YAPAY SİNİR AĞLARI

Prof.Dr. Ercan ÖZTEMEL

PAPATYA YAYINCILIK
İstanbul, Ankara, İzmir, Adana

BİLGİSAYAR SİS. SAN. VE TİC. A.Ş.
İnönü Cad. Hacıhanım Sok. 10/6, 80090, Gümüşsuyu/İstanbul

Tel : (212) 245 37 40, (532) 311 31 10
Faks : (212) 245 37 41
e-mail : bilgi@papatya.gen.tr
Web : http://www.papatya.gen.tr
http:// www.papatya.info.tr
Dağıtım : Toros Dağıtım Kitap (212 - 520 42 25)

Yapay Sinir Ağları – Ercan ÖZTEMEL

1. Basım Ağustos 2003.
2. Basım Ekim 2006. (Tıpkı Basım)

Akademik Danışman : Dr. Rifat ÇÖLKESEN (Beykent Üniversitesi)
Hakem Kurulu Başkanı : Dr. Cengiz UĞURKAYA
Türk Dili Uzmanı : Necdet AVCI
Üretim : Erdal GÜVEN
Sayfa Düzenleme : Papatya & Kelebek Tasarım (Olcay KAYA)
Kapak Tasarımı : Papatya & Kelebek Tasarım
Basım : Altan Basım Ltd.

Öztemel, Ercan.
Yapay Sinir Ağları / Ercan Öztemel. – İstanbul: Papatya Yayıncılık, 2006
xxii, 232s. ; 24 cm.
Kaynakça ve dizin var.
ISBN 975-67-97-39-8
1. Yapay Zeka. 2. Öğretmenli Öğrenme. 3. *Hopfield* Ağı. 4. *Kohonen* Katmanı. 5. LVQ Ağı
I. Title

En Büyük Şimak
Adına...

hayat arkadaşıma...

TEŞEKKÜR

Bu kitabın yazılmasında bana her türlü anlayışı gösteren başta eşim ve çocuklarım olmak üzere yanımda çalışan ve kendilerini zaman zaman ihmal ettiğim proje ekibime gösterdikleri her türlü anlayıştan dolayı teşekkür ederim.

Ayrıca kitabın basılması için beni cesaretlendiren Dr.Yalçın ÖZKAN'a ve kitabın titizlikle hazırlanmasına, dizgisine, tasarımına ve basımından dağıtımına kadar bütün aşamalarında emeği geçen Papatya Yayıncılık çalışanlarına, özellikle yayın ve akademik danışmanı Dr.Rifat ÇÖLKESEN'e teşekkür ederim.

Prof. Dr. Ercan Öztemel
Mart 2003 - İstanbul

İÇİNDEKİLER

ÖNSÖZ

Çağdaş dünyada bilgisayarlar ve bilgisayar sistemleri insan hayatının vazgeçilmez bir parçası haline gelmiştir. Elimizdeki cep telefonlarından, mutfaklardaki buzdolaplarına kadar birçok alet bilgisayar sistemi ile çalışmaktadır. İş dünyasından kamu işlerine, çevre ve sağlık organizasyonlarından askeri sistemlere kadar hemen hemen her alanda bilgisayarlardan yararlanmak olağan hale gelmiştir. Bunun aksini düşünmek bile teknolojinin nimetlerini hiçe saymak olarak görülmektedir. Teknolojinin gelişmesi izlendiğinde önceleri sadece elektronik veri transferi yapmak ve karmaşık hesaplamaları gerçekleştirmek üzere geliştirilen bilgisayarların zaman içerisinde büyük miktarlardaki verileri filtreleyerek özetleyebilen ve mevcut bilgileri kullanarak olaylar hakkında yorumlar yapabilen nitelikler kazandığı görülmektedir.

Bu çalışmaların temeline bakıldığında aslında bilgisayarlar, yüzyıllar boyunca insan beyninin nasıl çalıştığının merak edilmesi sonucunda ortaya çıkan ilkel hesap makinelerinin gelişmiş şekilleridir. Teknoloji geliştikçe insanlar bilgisayarların işlemleri hızlı yapmak gibi bazı özellikleri bu temel düşüncenin üstünü örtmüş ve insanlar teknolojinin geliştirilmesini teknolojiden elde edecekleri yarara bağlamışlardır. Bazı araştırmacılar ise ısrarla çalışmalarını insanın davranışlarının modellenmesi yolunda devam ettirmişler ve yapay zeka bilimi başta olmak üzere oldukça önemli gelişmeleri ortaya çıkartmışlardır.

1980'li yılların başlarına gelindiğinde aslında bilgisayar teknolojisinin günümüz ile kıyaslanamaz bir durumda olduğunu görüyoruz. O zamanlarda, insan beyninin bir benzeri yapılabilir mi? İnsan gibi davranabilen bir robot oluşturulabilir mi? Bilgisayarlar düşünebilirler mi? Olayları öğrenebilirler mi? vb. gibi soruların cevapları henüz bulunmamıştı. Aslında bu tür sorular yıllardır sürekli sorulmaktaydı. Sorulmaya da devam edilmiş ve sonuç olarak günümüzde bu soruların bazıları kısmen de olsa cevaplandırılmıştır. Hepsinin cevabı bulunamamıştır ama artık bilgisayarlar öğrenebilmektedir.

Diğer bir deyişle artık bilgisayarlar hem olaylar ile ilgili bilgileri toplayabilmekte, olaylar hakkında kararlar verebilmekte hem de olaylar arasındaki ilişkileri öğrenebilmektedir. Matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler bile sezgisel yöntemler yolu ile bilgisayarlar tarafından çözülebilmektedir. Bilgisayarları bu özellikler ile donatan ve bu yeteneklerinin gelişmesini sağlayan çalışmalar "yapay zeka" çalışmaları olarak bilinmektedir. İlk defa 1950'li yıllarda ortaya atılan yapay zeka terimi zaman içinde oldukça yoğun ilgi görmüş ve 40-50 yıllık bir zaman diliminde hayatın vazgeçilmez parçası olan sistemlerin doğmasına neden olmuştur. Yapay sinir ağları yapay zeka çalışmalarının da ivmesini artırmıştır. Bu teknoloji özellikle makine öğrenmesini sağlayan ve önemli gelişmelerin habercisi bir teknoloji olarak görülmüştür. Aslında sanayi toplumun bitip bilgi toplumunun başlamasına neden olan unsurlardan biriside yapay sinir ağları olmuştur.

Yapay sinir ağları, olayların örneklerine bakmakta onlardan ilgili olaya hakkında genellemeler yapmakta, bilgiler toplamakta ve daha sonra hiç görmediği örnekler ile karşılışınca öğrendiği bilgileri kullanarak o örnekler hakkında karar verebilmektedir. Bundan 15-20 yıl önce böyle bir şeyin olacağını düşünmek bile mümkün değildi. O zaman bilgisayarlar öğrenebilecek denilse idi bu hayal ürünü olarak görülürdü. Fakat zaman içinde gelişmeler

bunun bir hayal değil gerçek olabileceğini gösterdi. Gelecekte de bugün için hayal olarak görebileceğim bir çok yenilikler ortaya çıkacaktır.

1990'lı yıllardan beri bilgisayarların öğrenmesini sağlayan Yapay Sinir Ağları teknolojisinde oldukça hızlı bir gelişme görüldü. Bu teknoloji, kısa zamanda araştırmacıların dikkatlerini üzerine çeken bir bilim dalı olmayı başardı ve çalışmalar laboratuarlardan çıkarak günlük hayatın bir parçası haline gelmeye başladı. Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilen bilgisayar sistemler olduklarından hem yeni gelişmelere neden oluyor hem de nasıl çalıştığı bilinmeyen insan beyni hakkında yapılan araştırmalara da önemli katkılar sağlıyordu. Kısa bir zaman içinde,

Çok sayıda yapay sinir ağı modeli geliştirilmiş ve sayısız uygulama ortaya çıkmıştır. Gelişmeler bu sistemlerin gelecekte daha fazla insan hayatına gireceğini göstermektedir. Bu çalışmalar aslında insan beyninin nasıl çalıştığı ve öğrenme olayını nasıl gerçekleştirdiğini merak etme sonucunda ortaya çıkmıştır. İnsan beyninin nasıl çalıştığı günümüzde de henüz bilinmemektedir. Fakat yapılan çalışmalar ile bilgisayarların öğrenebildikleri ve başarılı sonuçlar ürettikleri görülmektedir. Özellikle çok sayıda bilginin değerlendirilmesini gerektiren olaylarda bu sistemler etkin olarak kullanılmaktadır. Endüstriyel hayattan finansal hayata, tıp biliminden askeri sistemlere kadar bir çok alanda uygulamalar görülmektedir. Bu uygulamalarda elde edilen başarılar hem yapay sinir ağlarının önemini artırmakta hem de bu sistemlere olan ilgiyi artırmaktadır. Her geçen gün bu kadar önemli olmasına rağmen Yapay sinir ağlarını ayrıntılı olarak anlatan Türkçe yazılmış bir eser olmadığı görülmüş ve bu amaçlar bu kitap yazılmıştır. Kitap içerisinde hem yapay sinir ağlarının felsefesi anlatılmış hem de bu teknolojinin teknik ayrıntıları verilmiştir. Yapay sinir ağlarında genel olarak ağa ne öğrenmesi gerektiğini söyleyen bir öğretmenin olup olmamasına göre değişen öğrenme stratejileri vardır. Kitabın içerisinde bu stratejilerin her birisi ile ilgili olarak bir yapay sinir ağının nasıl oluşturulabileceği, nasıl eğitileceği, nasıl test edileceği anlatılmıştır. Bu konuda okuyucunun dikkat etmesi gereken konularda açıklanmıştır. Bunun yanı sıra günümüzde en çok kullanılan ve özellikle endüstriyel ve sosyal hayatta kendisini göstermiş olan yapay sinir ağları da kısa kısa tanıtılmış ve özellikleri belirtilmiştir. Yapay sinir ağlarının uygulamaları genel olarak gözden geçirilmiş ve donanım olarak piyasa da ticari olarak geliştirilmiş sistemlerden örnekler verilerek bu teknolojinin sadece bir yazılım teknolojisi olmadığı aynı zamanda özel donanımlarında geliştirildiğine dikkatler çekilmiştir. Okuyucu bu kitabı okuyunca sadece kitapta anlatılan ağları öğrenmeyecek aynı zamanda diğer ağlar ile ilgili bilgileri bulabileceği kaynaklara da ulaşacaktır.

Kitabın okuyuculara yardımcı olmasını umarım.

Prof. Dr. Ercan Öztemel
Mart 2003 - İstanbul

BÖLÜM 1

YAPAY ZEKA VE MAKİNE ÖĞRENMESİNE GENEL BAKIŞ

Yapay sinir ağları yapay zeka biliminin altında araştırmacıların çok yoğun ilgi gösterdikleri bir araştırma alanıdır. Bilgisayarların öğrenmesine yönelik çalışmaları kapsamaktadır. Bu bölümde öncelikle yapay zeka bilimine genel bir giriş yapılarak bilgisayarların karar verme ve öğrenme niteliklerine dikkatler çekilecektir.

1.1. Yapay Zeka Teknolojisine Genel Bir Bakış

Çağdaş dünyada bilgisayarlar ve bilgisayar sistemleri yaşamın vazgeçilmez bir parçası haline gelmiştir. Elimizdeki cep telefonlarından, mutfaklardaki buzdolaplarına kadar birçok alet bilgisayar sistemi ile çalışmaktadır. İş dünyasından kamu işlerine, çevre ve sağlık organizasyonlarından askeri sistemlere kadar hemen hemen her alanda bilgisayarlardan faydalanmak olağan hale gelmiştir. Bunun aksini düşünmek bile teknolojinin nimetlerini hiçe saymak olarak görülmektedir. Teknolojinin gelişmesi izlendiğinde önceleri sadece elektronik veri transferi yapmak ve karmaşık hesaplamaları gerçekleştirmek üzere geliştirilen bilgisayarların zaman içerisinde büyük miktarlardaki verileri filtreleyerek özetleyebilen ve mevcut bilgileri kullanarak olaylar hakkında yorumlar yapabilen nitelikler kazandığı görülmektedir. Günümüzde ise bilgisayarlar hem olaylar hakkında karar verebilmekte hem de olaylar arasındaki ilişkileri öğrenebilmektedir. Matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler sezgisel yöntemler yolu ile bilgisayarlar tarafından çözülebilmektedir. Bilgisayarları bu özelliklerle donatan ve bu yeteneklerinin gelişmesini sağlayan çalışmalar "yapay zeka" çalışmaları olarak bilinmektedir. İlk defa 1950'li yıllarda ortaya atılan yapay zeka terimi zaman içinde oldukça yoğun ilgi görmüş ve 40-50 yıllık bir zaman diliminde hayatın vazgeçilmez parçası olan sistemlerin doğmasına neden olmuştur. Bu sistemler hem araştırmacılar hem de ticari olarak satıcılar tarafından "Zeki Sistemler" olarak adlandırılmaktadır. Zeki sistemlerin geliştirilmesinde yapay zeka biliminin katkısı çok fazladır. Benzer şekilde de zeki sistemlerdeki gelişmeler de yapay zeka biliminde gelişmelere neden olmaktadır.

Zeki sistemlerin en temel özellikleri olaylara ve problemlere çözümler üretirken veya çalışırken bilgiye dayalı olarak karar verebilme özelliklerinin olması ve eldeki bilgiler ile olayları öğrenerek, sonraki olaylar hakkında kararlar verebilmeleridir. Yapay zeka bilimindeki gelişmeler bu sistemlerde çeşitlendirmeye neden olmaktadır. Bir taraftan

donanım teknolojisi gelişmekte ve daha hızlı çalışabilen, daha çok bilgiyi saklayabilen, daha karmaşık sistemleri ve fonksiyonları yerine getiren bilgisayarlar ve bilgisayar sistemleri oluşturulmakta iken diğer taraftan yazlımı teknolojisi gelişmekte ve bilgi işleme yetenekleri, öğrenme, karar verebilme, problem çözme, muhakeme yapabilme yöntemleri ve bu yöntemlere dayalı yazılım sistemleri geliştirilmektedir. Bu gelişmeler ise zeki sistemlerin her geçen gün daha yaygın olarak bilinmesine ve gerçek hayatta insanların kullanımına alınmasına neden olmaktadır. Artık bilgisayarlar eskiden olduğu gibi sadece bilgi iletişiminin ve hesaplamaların otomasyonunu yapan sistemler olarak görülmemektedir. İnsan karar verme sürecine oldukça benzer bir karar verme sürecine kavuşmakta ve daha karmaşık fakat kullanışlı sistemler ortaya çıkmaktadır. Çalışmalar laboratuardan çıkmakta ve her geçen gün daha fazla ticari sistemler ortaya çıkmaktadır. İnsanlar olayları çözmek için her gün biraz daha zeki sistemler geliştirmek için yarışmaktadırlar. Bazı sistemler ise zeki olmadıkları halde işlemleri otomatik olarak gerçekleştirdiklerinden, zeki sistem diye tanıtılmakta ve pazarlanmaktadır. Bunlara otomasyon sistemleri demek daha doğru olur. Yeni çıkan zeki sistemler bu otomasyon sistemlerinin yerini almaktadır. Her geçen gün gerçek zeki sistemlerin sayısı artmakta ve zeki olmayan otomasyon sistemleri ortadan kalkmaktadır. Artık eskisi gibi otomasyon sistemlerine zeki sistemler denmemektedir. Sadece zeka ürünü olan fonksiyonları gerçekleştiren sistemler zeki sistemler olarak adlandırılmaktadır.

Bu gelişmeler ve ticari sistemlerin başarılı şekilde uygulanması yapay zeka teknolojisinin gelişimine ilgileri daha çok çekmiş ve yapay zeka önceleri sadece bir ilgi odağı iken bugün artık bir bilim dalı haline gelmiştir. Günümüzde üniversitelerde yapay zeka bölümleri açılmıştır. Yapay zeka mühendisleri (bilgi mühendisleri) yetiştirilmektedir. Yapay zeka bilimine genel bir bakış yapılırsa; Bu bilimin, bilginin organizasyonu, öğrenme, problem çözme, teorem ispatlama, bilimsel buluşların modellenmesi gibi bir çok konu ile ilgilendiği görülmektedir. Bu yetenekler ile donatılan bilgisayar sistemleri problemlere çözüm üretirken insanın problemleri çözme sürecini taklit etmektedir. Özellikle belirli bir algoritma veya formülasyon kullanılarak çözülemeyen problemlerin çözülmesi için yapay zeka sistemleri geliştirilmektedir. Problemin çözümünü sağlayan bir algoritma geliştirilmiş ise geleneksel bilgisayar sistemleri problemi çözmek için yeterli olmaktadır. Önemli olan problemin çözümünü veren bir formülün olmadığı durumlarda bilgisayarlara problemleri çözdürmektir. Yapay zeka bu görevi üstlenmiş bir bilim dalıdır. Bunu başarabilmek için problem ile ilgili her türlü bilgi ile bilgisayarın donatılmış olması gerekmektedir. Bilginin toplanması, derlenmesi ve bilgisayara verilmesi en önemli sorunlardan birisidir. Çünkü bilgisayarın sahip olduğu bilgi ne kadar doğru ve geçerli ise sonuçlarda o kadar doğru ve geçerli olacaktır. Bilginin elde edilmesinde değişik yöntemler kullanılmakla birlikte üç ana grupta toplamak mümkündür. Bunlar;

- Anketler, uzmanlar ile görüşmeler, mülakatlar, literatür taramaları vb. yollar ile ilgili olayı değişik açılardan inceleyerek bilgileri toplamak
- İşi uzmanları (çalışanları) ile birlikte yapmak
- İlgili olay için gerçekleşmiş örnekleri inceleyerek bilgileri elde etme (örneklerden öğrenme)

Bunlardan özellikle ikincisi bilinen örneklere bakarak bilinmeyen ve sonradan oluşacak benzeri örnekler hakkında karar verebilecek bilgilerin bulunmasıdır. Değişik örneklere tekrar tekrar bakarak önemli bilgiler ortaya çıkartılması gerekmektedir. Her örnekten bir şeyler öğrenilerek zaman içerisinde ilgili olay hakkında karar verebilecek bilgi düzeyine ulaşılmaktadır. Bu olaya *"makine öğrenmesi"* denmektedir. Bu kitabın konusu olan yapay sinir ağları makine öğrenmesi kapsamında geliştirilmiş yapay zekanın bir alt bilimidir. Kitabın konusu olmadığından bu kitapta yukarıdaki ilk iki yönteme dayalı bilgi toplama konusuna değinilmeyecektir. Örneklerden öğrenme ise ilerideki bölümlerde ayrıntılı olarak tanıtılacaktır. Bütünlük sağlanması amacı ile aşağıda yapay zeka teknolojilerine kısa bir bakış yapılmaktadır.

1.2. Yapay Zeka Teknolojileri

Yapay zeka çalışmaları değişik teknolojilerin doğmasına neden olmuştur. Çünkü günlük olaylar ve problemler sürekli değişmektedir. Değişik yerlerde olayların farklı yönleri insanları ilgilendirebilmektedir. Bir olay, değişik insanlar tarafından değişik şekillerde yorumlanmaktadır. Karşılaştıkları sorunlar farklı bölge ve kişilerce farklı şekillerde çözülebilmektedir. Bilgisayarların insanların karar verme ve problem çözme mekanizmalarını taklit etmesinin sağlanması da dolayısıyla farklı teknolojilerin doğmasına neden olmaktadır. Günümüzde 60'dan fazla yapay zeka teknolojisinden bahsedilmektedir. Kitabın konusu yapay zeka bilimini ayrıntılı olarak incelemek olmadığında burada hepsi listelenmeyecektir. Bu teknolojilerin çoğu henüz laboratuar çalışmaları düzeyindedir. Okuyucuya bilgi vermek amacı ile sadece bu teknolojilerin günümüzde günlük hayatta en yaygın olarak kullanılanları burada özetlenecektir. Bunlar arasında şunları saymak mümkündür.

- **Uzman Sistemler**

Bir problemi o problemin uzmanlarının çözdüğü gibi çözebilen bilgisayar programları geliştiren teknolojidir. Uzmanlar problemleri çözerken bilgilerini ve deneyimlerini kullanırlar. Bu bilgi ve deneyimlerin bilgisayar tarafından anlaşılabilir olması ve bilgisayarda saklanması gerekmektedir. Bilgi tabanında saklanan bu bilgileri kullanarak insan karar verme sürecine benzer bir süreç ile problemlere çözümler üretirler. Bir uzman sistemin 4 temel elemanı vardır:

a) ***bilginin temin edilmesi:*** Uzman sistemin uzmanlık alanı ile ilgili bilgilerin toplanması, derlenmesi ve bilgisayarın anlayacağı şekle dönüştürülmesi çalışmalarını kapsar.

b) ***Bilgi tabanı:*** Uzman sistemin uzmanlık alanı ile ilgili toplanan bilgilerin saklandığı yerdir. Bilgiler genellikle kurallar (EĞER... ise O ZAMAN... şeklinde), bilgi çatıları (çerçeveleri), bilgi sınıfları ve prosedürlerden oluşur. Bu bilgiler ilgili uzmanlık alanı hakkında uzmanların bildiği ve belirlediği gerçeklere dayanmaktadır.

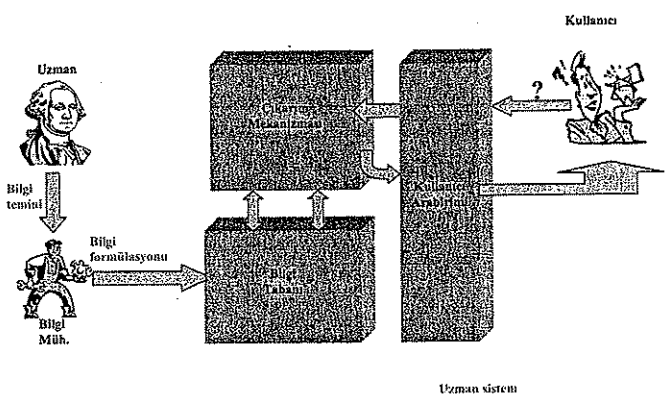
c) ***Çıkarım mekanizması:*** Bilgi tabanında bulunan bilgileri arayan, filtreleyen, yorumlayan ve sonuçlar çıkaran yani; çözüm üreten bir mekanizmadır. Genel olarak iki türlü çıkarım vardır.

a. ***İleri doğru zincirleme***: Bu durumda, ilgili problem hakkındaki gerçeklerden hareket edilerek sonuca gidilir.

b. ***Geri doğru zincirleme***: Bu durumda ise bir sonuç ele alınarak geriye doğru, o sonucu destekleyen gerçekler var mıdır? Sorusunun cevabı aranır.

*d) **Kullanıcı ara birimi:*** uzman sistemi kullanan kişiler ile uzman sistemin iletişimini sağlar. Problemlere üretilen sonuçların nasıl üretildiği ve niçin o sonuçlara varıldığını açıklar. Uzman sistemin bir uzman gibi görülmesi bu ara birimi ve açıklama yeteneğinin güçlü olmasına bağlıdır.

Uzman sistemler ve bu sistemlerin uygulama alanları ile ilgili ayrıntılı bilgiler Martin ve Oxman tarafından verilmiştir [1]. Bir uzman sistemin çalışma ilkesi Şekil-1.1'de gösterildiği gibidir:



Şekil-1.1. Uzman sistemin elemanları ve bilgi akışı

Görüldüğü gibi uzman sistem elemanlarından birisi de uzmanın kendisidir. Bir uzman sistemi geliştirmek için en az bir uzmana ihtiyaç vardır. Bilgi mühendisi bilgi temini yöntemlerini kullanarak uzmandan bilgileri elde eder. Topladığı bilgileri derleyerek gereksiz bilgileri uzmanın yardımıyla ayıklar ve bilgileri bilgisayarın anlayacağı biçime getirerek bilgi tabanına koyar. Kullanıcı uzman sisteme bir soru sorduğunda, çıkarım mekanizması bilgi tabanını arayarak sorulan sorunun cevabını arar. İlgili bilgileri belirleyip probleme çözüm ürettikten sonra kullanıcı arabirimi aracılığı ile kullanıcıya sorusunun cevabı verilir. Bazı durumlarda problemleri çözecek bilgiler bilgi tabanında bulunmayabilir. Bu durumda bilgi tabanına yeni bilgi eklenmesi veya güncellenmesi olasıdır. Zaman içinde bilgi tabanında yeterli bilgileri toplamak söz konusu olabilir. Bilginin çoğalması ise çıkarım mekanizmasının problemlere çözümler üretme gücünü artırmaktadır.

• Makine Öğrenmesi ve Yapay Sinir Ağları

Bilgisayarların olayları öğrenmesini sağlayan teknolojidir. Genellikle örnekler kullanılarak olayların girdi ve çıktıları arasındaki ilişkiler öğrenilir. Öğrenilen bilgiler ile benzer olaylar yorumlanarak kararlar verilir veya problemler çözülür. Bu kitabın konusu olduğundan bu teknoloji daha sonra ayrıntılı olarak açıklanacaktır.

• Genetik Algoritmalar

Karmaşık optimizasyon problemlerinin çözülmesinde kullanılan bir teknolojidir. Bir problemi çözebilmek için öncelikle rasgele başlangıç çözümleri belirlenmektedir. Daha sonra bu çözümler birbirleri ile eşleştirilerek performansı yüksek (daha iyi) çözümler üretilmektedir. Bu şekilde sürekli çözümler birleştirilerek yeni çözümler aranmaktadır. Bu arama iyi sonuç üretilmeyinceye kadar devam etmektedir. Genetik algoritmalar ile problemlerin çözülmesinde arzu edilen sonucu üretecek özelliklerin kalıtım yolu ile başlangıç çözümlerinden elde edilen yeni çözümlere onlardan da daha sonraki çözümlere geçtiği kabul edilmektedir. Bir genetik algoritmanın temel elemanları Şekil-1.2'de gösterilmiştir. Bu elemanlar kısaca şöyle tanımlanmaktadır.

*a) **Kromozom ve gen:*** Genetik algoritmanın çözmesi istenen problemin her bir çözümünü göstermektedir. Bir problem için *N* adet çözüm olabilir. Genetik algoritmanın bunların arasından en iyisini arayıp bulması istenmektedir. Kromozomlar ise bu çözümleri gösterirler. Başlangıçta rasgele atanan çözümler daha sonra genetik algoritmanın çalışma ilkesine göre iyileştirilmektedir. Bir kromozomun elemanlarından her birisi çözümün bir özelliğini göstermektedir. Bunlara da ***gen*** denilmektedir.

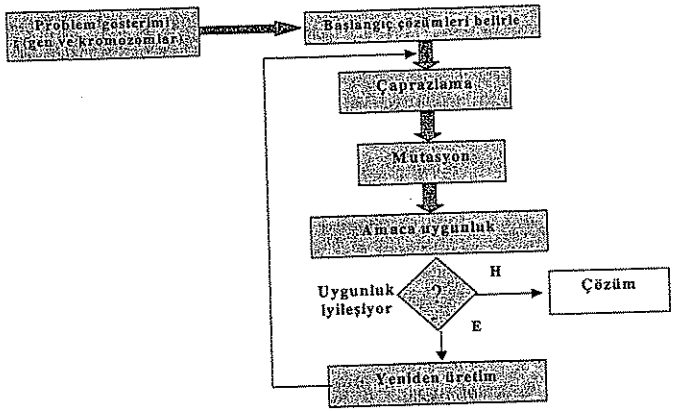
*b) **Çözüm havuzu:*** problemin en iyi çözümünü aramak için kullanılan ve rasgele belirlenmiş başlangıç çözüm setidir. Probleme göre değişen sayıda kromozom (başlangıç çözümü) belirlenebilir. Buna havuzun büyüklüğü denmektedir.

*c) **çaprazlama:*** Problem çözüm havuzunda bulunan çözümleri (kromozomları) ikişer ikişer birleştirerek yeni çözümler üretmektir. İki kromozomdan iki adet yeni kromozom üretilmektedir. Bir problem çözüm uzayından kaç adet kromozomun çaprazlanacağı ***çaprazlama oranına*** göre belirlenmektedir.

*d) **Mutasyon:*** Çaprazlama neticesinde farklı çözümlere ulaşmak bazen zor olmaktadır. Yeni çözüm aramanın kolaylaştırılması ve aramanın yönünü değiştirmek amacı ile bir kromozomun bir elemanının (bir genin) değiştirilmesi işlemidir. Bir problem havuzu içinden kaç kromozomun mutasyona uğratılacağına ***mutasyon oranına*** göre karar verilmektedir.

*e) **Uygunluk fonksiyonu:*** Belirlenen çözümlerin uygunluk derecelerinin ölçülmesini sağlayan bir fonksiyondur. Her problem için bir uygunluk fonksiyonunun belirlenmesi gerekmektedir. Bu fonksiyon probleme göre değişmektedir. Mesela, bir iş çizelgelemesi (sıralanması) probleminde belirlenen her sıraya göre, işlerin tamamının bitirilmesi zamanı uygunluk ölçütü olarak belirlenebilir. En kısa zamanda işleri bitiren iş sırasının belirlenmesi

istenmektedir. İşlerin toplam bitirilme zamanı azaltıldığı sürece daha farklı iş sıralarının aranmasına devam eder. Bu süre daha fazla azaltılamayacağı bir noktada en iyi çözüm bulunmuş olacaktır.



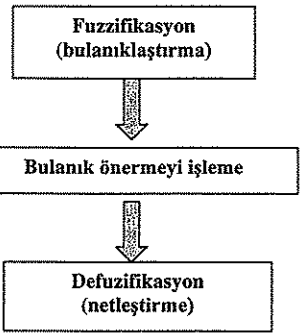
Şekil-1.2. Genetik algoritmanın elemanları ve çalışması

*f) **Yeniden üretim:*** Çözüm havuzundaki kromozomlar çaprazlama ve mutasyon neticesinde üretilen yeni kromozomlar nedeni ile çoğalacaktır. Bunların arasından problem havuz büyüklüğüne göre kromozomlar seçilerek diğerleri atılır. Seçilenler ise bir sonraki nesil çözümü olarak yeniden çaprazlanıp gelecek çözümleri üretirler. Yeniden üretim için değişik yöntemler geliştirilmiştir. ***Rus Ruleti*** en yaygın olarak kullanılan yeniden üretme yöntemidir. Bu yöntemde yeni havuz üyeleri rulet mantığı ile seçilmektedir. En iyi çözümlerin bir sonraki nesil (havuz) içine seçilme olasılıkları daha fazladır. Kötü çözümlerde seçilebilirler. Bunun nedeni belki bu kötü çözümler ileride daha iyi çözümlere yol açabileceklerindendir.

