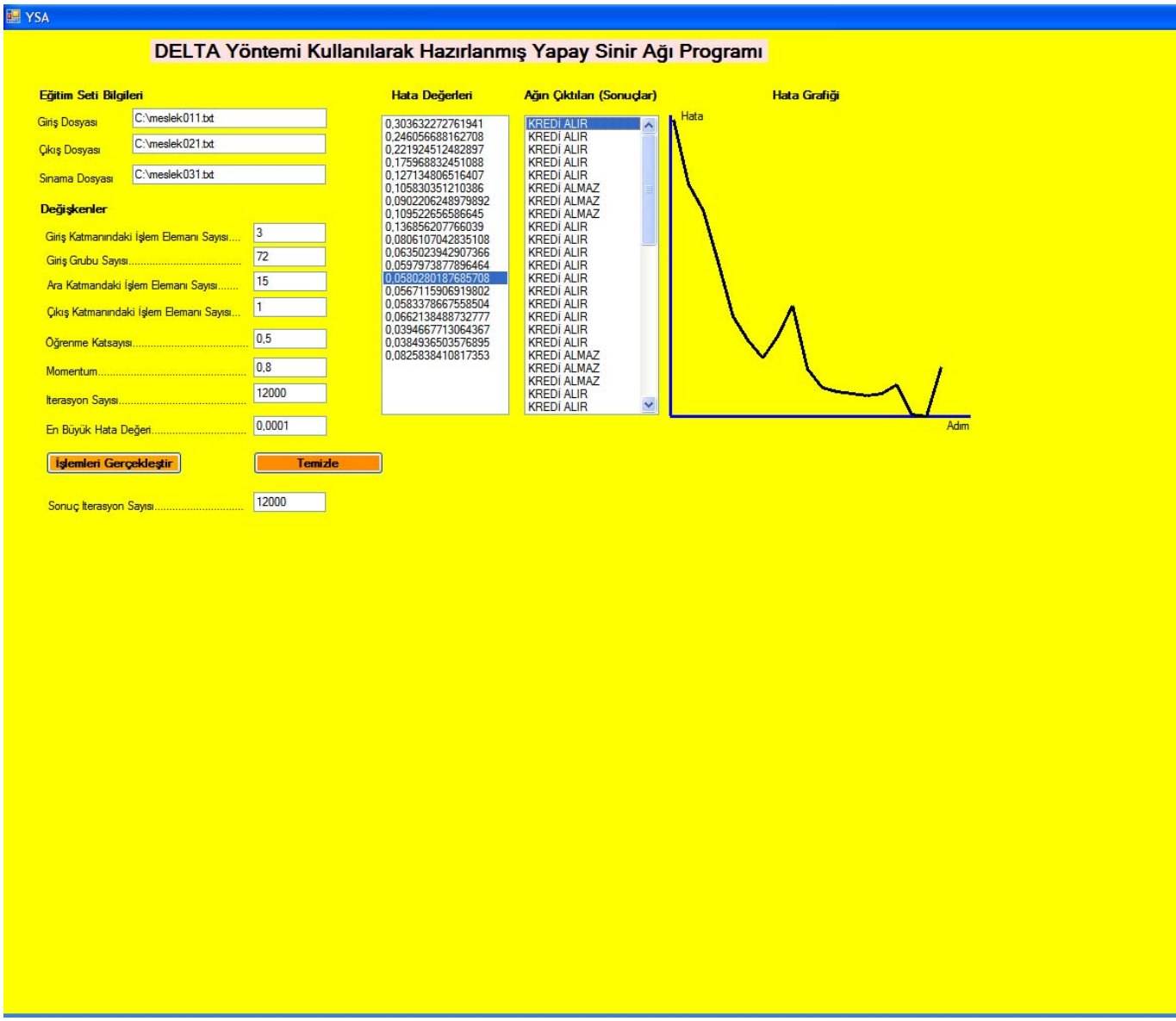
Burada birkaç uygulama alanından çok basit olarak söz edilmistir. Programa yapılacak ilave kodlar ile program daha da gelistirilerek daha kapsamlı çalışmalar da yapılabilir.
Bu durumda çok daha genis kullanım amaçlı olarak bir çok alanda
verimli ve bilimsel çalısmalar yapılabilir…

BÖLÜM 7 .

