T.C.
MİMAR SİNAN GÜZEL SANATLAR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI
YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI YAKLAŞIMI VE BİR UYGULAMA

Nuray BAŞ
DANIŞMAN: Yrd. Doç. Dr. Funda SEZGİN

İSTANBUL, 2006

ÖZET

Bu çalışmada amaç, çoklu doğrusal regresyon modeli ile yapay sinir ağları modellerinin öngörü performanslarını karşılaştırmaktır.

Bu amaç doğrultusunda ilk bölümde yapay sinir ağları genel hatlarıyla tanıtılmış olup, ikinci bölümde yapısı ayrıntılı olarak incelenmiştir. Üçüncü bölümde yapay sinir ağlarının hangi özelliklere göre sınıflandırıldığı ve temel öğrenme kurallarından bahsedilmiştir. Dördüncü bölümde ise en çok kullanılan yapay sinir ağları modelleri incelenerek, yapay sinir ağlarının istatistikle ilişkisine değinilmiştir. Uygulamaya ayrılmış olan beşinci bölümde, Türkiye'deki sanayi üretim indeksi için çoklu doğrusal regresyon analizi ile yapay sinir ağları modellerinin öngörü değerleri elde edilerek karşılaştırılmış ve sonuçlar yorumlanmıştır.

Anahtar kelimeler: Yapay Sinir Ağları, Çoklu Doğrusal Regresyon Analizi, Sanayi Üretim Indeksi

Danışman: Yrd. Doç. Dr. Funda SEZGİN, Mimar Sinan Güzel Sanatlar Üniversitesi, İstatistik Bölümü, İstatistik Anabilim Dalı

ABSTRACT

The main purpose of this study is to compare the forecast performances of multivariate linear regression model with artificial neural networks models.

In the light of this aim, artificial neural networks are introduced in general terms in the first part, and their structure is examined in detail for the next part. In the third part, the classification of artificial neural networks are mentioned in terms of their characteristics and the basic learning rules. As for the fourth part, generally used artificial neural networks models are examined and artificial neural networks' relation to statistics is given. The fifth part introduces the application, and prediction outcomes of both multivariate linear regression analysis and artifical neural networks models are compared and discussed for the Turkish Industrial Production Index.

Keywords: Artificial Neural Networks, Multivariate Linear Regression Analysis, Industrial Production Index

Advisor: Yrd. Doç. Dr. Funda SEZGİN, Mimar Sinan Fine Arts University, Department of Statistics, Statistics Section

TEŞEKKÜR

Bu çalışmanın ortaya çıkmasında göstermiş oldukları katkılarından dolayı,

Öncelikle, çalışmanın her aşamasında göstermiş olduğu ilgi; görüş ve eleştirileri için danışman hocam Yrd. Doç. Dr. Funda Sezgin'e,

Bana bu çalışmayı yapabilme fırsatını tanıyan ve yazım aşaması süresince gerçekleştirmiş olduğu gerekli düzeltmeler için Prof. Dr. Nalan Cinemre'ye,

Çalışmanın analizler kısmında benden yardımını hiç esirgemeyen Arş.Gör. Ahmet Öztopal'a,

Çalışmanın başından sonuna kadar değerli görüşleri ve eleştirileri ile bana yol göstermiş olan, yaptığı yardımlar ve manevi desteğiyle hep yanımda olan değerli dostum Arş.Gör. Elif Özge Özdamar'a,

Yazım aşamasındaki yardımları ve hiç eksik etmediği desteğiyle değerli dostum Arş.Gör. Elif Çoker'e,

Yaptıkları yardımlar ve destekleriyle Arş.Gör. Oğuz Akbilgiç, Arş.Gör.Eylem Deniz Akıncı'ya,

Her koşulda yanımda oldukları için canım annem ve babama ve yazım aşamasındaki yardımları olmasa bu çalışmanın sonlandırılamayacağını düşündüğüm kardeşim Burak'a,

Destekleriyle hep yanımda olan Bilkont Dış Ticaret çalışanı arkadaşlarım ve adlarını buraya sığdıramadığım dostlarım ve üzerimde emeği olan bütün herkese sonsuz teşekkürler...

İÇİNDEKİLER

Sayfa

ÖZET.....I
ABSTRACT.....II
TEŞEKKÜR.....III
İÇİNDEKİLER DİZİNİ..... IV
ŞEKİL DİZİNİ.....VII
TABLO DİZİNİ..... ..IX
GİRİŞ.....1
1. YAPAY SİNİR AĞLARI.....3
1.1. Yapay Zeka.....3
1.2.Yapay Sinir Ağları.....5
1.3.Yapay Sinir Ağlarının Tarihçesi.....8
1.4.Yapay Sinir Ağlarının Kullanım Alanları.....11
1.5.Yapay Sinir Ağlarının Avantaj ve Dezavantajları.....13
2.YAPAY SİNİR AĞLARININ YAPISI VE TEMEL BİLEŞENLERİ.....17
2.1.Biyolojik Sinir Ağları.....17
2.2.Yapay Sinir Hücresi.....21
2.2.1.Girdiler.....22
2.2.2.Ağırlıklar.....22
2.2.3.Toplama Fonksiyonu.....23
2.2.4.Aktivasyon Fonksiyonu.....23
2.2.4.1.Doğrusal Aktivasyon Fonksiyonu.....24
2.2.4.2.Adım Fonksiyonu.....25
2.2.4.3.Sigmoid Aktivasyon Fonksiyonu.....26
2.2.4.4.Hiperbolik Tanjant Fonksiyonu.....26
2.2.5.Hücrenin Çıktısı.....27
2.3.Yapay Sinir Ağlarının Yapısı.....27
2.4.Yapay Sinir Ağlarının Eğitimi ve Testi.....29
3.YAPAY SİNİR AĞLARININ SINIFLANDIRILMASI VE TEMEL ÖĞRENME KURALLARI.....31
3.1.Yapay Sinir Ağlarının Topolojilerine Göre Sınıflandırılması.....31

3.1.1.İleri Beslemeli Yapay Sinir Ağları.....31
3.1.2.Geri Beslemeli Yapay Sinir Ağları.....32
3.2.Yapay Sinir Ağlarının Öğrenme Metodlarına Göre Sınıflandırılması.....33
3.2.1.Danışmanlı Öğrenme.....33
3.2.2.Danışmansız Öğrenme.....34
3.3.Yapay Sinir Ağlarında Öğrenmenin Uygulamaya Göre Sınıflandırılması35
3.3.1.Çevrimiçi (On-line) Öğrenme.....35
3.3.2.Çevrimdışı (Offline) Öğrenme.....35
3.4.Temel Öğrenme Kuralları.....35
3.4.1.Hebb Kuralı.....35
3.4.2.Hopfield Kuralı..... 36
3.4.3.Delta Kuralı…….....36
3.4.4.Kohonen Kuralı.....……...36
4.YAPAY SİNİR AĞLARI MODELLERİ VE YAPAY SİNİR AĞLARI İLE İSTATİSTİK İLİŞKİSİ.....37
4.1.Yapay Sinir Ağlarının Tasarımı…….....37
4.2.Yapay Sinir Ağları Modelleri.....…….....39
4.2.1.İlk Yapay Sinir Ağları (Tek Katmanlı Yapay Sinir Ağları).....…….....39
4.2.1.1.Perceptron Modeli.....42
4.2.1.2.ADALINE Modeli.....…….....45
4.2.2. Çok Katmanlı Perceptron ve İleri Beslemeli Geri Yayılım Yapay Sinir Ağı.....47
4.2.2.1.Çok Katmanlı Perceptron.....47
4.2.2.2.İleri Beslemeli Geri Yayılım Yapay Sinir Ağı ve Geri Yayılım Algoritması.....50
4.2.3.LVQ Ağı.....58
4.2.4.Kohonen Ağı.....64
4.2.5.ART Ağları.....67
4.2.5.1.ART1 Ağı.....70
4.2.6.Hopfield Ağı.....75
4.2.7.Jordan Ağı.....79

4.2.8.Elman Ağı.....80
4.3.Yapay Sinir Ağları ile İstatistik İlişkisi.....84
5. UYGULAMA.....88
5.1.Verilerin Tanıtılması.....88
5.2.Çoklu Regresyon Modeli.....91
5.3.Yapay Sinir Ağları Modeli.....93
5.4 Model Karşılaştırması.....100
5.5 Sonuç.....104
KAYNAKLAR DİZİNİ.....105
EK.....110

ŞEKİL DİZİNİ

Şekil 2.1. Biyolojik Sinir Hücresinin Yapısı.....18
Şekil 2.2. Yapay Sinir Hücresinin Yapısı.....22
Şekil 2.3. Doğrusal Aktivasyon Fonksiyonu'nun Şekilsel Gösterimi.....25
Şekil 2.4. Adım Fonksiyonu'nun Şekilsel Gösterimi.....25
Şekil 2.5. Sigmoid Aktivasyon Fonksiyonu'nun Şekilsel Gösterimi..... 26
Şekil 2.6. Hiperbolik Tanjant Fonksiyonu'nun Şekilsel Gösterimi.....27
Şekil 2.7. Tipik 3 Katmanlı İleri Beslemeli Yapay Sinir Ağı Mimarisi.....28
Şekil 4.1. İki girdi ve bir çıktıdan oluşan en basit tek katmanlı yapay sinir ağı modeli.....40
Şekil 4.2. Ağırlıkların ve sınıfları birbirinden ayıran doğrunun geometrik gösterimi 41
Şekil 4.3. Bir perceptron yapısı.....43
Şekil 4.4. Bir ADALINE yapısı.....45
Şekil 4.5. Çok Katmanlı Perceptron Modeli.....49
Şekil 4.6. İleri Beslemeli Geri Yayılım Yapay Sinir Ağı Yapısı.....50
Şekil 4.7. LVQ ağının topolojik yapısı.....59
Şekil 4.8. Kohonen Ağı.....65
Şekil 4.9. ART ağlarının genel yapısı.....68
Şekil 4.10. ART1 modelinin geometrik gösterimi.....71
Şekil 4.11. Hopfield ağının yapısı.....76
Şekil 4.12. Jordan Ağı.....80
Şekil 4.13. Elman Ağı.....81
Şekil 4.14. Doğrusal Perceptron=Çok Değişkenli Çoklu Doğrusal Regresyon.....84
Şekil 4.15. Doğrusal Olmayan Perceptron = Lojistik Regresyon.....85
Şekil 4.16. ADALINE = Doğrusal Diskriminant Fonksiyonu.....85
Şekil 4.17. Çok Katmanlı Perceptron=Doğrusal Olmayan Basit Regresyon.86
Şekil 5.1. Sanayi Üretim İndeksi serisi (SUI).....89
Şekil 5.2. Toptan Eşya Fiyat İndeksi serisi (TEFE).....89
Şekil 5.3. Dolar Alış Kuru serisi (DOLAR).....90

Şekil 5.4. Çoklu Regresyon Modeli'nin Hata Paylarına İlişkin Grafik.....93
Şekil 5.5. Yapay Sinir Ağları Modeli'nin Mimarisi..... ..95
Şekil 5.6. Eğitim sürecinde Ortalama Mutlak Hata değerlerinin değişimi.....97
Şekil 5.7 Yapay Sinir Ağları Modeli'nin Tahminleri ve Gerçekleşen Değerler.....98
Şekil 5.8. YSA-1 Modeli'nin eğitim sürecinde Ortalama Mutlak Hata değerlerinin değişimi.....99
Şekil 5.9. YSA-1 Modeli'nin Tahminleri ve Gerçekleşen Değerler.....100

TABLO DİZİNİ

Tablo 2.1. Biyolojik Sinir Ağı ve Yapay Sinir Ağının Karşılaştırılması.....21
Tablo 4.1. Kullanım amaçlarına göre yapay sinir ağları.....38
Tablo 5.1. Öngörü Hatasının Doğruluk Ölçütleri.....103
Tablo 5.2. Öngörü Hatasının Yüzde Doğruluk Ölçütleri.....103

GİRİŞ

Yapay sinir ağları, özellikle son 20 yılda büyük gelişme gösteren bir çalışma disiplinidir. Bu çalışmanın amacı da, yapay sinir ağlarını tanıtmak ve yapay sinir ağlarını istatistiksel bir yöntem olan çoklu regresyon ile karşılaştırmaktır.

Çalışma beş bölümden oluşmaktadır. Çalışmanın ilk bölümünde yapay sinir ağları genel olarak açıklanmıştır. Bu kapsamda ilk olarak yapay sinir ağlarının da içine girdiği bir yöntemler bütünü olan yapay zekadan bahsedilmiş, daha sonra yapay sinir ağlarının tanımı verilmiş ve tarihçesi incelenmiştir. Yapay sinir ağlarının daha iyi anlaşılması açısından kullanım alanları tanıtılmış ve avantajları ve dezavantajlarından bahsedilerek bölüm sonlandırılmıştır.

İkinci bölümde, yapay sinir ağlarının yapısının anlaşılabilmesi için öncelikle biyolojik sinir ağları açıklanmış ve buna bağlı olarak yapay sinir hücresinin yapısı ayrıntılı olarak incelenmiştir. Daha sonra yapay sinir ağının yapısından kısaca bahsedilmiştir. Bölüm, yapay sinir ağlarının eğitimi ve test edilmesinin genel olarak açıklanmasıyla sonlandırılmıştır.

Üçüncü bölümde, yapay sinir ağlarının çeşitli özelliklerine göre sınıflandırılmalarının üzerinde durulmuş ve bu sınıflar genel olarak tanıtılmıştır. Bölüm, yapay sinir ağlarındaki temel öğrenme kurallarının kısaca açıklanmasıyla sonlandırılmıştır.

Dördüncü bölümde, öncelikle yapay sinir ağlarının tasarım aşaması anlatılmıştır. Daha sonra, literatürde en çok rastlanan ve belli sınıfların tipik özelliklerini gösteren yapay sinir ağları modelleri ayrıntılı bir şekilde ele alınmıştır. Son olarak da yapay sinir ağları ve istatistik arasındaki ilişki genel hatlarıyla anlatılmıştır.

Son bölüm olan beşinci bölümde ise öncelikle çoklu doğrusal regresyon analizi uygulaması gerçekleştirilmiş, daha sonra yapay sinir ağları modelinin uygulamasına geçilmiştir. Türkiye'deki sanayi üretim indeksi için çoklu doğrusal regresyon analizi ve yapay sinir ağları modellerinin öngörü performansları karşılaştırılmıştır. Sonuçlar doğrultusunda yapay sinir ağları modelinin çoklu doğrusal regresyon analizi modeline göre üstün olduğu kanıtlanmıştır.

BİRİNCİ BÖLÜM
YAPAY SİNİR AĞLARI

1.1 Yapay Zeka

Yapay zeka (Artificial intelligence), en kısa şekilde insan beyninin çalışma sistemini anlamak ve bu sistemi taklit ederek benzer etkinlikleri gerçekleştirebilecek bilgisayar işlemlerini oluşturmaya çalışmak olarak tanımlanabilir. Yapay zeka kullanılarak yapılan çalışmaların temel amacı, insanların doğuştan sahip oldukları zekayı modelleyerek bazı fonksiyonlarını bilgisayarlara ve bilgisayar denetimli makinelere kazandırabilmektir.

Anlama ve kavrama yeteneği olan zekanın pek çok tanımı bulunmaktadır. En basit tanımıyla zeka, "*Bir insanın belli bir duruma uyma, koşullara göre etkinlik araçları seçme yetkinliği; kavrama gücü; ayırt etme yetisidir*" (Büyük Larousse, 12726).

Zeka; eğitim, öğretim, bilgi, birikim ve deneyimlerle geliştirilebilen dinamik bir yapıya sahiptir. İlk kez karşılaşılan ya da ani olarak gelişen bir olaya uyum sağlayabilme, anlama, öğrenme, analiz etme, beş duyu, dikkat ve düşüncenin yoğunlaştırılması zeka ile gerçekleştirilebilmektedir (Elmas, 2003).

Zekanın yukarıda verilen tanımları ışığında yapay zeka çalışmalarının, bilgisayarlara ve bilgisayar denetimli makinelere, zekanın en temel özellikleri olan anlama, genelleme ve geçmiş deneyimlerden öğrenme gibi yeteneklerini kazandırmayı hedeflediği söylenebilir.

Günümüzde iş dünyasından kamu sektörüne, çevre ve sağlık organizasyonlarından askeri sistemlere kadar hemen hemen her alanda bilgisayar yaygın olarak kullanılmaktadır. Önceleri sadece elektronik veri transferi ve karmaşık hesaplamaları gerçekleştirmek için geliştirilen

bilgisayarlara zaman içinde verileri filtreleyerek özetleyebilme ve eldeki bilgileri kullanarak olaylar hakkında yorumlar yapabilme nitelikleri de kazandırılmıştır. Günümüzde ise bilgisayarlar hem olaylar hakkında karar verebilmekte hem de olaylar arasındaki ilişkileri öğrenebilmekte, matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler, sezgisel yöntemler kullanılarak çözülebilmektedir. Bilgisayarları bu özellikler ile donatan ve bu yeteneklerinin gelişmesini sağlayan çalışmaların hepsi yapay zeka çalışmaları olarak adlandırılır. Bir problemin çözümünü sağlayan formül ya da algoritmalar geliştirilmiş ise geleneksel bilgisayar sistemleri problemi çözmek için yeterlidir. Fakat önemli olan problemin çözümünün elde edilemediği durumlarda bilgisayarlara problemleri çözdürmektir. Yapay zeka bu görevi üstlenmektedir (Öztemel, 2003).

İnsan beyni ve işleyişinin insanoğlunun ilgisini çekmesi çok eskilere dayansa da, bu konuda pratiğe yönelik bilimsel çalışmalar 19. yüzyılın sonlarında yapılmaya başlanmıştır. 1956 yılında Dartmouth College'de yapılan ve pek çok bilim adamının katıldığı iki aylık bir konferansta yapay zekanın temellerini oluşturan çalışmalar bilim adamlarınca tartışılmış ve yapay zeka ismi ilk kez kullanılmıştır.

Yapay zekanın yaygın bir şekilde araştırma ve uygulama alanı bulan belli başlı alt dalları şöyle özetlenebilir (Elmas, 2003);

Uzman Sistemler, temelde insan düşüncelerini geliştirmek amacıyla bilgisayar tarafından işlenen yazılımlardır. Uzman sistemler geliştirilirken, belirli bir konuda uzmanlaşmış olan insanların bilgi ve deneyimlerinin bilgisayara aktarılması amaçlanır.

Bulanık Mantık, bulanık küme teorisine dayanan matematiksel bir disiplindir. Bulanık mantık; bilgisayarın, sadece bir durumun ya da karşıtının olabileceğini kabul eden mantığının yerine, insan mantığındaki gibi ara

değerleri de hesaba katarak karar vermesini sağlar. Örneğin bir bilgisayar için sadece uzun-kısa ya da sıcak-soğuk olabilen bir durum, bulanık mantık kullanılarak uzun-ortadan uzun-orta-ortadan kısa-kısa ya da sıcak-ılık-az soğuk-soğuk-çok soğuk gibi ara değerlere sahip olabilir.

Genetik algoritma, Darwin'in evrim kuramı "*doğada en iyinin yaşaması"* kuralından esinlenerek oluşturulan, bir veri kümesinden özel bir veriyi bulmak için kullanılan bir arama yöntemidir. Genetik algoritmalar, geleneksel yöntemlerle çözümü zor veya imkansız olan problemleri sanal olarak evrimden geçirerek en iyi çözümü elde etmeyi amaçlamaktadır.

Yapay sinir ağları, insan beyninde bulunan sinir ağlarının işleyişini modelleyen tekniklerdir. Yapay sinir ağlarına, ilgili olaya ait bilgiler örnekler üzerinde eğitilerek verilmektedir. Böylelikle, örnekler sayesinde açığa çıkarılmış özellikler üzerinde çeşitli genelleştirilmeler yapılarak, daha sonra ortaya çıkacak ya da o ana kadar hiç rastlanmamış olaylara da çözümler üretilmektedir.

Yapay zeka uygulama alanı olarak bilgisayarlar ve bilgisayar destekli makineleri kullandığı için bilgisayar bilimlerinin bir alt dalı gibi görünmesine rağmen, gelişebilmesi için konusu insan olan diğer pek çok bilim dalının çalışmalarından da faydalanmaktadır. Bu nedenle biyoloji, fizik, matematik, psikoloji, felsefe ve bilgisayar bilimleri yapay zeka çalışmalarına önemli katkılar sağlamaktadır. Böylece çalışma biçimi pek çok bilim dalının katkısıyla çözülmeye çalışılan beynin fonksiyonları, bilgisayarlar ve robotlar kullanılarak modellenmeye çalışılmaktadır.

1.2 Yapay Sinir Ağları

Günümüzde bilgi işleme, büyük ölçüde bilgisayarlarla gerçekleştirilmektedir. Bilgisayarlar çok büyük boyuttaki veri/bilgiyi insanlara göre çok daha hızlı bir şekilde işleyebilmektedir. Aynı şekilde sayısal işlemlerde bilgisayarlar, insan beyniyle kıyaslanamayacak kadar hızlıdır. Ancak ses ve görüntü algılama ve

işleme, plan yapma ve öğrenme gibi işlemleri insan beyni çok kısa sürede gerçekleştirebilirken, bilgisayarlar bu işlemleri teorik olarak sonsuz zaman ve sonsuz hafıza kullanarak, çok az miktarda ve çok daha az başarıyla sonuçlandırır.

Ayrıca insan beyni, bozulduğunda bütün beyin faaliyetinin durmasına yol açacak merkezi bir işlemciye sahip değildir. Bir bilgisayar programında tek bir komutun bile yanlış olması programın çalışmaması veya tamamen yanlış bir sonuç vermesine neden olurken, insan beynindeki bir sinir hücresinin (nöronun) faaliyetini yitirmesi beynin bütün faaliyetini veya bazı fonksiyonlarını tamamen yitirmesine neden olmaz. Bunun nedeni bilgisayarların seri, beynin elemanlarının ise paralel işlemci mantığı ile çalışmasından kaynaklanır, yani bilgisayarlarda tek bir merkezi işlemci her hareketi sırasıyla gerçekleştirmektedir. Beyinde ise her bir sinir hücresi, büyük bir problemin bir parçasıyla ilgilenen birbirine paralel bağlanmış bir işlemci eleman yapısındadır. Sinir hücreleri kendi başlarına yavaş olmalarına karşın sistem paralel çalışmasından dolayı hızlıdır.

Beynin ve bilgisayarların birbirlerinden farklı olarak üstün oldukları bu özellikler ve son zamanlarda beynin çalışma sistemi üzerine edinilen bilgilerin artması, insan beynini modelleyerek çalışan bilgisayar araştırmalarını da arttırmıştır.

Bu yaklaşımın sonucu, yapılan çalışmalarda beynin temel işlemci elemanları olan sinir hücrelerinin oluşturduğu ağ yapısının matematiksel modeli oluşturulmaya çalışılmıştır. Beynin bütün davranışlarını modelleyebilmek için fiziksel bileşenlerinin doğru olarak modellenmesi gerektiği düşüncesi ile çeşitli yapay sinir hücreleri ve ağ modelleri geliştirilmiştir. Böylece, “Yapay Sinir Ağları” denen günümüz bilgisayarlarının algoritmik hesaplama yöntemlerinden farklı bir çalışma disiplini ortaya çıkmıştır (Saraç, 2004).

Pek çok tanımı bulunan yapay sinir ağları basit olarak, beyindeki sinirlerin çalışmasından esinlenilerek sistemlere öğrenme, hatırlama, bilgiler arasında ilişkiler oluşturma gibi yetenekleri kazandırmayı amaçlayan bilgi işleme algoritmalarıdır. Literatürde çok rastlanan ve Teuvo Kohonen'e (1987) ait tanımda ise yapay sinir ağları "Biyolojik sinir sisteminde olduğu gibi gerçek yaşam nesneleriyle etkileşmeyi amaçlayan basit elemanların ve onların hiyerarşik düzenlemelerinin paralel, içice bağlantılı ağları" şeklinde bahsedilmektedir (Taşgetiren, 2005). Daha geniş bir tanım olan Haykin (1999)'in tanımına göre ise " Bir sinir ağı, basit işlem birimlerinden oluşan, deneyimsel bilgileri biriktirmeye yönelik doğal bir eğilimi olan ve bunların kullanılmasını sağlayan yoğun bir şekilde paralel dağıtılmış bir işlemcidir. Bu işlemci iki şekilde beyin ile benzerlik gösterir:

1. Bilgi, ağ tarafından bir öğrenme süreciyle çevreden elde edilir.
2. Elde edilen bilgileri biriktirmek için sinaptik ağırlıklar olarak da bilinen nöronlar arası bağlantı güçleri kullanılır" (Yurtoğlu, 2005).

Yapay sinir ağları ayrıca sinirsel ağlar (neural nets), bağlantılı ağlar (connectionist networks), yapay sinir sistemleri (artificial neural systems), paralel dağıtılmış ağlar (paralel distributed networks) olarak da adlandırılmaktadır.

Yapay sinir ağlarında bilgi işleme, sinir hücresi adı verilen bir çok basit elemanda gerçekleşmektedir. Bir sinir hücresinden gelen sinyal, sinir hücreleri arasındaki ilişkiyi sağlayan bağlantılarla iletilmektedir. Her bir bağlantının bir ağırlık değeri vardır ve girdiyi önemine göre ağırlıklandırarak geçişini sağlamaktadır. Sinir ağı içindeki her bir sinir hücresine ayrı bir aktivasyon fonksiyonu uygulanmaktadır (bu fonksiyon genelde doğrusal olmayan bir fonksiyondur) ve bu fonksiyonun çıkış değeri sayesinde sinir hücresinin çıkış sinyali hesaplanmaktadır. Herhangi bir yapay sinir ağı; sinir hücreleri arasındaki bağlantının bir modeli yani mimarisi, bağlantılar arasındaki ağırlıkların hesaplanması (bu hesaplama, öğrenme kuralı ya da

öğrenme algoritması olarak da adlandırılır) ve aktivasyon fonksiyonu ile tanımlanabilir (Aydın, 2002).

1.3 Yapay Sinir Ağlarının Tarihçesi

İnsan beyni hakkındaki çalışmalar binlerce yıl öncesine dayanmaktadır. Beynin fonksiyonları hakkında bilgi veren ilk eser 1890 yılında William James tarafından yayımlanmıştır. Nöroloji ve psikoloji alanlarında yapılan çalışmaların sinir hücrelerinin yapısı ve sinir ağlarının çalışma biçimlerindeki bilinmeyenleri aydınlatmada katedilen yol, yapay sinir ağlarının gelişimine önemli katkılar yapmıştır.

1940'dan önce Helmholtz, Pavlov, Poincare gibi bazı bilim adamlarının yapay sinir ağları kavramı üzerinde çalıştıkları bilinmektedir, ancak bu çalışmaların matematiksel bir tabanı yoktur. 1940'lı yıllarda McCulloch ve Pitts, Hebb, Rosenblatt gibi bilim adamlarının bu konudaki çalışmaları yapay sinir ağları çalışmalarının mühendislik alanına kayması ve günümüzdeki yapay sinir ağlarının temelinin oluşmasına neden olmuştur.

Modellemeye yönelik araştırmaların başlangıcı, aynı zamanda bir dönüm noktası olarak kabul edilen çalışma, 1943 yılında bir sinir doktoru olan Warren McCulloch ve bir matematikçi olan Walter Pitts tarafından Bulletin of Mathematical Biophysics dergisinde "A Logical Calculus of the Ideas Immanent in Nervous Activity" adı ile yayımlanan ve bir sinir hücresinin ilk matematiksel modelininin öne sürüldüğü çalışmalarıdır.

Donald Hebb 1949 yılında yayımlanan "The Organization of Behavior" adlı kitabı ile öğrenebilen ve uyum sağlayabilen sinir ağları modeli için temel olacak Hebb Kuralı'nı ortaya koymuştur. Hebb kuralı; sinir ağlarının bağlantı sayısının değiştirilebilmesi durumunda ağın öğrenebileceğini ileri sürer (Mehrotra, Mohan, Ranka, 1996).

1958 yılında Frank Rosenblatt tarafından geliştirilen “Perceptron” yapay sinir ağlarındaki çalışmaları hızlandırmıştır. Perceptron tek katmanlı, eğitilebilen ve tek çıkışa sahip olan bir yapay sinir ağıdır. Perceptronun daha sonra geliştirilecek ve yapay sinir ağları çalışmalarında devrim kabul edilen çok katmanlı yapay sinir ağlarının temelini oluşturması açısından tarihsel önemi vardır.

1959 yılında Bernand Widrow ve öğrencisi Marcian Hoff, ADALINE (ADaptive LInear NEuron) modelini ortaya atmışlardır. Yapay sinir ağlarının mühendislik uygulamalarında kullanılmaya başlanması için atılan ilk adımlardan biri olan bu model, 1970’lerin sonlarında ortaya çıkan ve ADALINE modelinin çok katmanlı hali olan MADALINE’nın temelini oluşturur ve telefon hatlarında oluşan yankıları yok eden bir sistem olarak kullanılması ile de gerçek dünya sorunlarına uyarlanmış ilk yapay sinir ağı ünvanını alır (Öztemel, 2003).

Yapay zeka çalışmalarının o devirde önde gelen isimleri Marvin Minsky ve Seymour Papert 1969 yılında yayımladıkları “Perceptrons” adlı kitaplarında perceptronun bilimsel bir değerinin olmadığını, doğrusal olmayan problemlere çözüm üretemediğini vurgulamışlar ve buna ispat olarak XOR[1] (exclusive-or) probleminin perceptron ile çözülememesini örnek göstermişler.

Bu olay yapay sinir ağlarının problem çözümlerinde uygulanabilirliğine olan inancın zayıflamasına, bu çalışmalara destek veren ve ABD Pentagon’da yer alan DARPA isimli organizasyonun desteğini çekmesine yol açmıştır. Yapay sinir ağları çalışmaları durma noktasına gelmiş, ancak yapay zeka alanındaki diğer yöntemlerin desteklenmesi artmıştır. Ancak Shun-ichi Amari, James Anderson, Michael Arbib, Kunihiko Fukushima, Stephen Grossberg, Teuvo Kohonen, Arthur Little, Cristoph von der Malsburg ve Paul Werbos gibi bazı

[1] XOR probleminin çözümü Ek’de verilmektedir.

bilim adamları yapay sinir ağları alanındaki çalışmalarını sürdürmüşlerdir (Gülseçen, 1993).

Seksenli yıllar, teknolojinin gelişmesiyle beraber yapay sinir ağları araştırmalarının hızla yaygınlaşmaya başladığı dönemdir. 1982 ve 1984 yıllarında John Hopfield tarafından yayımlanan çalışmalar, farklı tipteki yapay sinir ağlarının matematiksel modellerini üretmiş ve özellikle geleneksel bilgisayar programlama ile çözülemeyen problemlerin yapay sinir ağları ile çözülebileceğini göstermiştir. Bu çalışmaların pratikte uygulanabilirliği yapay sinir ağlarına olan ilginin yeniden artmasına neden olmuştur.

1982 yılında Kohonen "özörgütlemeli nitelik haritaları (selforganizing feature maps-SOM)" konusundaki çalışmasını yayımlamıştır. Bu çalışmadan yola çıkarak Hinton ve çalışma arkadaşları Boltzman makinesini geliştirmiştir.

1986'da Rumelhart ve McClelland karmaşık ve çok katmanlı ağlar için geri yayılım öğrenme algoritmasını (backpropagation) ortaya koymuşlardır. Bu çalışma ile Parker ve Werbos'un yaptıkları araştırmalar, tek katmanlı ağlarla çözülemeyen XOR probleminin çözülmesini sağlamıştır (Mehrotra, Mohan, Ranka, 1996).

1988'de Broomhead ve Lowe, Radyal Tabanlı Fonksiyonlar (Radial Basis Functions-RBF) modelini geliştirmişlerdir.

Doksanlı yılların başlarından itibaren sayısız çalışma ve uygulamalar geliştirilmiş, özellikle yapay sinir ağlarını eğitmek için gerekli süreleri kısaltmak, yeni ve daha verimli öğrenme algoritmaları geliştirmek, zamana bağlı olarak değişen modellere karşılık verebilen ağlar ve silikon sinir ağları geliştirmek, yapılan araştırmaların en önemli amaçlarını oluşturmaktadır.