Genetik algoritmalar ile ilgili ayrıntılı bilgiler [2] nolu referansta bulunabilir.

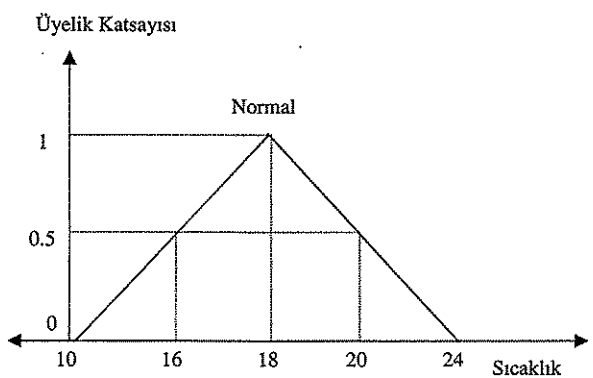
• Bulanık Önermeler Mantığı

Günümüzdeki bir çok olay belirsiz koşullarda gerçekleşmektedir. Beklenmedik olaylar ortaya çıkmakta karar vermeyi etkilemektedir. Seyahate gitmek isteyen bir kişinin yağmur yağması ile planlarını değiştirmesi gibi belirsiz olaylar insanların kararlarını etkilemektedir. Kesin olarak bilinemeyen olaylar hakkında karar vermek için uzmanlar *normal, yüksek, düşük, yaklaşık* gibi kavramları kullanmaktadırlar. Hava sıcaklığının belirsiz olması durumunda sıcaklık 20 derece olunca şu işi yap demek yerine sıcaklık normal olunca şu işi yap denilmektedir. Bulanık mantık teknolojisi bilgisayarında bu gibi durumlarda karar verebilmesi için geliştirilmiş teknolojidir. *"Normal"* gibi kavramların bilgisayar tarafından anlaşılmasını sağlar. Bulanık mantık Şekil-1.3'te gösterilen sürece göre çalışmaktadır. Bu süreçte,



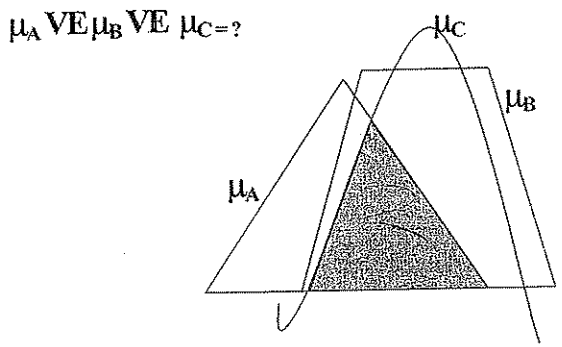
Şekil-1.3. Bulanık önermeler mantığının elemanları ve çalışması

*a) **Bulanıklaştırma:*** Çözülecek problem ile ilgili bulanık önerme değişkenlerinin ve karar verme kurallarının belirlenmesi ve üyelik fonksiyonunun oluşturulması işlemidir. Mesela, hava sıcaklığının *"normal"* olması durumunda değişkenin adı *"normal"* olabilir. Üyelik fonksiyonu ise herhangi bir sıcaklık değerinin "normal" olma üyeliğini gösterir. Üyelik değeri 0-1 arasında bir değerdir. 1 tam üyelik durumunu 0 ise üye olmama durumunu gösterir. Üyelik değerinin bir olasılık değeri olmadığını belirtmek gerekmektedir. 18 derece sıcaklığın normal olma üyelik değeri 1 ise bu normal olma olasılığı %100'dür demek değildir. Üyelik değerinin olasılık değeri yerine *"mümkünüyet değeri"* olarak görmek gerekir. Normal değişkeninin üyelik fonksiyonu Şekil-1.4'te gösterilmiştir. Görüldüğü gibi bu üyelik fonksiyonu $-\infty$ ile $+\infty$ arasındaki bütün sıcaklık değerlerinin üyelik katsayılarını bulmak mümkündür.



Şekil-1.4. Üyelik fonksiyonu örneği

*b) **Bulanık önerme işleme:*** Belirlenen bulanık önerme değişkenlerinin kurallarını kullanarak problemin çözüm alanını belirleme işlemidir. Genellikle üyelik fonksiyonlarının üst üste konulması sonucu kurallara göre ortak alanın bulunmasıdır. Eğer kurallar AND (VE) bağlacı ile bağlanmış ise üyelik fonksiyonlarının küçük değeri, OR (VEYA) bağlacı ile bağlanmış ise o zaman üyelik fonksiyonlarının büyük değerleri alınarak alan oluşturulur. Şekil-1.5, oluşturulmuş bir çözüm alanı örneğini göstermektedir. Şekildeki taralı alan çözüm uzayını göstermektedir.



Şekil-1.5. Bulanık önermeler çözüm uzayı örneği

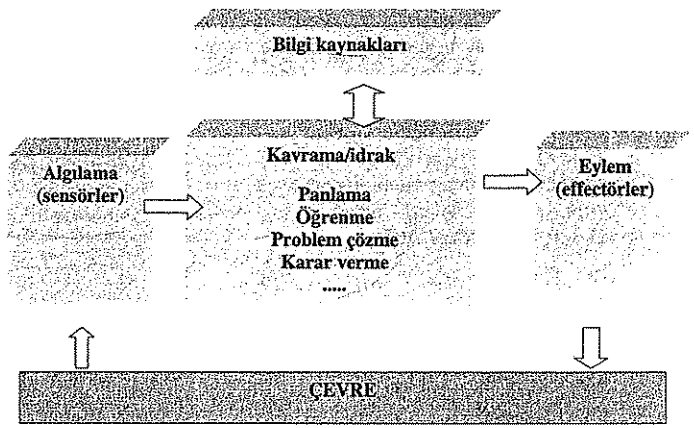
*c) **Netleştirme:*** Bulunan çözüm alanından tek bir değer elde edilmesi işlemidir. Genellikle üyelik değerinin en yüksek olduğu noktaya karşılık gelen değer problemin çözümü olan tek değerdir. Bu alandan böyle tek bir değer belirlenememesi durumunda en yüksek değerlerin ortalaması veya oluşan çözüm alanının ağırlık merkezine karşı gelen nokta çözüm değeri olarak alınır.

Bulanık mantık konusunda ayrıntılı bilgiler [3] nolu referansta bulunmaktadır.

• Zeki Etmenler

Bunlar bağımsız kararlar verebilen bilgisayar sistemleridir. Hem donanım hem de yazılım olarak geliştirilmektedirler. Birden fazla yapay zeka tekniğini kullanabilirler. Öğrenme ve gerçek zamanlı çalışabilme özellikleri vardır. Zeki etmenler birden fazla yapay zeka teknolojisini kullanabilirler. Genel olarak bir zeki etmenin yapısı Şekil-1.6'da verilmiştir. Şekilden görüldüğü gibi zeki etmenlerin 3 ana elemanı vardır. Bunlar:

*a) **Algılama**:* Dış dünyadan (çevreden) gelen bilgilerin algılanması, yorumlanması, gereksiz bilgilerin ayıklanması, bilgilerin önem derecelerine göre sıralanması ve önceliklendirilmesi gibi işlemleri gerçekleştirir.



Şekil-1.6. Bir zeki etmenin elemanları

*b) **Kavrama/idrak**:* Algılamadan gelen bilgilerin işlenmesi, karar verme, muhakeme etme, problem çözme, planlama, öğrenme vb. gibi işlemlerin gerçekleştirildiği birimdir. Problemler çözülürken öncelik ilişkileri göz önünde bulundurulur. Ani olarak çevreden gelen yeni bilgilerin kararlar üzerine etkisi göz önünde bulundurulur.

*c) **Eylem**:* Bilgi işleme neticesinde oluşturulan problem çözümler gereği etmenin davranışlarını belirler. Bu bir robotun yürümesi olabileceği gibi, bir reaktöre çalış komutunun gönderilmesi gibi bir eylemde olabilir. Eylemler neticesinde dış dünyadan (çevrede) değişiklikler oluşur. Bu değişiklikler algılama yolu ile yeniden algılanıp işlenerek yeni eylemlerin oluşturulmasına neden olur.

Zeki etmenler hakkında ayrıntılı bilgiler `Russel ve Norvig` (1995) tarafından verilmiştir [4].

1.3. Makine Öğrenmesi ve Öğrenme Türleri

Makine öğrenmesini anlayabilmek için öncelikle öğrenme kavramının tanımlanması gerekmektedir. Öğrenme kavramı değişik şekillerde tanımlanmakla birlikte genellikle `Simon` [5] tarafından önerilen tanım etrafında değişiklikler yapılmaktadır. `Simon` öğrenmeyi, *"zaman içinde yeni bilgilerin keşfedilmesi yoluyla davranışların iyileştirilmesi süreci"* olarak tanımlamaktadır. Makine öğrenmesi ise bu öğrenme işinin bilgisayarlar tarafından gerçekleştirilmesinin sağlanmasıdır. Burada zaman içerisinde iyileşme kavramına dikkat çekmek gerekmektedir. Bilgisayarın da insan gibi zaman içerisinde tecrübe kazanması istenmektedir. Diğer bir deyişle makine öğrenmesi *" bilgisayarın bir olay ile ilgili bilgileri ve tecrübeleri öğrenerek gelecekte oluşacak benzeri olaylar hakkında kararlar verebilmesi ve problemlere çözümler üretebilmesidir"* denilebilir. Bilgisayarın öğrenebilmesi ve tecrübe sahibi olabilmesi bilgisayarın

ilgili olay hakkında bilgiler ile donatılmasına bağlıdır. Yapay sinir ağları yolu ile öğrenen bilgisayarların bilgiler ile donatılması, örnekler yolu ile sağlanmaktadır. Öğrenecek olan bilgisayar sistemleri önce bir örnek almakta ve bu örnekten bazı bilgileri öğrenmektedir. Daha sonra ikinci örneğe bakarak biraz daha bilgi edinmektedir. Bu işlemi öğrenilecek olay ile ilgili bütün örnekleri defalarca gözden geçirerek tekrarlamak sonucunda olay ile ilgili genellemeler yapılmaktadır. Bu olaya tecrübelerden öğrenmenin bir yolu olarak bakmak mümkündür. Yapay sinir ağlarının dışında da değişik öğrenme şekilleri vardır. Bu konuda değişik sınıflandırmalar yapılmıştır. Bunların bazılarını şöyle sıralamak mümkündür:

- Alışkanlık yolu ile öğrenme
- Görerek öğrenme
- Talimatlardan öğrenme
- Örneklerden öğrenme
- Analoji yolu ile öğrenme
- Açıklamalardan öğrenme
- Deney yolu ile öğrenme
- Keşfetmek yolu ile öğrenme

Geliştirilen herhangi bir makine öğrenme sistemi yukarıdaki öğrenme türlerinden birini veya birkaçını birlikte kullanabilir. Genel olarak bakıldığında makine öğrenmesi çalışmalarının iki amaçla gerçekleştirildiği görülmektedir. Bunlar;

- ***Farklı çağrışım*** *(hetero-association):* Burada bir olay ister gözlemleyerek olsun, ister talimatlar yolu ile olsun, ister örnekler yolu ile olsun (yukarıdaki öğrenme türlerinden hangisi yoluyla olursa olsun) değişik açılardan incelenerek olayın genel yönleri ortaya çıkartılmakta ve daha sonra oluşan benzeri bir durumda bu genel yönler kullanılarak problemler çözülmektedir. Örnekleri kullanmak durumunda, öğrenilecek olay hakkında elde bulunan örnekler kullanılarak genellemeler yapılmakta ve sınıflandırma problemlerinde olduğu gibi örneklerin ortak noktası öğrenilmektedir. Bu tür öğrenmede, öğrenme yapacak sistemin girdisi ve çıktısı birbirinden farklıdır. Makine öğrenmesi sisteminin görevi, örneklerin çağrıştırdığı çıktıları belirlemektir. Sınıflandırma ve tahmin etme problemlerinde bu tür öğrenme yaygın olarak kullanılmaktadır. Günlük döviz kurlarındaki değişime bakarak bir sonraki günün döviz kurunu tahmin etmek güzel bir örnek olabilir. Bir prosesten gözlem yolu ile elde edilen 40-50 adet ölçüm ile prosesin normal davranış sergileyip sergilemediğini belirlemek gibi bir olayda bu tür öğrenmeye örnek olarak verilebilir.
- ***Oto çağrışım*** *(auto-association):* Bu durumda ise bir olay öğrenilerek daha sonra eldeki var bilgilerle o olayın kendisi karakterize edilmektedir. Eldeki bilgilerin eksik olması durumunda makine öğrenmesi sayesinde sistem eksik bilgiler tamamlayabilmektedir. Bu durumda, öğrenme yapacak sisteme (mesela yapay sinir ağlarına) bir örnek girdi olarak verilir ve aynı örnek çıktı olarak istenir. Bu, bir

insanın resmini öğretmek gibi bir olaydır. Burada amaç, ağa örnekleri göstererek ilgili olay hakkında eksik bilgilerin olması durumunda ilgili olayın bilgilerinin tamamını oluşturmaktır. Bir insanın resmini öğrendikten sonra yırtık bir resmin sahibinin belirlenmesi bu tür öğrenmeye güzel bir örnektir.

1.4. Öğrenme Paradigmaları

Öğrenme üzerine yapılan çalışmaları, makine öğrenmesi haline dönüştürebilmek (bilgisayarların öğrenmesini sağlamak) için çeşitli paradigmalar ve yaklaşımlar geliştirilmiştir. Bunlar arasında şunları saymak mümkündür.

- ***Sembol işleme yöntemi*** *(symbolic processing):* Bu sistemler geleneksel yapay zeka teknolojisine dayanan öğrenme sistemleridir. Belirli formatlarda bilgisayara sunulan (mesela kurallar halinde sunulan) bilgilere dayanarak muhakeme yolu ile öğrenme gerçekleştirilir. Bu konuda ayrıntılı bilgiler [6-8] nolu referanslarda bulunabilir.
- ***Yapay sinir ağları*** *(connectionist systems):* Bu sistemler örneklerden genellemeler yaparak öğrenirler. Bu kitabın konusu bu olduğundan bu sistemler ilerideki bölümlerde örnekleri ile ayrıntılı olarak açıklanmıştır.
- ***İstatistiksel örüntü tanıma*** *(Statistical pattern recognition):* Bu tür öğrenmede bir veri setinin istatistiksel özellikleri ve dağılımı incelenerek veri hakkında genellemeler yapmak söz konusudur. Bu konuda gerekli açıklamalar [9] nolu referansta bulunabilir.
- ***Genetik algoritmalar ve evrimsel programlama*** *(Genetic algorithms and evolutionary programming):* Bu paradigma bir problemin çözümü için başlangıç çözümleri rasgele atamak ve bu çözümlerden yeni çözümler üretmek ve daha sonra yeni çözümlerden daha iyi ve yeni çözümler üretmek amaçlanmaktadır. Bu işlemi tekrar ederek sürekli daha iyi çözümler üretilmektedir. Bu iyileştirme en son halini alıncaya (daha fazla iyileştirme sağlanamayınca) kadar devam etmektedir. Ayrıntılı bilgi [2,10] nolu referanslarda bulunabilir.
- ***Vaka tabanlı öğrenme*** *(Case based learning):* Bu teknoloji örneklerden de öte vakalara bakarak öğrenme esasına dayanmaktadır. Her vaka başlı başına bir olayı göstermekte ve birden fazla örnekten oluşmaktadır. Bir olay söz konusu olunca benzeri vakalara bakarak o vakalar için verilen kararların benzeri kararlar verilmektedir. Ayrıntılı bilgi [11] nolu referansta bulunabilir.

1.5. Örneklerden Öğrenme

Yapay sinir ağları gibi öğrenme yöntemleri örneklerden öğrenmeye dayanmaktadır. Örneklerden öğrenmenin temel felsefesi bir olay hakkındaki gerçekleşmiş örnekleri kullanarak olayın girdi ve çıktıları arasındaki ilişkileri öğrenmek ve bu ilişkilere göre daha sonra oluşacak olan yeni örneklerin çıktılarını belirlemektir. Burada bir olay ile ilgili örneklerin girdi ve çıktıları arasındaki ilişkinin olayın genelini temsil edecek

bilgiler içerdiği kabul edilmektedir. Değişik örneklerin olayı değişik açılardan temsil ettiği varsayılmaktadır. Farklı örnekler kullanılarak böylece olay değişik açılardan öğrenilmektedir. Burada bilgisayara sadece örnekler gösterilmektedir. Bunlardan başka herhangi bir ön bilgi verilmemektedir. Öğrenmeyi gerçekleştirecek sistem (mesela yapay sinir ağı) aradaki ilişkiyi kendi paradigmasını (algoritmasını) kullanarak keşfetmektedir. Sistem örneklerden öğrendiğine göre, örnek kavramının irdelenmesi gerekir. Örnek nasıl oluşturulmaktadır? Örnek nasıl formülüze edilmektedir? Bilgisayar örneği nasıl anlayacaktır? Başarılı bir öğrenme sistemi geliştirmek için bu ve benzeri soruların cevaplarının bilinmesi gerekmektedir.

Örnekler, gerçek hayatta ilgili olayın değişik parametrelerini göstermektedir. Bu örneklerin oluşturulması ve bilgisayara gösterimi konusu da önemlidir. Kitap içinde bu konuda ilgili yapay sinir ağı modelleri anlatılırken değişik örnekler verilecektir. Genel olarak bir örnek, girdi ve çıktı kombinasyonundan oluşur. Girdi vektörü *X*, ve çıktı vektörü *Y* ile gösterilirse, *X* vektörünün *n* adet elemanı olacaktır. Bunlar *x*1, *x*2, *x*3, ...*x*n şeklinde gösterilebilir. Benzer şekilde çıktı vektörünün de *m* adet boyutu olabilir ve *y*1, *y*2, *y*3, ...*y*m şeklinde gösterilebilir. Girdi ve çıktı elemanlarının her biri öğrenilecek olan olayın bir boyutunu gösterir. Örnek olarak, bir yapay sinir ağına döviz kurunu tahmin etmesi öğretilmek istenirse oluşturulacak olan örnek şu şekilde gösterilebilir.

Örneğin çıktısı:

*Y*1: *Bir sonraki dönem döviz kuru değerini*

Girdileri:

*X*1- *Bu dönem döviz kuru değeri*

*X*2- *Enflasyon oranı*

*X*3- *Tüketici indexi*

*X*4- *Borsa indexi*

*X*5- *İhracat periyodik artış/azalış oranı*

Bu örnek yapay sinir ağına (*X*1, *X*2, *X*3,*X*4, *X*5, *Y*1) şeklinde gösterilebilir. Her bir parametrenin farklı değerler ile farklı örnekler oluşturulabilir. Burada kullanılan bazı yapay sinir ağı modellerinin sadece girdileri istediğini, çıktıları ise ağın kendisinin oluşturduğunu belirtmekte fayda vardır.

Yapay sinir ağlarında öğrenme ileride tekrar ele alınacaktır. Burada, yapay sinir ağlarının birbirine bağlanan proses elemanlarından oluştuğu ve her bağlantının bir değerinin bulunduğu, öğrenme sırasında bu değerlerin sürekli değiştirilerek olması gereken bağlantı değerlerinin belirlendiğini belirtmekte fayda vardır. Öğrenme aslında bu değerlerin belirlenmesi sürecidir. Ağırlık değerlerinin değiştirilmesi ***öğrenme katsayısı*** denilen bir katsayının yardımı ile gerçekleştirilir.

1.6. Öğrenme Stratejileri

Yapay sinir ağları gibi örneklerden öğrenen sistemlerde değişik öğrenme stratejileri kullanılmaktadır. Öğrenmeyi gerçekleştirecek olan sistem ve kullanılan öğrenme algoritması bu stratejilere bağlı olarak değişmektedir. Genel olarak 3 öğrenme stratejisinin uygulandığı görülmektedir. Bunlar:

1.6.1. Öğretmenli (*Supervised*) Öğrenme

Bu tür stratejide öğrenen sistemin olayı öğrenebilmesine bir öğretmen yardımcı olmaktadır. Öğretmen sisteme öğrenilmesi istenen olay ile ilgili örnekleri Girdi/Çıktı seti olarak verir. Yani, her örnek için hem girdiler hem de o girdiler karşılığında oluşturulması gereken çıktılar sisteme gösterilirler. Sistemin görevi girdileri öğretmenin belirlediği çıktılara haritalamaktır. Bu sayede olayın girdileri ile çıktıları arasındaki ilişkiler öğrenilmektedir. Bu kitapta anlatılacak olan ***"çok katmanlı algılayıcı"*** ağı bu stratejiyi kullanan ağlara örnek olarak verilebilir.

1.6.2. Destekleyeci (*Reinforcement*) Öğrenme

Bu tür stratejide de öğrenen sisteme bir öğretmen yardımcı olur. Fakat öğretmen her girdi seti için olması gereken (üretilmesi gereken) çıktı setini sisteme göstermek yerine sistemin kedisine gösterilen girdilere karşılık çıktısını üretmesini bekler ve üretilen çıktının doğru veya yanlış olduğunu gösteren bir sinyal üretir. Sistem, öğretmenden gelen bu sinyali dikkate alarak öğrenme sürecini devam ettirir. Bu kitapta anlatılacak olan ***"LVQ"*** ağı bu stratejiyi kullanan sistemlere örnek olarak verilebilir.

1.6.3. Öğretmensiz (*Unsupervised*) Öğrenme

Bu tür stratejide sistemin öğrenmesine yardımcı olan herhangi bir öğretmen yoktur. Sisteme sadece girdi değerleri gösterilir. Örneklerdeki parametreler arasındaki ilişkileri sistemin kendi kendisine öğrenmesi beklenir. Bu, daha çok sınıflandırma problemleri için kullanılan bir stratejidir. Yalnız sistemin öğrenmesi bittikten sonra çıktıların ne anlama geldiğini gösteren etiketlendirmenin kullanıcı tarafından yapılması gerekmektedir. Bu kitapta anlatılacak olan ***"ART"*** ağları bu stratejiyi kullanan sistemlere örnek olarak verilebilir.

1.6.4. Karma Stratejiler

Yukarıdaki 3 stratejiden birkaçını birlikte kullanarak öğrenme gerçekleştiren ağlarda vardır. Burada kısmen öğretmenli, kısmen ise öğretmensiz olarak öğrenme yapan ağlar kastedilmektedir. Radial tabanlı yapay sinir ağları (RBN) ve olasılık tabanlı ağlar (PBNN) bunlara örnek olarak verilebilir.

1.7. Öğrenme Kuralları

Yapay sinir ağları gibi öğrenen sistemlerde öğrenme, yukarıda anlatılan stratejilerden hangisi uygulanırsa uygulansın bazı kurallara göre gerçekleştirilmektedir. Bu kuralların bazıları çevrimiçi (*on-line*) bazıları ise çevrimdışı (*off-line*) çalışmaktadır.

1.7.1. Çevrimiçi (*On-line*) Öğrenme Kuralları

Bu kurallar gerçek zamanlı çalışabilmektedir. Bu kurallara göre öğrenen sistemler gerçek zamanda çalışırken bir taraftan fonksiyonlarını yerine getirmekte diğer taraftan ise öğrenmeye devam etmektedir. ART ağının öğrenme kuralı ve *Kohonen* öğrenme kuralı bu sınıfta bulunan öğrenme kurallarına örnek olarak verilebilir.

1.7.2. Çevrimdışı (*Off-line*) Öğrenme Kuralları

Çevrimdışı öğrenme kurallarına dayalı öğrenen sistemler kullanıma alınmadan önce örnekler üzerinde eğitilirler. Bu kuralları kullanan sistemler eğitildikten sonra gerçek hayatta kullanıma alındığında artık öğrenme olmamaktadır. Sistemin öğrenmesi gereken yeni bilgiler söz konusu olduğunda sistem kullanımdan çıkarılmakta ve çevrimdışı olarak yeniden eğitilmektedir. Eğitim tamamlanınca sistem tekrar kullanıma alınmaktadır. Yapay sinir ağlarında yaygın olarak kullanılan ***"Delta Öğrenme Kuralı"*** bu tür öğrenmeye örnek olarak verilebilir.

1.7.3. Öğrenme Kurallarından Bazıları

Öğrenme sistemlerinde kullanılan değişik öğrenme kuralları vardır. Yapay sinir ağlarında bu öğrenme kurallarının çoğu *Hebb* kuralına dayanmaktadır. Bu kitapta her öğrenme stratejisini kullanan bir öğrenme kuralı ayrıntılı olarak anlatılacaktır. Bu bölümde ise kurallara genel bir bakış yapılacaktır.

- ***Hebb kuralı:*** Bilinen en eski öğrenme kuralıdır. Diğer öğrenme kurallarının temelini oluşturmaktadır. 1949 yılında geliştirilen bu kurala göre, bir hücre (yapay sinir ağı elemanı) diğer bir hücreden bilgi alırsa ve her iki hücrede aktif ise (matematik olarak aynı işareti taşıyorsa) her iki hücrenin arasındaki bağlantı kuvvetlendirilmelidir. Diğer bir deyişle bu kural şu şekilde özetlenebilir. Bir hücre kendisi aktif ise bağlı olduğu hücreyi aktif yapmaya pasif ise pasif yapmaya çalışmaktadır. Diğer öğrenme kurallarının çoğu bu felsefeyi baz alarak geliştirilmiştir.
- ***Hopfield kuralı***: Bu kural *Hebb* kuralına benzemektedir. Yapay sinir ağı elemanlarının bağlantılarının ne kadar kuvvetlendirilmesi veya zayıflatılması gerektiği belirlenir. Eğer beklenen çıktı ve girdiler ikisi de aktif /pasif ise öğrenme katsayısı kadar ağırlık değerleri kuvvetlendir/zayıflat denmektedir. Yani, ağırlıkların kuvvetlendirilmesi veya zayıflatılması öğrenme katsayısı yardımı ile gerçekleştirilmektedir. Öğrenme katsayısı genel olarak 0-1 arasında kullanıcı tarafından atanan sabit ve pozitif bir değerdir. *Hopfield* öğrenme kuralı 8. bölümde ayrıntılı olarak açıklanmaktadır.

- ***Delta kuralı***: Bu kural *Hebb* kuralının biraz daha geliştirilmiş şeklidir. Bu kurala göre beklenen çıktı ile gerçekleşen çıktı arasındaki farklılığı azaltmak için yapay sinir ağının elemanlarının bağlantılarının ağırlık değerlerinin sürekli değiştirilmesi ilkesine dayanarak geliştirilmiştir. Ağın ürettiği çıktı ile üretilmesi gereken (beklenen) çıktı arasındaki hatanın karelerinin ortalamasını enazlamak hedeflenmektedir. 5. Bölümde bu kural ayrıntılı olarak anlatılacaktır.
- ***Kohonen kuralı***: Bu kurala göre ağın elemanları (hücreleri) ağırlıklarını değiştirmek için birbirleri ile yarışırlar. En büyük çıktıyı üreten hücre kazanan hücre olmakta ve bağlantı ağırlıkları değiştirilmektedir. Bu, o hücrenin yakınındaki hücrelere karşı daha kuvvetli hale gelmesi demektir. Hem kazanan elemanların hem de komşuları sayılan elemanların (hücrelerin) ağırlıklarını değiştirmesine izin verilmektedir.

1.8. Özet

Yapay zeka teknolojisi her geçen gün daha fazla gelişmektedir. Yeni ürünler ortaya çıkmakta ve daha çok günlük hayatta kendisini göstermektedir. Otomasyon sistemleri de yapay zeka teknolojisi ile donatılarak bilgisayarın karar verme gücünden faydalanılmaktadır. Her geçen gün daha yeni ticari sistemler ortaya çıkmakta ve sistemlerin fonksiyonel özellikleri artmaktadır. Yapay zeka teknolojilerinden özellikle;

- ***Uzman sistemler***: Bir uzmanın problemleri çözdüğü gibi problemlere çözümler üreten sistemlerdir. Uzmanlık bilgisi ile donatılırlar. Çıkarım mekanizmaları bilgiler arasındaki ilişkileri kurarak kararlar verirler.
- ***Yapay sinir ağları***: Örneklerden olaylar arasındaki ilişkileri öğrenerek daha sonra hiçi görmediği örnekler hakkında öğrendikleri bilgileri kullanarak karar veren sistemlerdir.
- ***Genetik algoritmalar***: Geleneksel optimizasyon teknolojisi ile çözülemeyen problemleri çözmek üzere geliştirilmişlerdir. Problemlerin çözümlerini birleştirerek daha iyi çözümler üretmek felsefesine dayanmaktadır.
- ***Bulanık önermeler mantığı***: Belirsiz bilgileri işleyebilme ve kesin rakamlar ile ifade edilemeyen durumlarda karar vermeyi kolaylaştıran bir teknolojidir.
- ***Zeki etmenler***: değişik yapay zeka tekniklerini kullanabilen ve bağımsız olarak çalışabilen sistemlerdir. Esnek bir şekilde programlanmaktadırlar.

Bu teknolojiler günlük hayatta insanlara faydalı ürünlerin oluşmasına katkıda bulunmaktadırlar. Bunlardan yapay sinir ağları bilgisayarın öğrenmesini sağlamaktadır. Makine öğrenmesi zaman içinde davranışların iyileştirilmesi olarak tanımlanmaktadır. Değişik öğrenme paradigmaları geliştirilmiştir. Bu kitapta özellikle örneklerden öğrenme konusu anlatılacaktır. Bu öğrenme paradigmaları 3 strateji üzerine kurulmuştur. Bunlar;

- öğretmenli öğrenme
- destekli öğrenme
- öğretmensiz öğrenme.

Bu stratejilere dayanarak geliştirilmiş öğrenme kuralları vardır. Bu kuralların bazılar çevrimiçi bazılar ise çevrimdışı öğrenme yaparlar. Kitap içinde bu tür öğrenme gerçekleştirebilen kuralların örnekleri ayrıntılı olarak açıklanacaktır. Özellikle *delta* kuralı, *Hopfield* kuralı, *Kohonen* kuralı ve *Grosberg* kuralı ileride açıklanacaktır.

1.9. Kaynakça

[1] Martin J., Oxman S. (1988),Building Expert Systems: A Tutorial, New Jersey, Prentice Hall.

[2] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. New York: Addison Wesley.

[3] Cox E. (1994), "The Fuzzy Systems Handbook: A Practitioner's Guide to Building Using and Maintaining Fuzzy Systems", Boston, Academic Press.