7. EKLER

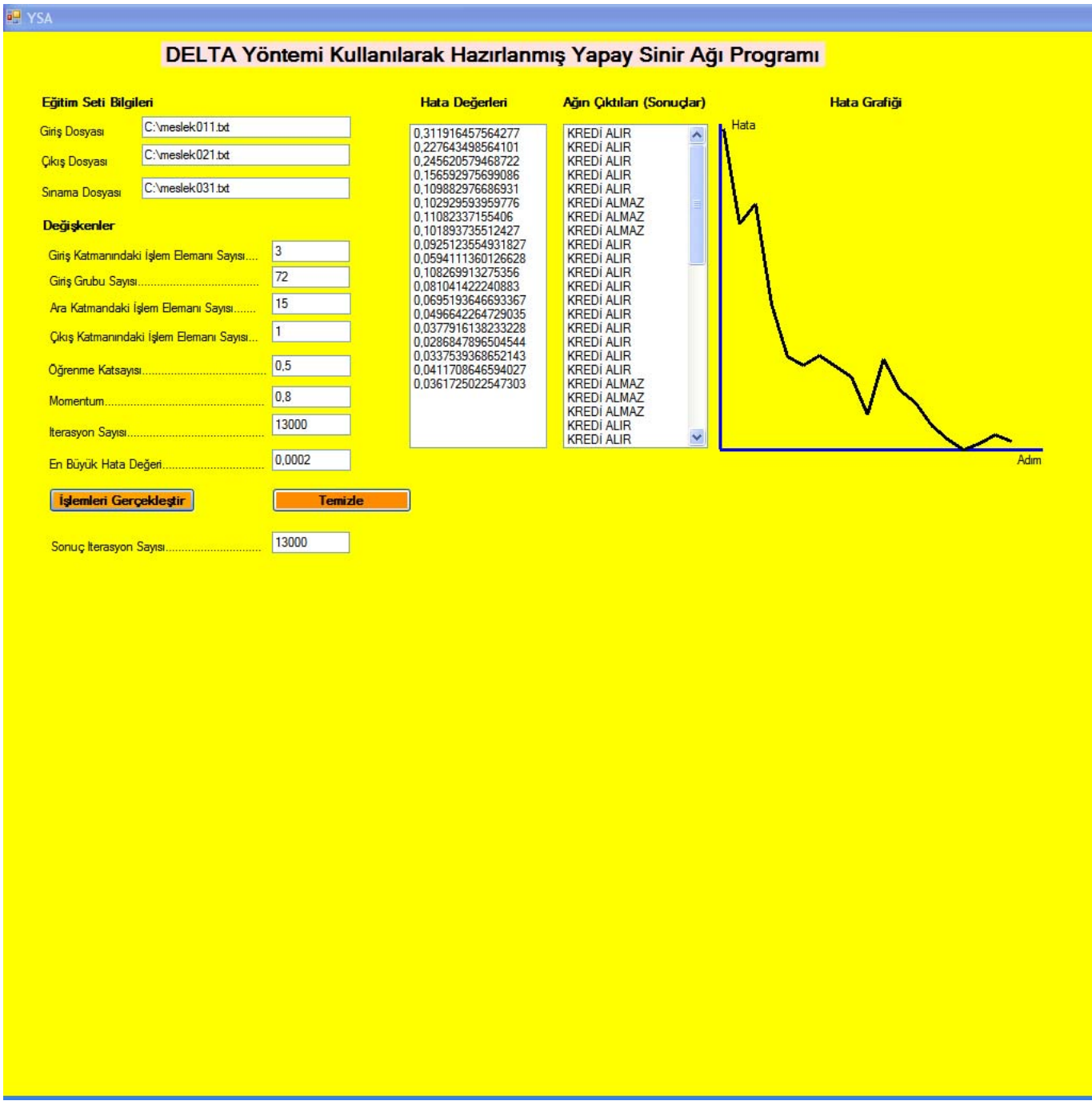
7.1. EK

İterasyon sayısı : 12000 , en büyük hata değeri :0,0001 ile programın çalışması başka bir deyiş ile ağın eğitimi ;