1.4 Yapay Sinir Ağlarının Kullanım Alanları

Günümüzde yapay sinir ağları eksik bilgilerle çalışabilme ve normal olmayan verilere çözüm üretebilme yeteneklerinden dolayı pek çok alanda kullanılabilmektedir. Doğrusal olmayan, çok boyutlu, gürültülü, karmaşık, kesin olmayan, eksik, kusurlu, hata olasılığı yüksek veriler ve problemlerin çözümü için özellikle bir matematiksel model ve algoritmanın bulunmaması durumlarında yaygın halde yapay sinir ağları uygulamaları yapılabilmekte ve başarılı sonuçlar elde edilmektedir. Bu amaçla geliştirilmiş ağlar genel olarak şu fonksiyonları yerine getirmektedir:

- Probabilistik fonksiyon kestirimleri
- Sınıflandırma
- İlişkilendirme ve örüntü tanımlama (pattern recognition engine)
- Zaman serileri analizleri
- Sinyal filtreleme
- Veri sıkıştırma
- Örüntü tanıma
- Doğrusal olmayan sinyal işleme
- Doğrusal olmayan sistem modelleme
- Optimizasyon
- Zeki ve doğrusal olmayan kontrol (Öztemel, 2003).

Yapay sinir ağları bu teorik uygulamaların ötesinde günlük hayatta kullanılan finansal konular, mühendislik ve tıp bilimi gibi bir çok alanda uygulanabilmektedir. Evimizdeki aletlerden cep telefonlarına kadar günlük hayatımızda yapay sinir ağları uygulamaları görmek mümkündür. Bu uygulamalar çok çeşitli olup, en önemlileri aşağıda açıklanmıştır.

Arıza analizi ve tespiti: Bir sistemin, cihazın ya da elemanın düzenli (doğru) çalışma şeklini öğrenen bir yapay sinir ağı yardımıyla bu sistemlerde meydana gelebilecek arızaların tanımlanma olanağı olmaktadır. Bu amaçla

yapay sinir ağları, elektrik makinelerinin, uçakların ya da bileşenlerinin, entegre devrelerin v.b. arıza analizinde kullanılmaktadır (Saraç, 2004).

Finansal uygulamalar: Makro ekonomik tahminler, kredi kartı hilelerinin tespiti, kredi kartı kurumlarında iflas tahminleri, banka kredilerinin değerlendirilmesi, emlak kredilerinin yönetilmesi, döviz kuru tahminleri, risk analizleri gibi örneklerde uygulama alanı bulmaktadır (Öztemel, 2003).

Tıp uygulamaları: EEG ve ECG gibi tıbbi sinyallerin analizi, kanserli hücrelerin analizi, protez tasarımı, transplantasyon zamanlarının optimizasyonu ve hastanelerde giderlerin optimizasyonu gibi konularda uygulanmaktadır (Saraç, 2004).

Savunma sanayi uygulamaları: Silahların otomasyonu ve hedef izleme, nesneleri/görüntüleri ayırma ve tanıma, yeni algılayıcı tasarımı ve gürültü önleme gibi alanlarda kullanılmaktadır (Saraç, 2004).

Haberleşme uygulamaları: Görüntü ve veri sıkıştırma, otomatik bilgi sunma servisleri, konuşmaların gerçek zamanda çevirisi gibi alanlarda uygulanmaktadır (Saraç, 2004).

Üretim uygulamaları: Üretim sistemlerinin optimizasyonu, ürün analizi ve tasarımı, ürünlerin kalite analizi ve kontrolü, planlama ve yönetim analizi gibi alanlarda kullanılmaktadır (Saraç, 2004).

Otomasyon ve kontrol uygulamaları: Uçaklarda otomatik pilot sistemi otomasyonu, ulaşım araçlarında otomatik yol bulma ve gösterme, robot sistemlerin kontrolü, doğrusal olmayan sistem modelleme ve kontrolü, elektrikli sürücü sistemlerin kontrolü gibi alanlarda uygulanmaktadır (Saraç, 2004).

Farklı alanlardaki uygulamalar incelendiğinde yapay sinir ağlarının genel olarak şu fonksiyonları gerçekleştirmek amacıyla kullanıldıkları görülmektedir (Öztemel, 2003).

Tahmin: Bu amaçla kullanılan yapay sinir ağları, ağa sunulan bilgilerden yararlanarak karşılık gelen çıktı değerlerini tahmin etmektedirler.

Sınıflandırma: Bu amaçla kullanılan yapay sinir ağları, kendilerine verilen bilgileri kategorize etmek görevini üstlenmektedirler.

Veri ilişkilendirme: Bu amaçla eğitilen ağlar, ağa sunulan verilerin hatalı olup olmadıklarını belirlemektedirler. Öğrendikleri bilgiler ile eksik bilgileri tamamlamaktadırlar.

Veri filtreleme: Bu amaçla eğitilen ağlar, birçok veri arasından uygun verileri belirleme görevini yerine getirmektedir.

Tanıma ve eşleştirme: Değişik şekil ve örüntülerin tanınması, eksik, karmaşık, belirsiz bilgilerin işlenerek eşleştirme ve tanıma fonksiyonlarını gerçekleştirebilmektedir.

Teşhis: Bu amaçla geliştirilen ağlar, sistemlerin olumsuzluklarının ortaya konulması ve problemlerin belirlenmesi işlemini yerine getirmektedirler.

Yorumlama: Bir olay hakkında toplanan örneklerden elde edilen ve eğitim sonucu oluşturulan bilgileri kullanarak yeri olayların yorumlanması işlemleri bu kapsamda düşünülmektedir.

1.5 Yapay Sinir Ağlarının Avantaj ve Dezavantajları

Yapay sinir ağları kullanılarak yapılan modeller, biyolojik sinir ağlarının çalışma biçimlerinden esinlenerek oluşturuldukları için biyolojik sinir ağlarının

üstünlüklerine sahiptir. Bu üstünlüklerin bir kısmı yapay sinir ağları fikrinin de ortaya çıkış sebepleridir ve şu şekilde özetlenebilir;

Doğrusal olmayan yapı: Yapay sinir ağlarının temel işlem elemanı olan hücre doğrusal değildir dolayısıyla bu hücrelerin birleşmesinden meydana gelen yapay sinir ağları da doğrusal olmamaktadır. Özellikle ekonomik veriler gibi yapıları gereği genellikle doğrusal olmayan veriler tahmin zorlukları nedeniyle genellikle doğrusal yöntemlerle analiz edilmektedir. Bu durum doğrusal olmayan bir veri setiyle çalışılması durumunda yanlış sonuçlara neden olabilmektedir. Yapay sinir ağlarının doğrusal olmayan yapıları da dikkate alması onu önemli kılmaktadır.

Paralellik: Günümüzde kullanılan bilgi işleme yöntemleri genelde seri işlemlerden oluşmaktadır. Yapay sinir ağları ise paralel işlemci kullanmaktadır. Seri işlemcilerde herhangi bir birimin yavaş olması tüm sistemi yavaşlatırken, yapay sinir ağlarında yavaş bir birimin etkisi az olmaktadır. Bu durum yapay sinir ağlarının daha hızlı ve güvenilir olması sonuçlarını doğurmaktadır.

Öğrenme: Geleneksel veri işleme yöntemlerinin çoğunluğunda bir problemin çözülebilmesi için probleme uygun bir algoritma geliştirilmesi ve programlama yolu ile hesaplama yapılması gerekmektedir. Bu nedenle tanımlı olmayan bir problemin çözümü yapılamaz. Yapay sinir ağları ise en önemli özelliği olan ve insan beyninin çalışma sistemi olan öğrenme yöntemini kullanmaktadır. Yapay sinir ağlarında öğrenme, özellikleri verilen örnekler yoluyla edinilmektedir ve örnekleri kullanarak probleme ilişkin genelleme yapabilecek yeteneğe ulaşmaktadır. Bu özelliği sayesinde geleneksel yöntemler için karmaşık olan sorunlara çözüm getirebilmektedirler. Ayrıca, insanların kolayca yapabildiği ancak geleneksel yöntemlerin uygulanamadığı basit işlemler için de uygun olmaktadırlar.

Hafıza: Yapay sinir ağlarında bilgi, işlemci elemanlar (yapay sinir hücreleri) arasındaki ağırlıklı bağlantılarda saklanmaktadır. Yani bilgi ağa dağıtılmış durumdadır ve ağın bütünü, öğrendiği olayın tamamını göstermektedir. Bu nedenle yapay sinir ağlarının dağıtılmış bellekte bilgi saklayabildikleri söylenebilir.

Yerel işlem ve esneklik: Yapay sinir ağları, geleneksel işlemcilerden farklı şekilde işlem yaparlar. Geleneksel işlemcilerde, tek bir merkezi işlemci eleman her hareketi sırasıyla gerçekleştirirler. Yapay sinir ağları, her biri büyük bir problemin bir parçası ile ilgilenen çok sayıda basit işlemci elemanlardan oluşmaları ve bağlantı ağırlıklarının ayarlanabilmesi gibi özelliklerinden dolayı önemli derecede esnek bir yapıya sahiptirler. Bu esnek yapı sayesinde ağın bir kısmının zarar görmesi modelde sadece performans düşüklüğü yaratır modelin işlevini tamamen yitirmesi sözkonusu olmaz. Ayrıca, toplam işlem yükünü paylaşan işlemci elemanların birbirleri arasındaki yoğun bağlantı yapısı sinirsel hesaplamanın temel güç kaynağıdır. Bu işlem yapısı sayesinde, yapay sinir ağları yöntemi en karmaşık problemlere bile uygulanabilmekte ve tatminkar sonuçlar sağlayabilmektedir (Yurtoğlu, 2005).

Genelleme: Yapay sinir ağları yine öğrenme yeteneği sayesinde bilinen örnekleri kullanarak daha önce karşılaşılmamış durumlarda genelleme yapabilmektedir. Yani, hatalı (gürültülü) veya kayıp veriler için çözüm üretebilmektedir. Yapay sinir ağları, tanımlanmamış veriler hakkında karar verirken genelleme yapabildikleri için iyi birer örüntü tanımlayıcısı ve güçlü sınıflandırıcılardır (robust classifier) (Yurtoğlu, 2005).

Uyarlanabilirlik: Yapay sinir ağları, ilgilendiği problemdeki değişikliklere göre ağırlıklarını ayarlayabilmektedir. Yani, belirli bir problemi çözmek amacıyla eğitilen yapay sinir ağı, problemdeki değişimlere göre tekrar eğitilebilmekte, değişimler devamlı ise gerçek zamanda da eğitime devam edilebilmektedir. Bu özelliği ile yapay sinir ağları, uyarlamalı örnek tanıma, sinyal işleme,

sistem tanımlama ve denetim gibi alanlarda etkin olarak kullanılabilmektedir (Saraç, 2004).

Eksik verilerle çalışma: Yapay sinir ağları, geleneksel sistemlerin aksine kendileri eğitildikten sonra eksik bilgiler ile çalışabilmekte ve gelen yeni örneklerde eksik bilgi olmasına rağmen sonuç üretebilmektedirler. Yapay sinir ağlarının eksik bilgiler ile çalışması, performanslarının düşeceği anlamına gelmemektedir. Performansın düşmesi eksik olan bilginin önemine bağlıdır. Hangi bilginin önemli olduğunu ağ kendisi eğitim sırasında öğrenmektedir. Tasarımcıların (kullanıcıların) bu konuda bir fikri yoktur. Ağın performansı düşük ise, eksik olan bilginin önemli olduğu, aksi durumda önemli olmadığı sonucuna varılır

Sınırsız sayıda değişken ve parametre kullanma: Yapay sinir ağları sınırsız sayıda değişken ve parametre ile çalışabilmektedir. Bu sayede mükemmele yakın öngörü doğruluğu ile genel çözümler sağlanabilmektedir (Yurtoğlu, 2005).

Yapay sinir ağlarının yukarıda belirtilen avantajları geniş uygulama alanları bulmalarını sağlamaktadır. Ancak yapay sinir ağlarının gözönünde bulundurulması gereken bazı dezavantajları da bulunmaktadır. Bunlar arasında en önemlisi, geniş veri seti gereksinimidir. Yapay sinir ağlarının eğitilebilmesine ve test edilebilmesine yetecek genişlikte veri setine ihtiyaç duyulur. Bununla birlikte, yeterli veri seti genişliği için kesin bir kriter yoktur, bu kriter uygulamaya bağlı olarak değişir. Dezavantaj sayılabilecek diğer bir nokta ise basit olarak görülebilecek modelleme yapılarına rağmen uygulamanın zor ve karmaşık olabilmesidir. Bazı durumlarda, bir yakınsama sağlamak bile imkansız olabilmektedir fakat bu durum da uygulama alanına bağlıdır ve genellikle çok karmaşık problemlerde ortaya çıkmaktadır (Yurtoğlu, 2005).

İKİNCİ BÖLÜM
YAPAY SİNİR AĞLARININ YAPISI VE TEMEL BİLEŞENLERİ

2.1 Biyolojik Sinir Ağları

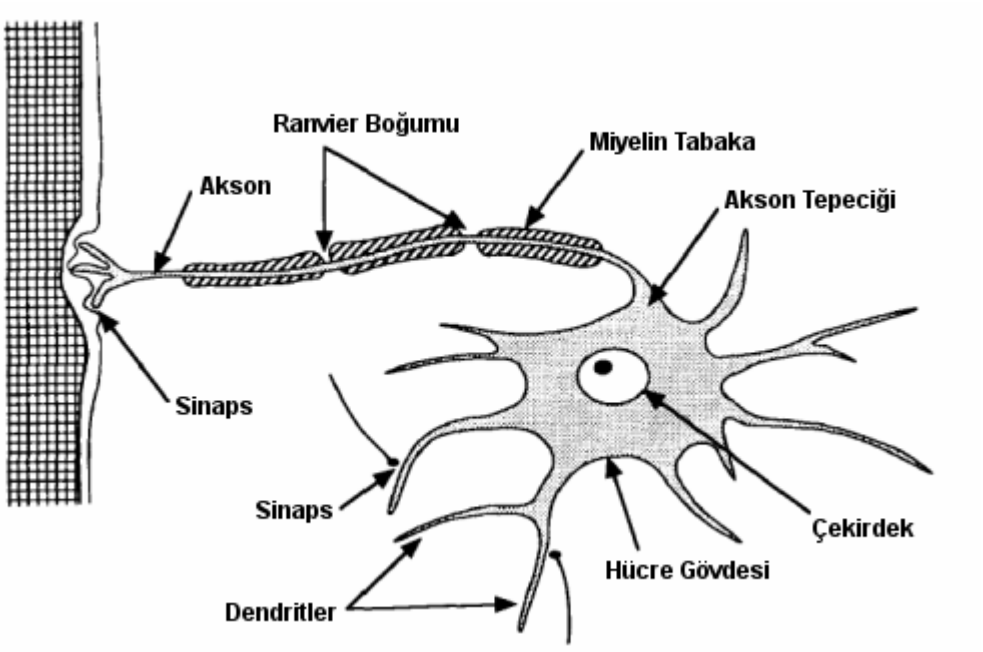
Beyinin çalışma sistemi, hala tam olarak çözülememiş bir sırdır. Özellikle, vücudun diğer organlarındaki hücrelerden farklı olarak yenilenebilme özelliği olmayan beynin, insana hatırlama, düşünme ve her hareketinde geçmiş deneyimlere başvurma yeteneğini sağlayan en temel elemanı olan sinir hücrelerinin çalışma sistemi hala gizliliğini büyük ölçüde korumaktadır. İnsan beyninde ortalama 100 milyar sinir hücresi ve her bir sinir hücresinin 1.000 ile 10.000 arası komşu bağlantısı vardır. Beynin gücü bu çok sayıdaki sinir hücresi ve aralarındaki sonsuz sayıdaki bağlantılardan, genetik yapıları ile öğrenme yeteneklerinden kaynaklanmaktadır (Anderson and McNeill, 1992).

Biyolojik sinir sistemi, merkezinde sürekli olarak bilgiyi alan, yorumlayan ve uygun bir karar üreten beynin bulunduğu üç katmanlı bir sistem olarak açıklanmaktadır. Bu katmanlar; çevreden gelen girdileri elektriksel sinyallere dönüştürerek beyine ileten Alıcı Sinirler (Receptor), beynin ürettiği elektriksel sinyalleri çıktı olarak uygun tepkilere dönüştüren Tepki Sinirleri ile alıcı ve tepki sinirleri arasında ileri ve geri besleme yaparak uygun tepkiler üreten Merkezi Sinir Ağı olarak sıralanır (Saraç, 2004).

Yetişkin bir insanda ortalama olarak 1.5 kg. ağırlıkta olan beynin çalışma frekansı 100 Hz'dir. Yenilenme özelliği olmayan beynin gelişmesi ve ağırlık kazanması, sinir hücrelerinin büyümesi ve aralarında yeni bağlantılar kurulmasından kaynaklanmaktadır (Kulkarni, 1994).

Bir sinir hücresi; hücre gövdesi (soma), bu gövdeyi çevreleyen ve saç teline benzeyen dendritler (dendrite) ile gövdeye bağlı bir kuyruk şeklindeki aksondan (axon) meydana gelmektedir. Temel olarak sinir hücrelerinde dendritler üzerinden girişler (sinaptik uçlara gelen sinirsel akımlar) alınmakta,

soma tarafından girişler genelde doğrusal olmayan bir şekilde işlenmektedir. Sinir hücresindeki sinyalleri taşıyan uzun bir sinirsel bağlantı halindeki akson ise, işlenen girişleri çıkışa aktarmaktadır. Aksonlar, girişlerin iletilme hızını artıran bir çeşit yalıtım maddesi olan miyelin tabaka (myelin sheath) ile kaplıdır. Miyelin tabakanın üzerinde birkaç milimetrede bir yer alan ve girişleri periyodik olarak yeniden üretmeye yarayan ranvier boğumu (ranvier node) yer almaktadır. Akson dendrit bağlantısı ise sinaps (synapse) olarak adlandırılmaktadır. Sinaps sinir hücreleri arasındaki elektrokimyasal bağlantıyı sağlamaktadır.



Şekil 2.1: Biyolojik Sinir Hücresinin Yapısı (Freeman ve Skapura, 1991)

Sinir hücreleri arasındaki iletişimi sağlayan elektrokimyasal bağlantının çalışma biçimi oldukça karmaşıktır. Sinir hücresi zarı, hücre içi ve dışı sıvılarda bulunan bazı iyonlara karşı geçirgen bir yapıya sahiptir. Sinir hücresi zarı, hücre içi ve dışı sıvılar arasında yer alan potansiyel sodyum-potasyum farkını "metabolik pompa" görevi görerek korumaktadır. Bu pompa yardımıyla sinir hücresi zarı, hücre içindeki sodyum iyonlarının dışarı, hücre

dışındaki potasyum iyonlarının içeri girmesini sağlamaktadır. Sıvı içerisinde ayrıca klor iyonu ve negatif (-) organik iyonlar yer almaktadır. Hücre zarının geçirgen yapısı klor iyonlarının geçişine izin vermekte ancak hücre zarının geçirgenliğine göre oldukça büyük olan organik iyonların geçişine olanak tanımamaktadır. Bu durum, hücre içinde organik iyonların fazlalaşmasına ve negatif etkileriyle klor iyonlarını nötrleştirmelerine yol açmaktadır. Bunun sonucunda hücre dışı sıvıda klor iyonu konsantrasyonu artmaktadır. Hücre zarı potasyum iyonlarını sodyum iyonlarına oranla daha kolay geçirmektedir. Kimyasal olarak hücre dışına çıkmaya eğilimli potasyum iyonları negatif organik iyonların yüksek çekim gücü sayesinde hücre içinde korunmaktadır. Bu pompa sistemi sayesinde daha fazla potasyum ve organik iyonlar hücre içinde, daha fazla klor ve sodyum iyonlar hücre dışında kalmakta ve bu bir denge durumuna ulaşılmasını sağlamaktadır. Oluşan bu denge ile hücre içi sıvının daha negatif olması sonucunda, hücre zarında 70-100 milivolt düzeyinde bir potansiyel enerji kalmaktadır.

Bu şekilde polarize duruma geçen sinir hücresi, kendisine yakın diğer komşu hücreler tarafından uyarıldığında bu durum tersine dönmekte ve hücre zarının geçirgenlik derecesi değişerek hücre içerisine hızla sodyum iyonları dolmaktadır. Böylece hücre pozitif (+) yüklenmektedir. Bu duruma hücre zarının depolarize olması denilmektedir. Hücre zarının bir noktasında meydana gelen depolarizasyon akson tepeciğinden (axon hillock) başlayarak aksonun sonuna kadar zincirleme bir reaksiyona neden olmakta ve sinir akımının hücre gövdesinden aksonun en ucunda yer alan ve diğer hücrelerin dendritleri ile ilişki kuran sinaptik uçlara kadar aktarılmasını sağlamaktadır.

Sinaptik uçlara gelen sinirsel akım, kalsiyum iyonlarının etkisiyle zarın geçirgenliğini değiştirmekte ve sinaptik keseciklerdeki (synaptic vesicles) biyokimyasal maddelerden oluşan sinirsel aktarıcılar olan nörotransmiterleri (neurotransmitters) etkilemektedir. Etkilenen sinirsel nörotransmiterler, 1 / 2.000.000 santimetre genişliğindeki sinaptik aralıktan (synaptic gap) geçerek, diğer sinir hücresinin dendritlerine ulaşmaktadır. Sinaptik aktarıcılar

bu tür sinaptik birleşmeler sırasında, diğer sinir hücresinde uyarıcı (excitatory) veya engelleyici (inhibitory) olmak üzere iki farklı durumun gerçekleşmesine neden olmaktadır. Alıcı durumunda bulunan sinir hücresindeki kimyasal faaliyet, bazı iyon örneklerinin daha kolay geçmesini sağlayacak şekilde zarın geçirgenliğini değiştirmektedir. Uyarıcı durumda, hücre zarının geçirgenliğinin değişmesi sonucunda hücre içerisine daha kolay akan pozitif iyonlar, sinir hücresinin depolarize olmasını sağlayıp, bu hücrede de sinirsel bir akımın başlamasına yol açmaktadır. Engelleyici durumda ise, negatif iyonlar sinir hücresini daha da polarize ederek (hyperpolarization), sinirsel akımın durmasını sağlamaktadır. Ancak sinir sistemindeki sinir hücrelerinin her birisi sadece bir tek sinir tarafından uyarılmamaktadır. Bazen bir sinir hücresine, birden fazla sinir hücresi tarafından aynı anda yapılan etkiler, hem uyarıcı hem de engelleyici olabilmektedir. Alıcı sinir hücresi, uyarıcı ve engelleyici nitelikteki bu etkileri akson tepeciğinde toplar. Elde edilen bu toplam, belirli bir eşik değerinden (threshold) büyükse, sinirsel akımın bu sinir hücresinde de devamı sağlanmaktadır (Akpınar, 1993).

Bir sinir hücresinin bu çok basitleştirilmiş elektrokimyasal davranışlarının yanısıra onun başka bir sinir hücresiyle olan sinaptik bağlantısında henüz tam olarak bilinmeyen birçok biyokimyasal süreç bulunmaktadır.

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 sayesinde son derece karmaşık, doğrusal olmayan ve paralel dağılmış bir bilgi işleme sistemi olarak tanımlanabilir (Kulkarni, 1994).

2.2 Yapay Sinir Hücresi

Biyolojik sinir ağlarının sinir hücrelerinden oluşması gibi, yapay sinir ağları da yapay sinir hücrelerinden meydana gelmektedir. Yapay sinir hücreleri ayrıca düğüm (node), birim (unit) veya işlemci eleman (processing unit) olarak da adlandırılmaktadır. Bir yapay sinir ağı, birbiriyle bağlantılı çok sayıda yapay sinir hücresinden meydana gelmektedir. Yapay sinir hücreleri biyolojik sinir hücrelerinin basit bir modelidir.

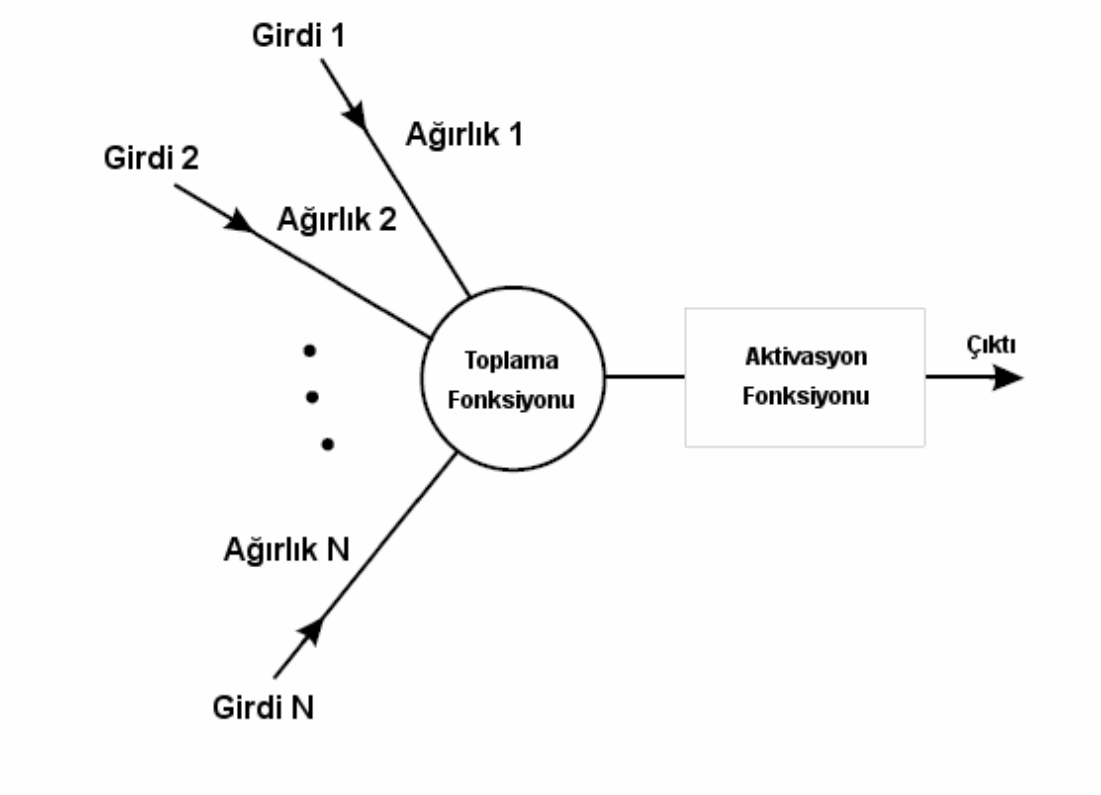
Tablo 2.1 Biyolojik Sinir Ağı ve Yapay Sinir Ağının Karşılaştırılması

Biyolojik Sinir Ağı	Yapay Sinir Ağı
Sinir Sistemi	Sinirsel Hesaplama Sistemi
Sinir Hücresi (Nöron)	İşlemci Eleman (Yapay Sinir Hücresi, Düğüm)
Sinaps	İşlemci elemanlar arasındaki bağlantı ağırlıkları
Dendrit	Toplama Fonksiyonu
Hücre Gövdesi	Aktivasyon Fonksiyonu
Akson	İşlemci Eleman çıktısı

Yapay sinir ağlarının içinde bulunan tüm sinir hücreleri bir veya birden fazla girdi alırlar ve tek bir çıktı verirler. Bu çıktı yapay sinir ağının dışına verilen bir çıktı olabileceği gibi başka bir yapay sinir hücresine girdi olarak da verilebilir. Bir yapay sinir hücresi genel olarak beş temel bileşenden oluşmaktadır.

- Girdiler
- Ağırlıklar
- Toplama fonksiyonu
- Aktivasyon fonksiyonu

- Çıktı



Şekil 2.2: Yapay Sinir Hücresinin Yapısı (Gurney, 1996)

2.2.1 Girdiler

Girdiler, bir yapay sinir hücresine gelen bilgilerdir. Bu bilgiler dış ortamlardan ya da diğer sinir hücrelerinden gelebilir. Dış ortamlardan gelen bilgiler, ağın öğrenmesi istenen örnekler tarafından belirlenmektedir.

2.2.2 Ağırlıklar

Ağırlıklar, gelen bilgilerin hücre üzerindeki etkisini belirleyen değerlerdir. Bilgiler, bağlantılar üzerindeki ağırlıklar üzerinden hücreye girmekte ve ağırlıklar yapay sinirde girdi olarak kullanılacak değerlerin göreceli kuvvetini (matematiksel katsayısını) göstermektedirler. Yapay sinir ağı içinde girdilerin hücreler arasında iletimini sağlayan tüm bağlantıların farklı ağırlık değerleri

bulunur. Böylelikle ağırlıklar her işlemci elemanın her girdisi üzerinde etki yapmış olur. Ağırlıklar değişken veya sabit değerler olabilirler.

2.2.3 Toplama Fonksiyonu

Toplama fonksiyonu, hücreye gelen net girdiyi hesaplayan fonksiyondur ve genellikle girişlerin kendi ağırlıklarıyla çarpımının toplamı;

$$s = \sum x_i w_i \tag{2.1}$$

biçiminde ifade edilir. Yapay sinir ağının yapısına göre toplama fonksiyonu, maksimum, minimum, çarpım veya çeşitli normalizasyon işlemlerinden birisi olarak da ifade edilebilir. Bir problem için en uygun toplama fonksiyonu çeşidini bulmak için herhangi bir formül yoktur. Toplama fonksiyonu genellikle deneme yanılma yoluyla bulunmaktadır. Ayrıca bir yapay sinir ağındaki bütün işlemci elemanların aynı toplama fonksiyonuna sahip olması gibi bir zorunluluk da yoktur. Bazen aynı yapay sinir ağı içindeki işlemci elemanların bazıları aynı toplama fonksiyonunu, diğerleri ise başka fonksiyonları kullanabilirler. Bu tamamen tasarımcının kendi kararına bağlıdır (Öztemel, 2003).

2.2.4 Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu, toplama fonksiyonundan gelen girdiyi işleyerek yapay sinir hücresinin çıkışını belirler. Transfer fonksiyonu olarak da adlandırılan aktivasyon fonksiyonu çeşitli tiplerde ve genellikle doğrusal olmayan bir fonksiyondur. Doğrusal fonksiyonların tercih edilmemesinin nedeni, doğrusal fonksiyonlarda girdi ile çıktının doğru orantılı olmasıdır. Bu durum ilk yapay sinir ağları denemelerinin başarısızlıkla sonuçlanmasının temel nedenidir (Minsky and Papert, 1969).

Uygun aktivasyon fonksiyonunun seçimi tasarımcının farklı fonksiyonları denemeleri sonucunda belirlenmektedir. Ancak çok katmanlı perceptron gibi bazı modeller aktivasyon fonksiyonunun, türevi alınabilir bir fonksiyon

olmasını şart koşmaktadır. Ayrıca fonksiyonun seçimi, yapay sinir ağının verilerine ve neyi öğrenmesinin istendiğine de bağlıdır. Aktivasyon fonksiyonu olarak en çok kullanılanlar sigmoid fonksiyon ve hiperbolik tanjant fonksiyonlarıdır.

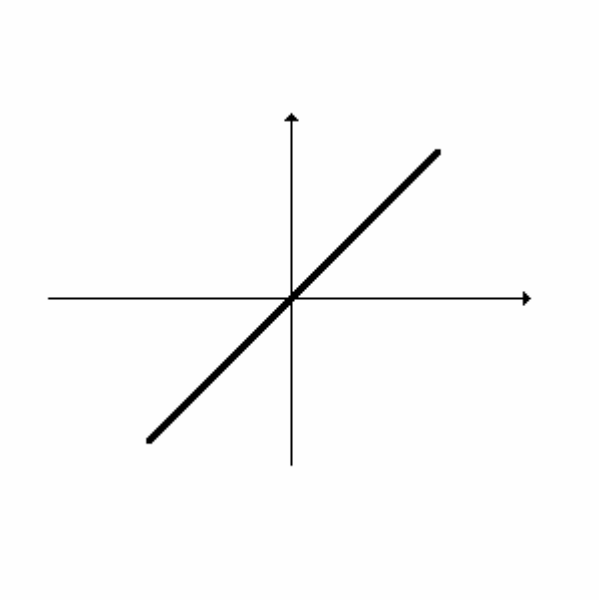
Aktivasyon fonksiyonu, toplama fonksiyonundan gelen girdiyi dönüştürerek istenilen değerler arasında sınırlandırmaktadır. Bu değerler kullanılan aktivasyon fonksiyonun tipine göre genellikle [0,1] veya [-1,1] arasındadır. Bu değer aktivasyonun fonksiyonunun, dolayısıyla yapay sinir hücresinin çıktı değeri olarak ya dış ortama ya da girdi olarak başka bir yapay sinir hücresine iletilmektedir.