[4] Russell, S., Norvig, P. (1995) Artificial Intelligence A Modern Approach. Prentice-Hall

[5] Simon, H. A. (1983). Why Should Machines Learn? In: Machine Learning -An Artificial Intelligence Approach. Michalski, R. S., Carbonell, J. G., & Mitchell,T. M. (Ed). Palo Alto, California: Tioga.

[6] Shavlik, J. W., & Dietterich, T. G. (1990). (Ed). Readings in Machine Learning. San Mateo, California: Morgan Kaufmann.

[7] Buchanan, B. G., & Wilkins, D. C. (1993). Readings in Knowledge Acquisition and Learning. San Mateo, California: Morgan Kaufmann.

[8] Michalski, R. S. (1993). Toward a Uni_ed Theory of Learning: Multi-Strategy Task- Adaptive Learning. In: Readings in Knowledge Acquisition and Learning. Buchanan, B. G.,& Wilkins, D. C. San Mateo, California: Morgan Kaufmann.

[9] Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. New York: Academic Press.

[10] Koza, J. (1992). Genetic Programming. Boston, Massachusetts: MIT Press.

[11] Leake D.B., 1996, Case based reasoning: experience, lessons and Future directions, AAAI Press, Menlo Park, California.

BÖLÜM 2

YAPAY SİNİR AĞLARINA GİRİŞ

Bir önceki bölümde Yapay Zeka bilimi ve Makine Öğrenmesi çalışmaları çok genel bir şekilde anlatılmıştır. Öğrenmeye ve yapay sinir ağlarına dikkatler çekilmiştir. Bu bölümde ise yapay sinir ağları genel olarak tanıtılmakta temel elemanları açıklanmaktadır.

2.1. Yapay Sinir Ağlarının Genel Tanımı

Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilen bilgisayar sistemleridir. Bu yetenekleri geleneksel programlama yöntemleri ile gerçekleştirmek oldukça zor veya mümkün değildir. O nedenle, yapay sinir ağlarının, programlanması çok zor veya mümkün olmayan olaylar için geliştirilmiş adaptif bilgi işleme ile ilgilenen bir bilgisayar bilim dalı olduğu söylenebilir.

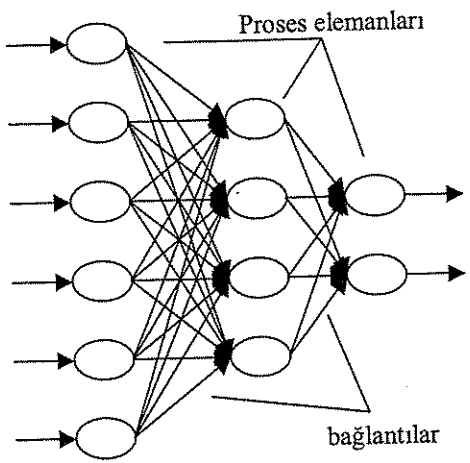
2.2. Yapay Sinir Ağı Tanımı ve En Temel Görevi

Yapay sinir ağları, insanlar tarafından gerçekleştirilmiş örnekleri (gerçek beyin fonksiyonlarının ürünü olan örnekleri) kullanarak olayları öğrenebilen, çevreden gelen olaylara karşı nasıl tepkiler üretileceğini belirleyebilen bilgisayar sistemleridir. İnsan beyninin fonksiyonel özelliklerine benzer şekilde,

- **öğrenme**
- **ilişkilendirme**
- **sınıflandırma**
- **genelleme**
- **özellik belirleme** ve
- **optimizasyon**

gibi konularda başarılı bir şekilde uygulanmaktadırlar. Örneklerden elde ettikleri bilgiler ile kendi deneyimlerini oluşturur; ve daha sonra, benzer konularda benzer kararları verirler.

Yapay sinir ağları günümüzde bir çok probleme çözüm üretebilecek yeteneğe sahiptir. Değişik şekillerde tanımlanmaktadır. Tanımların ortak bir kaç noktası vardır. Bunların en başında yapay sinir ağlarının birbirine hiyerarşik olarak bağlı ve paralel olarak çalışabilen yapay hücrelerden oluşmaları gelmektedir. Proses elemanları da denilen bu hücrelerin birbirlerine bağlandıkları ve her bağlantının bir değerinin olduğu kabul edilmektedir. Bilginin öğrenme yolu ile elde edildiği ve proses elamanlarının bağlantı değerlerinde saklandığı dolayısıyla dağıtık bir hafızanın söz konusu olduğu da ortak noktaları oluşturmaktadır (örnek tanım için bkz. [1], [33]). Proses elemanlarının birbirleri ile bağlanmaları sonucu oluşan ağa *yapay sinir ağı* denmektedir (bkz. Şekil-2.1). Bu ağın oluşturulması biyolojik sinir sistemi hakkındaki bulgulara dayanmaktadır. Biyolojik ve yapay sinir ağlarının arasındaki ilişki ileride açıklanacaktır.



Şekil-2.1. Bir yapay sinir ağı örneği

Teknik olarak da, bir yapay sinir ağının en temel görevi, kendisine gösterilen bir girdi setine karşılık gelebilecek bir çıktı seti belirlemektir. Bunu yapabilmesi için ağ, ilgili olayın örnekleri ile eğitilerek (öğrenme) genelleme yapabilecek yeteneğe kavuşturulur. Bu genelleme ile benzer olaylara karşılık gelen çıktı setleri belirlenir.

Ağı oluşturan proses elemanları, bunların bilgileri işleme yetenekleri, birbirleri ile bağlantılarının şekilleri değişik modelleri oluşturmaktadır. Bu modeller ile ilgili bilgiler ileride ayrıntılı olarak açıklanacaktır. Bu bölümde yapay sinir ağları konusunda genel bilgiler verilmeye çalışılacaktır.

Yapay sinir ağları aynı zamanda, *"bağlantılı ağlar (connectionist networks)"*, *"paralel dağıtılmış ağlar (parallel distributed networks)"*, *"nuromorfik sistemler (neuromorfic systems)"* olarak da adlandırılmaktadır. Yapay sinir ağları bilgisayar bilimine de bazı

yenilikler getirmiştir. Algoritmik olmayan, adaptif, paralel programlama, dağıtılmış programlama vb. gibi tekniklerinin gelişmesine katkıda bulunmuşlardır. Bilgisayarlarında öğrenebileceğini göstermişlerdir. Özellikle olaylar hakkında bilgilerin olmadığı fakat örneklerin bulunduğu durumlarda çok etkin olarak kullanılabilecek bir karar verme aracı ve hesaplama yöntemi olarak görülebilir.

2.3. Yapay Sinir Ağlarının Genel Özellikleri

Yapay sinir ağlarının karakteristik özellikleri uygulanan ağ modeline göre değişmektedir. İlgili modeller anlatılırken her modelin özellikleri ayrıntılı olarak anlatılacaktır. Burada bütün modeller için geçerli olan genel karakteristik özellikler verilmiştir. Bunlar aşağıdaki gibi sıralanabilir:

- ***Yapay sinir ağları makine öğrenmesi gerçekleştirirler***: Yapay sinir ağlarının temel işlevi bilgisayarların öğrenmesini sağlamaktır. Olayları öğrenerek benzer olaylar karşısında benzer kararlar vermeye çalışırlar.
- ***Programları çalışma stili bilinen programlama yöntemlerine benzememektedirler***. Geleneksel programlama ve yapay zeka yöntemlerinin uygulandığı bilgi işleme yöntemlerinden tamamen farklı bir bilgi işleme yöntemi vardır.
- ***Bilginin saklanması***: Yapay sinir ağlarında bilgi ağın bağlantılarının değerleri ile ölçülmekte ve bağlantılarda saklanmaktadır. Diğer programlarda olduğu gibi veriler bir veri tabanında veya programın içinde gömülü değildir. Bilgiler ağın üzerinde saklı olup ortaya çıkartılması ve yorumlanması zordur.
- ***Yapay sinir ağları örnekleri kullanarak öğrenirler***. Yapay sinir ağlarının olayları öğrenebilmesi için o olay ile ilgili örneklerin belirlenmesi gerekmektedir. Örnekleri kullanarak ilgili olay hakkında genelleme yapabilecek yeteneğe kavuşturulurlar (adaptif öğrenme). Örnek bulunamıyorsa ve yok ise yapay sinir ağının eğitilmesi mümkün değildir. Örnekler ise gerçekleşmiş olan olaylardır. Mesela bir doktor hastasına bazı sorular sorar ve aldığı cevaplara göre teşhis ederek ilaç yazar. Sorulan sorular ve verilen cevaplar ile konulan teşhis bir örnek olarak nitelendirilir. Bir doktorun belirli bir zaman içinde hastaları ile yaptığı görüşmeler ve koyduğu teşhisler not edilerek örnek olarak alınırsa yapay sinir ağı benzer hastalıklara benzer teşhisi koyabilir. Elde edilen örneklerin olayı tamamı ile gösterebilmesi çok önemlidir. Ağa olay bütün yönleri ile gösterilemez ve ilgili örnekler sunulmaz ise başarılı sonuçlar elde edilemez. Bu ağın sorunlu olduğundan değil olayın ağa iyi gösterilemediğindendir. O nedenle örneklerin oluşturulması ve toplanması yapay sinir ağı biliminde özel bir öneme sahiptir.
- ***Yapay sinir ağlarının güvenle çalıştırılabilmesi için önce eğitilmeleri ve performanslarının test edilmesi gerekmektedir***. Yapay sinir ağlarının eğitilmesi demek, mevcut örneklerin tek tek ağa gösterilmesi ve ağın kendi mekanizmalarını çalıştırarak örnekteki olaylar arasındaki ilişkileri belirlemesidir. Bu konu kitabın değişik bölümlerinde ayrıntılı olarak anlatılacaktır. Her ağı

eğitmek için elde bulunan örnekler iki ayrı sete bölünürler. Birincisi ağı eğitmek için (eğitim seti) diğeri ise ağın performansını sınamak için (test seti) kullanılır. Her ağ önce eğitim seti ile eğitilir. Ağ bütün örneklere doğru cevaplar vermeye başlayınca eğitim işi tamamlanmış kabul edilir. Daha sonra ağın hiç görmediği test setindeki örnekler ağa gösterilerek ağın verdiği cevaplara bakılır. Eğer ağ hiç görmediği örneklere kabul edilebilir bir doğrulukta cevap veriyor ise o zaman ağın performansı iyi kabul edilir ve ağ kullanıma alınarak gerekirse çevrimiçi (*on-line*) kullanılır. Eğer ağın performansı yetersiz olursa o zaman yeniden eğitmek veya yeni örnekler ile eğitmek gibi bir çözüme gidilir. Bu işlem ağın performansı kabul edilebilir bir düzeye gelinceye kadar devam eder.

- ***Görülmemiş örnekler hakkında bilgi üretebilirler***: Ağ kendisine gösterilen örneklerden genellemeler yaparak görmediği örnekler hakkında bilgiler üretebilirler.
- ***Algılamaya yönelik olaylarda kullanılabilirler***: Ağlar daha çok algılamaya yönelik bilgileri işlemede kullanılırlar. Bu konuda başarılı oldukları yapılan uygulamalarda görülmektedir. Bilgiye dayalı çözümlerde uzman sistemler kullanılmaktadır. Bazı durumlarda yapay sinir ağı ve uzman sistemleri birleştirmek daha başarılı sistemler oluşturmaya neden olmaktadır.
- ***Şekil (örüntü) ilişkilendirme ve sınıflandırma yapabilirler***: Genel olarak ağların çoğunun amacı kendisine örnekler halinde verilen örüntülerin kendisi veya diğerleri ile ilişkilendirilmesidir. Diğer bir amaç ise sınıflandırma yapmaktır. Verilen örneklerin kümelendirilmesi ve belirli sınıflara ayrıştırılarak daha sonra gelen bir örneğin hangi sınıfa gireceğine karar vermesi hedeflenmektedir.
- ***Örüntü tamamlama gerçekleştirebilirler***: Bazı durumlarda ağa eksik bilgileri içeren bir örüntü (*pattern*), veya bir şekil verilir. Ağın bu eksik bilgileri bulması istenir. Örneğim yırtık bir resmin kime ait olduğunu belirlemesi ve tam resmi vermesi gibi bir sorumluluk ağdan istenebilmektedir. Bu tür olaylarda yapay sinir ağlarının çok etkin çözümler ürettiği bilinmektedir.
- ***Kendi kendini organize etme ve öğrenebilme yetenekleri vardır:*** Yapay sinir ağlarının örnekler ile kendisine gösterilen yeni durumlara adapte olması ve sürekli yeni olayları öğrenebilmesi mümkündür.
- ***Eksik bilgi ile çalışabilmektedirler:*** Yapay sinir ağları kendileri eğitildikten sonra eksik bilgiler ile çalışabilir ve gelen yeni örneklerde eksik bilgi olmasına rağmen sonuç üretebilirler. Eksik bilgiler ile de çalışmaya devam ederler. Halbuki geleneksel sistemler bilgi eksik olunca çalışmazlar. Burada bir noktaya dikkatleri çekmekte fayda vardır. Yapay sinir ağlarının eksik bilgiler ile çalışması performanslarının düşeceği anlamına gelmez. Performansın düşmesi eksik olan bilginin önemine bağlıdır. Hangi bilginin önemli olduğunu ağ (*network*) kendisi eğitim sırasında öğrenmektedir. Kullanıcıların bu konuda bir fikri yoktur. Ağın performansı düşük olunca,

kayıp olan bilginin önemli olduğu kararına varılır. Eğer ağın performansı düşmez ise eksik olan bilginin önemli olmadığı anlaşılır.

- ***Hata toleransına sahiptirler:*** Yapay sinir ağlarının eksik bilgilerle çalışabilme yetenekleri hatalara karşı toleranslı olmalarını sağlamaktadır. Ağın bazı hücrelerinin bozulması ve çalışamaz duruma düşmesi halinde ağ çalışmaya devam eder. Ağın bozuk olan hücrelerinin sorumluluklarının önemine göre ağın performansında düşmeler görülebilir. Hangi hücrelerin sorumluluklarının önemli olduğuna da yine ağ eğitim esnasında kendisi karar verir. Bunu kullanıcı bilmemektedir. Ağın bilgisinin yorumlanamamasının sebebi de budur.
- ***Belirsiz, tam olmayan bilgileri işleyebilmektedirler:*** Yapay sinir ağlarının belirsiz bilgileri işleyebilme yetenekleri vardır. Olayları öğrendikten sonra belirsizlikler altında ağlar öğrendikleri olaylar ile ilgili ilişkileri kurarak kararlar verebilirler.
- ***Dereceli bozulma (Graceful degradation) gösterirler:*** Yapay sinir ağlarının hatalara karşı toleranslı olmaları bozulmalarının da dereceli (göreceli) olmasına neden olmaktadır. Bir ağ (*network*) zaman içerisinde yavaş yavaş ve zarif bir şekilde bozulur. Bu eksik olan bilgiden veya hücrelerin bozulmasından kaynaklanır. Ağlar, herhangi bir problem ortaya çıktığında hemen anında bozulmazlar.
- ***Dağıtık belleğe sahiptirler:*** Yapay sinir ağlarında bilgi ağa yayılmış durumdadır. Hücrelerin birbirleri ile bağlantılarının değerleri ağın bilgisini gösterir. Tek bir bağlantının bir anlamı yoktur. Daha önce belirtildiği gibi ağın bilgilerinin açıklanamamasının sebeplerinden birisi de budur. Bu ağlarda, ağın tamamı öğrendiği olayın bütününü karakterize etmektedir. O nedenle bilgiler ağa dağıtılmış durumdadır. Bu ise dağıtık bir belleğin doğmasına neden olmaktadır.
- ***Sadece nümerik bilgiler ile çalışabilmektedirler.*** Yapay sinir ağları sadece nümerik bilgiler ile çalışırlar. Sembolik ifadeler ile gösterilen bilgilerin nümerik gösterime çevrilmeleri gerekmektedir. Bunun nasıl yapıldığı daha sonra açıklanacaktır. Sembolik bilgilerin nümerik değerler ile ifade edilmesinde bilgilerin yorumlanmasını ve kararların (üretilen çözümlerin) açıklanmasını zorlaştırmaktadır.

Yukarıda belirtilen özelliklere ek olarak geliştirilmiş olan her modelin kendisine özgü özellikleri olabilmektedir. Bunlar ilgili bölümlerde açıklanmaktadır...

Burada açıklanan özellikler dikkatlice incelenirse aslında yapay sinir ağlarının bilgisayar bilimine oldukça avantajlı katkılarının olduğu görülebilir. Geleneksel bilgisayar yazılım teknolojisi ile çözülemeyen birçok problemin yapay sinir ağları ile çözülebileceği görülebilir. Mesela yapay sinir ağları, eksik, normal olmayan, belirsiz bilgileri işleyebilen en güçlü problem çözme tekniğidir denilse yanlış olmaz. Belirsiz bilgileri işlemede bulanık önermeler mantığı (*fuzzy logic*) gibi teknikler olsa bile eksik bilgi ile çalışabilen teknikler bulmak çok zordur.

2.4.Yapay Sinir Ağlarının Önemli Dezavantajları

Yapay sinir ağlarının yukarıda belirtilen bir çök avantajlı özelliklerinin yanı sıra bazı dezavantajları da vardır. Bunları kısaca aşağıdaki gibi özetlemek mümkündür:

- Yapay sinir ağlarının donanım bağımlı çalışmaları önemli bir sorun olarak görülebilir. Ağların temel varoluş nedenlerinden birisiide paralel işlemciler üzerinde çalışabilmeleridir. Ağların özellikle, gerçek zamanlı bilgi işleyebilmeleri paralel çalışabilen işlemcilerin varlığına bağlıdır. Günümüzdeki makinelerin çoğu seri şekilde çalışabilmekte ve aynı zamanda sadece tek bir bilgiyi işleyebilmektedir. Paralel işlemleri seri makinelerde yapmak ise zaman kaybına yol açmaktadır. Bunun yanı sıra bir ağın nasıl oluşturulması gerektiğini belirleyecek kuralların olmaması da başka bir dezavantajdır. Her problem farklı sayıda işlemci gerektirebilir. Bazı problemleri çözebilmek için gerekli olan paralel işlemcilerin tamamını bir arada (paralel olarak) çalıştırmak mümkün olmayabilir.
- Probleme uygun ağ yapısının belirlenmesi genellikle deneme yanılma yolu ie yapılmaktadır bu ise önemli bir problemdir. Çünkü eğer problem için uygun bir ağ oluşturulmaz ise çözümü olan bir problemin çözülememesi veya performansı düşük çözümlerin elde edilmesi söz konusu olabilir. Bu aynı zamanda bulunan çözümün en iyi çözüm olduğunu da garanti etmez. Yani yapay sinir ağları kabul edilebilir çözümler üretebilir. Optimum (en iyi) çözümü garanti etmez.
- Bazı ağlarda ağın parametre değerlerinin (mesela öğrenme katsayısı, her katmanda olması gereken proses elemanı (yapay hücrelerin) sayısı, katman sayısı vb. belirlenmesinde de bir kural olmaması diğer bir problemdir. Bu, iyi çözümler bulmayı zor durumda bırakan bir etken olarak görülebilir. Bu parametrelerin belirlenmesi de kullanıcının tecrübesine bağlıdır. Her problem için ayrı faktörleri dikkate almayı gerektirmektedir. Bu parametre değerleri için belirli standartların oluşturulması çok zor olduğundan her problem için ayrı ayrı değerlendirmeler yapılması gerektirmektedir. Bu da önemli bir dezavantaj olarak görülebilir.
- Ağın öğreneceği problemin ağa gösterimi de çok önemli bir problemdir. Yapay sinir ağları yukarıda belirtildiği gibi sadece nümerik bilgiler ile çalışmaktadırlar. Problemin nümerik gösterime dönüştürülmesi lazımdır. Bu ise kullanıcının becerisine bağlıdır. Uygun bir gösterim mekanizmasının kurulamamış olması problemin çözümünü engelleyebilir veya düşük performanslı bir öğrenme (çözüm) elde edilebilir. Problemin nümerik gösterimi mümkün olsa bile bunun ağa gösteriliş şekli problemin başarılı bir şekilde çözülmesini yakından etkiler. Örneğin bir olay hem ayrık (*binary-ikili*) hem de sürekli değerler ile gösterilebilir. Bunun hangisinin daha başarılı bir öğrenme gerçekleştirileceği ise bilinmemektedir. Bu konuda, kullanıcının tecrübesi de yeterli olmayabilir. Bu günümüzde bir çok olayın yapay sinir ağları ile çözülememesinin en önemli nedenlerinden birisidir.

- Ağın eğitiminin ne zaman bitirileceğine karar vermek içinde geliştirilmiş bir yöntem yoktur. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlanması için yeterli görülmektedir. Fakat neticede optimum (en iyi) öğrenmenin gerçekleştiği söylenememektedir. Sadece iyi çözümler üretebilen bir ağ oluştu denilmektedir. Optimum neticeleri veren bir mekanizma henüz geliştirilmemiştir. Bu konuda oldukça önemli olup çözülmesi için araştırmalar yapılması gerekmektedir.
- Bir diğer sorun ise, belki de yukarıdakilerin en önemlisi daha önce açıklandığı gibi ağın davranışlarının açıklanamamasıdır. Bir probleme çözüm üretildiği zaman bunun nasıl ve neden üretildiği konusunda bir bilgi bulmak mümkün değildir. Bu ise ağın sonucuna olan güveni azaltmaktadır.

Bütün bu dezavantajlara rağmen yapay sinir ağları tarafından her problem için değişik şekillerde çözümler üretilebilmekte ve başarılı uygulamalar oluşturmak mümkün olabilmektedir. Bu nedenle, bu dezavantajları yapay sinir ağlarına olan ilgiyi düşürmek için görmemek gerekir. Yapay sinir ağlarının her derde ilaç gibi görülmesinin doğru olmadığını vurgulamak için bunlar burada açıklanmıştır. Ağların bu dezavantajlardan kurtularak problemlere çözüm üretebilmesi için ağların oluşturulmasını titizlik ile gerçekleştirmek gerekmektedir. Hem çözülecek olan problemler hem de yapay sinir ağları konusunda yeterli oranda bilgi sahibi olmak başarılı sonuçlar elde edilmesini sağlayabilir. Yapay sinir ağı geliştirecek olanların bu gerçeği göz ardı etmeden problemlere çözüm üretecek bir ağ oluşturmanın mümkün olabileceğini fakat bunun o kadar kolay olmayacağını bilmesi gerekmektedir.

2.5.Yapay Sinir Ağları İle Neler Yapılabilir?

Yapay sinir ağları günümüzde geliştirilmiş en güncel ve en mükemmel örüntü tanıyıcı ve sınıflandırıcılardan sayılabilirler; bu ağları bu kadar güncel yapan da, yukarıda belirtildiği gibi, eksik bilgiler ile çalışabilme ve normal olmayan verileri işleyebilme yetenekleridir. Özellikle çok sayıda veriyi işleme gerektiren (radar verileri gibi) işlerde çok avantajlı sonuçlar üretebilmektedirler. Günümüzde bir çok problem aslında şekil tanıma problemi haline getirilmekte ve ondan sonra çözülmektedir. Bu nedenle, yapay sinir ağlarının kullanılabileceği bir çok alan vardır. Endüstriyel ve sosyal hayatta görülen binlerce örnek ile başarılı oldukları gösterilmiştir. Fakat her problemi yapay sinir ağı ile çözmek mantıklı olmayabilir. Eğer herhangi bir problemin çözümü için yeterli etkinlikte ve verimlilikte çözüm yöntemi söz konusu ise yapay sinir ağının kullanılmasının bir anlamı yoktur. İlgili olay hakkında örneklerin olmayışı da (bulunamayışı da) bu ağları kullanmamak için önemli bir nedendir. Bir problemin yapay sinir ağı ile çözülmesi için şu şartlardan birini sağlanması gerekir.

- Sadece yapay sinir ağları ile problemlere pratik çözümler üretebilme durumunun söz konusu olması gerekir.
- Başka çözüm yolları olmasına rağmen yapay sinir ağlarının daha kolay ve daha etkin çözümler üretilebilmesinin sağlanması gerekir.

Başarılı uygulamalar incelendiğinde yapay sinir ağlarının, doğrusal olmayan, çok boyutlu, gürültülü, karmaşık, kesin olmayan, eksik, kusurlu, hata olasılığı yüksek sensör verilerinin olması ve problemin çözümü için özellikle bir matematik modelin ve algoritmanın bulunmaması hallerinde yaygın olarak kullanıldıkları görülmektedir. Bu amaçla geliştirilmiş ağlar genel olarak şu fonksiyonları yerine getirmektedir:

- Probabilistik fonksiyon kestirimleri
- Sınıflandırma
- İlişkilendirme veya örüntü eşleştirme
- Zaman serileri analizleri
- Sinyal filtreleme
- Veri sıkıştırma
- Örüntü tanıma
- Doğrusal olmayan sinyal işleme
- Doğrusal olmayan sistem modelleme
- Optimizasyon
- Zeki ve doğrusal olmayan kontrol

Yukarıda listelenen konularda teorik uygulamaların ötesinde günlük hayatta kullanılan finansal konulardan mühendisliğe ve tıp bilimine kadar bir çok uygulamadan bahsetmek mümkündür. Bunlardan bazıları ise şöyle sıralanabilir:

- Veri madenciliği
- Optik karakter tanıma ve çek okuma
- Bankalardan kredi isteyen müracaatları değerlendirme
- Ürünün pazardaki performansının tahmin etme
- Kredi kartı hilelerini saptama
- Zeki araçlar ve robotlar için optimum rota belirleme
- Güvenlik sistemlerinde konuşma ve parmak izi tanıma
- Robot hareket mekanizmalarının kontrol edilmesi
- Mekanik parçaların ömürlerinin ve kırılmalarının tahmin edilmesi
- Kalite kontrolü
- İş çizelgelem ve iş sıralaması
- İletişim kanallarındaki geçersiz ekoların filtrelenmesi
- İletişim kanallarındaki trafik yoğunluğunu kontrol etme ve anahtarlama
- Radar ve sonar sinyalleri sınıflandırma
- Üretim planlama ve çizelgeleme
- Kan hücreleri reaksiyonları ve kan analizlerini sınıflandırma
- Kanserin saptanması ve kalp krizlerinin tedavisi
- Beyin modellenmesi çalışmaları

Bunların çoğaltılması mümkündür. Yukarıdakiler yalnızca genel olarak hangi alanlarda kullanılacaklarına göstermek amacıyla verilmiştir; yoksa hemen hemen her alanda örneklerini görmek mümkündür. Çünkü gerçek hayatta kullanılan sistemlerin çoğu doğrusal olmayan modellemeler gerektirmektedir. Bu ise geleneksel yöntemler ile çözüm üretilmesini zorlaştırmakta bazen de imkansızlaştırmaktadır. Kitabın en son bölümünde yapay sinir ağlarının uygulamaları ayrıntılı olarak yeniden irdelenecektir.

2.6. Yapay Sinir Ağlarının Kısa Bir Tarihçesi

İnsanoğlu tarih boyunca sürekli insan beyninin nasıl çalıştığını merak etmiştir. Bilgisayarların doğmasında aslında bu merakın bir neticesidir. İlk hesap makinelerinden günümüzdeki çok karmaşık bilgisayar sistemlerine geçişin temelinde bu merak ve arayışın rolünü unutmamak gerekmektedir. Gelişmelere bakarak gelecekte daha karmaşık sistemlerin çıkacağını da kestirmek zor değildir. Bilgisayarlar başlangıçta sadece aritmetik işlemler yapmak amacı ile geliştirilmiş iken, bugün olayları öğrenmeleri ve çevre şartlarına göre karar vermeleri istenmektedir. Gelecekte insanoğlunun gerçekleştirdiği çok yüksek oranda beyin gücü gerektiren işleri yapmalarının bekleneceğini kestirmek zor değildir. Yapay sinir ağları günümüzde bu gelişmeyi tetikleyen bilim dallarından birisidir. Gelecekte de yine en önemli bilim dallarından birisi olacaktır.

Yapay sinir ağlarının tarihçesi nörobiyoloji konusuna insanların ilgi duyması ve elde ettikleri bilgileri bilgisayar bilimine uygulamaları ile başlamaktadır. Yapay sinir ağları ile ilgili çalışmaları 1970 öncesi ve sonrası diye ikiye ayırmak gerekmektedir. Çünkü, 1970 yılında bu bilimin tarihinde bir önemli dönüm noktası başlamış ve o zamana kadar olmaz diye düşünülen bir çok sorun çözülmüş ve yeni gelişmeler başlamıştır. Herşey bitti derken yapay sinir ağları yeniden doğmuştur.

2.6.1. 1970 Öncesi Çalışmalar

İnsan beyninin nasıl çalıştığı ve fonksiyonları uzun yıllar araştırılmıştır. 1890 yılında beynin fonksiyonları hakkında bilgi veren ilk eser yayınlanmıştır [2] 1940 dan önceki yıllarda bazı bilim adamlarının (`Helmholtz`, `Pavlov`, `Poincare` vb.) yapay sinir ağı kavramı üzerinde çalıştıkları bilinmektedir. Fakat bu çalışmaların mühendislik değerinin olduğu söylenemez. 1940'lı yıllardan sonra `Hebb`, `McCulloch` ve `Pitts` [3,4] gibi bilim adamları yapılan araştırmaları mühendislik alanlarına kaydırmaya ve günümüzdeki yapay sinir ağlarının temellerini oluşturmaya başladılar. İlk yapay sinir hücresinin yapısını oluşturdular. Yapay sinir hücreleri ile her türlü mantıksal ifadeyi formülüze etmenin mümkün olduğunu gösterdiler. Hücrelerin birbirleri ile paralel çalışması gerektiği fikrini ortaya atarak öğrenme kurallarını belirlemeye başladılar. 1949 yılında `Donald Hebb`, yapay hücrelerden oluşan bir yapay sinir ağının değerlerini değiştiren bir öğrenme kuralı geliştirdi [4]. ***"Hebbian öğrenme"*** kuralı denilen bu kural günümüzde de bir çok öğrenme kuralının temelini oluşturmaktadır. 1950'li yıllarda çalışmalar daha açık bir şekilde fark edildi ve 1951 yılında ilk nuro-bilgisayar üretildi. Silicon teknolojinin geliştirilmesi ile bu çalışmalar 1960'lı yıllarda oldukça önemli gelişmelere neden oldu. 1954 yılında `Farley` and `Clark` tarafından rassal ağlar (*random networks*) ile adaptif tepki üretme kavramı ortaya atıldı [5] ve bu kavram daha sonraları 1958 yılında `Rosenblatt` [6] ve 1961 yılında `Caianiello` [7] tarafından geliştirildi. Özellikle `Rosenblatt` tarafından geliştirilen algılayıcı model (perceptron) yapay sinir ağları tarihinde önemli bir gelişmeye öncülük etmiştir. Çünkü bu model, daha sonraları geliştirilecek ve yapay sinir ağlarında devrim niteliğinde olacak olan çok katmanlı algılayıcıların temelini oluşturmaktadır.