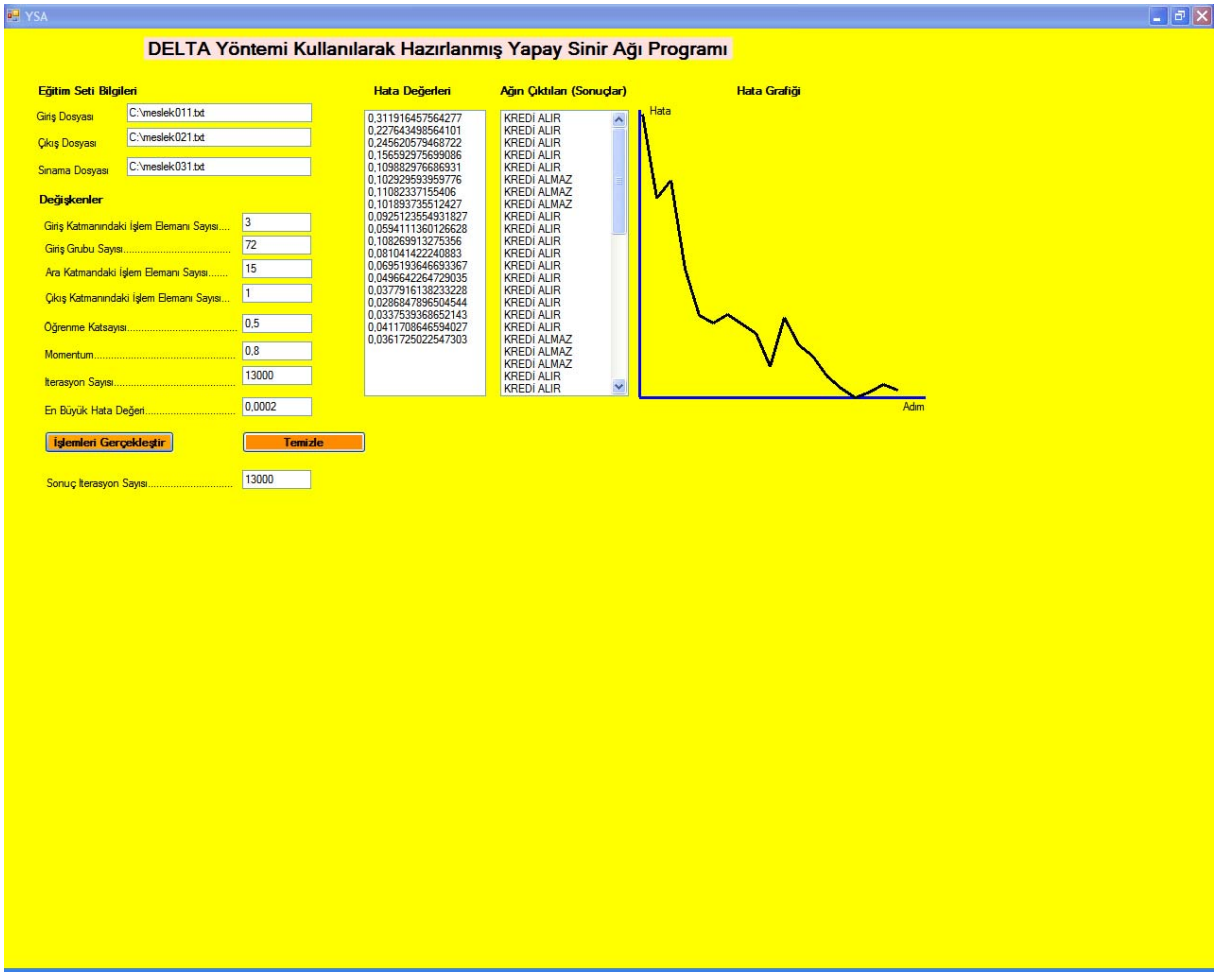
7. 2. EK

İterasyon sayısı 13000 , en büyük hata değeri 0,0002 ile programın çalışması başka bir deyiş ile ağın eğitimi ;



7. 3. EK

İterasyon sayısı 14000 , en büyük hata değeri 0,0004 ile programın çalışması başka bir deyiş ile ağın eğitimi ;



Daha hassas sonuçlara ulasmak ve çözüme daha fazla yaklasabilmek için iterasyon sayısının daha büyük alınması gerekir. Bu durumda programın çalısma süresi kuskusuz daha da uzayacaktır. Bu asamada daha hızlı CPU (Central processing unit – Merkezi islem birimi) olan bir bilgisayarın kullanılması tavsiye edilir.

BÖLÜM 8.