Aktivasyon fonksiyonu işlemi öncesinde, sisteme tekdüze (uniform) dağılmış bir rassal hata eklenebilmektedir. Bu rassal hatanın kaynağı ve büyüklüğü sistemin öğrenme sürecinde belirlenir ve sebebi, insan beyninin işlevinin, içinde bulunduğu ortamın koşullarından etkilenmesidir. Örneğin ortamın soğuk/sıcak olmasından insan beyni etkilenmektedir. Bu nedenle yapay sinir ağları literatüründe rassal hata ekleme işlemi “sıcaklık (temperature)” olarak da adlandırılmaktadır. Ancak günümüzde rassal hata işlevi tam olarak kullanılmamakta ve hala bir araştırma süreci içinde bulunmaktadır. Ayrıca bazı yapay sinir ağlarında, aktivasyon fonksiyonunun çıktısı üzerinde başka işlemler, ölçeklendirme ve sınırlandırma yapılabilmektedir (Yurtoğlu, 2005).

Yapay sinir hücrelerinde yaygın olarak kullanılan çeşitli aktivasyon fonksiyonları aşağıda verilmiştir.

2.2.4.1 Doğrusal Aktivasyon Fonksiyonu

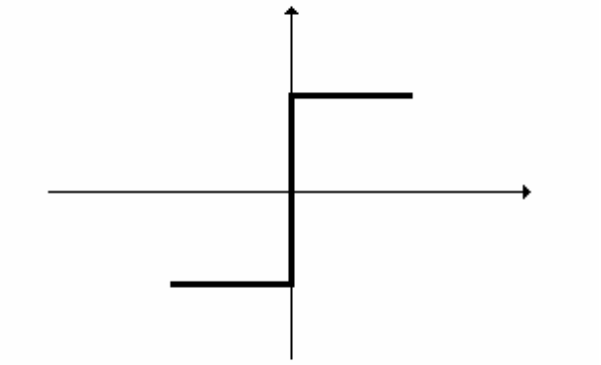
Doğrusal problemlerin çözümünde kullanılan bu fonksiyon, gelen net girdileri doğrudan hücre çıkışı olarak vermektedir. Matematiksel olarak $F(s) = s$ şeklinde tanımlanmaktadır.



Şekil 2.3: Doğrusal Aktivasyon Fonksiyonu'nun Şekilsel Gösterimi

2.2.4.2 Adım Fonksiyonu

Gelen net girdi değerinin belirlenen bir eşik değerinin altında ya da üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerlerini almaktadır.

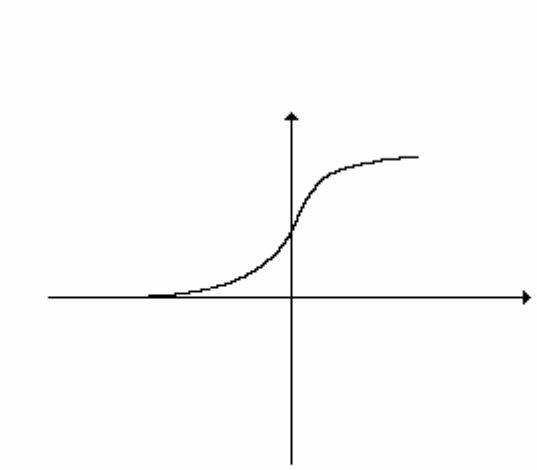


Şekil 2.4: Adım Fonksiyonu'nun Şekilsel Gösterimi

2.2.4.3 Sigmoid Aktivasyon Fonksiyonu

Sigmoid aktivasyon fonksiyonu, türevi alınabilir, sürekli ve doğrusal olmayan bir fonksiyon olması nedeniyle uygulamada en çok kullanılan aktivasyon fonksiyonudur. Bu fonksiyon, net girdinin her değeri için 0 ile 1 arasında bir değer üretmektedir ve formülü şu şekildedir.

$$y = F(s) = \frac{1}{1+e^{-s}} \tag{2.2}$$

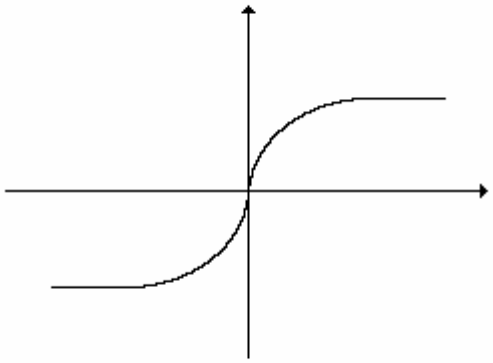


Şekil 2.5: Sigmoid Aktivasyon Fonksiyonu'nun Şekilsel Gösterimi

2.2.4.4 Hiperbolik Tanjant Fonksiyonu

Hiperbolik tanjant fonksiyonu, gelen net girdinin tanjant fonksiyonundan geçirilmesi ile hesaplanmaktadır ve sigmoid aktivasyon fonksiyonunun farklı bir çeşididir. Sigmoid aktivasyon fonksiyonunda çıktı 0 ile 1 arasında bir değer alırken, hiperbolik tanjant fonksiyonunda çıktı -1 ile 1 arasındadır ve şu şekilde hesaplanır.

$$y = F(s) = \frac{e^{s} + e^{-s}}{e^{s} - e^{-s}} \tag{2.3}$$



Şekil 2.6: Hiperbolik Tanjant Fonksiyonu'nun Şekilsel Gösterimi

2.2.5 Hücrenin Çıktısı

Aktivasyon fonksiyonu tarafından belirlenen çıktı değeridir. Bu değer ya başka bir yapay sinir hücresine girdi olarak ya da dış ortama gönderilmektedir. Bir işlemci elemanın birden fazla girdisi olmasına rağmen tek bir çıktısı olmaktadır.

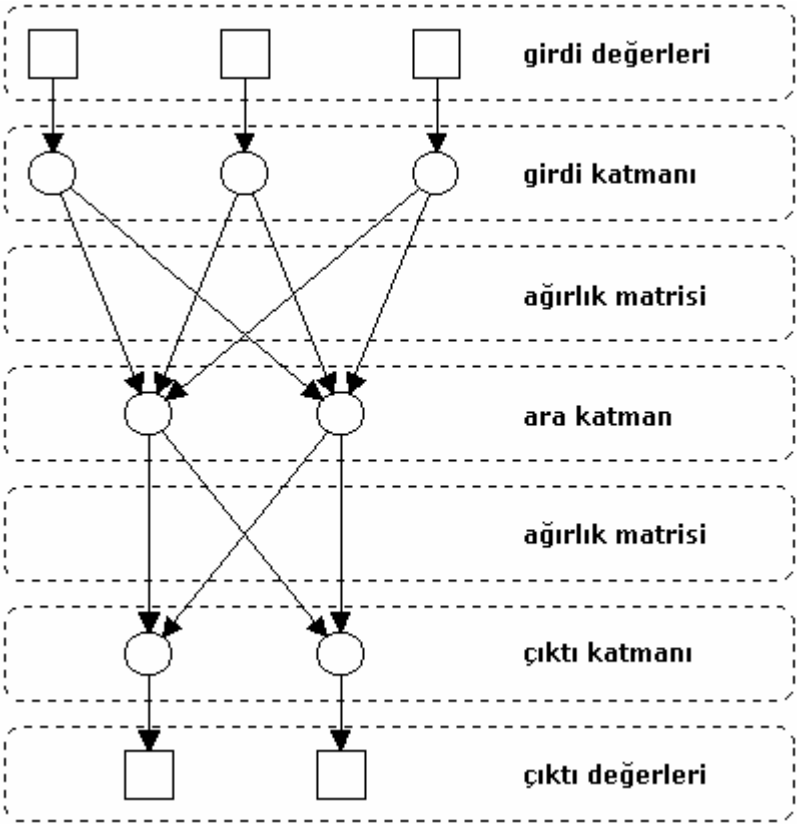
2.3 Yapay Sinir Ağlarının Yapısı

Yapay sinir hücreleri bir araya gelerek yapay sinir ağını meydana getirmektedir. Yapay sinir hücrelerinin bir araya gelmesi rasgele değildir. Yapay sinir hücrelerinin genellikle birbiriyle bağlantılı 3 katman halinde ve her katman içinde birbirine paralel olacak şekilde bir araya gelerek ağı oluşturdukları görülür. Bu 3 katman, girdi katmanı (input layer), ara katman (hidden layer) ve çıktı katmanı (output layer) olarak adlandırılır.

Girdi katmanı, dış ortamdan bilgileri alarak ara katmanlara iletmekle görevlidir. Bazı yapay sinir ağlarında bu katmanda herhangi bir bilgi işleme olmamaktadır.

Ara katman, girdi katmanından gelen bilgileri işleyerek çıktı katmanına iletmekle görevlidir. Gizli katman olarak da adlandırılan ara katman birden fazla olabilmektedir. Ara katmanlar çok sayıda yapay sinir hücresi içermektedirler ve bu hücreler yapay sinir ağı içindeki diğer hücrelerle bağlantılıdırlar.

Çıktı katmanı, ara katmandan gelen işlenmiş bilgileri dış ortama aktarmakla görevlidir.



Şekil 2.7: Tipik 3 Katmanlı İleri Beslemeli Yapay Sinir Ağı Mimarisi (Fröhlich, 1997)

2.4 Yapay Sinir Ağlarının Eğitimi ve Testi

Yapay sinir ağlarında işlemci elemanlar arasındaki bağlantıların ağırlık değerlerinin değiştirilmesi işlemine “ağın eğitilmesi” denilmektedir. Başlangıçta rastgele atanan bu ağırlık değerleri, ağa gösterilen örneklerle değiştirilmektedir. Amaç, ağa gösterilen örnekler için doğru çıktıları üretecek ağırlık değerlerinin belirlenmesidir. Yapay sinir ağının eğitilmesinde kullanılan girdi ve çıktı çiftlerinden oluşan verilerin tümüne “eğitim seti” adı verilmektedir.

Yapay sinir ağlarının eğitim süreci, belli kurallar çerçevesinde olmaktadır. Bu kurallara öğrenme kuralları adı verilmektedir. Öğrenme kuralları 3.4’üncü kısımda ayrıntılı olarak ele alınacaktır. Ağırlıkların değiştirilmesi öğrenme kurallarına göre yapılır. Yapay sinir ağında ağırlıkların doğru değerlere ulaşması, örneklerin temsil ettiği problem konusunda ağın *genellemeler* yapabilme yeteneğine kavuşması demektir. Genelleme, yapay sinir ağının eğitiminde kullanılmamış, ancak aynı evrenden gelen girdi-çıktı örneklerini doğru sınıflandırabilme yeteneği olarak tanımlanır. Ağın bu genelleştirme özelliğine kavuşması işlemine “ağın öğrenmesi” denilir.

Yapay sinir ağlarında öğrenme iki aşamada gerçekleşir. Birinci aşamada ağa gösterilen örnek için ağın üreteceği çıktı belirlenir. Bu çıktı değerinin doğruluk derecesine göre, ikinci aşamada ağın bağlantılarının sahip olduğu ağırlıklar değiştirilmektedir. Ağın çıktısının belirlenmesi ve ağırlıkların değiştirilmesi öğrenme kuralına bağlı olarak farklı şekillerde olmaktadır.

Bir yapay sinir ağının eğitiminin tamamlanmasının ardından, ağın öğrenip öğrenmediğini (performansını) ölçmek için denemeler yapılarak ağın test edilmesi gerekmektedir. Bir ağı test etmek için ağın eğitimi sırasında görmediği, yani veri setinden test amaçlı olarak ayrılan örnekler kullanılır ve bu örnekler “test seti” adını alır. Test işleminde ağın ağırlık değerleri değiştirilmemektedir. Örnekler ağa gösterilmekte ve ağ eğitimi sırasında belirlenen ağırlık değerlerini kullanarak daha önce görmediği bu örnekler için

çıktılar üretmektedir. Elde edilen çıktıların doğruluk dereceleri ağın öğrenmesi hakkında bilgi vermektedir. Sonuç ne kadar iyi olursa eğitimin performansı da o kadar iyi demektir (Öztemel, 2003).

Eğitim ve test setleriyle ilgili temel sorun, yeterli eğitim ve test verisi miktarının ne olması gerektiğidir. Sınırsız sayıda veri bulunabilmesi durumunda, yapay sinir ağı mümkün olduğunca çok veriyle eğitilmelidir. Eğitim verisinin yeterli olup olmadığı konusunda emin olmanın yolu, eğitim verisinin miktarının arttırılarak, bunun ağın performansında bir değişiklik yaratıp yaratmadığına bakmaktır. Ancak bunun mümkün olmadığı durumlarda yapay sinir ağının eğitim ve test verileri üzerindeki performansının yakın olması da verilerin yeterli olduğuna ilişkin bir gösterge olarak kabul edilebilir. Bununla birlikte eğitim setinin içermesi gereken veri miktarı değişik yapay sinir modellerine ve özellikle problemin gösterdiği karmaşıklığa göre farklılık göstermektedir.

ÜÇÜNCÜ BÖLÜM
YAPAY SİNİR AĞLARININ SINIFLANDIRILMASI
VE
TEMEL ÖĞRENME KURALLARI

Yapay sinir ağlarının topolojisi, işlemci elemanların birbirleriyle bağlantı şekillerine göre değişiklik gösterir. Problemlerin çözümü için ise, ağın topolojisine göre seçilen öğrenme algoritmaları yardımıyla ağın ağırlıkları belirlenir. Yapay sinir ağları topolojileri ve öğrenme metodlarına göre çeşitli sınıflara ayrılabilmektedir. Bu bölümde, yapay sinir ağlarının topolojilerine ve öğrenme metodlarına göre sınıflandırılmalarından bahsedilerek, temel öğrenme kuralları kısaca anlatılmıştır.

3.1 Yapay Sinir Ağlarının Topolojilerine Göre Sınıflandırılması

Ağ topolojisi, yapay sinir ağlarında işlemci elemanlar arasındaki bağlantının şeklidir. Yapay sinir ağları için iki genel topoloji bulunmaktadır; ileri beslemeli (feed-forward) ve geri beslemeli (feedback) veya yinelemeli (recurrent) (Gülseçen, 1993).

3.1.1 İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli ağlarda işlemci elemanlar genellikle katmanlara ayrılmışlardır. İşaretler, girdi katmanından çıktı katmanına tek yönlü bağlantılarla iletilir (Saraç, 2004). Bir katmandaki her işlemci eleman bir sonraki katmandaki tüm elemanlarla bağlantılıdır ancak aynı katmandaki elemanlar arasında herhangi bir bağlantı bulunmamaktadır. Bu nedenle ileri beslemeli yapay sinir ağlarında, işlemci elemanlar arasındaki bağlantılar bir döngü oluşturmamakta ve bu ağlar girilen verilere hızlı bir şekilde çıktı üretebilmektedirler.

İleri beslemeli yapay sinir ağlarında, hücreler katmanlar şeklinde düzenlenmekte ve bir katmandaki hücrelerin çıktıları bir sonraki katmana ağırlıklıklar üzerinden girdi olarak verilmektedir. Girdi katmanı, dışardan

aldığı bilgileri hiçbir değişikliğe uğratmadan ara (gizli) katmandaki hücrelere iletmektedir. Bu bilgi, ara katman ve çıktı katmanında işlenerek ağ çıktısı belirlenir (Saraç, 2004).

İleri beslemeli yapay sinir ağları, öğrenme algoritması olarak genellikle geri yayılım öğrenme algoritmasını kullanmakta ve bu nedenle bazen geri yayılım ağları olarak da adlandırılmaktadır. Şekil tanıma, sinyal işleme ve sınıflandırma gibi problemlerde genellikle bu topoloji uygulanır.

3.1.2 Geri Beslemeli Yapay Sinir Ağları

Geri beslemeli ya da diğer adıyla yinelemeli yapay sinir ağları, ileri beslemeli ağların aksine dinamik bir yapıya sahiptirler. Geri beslemeli yapay sinir ağları, çıktı veya ara katmanlardaki işlemci elemanların giriş veya önceki ara katmanlardaki işlemci elemanlara geri beslendiği bir yapıya sahiptir. Böylece girdiler hem ileri hem de geri yönde aktarılmış olurlar. Geri beslemeli yapay sinir ağları dinamik hafızaya sahiptir ve bir andaki çıktı hem o andaki hem de daha önceki girdileri yansıtır.

Geri beslemeli yapay sinir ağlarında, en az bir işlemci elemanın çıktısı, kendisine ya da diğer işlemci elemanlara girdi olarak verilmekte ve genellikle geri besleme bir geciktirme elemanı (ara katman veya çıktı katmanındaki aktivasyon değerlerini bir sonraki iterasyona girdi olarak taşımakla görevli eleman) üzerinden yapılmaktadır. Geri besleme, bir katmandaki işlemci elemanlar arasında olduğu gibi katmanlar arasındaki işlemci elemanlar arasında da olabilmektedir. Bu yapısı sayesinde geri beslemeli yapay sinir ağları, doğrusal olmayan dinamik bir davranış gösterirler. Bu sayede, geri beslemenin yapılış şekline göre farklı yapı ve davranışta geri beslemeli yapay sinir ağları elde edilebilir (Saraç, 2004).

Geri beslemeli yapay sinir ağları karmaşık bir çalışma düzeneğine sahip olmalarına rağmen, dinamik hafızaları nedeniyle önceden tahmin uygulamalarında başarılı sonuçlar verirler.

3.2 Yapay Sinir Ağlarının Öğrenme Metodlarına Göre Sınıflandırılması

Yapay sinir ağlarında öğrenmeyi açıklayabilmek için öncelikle öğrenme kavramının açıklanması gerekir. Öğrenme için en uygun tanımlardan biri Simon tarafından önerilen ve geniş kabul gören tanımdır. Bu tanıma göre öğrenme, "zaman içinde yeni bilgilerin keşfedilmesi yoluyla davranışların iyileştirilmesi sürecidir" (Öztemel, 2003). Bilgisayarların öğrenebilmesi ise bilgisayara ilgili problem hakkında gerekli bilgilerin verilmesi ile mümkündür. Böylece bilgisayarlar da insanlar gibi zaman içerisinde tecrübe kazanabilmektedirler.

Yapay sinir ağlarında öğrenme örnekler yolu ile sağlanmaktadır. Bilgisayarlar önce kendilerine verilen bir örneğe bakarak bilgi edinmekte, daha sonra ikinci örneğe bakarak biraz daha bilgi edinmekte ve bu işlem, ağ problemle ilgili genellemeler yapabilecek seviyeye gelene kadar devam etmektedir. Bu işlemin matematiksel karşılığı, birtakım metod, kural ve algoritmalar yardımıyla ağdaki işlemci elemanlar arasındaki bağlantı ağırlıklarının sürekli yenilenerek değiştirilmesidir. Yapay sinir ağlarında her ağ modeli kendine göre bir öğrenme algoritması kullanabilse de temel olarak danışmanlı (supervised) öğrenme ve danışmansız (unsupervised) öğrenme olmak üzere iki tür öğrenme metodu bulunur.

3.2.1 Danışmanlı Öğrenme

Yapay sinir ağlarında en fazla kullanılan öğrenme metodu olan danışmanlı öğrenmede, yapay sinir ağına örnek olarak bir çıktı (beklenen çıktı) verilir ve bu çıktıyla ağın ürettiği çıktı karşılaştırılır. İki çıktı arasındaki fark hata olarak alınır. Başlangıçta genellikle rassal olarak verilen ağırlıklar ağ tarafından hata minimize edilene kadar döngüler halinde değiştirilir (Anderson ve McNeill, 1992). Genelleştirilmiş delta kuralı ve geri besleme algoritması danışmanlı öğrenme metoduna örnek olarak verilebilir.

3.2.2 Danışmansız Öğrenme

Danışmansız öğrenmede yapay sinir ağına sadece girdiler verilmekte, ulaşılması gereken beklenen çıktılar verilmemektedir. Girişte verilen örnek değerlere bakarak yapay sinir ağı, parametreler arasındaki ilişkileri kendi kendine öğrenir. Yapay sinir ağı daha sonra bağlantı ağırlıklarını aynı özellikleri gösteren örüntüler (patterns) oluşturmak üzere ayarlar. Danışmansız öğrenme genellikle sınıflandırma problemlerinin çözümünde kullanılmaktadır. ART (Adaptive Resonance Theory) danışmansız öğrenmeye örnek olarak verilebilir (Saraç, 2004).

Danışmansız öğrenme metodu, yapay sinir ağlarında sürekli araştırılan ve gelişen bir öğrenme metodudur. Bu metod, gelecekte bilgisayarların insan yardımı olmadan öğrenebileceklerinin göstergesidir. Ancak günümüzde sınırlı kullanım alanları bulan ve hala yoğun araştırma konusu olan bir öğrenme metodudur (Anderson ve McNeill, 1992).

Ayrıca hem danışmanlı hem de danışmansız öğrenmeyi birlikte kullanan yapay sinir ağları da bulunmaktadır. Bu ağlarda ağırlıkların bir kısmı danışmanlı öğrenmeyle bir kısmı da danışmansız öğrenmeyle ayarlanır. Radyal tabanlı yapay sinir ağları (Radial Basis Networks - RBN) ve olasılık tabanlı yapay sinir ağları (Probability Based Neural Networks - PBNN) bunlara örnek olarak verilebilir.

Bu iki temel öğrenme metodundan başka literatürde destekleyici öğrenme (reinforcement learning) adı verilen bir metod daha bulunmaktadır. Bu metoddan bazı kaynaklarda danışmanlı bazı kaynaklarda danışmansız öğrenmenin bir alt türü olarak, bazı kaynaklarda ise kendi başına bir öğrenme metodu olarak bahsedilmektedir. Bu metoda göre, yapay sinir ağına sadece girdiler verilmekte, bu girdilere karşılık çıktıları üretmesi beklenmekte ve bu çıktıların ne derece doğru olduğunu belirten bir skor veya derece bildirilmektedir (Jain ve Mao, 1996).

3.3 Yapay Sinir Ağlarında Öğrenmenin Uygulamaya Göre Sınıflandırılması

3.3.1 Çevrimiçi (On-line) Öğrenme

Bu kurala göre öğrenen sistemler, gerçek zamanda çalışırken bir taraftan fonksiyonlarını yerine getirmekte, bir taraftandan da öğrenmeye devam etmektedirler.

3.3.2 Çevrimdışı (Offline) Öğrenme

Bu kurala dayalı sistemler, kullanıma alınmadan önce örnekler üzerinde eğitilirler. Bu kuralı kullanan sistemler eğitildikten sonra gerçek hayatta kullanıma alındığında artık öğrenme olmamaktadır.

3.4 Temel Öğrenme Kuralları

Yapay sinir ağları modellerinin eğitilmesinde kullanılan pek çok öğrenme algoritması bulunur. Bu algoritmalar, yapay sinir ağının topolojisine, karşılaşılan problemin niteliğine göre farklılıklar gösterse de bir çoğunun temel aldıkları öğrenme kuralı Hebb Kuralı ve onun geliştirilmiş versiyonlarıdır. Bunun yanında, farklı öğrenme kuralları da vardır ve bu konuda çalışmalar sürmektedir. Temel olarak 4 öğrenme kuralından bahsedilebilir. Bunlar Hebb Kuralı, Hopfield Kuralı, Delta Kuralı ve Kohonen Kuralı'dır.

3.4.1 Hebb Kuralı

Hebb (1949) tarafından ve biyolojik öğrenme temel alınarak geliştirilen bu kural bilinen en eski öğrenme kuralıdır. Diğer öğrenme kurallarına da temel olan Hebb Kuralı'na göre, bir yapay sinir hücresi diğer bir yapay sinir hücresinden girdi alırsa ve her iki hücre de yüksek derecede aktif ise (matematiksel olarak aynı işareti taşıyorsa) her iki hücrenin arasındaki bağlantının ağırlığı artırılmalıdır.

3.4.2 Hopfield Kuralı

Hopfield Kuralı, bir farklılık dışında Hebb Kuralına benzer. Bu farklılık, Hopfield Kuralı'nda, bağlantı ağırlığında yapılacak olan değişikliğin büyüklüğünün de belirlenmesidir. Buna göre, girdi ve istenilen çıktının ikisi de aktifse, bağlantı ağırlığı öğrenme katsayısı kadar arttılmakta, aksi durumda ise öğrenme katsayısı kadar azaltılmaktadır. Öğrenme katsayısı genel olarak 0-1 arasında tasarımcı tarafından atanan sabit ve pozitif bir değerdir.

3.4.3 Delta Kuralı

Widrow ve Hoff tarafından geliştirilen bu kural Hebb Kuralının gelişmiş şeklidir. En çok kullanılan kurallardan biri olan Delta Kuralı, yapay sinir hücresinin gerçek çıktısı ile beklenen çıktısı arasındaki farkı azaltmak için yapay sinir ağlarının işlemci elemanları arasındaki bağlantı ağırlık değerlerinin sürekli değiştirilmesi ilkesine dayanır. Bu kuralla, gerçek çıktı ile beklenen çıktı arasındaki hatanın karesi en aza indirilmeye çalışılmaktadır. Bu nedenle En Küçük Kareler Kuralı (Least Mean Square Rule-LMS) olarak da adlandırılır. Ayrıca bazı kaynaklarda Widrow-Hoff Kuralı olarak da geçer.

3.4.4 Kohonen Kuralı

Kohonen (1982) tarafından geliştirilen bu öğrenme kuralı biyolojik sistemlerdeki öğrenmeden esinlenilmiştir. Bu kuralda işlemci elemanlar, ağırlıklarının ayarlanması (öğrenme) için yarışmaktadırlar. Kazanan işlemci elemanın bağlantı ağırlıkları değiştirilmektedir. En uygun çıktıya sahip işlemci elemanın kazandığı kuralda bu işlemci eleman, kendisine komşu işlemci elemanların ağırlıklarının değiştirilmesine de izin vermektedir.

DÖRDÜNCÜ BÖLÜM
YAPAY SİNİR AĞLARI MODELLERİ
VE
YAPAY SİNİR AĞLARI İLE İSTATİSTİK İLİŞKİSİ

4.1 Yapay Sinir Ağlarının Tasarımı

Bir yapay sinir ağında, işlemci elemanların bağlanması sonucu oluşan topoloji, işlemci elemanların sahip oldukları toplama ve aktivasyon fonksiyonları, kullanılan öğrenme metodu ve öğrenme kuralı ağın modelini belirlemektedir. Yapay sinir ağı uygulamasının başarısı, modelin oluşturulması aşamasının en doğru şekilde yürütülmesi ile yakından ilgilidir. Bunun için yapay sinir ağı tasarımcısının, ağın yapısına ve işleyişine ilişkin şu kararları vermesi gerekmektedir:

- Ağ mimarisinin seçimi ve yapısal özelliklerinin belirlenmesi (katman sayısı ve katmandaki işlemci eleman sayısı gibi),
- İşlemci elemanların kullandığı fonksiyonların karakteristik özelliklerinin belirlenmesi,
- Öğrenme algoritması ve parametrelerinin belirlenmesi,
- Eğitim ve test setinin oluşturulması.

Bu kararların doğru verilmesi, yapay sinir ağının daha hızlı ve daha başarılı sonuçlar üretmesini sağlamaktadır.

Bir problemin çözümü için uygulanacak olan yapay sinir ağı modeli öncelikle problemin türüne bağlı olmaktadır. Hangi ağ modelinin hangi problemin çözümü için daha uygun olduğunun bilinmesi oldukça önemlidir. Kullanım amacı ve o alanda başarılı olan ağ modelleri aşağıda görülmektedir.

Tablo 4.1 Kullanım amaçlarına göre yapay sinir ağları (Saraç, 2004)

Kullanım Amacı	Ağ Türü	Ağın Kullanımı
Tahmin	• Çok Katmanlı Perceptron • Elman Ağı • Jordan Ağı	Ağın girdilerinden bir çıktı değeri tahmin edilmesi
Sınıflandırma	• LVQ Ağı • ART Ağları • Kohonen Ağı • Counterpropagation • Olasılık Tabanlı Yapay Sinir Ağları	Girdilerin hangi sınıfa ait olduğunun belirlenmesi
Veri İlişkilendirme	• Hopfield Ağı • Boltzman Makinesi	Girdilerin içindeki hatalı bilgilerin bulunması ve eksik bilgilerin tamamlanması

Yapay sinir ağı mimarisi ve öğrenme algoritmasının seçimi birbirleriyle yakın ilişki içerisindedir. Ağda kullanılacak öğrenme algoritması seçildiğinde, ağın mimarisi de zorunlu olarak seçilmiş olmaktadır veya tam tersi ağın mimarisi seçildiğinde kullanılabilecek öğrenme algoritması da büyük ölçüde belirlenmiş olmaktadır.

Yapay sinir ağı modelinin tasarlamasındaki bir diğer adım ise, çok katmanlı bir ağ modelinin seçilmesi durumunda, ağdaki ara katman sayısına karar vermektir. İşlemci elemanların aynı doğrultu üzerinde bir araya gelmeleriyle katmanlar oluşmaktadır. Çoğu problem için 2 veya 3 katmanlı bir ağ tatmin

edici sonuçlar üretebilmektedir. Katman sayısını belirlemenin en iyi yolu, birkaç deneme yapılarak ağın performansına bağlı olarak en uygun katman sayısına karar vermektir.

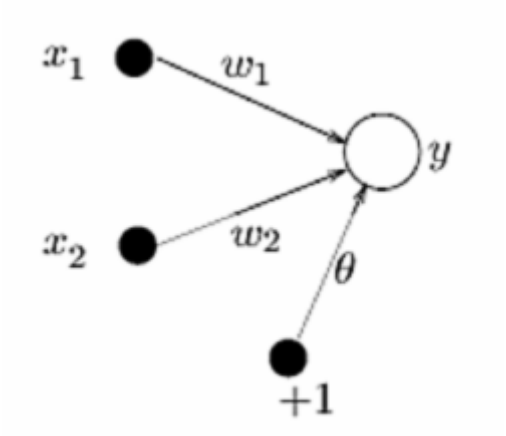
Ağın yapısal özelliklerinden bir diğeri de her katmandaki işlemci eleman sayısının belirlenmesidir. İşlemci elemanların sayısına da genellikle deneme-yanılma yolu ile karar verilir. Bir yapay sinir ağında işlemci eleman sayısının, olması gerekenden daha az olması ağın öğrenme yeteneğini azaltırken, gereğinden çok olması genelleme yeteneğini azaltmaktadır.

İşlemci elemanların karakteristik özelliklerinin belirlenmesi de yapay sinir ağının tasarımında önemli kararlardan biridir. Toplama ve aktivasyon fonksiyonlarının belirlenmesi, büyük ölçüde verilerin özelliklerine ve ağın neyi öğrenmesinin istendiğine bağlıdır. Aktivasyon fonksiyonları içinde en çok kullanılanlar, sigmoid ve hiperbolik tanjant fonksiyonlarıdır. Eğer ağın, bir modelin ortalama davranışını öğrenmesi isteniyorsa sigmoid, ortalamadan sapmasını öğrenmesi isteniyorsa hiperbolik tanjant fonksiyonlarının kullanılımı önerilir (Saraç, 2004).

4.2 Yapay Sinir Ağları Modelleri

4.2.1 İlk Yapay Sinir Ağları (Tek Katmanlı Yapay Sinir Ağları)

Tek katmanlı yapay sinir ağları sadece girdi ve çıktı katmanlarından oluşur. Her tek katmanlı yapay sinir ağının bir ya da birden fazla girdisi ve tek bir çıktısı vardır. Her girdi bir ağırlık değeriyle çıktıya bağlıdır. En basit tek katmanlı yapay sinir ağı iki girdi ve tek çıktıya sahip ağdır. Şekil 4.1'de tek katmanlı yapay sinir ağına örnek verilmektedir.