Benzer şekilde Widrow ve Hoff ADALINE (ADaptive LInear NEuron) modelini ortaya attılar [8]. Bu aslında yapay sinir ağlarının mühendislik uygulamalarına başlanması için ilk adımlardan sayılmaktadır. Bu model, Rosenblatt'ın algılayıcı modeli ile aynı niteliklere sahip bir model olup sadece öğrenme algoritması daha gelişmiş bir modeldir. Daha sonraki yapay sinir ağları modellerinin gelişmesine katkıda bulunmuş bir çalışmadır. Adaptif öğrenmenin de temellerinden olan ve 1970'li yılların sonlarına doğru ortaya çıkan MADALINE modelleri bu çalışmaların neticesinde ortaya çıkmış ve gelecekte faydalı çalışmalara temel oluşturmuştur.

Bu arada bilim dünyasında başka gelişmeler olmuştur. 1956 yılında "Yapay Zeka" kavramı ortaya atılmış ve bilim dünyasında kabul görmüştür. İlk yapay zeka çalışmaları yapay sinir ağırlıklarına pek değinmemiştir. Herkes ilgisini yapay zekaya çevirmiş ve nöro-bilgisayar ve yapay sinir ağları popülaritesini yitirmeye başlamıştır. Bu gidişata dur demek ve araştırmacıların ilgisini yapay sinir ağları üzerine tekrar çekmek için 1960'lı yıllarda Grosberg, Kohonen, Rosenblatt, Widrow, Nilssons, Fukushima vb. gibi bilim adamları tekrar konunun üzerine gitmeye başladılar. Özellikle Nilssons tarafından yazılan ilke ***"Öğrenen makineler"*** başlıklı kitap [9] yapılan çalışmaların teorik bir çerçevesini oluşturmaya ve çalışmaları bir araya getirmeye neden olmuştur.

Yapay zeka bilimi geliştikçe yapay zekacılar kendilerini her geçen gün daha çok öne çıkartmak amacı ile yapay sinir ağlarının çalışmalarını yakından takip ediyor ve eleştirilerde bulunuyorlardı.

1960'lı yılların sonunda yapay sinir ağı çalışmaları duraklama devrine girdi. Yapay sinir ağlarının tarihinde bir duraklama devrine neden olan ise Yapay Zeka biliminin o devirde önde gelen isimlerinden Misnky ve Pappert tarafından yazılan **algılayıcılar** (*perceptrons*) başlıklı kitap oldu [10]. Bu kitapta yazarlar özellikle yapay sinir ağlarına dayalı algılayıcıların bilimsel bir değerinin olmadığını ve doğrusal (*linear*) olmayan problemlere çözüm üretemediğini iddia ettiler. Tezlerini kanıtlamak için ise meşhur XOR probleminin çözülmemesini örnek gösterdiler. Bu örnek bir çok kişiyi tatmin etti ve çalışmalar bir bıçak gibi kesildi. Yapay sinir ağı yapmak mümkün değildir inancı yükselmeye başladı. Bir çok bilim adamı buna inandığından çalışmalar yok denecek kadar azaldı. Amerika Birleşik Devletlerinde araştırma geliştirme çalışmalarını yürüten ve eşgüdümünü sağlayan bir organizasyon olan DARPA yapay sinir ağları ile ilgili çalışmaları desteklemeyi durdurdu. Sadece bir iki bilim adamı bu konuda çalışmaya devam etti. Çalışmalar öyle kesildi ki, XOR problemi çözülünceye kadar dikkatleri yapay sinir ağına çekmek mümkün olamadı.

Bu zaman kadar yapılan çalışmaların bazıları kronolojik olarak aşağıdaki gibi listelenebilir:

1890- İnsan beyninin yapısı ve fonksiyonları ile ilgili ilk yayının yazılması

1911- İnsan beyninin bileşenlerinin belirli bir düzenek ile sinir hücrelerinden (nüronlar) oluştuğu fikrinin benimsenmesi

1943- Yapay sinir hücrelerine dayalı hesaplama teorisinin ortaya atılması ve eşik değerli mantıksal devrelerin (*threshold logic device*) geliştirilmesi

1949- Biolojik olarak mümkün olabilen öğrenme prosedürünün bilgisayarlar tarafından gerçekleştirilecek biçimde geliştirilmesi

1956-1962- ADALINE ve Widrow öğrenme algoritmasının geliştirilmesi

1957-1962- Tek katmanlı algılayıcının (*perceptron*) geliştirilmesi

1965- İlk makine öğrenmesi kitabının yayınlanması

1967-1969- Bazı gelişmiş öğrenme algoritmalarının (*Grosberg* öğrenme algoritması gibi) geliştirilmesi

1969- Tek katmanlı algılayıcıların problemleri çözme yeteneklerinin olmadığının gösterilmesi

1969- DARPA'nın yapay sinir ağlarını desteklemeyi durdurup diğer yapay zeka çalışmalarına destek vermesi

2.6.2. 1970 Sonrası Çalışmalar

Çalışmaların 1969 yılında sekteye uğraması ve gerekli finansal desteklerin kesilmesine rağmen bazı bilim adamları çalışmalarına devam ettiler. Özellikle Amari, Anderson, Cooper, Fukushima, Grossberg, Kohonen ve Hopfield gibi araştırıcıların çalışmaları 1980'li yıllara gelindiğinde meyvelerini vermeye başladı ve yapay sinir ağları çalışmalarındaki sessizlik sona erdi. 1972'lerde farklı disiplinlerde çalışan elektrik mühendisi Kohonen [11] ve nöropsikolojist Anderson [12] "çağrışımlı bellek*" (*associative memory*) konusunda hemen hemen birbirinin aynı çalışmalar yayınladılar. Bu çalışmalar daha sonraları geliştirilecek olan öğretmensiz öğrenme kurallarının temeli oldu. Kohonen daha sonra 1982 yılında "kendi kendine öğrenme nitellik haritaları (*self organzinig feature maps*- SOM) konusundaki çalışmasını yayınladı [13]. 1960'lı yılların sonlarına doğru sahneye çıkan Grosberg yapay sinir ağlarının psikolojik mantıksallığı ve mühendislik uygulamalarındaki kolaylığını gösterdi; Carpenter ile Adaptif Rezonans Teorisini (ART) geliştirdi. Bu öğretmensiz öğrenme konusunda zamanının geliştirilmiş en karmaşık yapay sinir ağı oldu [14-17].

1970'lerin sonlarına doğru Fukushima görsel şekil ve örüntü tanıma amaçlı geliştirdiği NEOCOGNITRON modelini tanıttı. Bu model önceleri öğretmensiz öğrenme yapan bir model olacak şekilde geliştirilmesine rağmen daha sonraları öğretmenli öğrenme yapacak hale getirilmiştir [18,19]. Bu çalışmaların neticesinde daha çok mühendislik uygulamaları görülmeye başlandı. Çalışmalar biyolojik olarak doğruluktan daha çok sonuçların kullanılabilirliği konusuna ağırlık veriyordu [20,21]. NEOCOGNITRON modelinde ara katmanlar kullanılarak öğrenme konusuna değiniyordu. Fakat çalışmalar hesaplama proseslerinin bilinmesini zorunlu kılıyordu. Bu çalışmalar günümüzde de etkin olarak yürütülmektedir.

1982 ve 1984 yıllarında Hopfield tarafından yayınlanan çalışmalar [22,23] ile yapay sinir ağlarının genelleştirilebileceği ve özellikle geleneksel bilgisayar programlama ile

* *Associative Memory,* bazı kaynaklarda içeriğiyle adreslenebilen bellek olarak ta kullanılmaktadır.

çözülmesi zor olan problemlere çözüm üretebileceğini gösterdi. Gezgin satıcı problemini çözmesi bunun en güzel örneğiydi. Çalışmasını mühendislerin kolaylıkla anlayabileceği şekilde sunduğundan yapay sinir ağlarına ilgi yeniden kurulmaya başlandı. Çalışmaların neticesi Hinton ve arkadaşlarının geliştirdikleri Boltzman makinesini doğmasına yol açıyordu.

Aynı zamanlarda Rummelhart ve arkadaşları paralel programlama konularındaki çalışmalarını sonuçlandırıyor ve 2 ciltlik bir eser ortaya koyuyordu [24-26]. Bu eserlerinden çok katmanlı algılayıcı modelini temellerini atıyorlar ve daha sonra bu modeli geliştiriyorlardı [27]. Çok katmanlı algılayıcıların bulunması yapay sinir ağlarının tarihsel gelişimi bakımından çok önemli bir adım oluyordu. Bu çalışmalardan sonra yapay sinir ağların olan ilgi yeniden ateşlendi. Yapılan seminerlerde çok sayıda makaleler sunuluyor ve yeni bir yapay sinir ağı dalgası bütün disiplinlerde kendini göstermeye başlıyordu. Çünkü tek katmanlı algılayıcının çözemediği XOR problemi çok katmanlı algılayıcıların bulunması ile çözülmüş ve yapay sinir ağlarının çalışmadığını söyleyen bütün tezler çürütülmüştü.

Aynı zamanlarda Parker [28] ve Werbos [29] tarafından da çok katmanlı algılayıcı ile ilgili olarak bazı çalışmalarda yürütülüyordu. Çok katmanlı algılayıcı sadece XOR problemini çözmekle kalmamış aynı zamanda Hopfield ve Boltzman makinelerinin sınırlamalarını da çözmüştü. Buda dikkatleri daha çok bu ağlar üzerine çekiyordu.

1988'de Broomhead ve Lowe Radyal tabanlı fonksiyonlar (*Radial Basis Funcitons*-RBF) modelini geliştirdiler [30]. Bu ağın çok katmanlı algılayıcılara alternatif olarak geliştirildiğini belirttiler. Özellikle filtreleme problemlerine oldukça başarılı sonuçlar ürettiler. Daha sonra Specht bu ağların daha gelişmiş şekli olan Probabilistik Ağlar (PNN) [31] ve Genel Regrasyon Ağları (GRNN) [32] geliştirdi.

1987 yılından bu yana her sene değişik sempozyumlar ve konferanslar ile yapay sinir ağları tartışılmakta ve yeni modeller ve öğrenme teknikleri ortaya atılmaktadır.

Yapay sinir ağlarındaki yeniden dirilişe donanım teknolojisindeki gelişmelerinde katkısı da büyük olmuştur. Bilgisayarlar boyut olarak küçülmüş fakat kapasite ve yetenek bakımından sürekli büyümüş ve gelişmişlerdir. Optik ve dijital teknolojilerdeki gelişmeler ve VLSI teknolojileri gibi teknolojilerin keşfedilmesi bilgisayarların hızlarını artırmış ve yapay sinir ağlarının kullanımını kolaylaştırmıştır.

Günümüzde yapay sinir ağları artık teorik ve laboratuar çalışmaları olmaktan çıkmış ve günlük hayatta kullanılan sistemler oluşturmaya ve pratik olarak insanlara faydalı olmaya başlamışlardır.

1970 yılından sonra yapılan çalışmaların bazılarını kronolojik olarak aşağıdaki gibi listelenebilir:

1) 1969-1972- Doğrusal ilişkilendiricilerin geliştirilmesi
2) 1972- Korelasyon Matriks belleğinin geliştirilmesi
3) 1974- Geriye yayılım modelinin (çok katmanlı algılayıcının ilk çalışmalarının geliştirilmesi
4) Öğretmensiz öğrenmenin geliştirilmesi
   - 1978- ART modelinin geliştirilmesi
   - 1982- *Kohonen* öğrenmesi ve SOM *modelling* geliştirilmesi
5) 1982- *Hopfield* ağlarının geliştirilmesi
6) 1982- Çok katmanlı algılayıcının geliştirilmesi
7) 1984- *Boltzman* makinesinin geliştirilmesi
8) 1985- Çok katmanlı algılayıcıların (genelleştirilmiş Delta öğrenme kuralı ile)
9) 1988- RBF *modellining* geliştirilmesi
10) 1988- PNN *modellining* geliştirilmesi
11) 1991- GRNN *modellining* geliştirilmesi
12) 1991'den günümüze sayısız çalışma ve uygulamalar geliştirilmiştir. Bunların listesini burada vermek nerede ise imkansızdır.

2.7. Özet

Yapay Sinir ağları insan beyninin em temel özelliği olan öğrenme fonksiyonunu gerçekleştiren bilgisayar sistemleridir. Öğrenme işlemini örnekler yardımı ile gerçekleştirirler. Bu ağlar birbirine bağlı proses elemanlarından (yapay sinir hücrelerinden) oluşur. Her bağlantının bir ağırlık değeri vardır. Yapay sinir ağının sahip olduğu bilgi bu ağırlık değerlerinde saklı olup ağa yayılmıştır.

Yapay sinir ağları bilinen hesaplama yöntemlerinden farklı bir hesaplama yöntemi önermektedir. Bulundukları ortama uyum sağlayan, adaptif, eksik bilgi ile çalışabilen, belirsizlikler altında karar verebilen, hatalara karşı toleranslı olan bu hesaplama yönteminin hayatın hemen hemen her alanında başarılı uygulamalarını görmek mümkündür. Oluşturulacak olan ağın yapısının belirlenmesinde, ağ parametrelerinin seçiminde, belirli bir standardın olmaması, problemlerin sadece nümerik bilgiler ile gösterilebilmesi, eğitimin nasıl bitirileceğinin bilinmemesi ve ağın davranışlarını açıklayamamasına rağmen bu ağlara olan ilgi her geçen gün artmaktadır. Özellikle, sınıflandırma, örüntü tanıma, sinyal filtreleme, veri sıkıştırma ve optimizasyon çalışmalarında yapay sinir ağları en güçlü teknikler arasında sayılabilirler. Veri madenciliği, optik karakter taşıma, optimum rota belirleme, parmak izi tanıma, malzeme analizi, iş çizelgelemesi ve kalite kontrol, tıbbi analizler gibi bir çok alanda günlük hayatımızda göreceğimiz başarılı örneklerine rastlamak mümkündür.

Yapay sinir ağlarının tarihsel gelişimine bakıldığında ise 1970 yılının bir dönüm noktası olduğu görülmektedir. Bu tarihten önce bir çok araştırmanın yapıldığı ve 1969 yılında XOR probleminin çözülememesi nedeni ile araştırmaların durduğu görülmektedir. 1970 yılından sonra sınırlı sayıda araştırmacının çalışmalarını sürdürmeleri ve XOR problemini çözmeleri sonucunda yapay sinir ağlarına olan ilgi yeniden alevlenmiştir. İzleyen 10 yıl içinde birbirinden farklı 30 civarından yeni model geliştirildi. Aynı zamanda çalışmalar laboratuarlardan çıkarak günlük hayatta kullanılan sistemler haline geldi. Bu çalışmalar hem yapay zeka hem de donanım teknolojisindeki gelişmeler ile de desteklenmiştir. Artık bilgisayarlarında öğrenebileceğini herkes kabul etmekte ve bu teknolojiden faydalanmak istemektedir.

2.8. Kaynakça

[1] Kohonen T., (1987), State of the art in neural computing, Proc.of the IEEE first international conferans on neural networks, San Diago, 21-24 June 1987, California, Vol 1., pp 77-91.

[2] James, W. (1890), ``Psychology (Briefer Course)", New York: Holt, , Chapter XVI,``Association", pp. 253-279.

[3] McCulloch, W. S. and Pitts, W. A. (1943),``A logical calculus of the ideas immanent in nervous activity", Buttetin of Mathematics and Biophysics, 5, pp. 115-133.

[4] Hebb, D. O., (1949), "The organization of behaviour", New York: Wiley, Introduction and Chapter 4,``The first stage of perception: growth of the assembly", pp. xi-xix, 60-78.

[5] Farely, B. G. and Clark, W. A. (1954),``Simulation of self-organizing systems by digital computers", IEE Transactions of Professional Group of Information Theory, PGIT-4, pp. 76-84.

[6] Rosenblatt, F.(1958), "The perceptron: A probabilistic model for information storage and organization in the brain", Psychoanalytic Review, 65, , pp. 386-408.

[7] Caianiello, E. R.(1961),``Outline of a theory of thought-processes and thinking machines", Journal of Theoretical Biology, 2, pp. 204-235.

[8] Widrow, B. and Hoff, M. E. (1960), ``Adaptive switching circuits", WESTCON Convention, Record Part IV, 1960, pp. 96-104.

[9] Nilson, N. J., (1965),``Learning Machines", McGraw-Hill,

[10] Minsky, M. and Papert, S. (1969),``Perceptrons", MIT Press, Cambridge, MA.

[11] Kohonen, T. (1972),``Correlation matrix memories", IEEE Transaction on Computers, C-21(4), , pp. 353-359.

[12] Anderson, J. A. (1972),``A simple neural network generating on interactive memory", Mathematical Biosciences, 14, pp. 197-220.

[13] Kohonen, T. (1982),``Self-organized formation of topologically correct feature maps", Biological Cybernetics, 43, 1982, pp. 59-69.

[14] Grossberg, S. A. (1973),``Contour enhancement, short term memory, and consultancies in reverberating neural networks", Studies in Applied Mathematics, 52(3), 1973, pp. 213-257.

[15] Grossberg, S. A. (1982),``Studies of mind and brain", Reidel Press, Dordrecht, Holland.

[16] Grossberg, S. A. (1988),``Neural networks and natural intelligence", MIT Press, Cambridge, MA.

[17] Carpenter, G. A. and Grossberg, S. A. (1987),``ART2: self-organization of stable category recognition codes for analog input patterns", Applied Optics, 26(3), pp. 4919-4930.

[18] Fukushima, K. (1980), "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", Biological Cybernetics, 36, pp. 193-202.

[19] Fukushima, K.,``Neocognitron: a new algorithm for pattern recognition of deformations and shifts in position", Pattern Recognition, 15, 1982, pp. 455-469.

[20] Fukushima, K., Miyake, S. and Ito T. (1983),``Neocognitron: a neural network model for a mechanism of visual pattern recognition", IEEE Transactions on Systems, Man and Cybernetics, SMC-13.

[21] Fukushima, K.,``A neural network model for selective attention in visual pattern recognition", Biol. Cybernetics, 55, 1986, pp. 5-15.

[22] Hopfield, J. J. (1982),``Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National. Academy of Sciences, 79, , pp. 2554-2558.

[23] Hopfield, J. J.(1982) ,``Neurons with graded response have collective computational properties like those of two state neurons", Proc. Natl. Acad. Sci., 81, pp. 3088-3092.

[24] Rumelhart, D. E. and McClelland, J. L. (1986),``Parallel distributed processing, explorations in the microstructure of cognition", Vol 1: Foundations, MIT Press, Cambridge, MA.

[25] McClelland, J. L. and Rumelhart, D. E. (1986), "Parallel distributed processing, explorations in the microstructure of cognitio", Vol 2: Psychological and biological models, MIT Press, Cambridge, MA.

[26] Rumelhart, D. E. and McClelland, J. L. (1988), "Parallel distributed processing, explorations in the microstructure of cognition, A handbook of models, programs, and exercises", MIT Press, Cambridge, MA.,

[27] Rumelhart, D. E., Hinton, D. E. and Williams, R. J. (1986), "Learning representation by backpropagating errors" Nature 323(9), pp. 533-536.

[28] Parker, D. B. (1985), "Learning-logic", M.I.T. Cen. Computational Res. Economics Management Sci., Cambridge, MA, TR-47.

[29] Werbos, P. J. (1974),``Beyond regression: new tools for prediction and analysis in the behavioural sciences", PhD. dissertation, Committee on Appl. Math., Harvard Univ., Cambridge, M. A.

[30] Broomhead, D. S., Lowe, D (1988).,``Radial basis-functions, multi-variable functional interpolation and adaptive networks", Royal signals and radar establishment memorandum 4148.

[31] Specht, D. F. (1988),"Probabilistic neural networks for classification, mapping, or associative memory", IEEE Conference on Neural Networks, Vol. I, San Diego, pp. 525-532.

[32] Specht, D. F. (1991),"A general regression neural network", IEEE Transactions on Neural Networks, Vol. 2, No. 6, pp. 568-576.

[33] Haykin, S. (1994), "Neural Networks, A Comprehensive Foundation'", Macmillan College Publishing Co. Inc.

BÖLÜM 3

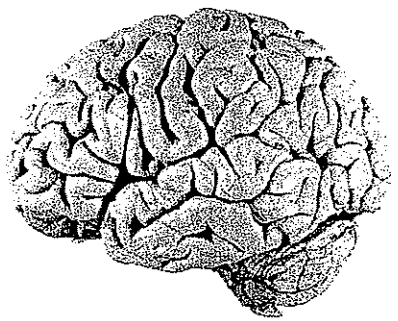
YAPAY SİNİR AĞLARININ YAPISI VE TEMEL ELEMANLARI

Önceki bölümlerde yapay zeka, makine öğrenmesi ve yapay sinir ağlarına genel bir giriş yapılmıştır. Bu bölümde ise yapay sinir ağlarının yapısı ve temel elemanları açıklanacaktır. Daha önce geliştirildiği gibi, yapay sinir ağları biyolojik sinir sisteminden esinlenerek geliştirilmiştir. O nedenle öncelikle biyolojik sinir sistemine bir göz atmakta yarar vardır. Sinir sistemi birbiri ile iletişim halinde olan sinir hücrelerinden oluşmaktadır. Aşağıda bir sinir hücresinin yapısı açıklanmıştır.

3.1. Biyolojik Sinir Hücreleri

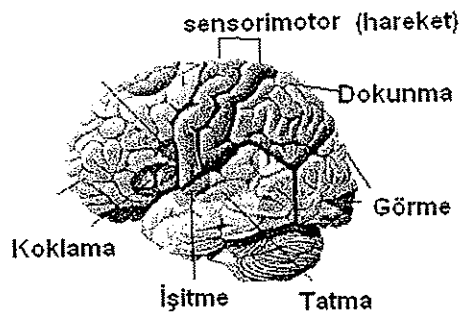
Biyolojik sinir ağları beynimizde bulunan bir çok sayıda sinir hücresinin bir koleksiyonudur. Bir sinir ağı milyarlarca sinir hücresinin bir araya gelmesi ile oluşmaktadır. Sinir hücreleri birbirleri ile bağlanarak fonksiyonlarını yerine getirirler. Beynimizde 10^{10} adet sinir hücresi ve bunlarında $6x10^{13}$'ten fazla sayıda bağlantısının olduğu söylenmektedir. İnsan beyni, çok hızlı çalışabilen mükemmel bir bilgisayar gibi görülebilir. Bir grup insan resmi içinden tanıdık bir resmi 100-200 ms gibi kısa bir sürede fark edebilir. Halbuki geleneksel bilgisayarların böyle bir tanıma işlemini yapması çok daha uzun zamanlar alabilir. Bugün insan beyinin kapasitesinin çok küçük bir oranında kapasiteye sahip ve çalışabilen bir makine yapılsa olağanüstü bilgi işleme ve kontrol edebilme mekanizmaları geliştirmek ve mükemmel sonuçlar elde etmek mümkün olabilir. Biyolojik sinir ağlarının performansları küçümsenemeyecek kadar yüksek ve karmaşık olayları işleyebilecek yeteneklerdir. Yapay sinir ağları ile bu yeteneğin bilgisayara kazandırılması amaçlanmaktadır.

Biyolojik sinir ağları insan beyninin çalışmasını sağlayan en temel taşlardan birisidir. İnsanın bütün davranışlarını ve çevresini anlamasını sağlarlar. Biyolojik sinir ağları beş duyu organından gelen bilgiler ışığında geliştirdiği algılama ve anlama mekanizmalarını çalıştırarak olaylar arasındaki ilişkileri öğrenir. Şekil-3.1 insan beyninin resmini göstermektedir.



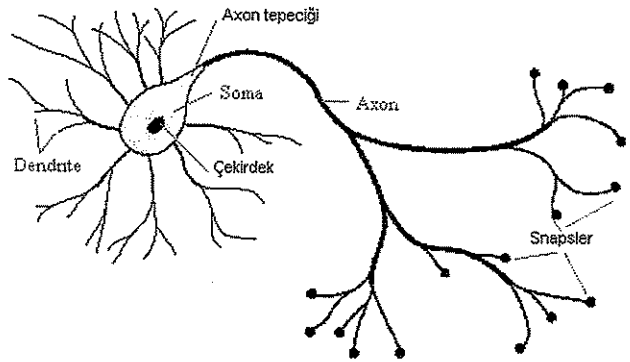
Şekil-3.1. İnsan beyni

İnsan beyninin değişik bölgeleri değişik fonksiyonları yerine getirmektedir. Şekil-3.2. ise beş duyunun beyin içindeki bölgelerini göstermektedir:



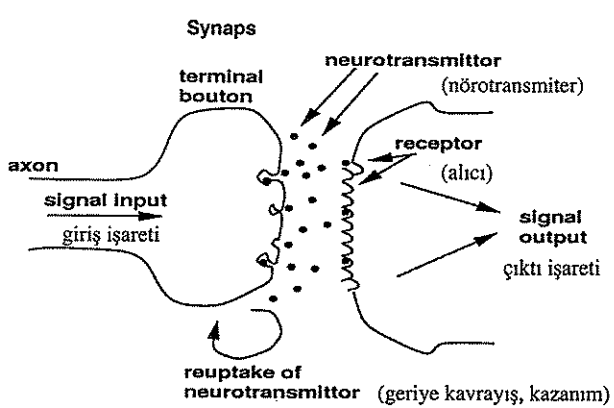
Şekil-3.2. Beyin üzerinde beş temel duyu bölgesi

Duyu organlarından gelen bilgiler (sinyaller) beyin sinir sistemi sayesinde beyine taşınır ve beynin oluşturduğu kararları da yine sinir sistemi tarafından vücudun organlarına eylem olarak gönderilir. Bir sinir hücresi Şekil-3.3'te şematik olarak gösterilmektedir.

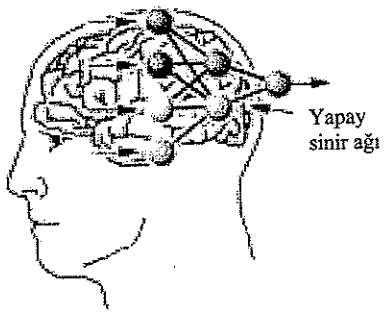


Şekil-3.3. Bir biyolojik sinir hücresinin yapısı

Şekil-3.3'de gösterildiği gibi temel bir biyolojik sinir hücresi *sinapsler*, *soma*, *axon*, ve *dentrite*'lerden oluşmaktadır. *Snapsler* sinir hücreleri arasındaki bağlantılar olarak görülebilir. Bunlar fiziksel bağlantılar olmayıp bir hücreden diğerine elektrik sinyallerinin geçmesini sağlayan boşluklardır. Bu sinyaller somaya giderler. Soma bunları işleme tabi tutar, sinir hücresi kendi elektrik sinyalini oluşturur ve *axon* aracılığı ile *dendrite*'lere gönderir. *Dendrit*'ler ise bu sinyalleri *snapslere* göndererek diğer hücrelere gönderilir. İki hücrenin birbirleri ile bilgi alış verişi *snaptik* bağlantılarda *neurotransmitter*'lar yolu ile sağlanmaktadır. Şekildeki *axon* uçlarının her birisi başka bir hücre ile birleşmektedir. Şekil-3.4 iki biyolojik hücrenin *neurotransmitter*'ler yolu ile bilgi alış verişini göstermektedir:



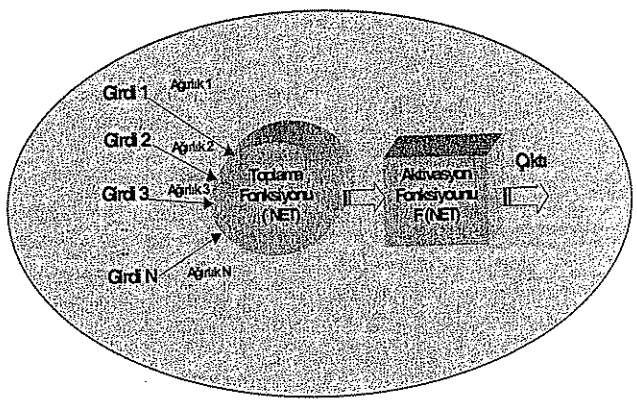
Verilen özellikte milyarlarca sinir hücresi bir araya gelerek sinir sistemini oluşturmaktadır. Yapay sinir ağları biyolojik hücrelerin bu özelliklerinden yararlanarak geliştirilmiştir. Biyolojik sinir ağlarının yapay sinir ağlarına bir dayanak olduğu şematik olarak Şekil-3.5'de gösterilmiştir:



Şekil-3.5. Biyolojik ve yapay sinir ağları

3.2. Yapay Sinir Hücresi (Proses Elemanı)

Biyolojik sinir ağlarının sinir hücreleri olduğu gibi yapay sinir ağlarının da yapay sinir hücreleri vardır. Yapay sinir hücreleri mühendislik biliminde *proses elamanları* olarak da adlandırılmaktadır. Her proses elemanının 5 temel elemanı vardır (bkz. Şekil-3.6). Bunlar:



Şekil-3.6. Yapay sinir hücresinin yapısı

❶ **Girdiler:** Bir yapay sinir hücresine (proses elemanına) dış dünyadan bilgilerdir. Bunlar ağın öğrenmesi istenen örnekler tarafından belirlenir. Yapay sinir hücresine dış dünyadan olduğu gibi başka hücrelerden veya kendi kendisinden de bilgiler gelebilir.

❷ **Ağırlıklar**: Ağırlıklar bir yapay hücreye gelen bilginin önemini ve hücre üzerindeki etkisini gösterir. Şekildeki Ağırlık 1, Girdi 1'in hücre üzerindeki etkisini göstermektedir. Ağırlıkların büyük yada küçük olması önemli veya önemsiz olduğu anlamına gelmez. Bir ağırlığın değerinin sıfır olması o ağ için en önemli olay olabilir. Eksi değerler önemsiz demek değildir. O nedenle artı veya eksi olması etkisinin pozitif veya negatif olduğunu gösterir. Sıfır olması ise herhangi bir etkinin olmadığını gösterir. Ağırlıklar değişken veya sabit değerler olabilirler.