8. KAYNAKLAR

[1] : ALGAN, Sefer; “Her Yönüyle C#” , Pusula Yayıncılık,İstanbul, Mayıs, 2005

[2] : B. Dengiz, Sağlık Bilimlerinde Yapay Sinir Ağları ,2006

[3] : CAUDILL, Maureen “Neural Network Primar Part 1” AI Expert, Aralık 1987, sayfa 47

[4] : ELMAS ,Çetin ,Seçkin yayıncılık , 2003

[5] : KAYNAK,Oktay , Yapay sinir ağları ve uygulamaları , Boğaziçi yayıncılık, Ekim, 2008

[6] : Mehra Pankaj Wah W Benjamin, “Artifisal Neural Networks Concepts and Theory”, IEEE Computer Society Press, Washington, 1992, sayfa 45

[7] : ÖZBAYOGLU, Murat; Özellik Tanıma ve Görüntü, Mayıs, 1996

[8] : ŞAHİN,Metin; Karakter tanıma yüksek lisans tezi, Ocak, 2008

[9] : YALÇIN, Mutsak Erhan; Yapay Sinir Agları

[10] : http://www.yapay-zeka.org/modules/mydownloads/viewcat.php?cid=5

[11] : www.cmpe.boun.edu.tr/~akin/papers/EMO2.pdf

[12] : ekutup.dpt.gov.tr/ekonomi/tahmin/yurtoglh/ysa.pdf

ÖZGEÇMİŞ

Kişisel Bilgiler

Doğum tarihi : 14/12/1983
Adres : Tahsin Bey Sk Ada Apt no:2/14 Küçükyalı
Gsm telefon : (532)5241691…
Ev telefon : (216)4890335…
E-mail : onursanli@hotmail.com-onur.sanli@millenniumbank.com.tr

Deneyim

- Ekim´2006-// MILLENNIUM BANK A.Ş.
  Dağıtım Kanalları Bölümü – **Gelen aramalar yetkilisi**

Banka Müşterilerinin tüm finansal işlemler ; döviz alış-satış ya da tranferi, hazine bonusu, devlet tahvili ve diğer tüm sabit getirili menkul kıymetler ile ilgili olan finansal işlemlerin gerçekleştirdim.
Banka içindeki diğer bölümlere ; şubeler, pazarlama, satış destek ve muhasebe bölümlerine ihtiyaç duyulması halinde destek verdim.
Gerektiğinde esnek çalışma saatlerinde çalıştım.
Bankacılık ürünleri, kredi anketleri ve kobi bölümlerine gerektiğinde dertsek verdim.
Her ay düzenli olarak gerçekleştirilen çağrı merkezi eğitimlerine katılarak kendimi geliştirdim.

- MART'2006 –EKİM 2006´ MILLENNIUM BANK A.Ş.

Dağıtım Kanalları Bölümü – Satış uzmanı
Satış bölümünde yedi ay çalıştım.Hedef odaklı bir işti.Temel satış amacım daha önceden banka tarafından belirlenen müşteri adaylarıyla banka ve bankacılık ürünleri hakkında görüşerek , müşteri adaylarını banka müşterisine dönüştürmekti.
Günlük satış miktarı 10 satıştı, günlük ortalamam 10 satıştan daha fazlasını başardım.
Son satış başarım günlük 23 satıştı ve yüksek profili bir müşteri adayını bankanın müşterisi yapmış olmamdır.Bu başarımdan sonra terfi ettim.

- KASIM 2004- MART 2006' METEOR A.Ş.
  Finance Bölümü. – Muhasebe yetkilisi olarak çalıştım

- TEMMUZ 2004- AĞUSTOST 04' YAPI KREDI BANK A.Ş.

Akaretler şubesi / Stajjer olarak çalıştım

- MART`2003 – HAZİRAN`03 T.C HALİÇ ÜNİVERSİTESİ
  Kütüphane bölümü / Stajjer olarak çalıştım

Eğitim

2006-2008 – HALİÇ ÜNİVERİSTESİ
Yönetim Bilişim Sistemleri bölümünde Yüksek lisans yapmaktadır.

2001-2005 – HALİÇ ÜNİVERSİTESİ
İşletme bölümünü bitirdi.

1998-2001 – ÖZEL MARMARA LİSESİ

Yabancı dil bilgisi

İngilizce (İleri seviyededir.)

Almanca (Düşük Seviyededir)

Bilgisayar Bilgisi

Windows NT, Microsoft Office, C# ,C++

Sertifikalar

AĞUSTOS 2007` Sermaye Piyasaı Kurulu temel düzey takas sertifikası kazanmıştır.