Şekil 4.1: İki girdi ve bir çıktıdan oluşan en basit tek katmanlı yapay sinir ağı modeli (Kröse, van der Smagt, 1996)

Bu ağlarda işlemci elemanların, dolayısıyla ağın çıktısının 0 (sıfır) olmasını önleyen bir eşik değeri (θ) bulunur ve değeri 1 olarak alınır. Tek katmanlı yapay sinir ağlarında, ağırlıklandırılmış girdi değerleri eşik değeri ile toplanıp aktivasyon fonksiyonundan geçirilerek ağın çıktısı elde edilir. Bu işlem şu şekilde,

$$y = F\left(\sum_{i=1}^{m} w_i x_i + \theta\right) \tag{4.1}$$

formülize edilmektedir.Tek katmanlı yapay sinir ağlarında çıktı fonksiyonu doğrusal bir fonksiyondur. Yani, ağa gösterilen örnekler iki sınıf arasında paylaştırılarak, bu iki sınıfı birbirinden ayıran doğruyu bulmak amaçlanır. Eşik değer fonksiyonu bu nedenle kullanılır.

Burada ağın çıktısı sınıfları temsil eden +1 veya -1 değerlerini alır. bu çıktı +1 ise örnek birinci, -1 ise ikinci sınıfta kabul edilir.

$$F(s) = \begin{cases} 1 & Eğer \quad s > 0 \\ -1 & Aksi \;\; takdirde \end{cases} \tag{4.2}$$

İki sınıfı birbirinden ayıran doğru;

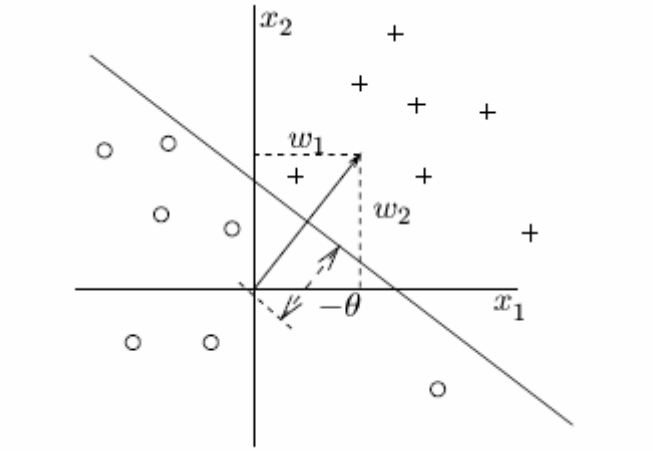
$$w_1 x_1 + w_2 x_2 + \theta = 0 \qquad (4.3)$$

şeklinde hesaplanır. Bu eşitlik ;

$x_1 = -(w_2 / w_1) x_2 - \theta / w_2$ ve

$$x_2 = -(w_1 / w_2) x_1 - \theta / w_2 \qquad (4.4)$$

olarak hesaplanır. Bu iki eşitlik kullanılarak sınıfları birbirinden ayıran doğru çizilir. Bu doğrunun geometrik gösterimi Şekil 4.2'deki gibidir;



Şekil 4.2: Ağırlıkların ve sınıfları birbirinden ayıran doğrunun geometrik gösterimi (Kröse, van der Smagt, 1996)

Tek katmanlı yapay sinir ağlarında öğrenmeden kastedilen, sınıfları birbirinden ayıran en uygun doğrunun bulunmasıdır. Bunun için ağırlık değerleri değiştirilmektedir. Ağırlıkların değiştirilmesi doğrunun eğiminin değiştirilmesi anlamına gelir. t zaman biriminde ağırlık değerleri Δw kadar değiştirilir ise;

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \tag{4.5}$$

olmaktadır. Öğrenme sırasında bu değişim her iterasyonda gerçekleştirilerek doğrunun en uygun eğimi bulunmaya çalışılır. Ancak bu işlem yeterli olmayabilir. Bu nedenle eşik değerinin de değiştirilmesi gerekir. Bu işlem, doğrunun sınıflar arasında kaymasına yardımcı olmaktadır. Böylece aktivasyon fonksiyonunun konumu belirlenmiş olur. Bu durumda t anında eşik değeri;

$$\theta(t+1) = \theta(t) + \Delta\theta(t) \tag{4.6}$$

şeklinde değiştirilmektedir. Öğrenme sırasında ağırlıklarda olduğu gibi eşik değeri de her iterasyonda $\Delta\theta$ kadar değiştirilmektedir.

Tek katmanlı yapay sinir ağlarında önemli iki modelden bahsedilebilir. Bu modeller perceptron ve Adaline/Madaline'dir.

4.2.1.1 Perceptron Modeli

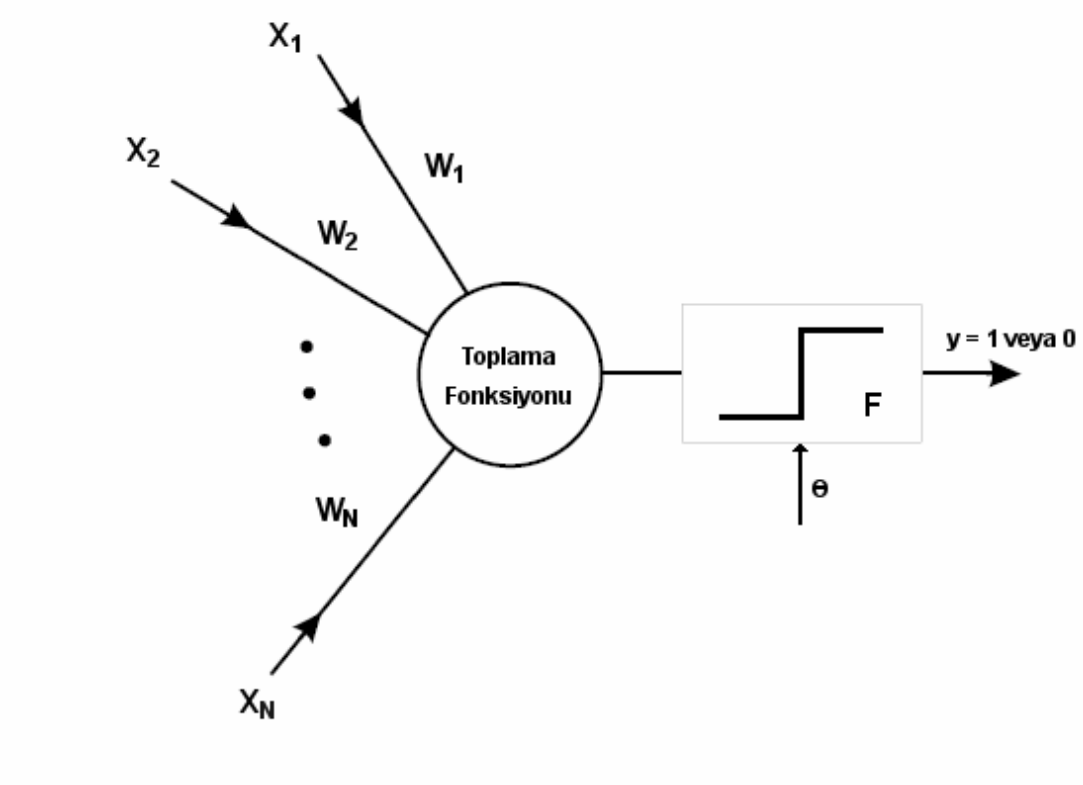
Perceptron, F.Rosenblatt tarafından 1950'li yılların sonlarında ortaya atılmış bir modeldir. İlk yapay sinir ağı modellerinden olan perceptronlar son derece sınırlı olmalarına karşın daha sonra geliştirilen modellere temel olmaları ve mevcut algoritmalarla yakın ilişki içinde oldukları için oldukça önemlidirler.

Perceptron en basit şekilde bir yapay sinir hücresinin birden fazla girdiyi alarak tek bir çıktı üretmesi temeline dayanmaktadır. Ağın çıktısı 1 veya 0 olmakta ve çıktı değerinin hesaplanması adım fonksiyonu kullanılarak yapılmaktadır.

Perceptron, eğitilebilen tek bir işlemci elemandan oluşmaktadır. İşlemci elemana girdi değerleri ve bu değerlere karşılık gelen çıktı değerleri gösterilerek öğrenme kuralına göre ağın çıktı değeri hesaplanmaktadır.

Olması gereken çıktı değerine ulaşılana kadar ağırlıklar ve eşik değerleri değiştirilmektedir.

Rosenblatt perceptronların eğitimi için perceptron öğrenme algoritmasını geliştirmiştir.



Şekil 4.3 Bir perceptron yapısı (Gurney, 1996)

Perceptron öğrenme algoritması dört adımda açıklanabilmektedir;

Adım 1: Yapay sinir ağına girdi değerleri ve ona karşılık gelen çıktı değeri gösterilmektedir. Birden fazla girdi değeri olabilmektedir. Buna karşılık çıktı değeri 1 ve 0 değerlerinden birini almaktadır.

Adım 2: İşlemci elemana gelen net girdi,

$$s = \sum_{i=1}^{m} w_i x_i \qquad (4.7)$$

kullanılarak hesaplanır

Adım 3 : Bu adımda işlemci elemanın çıktısı hesaplanır. Net girdi, eşik değeri ile kıyaslanır ve eğer net girdi eşik değerinden büyükse çıktı 1, küçükse 0 değerini alır. Yani;

$$y = \begin{cases} 1 & Eğer \quad s > \theta \\ 0 & Eğer \quad s \leq \theta \end{cases} \tag{4.8}$$

Eğer gerçekleşen çıktı ile beklenen çıktı aynı değeri alırlarsa, ağırlıklarda herhangi bir değişiklik olmaz. Eğer farklı bir çıktı üretilirse, şu iki durum söz konusu olur.

1. *Durum* : Eğer ağın beklenen çıktı değeri 0, ancak gerçekleşen çıktı değeri 1 ise, ağırlık değerleri azaltılır. Bunun için vektörel olarak şu eşitlik kullanılır;

$$W_n = W_o - \lambda X \tag{4.9}$$

burada λ, öğrenme katsayısıdır ve sabit bir değer olarak alınır. W_n yeni ağırlık vektörünü, W_o ise eski ağırlık vektörünü gösterir. Bu formül kullanılarak ağırlıklar girdi değerlerinin belirli bir oranında azaltılır.

2. *Durum* : Eğer ağın beklenen çıktı değeri 1, ancak gerçekleşen çıktı değeri 0 ise, ağırlık değerleri artırılır. Bunun için kullanılan vektörel eşitlik ise;

$$W_n = W_o + \lambda X \tag{4.10}$$

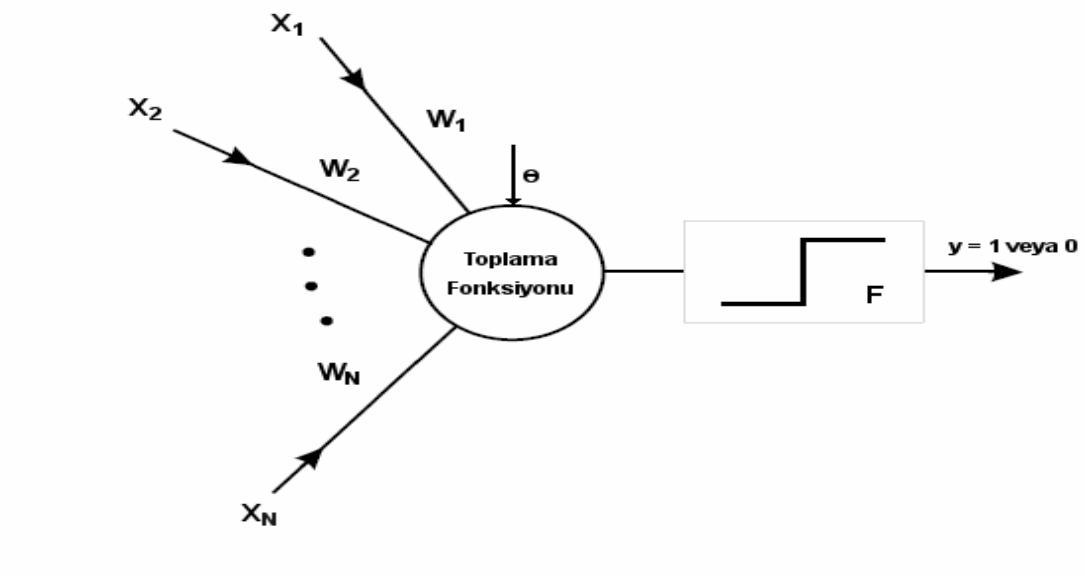
şeklindedir.

Adım 4 : Yukarıdaki ilk 3 adım, bütün girdi setindeki örnekler için doğru sınıflandırmalar yapılıncaya kadar tekrarlanır.

4.2.1.2 ADALINE Modeli

ADALINE, Widrow ve Hoff tarafından 1959 yılında geliştirilmiştir. Adaptif Doğrusal Eleman (ADAptive LİNear Element)'ın kısaltmasıdır. ADALINE genel olarak ADALINE ünitesi olarak adlandırılan tek bir işlemci elemandan oluşur.

ADALINE modeli, en küçük kareler ortalaması (least mean square) yöntemine dayanmaktadır. Öğrenme kuralına delta kuralı da denilmektedir. ADALINE modelinin yapısı perceptrona benzer, aralarındaki fark öğrenme kurallarının farklılığıdır. ADALINE modelinin öğrenmesi şu şekildedir.



Şekil 4.4 Bir ADALINE yapısı (Gurney, 1996)

Tek katmanlı ve doğrusal aktivasyon fonksiyonuna sahip tek çıktılı bir yapay sinir ağı için çıktı değeri,

$$y = \sum_j w_j x_j + \theta \tag{4.11}$$

ile elde edilir. Burada x_j girdi; w_j her girdinin ağırlık değeri; θ ise ağ çıktısının 0'dan farklı bir değer olmasını sağlayan eşik değeridir.

Beklenen çıktının d olması durumunda hata değeri;

$$E = d - y \tag{4.12}$$

şeklindedir. En Küçük Kareler kuralı olarak da adlandırılan Delta kuralında, en küçük hata fonksiyonunu verecek ağırlıklar dereceli azalma (gradient descent) yöntemi ile bulunur. Bu yönteme göre, herbir ağırlık değerinin hataya bağlı kısmi türevlerinin negatif (-) değerleri kullanılarak hata azaltılır.

ADALINE ağının çıktı değerini üreten aktivasyon fonksiyonu adım fonksiyonudur. Eşitlik 4.11 de verilen çıktı fonksiyonunun değeri 0'dan küçük ise çıktı -1; büyük ise +1 değerini alır. ADALINE modelinde herhangi bir t anında hatayı azaltmak için kullanılan kural,

$$w_i(t) = w_i(t-1) + \lambda E x_i \tag{4.13}$$

eşitliği ile gösterilir. Burada $w_i(t)$ ağırlığın t zamanındaki yeni değerini, $w_i(t-1)$ ağırlığın değişmeden önceki değerini, λ öğrenme katsayısını, E beklenen değer ile çıktı arasındaki hatayı ve x_i de girdi değerini göstermektedir.

Benzer şekilde θ eşik değeri de yine zaman içerisinde değiştirilerek olması gereken eşik değeri bulunmaktadır. Bunun matematiksel ifadesi ise;

$$\theta_i(t) = \theta_i(t-1) + \lambda E \quad (4.14)$$

şeklindedir.

Birden fazla ADALINE ünitesinin bir araya gelerek oluşturdukları yapay sinir ağına MADALINE ağı denilmektedir. MADALINE ağı ADALINE ile aynı öğrenme kuralını kullanır.

4.2.2. Çok Katmanlı Perceptron ve İleri Beslemeli Geri Yayılım Yapay Sinir Ağı

4.2.2.1. Çok Katmanlı Perceptron

Çok katmanlı perceptron (multi-layered perceptron-MLP), tek katmanlı perceptronların aksine doğrusal olmayan problemlere çözüm üretmeleri nedeniyle günümüzde geniş kullanım alanları bulan en popüler yapay sinir ağıdır. Ayrıca kullandığı öğrenme algoritması nedeniyle geri yayılım ağı olarak da anılmaktadır.

Minsky ve Papert doğrusal olmayan bir ilişki gösteren XOR problemini örnek göstererek tek katmanlı perceptronların doğrusal olmayan problemlerin çözümünde, yani gündelik hayatta karşımıza çıkan asıl sorunların çözümünde başarısız olduğunu ispatlamışlardır (Michie, Spiegelhalter, Taylor, 1994). Bu olay yapay sinir ağları üzerine yapılan çalışmaların durma noktasına gelmesine neden olmuştur. Ancak bazı bilim adamları çalışmalarına devam etmiştir. 1970'li yıllarda birbirinden bağımsız birkaç araştırmacının çok katmanlı perceptron mimarisi geliştirmesi ve Rumelhart, Hinton ve Williams'ın 1986 yılında geri yayılım algoritmasını geliştirerek XOR problemini çözmesiyle birlikte çalışmalar tekrar hız kazanmıştır (Kröse, van der Smagt, 1996).

Yapay sinir ağlarına olan ilginin tekrar artmasına neden olan çok katmanlı perceptronlar, başlangıçta akademik çevrelerin ilgisini çekerek üzerinde

yapılan çalışmaları yoğunlaştırmış, daha sonra mühendislik problemlerinin hemen hepsine çözüm üretebilecek bir güce sahip olması nedeniyle endüstriyel alanda yoğun bir şekilde kullanılmaya başlanmıştır. Özellikle sınıflandırma, tanıma ve genelleme yapmayı gerektiren problemlerde çok önemli bir çözüm aracıdır.

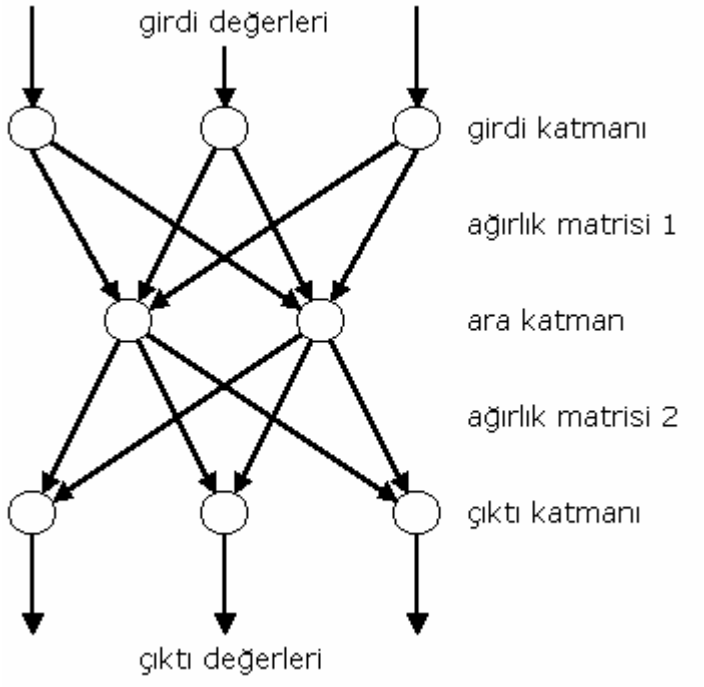
Çok katmanlı perceptron, Genelleştirilmiş Delta Kuralı (Generalised Delta Rule) adı verilen öğrenme kuralını kullanır. Genelleştirilmiş Delta Kuralı, en küçük kareler yöntemine dayalı bir öğrenme kuralı olan ve ADALINE ve tek katmanlı perceptron modellerinin öğrenme kurallarının daha gelişmiş hali olan Delta Kuralı'nın (Delta Rule) genelleştirilmiş şeklidir. Delta Kuralı ve dolayısıyla Genelleştirilmiş Delta Kuralı, danışmanlı bir öğrenme kuralıdır. Burada da temel amaç, ağın beklenen çıktısı ile ürettiği çıktı arasındaki hatayı en aza indirmektir. Çok katmanlı perceptrona eğitim sırasında hem girdiler hem de o girdilere karşılık gelen çıktılar gösterilir.

Çok katmanlı yapay sinir ağları 3 katmandan oluşmaktadır. Girdi katmanı, ara katman ve çıktı katmanı. Girdi katmanı; dış ortamdan gelen bilgileri alarak ara katmana göndermekle görevlidir. Bu katmanda herhangi bir bilgi işleme olmaz. Çok katmanlı perceptronlara birden fazla girdi gelebilmektedir ancak her işlemci elemanın bir tek girdisi ve bir tek çıktısı olur. Bu çıktı bir sonraki katmandaki tüm işlemci elemanlara gönderilir, yani bir katmandaki bir işlemci eleman, bir sonraki katmandaki tüm işlemci elemanlara bağlanır. Ancak çok katmanlı perceptronlarda bütün katmanlardaki işlemci elemanlar yalnızca bir sonraki işlemci elemanlarla bağlantılıdır, kendi içlerinde bağlantıları bulunmaz. Girdi katmanındaki işlemci eleman sayısı uygulanan problemin giriş sayısına bağlıdır.

Ara katman; girdi katmanından gelen bilgileri işleyerek bir sonraki katmana gönderir. Gizli katman olarak da adlandırılan ara katmanlar, bir çok katmanlı perceptronda birden fazla sayıda olabilmektedir. Ayrıca her ara katmanda birden fazla işlemci eleman bulunabilmektedir. Ara katman sayısı ve ara

katmanlardaki işlemci eleman sayısı deneme-yanılma yoluyla bulunur. Ara katmanda da her işlemci eleman bir sonraki katmandaki tüm işlemci elemanlarla bağlantılıdır.

Çıktı katmanı; ara katmandan gelen bilgileri işleyerek dış ortama gönderir. Bütün çok katmanlı perceptronlarda bir tek çıktı katmanı bulunmakta, ancak çıktı katmanında birden fazla işlemci eleman olabilir. Çıktı katmanındaki işlemci eleman sayısı ise yine uygulanan probleme bağlıdır.



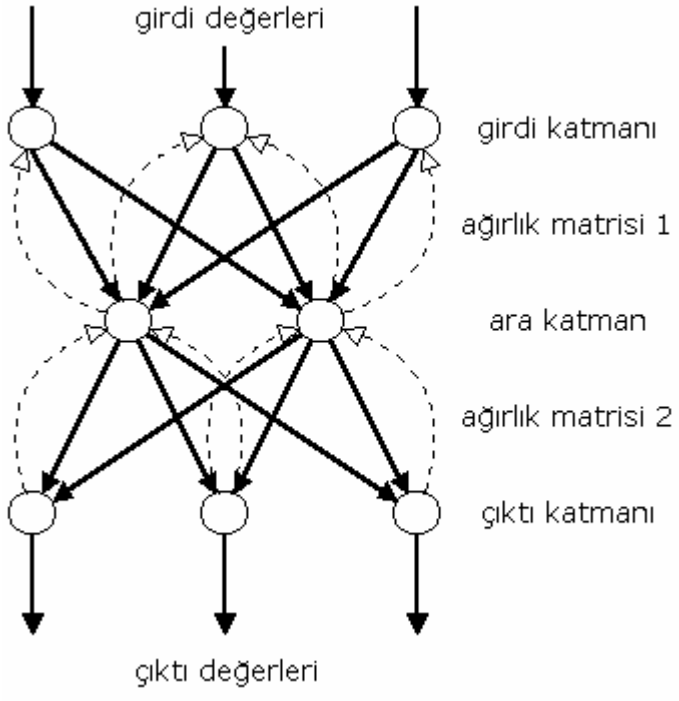
Şekil 4.5 Çok Katmanlı Perceptron Modeli (Fröhlich, 1997)

Çok katmanlı perceptronda bilgiler girdi katmanından ağa sunulur, ara katmanlardan geçerek çıktı katmanına ulaşır ve bu katman tarafından ağın cevabı olarak dış ortama iletilir. Bilgi akışı ileri doğru olduğu için çok katmanlı perceptronlar ileri beslemeli yapay sinir ağları sınıfına girmektedir.

4.2.2.2 İleri Beslemeli Geri Yayılım Yapay Sinir Ağı ve Geri Yayılım Algoritması

Çok katmanlı perceptron mimarisine sahip ve öğrenme yöntemi olarak geri yayılım algoritmasını kullanan ağlara, ileri beslemeli geri yayılım yapay sinir ağı adı verilmektedir. Ortaya çıkışından itibaren hem çok etkili hem de çok kullanışlı olmasından dolayı büyük bir popülarite kazanmıştır. Günümüzde en çok kullanılan yapay sinir ağı olarak bilinir.

Tipik bir ileri beslemeli geri yayılım ağının, bir girdi katmanı, bir çıktı katmanı ve en az bir ara katmanı vardır. Ara katman sayısı için teorik bir kısıt olmamakla birlikte genel olarak bir veya iki ara katman bulunur. Yapılan bazı çalışmalar, karmaşık problemlerin çözümünde en az üç ara katman kullanılmasının daha iyi sonuç verdiğini göstermiştir.



Şekil 4.6 İleri Beslemeli Geri Yayılım Yapay Sinir Ağı Yapısı (Fröhlich, 1997)

İleri beslemeli geri yayılım ağlarında katman sayısı ve her katmandaki işlemci eleman sayısı ağın başarılı sonuç üretebilmesi açısından önem taşır. Ancak

bu sayıların neler olacağına ilişkin net seçim kriterleri bulunmamaktadır. Bunun yerine uygulamalar sonucunda ortaya çıkmış ve araştırmacılar tarafından kabul görmüş bazı kurallar bulunmaktadır. Bu kurallar şu şekilde özetlenebilir (Anderson ve McNeill, 1992) :
Kural – 1 : Girdi ve çıktı veriler arasındaki ilişkinin karmaşıklık derecesi arttıkça işlemci eleman sayısı da arttırılmalıdır.

Kural – 2 : Eğer modellenen problem birçok aşamaya ayrılabiliyorsa, fazla sayıda ara katman kullanılmalıdır. Eğer az sayıda ara katman kullanılırsa ağ öğrenmeyi başaramaz. Gereğinden fazla ara katman kullanılması durumunda ise ağ ezberlemektedir (memorization). Bu da ağın, yeni örnekler için genelleme yeteneğini azaltmaktadır.

Kural – 3 : Yapay sinir ağının eğitilmesinde kullanılan örnek setinin genişliği, ara katmanlardaki işlemci elemanların sayısı için bir üst limit kriteri oluşturmaktadır. Bu üst limiti belirlemek için önce eğitim setindeki girdi-çıktı çiftlerinin sayısı bulunmalıdır. Bulunan bu sayı ağdaki toplam girdi ve çıktı işlemci elemanlarının sayısına bölünmesiyle elde edilen sonuç, bir ölçeklendirme katsayısı olarak kullanılabilir. Bu katsayı genellikle 5 ile 10 arasında bir değerdir. Fazla hata içeren veri setleri için bu katsayı 20 ile 50 arasında değerler alabilir. Veri setinin hemen hemen hiç hata içermemesi durumunda bu katsayı 2 seviyesine kadar düşürülebilir. Bu yöntemle, ölçeklendirme katsayısının ne olacağına bağlı olarak kesin bir sonuca ulaşılamasa da bir fikir edinmek mümkün olmaktadır. Ayrıca, genelleme yeteneğinin kaybolabilmesi ve dolayısıyla yeni veriler tanıtıldığında ağın kullanışsız kalması sonucunu doğurabileceğinden, bir ara katmandaki işlemci eleman sayısının çok fazla olmaması yararlı olacaktır.

Bir ağ oluşturmak için yukarıda bahsedilen kurallar uygulandıktan sonra, eğitme işlemine başlanır.

İleri beslemeli geri yayılım yapay sinir ağlarının öğrenme algoritması olan geri yayılım algoritması, yapay sinir ağlarında en çok kullanılan öğrenme algoritmasıdır. Özellikle öngörü ve sınıflandırma problemlerindeki üstün başarısı, kullanım alanlarını yaygınlaştırmıştır. Geri yayılım algoritması çok katmanlı bir ağ mimarisini gerektirir.

Geri yayılım algoritmasında hata, ağdaki ağırlıkların bir fonksiyonu gibi görülmekte ve hataların kareleri toplamı Delta kuralında olduğu gibi dereceli azalma yöntemi kullanılarak, minimize edilmeye çalışılmaktadır. Bu algoritma, hataları çıkıştan girişe geriye doğru azaltmaya çalışmasından dolayı geri yayılım ismini almıştır (Saraç, 2004).

Geri Yayılım Algoritması, danışmanlı öğrenme metodunu kullanmaktadır. Ağın öğrenebilmesi için eğitim seti adı verilen ve örneklerin bir kısmından oluşan bir sete ihtiyaç vardır. Bu set içinde her örnek için hem girdiler hem de o girdiler için ağın üretmesi gereken çıktılar belirlenir. Öğrenme sırasında her örnek için ağın çıktı değeri ile beklenen çıktı değeri karşılaştırılır ve hata değeri ağa tekrar geri besleme şeklinde verilir. Örnek setindeki hata kareleri toplamını azaltmak için işlemci elemanlar arasındaki bağlantı ağırlıkları değiştirilmektedir.

Bir yapay sinir ağının geri yayılım yoluyla eğitilmesi iki aşamadan oluşur.

1.aşama : İleri doğru hesaplama

2.aşama : Geriye doğru hesaplama

İleri doğru hesaplama, ağın çıktısını hesaplama aşamasıdır. Bilgi işleme, girdi setindeki bir örneğin ağa gösterilmesi ile başlamaktadır. Daha önce de değinildiği gibi, girdi katmanında herhangi bir bilgi işleme gerçekleşmemektedir. Gelen girdiler hiç bir değişiklik yapılmadan ara katmana gönderilir. Bu durum $y_k^i = x_k$ eşitliği ile gösterilebilir.

Ara katmandaki her işlemci eleman, girdi katmanındaki bütün işlemci elemanlardan gelen bilgileri bağlantı ağırlıklarıyla ağırlıklandırarak alır. Önce ara katmandaki işlemci elemanlara gelen net girdi;

$$s_j^a = \sum_{k=1}^{n} w_{kj} y_k^i \qquad (4.15)$$

ile hesaplanır.

Burada w_{kj} , $k.$ girdi katmanı elemanını, $j.$ ara katman elemanına bağlayan bağlantının ağırlık değerini; y_k^i ise girdi katmanındaki $k.$ işlemci elemanın çıktısını göstermektedir. $j.$ ara katman elemanının çıktısı ise bu net girdinin aktivasyon fonksiyonundan geçirilmesiyle hesaplanır. Aktivasyon fonksiyonu olarak genellikle sigmoid fonksiyon kullanılmakla birlikte, bu bir zorunluluk değildir, fakat geriye doğru hesaplamada fonksiyonun türevinin alınması gerektiğinden seçilen fonksiyonun, türevi alınabilir bir fonksiyon olmasına dikkat edilmelidir. Sigmoid fonksiyon kullanılması durumunda çıktı;

$$y_j^a = \frac{1}{1+e^{-(s_j^a+\beta_j^a)}} \qquad (4.16)$$

şeklindedir. Burada β_j , ara katmanda bulunan $j.$ elemana bağlanan eşik değer elemanın ağırlığıdır. Bu ağırlık değeri sigmoid fonksiyonun yönünü belirler. Eğitim esnasında ağ bu değeri kendisi belirler (Öztemel, 2003).

Ara ve çıktı katmanların işlemci elemanlarının çıktıları, aynı şekilde kendilerine gelen net girdinin (s_j^a) hesaplanması ve sigmoid fonksiyonundan geçirilmesi sonucu belirlenmektedirler. Çıktı katmanından çıkan değerler, yani çıktılar, elde edilince ağın ileri hesaplama aşaması tamamlanmış olur.