❸ **Toplama fonksiyonu:** Bu fonksiyon, bir hücreye gelen net girdiyi hesaplar. Bunun için değişik fonksiyonlar kullanılmaktadır. En yaygın olanı ağırlıklı toplamı bulmaktır. Burada her gelen girdi değeri kendi ağırlığı ile çarpılarak toplanır. Böylece ağa gelen net girdi bulunmuş olur. Bu şu şekilde formülize edilmektedir.

$$NET = \sum_{i}^{n} G_i A_i$$

Burada G girdileri, A ise ağırlıkları, n ise bir hücreye gelen toplam girdi (proses elemanı) sayısını göstermektedir. Yalnız yapay sinir ağlarında daima bu formülün kullanılması şart değildir. Uygulanan yapay sinir ağı modellerinden bazıları kullanılacak toplama fonksiyonunu belirleyebilmektedir. Literatürde yapılan araştırmalarda toplama fonksiyonu olarak değişik formüllerin kullanıldığı görülmektedir. Tablo-3.1'de değişik toplama fonksiyonlarına örnekler verilmektedir. Görüldüğü gibi, bazı durumlarda gelen girdilerin değeri dikkate alınırken bazı durumlarda ise gelen girdilerin sayısı önemli olabilmektedir. Bir problem için en uygun toplama fonksiyonunu belirlemek için bulunmuş bir formül yoktur. Genellikle deneme yanılma yolu ile toplama fonksiyonu belirlenmektedir. Bir yapay sinir ağında bulunan proses elemanlarının tamamının aynı toplama fonksiyonuna sahip olmaları gerekmez. Her proses elemanı bağımsız olarak farklı bir toplama fonksiyonuna sahip olabilecekleri gibi hepsi aynı proses elemanına sahip olabilir. Hatta ağın bazı proses elemanları grup halinde aynı toplama fonksiyonuna sahip olabilir. Diğerleri ise farklı fonksiyonlar kullanabilirler. Bu tamamen tasarımcının kendi öngörüsüne dayanarak verdiği karara bağlıdır.

Tablo-3.1. Toplama fonksiyonu örnekleri

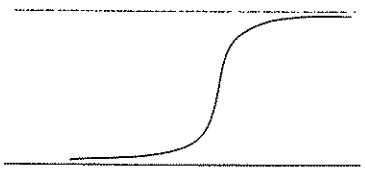
Net giriş	Açıklama
Çarpım Net Girdi= $\prod_i G_i A_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleri ile çarpılarak net girdi hesaplanır.
Maksimum Net Girdi= Max $(G_i A_i)$, i=1....N	N adet girdi içinden ağırlıklar ile çarpıldıktan sonra en büyüğü yapay sinir hücresinin net girdisi olarak kabul edilir.
Minimum Net Girdi= Min $(G_i A_i)$, i=1....N	N adet girdi içinden ağırlıklar ile çarpıldıktan sonra en küçüğü yapay sinir hücresinin net girdisi olarak kabul edilir.
Çoğunluk Net Girdi= $\sum_i$ sgn $(G_i A_i)$	N adet girdi içinden ağırlıklar ile çarpıldıktan sonra pozitif ve negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kumilatif toplam Net Girdi= Net(eski)+ $\sum_i (G_i A_i)$	Hücreye gelen bilgiler ağırlıklı olarak toplanır ve daha önce gelen bilgilere eklenerek hücrenin net girdisi bulunur.

❹ **Aktivasyon fonksiyonu**: Bu fonksiyon, hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Toplama fonksiyonunda olduğu gibi aktivasyon fonksiyonu olarak da çıktıyı hesaplamak içinde değişik formüller kullanılmaktadır. Bazı modeller (mesela çok katmanlı algılayıcı) bu fonksiyonun türevinin alınabilir bir fonksiyon olmasını şart koşmaktadır. Toplama fonksiyonunda olduğu gibi aktivasyon fonksiyonunda da ağın proses elemanlarının hepsinin aynı fonksiyonu kullanması gerekmez. Bazı elemanlar aynı fonksiyonu diğerleri farklı fonksiyonları kullanabilirler. Bir problem için en uygun fonksiyonda yine tasarımcının denemeleri sonucunda belirleyebileceği bir durumdur. Uygun fonksiyonu gösteren bir formül bulunmuş değildir.

Günümüzde en yaygın olarak kullanılan ***Çok Katmanlı Algılayıcı*** modelinde Genel olarak aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmaktadır. Bu fonksiyon şu formül ile gösterilmektedir.

$$F(NET) = \frac{1}{1 + e^{-NET}}$$

Burada NET proses elemanına gelen NET girdi değerini göstermektedir. Bu değer toplama fonksiyonu kullanılarak belirlenmektedir.



Şekil-3.7. *Sigmoid* fonksiyonunun şekilsel gösterimi

Sigmoid fonksiyonu şekilsel olarak da Şekil-3.7'te gösterilmiştir. Aktiviasyon fonksiyonu olarak kullanılacak olan diğer fonksiyonlara örnekler ise Tablo-3.2'de verilmiştir:

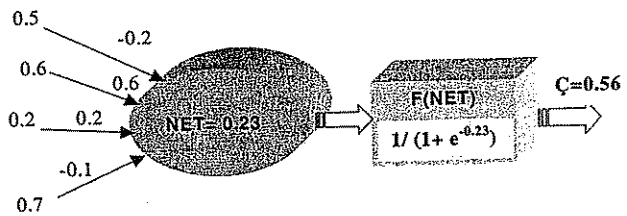
Tablo-3.2. Aktivasyon fonksiyonu örnekleri

Aktivasyon fonksiyonu	Açıklama
Lineer fonksiyon F(NET)= NET	Gelen girdiler olduğu gibi hücrenin çıktısı olarak kabul edilir.
Step fonksiyonu F(NET) = { 1 if NET > eşik_değer ; 0 if NET <= eşik_değer }	Gelen NET girdi değerinin belirlenen bir eşik değerinin altında veya üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerlerini alır.
Sinus fonksiyonu F(NET) = Sin (NET)	Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır.
Eşik değer fonksiyonu F(NET) = { 0 if NET<= 0 ; NET if 0<NET< 1 ; 1 if NET>=1 }	Gelen bilgilerini 0 veya 1'den büyük veya küçük olmasına göre bir değerler alır. 0 ve 1 arasında değerler alabilir. Bunların dışında değerler alamaz.
Hiperbolik tanjant fonksiyonu F(NET) = $(e^{NET} + e^{-NET})/(e^{NET} - e^{-NET})$	Gelen NET girdi değerinin tanjant fonksiyonundan geçirilmesi ile hesaplanır.

❺ **Hücrenin çıktısı**: Aktivasyon fonksiyonu tarafından belirlenen çıktı değeridir. Üretilen çıktı dış dünyaya veya başka bir hücreye gönderilir. Hücre kendi çıktısını kendisine girdi olarak da gönderebilir. Bir proses elemanının birden fazla çıktısı olmasına rağmen sadece bir çıktısı olmaktadır. Ağ şeklinde gösterildiğinde bir proses elemanının birden faza çıktısı varmış gibi görülmektedir. Bu sadece gösterim amacıyladır. Aslında bir proses elemanından çıkan tek bir çıktı değeri vardır. Aynı değer birden fazla proses elemanına girdi olarak gitmektedir.

3.3. Yapay Sinir Hücresinin Çalışma Prensibi

Bir yapay sinir hücresini (proses elemanını) nasıl çalıştığını daha kolay anlamak için bir örnek vermek yararlı olur. Bir proses elemanına gelen bilgiler ve ağırlıklar Şekil-3.8'te verildiği gibi varsayılsın. Görüldüğü gibi proses elemanının 4 girdisi ve 4 ağırlık değeri vardır.



Şekil-3.8. Bir yapay sinir hücresinin çalışması örneği

Hücreye gelen NET bilgi, ağırlıklı toplam alınarak şu şekilde hesaplanır.

NET = 0.5 * (-0.2) + 0.6 * 0.6 + 0.2 *0.2 + 0.7 * (-0.1)
NET = -0.1 + 0.36 + 0.04 – 0.07
NET= 0.23

Hücrenin *sigmoid* fonksiyonuna göre çıktısı (Ç) hesap edilir ise;

$Ç= 1/ (1+e^{-0.23})$
Ç= 0.56

Bir ağdaki bütün proses elemanlarının çıktılarının bu şekilde hesaplanması sonucu ağın girdilere karşılık çıktıları nasıl ürettiği görülür. Bu konuda ileride örnekler verilmektedir.

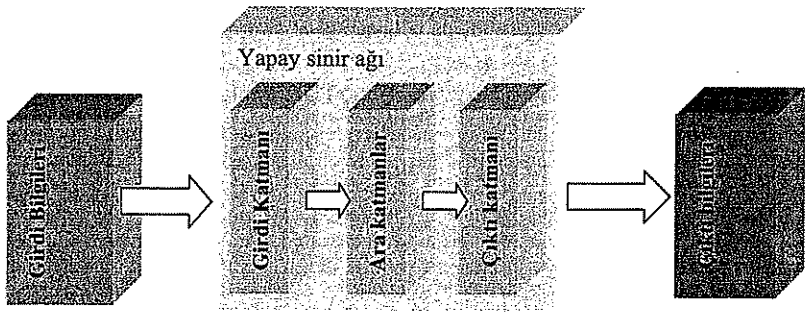
3.4. Yapay Sinir Ağının Yapısı

Daha önce belirtildiği gibi, yapay sinir hücreleri bir araya gelerek yapay sinir ağını oluştururlar. Sinir hücrelerinin bir araya gelmesi rasgele olmaz. Genel olarak hücreler 3 katman halinde ve her katman içinde paralel olarak bir araya gelerek ağı oluştururlar. Bu katmanlar:

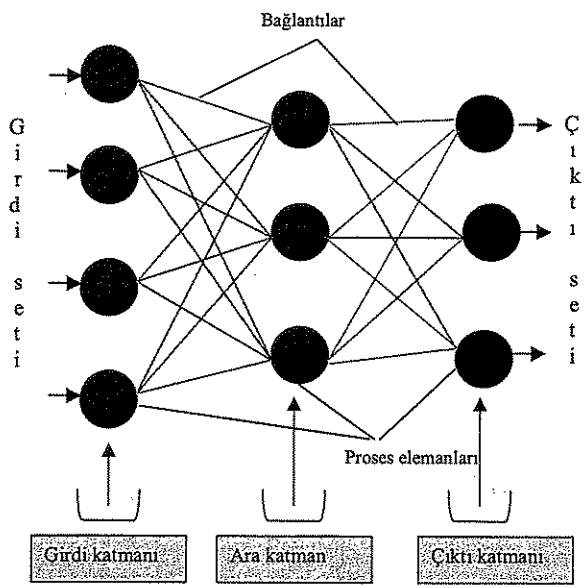
- **Girdi katmanı**: Bu katmandaki proses elemanları dış dünyadan bilgileri alarak ara katmanlara transfer etmekle sorumludurlar. Bazı ağlarda girdi katmanında herhangi bir bilgi işleme olmaz.
- **Ara katmanlar**: girdi katmanından gelen bilgiler işlenerek çıktı katmanına gönderirler. Bu bilgilerin işlenmesi ara katmanlarda gerçekleştirilir. Bir ağ için birden fazla ara katmanı olabilir.

- **Çıktı katmanı:** Bu katmandaki proses elemanları ara katmandan gelen bilgileri işleyerek ağın girdi katmanından sunulan girdi seti (örnek) için üretmesi gereken çıktıyı üretirler. Üretilen çıktı dış dünyaya gönderilir.

Şekil-3.9, bu üç katmanın bir biri ile ilişkisini göstermektedir:



Şekil-3.9. Yapay sinir ağı katmanlarının birbirleri ile ilişkileri

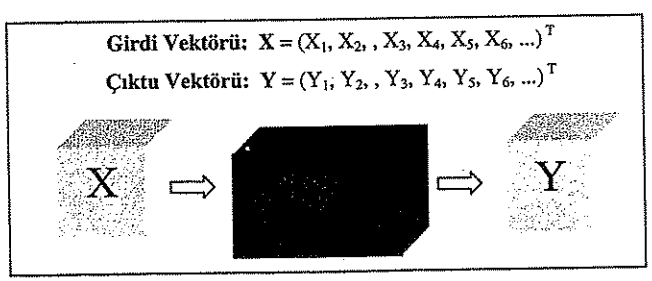


Şekil-3.10. Bir yapay sinir örneği

Bu üç katmanın her birinde bulunan proses elemanları ve katmanlar arası ilişkileri şematik olarak Şekil-3.10'da gösterilmektedir. Şekildeki yuvarlak şekiller proses elemanlarını göstermektedir. Her katmanda birbirine paralel elemanlar söz konusudur. Proses elemanlarını birbirlerine bağlayan çizgiler ise ağın bağlantılarını göstermektedir. Proses elemanları ve bağlantıları bir yapay sinir ağını oluştururlar. Bu bağlantıların ağırlık değerleri öğrenme sırasında belirlenmektedir. Öğrenmenin nasıl gerçekleştiği ve ağırlıkların nasıl belirlendiği ilerideki bölümlerde örnekler ile anlatılacaktır.

3.5. Yapay Sinir Ağlarının Çalışması (Kara Kutu Yakıştırması)

Şekil-3.11'de gösterildiği gibi yapay sinir ağlarının genel çalışma prensibi, bir girdi setini (örnekleri) alarak onları çıktı setine çevirmek olarak açıklanabilir. Bunun için ağın kendisine gösterilen girdiler için doğru çıktıları üretecek hale gelmesi (yani eğitilmesi) gerekmektedir. Ağa gösterilecek örnekler öncelikle bir vektör haline getirilir. Bu vektör ağa gösterilir ve ağ bu vektör için gerekli çıktı vektörünü üretir. Ağın parametre değerleri doğru çıktıyı üretecek şekilde düzenlenir. Girdi vektörü, haftanın günlerini gösteren sayısal (nümerik) değerler, bir resmin gri tonları, bir parmak izini gösteren nümerik değerler, borsada bir kağıdın haftalık veya günlük borsa değerleri, bir ürünün satış miktarı vb. gibi değişik olayları gösteren nümerik değerlerden oluşan vektörler olabilirler. Benzer şekilde çıktı vektörü de girdi vektörünün sınıfını gösterebilir. Bir değerin tahmin edilmesi olabilir. Bir resmin gri tonları olabilir. Girdi ve çıktı vektörlerinin tasarımı ağı geliştiren kişi tarafından belirlenir ve örnekler (girdiler) belirlenen formatta toplanarak eğitim esnasında ağa gösterilirler.



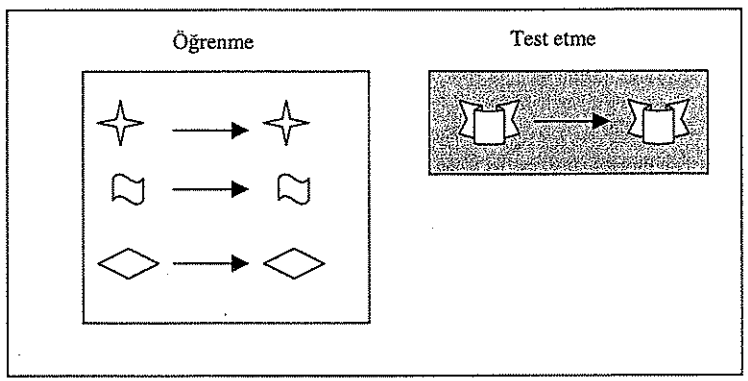
Şekil-3.11. Yapay Sinir Ağı (YSA), girdi, çıktı ilişkisi

Burada bir noktaya dikkatleri çekmekte yarar vardır. Bir yapay sinir ağı, herhangi bir girdi vektörünü çıktı vektörüne nasıl dönüştürdüğü konusunda bir bilgi vermez. Mühendislik açısından bakıldığında yapay sinir ağları "kara kutu" gibi görülebilirler. Kara kutu, dışarıdan bilgileri alıp, dışarıya ürettiği çıktıları vermektedir. İçeride neler olduğu ise bilinmemektedir. Diğer bir deyişle, yapay sinir ağının sonuçları nasıl oluşturduğunu açıklama yeteneği yoktur. Bu ağa olan güveni sarsmakla birlikte başarılı uygulamalar yapay sinir ağlarına olan ilgiyi sürekli artırmaktadır. Açıklama yeteneğinin kazandırılması bilim dünyasına çok önemli bir katkı oluşturabilecektir. Bu konu günümüzde önemli bir araştırma alanı olarak görülebilir.

3.6. Yapay Sinir Ağlarında Öğrenme, Adaptif Öğrenme ve Test Etme

Yukarıda belirtildiği gibi yapay sinir ağlarında proses elemanlarının bağlantılarının ağırlık değerlerinin belirlenmesi işlemine ***"ağın eğitilmesi"*** denir. Başlangıçta bu ağırlık değerleri rasgele olarak atanır. Yapay sinir ağları kendilerine örnekler gösterildikçe bu ağırlık değerlerini değiştirirler. Amaç ağa gösterilen örnekler için doğru çıktıları üretecek ağırlık değerlerini bulmaktır. Örnekler ağa defalarca gösterilerek en doğru ağırlık değerleri bulunmaya çalışılır. Ağın doğru ağırlık değerlerine ulaşması örneklerin temsil ettiği olay hakkında genellemeler yapabilme yeteneğine kavuşması demektir. Bu genelleştirme özelliğine kavuşması işlemine *ağın öğrenmesi* denir. Ağırlıkların değerlerinin değişmesi belirli kurallara göre yürütülmektedir. Bu kurallara *öğrenme kuralları* denir. Daha önce belirtildiği gibi, Kullanılan öğrenme stratejisine göre değişik öğrenme kuralları geliştirilmiştir. İlerideki bölümlerde değişik modelleri anlatırken her model için geliştirilmiş olan öğrenme kuralları ayrıntılı olarak anlatılacaktır.

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



Şekil-3.12. Öğrenme ve test etme

Ağın eğitimi tamamlandıktan sonra öğrenip öğrenmediğini (performansını) ölçmek için yapılan denemelere ise *ağın test edilmesi* denmektedir. Test etmek için ağın öğrenme sırasında görmediği örnekler kullanılır. Test etme sırasında ağın ağırlık değerleri değiştirilmez. Test örnekleri ağa gösterilir. Ağ eğitim sırasında belirlenen bağlantı ağırlıklarını kullanarak görmediği bu örnekleri için çıktılar üretir. Elde edilen çıktıların doğruluk değerleri ağın öğrenmesi hakkında bilgiler verir. Sonuçlar ne kadar

iyi olursa eğitimin performansı da o kadar iyi demektir. Eğitimde kullanılan örnek setine *eğitim seti*, test için kullanılan sete ise *test seti* adı verilmektedir. Yapay sinir ağlarının bu şekilde bilinen örneklerden belirli bilgileri çıkartarak bilinmeyen örnekler hakkında yorumlar yapabilme (genelleme yapabilme) yeteneğine *Adaptif öğrenme* denir. Şekil-3.12'de öğrenme ve test etme olayı şematik olarak gösterilmektedir.

3.7. Yapay Sinir Ağlarında Bilgi ve Zeka

Yapay sinir ağlarında bilgi, ağın bağlantılarının sahip olduğu ağırlık değerlerinde saklanır. Diğer bir deyişle, bir yapay sinir ağının zekası ağın bağlantılarının sahip olduğu ağırlık değerlerinde saklıdır. Ağın sahip olduğu ağırlık değerleri ne kadar doğru ise ağın performansı da o kadar yüksek olur. Ağırlık değerleri bütün ağa yayılmış olduğundan ağın belleği de dağıtık bir hafızadır. O nedenle tek bir ağırlık değeri bir anlam ifade etmemektedir. Bilginin dağıtılmış olması bazı ağırlık değerlerinin o veya bu şekilde kaybolması sonucunda dahi ağın çalışmasını sürdürmesine neden olmaktadır. Daha önce belirtildiği gibi bu ise yapay sinir ağlarının en önemli özelliklerinden ve güçlü yanlarından birisidir.

3.8. Yapay Sinir Ağlarından En Çok Kullanılan Modeller

Bir yapay sinir ağında proses elemanlarının bağlanması sonucu oluşan topoloji, proses elemanlarının sahip oldukları toplama ve aktivasyon fonksiyonları, öğrenme stratejisi ve kullanılan öğrenme kuralı ağın modelini belirlemektedir. Günümüzde çok sayıda model geliştirilmiştir. Bunların en yaygın olarak kullanılanları ve pratik hayatta uygulananları şunlardır.

- Algılayıcılar
- Çok katmanlı algılayıcılar (hatayı geriye yayma modelleri)
- Vektör Kuantizasyon modelleri (LVQ)
- Kendi kendini organize eden model (SOM)
- Adaptif Rezonans Teorisi modelleri (ART)
- *Hopfield* ağları
- *Counterpropagation* ağı
- *Neocognitron* ağı
- *Boltzman* makinesi
- *Probabilistic* ağlar (PNN)
- Elman ağı
- Radyal temelli ağlar (RBN)

İlerideki bölümlerde bu modellerin bazıları ayrıntılı olarak açıklanmaktadır.

3.9. Özet

Yapay sinir ağları biyolojik sinir sisteminden etkilenerek geliştirilmiştir. Biyolojik sinir hücreleri birbirleri ile *synapsler* vasıtası ile iletişim kurarlar. Bir sinir hücresi işlediği bilgileri *axon*'ları yolu ile diğer hücrelere gönderirler. Benzer şekilde yapay sinir hücreleri dışarıdan gelen bilgileri bir toplama fonksiyonu ile toplar ve aktivasyon fonksiyonundan geçirerek çıktıyı üretip ağın bağlantılarının üzerinden diğer hücrelere (proses elemanlarına) gönderir. Değişik toplama ve aktivasyon fonksiyonları vardır. Yapay sinir ağlarını birbirlerine bağlayan bağlantıların değerlerine ağırlık değerleri denmektedir. Proses elemanları birbirlerine paralel olarak 3 katman halinde bir araya gelerek bir ağ oluştururlar. Bunlar;

- Girdi katmanı
- Ara katmanlar
- Çıktı katmanı

Bilgiler ağa girdi katmanından iletilir. Ara katmanlarda işlenerek oradan çıktı katmanına gönderilirler. Bilgi işlemeden kasıt ağa gelen bilgilerin ağın ağırlık değerleri kullanılarak çıktıya dönüştürülmesidir. Ağın girdiler için doğru çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekmektedir. Doğru ağırlıkların bulunması işlemine ağın eğitilmesi denmektedir. Bu değerler başlangıçta rasgele atanırlar. Daha sonra eğitim sırasında her örnek ağa gösterildiğinde ağın öğrenme kuralına göre ağırlıklar değiştirilir. Daha sonra başka bir örnek ağa sunularak ağırlıklar yine değiştirilir ve en doğru değerleri bulunmaya çalışılır. Bu işlemler ağ eğitim setindeki örneklerin tamamı için doğru çıktılar üretinceye kadar tekrarlanır. Bu sağlandıktan sonra test setindeki örnekler ağa gösterilir. Eğer ağ test setindeki örneklere doğru cevaplar verirse ağ eğitilmiş kabul edilmektedir. Ağın ağırlıkları belirlendikten sonra her bir ağırlığın ne anlama geldiği bilinmemektedir. O nedenle yapay sinir ağlarına ***"kara kutu"*** yakıştırması yapılmaktadır. Ağırlıkların tek tek ne anlama geldikleri bilinmemekle birlikte ağın girdiler hakkındaki kararını bu ağırlıkları kullanarak vermesi, ağın zekasının bu ağırlıklarda saklandığı söylenebilir. Ağın bir olayı öğrenmesi o olay için en doğru yapay sinir ağı modelini seçmekle mümkündür. Şu ana kadar bir çok yapay sinir ağı modeli geliştirilmiştir. Bir yapay sinir ağının modelini şu bilgiler karakterize etmektedir.

- Ağın topolojisi
- Kullanılan toplama fonksiyonu
- Kullanılan aktivasyon fonksiyonu
- Öğrenme stratejisi
- Öğrenme kuralı

Geliştirilen modeller arasında en yaygın olarak kullanılanları, tek ve çok katmanlı algılayıcılar, LVQ, ART ağları, SOM, Elman ağı gibi ağlardır. Bu modeller ilerideki bölümlerde ayrıntılı olarak anlatılacaktır.

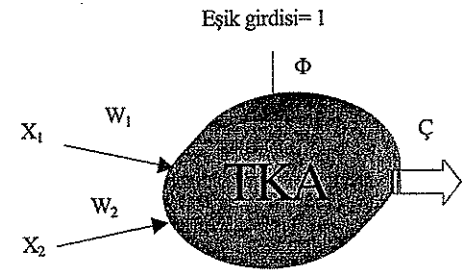
BÖLÜM 4

İLK YAPAY SİNİR AĞLARI

Yapay sinir ağlarına ve proses elemanlarının yapısına genel bir bakış yaptıktan sonra ilk yapay sinir ağı çalışmaları ve geliştirilen ilk modeller bu bölümde açıklanacaktır. Bu kapsamda tek katmanlı algılayıcılar, *Perseptron* ve ADALINE/MADALINE modellerinin yapısı ve öğrenme kuralları ayrıntılı olarak açıklanacaktır. Bu modeller daha sonra geliştirilen modellere temel oluşturduklarından önemlidir.

4.1. Tek Katmanlı Algılayıcılar (TKA)

Tek katmanlı yapay sinir ağları sadece girdi ve çıktı katmanlarından oluşur. Her ağın bir veya daha fazla girdisi ve çıktısı (Ç) vardır. Çıktı üniteleri bütün girdi ünitelerine (X) bağlanmaktadır. Her bağlantının bir ağırlığı vardır (W). En basit şekliyle tek katmanlı bir ağa örnek vermek gerekirse, Şekil-4.1'deki ağ iki girdisi ve bir çıktıdan oluşmaktadır. Bu ağlarda proses elemanlarının değerlerinin ve dolayısıyla ağın çıktısının sıfır olmasını önleyen birde eşik değeri (Φ) vardır. Eşik değerinin girdisi daima 1'dir.



Şekil-4.1. İki girdi ve bir çıktıdan oluşan en basit TKA model

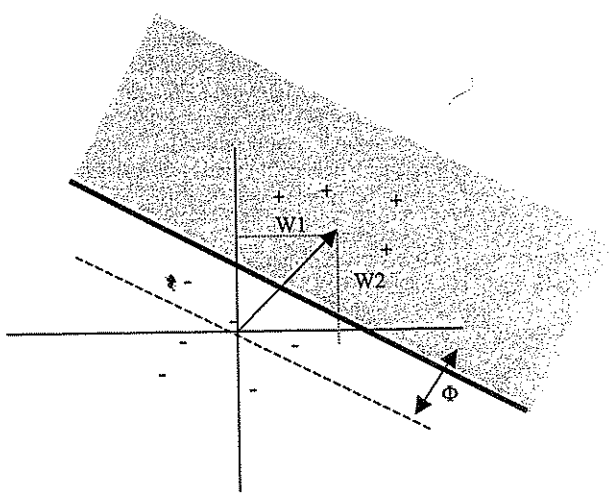
Ağın çıktısı ağırlıklandırılmış girdi değerlerinin eşik değeri ile toplanması sonucu bulunur. Bu girdi değeri bir aktivasyon fonksiyondan geçirilerek ağın çıktısı hesaplanır. Bu, şu şekilde formülize edilmektedir.

$$Ç = f\left(\sum_{i=1}^{m} w_i x_i + \Phi \right)$$

Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusal fonksiyondur. Yani ağa gösterilen örnekler iki sınıf arasında paylaştırılarak iki sınıfı birbirinden ayıran doğru bulunmaya çalışılır. Onun için eşik değer fonksiyonu kullanılmaktadır. Burada ağın çıktısı 1 veya –1 değerlerini almaktadır. 1 ve –1 sınıfları temsil etmektedir. Eğer ağın çıktısı 1 ise birinci sınıfta –1 ise ikinci sınıfta kabul edilmektedir (Bazı araştırıcılar sınıfları 1 veya 0 olarak da gösterilmektedir). Felsefede bir değişiklik yoktur.

$$f(g)= \begin{cases} 1 & \text{Eğer } Ç > 0 \text{ ise} \\ -1 & \text{aksi taktirde} \end{cases}$$

Bu formül incelendiğinde ağa gelen toplam girdinin pozitif olması durumunda ağa sunulan örnek 1 sınıfına negatif olması durumunda ise -1 sınıfına ait demektir. Sıfır olması durumu ise tasarımcının kabulüne kalmıştır. Yukarıdaki formülde -1 sınıfına konulmuştur. Dikkat edilirse, iki sınıfı ayıran bir doğrudur.



Şekil-4.2. Ağırlıkların ve sınıf ayracı olan doğrunun geometrik gösterimi

Sınıf ayracı da denilen bu doğru şu şekilde tanımlanmaktadır.

$W1 .X1 + W2 .X2 + \Phi = 0$

Buradan $X2 = -(W1 / W2) X1 - \Phi / W2$ olur.

Benzer şekilde, $X1 = -(W2 / W1) X2 - \Phi / W2$ olarak hesaplanır. Bu iki formülden hareketle sınıfın ayracı doğrusu çizilebilir. Bu doğrunun geometrik gösterimi ise Şekil-4.2'de verilmiştir.

Bu ağlarda öğrenmeden kasıt ağın sınıf ayracı doğrusunun pozisyonunu her iki grubu en iyi şekilde ayıracak şekilde belirlemektir. Bunun için ağırlık değerlerinin değiştirilmesi gerekmektedir. Yani t zaman biriminde ağırlık değerleri ΔW kadar değiştirilir ise;

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

olacaktır. Öğrenme sırasında bu değişim her iterasyonda gerçekleştirilerek sınıf ayracının en doğru pozisyonu bulunmaya çalışılır. Ağırlıkların değiştirilmesi doğrunun eğimini değiştirmek anlamına gelmektedir. Bu yeterli olmayabilir. Eşik değerinin de değiştirilmesi gerekir. Bu, doğrunun sınıflar arası kaymasına yardımcı olmaktadır. Böylece aktivasyon fonksiyonun konumu belirlenmektedir. Bu durum da t anında eşik değerinin de;

$$\Phi(t+1) = \Phi(t) + \Delta\Phi(t)$$

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

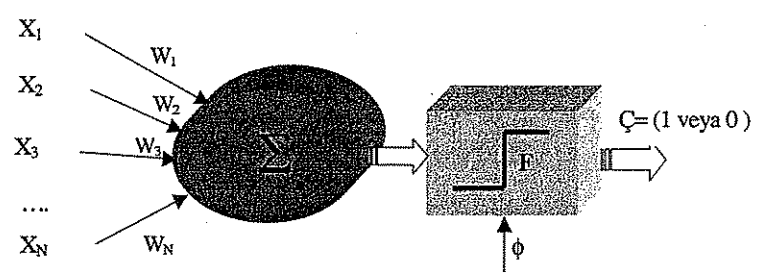
Tek katmanlı algılayıcılarda önemli görülen iki modelden bahsedilebilir. Bunlardan birisi algılayıcı (*perseptron*) modeli, diğeri ise Adaline/Madaline ünitesidir.

4.2. Basit Algılayıcı Modeli (*Perseptron*)

İlk defa 1958 yılında `Rosenblat` tarafından örüntü (şekil) sınıflandırma amacı ile geliştirilmiştir [1].

4.2.1. Basit Algılayıcıların Yapısı

Perseptron bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanmaktadır. Ağın çıktısı bir veya sıfırdan oluşan mantıksal (*boolean*) değerdir. Çıktının değerinin hesaplanmasında eşik değer fonksiyonu kullanılır. *Perseptron*'un yapısı Şekil-4.3'de gösterildiği gibidir:



Şekil-4.3. Bir basit algılayıcı yapısı

Şekilden de görüldüğü gibi, *perseptron* eğitilebilen tek bir yapay sinir hücresinden (proses elemanından) oluşur. Eğitilebilirden kasıt ağırlıkların (W) değiştirilebilir olması demektir. Girdiler proses elemanına gösterilirler. Her girdi setine karşılık gelen çıktı değerleri de ağa gösterilir. Daha sonra öğrenme kuralına göre ağın çıktı değeri hesaplanır. Eğer ağın çıktısı olması gereken çıktıdan farklı ise ağırlıklar ve eşik değerleri değiştirilir. Değişikliğin nasıl yapılacağını ise öğrenme kuralı belirler. Girdilere karşılık gelene çıktı değerleri bir veya sıfırdan oluşmaktadır.

Yukarıda belirtildiği gibi eşik değeri, aktivasyon fonksiyonunun konumunu belirlemek için kullanılır. Şekil-4.4 eşik değerinin sıfır ve "a" pozitif değerleri alması durumunda aktivasyon fonksiyonunun konumunu göstermektedir.

Şekil-4.4. Eşik değerinin aktivasyon fonksiyonunun konumuna etkisi

4.2.2. Basit Algılayıcı Öğrenme Kuralı

Basit algılayıcıların öğrenme kuralı adım adım aşağıda açıklanmıştır:

Adım 1: Ağa girdi setini ve ona karşılık olarak beklenen çıktı gösterilir **(X,B)**. Burada birden fazla girdi değeri olabilir. Yani $X= (x1, x2, x3...xN)$ demektir. Çıktı değeri ise 1 ve 0 değerlerinden birisini alır.

Adım 2: *Perseptron* ünitesine gelen net girdi şu şekilde hesaplanır:

$$NET = \sum_{i=1}^{m} w_i x_i$$

Adım 3: *Perseptron* ünitesinin çıktısı hesaplanır. Net girdinin eşik değerinden büyük veya küçük olmasına göre çıktı değeri 0 ve 1 değerlerinden birisini alır. Yani;

$$Ç = \begin{cases} 1 & \text{Eğer } NET > \phi \\ 0 & \text{Eğer } NET <= \phi \end{cases}$$

Eğer gerçekleşen çıktı ile beklenen çıktı aynı olursa ağırlıklarda herhangi bir değişiklik olmaz. Ağ, beklenmeyen bir çıktı üretmiş ise o zaman iki durum söz konusudur:

a) Ağın beklenen çıktısı 0 değeridir. Fakat NET girdi eşik değerinin üstündedir. Yani ağın gerçekleşen çıktısı 1 değeridir. Bu durumda ağırlık değerleri azaltılmaktadır. Ağırlıkların değişim oranı girdi değerlerinin belirli bir oranı kadardır. Yani vektörel olarak;

$$Wn = Wo - \lambda X$$

olur. Burada λ *öğrenme katsayısıdır*. Ağırlıkların değişim miktarlarını belirlemekte ve sabit bir değer olarak alınmaktadır.

b) Beklenen çıktının 1 olması ve ağın gerçek çıktısının 0 olması durumudur. Yani net girdi eşik değerinin altındadır. Bu durumda ağırlıkların değerinin artırılması gerekmektedir. Yani vektörel olarak

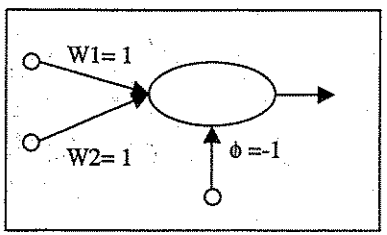
$$Wn = Wo + \lambda X$$

olacaktır.

Adım 4: Yukarıdaki adımları bütün girdi setindeki örnekler için doğru sınıflandırmalar yapılıncaya kadar ilk üç adımdaki işlemler tekrarlanır.

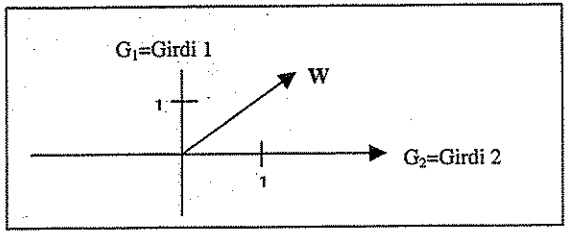
4.2.3. Basit Algılayıcı Öğrenmesine Bir Örnek

Yukarıda anlatılan öğrenme kuralının çalışmasını bir örnek ile açıklamak yararlı olur. Bu aynı zamanda, daha sonra anlatılacak olan yapay sinir ağlarında kullanılan öğrenme kurallarının anlaşılmasına da yardımcı olacaktır. Örnek anlaşılabilmesi için mümkün olduğu kadar basitleştirilmiş ve tek elemandan oluşan bir algılayıcı seçilmiştir. O nedenle, örnek, 2 girdiden oluşan bir ağ olup ağırlık değerleri Şekil-4.5'de verildiği gibi seçilmiştir.



Şekil-4.5. İki girdisi olan bir *perceptron* örneği

Bu örnekte karar uzayı Şekil-4.6'da gösterildiği gibi olur:



Şekil-4.6. Girilen örnek için karar uzayı

Bu karar uzayını iki ayrı sınıfa en uygun bölmek istenmektedir. Bunun için,

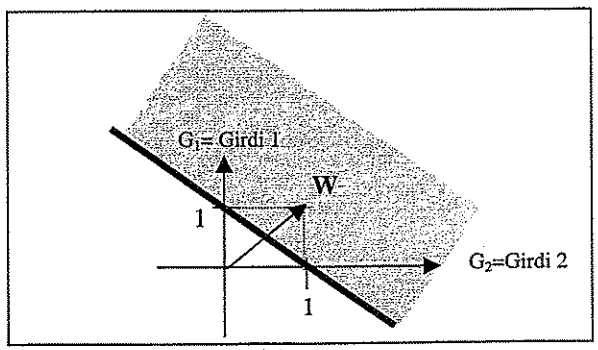
$W1 * G1 + W2 * G2 - 1 = 0$

eşitliğini sağlayan değerlerin bulunması gerekir. Öncelikle ağa gelen girdi değerlerinden her birisi sıfıra eşitlenerek diğeri bulunur. Yani,

$G1 = 0$ olursa o zaman $G2 = 1/W2 = 1/1 = 1$ olur
$G2 = 0$ olursa o zaman $G1 = 1/W1 = 1/1 = 1$ olur.

Şekil üzerinde bu iki değer birleştirilirse karar uzayını bölen doğru (sınıf ayraç çizgisi) bulunur (bkz. Şekil-4.7). Bu doğrunun bir tarafı çıktının 1 olduğu durumu diğer tarafı ise çıktının 0 olduğu durumu göstermektedir. Hangi tarafın 1 veya 0 tarafından gösterildiğini bulmak kolaydır. Bunun için bir tarafta kalan noktalar formüle konularak sonuca bakılır. Eğer ağın sonucu eşik değerinden büyük ise verilerin alındığı taraf 1'in gösterdiği sınıfı temsil ediyor demektir. Aksi durumda ise 0'ın temsil ettiği sınıfın tarafı belirlenmiş olacaktır. Örneğin $G1=0$ ve $G2=2$ noktası taralı olmayan alanda bir değerdir Bunları formülde bunları yerine koyarsak; 0.1+2.0-1 =1 değeri elde edilir.

Bu değer *perseptron* öğrenim kuralına göre eşik değerinden (ki burada –1 değeridir) büyük olduğundan ağın çıktısı 1 olacak demektir. Buda, taralı kısmın çıktının 1 olması durumunu temsil ettiğini göstermektedir.



Şekil-4.7. Girilen örnek için sınıf ayraç çizgisi

Girdi sayısının çoğalması durumunda bu doğru bir düzleme dönüşmektedir. Bunun bulunması ise yukarıdaki gibi kolay olmamakta ve *perseptron* öğrenme algoritmasını kullanmak gerekmektedir. Bu algoritma, doğrunun yerini belirlemeye yaramaktadır. Aşağıda basit bir örnekte öğrenme kuralının (algoritmasının) nasıl çalıştığı anlatılmıştır.

Bu örnekte de iki girdi ve bir çıktıdan oluşan bir basit algılayıcı kullanılmıştır. Ağın öğrenmesi gerek 2 tane örnek vardır. Belirlenen örneklerin girdileri (X), ağırlıkları (W), beklenen çıktı değerleri (B) ve eşik ünitesinin ağırlık değerinin (ϕ) aşağıdaki gibi verildiği varsayılsın,

1. örnek: $X1 = (x1, x2) = (1,0)$, $B1 = 1$

2. örnek: $X2 = (x1, x2) = (0,1)$, $B2 = 0$

Ağırlıklar: $W = (w1, w2) = (1,2)$

Eşik değeri: $\phi = -1$

Öğrenme katsayısı: $\lambda = 0.5$

1.iterasyon da 1. örnek ağa gösterilir. Bu durumda,

Basit algılayıcını NET girdisi;

$NET = w1*x1 + w2*x2 = 1*1+2*0 = 1$

NET > ϕ olduğundan Gerçekleşen çıktı, $Ç = 1$ olacaktır. $Ç = B1$ olduğundan ağırlıklar değiştirilmez. Böylece öğrenmede birinci iterasyon tamamlanmış olur.

İkinci örnek ağa gösterilerek aynı işlemler tekrarlanır. Bu ikinci iterasyon demektir. Burada birinci iterasyonda değiştirilen (bu örnekte olmamıştır) ağırlık ve eşik değerleri kullanılır.

2. iterasyon- 2. örnek ağa gösterilir.

Bu durumda algılayıcının NET girdisi,

$NET = w1*x1 + w2*x2 = 1*0+2*1 = 2$

NET > ϕ olduğundan Gerçekleşen çıktı, $Ç = 1$ olacaktır. $Ç \neq B2$ olduğundan ağırlıklar,

$Wn = Wo - \lambda X$

formülü kullanılarak değiştirilir. Ağırlıkların değerleri belirlenir. Sonuçta yeni değerler şöyle elde edilir.

$w1 = w1 - \lambda x1$
$w1 = 1 - 0.5 * 0 = 1$
$w2 = w2 - \lambda x2$
$w2 = 2 - 0.5 * 1 = 1.5$

3. iterasyon 1. örnek tekrar gösterilirse

Benzer şekilde;

NET= $w1*x1 + w2*x2 = 1*1+1.5*0= 1$

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. $Ç1=B1$ olduğundan ağırlıklar değiştirilmez

4. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0+1.5*1= 1.5$ olarak hesaplanır.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

$Wn = Wo - \lambda X$

formülü kullanılarak değiştirilir ve şu sonuçlar elde edilir. Burada bir önceki iterasyonlarda değiştirilen ağırlıkların kullanıldığına dikkat ediniz.

$w1= w1 - \lambda x1$
$w1= 1- 0.5 * 0 = 1$
$w2= w2 - \lambda x2$
$w2= 1.5- 0.5 * 1 = 1$

5. iterasyon 1. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*1+1*0= 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. $Ç1=B1$ olduğundan ağırlıklar değiştirilmez

6. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0+1*1= 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

$Wn = Wo - \lambda X$

formülü ile yeni ağırlıklar şöyle bulunur.

$w1= w1 - \lambda x1$ $\quad w1= 1- 0.5 * 0 = 1$
$w2= w2 - \lambda x2$ $\quad w2= 1- 0.5 * 1 = 0.5$

7. iterasyon 1. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*1+0.5*0= 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. $Ç1=B1$ olduğundan ağırlıklar değiştirilmez.

8. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0+0.5*1= 0.5$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

$Wn = Wo - \lambda X$

formülü ağırlıklar şu değerleri alır.

$w1= w1 - \lambda x1$
$w1= 1- 0.5 * 0 = 1$
$w2= w2 - \lambda x2$
$w2= 0.5- 0.5 * 1 = 0$

9. iterasyon 1. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*1+0*0= 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. $Ç1=B1$ olduğundan ağırlıklar değiştirilmez

10. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0+0.0*1= 0$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

$Wn = Wo - \lambda X$

formülü kullanılarak değiştirilir ve sonuçta şu değerler oluşur.

$w1= w1 - \lambda x1$
$w1= 1- 0.5 * 0 = 1$
$w2= w2 - \lambda x2$
$w2= 0- 0.5 * 1 = -0.5$

11. iterasyon 1. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*1+ (-0.5)*0= 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. $Ç1=B1$ olduğundan ağırlıklar değiştirilmez

12. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0+(-0.5)*1= -0.5$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1= 1$ olacaktır. Beklenen çıktıdan farklı olduğundan benzer şekilde ağırlıklar;

$Wn = Wo - \lambda X$

formülü kullanılarak şu şekilde değiştirilir.

$w1 = w1 - \lambda x1$
$w1 = 1 - 0.5 * 0 = 1$
$w2 = w2 - \lambda x2$
$w2 = -0.5 - 0.5 * 1 = -1$

13. iterasyon 1. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*1 + (-1)*0 = 1$ olur.

NET > ϕ olduğundan Gerçekleşen çıktı $Ç1 = 1$ olacaktır. $Ç1 = B1$ olduğundan ağırlıklar değiştirilmez

14. iterasyon 2. örnek tekrar gösterilirse

NET= $w1*x1 + w2*x2 = 1*0 + (-1)*1 = -1$ olur.

NET = ϕ olduğundan Gerçekleşen çıktı $Ç1 = 0$ olacaktır. $Ç1 = B1$ olduğundan ağırlıklar değiştirilmez.

Bundan sonra her iki örnekte doğru olarak sınıflandırılır. Öğrenme sonunda ağırlıklar:

$w1 = 1$
$w2 = -1$

değerlerini alınca örnekler doğru sınıflandırılabilir demektir. Bu ağırlık değerleri kullanılarak ile iki örnek tekrar ağa tekrar gösterilirse, ağın çıktıları şöyle olur:

1. örnek için: NET= $w1*x1 + w2*x2 = 1*1 + (-1)*0 = 1 > \phi$ $Ç1 = 1 = B1$
2. örnek için: NET= $w1*x1 + w2*x2 = 1*0 + (-1)*1 = -1 = \phi$ $Ç2 = 0 = B2$

Görüldüğü gibi her iki örnek içinde ağ tarafından doğru sınıflandırma yapılmaktadır. O nedenle, ağ öğrenmeyi tamamlamıştır denilebilir.

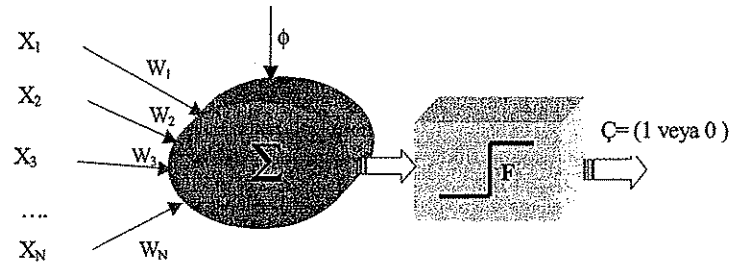
Tek katmanlı algılayıcının en önemli problemi doğrusal olmayan problemlerin çözülmesinde başarılı olmamasıdır. Daha önce belirtildiği gibi, bilinen XOR problemine çözüm üretemediği için uzun süre yapay sinir ağları araştırmalarının sekteye uğramasına neden olmuştur. Daha sonra çok katmanlı algılayıcıların (*backpropogation* metodunun) bulunması ile bu sorun çözülmüştür. İleride bu konuda daha ayrıntılı bir örnek irdelenecektir.

Tek katmanlı algılayıcının diğer bir problemi ise ağın her iterasyonda ağırlıklarını değiştirdikçe öğrendiklerini unutması olasılığıdır. Bir girdi seti ağırlıkları artırdıkça diğeri azaltmaktadır. Bu sorun çoğu zaman içerisinde ağırlıkların bulunması ile çözülebilmektedir. Fakat eğitim zamanının uzamasına neden olmaktadır.

4.3. ADALINE/MADALINE Modeli

ADALINE `Widrow ve Hoff` [2] tarafından 1959 yılında geliştirilmiş olup adaptif doğrusal eleman (*Adaptif Linear Element*) ağının kısaltılmış şeklidir. Genel olarak ADALINE bir proses elemanından (*Adaline unitesi*) oluşan bir ağdır.

Bu ağ en küçük ortalamaların karesi (*least mean square*) yöntemine dayanmaktadır. Öğrenme kuralına delta kuralı da denmektedir. Öğrenme kuralı, ağın çıktısının beklenen çıktı değerine göre hatasını enazlayacak şekilde ağın ağırlıklarının değiştirilmesi prensibine dayanır. ADALINE ünitesinin yapısı Şekil-4.8'de verilmiştir. ADALINE'nin yapısının tek katmanlı algılayıcıya benzediğine dikkat ediniz. Aradaki fark öğrenme kuralında görülmektedir.



Şekil-4.8. Adaline unitesi

Şekilden de görüldüğü gibi bir ADALINE ünitesi şu notasyon ile gösterilmektedir:

N adet girdiler: $X1, X2, X3, \ldots\ldots XN$

Her girdinin ADALINE elemanı üzerinde etkisinin gösteren ağırlıklar: $W1, W2, W3, \ldots WN$,

ADALINE biriminin çıktısının sıfırdan farklı bir değer olmasını sağlayan eşik değer: Ø

4.3.1. ADALINE Ünitesinin Öğrenme Kuralı

ADALINE ünitesi en küçük kareler yöntemini kullanarak öğrenme gerçekleştirir. *Perseptron* algoritmasına çok benzemektedir. Bu algoritmanın performansı `Zeidler` [3] tarafından incelenmiştir. Öğrenme kuralı yapay sinir ağlarında genel öğrenme prensibine göre çalışmaktadır. Girdilerden çıktılar hesaplanır ve ağırlıklar çıktıya göre değiştirilir. Öğrenme şu şekildedir.

ADALINE ünitesinin net girdisi NET ve çıktısı ($\hat{C}$) şu şekilde hesaplanmaktadır.

$$NET = \sum_{i=1}^{m} w_i x_i + \Phi \quad \text{yani,}$$

NET= $\emptyset + X_1 W_1 + X_2 W_2 + X_3 W_3 + \ldots\ldots + X_N W_N$
Çıktı ($Ç$)= 1 Eğer NET >=0 ise
Çıktı ($Ç$)= -1 Eğer NET < 0 ise

Ağın çıktısını üreten fonksiyon bilinen step fonksiyonudur. Beklenen değerin *B* olduğu varsayılırsa; Adaline ünitesinin çıktısını ürettikten sonraki hatası;

$$E = B\text{-}Ç$$

olacaktır. Amaç bu hatayı en aza indirecek ağırlıkları bulmaktır. Bunun için her seferinde ağa farklı örnekler gösterilerek hatalar hesaplanmakta ve ağırlıklar hatayı azaltacak şekilde değiştirilmektedir. Zaman için de hata, olması gereken en küçük değere düşmektedir. Bu hatayı azaltmak için kullanılan kural şu formül ile gösterilmektedir.

$$W_y = W_e + \alpha(B\text{-}Ç)\,X$$

Diğer bir deyişle her hangi bir *t* anında,

$$W_i\,(t) = W_i\,(t\text{-}1) + \alpha * E * X_i$$

olacaktır. Bu formülde *W*(t) ağırlıkların *t* zamanındaki yeni değerlerini, *W* (*t*-1) ağırlıkların değişmeden önceki (*t*-1. zamandaki) değerlerini, α öğrenme katsayısını, *B* beklenen çıktıyı, *E* beklenen değer ile çıktı arasındaki hatayı ve *X*'de girdileri göstermektedir.

Benzer şekilde eşik değeri de yine zaman içerisinde değiştirilerek olması gereken eşik değeri bulunur. Buda şu formüle göre yapılır.

$$\phi_y = \phi_e + \alpha\,(B\text{-}Ç)$$

Burada, ϕ_y yeni eşik değerini, ϕ_e değiştirilmeden önceki eşik değerini göstermektedir.

4.3.2. ADALINE Ünitesinin Öğrenmesine Bir Örnek

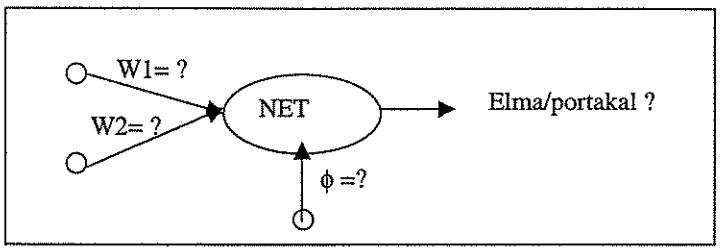
ADALINE ünitesinin çalışma prensibini göstermek için şu örnek verilebilir.

Bir meyve üreticisi firmanın elma ve portakalların ambara geldiklerinde karışmasının önlemek için bir makine yapmak istediğini varsayınız. Bu amaçla bir yapay sinir ağının kurulabilmesi nasıl mümkün olacaktır?

Meyveleri gösteren ve birbirinden farklılıklarını ortaya koyan örnekler oluşturmak yapılacak ilk iştir. Bunun için meyveleri ve onun özelliklerini gösteren vektörleri belirlemek gerekmektedir. Meyvelerin şeklini, görüntüsünü ve ağırlığını temsil etmek üzere 3 boyutlu bir vektör oluşturulabilir. Elma ve portakalı gösteren prototiplerin şu vektörler ile gösterildiği varsayılırsa örnek setinde iki örnek olacaktır.

Örnek 1: Portakal $X1 = (1,0)$; Bu örneğin beklenen çıktısı $Ç1 = -1$
Örnek 2: Elma $X2 = (0.1)$; Bu örneğin beklenen çıktısı $Ç2 = 1$

Bu problemi çözebilmek için 2 girdisi olan bir Adaline ünitesi tasarlanacaktır. Öğrenmenin amacı problem girdilerini doğru sınıflandıracak ağırlık değerleri ve eşik değerini bulmaktır (bkz. Şekil-4.9).



Şekil-4.9. İki girdili bir adaline ünitesi

Problemin çözümü için ağırlık değerleri (*W*1, *W*2 ve eşik değerinin değerleri başlangıçta rasgele atanmaktadır). Bunun aşağıdaki gibi olduğu varsayılsın.

*W*1=0.3
*W*2= 0.2
α=0.5
ϕ =0.1

Birinci iterasyon

Bu iterasyonda öncelikle birinci girdi vektörünün ağa gösterilmesi sonucu ağın çıktısı hesaplanır.

$$NET = \sum_{i=1}^{m} w_i x_i + \Phi$$

$$NET = [\,0.3,\ 0.2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.1$$

NET=0.3+0+0.1= 0.4 >0 → Ç= 1

NET değeri sıfırdan büyük bir değer olduğundan ağın çıktı değeri 1 kabul edilir. *B*=-1 olduğundan ağın ağırlıklarının değiştirilmesi gerekmektedir. Beklenen ve gerçekleşen çıktılar arasındaki hata *E* ile gösterilirse, *E*= *B*-Ç= -1-1=-2 olur.

Ağın ağırlıkları da yukarıda verilen formüle göre değiştirilecek olursa yeni değerleri şöyle olur.

$W_y = W_e + \alpha(B\text{-}Ç)\,X$
Wy= [0.3, 0.2] + 0.5*(-2) [1, 0]
Wy= [0.3, 0.2]+ (-1) [1,0]
Wy= [0.3-1, 0.2-0] =[-0.7, 0.2]

Eşik değer ünitesinin ağırlığı da benzer şekilde değiştirilir.

$\phi_y = \phi_e + \alpha\,(B\text{-}Ç)$
$\phi_y = 0.1 + 0.5^* (-2) = -0.9$

Böylece öğrenmede birinci iterasyon tamamlanmış olur.

İkinci iterasyon

İkinci iterasyonda benzeri işlemler ikinci örnek için yapılır. Ağırlıkların ve eşik değerinin birinci iterasyonda değiştirilen değerleri kullanılarak NET girdi ve çıktı değeri benzer şekilde hesaplanır.

$$NET = [-0.7, 0.2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 0.9$$

$NET = -0.7^*0 + 0.2^*1 - 0.9 = -0.9 < 0 \Rightarrow Ç = -1$

Bu örnek için B= 1 olması gerektiğinden ortaya çıkan hata; $E = B\text{-}Ç = 1-(-1) = 2$ olur. Yeni ağırlık değerleri ise,

$Wy = [-0.7, 0.2] + 0.5^*2\,[0, 1]$
$Wy = [-0.7, 0.2] + [0, 1]$
$Wy = [-0.7, 1.2]$
$\phi y = -0.9 + 0.5^* (2) = 0.1$

şeklinde hesaplanır.

Üçüncü iterasyon:

$$NET = [-0.7, 1.2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.1$$

$NET = -0.7 + 0 + 0.1 = -0.6 < 0 \Rightarrow Ç = -1$

Bu örnek için B= -1 olması gerektiğinden ağın sınıflandırması doğrudur. Bu ağırlıklarda bir değişiklik yapılmasını gerektirmez. Çünkü $B\text{-}Ç = 0$ olacak ve formülde herhangi bir değişiklik olmayacaktır.

Dördüncü iterasyon:

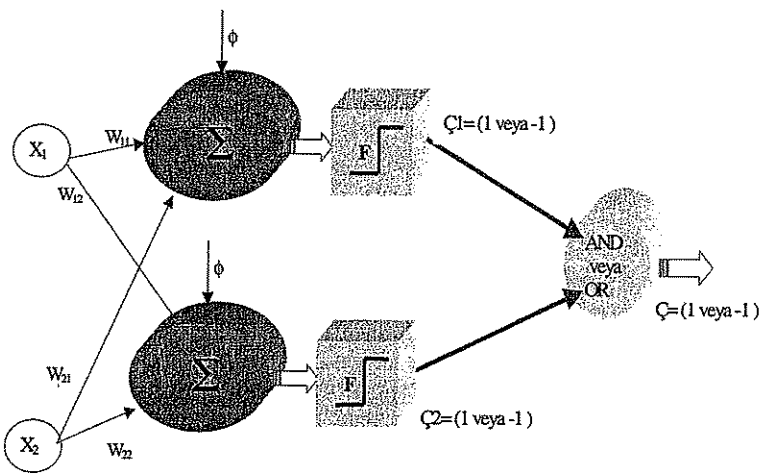
$$NET = [-0.7, 1.2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 0.1$$

$NET = 0 + 1.2 + 0.1 = 1.3 > 0 \Rightarrow Ç = 1$

Bu örnek için $B = 1$ olması gerektiğinden ağın sınıflandırması doğrudur. İki örneği de doğru sınıflandırdığına göre öğrenme tamamlanmıştır. Ağırlıkların ve eşik değerinin aşağıdaki gibi olması sonucu bu ağ meyveleri sınıflandırıcı olarak (bu örnek için) kullanılabilir.

4.4. MADALİNE

MADALINE ağları birden fazla ADALINE ünitesinin bir araya gelerek oluşturdukları ağa verilen isimdir. MADALINE ile ilgili ayrıntılı açıklamalar `Widrow` ve `Lehr` [4] tarafından verilmiştir. MADALINE ağları genel olarak iki katmandan oluşmaktadır. Her katmanda değişik sayıda adaline ünitesi bulunmaktadır. Ağın çıktısı da yine 1 ve –1 değerleri ile gösterilmektedir. Her biri bir sınıfı temsil etmektedir. Şekil-4.10'da iki adaline ünitesinden oluşan bir MADALINE gösterilmiştir.



Şekil-4.10. İki ADALINE ağından oluşan MADALINE ağı

MADALINE'nin öğrenme kuralı ADALINE üniteleri için kullanılan öğrenme kuralı ile aynıdır. Burada en sonda bulunan AND veya OR sonlandırıcısı önemlidir. Klasik mantık teorisinde olduğu gibi AND sonlandırıcısı olması durumunda bütün ADALINE ünitelerinin 1 değerini üretmesi sonucu MADALINE ağının çıktısı 1 olur. Aksi halde –1 (veya 0) değerini alır. OR sonlandırıcısı olması durumunda ADALINE ünitelerinin birisinin 1 üretmesi MADALINE ağının çıktısının 1 olması için yeterlidir.

4.5. Özet

Yapay sinir ağları ile ilgili çalışmaları tek katmanlı algılayıcılar ile başlamıştır. Bu algılayıcıların en önemli özelliği problem uzayını bir doğru veya bir düzlem ile sınıflara ayırmalarıdır. Problemin girdileri ağırlıklar ile çarpılıp toplandıktan sonra elde

edilen değerin bir eşik değerinden büyük veya küçük olmasına göre girdinin sınıfı belirlenir. Sınıflar 1 veya –1 rakamları (bazen 1 ve 0 rakamları) ile gösterilir. Öğrenme sırasında hem ağın ağırlıkları hem de eşik değer ünitesinin ağırlık değeri değiştirilir. Eşik değer ünitesinin çıktısı sabit olup 1'dir. Bilinen en nemli tek katmanlı algılayıcılar şunlardır.

- Basit tek katmanlı algılayıcılar (*perseptron*)
- ADALINE/MADALINE üniteleri

Bu modelleri birbirinden ayıran tek şey öğrenme kuralıdır.

Basit tek katmanlı algılayıcılarda ağırlıklar değiştirilir iken girdilerin öğrenme katsayısı (λ) denilen bir sabit ile çarpılıp ağırlıklara eklenmesi veya çıkartılması ile gerçekleştirilir. Ağa sunulan girdilere dayanarak üretilen çıktının değerine göre ağırlıklar artırılır veya azaltılır.

ADALINE ünitesinde ise ağırlıkların değiştirilmesi beklenen çıktı ile gerçekleşen çıktı arasındaki farka dayanarak gerçekleştirilir. Aradaki bu farka hata denirse; yeni ağırlık değerleri bu hatanın bir öğrenme katsayısı (α) ile girdilerin çarpılmasın sonucu elde edilen değerin eski ağırlıklara eklenmesi ile belirlenir.