Geriye doğru hesaplama aşamasında ise ağa sunulan girdi için ürettiği çıktı ile ağın beklenen çıktısı karşılaştırılmaktadır. ADALINE modelinin öğrenmesinde açıklanan Delta öğrenme kuralının genelleştirilmiş halini kullanan geri yayılım algoritmasında daha önce verilen Delta kuralının doğrusal fonksiyonlarının doğrusal olmayan aktivasyon fonksiyonlarına göre uyarlanması gerekmektedir. Çıktı katmanındaki m. işlemci eleman için oluşan hata;

$$E_m = d_m - y_m \tag{4.17}$$

'dir. Bu bir tek işlemci eleman için oluşan hatadır. Çıktı katmanı için oluşan toplam hata ise;

$$E = \frac{1}{2}\sum_m E_m^2 \tag{4.18}$$

ile hesaplanır. Toplam hatayı en azlamak için işlemci elemanların hataları önceki katmanlara iletilir (geri yayılım). Hata önceki katmanlara iletilirken, aktivasyon fonksiyonun türevi ile bir aktivasyon işlemi uygulanmaktadır. Hatanın iletilmesi, katman katman geriye doğru olmakta ve bu süreçte Delta kuralı kullanılarak bağlantı ağırlıkları ayarlanmaktadır. İşlem girdi katmanına ulaşılana kadar devam eder ve bu noktada yeni bir döngüye başlanır. Ağın ağırlıkları değiştirilirken iki durum sözkonusudur. Bunlardan ilki, ara katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi; ikincisi ise ara katmanlar arası veya ara katman ile girdi katmanı arasındaki ağırlıkların değiştirilmesidir. Bu iki duruma göre hesaplama işlemi değişiklik gösterir.

Ara katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi şu şekilde açıklanabilir. Ara katmandaki j. işlemci elemanı çıktı katmanındaki m. işlemci elemana bağlayan bağlantının ağırlığındaki değişim miktarı Δw^a

olmak üzere; herhangi bir t zamanında (t. iterasyonda) ağırlığın değişim miktarı,

$$\Delta w_{jm}^{a}(t) = \lambda \delta_m y_j^a + \alpha \Delta w_{jm}^{a}(t-1) \tag{4.19}$$

ile hesaplanır. Burada λ öğrenme oranını, α ise momentum katsayısını göstermektedir. Öğrenme oranı, ağırlıkların bir sonraki düzeltmede hangi oranda değiştirileceğini belirlemektedir. Ağ üzerinde önemli bir etkisi bulunan öğrenme oranının değeri, uygulanan probleme göre değişmekle birlikte küçük bir değer seçilmesi, öğrenme süresinin uzamasına, büyük bir değer seçilmesi ise ağın yerel çözümlere takılmasına neden olur.

Yerel çözümün tanımı, şu şekilde açıklanabilir: Bir problemin çözümü için en az hatayı veren ağırlık vektörünü pratikte her zaman yakalamak mümkün olmayabilmektedir. Bu çözüm, ağın sahip olabileceği *en iyi çözüm*dür. Fakat bu çözüme nasıl ulaşılacağı tam olarak bilinmemektedir. Ağ eğitim sırasında bu çözüme ulaşmaya çalışır, ancak bazen ağ farklı bir çözüme takılabilmekte ve performansını daha fazla iyileştirmek mümkün olmamaktadır. Bu çözüm vektörlerine ***yerel çözümler*** denir. O nedenle tasarımcılar ağların ürettikleri çözümlerde ε (tolerans değeri) kadar hatayı kabul etmektedirler. Tolerans değerinin altındaki herhangi bir noktada problem öğrenilmiş kabul edilir (Öztemel, 2003).

Momentum katsayısı, ağın performansı üzerinde etkisi bulunan diğer bir parametredir. Özellikle yerel çözümlere takılan ağların bir sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacıyla kullanılır. Bu değerin küçük olması yerel çözümlerden kurtulmayı, büyük olması ise tek bir çözüme ulaşmayı zorlaştırabilmektedir.

Eşitlikteki δ_m, $m.$ işlemci elemanın hatasını göstermekte ve şu şekilde hesaplanmaktadır;

$$\delta_m = f'(s).E_m \tag{4.20}$$

Buradaki $f'(s)$ aktivasyon fonksiyonunun türevidir. Sigmoid fonksiyonu kullanılması durumunda;

$$\delta_m = y_m(1-y_m).E_m \tag{4.21}$$

olacaktır. Değişim miktarı hesaplandıktan sonra ağırlıkların $t.$ İterasyondaki yeni değerleri,

$$w_{jm}^a(t) = w_{jm}^a(t-1) + \Delta w_{jm}^a(t) \tag{4.22}$$

şeklini alır. Benzer şekilde eşik değer elemanının da ağırlıklarını değiştirmek gerekir. Onun için önce değişim miktarı hesaplanmalıdır. Eğer çıktı katmanındaki işlemci elemanların eşik değer ağırlıkları β^o ile gösterilirse, bu elemanın çıktısının sabit ve 1 olması nedeni ile değişim miktarı,

$$\Delta\beta_m^o(t) = \lambda\delta_m + \alpha\Delta\beta_m^o(t-1) \tag{4.23}$$

olacaktır. Eşik değerin $t.$ İterasyondaki ağırlığının yeni değeri ise,

$$\beta_m^o(t) = \beta_m^o(t-1) + \Delta\beta_m^o(t) \tag{4.24}$$

şeklinde hesaplanacaktır.

Geriye doğru hesaplama aşamasındaki ikinci durum ara katmanlar arası veya ara katman ile girdi katmanı arasındaki ağırlıkların değiştirilmesidir. Ara katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesinde her ağırlık için sadece çıktı katmanındaki bir işlemci elemanın hatası dikkate alınırken, girdi katmanı ile ara katman arasındaki (veya iki ara katman arasındaki) ağırlıkların değiştirilmesinde çıktı katmanındaki bütün işlemci elemanların hatasından payını alması gerekmektedir. Bu ağırlıklardaki değişim (mesela girdi katmanı ile ara katman arasındaki ağırlıkların değişimi) Δw^i ile gösterilirse değişim miktarı;

$$\Delta w_{kj}^i(t) = \lambda \delta_j^a y_k^i + \alpha \Delta w_{kj}^i(t-1) \tag{4.25}$$

olacaktır. Buradaki hata terimi δ_j^a ise şu şekilde hesaplanmaktadır.

$$\delta_j^a = f'(s) \sum_m \delta_m w_{jm}^a \tag{4.26}$$

Aktivasyon fonksiyonu olarak sigmoid fonksiyon kullanılması durumunda hata değeri,

$$\delta_j^a = y_j^a (1 - y_j^a) \sum_m \delta_m w_{jm}^a \tag{4.27}$$

ile hesaplanır.

Değişim miktarı hesaplandıktan sonra ağırlıkların t. iterasyondaki yeni değerleri ise,

$$w_{kj}^i(t) = w_{kj}^i(t-1) + \Delta w_{kj}^i(t) \tag{4.28}$$

kullanılarak hesaplanır.

Benzer şekilde ara katman eşik değer ağırlıkları β^a ile gösterilirse değişim miktarı;

$$\Delta\beta_j^a(t) = \lambda\delta_j^a + \alpha\Delta\beta_j^a(t-1) \tag{4.29}$$

olacaktır. Ağırlıkların yeni değerleri ise t. iterasyonda şu şekilde hesaplanmaktadır.

$$\beta_j^a(t) = \beta_j^a(t-1) + \Delta\beta_j^a(t) \tag{4.30}$$

Böylelikle ağın ağırlıklarının hepsi değiştirilmiş olur. Bir iterasyon hem ileri hem de geriye hesaplamaları yapılarak tamamlanır. İkinci bir örnek verilerek sonraki iterasyona başlanmakta ve aynı işlemler öğrenme tamamlanıncaya kadar yinelenmektedir.

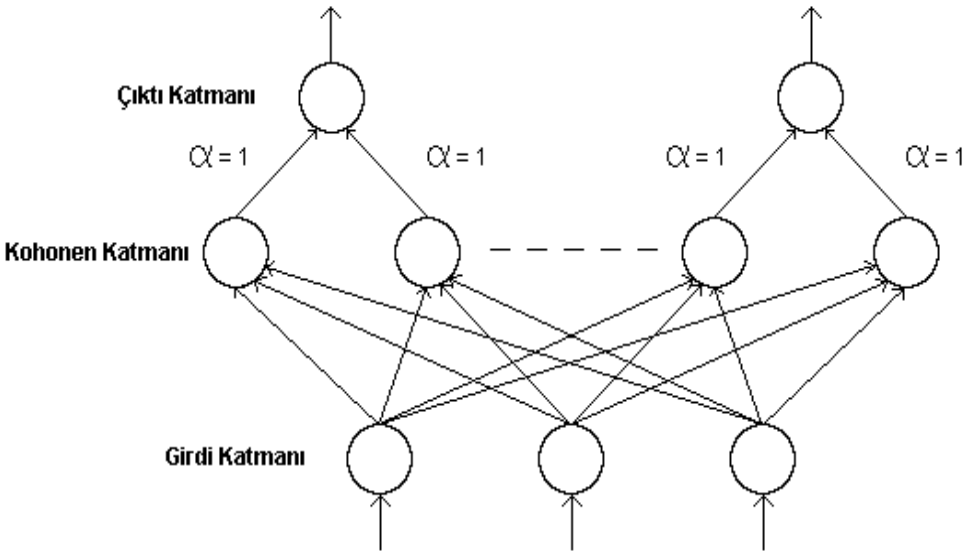
4.2.3 LVQ Ağı

LVQ (Learning Vector Quantization) ağı 1984 yılında Tuevo Kohonen tarafından geliştirilmiştir. Bu model ve Kohonen'in LVQ ağından önce geliştirdiği SOM (Self-Organizing Maps) modeli Kohonen katmanı denilen bir yapı üzerine temellendirilmiştir. Kohonen katmanı n boyutlu bir vektörü bir vektörler setine uydurmaktadır (mapping) (Anderson ve McNeill, 1992). Burada amaç, bir vektörün belirli sayıda vektör ile gösterilmesidir. LVQ ağının öğrenmesi ile de girdi vektörünün hangi vektör seti tarafından temsil edilmesi gerektiğinin bulunması kastedilmektedir. Bu vektör setine referans vektörleri denirse LVQ ağının görevi, öğrenme yolu ile bu referans vektörleri belirlemektir. Yani, girdi vektörlerinin üyesi olabilecekleri vektör sınıfını belirlemektir (Öztemel, 2003).

Çok katmanlı perceptronlar gibi LVQ ağları da 3 katmandan oluşmaktadır. İlk katman olan girdi katmanında bilgi işleme olmamaktadır. Dış ortamdan gelen bilgiler bu katmandan ağa giriş yaparlar. Gelen bilgiler girdi katmanını oluştururlar. İkinci katman, Kohonen katmanı da denilen ara katmandır. Bu

katmanda girdi setine en yakın olan ağırlık vektörü belirlenir. Bu katmandaki her eleman bir referans vektörünü gösterir. Girdi vektörü, girdi katmanı ile Kohonen katmanı arasındaki ağırlıkların oluşturduğu referans vektörlerine uydurulmaktadır. Üçüncü katman olan çıktı katmanında ise girdinin ait olduğu sınıf belirlenir.

LVQ ağları, girdi ve Kohonen katmanları arasında tam bağlantılı, Kohonen ve çıktı katmanları arasında ise kısmi bağlantılı olmaktadır. Yani girdi katmanındaki her işlemci eleman Kohonen katmanındaki tüm işlemci elemanlarla bağlantılıdır. Kohonen katmanındaki işlemci elemanlar ise çıktı katmanındaki bir tek işlemci elemanla bağlantılıdır. Kohonen katmanı ile çıktı katmanı arasındaki ağırlıklar (α) sabit olup 1'e eşittir ve bu ağırlıklar değişmez. Ağın eğitimi sadece Kohonen katmanı ile girdi katmanı arasındaki ağırlıkların yani referans vektörlerin değerlerinin değiştirilmesi ile gerçekleştirilir.



Şekil 4.7 LVQ ağının topolojik yapısı

LVQ ağı, öğrenme kuralı olarak Kohonen öğrenme kuralını kullanmaktadır. Danışmansız öğrenmenin bir alt türü olan yarışmacı öğrenme (competitive learning) sınıfına giren ağda, eğitim sırasında girdi vektörü ile referans vektörü arasındaki en kısa mesafe aranmakta ve girdi vektörünün en kısa mesafede bulunan vektör grubuna ait olduğu varsayılmaktadır. Ağın ağırlıkları değiştirilerek girdileri doğru sınıflara ayıracak referans vektörleri belirlenmektedir. Çıktı değerlerinin belirlenmesinde "*kazanan herşeyi alır (winner takes all)*" stratejisi uygulanır. Ağ eğitilirken her iterasyonda ağın ürettiği çıktının değeri yerine sadece doğru olup olmadığı belirlenir. Sadece girdi vektörüne yakın olan vektörün (kazanan vektör) değerleri (ağın bu vektöre ait ağırlıkları) değiştirilir (Kröse, van der Smagt, 1996).

LVQ ağlarında öğrenme hızı çok katmanlı perceptron gibi ağlara göre daha hızlı gerçekleşmektedir. Hem Kohonen katmanındaki hem de çıktı katmanındaki her işlemci elemanın çıktıları ikili (binary) değerler almaktadır ve sadece bir işlemci elemanın çıktı değeri 1 diğerleri ise 0 olmaktadır. Çıktı değerinin 1 olması girdinin, ilgili çıktının temsil ettiği sınıfa ait olduğunu göstermektedir.

LVQ ağının eğitilmesindeki amaç her iterasyonda girdi vektörüne en yakın referans vektörünü bulmaktır. Referans vektörleri daha önce de belirtildildiği gibi Kohonen katmanındaki işlemci elemanları girdi katmanındaki işlemci elemanlara bağlayan ağırlık değerleridir. Öğrenme esnasında sadece referans vektörlerinin ağırlık değerleri değiştirilmektedir. Bu işlem Kohonen öğrenme kuralı kullanılarak yapılmaktadır. Kohonen öğrenme kuralı, Kohonen tabakasındaki işlemci elemanların birbirleriyle yarışması ilkesine dayanmaktadır. Yarışma kriteri, girdi vektörü ile ağırlık vektörleri (referans vektörler) arasındaki öklid (euclid) mesafesine bağlıdır. Hangi işlemci elemanın referans vektörü girdi vektörüne en yakın ise yarışmayı o kazanır. Girdi vektörü X ile referans vektörü w arasındaki mesafe d ile gösterilirse; $i.$ işlemci elemanın mesafesi,

$$d_i = \|W_i - X\| = \sqrt{\sum_j (w_{ij} - x_j)^2} \quad (4.31)$$

ile hesaplanır. Burada w_{ij}, ağırlık vektörünün $j.$, x_j ise girdi vektörünün $j.$ değerini gösterir. Girdi vektörü ile referans vektörlerinin hepsinin aralarındaki mesafe tek tek hesaplandıktan sonra hangi işlemci elemanın referans vektörü girdi vektörüne en yakın ise yarışmayı o kazanır. Öğrenme sırasında, sadece girdi katmanını bu işlemci elemana bağlayan ağırlık değerleri değiştirilir. Kazanan işlemci eleman için iki durum söz konusu olmaktadır.

İlk durumda kazanan işlemci eleman doğru sınıfın bir üyesidir. Bu durumda ilgili ağırlıklar girdi vektörüne biraz daha yaklaştırılmaktadır. Bu, aynı örnek ağa tekrar gösterildiğinde yine aynı işlemci elemanın kazanması için yapılır. Bu durumda ağırlıkların değiştirilmesi,

$$W_n = W_o + \lambda(X - W_o) \quad (4.32)$$

kullanılarak gerçekleştirilir. Burada λ, öğrenme katsayıdır. Öğrenme katsayısı zaman içinde 0 değerini alacak şekilde monoton olarak azalır. Bunun nedeni, girdi vektörünün referans vektörüne çok yaklaştığında durması ve aksi yönde tekrar uzaklaşmamasıdır. Aksi takdirde ters yönde tekrar uzaklaşma meydana gelecektir.

Kazanan işlemci elemanın ikinci durumunda ise, kazanan işlemci eleman yanlış sınıftandır. Bu durumda ağırlık vektörü girdi vektöründen uzaklaşmaktadır. Bunun amacı, bir daha aynı örnek geldiğinde aynı işlemci elemanın kazanmamasıdır. Bu durumda ağırlıklar,

$$W_n = W_o - \lambda(X - W_o) \quad (4.33)$$

kullanılarak değiştirilir. Öğrenme katsayısının zaman içinde azalması burada da geçerli olmaktadır.

Kohonen katmanı ile çıktı katmanı arasındaki ağırlıklar (α) eğitim sırasında değiştirilmemektedir.

Kohonen katmanındaki her işlemci elemanın çıktısı y^k ise,

$$y_i^k = \begin{cases} 1 & \text{Eğer i. işlemci eleman yarışı kazanırsa} \\ 0 & \text{Aksi halde} \end{cases} \qquad (4.34)$$

Kohonen katmanındaki işlemci elemanların çıktıları, bu işlemci elemanları çıktı katmanına bağlayan ağırlık değerleri ile çarpılarak ağın çıktısı hesaplanmaktadır. Yani,

$$y_i = \sum_j y_j^k \alpha_{ki} \qquad (4.35)$$

olmaktadır. Bu Kohonen katmanında yarışmayı kazanan işlemci elemana bağlı olan çıktı elemanın değerinin 1, diğerlerinin değerinin 0 olması anlamına gelmektedir. Ağın çıktıları belirlendikten sonra çıktının doğru sınıflandırılıp sınıflandırılmadığı sorgulanır. Bu sorunun cevabına göre Kohonen katmanındaki yarışmayı kazanan işlemci elemanı girdi katmanına bağlayan ağırlıklar değiştirilmektedir.

Eğitim setindeki tüm örnekler doğru sınıflandırılıncaya kadar bu işlemler tekrarlanır. Hepsi doğru sınıflandırılınca öğrenme gerçekleştirilmiş olur. Anlatılan bu model standart bir LVQ modelidir.

LVQ modelinin bazı dezavantajları bulunmaktadır. Bunlardan ilki öğrenme katsayısının zamanında 0 değerini almaması durumunda ağın doğru ağırlık değerlerinden uzaklaşmasıdır. Ayrıca bazı problemlerde sürekli aynı referans vektörü yarışmayı kazanmaktadır. Bu durum da ağın esnekliğini ortadan kaldırmaktadır. Diğer dezavantaj ise sınıflandırmayı yaparken iki sınıfın tam ortasında veya sınırlara çok yakın bulunan vektörlerin hangi sınıfa gireceklerinin belirlenememesidir. Bu sorunları ortadan kaldırmak için LVQ ağının farklı modelleri geliştirilmiştir.

Bu modellerden LVQ2 ağı yine Kohonen tarafından, sınır değerlerindeki yanlış sınıflandırmayı önlemek amacıyla geliştirilmiştir. Standart LVQ metodunun aksine LVQ2 ağı eğitim sırasında aynı anda 2 referans vektörünü değiştirmeyi önermektedir. Bu iki vektöre W_1 ve W_2 denirse, bu vektörlerin ağırlıklarının değiştirilmesi için iki koşulun sağlanması gerekmektedir. Bu koşullar;

1. W_1 girdi vektörüne en yakın ağırlık vektörü, W_2 ise ondan sonraki en yakın ağırlık vektörü olmak üzere W_1 yanlış, W_2 ise doğru sınıftandır.

2. Girdi vektörü W_1 ve W_2 vektörlerinin arasında merkezi olarak belirlenmiş bir aralık içerisinde kalmaktadır.

Bu iki koşulun sağlanması durumunda W_1 ve W_2 vektörlerinin ikisinin de ağırlıkları değiştirilir. Bu ağırlık vektörlerinin yeni değerleri $(W_{1n}$ ve $W_{2n})$ şu şekilde hesaplanır.

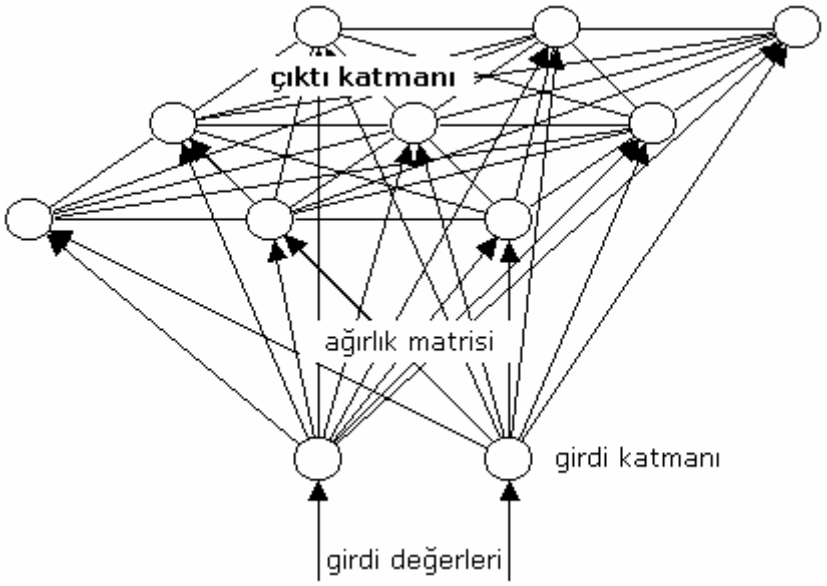
$$W_{1n} = W_{1o} - \lambda(X - W_{1o})$$
$$W_{2n} = W_{2o} - \lambda(X - W_{2o}) \qquad (4.36)$$

Burada W_{1o} ve W_{2o} ağırlık vektörlerinin değişmeden önceki değerlerini göstermektedir. LVQ ağları özellikle sınıflandırma problemlerinin çözümünde kullanılır.

4.2.4 Kohonen Ağı

SOM (Self Organization Feature Map Network-Özörgütlemeli Özellik Haritası) olarak da adlandırılan Kohonen ağı, beynin Neocortex tabakasında yaygın olan duyusal haritalardan esinlenilerek 1972 yılında Kohonen tarafından geliştirilmiştir. Genel olarak sınıflandırma yapmak için kullanılan Kohonen ağının, girdi vektörlerini sınıflandırma ve girdi vektörlerinin dağılımını öğrenebilme yetenekleri oldukça yüksektir.

Eğitimi sırasında herhangi bir danışmana veya beklenen çıktıya gerek duymayan bu yapay sinir ağı girdi ve çıktı katmanı olmak üzere iki katmandan oluşur. Çıktı katmanı iki boyutlu bir düzlemi, işlemci elemanlar ise bu düzlem üzerine dağılmış vektörleri gösterirler. Girdi elemanları çıktı elemanlarının tamamıyla bağlantılıdır. Yarışmacı öğrenmeyi kullanan Kohonen ağında kazanan işlemci eleman 1 diğer elemanlar ise 0 değerini alır. Eğitimi sırasında hem yarışmayı kazanan işlemci elemanın hem de komşusu olan işlemci elemanların ağırlıkları değiştirilir.



Şekil 4.8 Kohonen Ağı (Fröhlich, 1997)

Kohonen ağının eğitiminde, herhangi bir t zamanında örnek setinden herhangi bir örnek ağa gösterilir. Girdi vektörü X ve ağırlık vektörü W normalize edilmiş olmalıdır. Çıktı elemanlarından kazanan işlemci elemanı bulmak için iki yöntem bulunmaktadır. Bunlardan ilkinde, her elemanın çıktısı ağırlıklarla girdilerin çarpımının toplamı ile bulunur. Bu toplamın matematiksel ifadesi,

$$y_i = \sum_i w_{io} x_i \tag{4.37}$$

şeklindedir. Bu çıktı değerlerinden en yüksek değere sahip olan işlemci eleman yarışmayı kazanır. Bu işlemci elemanın $k.$ eleman olması durumunda,

$$y_k = 1$$

$$y_i = 0 \qquad i = 1,2,\ldots \text{ ve } i \neq k \tag{4.38}$$

olur. İkinci yöntemde ise, Öklit mesafesi (d) kullanılarak elde edilen, girdi vektörüne en yakın ağırlık vektörüne sahip işlemci eleman kazanan elemandır. İki vektör arasındaki mesafe,

$$d_j = \left\| X - W_j \right\| \tag{4.39}$$

ile hesaplanır. Her çıktı elemanı için bu mesafeler hesaplanmakta ve en küçük mesafe değerine sahip işlemci eleman kazanan eleman olarak belirlenmektedir. Kazanan işlemci elemanın belirlenmesinin ardından bu elemanın ve komşularının ağırlıkları,

$$W(t+1) = W(t) + \lambda g(i,k)(X(t) - W(t)) \tag{4.40}$$

formülü kullanılarak değiştirilir. Burada λ öğrenme katsayısıdır. Eğitim esnasında değeri zamanla küçültülür. $g(i,k)$, komşuluk fonksiyonudur ve i ve k elemanlarının komşuluklarını belirlemektedir. $i = k$ olması durumunda $g(i,k) = 1$ olur. Bu fonksiyon zaman içerisinde azalan bir fonksiyondur. Genel olarak $g(i,k)$,

$$g(i,k) = (exp(-\left\| d_i - d_k \right\|^2 / (2\sigma^2))) \tag{4.41}$$

şeklinde ifade edilir. Formülde, d_i ve d_k $i.$ ve $k.$ elemanların pozisyonunu gösteren vektörler, σ ise komşuluk alanının genişliğini gösteren ölçüdür. Bu genişlik zaman içerisinde azalmaktadır.

Kazanan işlemci elemanın onunla birlikte ağırlıklarının değiştirileceği komşularını belirlemek için iki yöntem bulunmaktadır. Bunların ilki, kazanan işlemci elemanın etrafındaki elemanları bir kare/dikdörtgen içine almak ve bunun içinde kalanların ağırlıklarını değiştirmektir. İkinci yöntem ise, kazanan

işlemci elemanın etrafındaki elemanlar bir çokgen içine alınarak içinde bulunan elemanların ağırlıklarının değiştirilmesidir.

4.2.5 ART Ağları

ART (Adaptive Resonance Theory-Uyarlamalı Rezonans Teorisi) ağları Grossberg'in 1976 yılında beynin fonksiyonlarını açıklamaya yönelik araştırmalarının sonucunda ortaya çıkmıştır. Bu model, biyolojik sisteme ait şu 3 temel üzerine kurulmuştur (Kröse, van der Smagt, 1996).

1 – Normalizasyon : Biyolojik sistemler bulundukları çevredeki değişimlere duyarlıdır ve bu değişikliklere uyum sağlayabilme özelliklerine sahiptir. Örneğin, çok fazla gürültü bulunan bir ortamda insanın bir süre sonra gürültüden rahatsız olmaması onun ortama uyum sağladığını ve çevresindeki olayları normalize ettiğini göstermektedir.

2 – Ayrıştırabilme : İnsanların farkedilmesi zor olan ayrıntıları ayrıştırabilme yeteneği bazen hayat kurtaran bir özelliktir. Örneğin, insan dinlenen bir kaplanla saldırmak üzere olan bir kaplan arasındaki farkı kolayca anlayabilecek bir sisteme sahiptir.

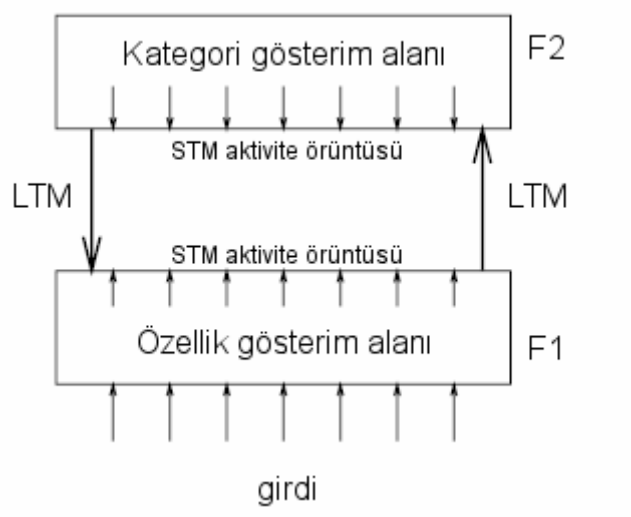
3 – Ayrıntıların Saklandığı Kısa Dönemli Hafıza : Belirlenen farklılıklar ve çevresel değişimler davranışlara dönüşmeden önce hafızada saklanmaktadır. Bilgilerin geçici olarak tutulduğu ve zaman içinde yok olarak yerine yeni bilgilerin saklandığı yer kısa dönemli hafızadır. Ancak sürekli aynı şeylerin tekrar edilmesi sonucunda bilgiler kısa dönemli hafıza yerine uzun dönemli hafızada saklanabilmektedir. Uzun dönemli hafıza bilgilerin sürekli tutulduğu ve kolay kolay unutulmadığı hafıza türüdür.

Grossberg ve arkadaşları beynin bu özelliklerinden yola çıkarak, beynin kullandığı sezgisel yaklaşımları matematik modele dönüştürmüşler ve ART ağlarını oluşturmuşlardır.

ART, adını öğrenme ve hatırlama arasındaki karşılıklı etkileşimi gerçekleştiren yöntemden yani rezonanstan almıştır. ART yapay sinir ağlarında işlemci elemanların çıktıları katmanlar arasında sürekli olarak ileri geri hareket etmektedir. Bu esnada eğer örnek girdi belirlenmiş bir sınıfa uyuyorsa ağ kararlı hale gelmekte ve ART ağı rezonanstadır denilmektedir (Gülseçen, 1993).

ART ağları, geribeslemeli ve danışmansız öğrenmeye dayalı yapay sinir ağlarıdır. Sınıflandırma problemlerinin çözümünde kullanılmak üzere geliştirilmiş olan ART ağları, daha sonra geliştirilen danışmansız öğrenme ağlarının ortaya çıkmalarında da temel olmuştur. ART ağları gerçek zamanda çalışabilir ve çevrimiçi öğrenebilirler.

ART ağları, genel olarak iki katmandan oluşur. Bu katmanlar F1 ve F2 olarak isimlendirilmiştir. F1 katmanı girdinin özelliklerini, F2 katmanı kategorileri (ayrıştırılmış sınıfları) gösterir. Bu iki katman birbirlerine uzun dönemli hafıza (LTM) ile bağlanırlar. Girdi bilgileri F1 katmanından alınarak, F2 katmanında sınıflandırma yapılmaktadır.



Şekil 4.9 ART ağlarının genel yapısı (Kröse, van der Smagt, 1996)

ART ağlarında girdiler sınıflandırılmadan önce, girdilerin özellikleri incelenerek F1 katmanının aktivasyonu belirlenir. LTM'de ki bağlantı değerleri ile girdiler kategorilere ayrılarak F2 katmanına gönderilir. F2 katmanındaki sınıflandırma ile F1 katmanından gelen sınıflandırma birbirleri ile eşleştirilerek, eğer örnek belirlenmiş bir sınıfa uyuyorsa o kategoriye ait olarak kabul edilir. Aksi takdirde, ya yeni bir sınıf oluşturulmakta ya da girdinin sınıflandırması yapılmaktadır.

ART ağlarının çalışması, aşağıdan yukarı (F1'den F2'ye) bilgi işleme ve yukarıdan aşağı (F2'den F1'e) bilgi işleme olmak üzere iki şekilde olmaktadır. Aşağıdan yukarı bilgi işleme durumunda bir girdi örüntüsü ağa gösterilir. Bu örüntü hem F1 katmanında STM'de X aktivite örüntüsünü oluşturur hem de oryantasyon sistemini veya diğer bir deyişle reset'i (yeniden yerleştirme modülünü) aktif etmek üzere bir işaret (signal) gönderir. Benzer şekilde oluşturulan X örüntüsü hem reset'e bir engelleyici (inhibitory) işaret göndermekte hem de F1 katmanından bir çıktı örüntüsü oluşturmaktadır. Bu çıktı örüntüsü F2 katmanına giden bir girdi örüntüsüne dönüştürülür. Bu girdi örüntüsü ise F2 katmanının çıktısı olan örüntüyü oluşturur. Bu aynı zamanda ağın çıktısıdır. Bu şekilde aşağıdan yukarı (F1 katmanından F2 katmanına) bilgi işleme tamamlanmış olmaktadır.

Yukarıdan aşağı bilgi işleme de benzer şekilde gerçekleşmektedir. Bu durumda, F2 katmanında oluşturulan çıktı örüntüsü yukarıdan aşağı bir sinyal gönderir. Bu sinyal daha sonra beklenen örüntüye dönüştürülür. Aynı zamanda kontrol faktörü (kazanç) için engelleyici (inhibitory) bir işaret gönderir. Bundan sonra beklenen örüntünün girdi örüntüsü ile eşlenip eşlenemiyeceğine bakılır. Eğer böyle bir eşleşme mümkün değilse F1 katmanında yeni bir STM örüntüsü (X^*) oluşturulur. Bu örüntü oryantasyon sistemindeki engelleyici işaretin etkisini azaltmaktadır.