ADALINE üniteleri bir araya gelerek MADALINE ağının oluştururlar. MADALINE ağının öğrenme kuralı ADALINE ünitesi ile aynıdır. MADALINE ağında ADALINE üniteleri birbirlerine AND veya OR operatörleri kullanılarak bağlanırlar. Bu operatörler klasik mantıkta kullanıldıkları şekilde kullanılırlar. Her ADALINE çıktısı bu operatörler yolu ile MADALINE ağının çıktılarına dönüştürülürler.

Tek katmanlı algılayıcıların en önemli problemi doğrusal olmayan olayları öğrenememeleridir. O nedenle bunlar geliştirilmiş ve yeni modeller oluşturulmuştur. İlerideki bölümlerde bunlara örnekler verilecektir.

4.6. Kaynakça

[1] Rosenblatt, F. (1958),``The perceptron: A probabilistic model for information storage and organization in the brain", Psychoanalytic Review, 65, , pp. 386-408.

[2]Widrow, B. and Hoff, M. E. (1960), ``Adaptive switching circuits", WESTCON Convention, Record Part IV, pp. 96-104.

[3]Zeidler, J. R.(1990), ``Performance analysis of LMS adaption filters", Proceedings of IEEE, Vol. 78, No. 12, pp. 1781-1806.

[4]Widrow B. Ve Lehr M.A. (1990), "30 years of adaptif neural networks: Perceptron, Madaline and Backpropogation", proceedings of the IEEE, Vol 78, No 9.

BÖLÜM 5

YAPAY SİNİR AĞI MODELİ (ÖĞRETMENLİ ÖĞRENME) ÇOK KATMANLI ALGILAYICI

Bir önceki bölümde anlatılan yapay sinir ağlarının ilk modellerinin en temel özellikleri doğrusal olan olayları çözebilme yeteneklerine sahip olmalarıdır. Bu ağlar ile doğrusal olmayan ilişkiler öğrenilememektedir. Bu sorunu çözmek için çok katmanlı algılayıcılar geliştirilmiştir. Bu bölümde çok katmanlı algılayıcılar detaylı olarak anlatılacaktır.

5.1. Çok Katmanlı Algılayıcı (ÇKA)

Bir yapay sinir ağının öğrenmesi istenen olayların girdi ve çıktıları arasındaki ilişkiler doğrusal olmayan ilişkiler olursa o zaman daha önce anlatılan modeller ile öğrenme gerçekleştirmek mümkün değildir. Bu tür olayların öğrenilmesi için daha gelişmiş modellere ihtiyaç vardır. Bu bölümde anlatılan Çok Katmanlı algılayıcı modeli bunlardan birisidir. Olayın doğrusal olup olmaması ne demektir? Bu konuyu iyi anlayabilmek için ünlü XOR problemine bakmak gerekir. Bu problemin özelliği doğrusal olmayan bir ilişkiyi göstermesidir. Yani çıktıların arasına bir doğru veya doğrular çizerek onları iki veya daha fazla sınıfa ayırmak mümkün değildir. Bu problem Tablo-5.1'de gösterildiği gibidir. Basit algılayıcı ve ADALINE ile bu problemi çözmek mümkün olmamıştır.

Tablo-5.1. XOR problemi

Girdi 1	Girdi 2	Çıktı
0	0	0
0	1	1
1	0	1
1	1	0

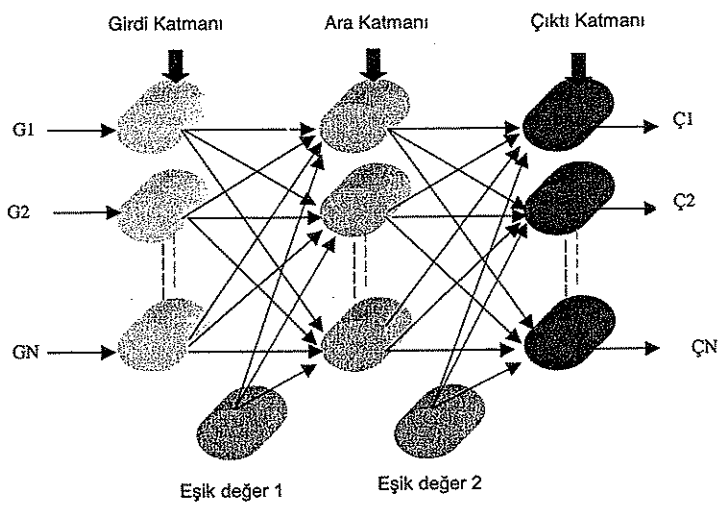
Bu problem, hemen hemen her yapay sinir ağının anlatan her kitapta örnek olarak verilmektedir. Bundan çok yaygın olarak bahsedilmesinin nedeni şöyle açıklanabilir. Minsky [1] özellikle basit algılayıcı (*perseptron*) modelinin bu probleme çözüm üretemediğini göstermiş ve yapay sinir ağlarının doğrusal olmayan problemlere çözüm

üretemediğini iddia ederek bilimsel araştırmaların durmasına neden olmuştur. Çünkü günlük olayların çoğu (hemen hemen hepsi) doğrusal olmayan bir nitelik taşımaktadır. XOR probleminin çözülememesinden sonra neredeyse bütün çalışmalar durmuş sadece birkaç araştırmacı çalışmalara devam etmiştir. Bu problemi çözerek yapay sinir ağlarına tekrar dikkatleri çekmeyi başarmışlardır. O nedenle bu problem yapay sinir ağı araştırmalarında bir kilometre taşı olarak görülmektedir.

XOR problemini çözmek amacı ile yapılan çalışmalar sonucu *çok katmanlı algılayıcı modeli (ÇKA)* geliştirilmiştir. `Rumelhart` ve `arkadaşları` [2] tarafından geliştirilen bu modele *hata yayma modeli veya geriye yayım modeli (backpropogation network)* de denmektedir. ÇKA modeli yapay sinir ağlarına olan ilgiyi çok hızlı bir şekilde artırmış ve yapay sinir ağları tarihinde yeni bir dönemin başlamasına neden olmuştur. Bu model günümüzde mühendislik problemlerinin hemen hemen hepsine çözümler üretebilecek bir güce sahiptir. Özellikle sınıflandırma, tanıma, ve genelleme yapmayı gerektiren problemler için çok önemli bir çözüm aracıdır. Bu model *Delta Öğrenme Kuralı* denilen bir öğrenme yöntemini kullanmaktadır. Bu kural aslında ADALINE ve basit algılayıcı modellerinin öğrenme kurallarının geliştirilmiş bir şeklidir. Temel amacı ağın beklenen çıktısı ile ürettiği çıktı arasındaki hatayı en aza indirmektir. Bunu hatayı ağa yayarak gerçekleştirdiği için bu ağa hata yayma ağıda denmektedir.

5.2. ÇKA Modelinin Yapısı

ÇKA ağlarının yapısı Şekil-5.1'de gösterildiği gibidir. Şekilden de görüldüğü gibi ÇKA ileriye doğru bağlantılı ve 3 katmanda oluşan bir ağdır. Bunlar:



Şekil-5.1. ÇKA modeli

- ***Girdi katmanı***: Dış dünyadan gelen girdileri (*G1*, *G2*, ... *GN*) alarak ara katmana gönderir. Bu katmanda bilgi işleme olmaz. Gelen her bilgi geldiği gibi bir sonraki katmana gider. Birden fazla girdi gelebilir. Her proses elemanın sadece bir tane girdisi ve bir tane çıktısı vardır. Bu çıktı bir sonraki katmanda bulunan bütün proses elemanlarına gönderilir. Yani, girdi katmanındaki her proses elemanı bir sonraki katmanda bulunan proses elemanlarının hepsine bağlıdır.
- ***Ara katmanlar***: Ara katmanlar girdi katmanından gelen bilgileri işleyerek bir sonraki katmana gönderir. Bir ÇKA ağında birden fazla ara katman ve her katmanda birden fazla proses elemanı olabilir. Ara katmandaki her proses elemanı bir sonraki katmandaki bütün proses elemanlarına bağlıdır.
- ***Çıktı katmanı:*** Ara katmandan gelen bilgileri işleyerek ağa girdi katmanından verilen girdilere karşılık ağın ürettiği çıktıları (*Ç1*, *Ç2*, ...*ÇN*) belirleyerek dış dünyaya gönderir. Bir çıktı katmanında birden fazla proses elemanı olabilir. Her proses elemanı bir önceki katmanda bulunan bütün proses elemanlarına bağlıdır. Her proses elamanın sadece bir tane çıktısı vardır.

ÇKA ağında bilgiler girdi katmanından ağa sunulur ve ara katmanlardan geçerek çıktı katmanına gider ve ağa sunulan girdilere karşılık ağın cevabı dış dünyaya iletilir.

ÇKA ağı öğretmenli öğrenme stratejisini kullanır. Ağa, hem örnekler hem de örneklerden elde edilmesi gereken çıktılar (beklenen çıktı) verilmektedir. Ağ kendisine gösterilen örneklerden genellemeler yaparak problem uzayını temsil eden bir çözüm uzayı üretmektedir. Daha sonra gösterilen benzer örnekler için bu çözüm uzayı sonuçlar ve çözümler üretebilmektedir.

5.3. ÇKA Ağının Öğrenme Kuralı

ÇKA ağları öğretmenli öğrenme stratejisine göre çalışırlar. Yani; bu ağlara eğitim sırasında hem girdiler hem de o girdilere karşılık üretilmesi gereken (beklenen) çıktılar gösterilir. Ağın görevi her girdi için o girdiye karşılık gelen çıktıyı üretmektir. ÇKA ağının öğrenme kuralı en küçük kareler yöntemine dayalı Delta Öğrenme Kuralının genelleştirilmiş halidir. O nedenle öğrenme kuralına *Genelleştirilmiş Delta Kuralı* da denmektedir. Ağın öğrenebilmesi için eğitim seti adı verilen ve örneklerden oluşan bir sete ihtiyaç vardır. Bu set içinde her örnek için ağın hem girdiler hem de o girdiler için ağın üretmesi gereken çıktılar belirlenmiştir. Genelleştirilmiş "Delta Kuralı" iki safhadan oluşur.

❶ *safha- ileri doğru hesaplama*: Ağın çıktısını hesaplama safhasıdır.

❷ *safha- geriye doğru hesaplama*: Ağırlıkları değiştirme safhasıdır.

5.3.1. İleri Doğru Hesaplama

Bu safhada bilgi işleme eğitim setindeki bir örneğin Girdi Katmanından (*G1, G2...*) ağa gösterilmesi ile başlar. Daha önce belirtildiği gibi, girdi katmanında herhangi bir bilgi işleme olmaz. Gelen girdiler hiç bir değişiklik olmadan ara katmana gönderilir. Yani girdi katmanındaki *k.* Proses elemanının çıktısı $Ç_k^i$ şu şekilde belirlenir.

$$Ç_k^i = G_k$$

Ara katmandaki her proses elemanı girdi katmanındaki bütün proses elemanlarından gelen bilgileri bağlantı ağırlıklarının (*A1, A2...*) etkisi ile alır. Önce ara katmandaki proses elemanlarına gelen net girdi (NET_j^a) şu formül kullanılarak hesaplanır.

$$NET_j^a = \sum_{k=1}^{n} A_{kj} Ç_k^i$$

Burada A_{kj} *k.* girdi katmanı elemanını *j.* ara katman elemanına bağlayan bağlantının ağırlık değerini göstermektedir. *j.* ara katman elemanın çıktısı ise bu net girdinin aktivasyon fonksiyonundan (genellikle *sigmoid* fonksiyonundan) geçirilmesiyle hesaplanır. Uygulamada genellikle bu fonksiyon kullanılmakla beraber, kullanılması zorunlu değildir. Önemli olan burada türevi alınabilir bir fonksiyon kullanmaktır. Daha önce belirtilen aktivasyon fonksiyonlardan herhangi birisini burada kullanmak mümkündür. Yalnız geriye doğru hesaplamada burada kullanılan fonksiyonun türevinin alınacağını unutmamak gerekir. *Sigmoid* fonksiyonu kullanılması halinde çıktı,

$$Ç_j^a = \frac{1}{1+e^{-(NET_j^a + \beta_j^a)}}$$

şeklinde olacaktır. Burada β_j, ara katmanda bulunan j. elemana bağlanan eşik değer elemanın ağırlığını göstermektedir. Bu eşik değeri ünitesinin çıktısı sabit olup 1'e eşittir. Ağırlık değeri ise *sigmoid* fonksiyonunun oryantasyonunu belirlemek üzere konulmuştur. Eğitim esnasında ağ bu değeri kendisi belirlemektedir.

Ara katmanın bütün proses elemanları ve çıktı katmanın proses elemanlarının çıktıları aynı şekilde kendilerine gelen NET girdinin hesaplanması ve *sigmoidi* fonksiyonundan geçirilmesi sonucu belirlenirler. Çıktı katmanından çıkan değerler, yani çıktıları, (*Ç1, Ç2, ...*) bulununca ağın ileri hesaplama işlemi tamamlanmış olur.

5.3.2. Geriye Doğru Hesaplama

Ağa sunulan girdi için ağın ürettiği çıktı ağın beklenen çıktıları (*B1, B2,...*) ile karşılaştırılır. Bunların arasındaki fark hata olarak kabul edilir. Amaç bu hatanın düşürülmesidir. O nedenle geriye hesaplamada bu hata ağın ağırlık değerlerine dağıtılarak bir sonraki iterasyonda hatanın azaltılması sağlanır. Çıktı katmanındaki *m.* proses elemanı için oluşan hata (E_m).

$$E_m = B_m - Ç_m$$

olacaktır. Bu bir proses elemanı için oluşan hatadır. Çıktı katmanı için oluşan toplam hatayı (TH) bulmak için bütün hataların toplanması gerekir. Bazı hata değerleri negatif olacağından toplamın sıfır olmasını önlemek amacı ile ağırlıkların kareleri hesaplanarak sonucun kare kökü alınır. ÇKA ağının eğitilmesindeki amaç bu hatayı en azlamaktır. TH şu formül ile bulunur.

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

Toplam hatayı enazlamak için bu hatanın kendisine neden olan proses elemanlarına dağıtılması gerekmektedir. Bu ise proses elemanlarının ağırlıklarını değiştirmek demektir. Ağın ağırlıklarını değiştirmek için iki durum söz konusudur.

- Ara katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi
- Ara katmanlar arası veya ara katman girdi katmanı arasındaki ağırlıkların değiştirilmesi

Ara katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi

Ara katmanındaki j. proses elemanını çıktı katmanındaki m. proses elemanına bağlayan bağlantının ağırlığındaki değişim miktarına ΔA^a denirse; herhangi bir t zamanında (t. iterasyonda) ağırlığın değişim miktarı şöyle hesaplanır.

$$\Delta A_{jm}^a(t) = \lambda \delta_m Ç_j^a + \alpha \Delta A_{jm}^a(t-1)$$

Burada λ öğrenme katsayısını, α momentum katsayısını göstermektedir. Öğrenme katsayısı ağırlıkların değişim miktarını, Momentum katsayısı ise ÇKA ağının öğrenmesi esnasında yerel bir optimum noktaya takılıp kalmaması için ağırlık değişim değerinin belirli bir oranda bir sonraki değişime eklenmesini sağlarlar. Bu konu aşağıda tekrar tartışılacaktır. Eşitlikteki δ_m ise m. çıktı ünitesinin hatasını göstermektedir. Şu şekilde hesaplanır.

$$\delta_m = f'(NET).E_m$$

Buradaki f'(NET) aktivasyon fonksiyonunun türevidir. *Sigmoid* fonksiyonunun kullanılması durumunda;

$$\delta_m = Ç_m(1 - Ç_m).E_m$$

olacaktır. Değişim miktarı hesaplandıktan sonra ağırlıkların t. iterasyondaki yeni değerleri şöyle olacaktır.

$$A_{jm}^a(t) = A_{jm}^a(t-1) + \Delta A_{jm}^a(t)$$

Benzer şekilde eşik değer ünitesinin de ağırlıklarını değiştirmek gerekmektedir. Onun için öncelikle değişim miktarını hesaplamak gerekir. Eğer çıktı katmanında bulunan proses elemanlarının eşik değer ağırlıkları β^c ile gösterilirse; bu ünitenin çıktısının sabit ve 1 olması nedeni ile değişim miktarı,

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

olacaktır. Eşik değerin *t.* iterasyondaki ağırlığının yeni değeri ise,

$\beta_m^{\varsigma}(t) = \beta_m^{\varsigma}(t-1) + \Delta\beta_m^{\varsigma}(t)$ şeklinde hesaplanacaktır.

Ara katmanlar arası veya ara katman girdi katmanı arasındaki ağırlıkların değiştirilmesi

Dikkatli incelenirse, ara katman ile çıktı katman arasındaki ağırlıkların değişiminde her ağırlık için sadece çıktı katmanındaki bir proses elemanının hatası dikkate alınmıştır. Bu hataların oluşmasında girdi katmanı ve ara katman arasındaki ağırlıkların (varsa iki ara katman arasındaki ağırlıkların) payı vardır. Çünkü, en son ara katmana gelen bütün bilgiler girdi katmanı veya önceki ara katmandan gelmektedir. O nedenle girdi katmanı ile ara katman arasındaki (veya iki ara katman arasındaki) ağırlıkların değiştirilmesinde çıktı katmanındaki proses elemanların hepsinin hatasından payını alması gerekir. Bu ağırlıklardaki değişimi (mesela girdi katmanı ile ara katman arasındaki ağırlıkların değişimi) ΔA^i ile gösterilirse değişim miktarı;

$$\Delta A_{kj}^i(t) = \lambda\delta_j^a Ç_k^i + \alpha\Delta A_{kj}^i(t-1)$$

olacaktır. Buradaki hata terimi δ^a ise şöyle hesaplanacaktır.

$$\delta_j^a = f'(NET)\sum_m \delta_m A_{jm}^a$$

Aktivasyon fonksiyonu olarak *sigmoid* fonksiyonu düşünülürse bu hata değeri şu şekilde hesaplanacaktır.

$$\delta_j^a = Ç_j^a(1-Ç_j^a)\sum_m \delta_m A_{jm}^a$$

Hata değeri hesaplandıktan sonra yukarıda verilen eşitlik ile değişim miktarını bulmak mümkün olur. Ağırlıkların yeni değerleri ise,

$$A_{kj}^i(t) = A_{kj}^i(t-1) + \Delta A_{kj}^i(t)$$

şeklinde olacaktır. Benzer şekilde, eşik değer ünitesinin yeni ağırlıkları da yukarıdaki gibi hesaplanır. Ara katman eşik değer ağırlıkları β^a ile gösterilirse değişim miktarı,

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

olacaktır. Ağırlıkların yeni değerleri ise *t.* iterasyonda şöyle hesaplanacaktır.

$$\beta_j^a(t) = \beta_j^a(t-1) + \Delta\beta_j^a(t)$$

Böylece ağın ağırlıklarının hepsi değiştirilmiş olacaktır. Bir iterasyon hem ileri hem de geriye hesaplamaları yapılarak tamamlanmış olacaktır. İkinci bir örnek verilerek sonraki iterasyona başlanır ve aynı işlemler öğrenme tamamlanıncaya kadar yinelenir.

5.4. ÇKA Ağının Çalışma Prosedürü

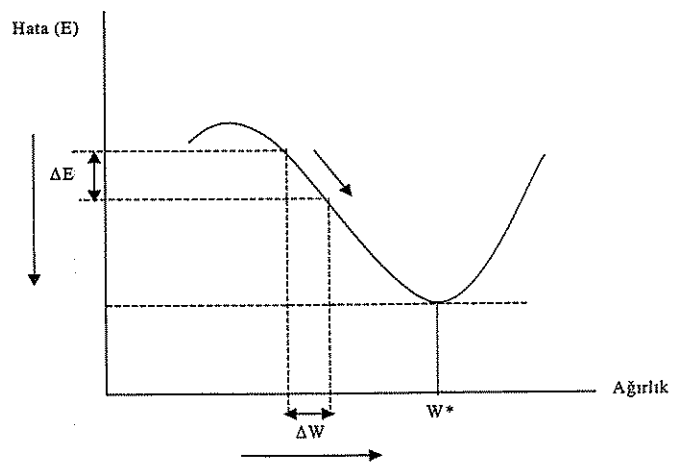
ÇKA ağlarının çalışması şu adımları içermektedir:

- *Örneklerin toplanması*: Ağın çözmesi istenilen olay için daha önce gerçekleşmiş örneklerin bulunması adımıdır. Ağın eğitilmesi için örnekler toplandığı gibi (eğitim seti) ağın test edilmesi için de örneklerin (test seti) toplanması gerekmektedir. Ağın eğitilmesi sırasında test seti ağa hiç gösterilmez. Eğitim setindeki örnekler tek tek gösterilerek ağın olayı öğrenmesi sağlanır. Ağ olayı öğrendikten sonra test setindeki örnekler gösterilerek ağın performansı ölçülür. Hiç görmediği örnekler karşısındaki başarısı ağın iyi öğrenip öğrenmediğini ortaya koymaktadır.
- *Ağın topolojik yapısının belirlenmesi*: Öğrenilmesi istenen olay için oluşturulacak olan ağın topolojik yapısı belirlenir. Kaç tane girdi ünitesi, kaç tane ara katman, her ara katmanda kaç tane proses elemanı ve kaç tane çıktı elemanı olması gerektiği bu adımda belirlenmektedir.
- *Öğrenme parametrelerini belirlenmesi*: Ağın öğrenme katsayısı, proses elemanlarının toplama ve aktivasyon fonksiyonları, momentum katsayısı gibi parametreler bu adımda belirlenmektedir.
- *Ağırlıkların başlangıç değerlerinin atanması*: Proses elemanlarını birbirlerine bağlayan ağırlık değerlerinin ve eşik değer ünitesinin ağırlıklarının başlangıç değerlerinin atanması yapılır. Başlangıçta genellikle rasgele değerler atanır. Daha sonra ağ uygun değerleri öğrenme sırasında kendisi belirler.
- *Öğrenme setinden örneklerin seçilmesi ve ağa gösterilmesi*: Ağın öğrenmeye başlaması ve yukarıda anlatılan öğrenme kuralına uygun olarak ağırlıkları değiştirmesi için ağa örnekler (Girdi/Çıktı değerleri) belirli bir düzeneğe göre gösterilir.
- *Öğrenme sırasında ileri hesaplamaların yapılması*: Yukarıda anlatıldığı şekilde sunulan girdi için ağın çıktı değerleri hesaplanır
- *Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması*: Ağın ürettiği hata değerleri bu adımda hesaplanır.
- *Ağırlıkların değiştirilmesi*: Yukarıda anlatıldığı gibi geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

Yukarıdaki adımlar ÇKA ağının öğrenmesi tamamlanıncaya, yani gerçekleşen çıktılar ile beklenen çıktılar arasındaki hatalar kabul edilir düzeye ininceye, kadar devam eder. Ağın öğrenmesi için bir durdurma kriterinin olması gerekmektedir. Bu ise genellikle üretilen hatanın belirli bir düzeyin altına düşmesi olarak alınmaktadır.

5.5. Ağın Eğitilmesi

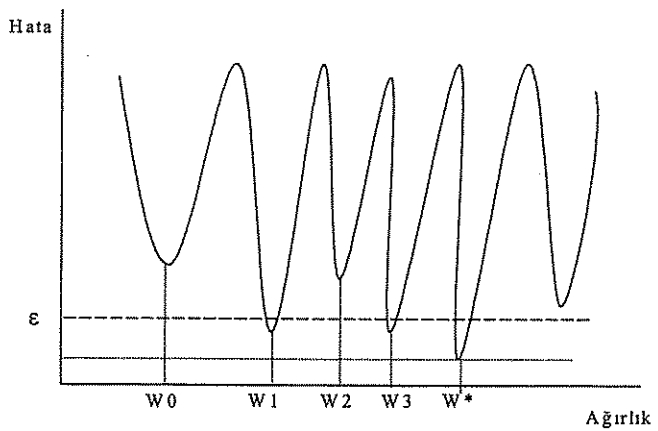
ÇKA ağlarının eğitilmesi felsefesi diğer ağlarınkinden farklı değildir. Ağın kendisine gösterilen girdi örneği için beklenen çıktıyı üretmesini sağlayacak ağırlık değerleri bulunmaktadır. Başlangıçta bu değerler rasgele atanmakta ve ağa örnekleri gösterdikçe ağın ağırlıkları değiştirilerek zaman içerisinde istenen değerlere ulaşması sağlanmaktadır. İstenen ağırlık değerlerinin ne olduğu bilinmemektedir. Bu nedenle yapay sinir ağlarının davranışlarını yorumlamak ve açıklamak mümkün olmamaktadır. Zaten bu ağların diğer yapay zeka tekniklerinden meselâ uzman sistemlerden, ayıran en önemli farkı da davranışlarını açıklayamamasıdır. Bunun temel nedeni Bölüm 2'de açıklandığı gibi bilginin ağ üzerine dağıtılmış olması ve ağırlık değerlerinin kendi başlarına herhangi bir anlam göstermemeleridir. Ağ ile ilgili bilinen konu problem uzayında en az hata verebilecek ağırlık değerlerinin bulunmasıdır. Hatanın en az değeri kavramını anlatabilmek için basit bir problem düşünülürse, ağın öğrenmesi istenen olayın (problem uzayının) Şekil-5.2'de gösterildiği gibi bir hata uzayının olduğu varsayılsın. Şekildeki W* en az hatanın olduğu ağırlık vektörünü göstermektedir.



Şekil-5.2. Öğrenmenin hata uzayındaki gösterimi

Ağın W* değerine ulaşması istenmektedir. Bu ağırlık değeri problem için hatanın en az olduğu noktadır. O nedenle her iterasyonda ΔW kadar değişim yaparak hata düzeyinde ΔE kadar bir hatanın düşmesi sağlanmaktadır. Burada bir noktaya dikkatleri çekmek gerekir. Problemin hata düzeyi her zaman böyle basit ve iki boyutlu olmayacaktır. Şekil-5.3 daha karmaşık bir hata düzeyini göstermektedir. Görüldüğü gibi problemin çözümü için en az hatayı veren ağırlık vektörü W* olmasına rağmen pratikte bu hata değerini yakalamak çoğu zaman mümkün olmayabilmektedir. Bu çözüm ağın sahip olabileceği ***en iyi çözüm***dür. Fakat bu çözüme nasıl ulaşılacağı konusunda elimizde bir bilgi yoktur. Ağ, eğitim sırasında kendisi bu çözümü yakalamaya çalışmaktadır.

Bazen farklı bir çözüme takılabilmekte ve performansı daha iyileştirmek mümkün olamamaktadır. O nedenle kullanıcılar ağların performanslarında (problemlere ürettikleri çözümlerde) ε kadar hatayı kabul etmektedirler (tolerans değeri). Tolerans değeri altındaki her hangi bir noktada olay öğrenilmiş kabul edilmektedir. Şekildeki W0 ve W1 çözümlerinin hataları kabul edilebilir hata düzeyinin üzerinde olduğundan bu çözümler kabul edilemez çözümlerdir. Bunlara ***yerel çözümler*** denilmektedir. W1 ve W3 çözümleri en iyi çözüm olmamalarına rağmen kabul edilebilir hata düzeyinin altında bir hataya sahiptiler. Bunlarda yerel çözümler olmalarına rağmen kabul edilebilir çözümlerdir. Görüldüğü gibi bir problem için birden fazla çözüm üretilebilmektedir. Bu nedenle yapay sinir ağlarının her zaman en iyi çözümü ürettikleri söylenemez. Kabul edilebilir bir çözüm ürettiklerini söylemek daha doğrudur. Üretilen çözüm en iyi çözüm olsa bile bunun bilinmesi zordur. Çoğu durumda bilinmesi mümkün değildir.



Şekil-5.3. Çok boyutlu hata uzayı

Şekil-5.3 aynı zamanda başka bir gerçeği daha göstermektedir. Neden en iyi sonuç bulunamamaktadır? Bunun başka nedenleri de olabilir. Örneğin,

- Problem eğitilirken bulunan örnekler problem uzayını %100 temsil etmeyebilir.
- Oluşturulan ÇKA ağı için doğru parametreler seçilmemiş olabilir.
- Ağın ağırlıkları başlangıçta tam istenildiği şekilde belirlenmemiş olabilir.
- Ağın topolojisi yetersiz seçilmiş olabilir.

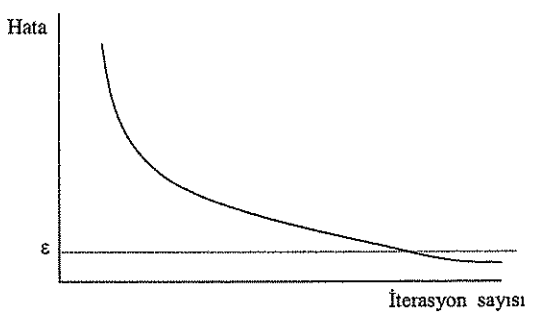
Bu ve benzeri nedenlerden dolayı ağ, eğitim sırasında, hatayı belirli bir değerin altına düşüremeyebilir. Mesela W1 ağırlıklarını bulur hatayı daha aşağıya düşüremez. Bu aslında yerel bir çözümdür. En iyi çözüm değildir. Hata kabul edilebilir düzeye indiğinden yerel en iyi bir çözüm olarak görülebilir. Global çözümün bulunması da mümkün olabilir. Bu tamamen ağın tasarımına, örneklerin niteliğine ve eğitim sürecine bağlıdır.

Bazı durumlarda ağın takıldığı lokal sonuç kabul edilebilir hata düzeyinin üstünde kalabilir (Mesela Şekil-5.4'teki W0 ağırlıklarının bulunması ve hatanın daha fazla azaltılmasının mümkün olmaması). Bu durumda ağın olayı öğrenmesi için bazı değişiklikler yapılarak yeniden eğitilmesi gerekir. Bu değişiklikler arasında şunlar sayılabilir:

- Başka başlangıç değerlerinin kullanılabilir.
- Topolojide değişiklikler yapılabilir (ara katman sayısını artırmak, proses elemanı sayısını artırmak veya azaltmak gibi),
- Parametrelerde değişiklik yapılabilir (fonksiyonların başka seçilmesi, öğrenme ve momentum katsayılarının değiştirilmesi gibi)
- Problemin gösterimi ve örneklerin formülasyonu değiştirilerek yeni örnek seti oluşturulabilir
- Öğrenme setindeki örneklerin sayısı artırılabilir veya azaltılabilir
- Öğrenme sürecinde örneklerin ağa gösterilmesi

ÇKA ağlarının yerel sonuçlara takılıp kalmaması için momentum katsayısı geliştirilmiştir. Bu katsayının iyi kullanılması yerel çözümleri kabul edilebilir hata düzeyinin altına çekebilmektedir.