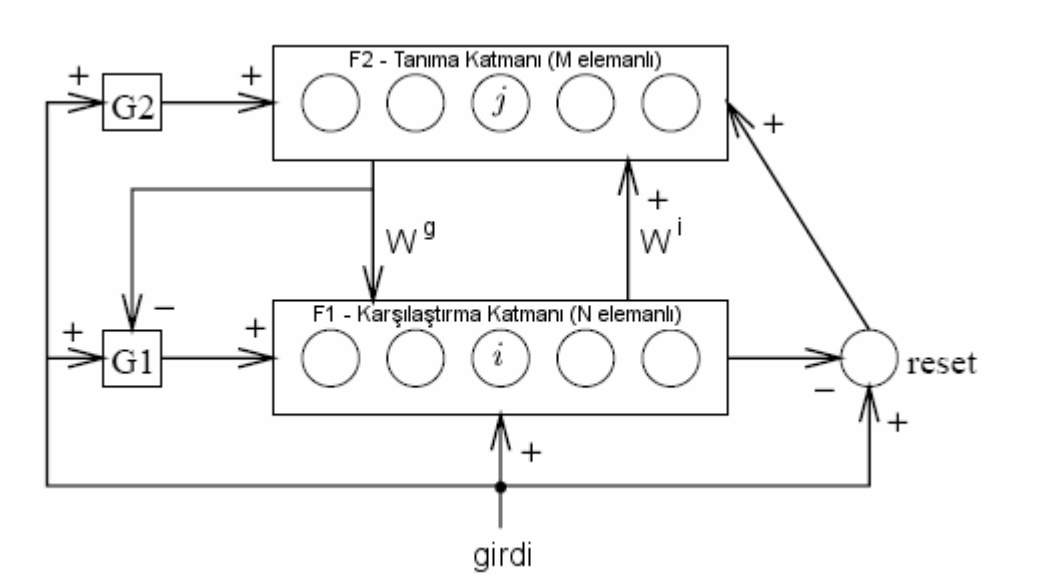
Oluşturulan X^* sinyali oryantasyon sistemindeki engelleyici işaretin etkisini azaltarak reset'in F2 katmanına bir sinyal göndermesini sağlar. Bu işaret F2 katmanında Y^* örüntüsünü oluşturur. Böylece ağa gösterilen girdi örüntüsü için doğru sınıfı gösteren Y^* çıktısı üretilir.

Eğer üretilen örüntü ile girdi örüntüsü eşleşirse o zaman sadece yukarıdan aşağı o girdinin sınıfını gösteren ağırlıklar değiştirilir. Bu değiştirme öğrenme kuralına göre gerçekleştirilmektedir.

1976 yılından bugüne dek çok sayıda ART ağları tanımlanmıştır. Bunlar arasında ART1, ART2, ART3, ARTMAP, Fuzzy ART gibi ağları saymak mümkündür. Bu ağların hepsi aynı temel felsefeye dayanmakta ancak öğrenme kurallarında çok az farklılıklar göstermektedir. Bu çalışmada yalnızca ilk ART ağı olarak tanımlanabilecek ART1 ağından bahsedilecektir.

4.2.5.1 ART1 Ağı

ART1 ağı, sadece ikili (binary) girdiler ile çalışan en basit ART ağı modelidir. ART1 ağında, F1 katmanı karşılaştırma katmanı, F2 katmanı tanıma katmanı olarak adlandırılır. F1 katmanındaki bütün işlemci elemanlar F2 katmanındaki bütün işlemci elemanlarla bağlantılıdır. Bu bağlantılar sürekli değerlerden oluşan ve w^i ile gösterilen LTM bağlantılarıdır. Bu bağlantıların özellikleri ileri doğru bağlantılar olmalarıdır. Aynı zamanda F2 katmanından F1 katmanına geriye doğru ikili değerlerden meydana gelen ve w^g ile gösterilen bağlantılar bulunmaktadır. Modelde G1 ve G2 ile gösterilen iki tane kazanç modülü ve bir tane de reset vardır.



Şekil 4.10 ART1 modelinin geometrik gösterimi (Kröse, van der Smagt, 1996)

Karşılaştırma katmanındaki her işlemci elemanın 3 girdisi bulunur. Bunlar:

- Girdi örneğinin bir elemanı (aşağıdan yukarı)
- Geri besleme değeri (yukarıdan aşağı)
- Kazanç değeri (G1'den gelen değer)

F1 katmanındaki işlemci eleman bu üç girdiden en az ikisinin aktif olması durumunda 1 değerini oluşturmaktadır. Buna 2/3 kuralı denilir.

F2 katmanındaki işlemci elemanlar ise kendilerine gelen net girdiyi hesaplarlar ve birbirleri ile yarışırlar. Genellikle en fazla çıktıyı üreten işlemci eleman yarışmayı kazanır.

G2 girdi örneğindeki bütün elemanlar sadece yeni bir sınıfın oluşmasında kullanılmaktadırlar.

G1 modülü girdi örüntüsünün değerlerini hafızada bulunan değerler ile eşleştirmede kullanılmaktadır. Gelen bilgilerin aşağıdan yukarı veya yukarıdan aşağı olup olmadığının bilinmesini sağlar. Aşağıdan gelen değerler bu kazanç değerini aktif, yukarıdan gelenler ise pasif yaparlar.

reset ise girdi vektörü ile F1 katmanı çıktısı (çıktı vektörü) arasındaki farkın belirli bir değeri aşması durumunda F2 katmanına bir sinyal gönderir. İki vektörün arasındaki fark benzerlik katsayısı (vigilance) denilen bir değer ile karşılaştırılır. Bu katsayı girdi vektörünün hafızada bulunan çıktı vektörlerine uygunluğunu belirlemektedir. Aradaki fark benzerlik katsayısından küçükse o zaman benzerlik yok demektir. Bu durumda girdi vektörü için yeni bir sınıf oluşturmak ve hafızaya yerleştirmek gerekir.

ART1 ağının eğitiminde öncelikle ağırlıklara başlangıç değerlerinin atanması gerekir. F1 katmanından ağa sunulan örnek sayısı N, F2 katmanında çıktı işlemci eleman sayısı M ise, M≥N olması önemlidir çünkü N adet örneğin hepsinin birbirinden farklı olması durumunda ağın her biri için bir sınıf ayıracak durumda olması gerekir. Bazı uygulamalarda sınıf sayısı belirli bir sayı ile sınırlandırılmaktadır. Bu durumda ağın görevi, girdi setini belirlenen sayıda sınıfa ayırmaktır. Bunu sağlamak için de benzerlik katsayısı kullanılır. ART ağlarının başlangıç değerlerinin atanmasında da iki durum sözkonusudur.

a) F2 ve F1 arasındaki geriye doğru ağırlıkların bütün değerleri başlangıçta 1 değerini almaktadır. Yani:

$$w_{ji}^{g} = 1$$
$$j : 1,2,3,...M$$
$$i : 1,2,3,...n \qquad (4.42)$$

burada M çıktı elemanı, n ise girdi elemanı sayısını (girdi vektörünün boyutunu) göstermektedir.

b) F1 ve F2 arasındaki ileri doğru ağırlıkların başlangıç değerleri, F1 katmanındaki işlemci eleman sayısı n olmak üzere,

$$w_{ij}^{i} = \frac{1}{1+n} \tag{4.43}$$

şeklinde atanır. Başlangıç değerlerinin atanmasından sonra benzerlik katsayısı belirlenir. Benzerlik katsayısı ρ ile gösterilir ve 0 ile 1 arasında bir değer alır. Bu katsayı, iki vektörün aynı sınıfın elemanı sayılabilmesi için birbirlerine ne kadar benzemesi gerektiğini belirlemektedir. Örneğin, bu değerin 0.8 olması yani benzerliğin %80 olması, girdi vektörünün çıktı vektörü ile aynı sınıfın elemanı olması için yeterlidir. Bunun altında bir benzerlik aynı grubun elemanı olmak için yeterli görülmemektedir.

Benzerlik katsayısı ne kadar büyük alınırsa, farklı sınıf sayısı da o kadar çoğalmaktadır. Bu sayı düşük alınırsa o zaman sınıf sayısı da az olmaktadır, fakat bu durumda ağın öğrenme performansı düşük olabilmektedir. Sınıf sayısının az veya çok olmasından çok öğrenilmesi istenilen olayı doğru temsil edebilen sınıf sayısını oluşturacak şekilde benzetim katsayısı belirlenmelidir.

Bir sonraki aşamada girdi setindeki örnek $X(x_1, x_2, \ldots x_n)$ vektörü olarak (her bir elemanı x_i olarak tanımlanmış şekilde) ağa gösterilmektedir.

Girdi setindeki örneğin ağa gösterilmesinin ardından F2 katmanındaki her işlemci elemanın çıktı değeri,

$$y_j' = \sum_{i}^{N} w_{ij}^{i}(t) x_i \qquad j : 1, 2, \ldots M \tag{4.44}$$

ile hesaplanmaktadır. Kazanan eleman, en büyük çıktıya sahip işlemci elemandır. Bu elemanın sahip olduğu ağırlık vektörüne en uygun sınıf

(kategori) gösterim vektörü denmektedir. Bunun $k.$ işlemci eleman olduğu varsayılırsa kazanan elemanın çıktısı,

$$y_k^* = max(y_j') \tag{4.45}$$

olacaktır. Burada * işareti kazanan elemanı temsil etmektedir. Bu elemanın ağırlık vektörü girdi vektörü ile karşılaştırılarak benzerlik katsayısına göre uygunluk sınaması yapılır. Burada kazanan elemanın ağırlık vektörü ile girdi vektörünü temsil edip edemeyeceğine karar verilir. Bunun için önce girdi vektöründe bulunan 1 sayısı $s1$ belirlenir.

$$s1 = |X| = \sum x_i \tag{4.46}$$

Daha sonra kategori gösterim vektörü (kazanan elemanın ağırlık vektörü) ile girdi vektörünün uyuştuğu 1 sayısı ($s2$) bulunmaktadır. Formülü,

$$s2 = |w_k^g . X| = \sum w_{jk}^g x_i \tag{4.47}$$

şeklindedir. Eğer, $\frac{s2}{s1} \geq \rho$ ise o zaman iki vektör birbirinin benzeri kabul edilmektedir. Yani kategori gösterim vektörü girdi vektörünü temsil edebiliyor demektir. Bu durumda ağırlıklar şu şekilde değiştirilmektedir.

$$w_{jk}^g(t+1) = w_{jk}^g(t)x_i$$

$$w_{ik}^i(t+1) = \frac{w_{jk}^g(t)x_i}{0.5 + \sum_i^N w_{ki}^g(t)x_i} \tag{4.48}$$

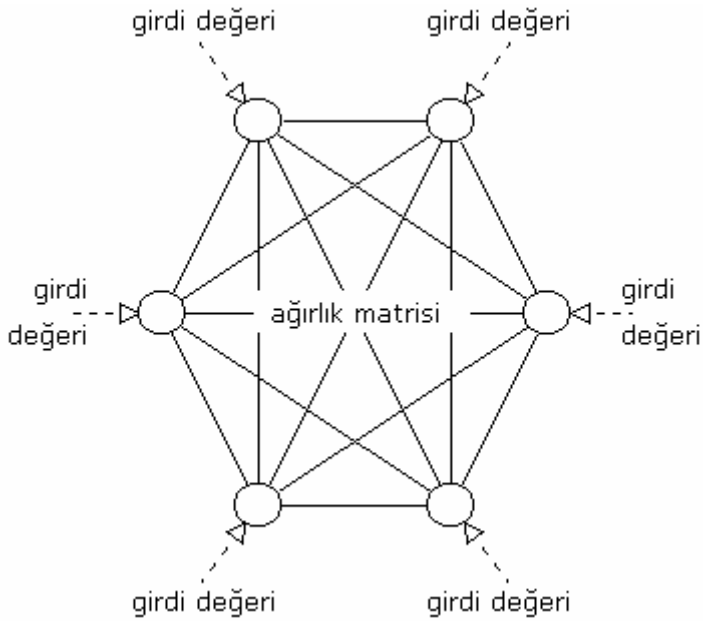
Eğer; $\frac{s2}{s1} < \rho$ ise o zaman en büyük çıktıyı veren işlemci elemanın çıktısı 0 olarak atanmakta ve ondan sonraki en büyük çıktıyı veren işlemci eleman

seçilerek F2 katmanındaki işlemci elemanların çıktılarının hesaplanmasından itibaren işlemler tekrar edilmektedir. Girdi vektörü bu sınıfın elemanı olarak atanmaz ise üçüncü en büyük çıktıyı veren işlemci elemanın sınıf (kategori) gösterim vektörü ile benzerliğine bakılarak benzerlik olması durumunda ağırlıklar değiştirilmektedir. Başlangıçta birinci örnek birinci sınıfın temsilcisi olarak atanır, ikinci örnek birinci ile aynı sınıftan ise (benzer ise) birinci sınıfın, aksi takdirde ikinci sınıfın elemanı olarak sayılır. Böylece örneklerin hepsi ya var olan sınıflardan birine girmekte ve ağırlıkları değiştirilmekte ya da yeni bir sınıfın temsilcisi olmaktadır. En kötü olasılık ile N örnek için N adet sınıf oluşturulabilir ($N = M$). Yukarıdaki işlemler bütün girdi vektörleri sınıflandırılıncaya kadar devam etmektedir.

4.2.6 Hopfield Ağı

Hopfield ağı modeli 1982 yılında bir fizikçi olan John Hopfield tarafından geliştirilmiştir. Hopfield ağı, beyine benzer bir şekilde belirli hafızaları veya örüntüleri depolayan basit bir sinir ağıdır. Basit yapısı ve matematiksel tutarlılığı nedeniyle yapay sinir ağları çalışmalarında önemli yeri olan ve tercih edilen bir modeldir. Her probleme uygulanamamasına rağmen, daha karmaşık modelleri anlamak için giriş bilgisi sağlamaktadır (Gülseçen, 1993).

Hopfield ağı tek katmanlı ve geri beslemeli bir yapay sinir ağıdır ve genellikle ikili (0 veya 1) ve bipolar (-1 veya +1) girişleri kabul etmektedir. Hopfield ağında işlemci elemanlar, birbirleriyle tam bağlantı halindedir ve hem girdi hem de çıktı elemanlarıdır. Bağlantılar çift yönlüdür (bilgi her iki yönde de akmaktadır) ve simetriktir. Ağın bağlantı değerleri bir enerji fonksiyonu olarak saklanır. Her iki yönde akan bilgiye uygulanan ve her bağlantı için hesaplanan bir ağırlık değeri vardır.



Şekil 4.11 Hopfield ağının yapısı (Fröhlich, 1997)

Her işlemci eleman diğer işlemci elemanlardan gelen girdilerin toplamını bir eşik değeri ile karşılaştırmakta ve diğer işlemci elemanlara çıktı olarak iletmektedir.

Hopfield ağında örnek örüntü seçilmekte ve ağın ağırlıklarının başlangıç değerlerini saptamak için kullanılmaktadır. Bu bir kere yapıldıktan sonra herhangi bir örüntü ağa sunulmakta ve bu da giriş örüntüsüne en çok benzeyen örüntülerden biriyle sonuçlandırılmaktadır. Çıktı örüntüsü, birimlerin durumlarına bakılarak ağdan okunabilmektedir (Elmas, 2003). Hopfield ağında ağırlık değerleri değişmemektedir.

Hopfield ağı, çıktı değerlerinin kesikli ve sürekli oluşlarına göre ikiye ayrılır, bu nedenle kesikli ve sürekli olmak üzere iki tür Hopfield ağından sözedilebilir.

Kesikli Hopfield Ağları, Birleşik Ağlar (Associative Networks) sınıfına girmektedir. Birleşik Ağlar beynimizin en önemli fonksiyonlarından biri olan

Birleşik Bellek kavramına dayanır. Bir ismin bir kişiyi, bir telefon numarasını, bir başka kişiyi çağrıştırması beynin Birleşik Bellek özelliğinden kaynaklanmaktadır. Birleşik Bellek kavramından hareketle bir grup yapay sinir ağı modelini kapsayan Birleşik Ağlar ortaya çıkmıştır. Birleşik Ağlar kabaca " bir vektör setini başka bir vektör setine bağlayan ağ" olarak tanımlanabilir. Kesikli Hopfield ağı ve Grossberg ağı bu tür ağlardır (Gülseçen, 1993).

Kesikli ve Sürekli Hopfield Ağlarının çalışma biçimleri aynıdır ancak kullanılan aktivasyon fonksiyonları farklıdır. Kesikli Hopfield Ağları aktivasyon fonksiyonu olarak signum fonksiyonunu kullanırken Sürekli Hopfield Ağları sigmoid fonksiyonunu kullanmaktadır.

Kesikli Hopfield ağının çalışması şu şekilde açıklanabilir. Bu ağdaki her elemanın iki değeri bulunur. İşlemci elemanın değeri +1 veya -1 olabilmektedir. Bir işlemci elemanın t. zamandaki girdisi $y_k(t)$ olarak gösterilirse hesaplanma formülü;

$$x_k(t) = \sum_{j \neq k}^{N} w_{jk} y_j(t-1) - \theta_k \qquad (4.49)$$

şeklindedir. Burada kullanılan w, ağırlık değerini, $y(t-1)$, işlemci elemanın bir önceki zaman dilimindeki çıktı değerini, θ, ise sabit eşik değerini göstermektedir.

Aynı işlemci elemanın çıktısı; $y(t)$ ise şöyle hesaplanmaktadır;

$$y(t) = sgn(x(t)) \qquad (4.50)$$

Buradaki sgn, signum fonksiyonunu göstermektedir. Yani,

$$y_k(t) = \begin{cases} +1 & \text{Eğer} \quad x_k(t) > u_k \\ -1 & \text{Eğer} \quad x_k(t) > u_k \\ y_k(t-1) & \text{Aksi halde} \end{cases} \tag{4.51}$$

Burada kullanılan u değeri de bir eşik değeridir. Pratikte 0 değeri seçilmekle birlikte böyle bir zorunluluk bulunmamaktadır.

Ağın öğrenme aşaması ağırlıkların belirlenmesi aşamasıdır ve bu işlem tek sefer yapılmaktadır. Yani ağın eğitimi bir defada olmaktadır. Bu işlem aşağıdaki formül yardımıyla gerçekleştirilir;

$$w_{ij} = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{p=1}^{M} \chi_i^p \chi_j^p & Eğer \quad j \neq k \quad ise \\ 0 & Aksi \quad halde \end{cases} \tag{4.52}$$

Burada M öğrenilecek (saklanacak) örnek sayısını, χ ise bir örneğin $i.$ ve $j.$ elemanının değerini göstermektedir. Bu ağırlıklar hesaplandıktan sonra sabitlenirler. Burada w_{ij} ve w_{ji} ağırlıkları aynıdır yani oluşturulan ağırlık matrisi simetrik bir matrisdir.

Ağın kullanılabilmesi için durağan (stable) hale gelmiş olması gerekir. Bunun için ağa daha önce görmediği yani eğitim setinde olmayan bir örnek gösterilir. Bu örnek eksik bilgiler içerebilir. Ağın görevi bu eksiklikleri gidermek ve örneğin tamamını hafızadan bulmaktır. Bunun için örnek ağa sunulmakta ve ağ durağan hale gelene kadar iterasyonlar yapılması gerekmektedir. Ağ durağan hale gelince ürettiği çıktı, ağın kendisine gösterilen girdiye cevabı

olarak görülmektedir. Girdi örneği $X \quad (X1, X2, X3, ...XN)$ başlangıç değerlerine atanmak üzere ağın iterasyonları yukarıda da gösterilen şu formüle göre devam etmektedir.

$$y_k(t+1) = sgn(\sum_{j=1, j'k}^{N} w_{jk} y_j(t) - \theta_{k)} \tag{4.53}$$

Bu formülün çalışabilmesi için girdi vektörü, ağın başlangıç çıktı değerleri olarak atanmaktadır. Yani,

$$x(0) = X = (X1, X2, X3, ...XN) \tag{4.54}$$

olarak alınmaktadır. Ağın durağan hale gelebilmesi için enerji fonksiyonunun değerinin en azlanması gerekmektedir. Bu enerji fonksiyonu;

$$\varepsilon(t) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} w_{ij}(y_i(t)y_t(t) + \sum_{i=1}^{N} y_i(t)\theta_i \tag{4.55}$$

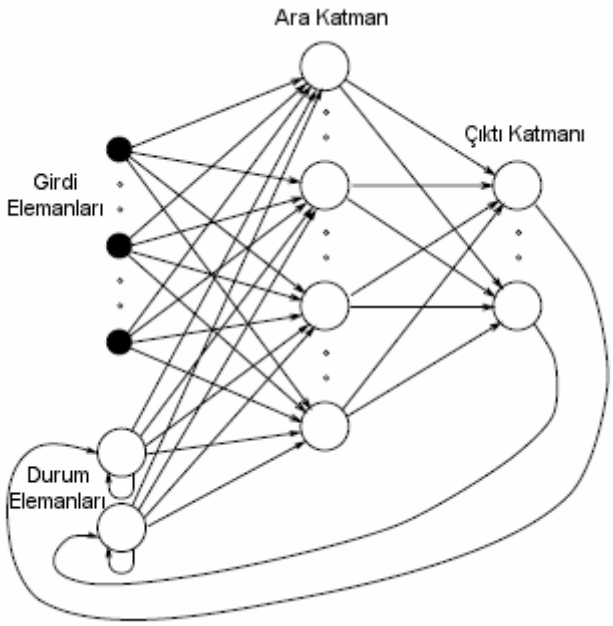
şeklinde verilmektedir. Ağ çalışırken bu enerji fonksiyonu ya azalır ya da değişmez. Dolayısıyla zaman içinde ağın minimum hata düzeyine ulaşması (durağan hale gelmesi) her durumda mümkün olmaktadır.

Hopfield ağlarının en önemli uygulamalarından birisi çözümü çok zor olan gezgin satıcı problemini çözmesidir. Bu problemde bir satıcı N adet şehire gitmek zorundadır. Bir şehre bir defa uğramak koşulu ile en kısa zamanda bütün şehirleri gezebilmesi için izlemesi gereken rotanın bulunması istenmektedir.

4.2.7 Jordan Ağı

Jordan ağı, çok katmanlı geri beslemeli (yinelemeli) bir yapay sinir ağıdır. Çok katmanlı perceptronlara benzer bir yapıda olan Jordan ağlarında, girdi, çıktı ve ara elemanlara ek olarak durum elemanları (state units) adı verilen

özel işlemci elemanlar da bulunmaktadır. Durum elemanları, çıktı tabakasından aldıkları aktivasyon değerlerini bir sonraki iterasyona girdi olarak taşımakla görevli elemanlardır. Diğer katmanlardaki işlemci elemanlar, çok katmanlı perceptronlardaki işlemci elemanlara benzer şekilde çalışırlar. Ara katman elemanları hem doğrusal hem de doğrusal olmayan aktivasyon fonksiyonlarına sahip olabilmektedir. Durum elemanları ile çıktı elemanları arasındaki bağlantı ağırlıkları sabittir ve durum tabakasındaki her işlemci elemanın kendisine bağlantısı bulunmaktadır. Çok katmanlı perceptronların kullandığı öğrenme kuralları, Jordan ağının eğitiminde de kullanılabilir (Kröse, van der Smagt, 1996).

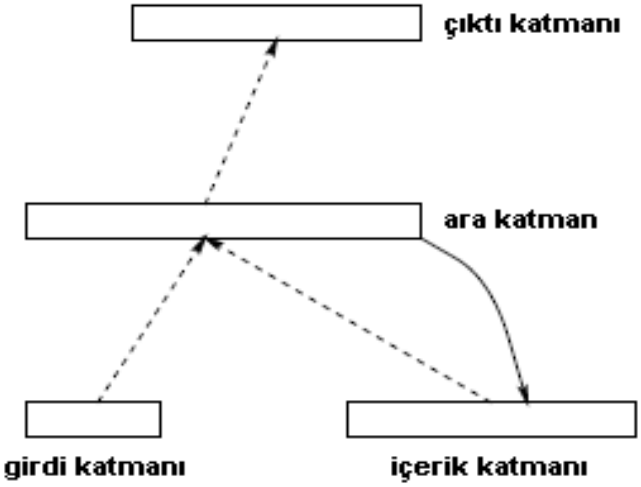


Şekil 4.12 Jordan Ağı (Kröse, van der Smagt, 1996)

4.2.8 Elman Ağı

Elman ağı, 1990 yılında Elman tarafından geliştirilmiştir. Elman ağı, girdi katmanı, çıktı katmanı, ara katmanlar ve bunlara ek olarak içerik katmanından (context layer) oluşmaktadır. İçerik elemanları, ara katmandan

aldıkları aktivasyon değerlerini bir sonraki iterasyona girdi olarak taşırlar. Elman ağları, Jordan ağına oldukça benzemekle birlikte iki önemli farka sahiptir. Bunlardan ilki geri besleme yaptıkları aktivasyon değerlerini çıktı katmanından değil ara katmandan almaları, ikincisi ise içerik elemanlarının kendilerine bağlantılarının bulunmamasıdır. Elman ağlarında da ara elemanlar ve içerik elemanları arasındaki bağlantı ağırlıkları sabit bir değer olmaktadır.



Şekil 4.13 Elman Ağı (Kröse, van der Smagt, 1996)

Elman ağının öğrenmesi, çok katmanlı perceptronların eğitiminde de kullanılan Genelleştirilmiş Delta öğrenme kuralına göre gerçekleşmektedir. Ara katmanda bulunan işlemci elemanlara gelen net girdi değeri, girdi katmanındaki elemanın değeri (u) ile ağırlığının (W^g) çarpılıp toplanması sonucunda bulunan değerlere içerik elemanından gelen bağlantı değerlerinin ara katmanlarının (W^i) bir önceki aktivasyon değerleri ile çarpılıp eklenmesi sonucunda bulunur. Bu değer bir fonksiyondan geçirilerek ara katman elemanlarının çıktısı (X) bulunmaktadır. Yani herhangi bir t zaman diliminde kullanılan aktivasyon fonksiyonunun sigmoid olması durumunda ara katman işlemci elemanlarının çıktıları,

$$x_i(t) = \frac{1}{1 + e^{s_i(t)}} \tag{4.56}$$

şeklinde hesaplanmaktadır. Burada hesaplanan net girdi (s) ise, yukarıda belirtildiği gibi ara katmanlardan gelecek olan geri beslemeler dikkate alınarak hesaplanmaktadır. Bilinen geri yayılım algoritmasında hesaplanan net girdi değerindeki farklılık buradadır. Yani Elman ağında,

$$s_k = W^g u(t) + W^i X(t-1) \tag{4.57}$$

olmaktadır. Bu formül açık olarak yazılırsa;

$$s_i(t) = \sum_{j=1}^{N} w_{ji}^g u(t) + \sum_{i=1}^{M} w_{ij}^i x(t-1) \tag{4.58}$$

formülü elde edilir. Burada N girdi katmanındaki işlemci eleman sayısını, M ise ara katmandaki işlemci eleman sayısını göstermektedir.

Elman ağlarında da kullanılan aktivasyon fonksiyonları türevi alınabilir fonksiyonlar olmak üzere tasarımcı tarafından seçilmekte ve genelleştirilmiş delta kuralına göre hem işlemci elemanların çıktıları hesaplanmakta , hem de ağırlıkların değişim miktarları belirlenmektedir. Ağın çıktısı ise çıktı katmanına gelen net değerin doğrusal fonksiyondan geçirilmesi ile belirlenir. Yani, çıktı katmanındaki işlemci elemanların aktivasyon fonksiyonları doğrusal olduğu için, herhangi bir t zamanında çıktı katmanındaki çıktı elemanının değeri, W^a ağırlık değerleri ile ara katman elemanlarının çıktıları (X) kullanılarak;

$$y(t) = W^a(t)X(t) \tag{4.59}$$

şeklinde hesaplanır. $t.$ zaman diliminde çıktı katmanındaki $j.$ elemanın çıktısı,

$$y_j(t) = \sum_{i=1}^{M} w_i^a(t) x_i(t) \tag{4.60}$$

'dır. Böylece ağın ileri doğru bilgi işlemesi tamamlanmış olur. Elman ağına sunulan herbir örnek için beklenen çıktı (d) belirli olduğu için, $t.$ zaman diliminde $j.$ çıktı elemanında oluşan hata (E_j);

$$E_j = d_j(t) - y_j(t) \tag{4.61}$$

olmaktadır. Bu hata, genelleştirilmiş delta kuralına göre değişebilen ağırlıklara dağıtılmaktadır. Önce bu hata değeri çıktı fonksiyonunun türevi ile çarpılarak ağırlıklara dağıtılacak hata oranları (δ) belirlenmektedir. Çıktı fonksiyonunun sigmoid olması durumunda $t.$ zaman diliminde ağırlıklara dağıtılacak olan hata,

$$\delta(t) = y(t) - \left[1 - y(t)\right] E(t) \tag{4.62}$$

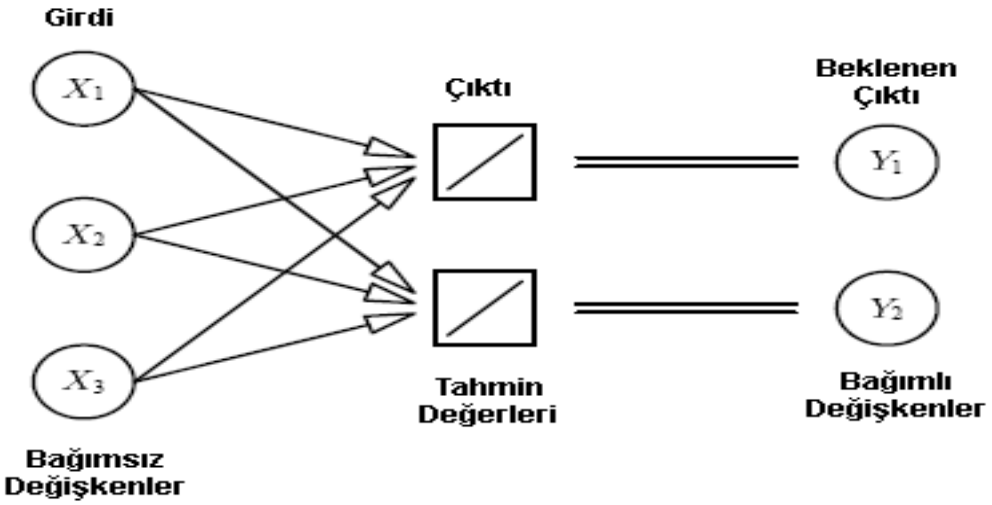
ile belirlenir. Ağırlıkların değişimi için önce değişimin miktarı hesaplanarak bu değişim miktarı ağırlıklara eklenir. 4.2.2.2'de Genelleştirilmiş Delta Kuralı'na göre ağırlıkların değişimi için verilmiş olan formüller Elman ağı için de geçerlidir. Burada dikkat edilmesi gereken nokta, geri dönüşüm ağırlıklarının değerlerinin sabit olduğu ve değiştirilmiyeceğidir. Yani ağırlıkların değerleri değiştirilirken geri dönüşüm ağırlıklarını dikkate almamak gerekir. Bu ağırlıklar ileri doğru bilgi işlerken içerik elemanlarının girdisini oluşturmada (geri besleme değerlerini girdi olarak içerik elemanlarına taşımada) kullanılırlar. Ağın ağırlıklarını değiştirirken geri dönüşüm bağlantı ağırlıklarının dikkate alınmaması ve içerik elemanlarının girdi elemanları

olarak düşünülmesi durumunda Elman ağı Çok Katmanlı Perceptron ile aynı olur (Öztemel, 2003).

4.3 Yapay Sinir Ağları ile İstatistik İlişkisi

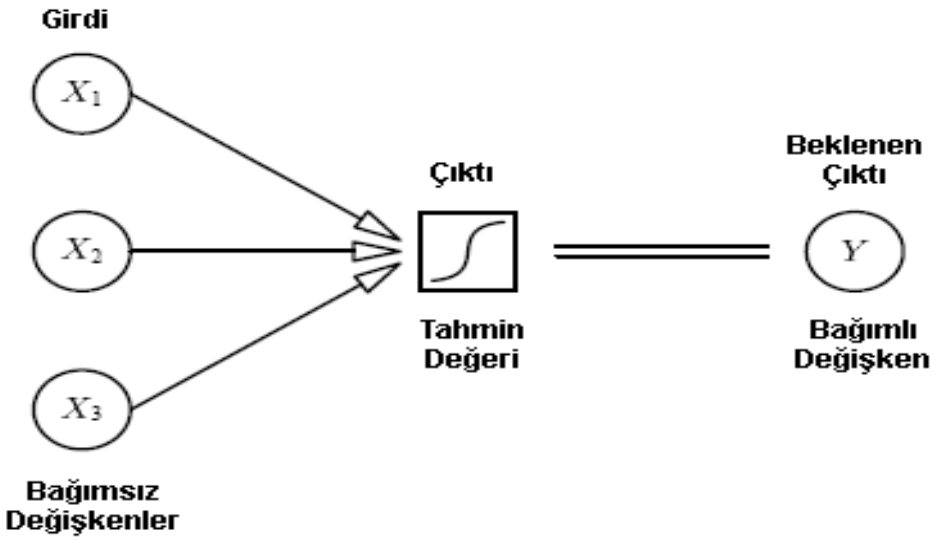
Yapay sinir ağları ve istatistik arasında sıkı bir ilişki bulunmaktadır. İstatistik veri analiziyle uğraşan bir alandır. Yapay sinir ağları terminolojisinde ise istatistiksel çıkarım, genelleme yapabilme yeteneğidir. Dolayısıyla yapay sinir ağları da bazı modelleri hariç genellikle veri analiziyle ilgilenir. Danışmansız öğrenme sınıfına giren Hopfield ağları gibi bazı modellerin istatistikle ilişkisi sınırlı olmakla beraber pek çok model, hatalı verilerden öğrenme ve genelleme yapabilme yetenekleri sayesinde istatistiksel yöntemlere oldukça benzemektedirler.

Yapay sinir ağları ve istatistiksel yöntemler arasındaki benzerliklere pek çok örnek gösterilebilir. Doğrusal perceptron, çok değişkenli çoklu doğrusal regresyona karşılık gelir. Bu ilişki, Şekil 4.14'de gösterilmektedir.



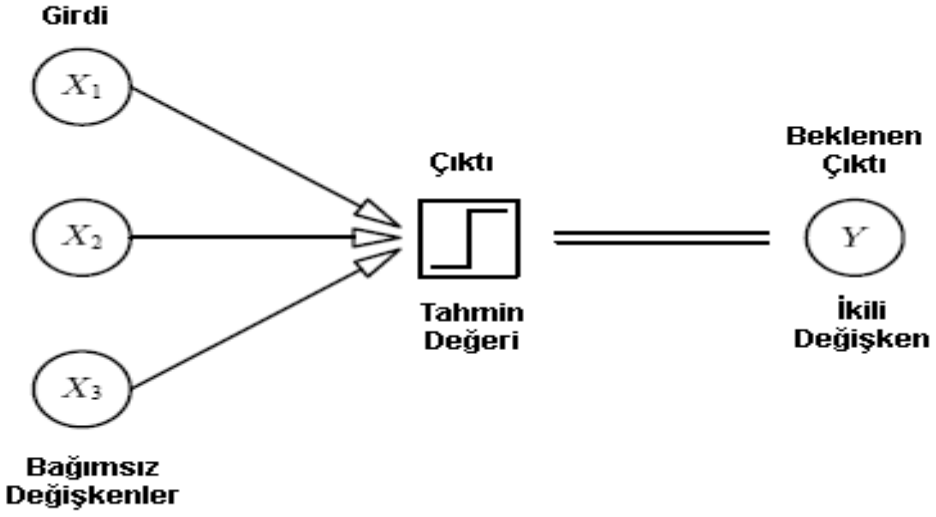
Şekil 4.14 Doğrusal Perceptron = Çok Değişkenli Çoklu Doğrusal Regresyon (Sarle, 1994)

Lojistik aktivasyon fonksiyonuna sahip perceptron ile lojistik regresyon arasındaki ilişki, Şekil 4.15'de görülmektedir.



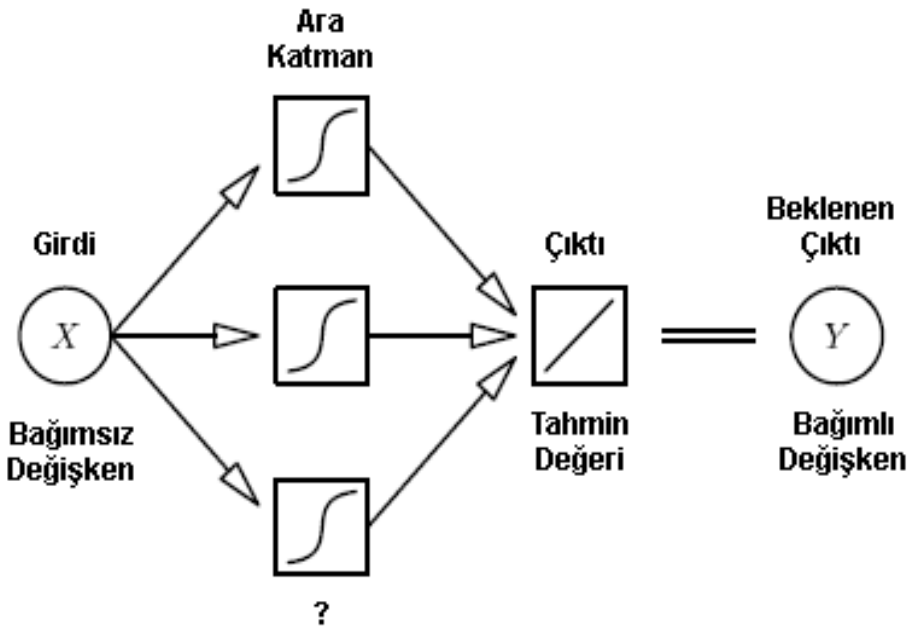
Şekil 4.15 Doğrusal Olmayan Perceptron = Lojistik Regresyon (Sarle, 1994)

ADALINE ile doğrusal diskriminant fonksiyonu arasındaki ilişki, Şekil 4.16'da görülmektedir.



Şekil 4.16 ADALINE = Doğrusal Diskriminant Fonksiyonu (Sarle, 1994)

Çok katmanlı perceptron ile doğrusal olmayan basit regresyon arasındaki ilişki, Şekil 4.17'de görülmektedir.



Şekil 4.17 Çok Katmanlı Perceptron = Doğrusal Olmayan Basit Regresyon (Sarle, 1994)

Bu örnekleri genişletmek gerekirse, olasılık tabanlı yapay sinir ağları temel fark analizine karşılık gelir ve Hebb öğrenme kuralı temel bileşen analizi ile oldukça ilişkilidir. Bunun yanında LVQ ağı ve Kohonen ağı gibi modellerin istatistiksel karşılıkları olmamasına rağmen veri analizi için kullanışlı olan modellerdir (Sarle, 1994).

İleri beslemeli ağlar, doğrusal olmayan regresyon ve ayrıştırma (discrimination) modellerinin oluşturduğu sınıfın bir alt kümesidir. Doğrusal olmayan modeller hakkındaki istatistiksel teorilerden elde edilen bir çok sonuç ileri beslemeli ağlara uygulanmaktadır. Ayrıca, doğrusal olmayan

modeller için kullanılan yöntemler, örneğin Levenberg-Marquardt algoritma, ileri beslemeli ağları eğitmek için kullanılmaktadır (Yurtoğlu, 2005).

Yapay sinir ağları genelde algoritmaları ve uygulamalarına göre tanımlanırken, istatistiksel yöntemler genellikle sonuçlarına göre tanımlanırlar. Örneğin aritmetik ortalama, formülünün ağ içinde kullanılmasıyla ileri beslemeli geri yayılım algoritmasında kolayca hesaplanabilir. Sonuç olarak aritmetik ortalama çıktı olarak elde edilir. Bu nedenle bir istatistikçi, aynı problemi çözmek için geri yayılım algoritması veya Levenberg-Marquardt algoritma gibi eğitme algoritmalarından istediğini ileri beslemeli bir ağ mimarisine uygulayarak sonuç elde edebilir. Bununla beraber, bir istatistikçi farklı eğitme kriterlerini farklı istatistiksel özelliklere sahip farklı tahmin yöntemleri olarak görür (Sarle, 1995).

Yukarıda anlatılanların gösterdiği gibi yapay sinir ağları ve istatistik arasında sıkı bir ilişki vardır. Bu çalışmanın uygulama kısmını oluşturan ekonomik değişkenlerin tahmini ve modellenmesi ise büyük ölçüde istatistiksel yöntemler kullanılarak yapılmaktadır. Bu yöntemlerden özellikle çoklu regresyon analizi, ekonomik değişkenlerin tahmin edilmesi ve karar vericiye politika saptamakta yol gösterici olması açısından çok fazla uygulamada başarılı bir şekilde kullanılmaktadır. Daha önce de belirtildiği gibi çoklu regresyon analizi yapay sinir ağlarında doğrusal perceptron modeline karşılık gelmektedir. Ancak bu çalışmanın uygulama bölümünde, hem doğrusal hem de doğrusal olmayan problemlerin çözümünde çok daha iyi sonuçlar verdiği için “İleri Beslemeli Geri Yayılım Ağı” kullanılmıştır.

BEŞİNCİ BÖLÜM
UYGULAMA

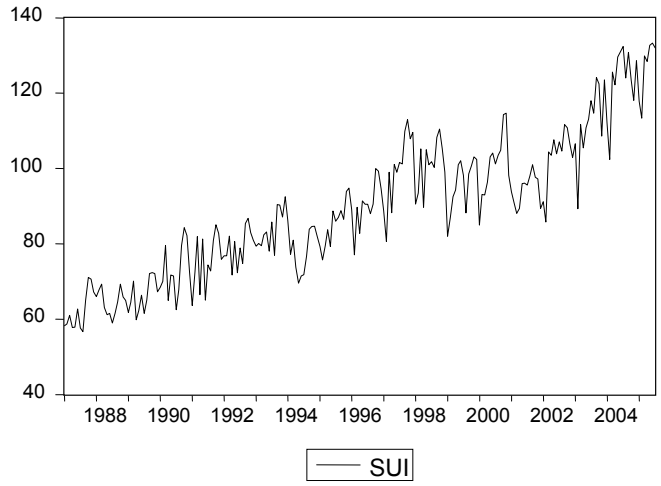
5.1 Verilerin Tanıtılması

Çalışmada, sanayi üretim indeksinin tahminine yönelik yapay sinir ağları modeli oluşturulmuş ve bu tekniğin tahmin gücüne yönelik karşılaştırma yapabilmek amacıyla çoklu doğrusal regresyon analizi uygulanmıştır. Karşılaştırılabilir sonuçlar elde edebilmek amacıyla benzer yapıda modeller oluşturulmuştur.

Uygulama için, sanayi üretim indeksi (1997=100 bazlı indeks, SUI) üzerinde etkili olduğu düşünülen bağımsız değişkenler; Dolar alış fiyatı (DOLAR), Toptan Eşya Fiyat İndeksi (1968=100 bazlı indeks, TEFE), para arzı (M2) ve mevduat faiz oranı (MFO) aylık değerler olarak belirlenmiştir. Sözkonusu değişkenlerle elde edilen çoklu regresyon modeli E.K.K. varsayımlarını sağlamadığı için düzeltici yöntemler uygulanmıştır. Otokorelasyon problemini gidermek için genelleştirilmiş farklar yöntemi, heteroskedasite problemi için ise, tam logaritmik dönüşüm yapılmıştır. Bu dönüşümlere ek olarak, çoklu doğrusal bağlantının giderilmesi için, bağlantıya yol açan değişkenler M2 ve MFO'nun elenmesi metodu izlenmiştir. Bu değişkenlerin modelden çıkarılması sonucu normallik varsayımının sağlanmadığı Jarque-Bera testi ile anlaşılmıştır. Bunun üzerine değişken seçimi konusunda birçok alternatif deneme sonucunda otoregresif model yapısının varsayımları sağladığı ve istatistiksel olarak anlamlı olduğu belirlenmiştir. Çoklu regresyon tahmin modeli belirlendikten sonra, YSA modeli aynı değişkenler için uygulanarak öngörü karşılaştırması yapılmıştır.

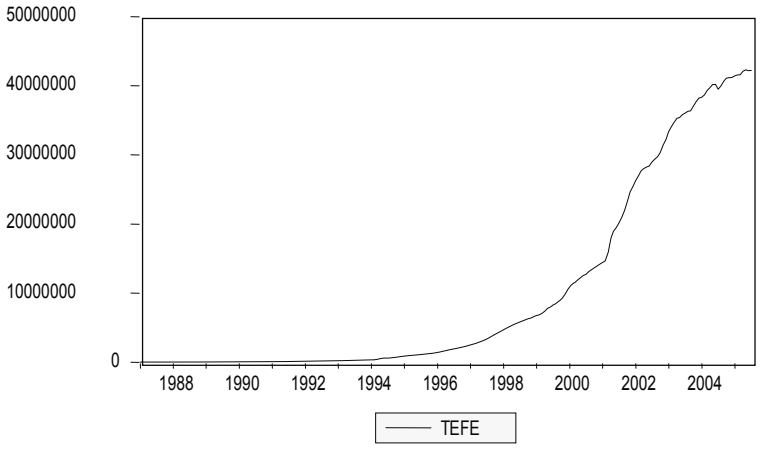
Veriler, T.C. Merkez Bankası internet sayfasından ve yine T.C. Merkez Bankası 3 aylık bültenlerinden derlenmiştir. Çalışmada, 1987 yılının ilk ayından, 2005 yılının temmuz ayına kadar olan (1987:01-2005:07) dönem ele alınmıştır.

İncelenen döneme ilişkin model değişkenlerin özelliklerinin belirlenmesi için grafik yorumları aşağıdaki gibidir.



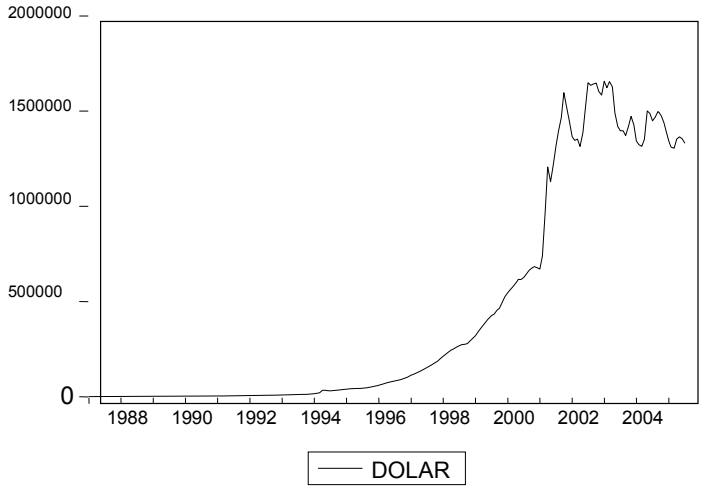
Şekil 5.1 Sanayi Üretim İndeksi serisi (SUI)

Şekil 5.1'deki grafikten, 1987 yılının Ocak ayından 2005 yılının Temmuz ayı sonuna kadar SUI serisinde hem yığılım hem de artan bir trend etkisi görülmektedir.



Şekil 5.2 Toptan Eşya Fiyat İndeksi serisi (TEFE)

Şekil 5.2'deki grafikten, 1987 yılının Ocak ayından 2005 yılının Temmuz ayı sonuna kadar TEFE serisinde 1994 yılından itibaren hızla artan bir trend görülmektedir.



Şekil 5.3 Dolar Alış Kuru serisi (DOLAR)

Şekil 5.3'deki grafikten, 1987 yılının Ocak ayından 2005 yılının Temmuz ayı sonuna kadar DOLAR serisinde artan bir trend etkisi ve 2000 yılında yapısal bir kırılmanın yanısıra bu dönem sonrası yığılım etkisi görülmektedir.

Zaman serisi özelliği taşıyan veri gruplarında durağanlaştırma işlemi yapılmadan regresyon modeli kurulduğunda E.K.K. varsayımları sağlanamaz. Burada uygulamacılar için iki alternatif sözkonusudur. Değişkenler durağan hale getirildikten sonra tek ya da çok değişkenli zaman serisi yöntemleri kullanılarak tahmin modeli elde etmek birinci yol olarak belirtilebilir. Diğer alternatif ise, zaman serileri yöntemleri yerine ekonometrik modelleme tekniklerinin seçilerek, varsayımların sağlanamayacağını başından kabul edip, daha sonradan model üzerinde belli dönüşümler ve tahminleme teknikleri kullanarak uygun tahmin modelinin belirlenmesidir. Bu çalışmada, ikinci alternatif benimsenerek uygulama gerçekleştirilmiştir. Ele alınan değişkenlerin trend etkisi ekonometrik modelde otokorelasyon problemine ve yığılım etkisi ise, heteroskedasiteye yol açacaktır. Çalışılan değişkenler zaman serisi özelliği taşıdığından çoklu regresyon modelinde hem otokorelasyon hem de heteroskedasite problemleriyle karşılaşılmış ve dönüşümler yoluyla model, tahmine uygun hale getirilmiştir.

5.2. Çoklu Regresyon Modeli

Çoklu regresyon modeli için genel fonksiyonel gösterim aşağıdaki gibidir.

$$LOGSUI = f(LOGSUI_{-1}, LOGTEFE_{-1}, LOGDOLAR_{-1}) \quad (5.1)$$

Modelin belirlenmesinde bazı teorik kısıtlar göz ardı edilmiştir. Bunun nedeni, çalışmanın bir politika analizi olmayıp, iki yöntem arasındaki performans karşılaştırması amacını taşımasıdır. Bu sebeple, bağımsız değişkenlerin belirlenmesinde, varsayımları sağlayan ve istatistiksel anlamlılığı veren az sayıda etkin değişken ele alınmıştır. Çünkü, yapay sinir ağları modelleri sınırsız sayıda bağımsız değişken kullanarak analiz yapabilmesine karşın, regresyon yöntemi çoklu doğrusal bağlantı probleminin oluşmaması için, sınırlı sayıda ve ilişkisiz bağımsız değişken kullanabilmektedir. Bu yüzden, her iki yöntem için en uygun modelleri bulmak yerine belirli bir modelle çalışılarak, karşılaştırma yapılması sağlanmıştır.

Eviews 4.1 yardımıyla elde edilen sonuçlar aşağıdaki gibidir.

Dependent Variable: LOG(SUI)
Method: Least Squares
Date: 04/26/06 Time: 14:01
Sample(adjusted): 1985:02 2003:07
Included observations: 222 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	1.546231	0.179788	8.600326	0.0000
LOG(DOLAR(-1))	-0.220393	0.036575	-6.025736	0.0000
LOG(TEFE(-1))	-0.266888	0.039321	-6.787441	0.0000
LOG(SUI(-1))	0.356136	0.062789	5.671955	0.0000

R-squared	0.898371	Mean dependent var	4.473974
Adjusted R-squared	0.896973	S.D. dependent var	0.210361
S.E. of regression	0.067521	Akaike info criterion	-2.534889
Sum squared resid	0.993894	Schwarz criterion	-2.473580
Log likelihood	285.3727	F-statistic	642.3551
Durbin-Watson stat	2.075472	Prob(F-statistic)	0.000000

Modelin bağımsız değişkenlerinin herbiri için $p<0.05$ olduğundan H_1 hipotezi kabul edilerek istatistiksel olarak anlamlı ve önemli sonucuna ulaşılır. Değişkenler iktisadi işaretleri açısından iktisat teorisine ters düşmemektedir.

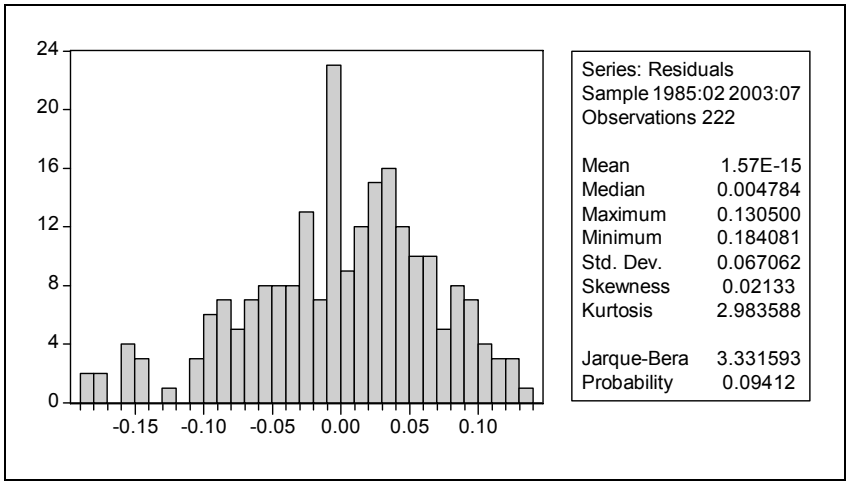
Modelin açıklayıcılığı %89,8 olarak elde edilmiştir. F testi sonucunda, $p<0.05$ olduğundan modelin anlamlılığını belirten H_1 hipotezi kabul edilmiştir. Daha sonra varsayımlar test edilerek modelin tahmine uygunluğu araştırılmıştır. İlk olarak otokorelasyon sınamasında LM test uygulanmıştır.

Breusch-Godfrey Serial Correlation LM Test:			
F-statistic	2.035934	Probability	0.155057
Obs*R-squared	2.063485	Probability	0.150865

LM testi sonucunda $p>0.05$ olduğundan H_0 kabul edilerek otokorelasyon olmadığı sonucuna ulaşılır. Heteroskedasite sınaması için White testi kullanılmıştır.

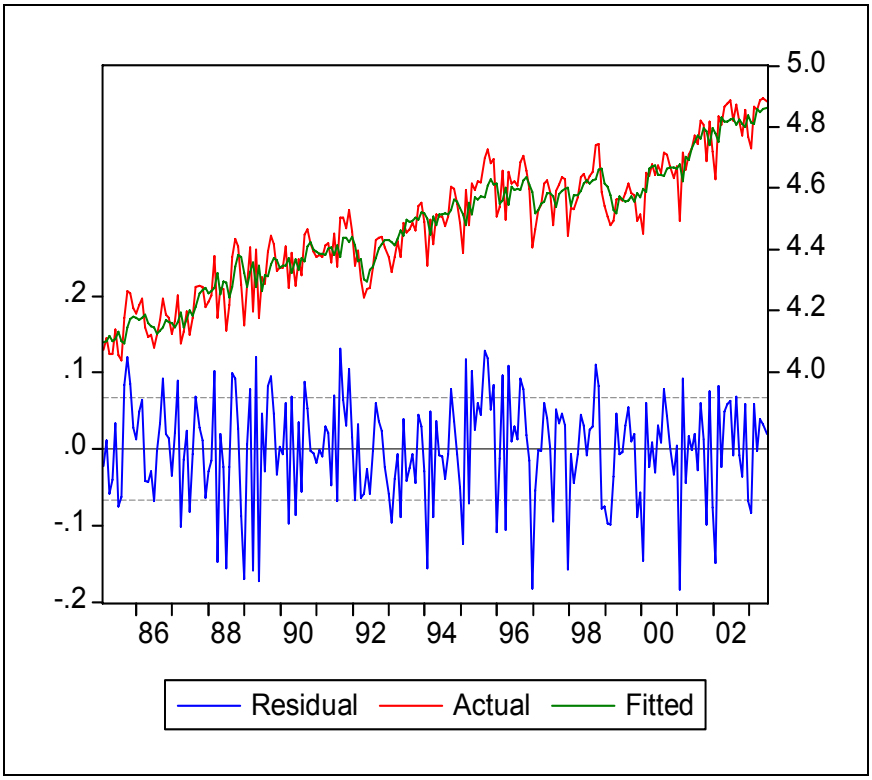
White Heteroskedasticity Test:			
F-statistic	1.470864	Probability	0.160315
Obs*R-squared	13.04748	Probability	0.160469

White testi sonucunda $p>0.05$ olduğundan heteroskedasite olmadığını belirten H_0 hipotezi kabul edilmiştir. Normallik varsayımının testi için Jarque-Bera (JB) testi kullanılmıştır.



JB test istatistiği $p>0.05$ olduğundan normal dağılımı belirten H_0 hipotezi kabul edilir.

Modelin hata paylarına ilişkin grafiği aşağıdaki gibidir.



Şekil 5.4 Çoklu Regresyon Modeli'nin Hata Paylarına İlişkin Grafik

Gerçekleşen ve tahmin edilen SUI serilerinin birbirine örtüşen yapıda olduğu ve aralarındaki sapmaların aşırılık göstermediği görülmektedir. Elde edilen bu modeli tahmine uygun olarak nitelemek yanlış olmayacaktır.

5.3 Yapay Sinir Ağları Modeli

Çalışmanın temel konusu olan yapay sinir ağları tekniği, Bölüm 1.5'de anlatıldığı gibi pek çok alanda başarıyla uygulanmaktadır. Ekonomik değişkenlerin modellenmesi de bu alanlardan bir tanesidir. Özellikle makro ve mikro ekonomik değişkenlere yönelik pek çok çalışma bulunmaktadır. Kuan ve White'ın (1994) makroekonomik değişkenlerin, Heravi, Osborn ve Birchenhall'ın (2002) üretim serilerinin yapay sinir ağlarıyla modellenmesine ve Moshiri ve Cameron'ın (1998) yapay sinir ağları kullanarak enflasyon

tahminine ilişkin çalışmaları bunlara örnek olarak gösterilebilir. West (2000) ise kredi taleplerinin yapay sinir ağları kullanılarak değerlendirilmesine ilişkin bir çalışma yapmıştır. Bunun yanında yapay sinir ağları ile Türkçe çalışmaların çok olmadığı görülmektedir. Az sayıdaki bu çalışmalara, Gülseçen'in (1993) işletme alanlarında yapay sinir ağları kullanımına, Üngör'ün (1998) döviz kuru tahmini ve Akpınar'ın (1993) kredi taleplerinin değerlendirilmesinde yapay sinir ağlarının kullanımı konusundaki çalışmaları örnek olarak verilebilir. Bu çalışmalarda genel olarak, ekonomik verilere ilişkin öngörülerde yapay sinir ağlarının diğer tekniklere göre daha başarılı olduğuna ilişkin kanıtlara işaret edilmektedir. Bu tür kanıtların Türkiye ekonomisine ait değişkenler için araştırılması da bu çalışmanın kapsamında bulunmaktadır.

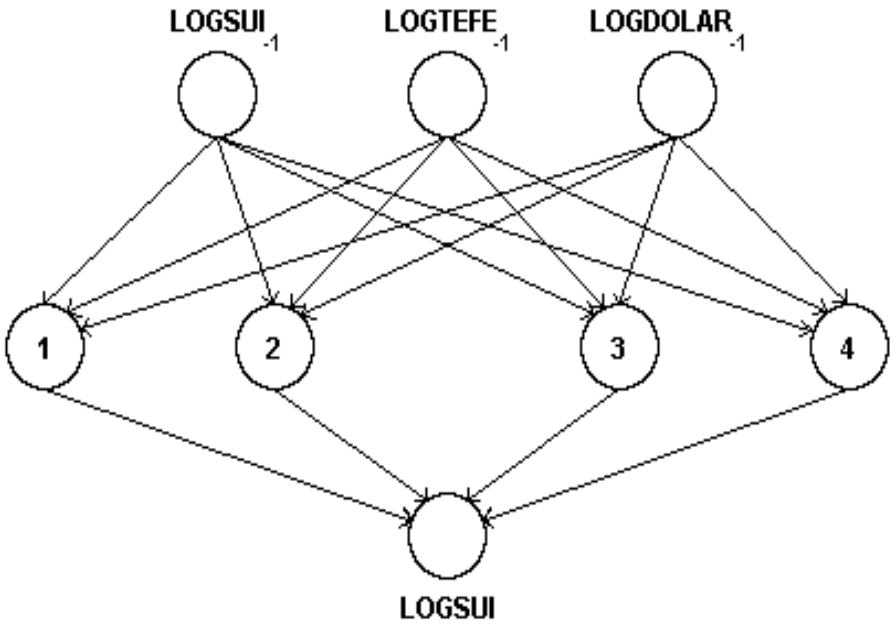
Uygulamada kullanılan yapay sinir ağları modeli, 4.2.2.2'de ayrıntılı olarak anlatılan İleri Beslemeli Geri Yayılım yapay sinir ağıdır. Danışmanlı öğrenme metodunu kullanan İleri Beslemeli Geri Yayılım yapay sinir ağının bu çalışmada tercih edilmesinin nedeni, ekonometrik verilerin modellenmesi ve öngörü çalışmalarında en çok kullanılan yöntem olması ve hem doğrusal hem de doğrusal olmayan yapıların modellenmesinde gösterdiği öngörü başarısıdır. Ayrıca kullanım kolaylığı da diğer bir tercih edilme nedenidir.

Kurulan model, Dr. İbrahim Sönmez tarafından Fortran Programlama Dili kullanılarak oluşturulmuş bir algoritma yardımıyla gerçekleştirilmiştir.

Modelde, bağımlı değişken olarak daha önce de belirtildiği gibi aylık Sanayi Üretim İndeksi'nin doğal logaritması kullanılmıştır. Modeldeki bağımsız değişkenler ise, Dolar Alış Kuru ve Toptan Eşya Fiyat İndeksi' nin doğal logaritmalı birinci gecikme değerleridir. Ayrıca bunlara ek olarak aylık Sanayi Üretim İndeksi'nin doğal logaritmalı birinci gecikme değerleri bağımsız değişken olarak modele eklenmiştir. Yani model, üç girdi değişkenine

$(LOGSUI_{-1}, LOGTEFE_{-1}, LOGDOLAR_{-1})$ ve bir çıktı değişkenine $(LOGSUI)$ sahiptir.

Bu üç girdi değişkeni ve bir çıktı değişkenine sahip modeli tahmin etmek için tek ara katmana sahip bir İleri Beslemeli Geri Yayılım yapay sinir ağı oluşturulmuştur. Şekil 5.5'de görülen bu mimarinin girdi katmanında girdi değişkenlerinin ağa sunulmasını sağlayan 3 adet işlemci eleman ve çıktı tabakasında ise bağımlı değişkene ait ağ çıktısının alındığı 1 adet işlemci eleman bulunmaktadır. Ara katmandaki işlemci eleman sayısı için ise daha önce belirtildiği gibi herhangi bir kısıt kullanılmamıştır. Ancak değişik sayıdaki işlemci elemanlarla yapılan denemeler sonucunda, ara katmandaki işlemci eleman sayısının 4 adet olmasına karar verilmiştir.

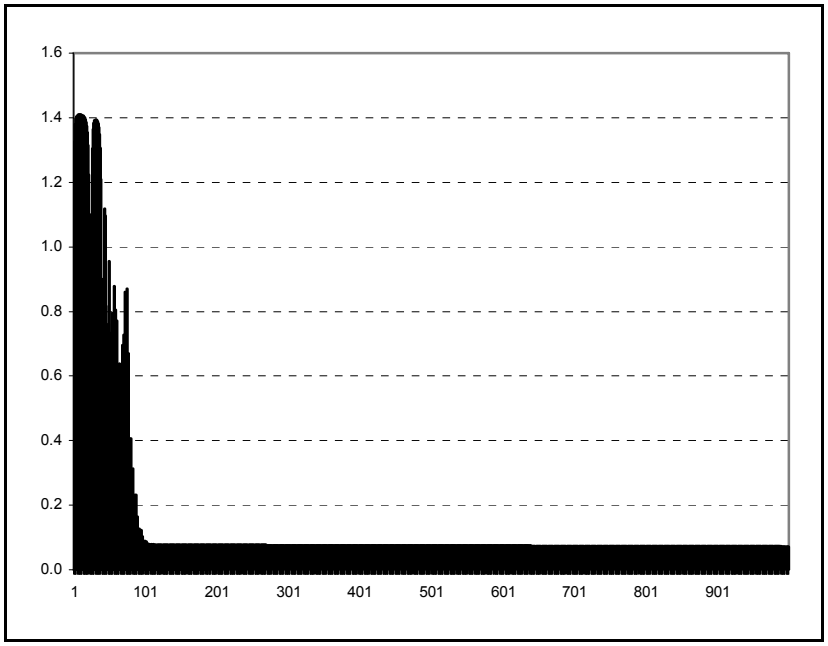


Şekil 5.5 Yapay Sinir Ağları Modeli'nin Mimarisi

Oluşturulan yapay sinir ağları modelinde işlemci elemanların kullandığı toplama fonksiyonu olarak doğrusal toplama fonksiyonu kullanılmıştır. Katmanların çıktı değerlerini hesaplayan aktivasyon fonksiyonu olarak ise hem ara katmanda hem de çıktı katmanında sigmoid aktivasyon fonksiyonu tercih edilmiştir. Bu tercihler yapılırken, ara katmandaki işlemci eleman sayısının belirlenmesinde olduğu gibi bir performans değerlendirmesinin sonuçları gözönüne alınmıştır.

Modelin mimarisinin oluşturulmasının ardından, eğitme sürecine geçilmiştir. Eğitme sürecinin başlangıcında ağın bağlantı ağırlıkları rassal olarak atanmıştır. Daha sonra Ortalama Mutlak Hata - OMH (Mean Absolute Error – MAE) değerlerinin minimize edilebilmesi için bağlantı ağırlıklarının ayarlanması (öğrenme) işlemi gerçekleştirilmiştir. Eğitme süreci için eğitim seti olarak bütün örneklem seti kullanılmıştır. Bu karar, karşılaştırma yapılacak ekonometrik model olan çoklu regresyon modelinin tüm örneklerin kullanılarak oluşturulması nedeniyle verilmiştir.

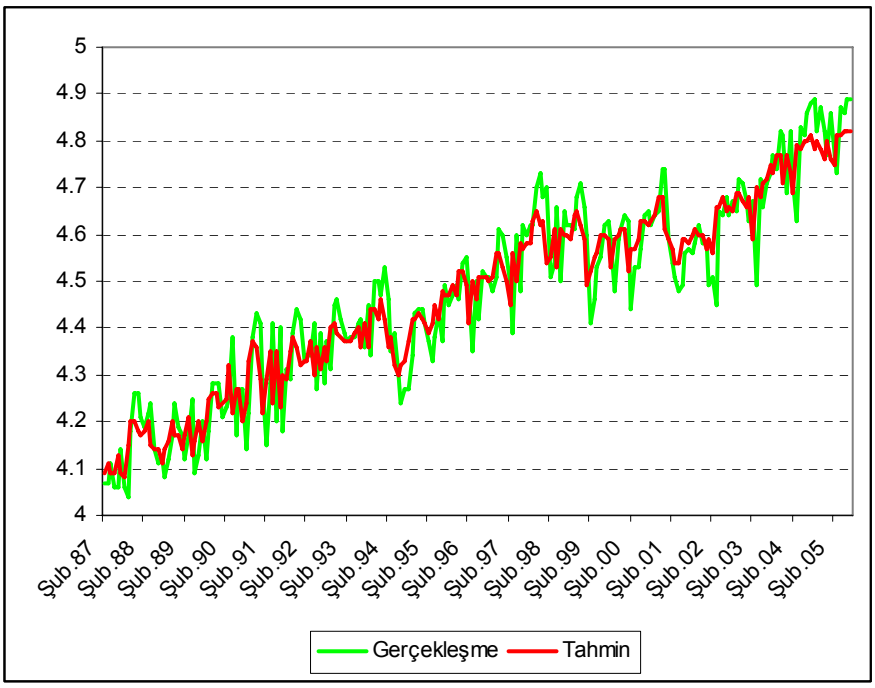
Model, 1.000 döngü (epoch) kullanılarak eğitilmiştir ve MAE fonksiyonunun bu eğitme işlemi boyunca nasıl minimize edildiği Şekil 5.6'deki grafikte gösterilmiştir. Grafikten görüldüğü gibi, eğitim sürecinde Ortalama Mutlak Hata değerleri ilk döngülerde hızlı olmak üzere giderek azalmış ve 0.0723 civarında sabitlenmiştir. Böylece ağın eğitimi tamamlanmıştır.



Şekil 5.6 Eğitim sürecinde Ortalama Mutlak Hata değerlerinin değişimi

Ağın eğitiminin tamamlanması daha önce de belirtildiği gibi ağın ağırlıklarının belirlenmesi anlamına gelir.

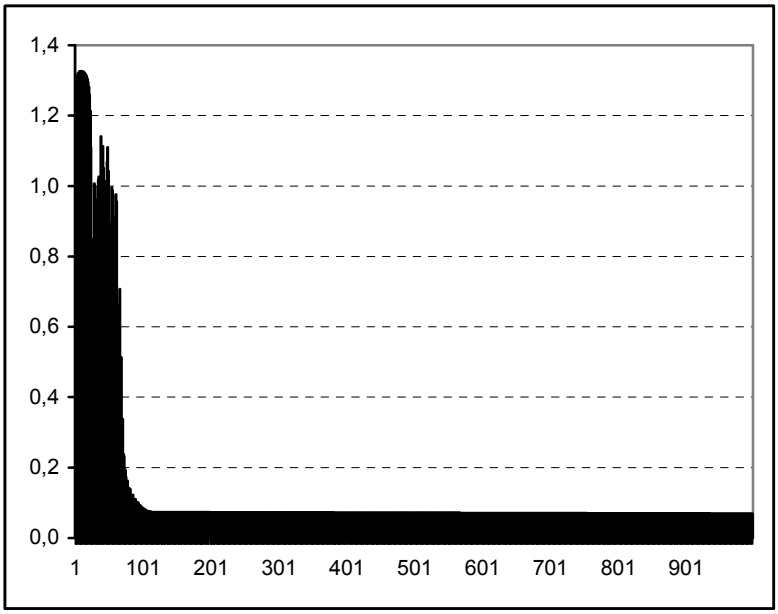
Kurulan modelin elde ettiği tahmin değerleri ile gerçekleşen değerler karşılaştırmalı olarak Şekil 5.7'da sunulmuştur.



Şekil 5.7 Yapay Sinir Ağları Modeli'nin Tahminleri ve Gerçekleşen Değerler

Şekil 5.7'de de görüldüğü gibi modelin elde ettiği tahmin değerleri gerçekleşen değerlere yakın ve oldukça başarılıdır.

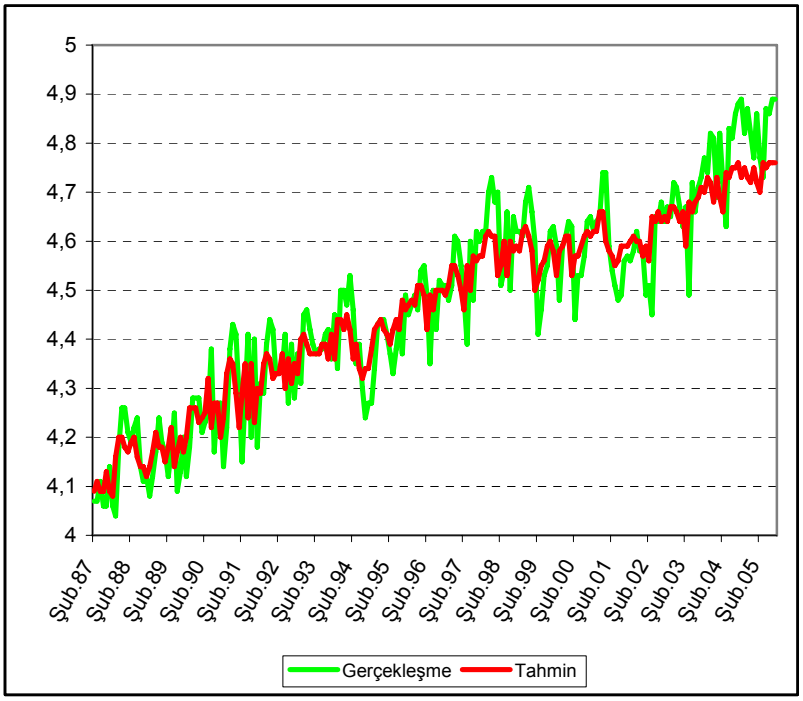
Bu model daha önce de belirtildiği gibi çoklu regresyon modeliyle karşılaştırılabilmesi için tüm veri eğitime alınarak kurulmuştur. Ayrıca çalışmada karşılaştırma amacıyla kullanılmayacak ancak ek bir bilgi vermesi amacıyla verinin bir kısmının "test seti" olarak ayrıldığı bir model daha kurulmuştur ve herhangi bir karışıklık olmaması için bu modele YSA-1 adı verilmiştir. Bu modelde Ocak 1987 – Ağustos 2003 (1987:01-2003:08) yılları arasındaki veriler "eğitim seti" olarak alınarak ağın eğitiminde kullanılmış, Eylül 2003 – Temmuz 2005 (2003:09-2005:07) yılları arasındaki veriler ise "test seti" olarak alınmıştır. Yine bir önceki model gibi 3 girdi işlemci elemanı bir çıktı işlemci elemanına sahip bu modelde de ara katman tek bir tanedir ve bu katmandaki işlemci eleman sayısı 4 olarak belirlenmiştir. Yine 1.000 döngü ile eğitilen bu modelin Ortalama Mutlak Hata grafiği Şekil 5.8'de görülmektedir.



Şekil 5.8 YSA-1 Modeli'nin eğitim sürecinde Ortalama Mutlak Hata değerlerinin değişimi

Burada da ilk modelde olduğu gibi Ortalama Mutlak Hata değerleri ilk döngülerde hızlı olmak üzere giderek azalmış ve 0.0691 civarında sabitlenmiştir.

Kurulan bu ikinci modelin elde ettiği tahmin değerleri ile gerçekleşen değerler karşılaştırmalı olarak Şekil 5.9'de sunulmuştur.



Şekil 5.9 YSA-1 Modeli'nin Tahminleri ve Gerçekleşen Değerler

Şekil 5.9'da da görüldüğü gibi bu ikinci yapay sinir ağı modeli de oldukça başarılı sonuçlar vermiştir. Ayrıca test seti olarak ve ağın daha önce görmediği verilere ilişkin tahmin değerleri de oldukça başarılıdır. Bu da ağda bir ezberleme probleminin olmadığını göstermektedir.

Burada belirtilmesi gereken bir diğer nokta da yapay sinir ağları yönteminde verilerin herhangi bir dönüşüme gerek duymadığıdır. Bu çalışmada karşılaştırma yapılacak olan çoklu regresyon modeliyle aynı yapıda bir model kurulması gerektiği için verilerin doğal logaritmaları kullanılmıştır. Ancak veriler herhangi bir dönüşüme uğramadan yapılan bir denemede de, yapay sinir ağları modelinin oldukça iyi sonuçlar verdiği görülmüştür. Bunun nedeni yapay sinir ağlarının doğrusal olmayan yapısıdır.

5.4 Model Karşılaştırması

Öngörü modellemesi, diğer birçok alanda olduğu gibi ekonomi alanında da büyük bir önem taşımakta ve yaygın bir şekilde kullanılmaktadır. Öngörü

modellemesinin ekonomi alanındaki öneminin temel nedeni, karar mekanizmasında önemli rol oynamasıdır. İyi öngörüler iyi kararlar alınmasına neden olmaktadır. Bu noktada iyi öngörülerin elde edilebilmesinin önemi ortaya çıkmaktadır. Bu yönde, öngörü modellemesi metodlarının geliştirilmesine yönelik çalışmalar yapılmaktadır.

Bu çalışmanın konusunu oluşturan yapay sinir ağları metodu ekonomi alanında ve başka birçok alanda öngörü modellemesi için geliştirilen yeni bir yöntemdir. Bundan önceki bölümlerde teorisi ve uygulaması anlatılan yapay sinir ağlarının bu bölümde çoklu doğrusal regresyon analizi ile bir karşılaştırması yapılacaktır. Bu karşılaştırma için, uygulamanın bundan önceki bölümlerinde tahmin edilen modeller için bazı öngörü değerlendirme analizleri yapılacaktır.

Öngörü doğruluğunu ölçmek için pek çok ölçüt vardır. Bunlardan bazılarının avantajları üzerinde fikir birliğine varılmıştır (Clements ve Hendry, 1993). Bu doğruluk ölçütlerinin temelini, sapmayı ölçmek için kullanılan Ortalama Hata (Mean Error – ME) kavramı oluşturur ve gösterimi,

$$ME = \frac{1}{T}\sum_{t=1}^{T} e_{t+k,\,t} \tag{5.2}$$

şeklindedir. Burada T öngörü aralığını, t zamanı, k öngörü uzunluğunu ve e öngörü hatasını gösterir. Ortalama Hata kavramını temel alan ve yaygın olarak kullanılan diğer ölçütler arasında Ortalama Hata Kareleri (Mean Squared Error – MSE), Ortalama Hata Kareleri Kökü (Root Mean Squared Error – RMSE) ve Ortalama Mutlak Hata (Mean Absolute Error – MAE) sayılabilir. Bu ölçütlerin formülleri aşağıda verilmiştir.

$$MSE = \frac{1}{T}\sum_{t=1}^{T} e_{t+k,\,t}^{2} \tag{5.3}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T} e_{t+k,t}^2} \tag{5.4}$$

$$MAE = \frac{1}{T}\sum_{t=1}^{T} \left| e_{t+k,t} \right| \tag{5.5}$$

Ayrıca, yukarıda anlatılan doğruluk ölçütlerinde öngörü hatası (e) yerine öngörü hatasının yüzdesi $(p_{t+k,t} = (y_{t+k} - \overset{\Lambda}{y}_{t+k,t}) / y_{t+k})$ kullanılabilmektedir. Bu durumda ölçütler Ortalama Yüzde Hata (Mean Percent Error – MPE), Ortalama Yüzde Hata Kareleri (Mean Squared Percent Error – MSPE), Ortalama Yüzde Hata Kareleri Kökü (Root Mean Squared Percent Error – RMSPE) ve Ortalama Mutlak Yüzde Hata (Mean Absolute Percent Error – MAPE) olarak adlandırılmaktadır ve formülleri aşağıdaki gibidir.

$$MPE = \frac{1}{T}\sum_{t=1}^{T} p_{t+k,t} \tag{5.6}$$

$$MSPE = \frac{1}{T}\sum_{t=1}^{T} p_{t+k,t}^2 \tag{5.7}$$

$$RMSPE = \sqrt{\frac{1}{T}\sum_{t=1}^{T} p_{t+k,t}^2} \tag{5.8}$$

$$MAPE = \frac{1}{T}\sum_{t=1}^{T} \left| p_{t+k,t} \right| \tag{5.9}$$

Bu çalışmada tahmin edilen çoklu doğrusal regresyon modeli ve yapay sinir ağları modeline ilişkin yukarıda verilen öngörü doğruluk ölçütlerinin değerleri aşağıdaki tablolarda görülmektedir.

Tablo 5.1 Öngörü Hatasının Doğruluk Ölçütleri

Öngörü Hatası Doğruluk Ölçütleri				
	ME	**MSE**	**RMSE**	**MAE**
Çoklu Regresyon Modeli	0,0002	0,0521	0,0747	0,651
Yapay Sinir Ağları Modeli	0,0001	0,0031	0,0521	0,0213

Tablo 5.2 Öngörü Hatasının Yüzde Doğruluk Ölçütleri

Öngörü Hatasının Yüzde Doğruluk Ölçütleri				
	MPE	**MSPE**	**RMSPE**	**MAPE**
Çoklu Regresyon Modeli	-0,0012	0,0067	0,049	0,071
Yapay Sinir Ağları Modeli	-0,0001	0,0041	0,0038	0,044

Her iki tabloda da görüldüğü gibi bütün ölçütler, yapay sinir ağları modeli için daha küçük değerler vermiştir. Bu da kurulan iki modelden, yapay sinir ağları modelinin çoklu doğrusal regresyon modeline göre daha başarılı olduğunu göstermektedir.

Sonuçlar incelendiğinde, Ocak.1987 – Temmuz.2005 döneminin öngörüsü için kurulan iki modelin analizleri sonucunda yapay sinir ağları metodunun çoklu doğrusal regresyon analizi metoduna göre daha etkin bir performans sergilediği görülmektedir.

5.5 Sonuç

Bu çalışmada, öngörü modellemesi metodu olarak yapay sinir ağları yönteminin kullanımı araştırılmıştır. Bu doğrultuda, yapay sinir ağları teorisi detaylı bir şekilde incelenmiş ve çeşitli alanlarında yaygın olarak kullanılan yapay sinir ağları modelleri tanıtılmıştır. Uygulama, Türkiye'deki sanayi üretim indeksinin tahmin modelinin oluşturulmasına yöneliktir. Çoklu regresyon modeli ve yapay sinir ağları yöntemiyle elde edilen modellerin performansı öngörü doğruluğu ölçütleri kullanılarak karşılaştırılmıştır.

Yapay sinir ağlarının pek çok modelleri arasından, yaygın olarak kullanılması ve başarılı sonuçlar veren bir model olması nedeniyle İleri Beslemeli Geri Yayılım Ağı seçilmiştir. Yapay sinir ağı modelinin öngörü performansıyla bir karşılaştırma yapılabilmesi ve yorumlanabilir olması için çoklu doğrusal regresyon analizi aynı değişkenler kümesine uygulanmıştır. Modellerin performanslarını karşılaştırabilmek amacıyla kullanılan öngörü doğruluğu ölçütlerinin sonuçları ise, yapay sinir ağları modelinin daha etkili bir öngörü tekniği olduğunu göstermiştir.

Sonuç olarak; ezberleme, mimarinin hatalı oluşturulması gibi problemleri olmayan bir İleri Beslemeli Geri Yayılım Ağı, öngörü gücü yüksek bir ekonometrik model olan çoklu doğrusal regresyon analizine göre daha yüksek bir öngörü performansına sahiptir. Doğrusal olmayan bir modelleme tekniği olan yapay sinir ağları yönteminin doğrusal yöntemlerle karşılaştırılması durumunda, Türkiye ekonomisine ait birçok değişken için doğrusal olmayan modellemenin daha etkili olacağı biçiminde genellemeye gitmek mümkün olacaktır.

KAYNAKLAR DİZİNİ

Adıgüzel, F., 1999, A Comparative Study of Artificial Neural Network and the Alternative Statistical Methods, A Master's Thesis, The Graduate School of Natural and Applied Sciences of The Middle East Technical University, Ankara.

Akpınar, H., 1993, Yapay Sinir Ağları ve Kredi Taleplerinin Değerlendirilmesinde Bir Uygulama Önerisi, Yayınlanmamış Araştırma, İstanbul Üniversitesi İşletme Fakültesi, İstanbul.

Aksel, F., 2000, Regresyon Analizi ve Yapay Sinir Ağı Yöntemleri ile Uzun Dönem Yük Tahmini, Basılmamış Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Anderson, D.and McNeill, G., 1992, Artificial Neural Networks Technology, Kaman Sciences Corporation, New York.

Aydın, B., 2002, Basit ve Geri Yayılımlı Yapay Sinir Ağları, Uygulama Alanları, Basılmamış Yüksek Lisans Tezi, Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Çanakkale.

Büyük Larousse, 24. cilt, sayfa 12726.

Chatterjee S., Laudato, M., 1995, “Statistical Applications of Neural Networks”, Northeastern University Boston, Massachusetts.

Clements, M.P. and Hendry, D.F., 1993, “On the Limitations of Comparing Mean Squared Forecast Errors”, Journel of Forecasting, 12, 617-638.

Elmas, Ç., 2003, Yapay Sinir Ağları (Kuram, Mimari, Eğitim, Uygulama), Seçkin Yayıncılık, Ankara.

Flexer, A.,1995, “Connectionists and Statisticians, Friends or Foes?”, http://www.ofai.at/cgi-bin/tr-online?number+95-06.

Freeman, J.A. and Skapura, D.M., 1991, Neural Networks Algorithm, Applications and Programming Techniques, Addison-Wesley Publishing Company.

Freeman, J.A., 1994, Simulating Neural Networks, Addison-Wesley Publishing Company.

Fröhlich, J., 1997, “Neural Net Overview”, http://fbim.fh-regensburg.de/~saj39122/jfroehl/diplom/e-1.html.

Gurney, K., 1996, “Computers and Symbols versus Nets and Neurons”, UCL Draft Papers, No:1, U.K.

Gülseçen, S., 1993, Yapay Sinir Ağları, İşletme Alanında Uygulanması ve Bir Örnek Çalışma, Basılmamış Doktora Tezi, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü İşletme Fakültesi Sayısal Yöntemler Anabilim Dalı, İstanbul.

Heravi, S., Osborn, D.R., Birchenhall C.R., 2002, “Linear versus Neural Network Forecasts for European Industrial Production Series”, http://ideas.repec.org/a/eee/intfor/v20y2004i3p435-446.html.

Jain.,A.K. and Mao J., 1996, “Artificial Neural Networks: A Tutorial”, IEEE Computer Special Issue on Neural Computing.

Kaashoek, J.F., Dijk, H.K., 2001, “Neural Networks as Econometric Tool”, Econometric Institute Report EI 2001-5.

Kayaönü, E., 2000, Yapay Zekanın Teorik Temelleri, Basılmamış Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi.

Kröse, B. and van der Smagt P., 1996, An Introduction to Neural Networks, The University of Amsterdam.

Kuan, C.M. and White, H., 1994, “Artificial Neural Networks: An Econometric Perspective”, Econometric Review 13, 1-91.

Kulkarni, A.D., 1994, Artificial Neural Networks for Image Understanding, Van Nostrand Reinhold, New York.

Linggard, R., Myers, D.J. and Nightingale, C., 1992, Neural Networks for Vision, Speech and Natural Language, Chapman&Hall.

Lisboa, P.G.J., 1992, Neural Networks Current Applications, Chapman&Hall.

Mainland, D.D., 1997, “Econometrics or Neural Networks? - A Case Study of Marketing Margins In The Meat Industry In The UK”, http://www.dina.dk/efita-conf/program/paperspdf/x_b_1.pdf.

Mehrotra K., Mohan C.K. and Ranka S., 1997, Elements of Artificial Neural Networks, The MIT Press, London.

Michie, D., Spiegelhalter D.J., Taylor C.C., 1994, Machine Learning, Neural and Statistical Classification, Elis Horwood, London.

Minsky, M.L. and Papert, S.S., 1969, Perceptrons, MA:MIT Press, Cambridge.

Moody, J., 1995, “Economic Forecasting: Challenges and Neural Network Solutions”, International Symposium on Artificial Neural Networks 1995.
Moshiri, S., Cameron, N., 1998, “Neural Network vs. Econometric Models in Forecasting Inflation”, Departman of Economics University of Manitoba.

Orhunbilge, N., 2002, Uygulamalı Regresyon ve Korelasyon Analizi, İ.Ü. Basım ve Yayınevi Müdürlüğü, İstanbul.

Öztemel, E., 2003, Yapay Sinir Ağları, Papatya Yayıncılık, İstanbul.

Ripley, B.D., 1997, “Can Statistical Theory Help Us Use Neural Networks Better?”, 29th Symposium on the Interface: Computing Science and Statistics.

Saraç, T., 2004, Yapay Sinir Ağları, Basılmamış Seminer Projesi, Gazi Üniversitesi Endüstri Mühendisliği Bölümü Anabilim Dalı, Ankara.

Sarle, W. S., 1994, “Neural Networks and Statistical Models”, in Proceedings of the Nineteenth Annual SAS Users Group International Conference, pp. 1538-1550, Cary, NC.

Setiono, R., Thong, J.Y.L., 2003, “An Approach to Generate Rules from Neural Networks for Regression Problems”, Europan Journals of Operational Research 2003.

Taşgetiren, M.F., 2005, “Çok Katmanlı Yapay Sinir Ağları”, http://bilim.ficicilar.name.tr/sayfa/Fatih_Tasgetiren-Cok_Katmanli_Yapay_Sinir_Aglari.html.

Uçar, N., 2001, Comparison of the Forecast Performances of Linear Time Series and Artificial Neural Network Models Within the Context of Turkish

Inflation, A Master's Thesis, The Departman of Economics Bilkent University, Ankara.

Üngör, A., 1998, Yapay Sinir Ağları ve Box-Jenkins Modeli Kullanarak Döviz Kuru Tahmini, Basılmamış Yüksek Lisans Tezi, Ortadoğu Teknik Üniversitesi.

West, D., 2000, “Neural Network Credit Scoring Models”, Computers and Operations Research 27 (2000) 1131-1152.

White, H., 1988, “Economic Prediction Using Neural Networks : The Case of IBM Daily Stock Returns”, In Proceedings of the IEEE International Conference of Neural Networks, July 1988.

Yurtoğlu, H., 2005, Yapay Sinir Ağları Metodolojisi İle Öngörü Modellemesi: Bazı Makroekonomik Değişkenler İçin Türkiye Örneği, Uzmanlık Tezi, Ekonomik Modeller ve Stratejik Araştırmalar Genel Müdürlüğü, Ankara.

XOR Probleminin Çözümü[1]

Yapay sinir ağları literatüründe önemli bir yere sahip olan XOR problemine İleri Beslemeli Geri Yayılım yapay sinir ağıyla çözüm üretilmiştir. XOR problemi için 4 örnek vardır ve her örnek için girdiler ve beklenen çıktı aşağıda sunulmuştur.

	Girdi 1 (x_1)	**Girdi 2** (x_2)	**Çıktı** (d)
Örnek 1	0	0	0
Örnek 2	0	1	1
Örnek 3	1	0	1
Örnek 4	1	1	0

XOR probleminde iki girdi ve bir de çıktı olduğu için, oluşturulacak olan İleri Beslemeli Geri Yayılım yapay sinir ağının girdi katmanında 2 adet işlemci eleman ve çıktı katmanında 1 adet işlemci eleman olacaktır. 1 ara katman ve 2 tane de ara katman işlemci elemanıyla problemin çözüleceği varsayılmaktadır. Ayrıca ara katman için 1 adet ve çıktı katmanı için de 1 adet eşik değer elemanı vardır.

Oluşturulan ağ için aktivasyon fonksiyonu olarak sigmoid fonksiyon kullanıldığı, öğrenme oranı (λ) ve momentum katsayısı (α) olarak da şu değerlerin belirlendiği varsayılsın,

λ =0.5
α =0.8

Oluşturulan ağ için ağırlık vektörleri ve başlangıç değerleri de şu şekilde belirlenmiş olsun.

[1] XOR probleminin çözümü (Öztemel, 2003)'den alınmıştır.

Girdi katmanı ile ara katman arasındaki ağırlıklar w^i matrisi ile gösterilsin;

$$w^i = \begin{bmatrix} 0.129952 & 0.570345 \\ -0.923123 & -0.328932 \end{bmatrix}$$

Çıktı katmanı ile ara katman arasındaki ağırlıklar ise w^a ile gösterilsin;

$$w^a = \begin{bmatrix} 0.164732 & 0.752621 \end{bmatrix}$$

Eşik değer ağırlıkları şöyle olsun.

$$\beta^a = \begin{bmatrix} 0.341332 & -0.115223 \end{bmatrix}$$

$$\beta^o = \begin{bmatrix} -0.993423 \end{bmatrix}$$

Birinci örnek $x_1 = 0$, $x_2 = 0$ ve $d = 0$ olarak belirlenmiştir. Bu örneğin ağa gösterilmesi sonucunda, ara katman işlemci elemanlarının net girdileri (eşik değer işlemci elemanının ağırlık değerleri eklenmiş olarak) şu şekilde hesaplanır.

s_1 = (0*0.129952)+(0*-0.923123)+(1*0.341232)=0.341232

s_2 = (0*0.570345)+(0*-0.328932)+(1*-0.115223)=-0.115223

Ara katman işlemci elemanlarının çıktı değerleri ise şöyle hesaplanır.

$$y_1 = \frac{1}{1+e^{-0.341232}} = 0.584490$$

$$y_2 = \frac{1}{1+e^{0.115223}} = 0.471226$$

Çıktı katmanındaki işlemci elemanın net girdisi hesaplanırsa;

s = (1*-0.993423)+(0.584490*0.164732)+(0.471226*0.752621)=-0.542484

değeri bulunur. Bu değer ile ağın çıktısı;

$$y = \frac{1}{1+e^{0.542484}} = 0.367610$$

olur. Beklenen çıktı 0 olduğuna göre ağın hatası: $E =$ 0-0.367610=-0.367610 olur. Bu hatanın geriye doğru yayılması sonucu ara katman ile çıktı katmanı arasındaki ağırlıkların değişim miktarları şu şekilde hesaplanır.

$$\delta_1 = y_1(1-y_1)E_1$$

δ_1 =0.367610*(1-0.367610)*(-0.367610)

δ_1 =-0.085459

$$\Delta w_{11}^a(t) = 0.5 * -0.085459 * 0.584490 + 0.8 * 0 = -0.024875$$

$$\Delta w_{21}^a(t) = -0.020135$$

$$\Delta \beta_1^o(t) = -0.042730$$

Ağırlıklardaki bu değişim miktarları ile ara katman ve çıktı katmanı arasındaki ağırlıklar yeniden hesaplanabilir.

$$w_{11}^a(t) = w_{11}^a(t-1) + \Delta w_{11}^a(t)$$

$w_{11}^a(t) =$ 0.164732-0.024975=0.139757

$w_{21}^a(t) =$ 0.752621-0.020135=0.732486

$\beta_1^o(t) =$ -0.0993423-0.042730=-1.036153

Benzer şekilde, girdi katmanı ile ara katman arasındaki ağırlıkların değişim miktarları ve yeni ağırlık değerleri hesaplanır. Ara katmandaki hata oranları ve değişim miktarları şu şekilde bulunur.

$\delta_1^a(t) = y_1(1-y_1)\delta_1 w_{11}^a(t-1)$

δ_1^a =0.584490*(1-0.584490)*(0.164732)*(-0.085459)

δ_1^a =-0.034190

δ_2^a =-0.160263

$\Delta w_{11}^i(t)$ =0.5*-0.034190*0+0.8*0=0

$\Delta w_{12}^i(t)$ =0.5*-0.034190*0+0.8*0=0

$\Delta w_{21}^i(t)$ =0

$\Delta w_{22}^i(t)$ =0

$\beta_1^a(t)$ = 0.5*1*-0.034190=-0.017095

$\beta_2^a(t)$ = 0.5*1*-0.160263=-0.080132

Bu değerler kullanılarak ağırlıklar değiştirilir. Ağırlıklardaki değişim miktarı 0 olduğu için ağırlık değerlerinde herhangi bir değişiklik olmayacak ancak eşik değeri ağırlıklarında değişiklik olacaktır.

$\beta_1^a(t)$ = 0.341232-0.017095=0.3242038

$\beta_2^a(t)$ = 0.115223-0.081325=-0.0350905

Birinci iterasyon bittikten sonra ikinci iterasyon başlayacaktır. Bu kez ikinci örnek ağa gösterilir. $y_1 = 0$, $y_2 = 1$ ve $d = 0$ olacaktır. Yukarıdaki işlemler aynı şekilde tekrar edilir. Bu iterasyonlar bütün çıktılar doğru cevap verinceye kadar devam etmelidir. Ağın öğrendikten sonraki ağırlık değerleri;

$$w^i = \begin{bmatrix} -6.072185 & -4.894898 \\ 2.454509 & 7.283063 \end{bmatrix}$$

$$w^a = \begin{bmatrix} 9.484580 & -4.473972 \end{bmatrix}$$

$$\beta^a = \begin{bmatrix} -6.062263 & -4.893081 \end{bmatrix}$$

$$\beta^o = \begin{bmatrix} -9.792470 \end{bmatrix}$$

Bu ağırlıklar ile girdiler ağa tekrar gösterildiğinde aşağıdaki sonuçlar elde edilir. Bu sonuçlar ağın problemi çok düşük hatalar ile çözebilecek şekilde öğrendiğini göstermektedir.

Girdi 1	**Girdi 2**	**Beklenen Çıktı**	**Ağın çıktısı**	**Hata**
0	0	0	0.017622	-0.017
1	0	1	0.981504	0.018
0	1	1	0.981491	0.018
1	1	0	0.022782	-0.020