ÇKA ağlarının eğitilmesinde diğer önemli bir sorun ise öğrenme süresinin çok uzun olmasıdır. Ağırlık değerleri başlangıçta büyük değerler olması durumunda ağın lokal sonuçlara düşmesi ve bir lokal sonuçtan diğerine sıçramasına neden olmaktadır. Eğer ağırlıklar küçük aralıkta seçilirse o zaman da ağırlıkların doğru değerleri bulması uzun zamanlar almaktadır. Bazı problemlerin çözümü sadece 200 iterasyon sürerken bazıları 5-10 Milyon iterasyon gerektirmektedir. Bu konuda da elimizde bilimsel bir yaklaşım yoktur. Tamamen deneme yanılma yolu ile en uygun başlama koşullarının belirlenmesi gerekmektedir. Aşağıda ÇKA ağlarını daha etkin kullanabilmek için bu kapsamda bazı öneriler verilecektir.



Şekil-5.4. Öğrenen bir ÇKA ağının öğrenme eğrisine örnek

Ağın öğrenmesinin gösterilmesinin en güzel yol hata grafiğini çizmektir. Öğrenen bir ağ için her iterasyonda oluşan hatanın grafiği çizilirse Şekil-5.4'te gösterildiği şekle benzer bir hata grafiği oluşur. Burada hatanın zaman içerisinde düştüğü görülür.

Belirli bir iterasyondan sonra hatanın daha fazla azalmadığı görülür. Bu ağın öğrenmesini durdurduğu ve daha iyi bir sonuç bulunamayacağı anlamına gelir. Eğer elde edilen çözüm kabul edilemez ise o zaman ağ yerel bir çözüme takılmış demektir. Bu durumda ağın yukarıda önerilen değişiklikleri yaparak yenide eğitilmesi gerekir.

5.6. XOR Probleminin Çözülmesi

ÇKA ağlarının yapısını ve öğrenme kuralını anlattıktan sonra XOR problemine nasıl sonuç ürettiklerini bir örnek üzerinde göstermek faydalı olacaktır. Çünkü bu örnek, ÇKA ağlarının bulunmasının en temel nedenlerinden birisidir. Bu problem, yapay sinir ağlarında bir devrin kapanıp bir devrin açılmasına neden olmuş önemli bir kilometre taşıdır. Yukarıda anlatılan ÇKA ağının çalışma süreci XOR problemine şöyle uygulanır.

1. Adım: *Örneklerin Toplanması:*

Daha önce Şekil-5.1'de de gösterildiği gibi XOR problemi için 4 örnek vardır. Bunlar 1 ve 0 değerlerinden oluşmaktadır. Her örnek için girdiler ve beklenen çıktı şöyledir:

	Girdi 1	Girdi 2	Çıktı
Örnek 1	0	0	0
Örnek 2	0	1	1
Örnek 3	1	0	1
Örnek 4	1	1	0

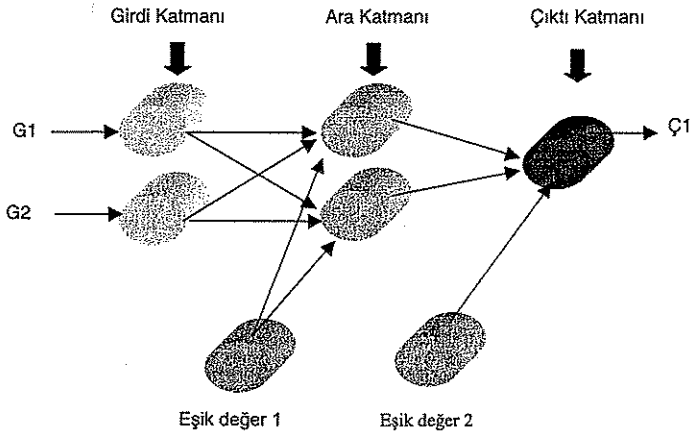
2. Adım: *Ağın Topolojik Yapısının Belirlenmesi:*

XOR probleminde 2 girdi ve 1 de çıktı olduğundan, oluşturulacak olan ÇKA ağının da 2 girdi ünitesi ve 1 çıktı ünitesi olacaktır. 1 ara katman ve 2 tanede ara katman proses elemanının bu problemi çözebileceği varsayılmaktadır. Şekil-5.5'de oluşturulan ağın toplojisi gösterilmektedir. Görüldüğü gibi ara katman için bir adet, çıktı katmanı içinde bir adet eşik değer ünitesi vardır.

3. Adım: *Öğrenme Parametrelerini Belirlenmesi:*

Oluşturula ağ için aktivasyon fonksiyonu olarak *sigmoid* fonksiyonunun kullanıldığını, öğrenme (λ) ve momentumun (α) katsayılarının ise şu şekilde belirlendiği varsayılsın,

$\lambda = 0.5$
$\alpha = 0.8$



Şekil-5.5. XOR problemi için tasarlanmış bir ÇKA

4. Adım: *Ağırlıkların Başlangıç Değerlerinin Atanması:*

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

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

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

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

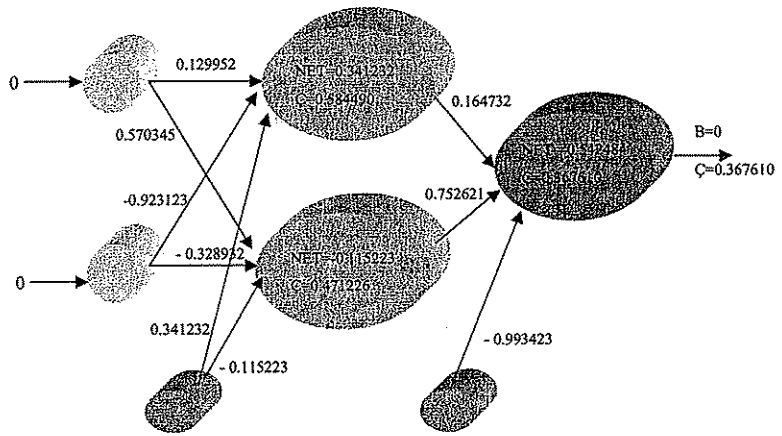
$$A^a = [0.164732 \quad 0.752621]$$

Eşik değeri ağırlıkları şöyle olsun.

$$\beta^a = [0.341332 \quad -0.115223]$$
$$\beta^ç = [-0.993423]$$

5. Adım: *Örneklerin Ağa Gösterilmesi ve İleri Doğru Hesaplama:*

Birinci örnek *G1*=0, *G2*=0 ve *B*=0 olarak belirlenmiştir. Bu örnek ağa gösterilirse, ileri doğru hesaplama Şekil-5.6'da gösterildiği gibi olacaktır.



Şekil-5.6. ileri doğru hesaplama

Ara katman ünitelerinin NET girdileri (eşik değer ünitesinin ağırlık değerleri eklenmiş olarak) şu şekilde hesaplanır.

Net1= (0 * 0.129952) + (0 * -0.923123)+ (1 * 0.341232) = 0.341232
Net2=+ (0 * 0.570345) + (0 * -0.328932) +(1 *-0.115223) = -0.115223

Ara katman ünitelerinin çıktıları ise şöyle hesaplanır.

$$Ç_1 = \frac{1}{1+e^{-0.341232}} = 0.584490$$

$$Ç_2 = \frac{1}{1+e^{0.115223}} = 0.471226$$

Çıktı katmanındaki proses elemanının NET girdisi hesaplanırsa;

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

Değeri bulunur. Bu değer ile ağın çıktısı;

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

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.

$$\delta_1 = Ç_1(1 - Ç_1).E_1$$
$$\delta_1 = 0.367610 * (1 - 0.367610) * (- 0.367610)$$
$$\delta_1 = -0.085459$$

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

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

$$\Delta \beta_1^{c}(t) = -0.042730$$

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

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

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

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

$$\beta_1^{c}(t) = -0.993423 - 0.042730 = -1.036153$$

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.

$$\delta_1^{a}(t) = Ç_1(1 - Ç_1)\delta_1 A_{11}^{a}(t-1)$$

$$\delta_1^{a} = 0.584490 * (1 - 0.584490) * (0.164732) * (-0.085459)$$

$$\delta_1^{a} = -0.034190$$

$$\delta_2^{a} = -0.160263$$

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

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

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

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

$$\Delta \beta_1^{i}(t) = 0.5 * 1 * -0.034190 = -0.017095$$

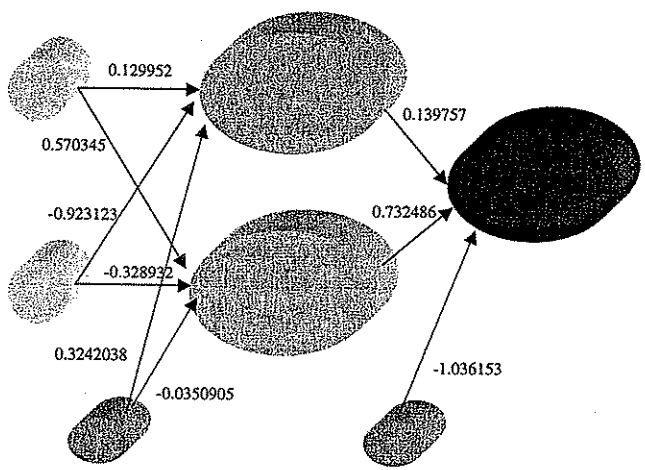
$$\Delta \beta_2^{i}(t) = 0.5 * 1 * -0.160263 = -0.080132$$

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

$$\beta_1^{i}(t) = 0.341232 - 0.017095 = 0.3242038$$

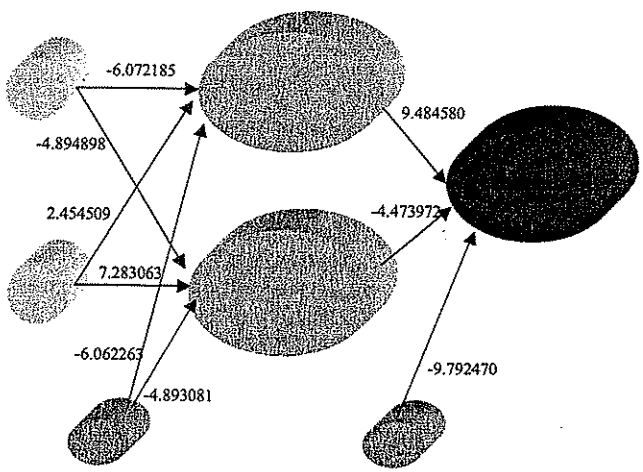
$$\beta_2^{i}(t) = 0.115223 - 0.081325 = -0.0350905$$

Şekil-5.7'de yeni belirlenen ağırlık değerleri gösterilmektedir:



Şekil-5.7. XOR ağının Ağırlıklar değiştirildikten sonraki durumu

Birinci iterasyon bittikten sonra ikinci iterasyon başlayacaktır. Bu kez ikinci örnek ağa gösterilir. G1=0, G2=1 ve B=1 olacaktır. Yukarıdaki işlemler aynı şekilde tekrar edilir. Bu iterasyonlar bütün çıktılar doğru cevap verinceye kadar devam etmelidir. Şekil-5.8 oluşturulan ÇKA ağı öğrendikten sonra ağın ağırlık değerlerini göstermektedir.



Şekil-5.8. XOR problemi öğrendikten sonraki ağırlıklar

Bu ağırlıklar ile girdiler ağa tekrar gösterildiğinde o zaman Tablo-5.2'de gösterilen sonuçlar elde edilir. Bu sonuçlar ağın problemi çok düşük hatalar ile çözebilecek şekilde öğrendiğini göstermektedir.

Tablo-5.2: XOR problemini öğrendikten sonra ağın ürettiği çözümler ve hata oranları

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

5.7. ÇKA Ağının Performansının Ölçülmesi

Bir yapay sinir ağının performansı denilince öğrenme yeteneğinin ölçülmesi anlaşılır. Ağ eğitim sırasında kendisine gösterilen bütün örneklere doğru cevaplar üretiyor diye performansı iyidir denemez. Bu ağın öğrenip öğrenmediğini gösterebilir ama iyi öğrenip öğrenmediğini göstermez. O nedenle, öğrenen ağların daha önce görmedikleri örnekler karşısında da beklenen performansı gösterip göstermeyeceklerinin ölçülmesi gerekmektedir. Bunun için genellikle ağın eğitildiği problem üzerinden hem eğitimde kullanılacak hem de test esnasında kullanılacak örnekler seçilir. Eğitim sırasında ağa sadece eğitim setindeki örnekler gösterilir. Ağ bunları öğrenince (hepsi için veya kabul edilebilir oranda hepsi için doğru cevaplar üretmeye başlayınca) ağa hiç görmediği test setindeki örnekler gösterilir. Ağın performansı bu görmediği örnekler karşısında ürettiği doğru cevaplar oranı ile şekilde ölçülür.

$$P = \frac{D}{T} x100$$

Burada D test setinden doğru olarak cevaplandırılan örnek sayısını, T test setinde bulunan toplam örnek sayısını, P ise performans oranını göstermektedir. Eğer performans oranı (P) istenilen düzeyde veya kabul edilebilir bir değerde değil ise ağ eğitim setindeki bütün örnekleri doğru cevaplasa bile iyi öğrenmiştir denemez. O zaman eğitime biraz daha devam etmek gerekebilir. Eğitim iterasyonlarını artırmaya rağmen hala performans artmıyor ise o zaman örneklerin problem uzayını iyi temsil edemedikleri veya ağın parametrelerinin veya topolojisinin iyi seçilemediği anlaşılır. Daha önce belirtilen değişiklikleri yaparak ağı yeniden eğitmek gerekebilir. Ağın eğitilmesi sırasında dikkat edilmesi gereken konulara aşağıda da değinilecektir.

5.8. ÇKA Ağının Öğrenmek Yerine Ezberlemesi

Bazı durumlarda eğitilen ÇKA ağı eğitim setindeki bütün örneklere %100 doğru cevap üretmesine rağmen test setindeki örneklere doğru cevaplar üretememektedir. Ancak %10-%20 civarında bir performans yakalanabilmektedir. Hatta bu oranlara bile

ulaşılamamaktadır. Bu durumda ÇKA ağının öğrenmediği fakat öğrenme setini ezberlediği görülmektedir. ÇKA ağı tasarımcıları bu durumdan kurtulmak istemektedirler. Çünkü öğrenim%100 görülmekte fakat ağ öğrenmemektedir. Bu ağın günlük kullanıma alınması mümkün olmaz. Onun için aşağıda belirtilen konular tekrar gözden geçirilerek ağın ezberlemesinden kurtulmak ve gerçekten öğrenmesini sağlamak gerekir. Çok iyi ezberleyen bir ağ yerine azar azar öğrenen ve kabul edilebilir bir hata ile öğrenme gerçekleştiren performansı düşük ağ daha iyidir.

5.9. Bir ÇKA Ağının Oluşturulmasında Dikkat Edilmesi Gereken Bazı Önemli Noktalar

Yapılan araştırmalar ve tecrübeler bir ÇKA ağının performansını etkileyen unsurların şunlar olduğunu göstermektedir.

- Örnekleri seçilmesi
- Girdi ve çıktıların ağa gösterimi
- Girdilerin nümerik gösterimi
- Çıktıların nümerik gösterimi
- Başlangıç değerlerinin atanması
- Öğrenme ve momentum katsayılarının belirlenmesi
- Örnekleri ağa sunulması
- Ağırlıkların değiştirilme zamanları
- Girdi ve çıktıların ölçeklendirilmesi
- Durdurma Kriterinin belirlenmesi
- Ara katmanların ve her katmandaki proses elemanlarının sayısının belirlenmesi
- Ağların büyütülmesi veya budanması

Bu unsurların ağın performansına etkileri aşağıda tartışılmış ve her birisi ile ilgili önerilerde bulunulmuştur.

5.9.1. Örneklerin Seçilmesi

Örneklerin seçilmesi ağın performansını yakından ilgilendiren bir konudur. Çünkü ağ, bu örnekleri dikkate alarak ağırlıklarını değiştirmektedir. Seçilen örneklerin problem uzayını temsil edebilecek nitelikte olması çok önemlidir. Bazı ÇKA ağı tasarımcıları problem uzayının sadece bir dilimini gösteren örnekleri ağa göstermekte fakat tamamı ile ilgili yorumlar yapmasını beklemektedir. Bu mümkün değildir. Bazıları ise elmayı gösterip portakalı sormaktadır. Unutulmaması gereken şudur ki ağa ne gösterilirse ağ ancak o konularda yorumlar yapabilir ve ancak o konuda görmediği örneklere çözümler üretebilir. 2X2=4 diye ağa öğretirseniz 3X3 kaç eder diye soramazsınız. Çünkü ağ bu konuda bir örnek görmemiş ve genellemeleri yapacak durumda olmamıştır.

Bazı durumlarda ise ağa problem uzayının sadece uç değerleri gösterilmekte ve bütün problem uzayını öğrenmesi istenmektedir. Bu durumda da ağın problem uzayını öğrenmesi beklenemez. ÇKA ağı tasarımcılarının problem uzayının her bölgesinden ve uzayı temsil eden örnekler seçmesi gerekmektedir. Özellikle ÇKA ağının yapısını ve çalışmasını yeni öğrenmeye başlayanlara şu örneği denemelerini önermek isterim. Çünkü bu örnek ile yapay sinir ağları ve ÇKA ağının eğitilmesi ve çalıştırılmasını çok kolay anlamak mümkün olduğu gibi örnek uzayı hakkında da bilgiler bulmak mümkündür. Örnekte; bir ÇKA ağına Çarpım tablosundan 2 ile çarpım yapması öğretilmek istenmektedir. Bu örnekte problem uzayı toplam 10 örnekten oluşmaktadır. Bunlar Tablo-5.3'de verilmiştir.

Tablo-5.3. İki ile çarpım tablosu değerleri

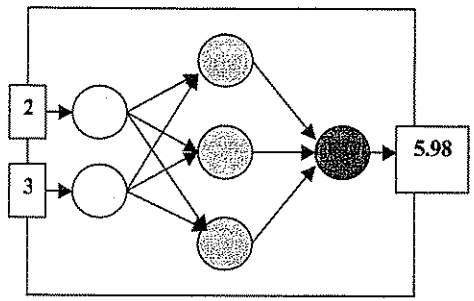
2 ile çarpma	
2x1=2	2x6=12
2x2=4	2x7=14
2x3=6	2x8=16
2x4=8	2x9=18
2x5=10	2x10=20

Bu örnek seti ikiye bölünürse birisi eğitim seti diğeri ise test seti olarak kullanılabilir. Bu bölme işlemini nasıl yapmak gerekir? Örneğin ilk 5 tanesi eğitim setine konulursa problem uzayının sadece ilk yarısını öğretilmiş olur. Diğer yarısı ile ilgili sorulara ağın doğru cevap vermesi beklenemez. Eğitim setine baştan 2 sondan 3 örnek alınsa da yine aynı şekilde problemi tam temsil eden bir eğitim setine kavuşulmuş olunmaz. Örneklerin tamamı eğitim setine konulsa performansı test edecek bir örnek kalmaz. Bu durumda mesela çift sayılar ile çarpımı sonuçlarını eğitim setine koyup öğrendikten sonra tek sayıların 2 ile çarpımı sorulabilir. Bu şekilde oldukça başarılı bir ağ oluşturmak mümkün olabilir. Çünkü problem uzayının her bölgesinden örnekler alınmakta ve ağın genelleme yapabilmesi sağlanmaktadır. Yapılan genelleme ile görmediği örnekler için de doğru çarpım sonuçları vermektedir. Bu açıklamaya dayanarak oluşturulan eğitim ve test setleri Tablo-5.4'de gösterildiği gibi tasarlanabilir.

Tablo-5.4. İki ile çarpım tablosu eğitim ve test örnekleri

Eğitim seti	Test seti
2x2=4	2x1=2
2x4=8	2x3=6
2x6=12	2x5=10
2x8=16	2x7=14
2x10=20	2x9=18

Böyle bir örnek için tasarlanan bir ağ, 200 iterasyonluk öğrenme sonunda kendisine sorulan 2x3 kaç eder sorusuna 5.98 gibi bir cevap vermiştir. Bu ise öğrenmenin ne kadar başarılı olduğunu göstermektedir. Çünkü ağ, hiç görmediği bir örnek için %0.2'lik bir hata ile doğru cevabı yakalamıştır. Bu sonucu üreten ağın topolojisi Şekil-5.9'te verilmektedir. Şekildeki dikdörtgen kutucuklar, girdilerin ağa sunulmadan önce bir ön işleme tutulduğunu göstermektedir. Benzer şekilde çıktı değerleri de üretildikten sonra bir işlemden geçirilerek dış dünyaya iletilmektedir. Bu işleme neden gerek duyulduğu ileride anlatılacaktır.



Şekil-5.9. İki ile çarpım tablosunu öğrenen ağın yapısı

Hangi örneklerin problem uzayını temsil ettiğini belirlemek burada verilen örnekteki gibi kolay olmayabilir. Belirlenen örneklerin uzayı temsil etme yeteneklerini ölçebilecek bir yöntemde şu ana kadar geliştirilmemiştir. Fakat tasarımcı kendisi örnekleri seçerken ekstrem uçlardan ve sadece belirli bölgelerden örnekler almaktan kaçınılmalıdır. Örneğin Kadın ve Erkek resimlerini birbirlerinden ayıran bir ÇKA modeli tasarlanması istense erkek resimlerini temsilen sadece gür saçlı erkek resimlerini kullanıp ondan sonra saçı kel olan bir erkek resmini tanımasını istemek doğru olmaz. Bıyıklı ve sakallı erkek resimlerinin de tanınması söz konusu olacak ise o zaman onlarında örneklerinin eğitim seti içinde yer alması gerekmektedir. Aksi taktirde ya ağ başarısız olacak ya da problemin tanımı sadece gür saçlı erkekleri kadınlardan ayıran bir ÇKA ağı olacaktır. Bu konuya tasarımcı mümkün olduğu kadar dikkat etmelidir. Bu konuda önerilen, öncelikle bütün örnekleri belirleyip onları eğitim ve test seti olarak ikiye bölmektir. Ağ eğer test setine başarılı sonuçlar üretiyorsa öğrenmiş demektir. Çünkü test setindeki örnekleri ağ öğrenirken görmemektedir. Test setinde başarılı olan fakat günlük kullanımda sorunlar gösteren bir ağ için başarısız demek doğru değildir. Problem ağa iyi gösterilememiş ve iyi örnekler seçilememiş olması olasıdır diye bakmak lazımdır. Onun için bu gibi durumlarda öncelikle örnek setinin gözden geçirilmesi gerekmektedir. Ağın başarısız olduğu örnekler belirlenip toplanarak eğitim setine katılır ve ağ yeniden eğitilerek performansı artırılır. Böylece, zaman içinde problem uzayını gösteren eğitim setine ve ağa kavuşmak mümkün olabilir. Yalnız böyle bir uygulama ilgili ağın test seti üzerindeki performansının yüksek olması

durumunda yapılması gerekir. Test setine iyi cevaplar üretemeyen ağ zaten iyi öğrenememiştir demektir. Orada örnek seti kadar başka sorunlarda söz konusu olabilir. Bu durumda; aşağıda anlatılanlar ışığında örnek setini de içerecek bir çalışma ile düzenlemeler yapılarak ağın performansının artırılması gerekir. Ağın yeni örnekler ile eğitilmesi öğrenmenin gerçekleşmesi durumunda faydalı olacaktır.

Bu konudaki diğer bir önemli konuda seçilen örneklerin ilgili problem uzayında gerçekleşmiş ve çözülmüş gerçek örnekler olmasıdır. Hayali ve varsayılan çözümler ve örnekler başarılı sonuçların doğmasını önleyebilirler. Bazı durumlarda örnek setini oluşturmak çeşitli nedenlerden dolayı mümkün olmayabilir. Bu durumlarda ilgili problem uzayının bir benzetim modeli oluşturularak gereken örnekleri üretmek mümkün olabilir. Bu durumda benzetim modelinin gerçeğe ne kadar yakın tasarlandığı önemlidir. Ağın bu benzetim modeli üzerinden gelen örnekler ile öğrenip günlük kullanımda sonuçlar üretememesi ağın başarısız olduğu anlamına gelmez. Tasarımcılar başarısız sonuçlar elde edildiği zaman öncelikle benzetim modelinin iyi tasarlandığından emin olmalıdırlar. Benzer şekilde benzetim modelinin de yine bütün problem uzayını gösteren örnekler üretecek şekilde tasarlanması gerekmektedir.

Örneklerin seçiminin önemini kavramayan bazı tasarımcılar aylarca ağları eğitememekte ve öğrenmiyor diye şikayet etmektedirler. Örneklerini inceleyip gerekli düzenlemeleri yapsalar belki de 1-2 saat içinde öğrenecek bir ağ tasarlayabileceklerdir. Kısaca ağa ne gösterilirse karşılığında onun benzeri sonuçlar alınır. Bu gerçeği göz ardı etmeden eğitim ve test seti oluşturulmalıdır.

5.9.2. Girdi ve Çıktıların Gösteriminin Belirlenmesi

Örneklerin belirlenmesi kadar belki ondan da daha önemlisi örneklerin gösteriminin nasıl olacağının belirlenmesidir. Girdi/çıktı çiftlerinden oluşan örnekler ağa nasıl gösterilecektir? Yapay sinir ağları daha önce belirtildiği gibi sadece rakamlar ile çalışmaktadırlar. Eğer problem uzayında sayısal (nümerik) olmayan faktörleri dikkate almak gerekiyor ise o zaman onların rakamlar ile temsil edilebilmesi gerekmektedir. Bu dönüştürme çeşitli şekillerde olabilmekte ve bu da ağın performansını etkilemektedir. Hem girdi değerlerinin hem de beklenen çıktı değerlerinin nümerik olarak gösterilmesi gerekmektedir.

5.9.2.1. Girdi Değerlerinin Nümerik Gösterimi

Ağa gösterilen girdi değerlerini ağın anlayabilmesi için nümerik olma zorunluluğu problemin girdilerinin nümerik gösterimini gerektirmektedir. Bu ise her zaman kolay olmamakta ve problem tasarımcısını zor durumda bırakabilmektedir. Çünkü bu güne kadar geliştirilmiş her olay için uygulanabilir bir dönüştürme mekanizması geliştirilmemiştir. Her olay için ayrı bir yöntem uygulanabilmektedir. Hatta birden fazla yöntem arasından bir tanesi seçilmektedir. Bu seçim de ağın performansı üzerinde etkili olabilmektedir.

Nümerik gösterime dönüştürmek için şu örneğe bakılabilir; kadın ve erkek resimlerini birbirlerinden ayıran bir ağ tasarlanacak olsa, bu ağ için girdi ve çıktı ne olabilir? Kadın veya erkek resmini gösteren ve rakamlar ile ifade edilebilen bir gösterim şekline ihtiyaç vardır. Bunun için akla ilk gelen şey resmin bilgisayar ortamındaki gri tonlarını kullanmaktır. Bir resmin 256x256 pixelden oluşan bir çerçevede çekilmesi sonucu resmin her noktasının gri tonunu gösteren 0-256 arasında bir sayı vardır. Bu kolay bir çözümdür. Ama uygulaması çok zordur. Çünkü bir resim için yaklaşık 256x256= 65536 adet rakam söz konusudur. Her rakam için bir proses elemanı kullanılsa bu oluşturulacak olan ağın 65536 adet girdi ünitesinin olması demektir. Bu ise oldukça büyük bir ağ olur ve hem öğrenmesi hem de çalışmasında zaman bakımından sorunlar yaşanabilir. Çünkü böyle bir ağ için belirlenecek olan ara katman ünite sayısı ile girdi sayısının çarpımı kadar bir hesaplama sadece girdi ve ara katman arasında hem ileri ve hem de geri doğru yapılması gerekmektedir. Bunun yerine, bu 65536 rakamın sınıflandırılması söz konusu olabilir. Sadece aynı rakamlar sayılsa o zaman 257 tane girdi olur. Bu durumda 0 ila 256 arasında her sayıdan kaç tane resim üzerinde var ise sayılarak frekanslar bulunup girdi olarak ağa verilebilir. Bu yaklaşımın diğerine göre hesaplama açısından ne kadar büyük bir rahatlık getirdiği ortadadır. Bu yolla, zamanı yaklaşık olarak 256 misli kısaltmak mümkün olacaktır. Bazı durumlarda hesaplama etkinliği açısından bu sayıda çok olabilir. O zaman sınıflandırmayı belirli aralıklara çekerek bir girdi seti oluşturulabilir. Bu aralıkların genişliği probleme göre değişebilir. Burada örnek olsun diye 0-256 arasının 10 parçaya bölünmesi sonucu ilgili resim üzerindeki gri tonların değerleri sayılarak her gruba düşen değerlerin sayısı bulunur ve ağın girdi örneği oluşturulur. Böylece ağın 10 adet girdisi olacak demektir. Bu yapılan dönüşümlerde birinciden üçüncüye kadar sürekli bilgi kaybının da olabileceğini ve resimlerin bazı ince detaylarının kaybolabileceğini ve ağın görmesinin mümkün olamayacağını kabul etmek gerekir. Bu ise tasarımcının hesaplamaya ayıracağı kaynaklar ile performans arasındaki dengeyi kurması sonucu tercihine bağlıdır. Ama yapılan çalışmalar resim tanımada frekansların gruplandırılması sonucu oluşturulan ağların çok başarılı sonuçlar verdiğini göstermektedir. Bugün endüstriyel uygulamalarda özellikle kalite kontrol yaparken ürünlerin üzerinde oluşan bozuklarının bulunmasında yapay sinir ağlarının aynı yöntemle başarılı bir şekilde kullanıldığı görülmektedir.

Bazı durumlarda problemin nümerik olarak gösterilmesi tamamen tasarımcının kontrolündedir. Girdilerin nümerik bilgiler ile hiç bir ilişkisi yoktur. Mesela, araba pistonlarının üretilmesi sırasında piston dökümü sonucu piston üzerinde bir eğim oluşmaktadır (buna sehim denmektedir). Bu eğimin önlenmesi mümkün değildir. Fakat en az sehim olacak şekilde bir döküm tasarlanması için ÇKA ağını kullanmak isteseniz nasıl bir girdi seti oluşturursunuz? (Bu örnek detaylı olarak ileride anlatılacaktır). Bu örnekte sehime neden olarak Tablo-5.5'te gösterilen faktörler ve değerleri belirlenmiştir.

Tablo-5.5. Örneklerdeki sayısal ve alfanümerik değerlere örnekler

Faktörler	Değer 1	Değer 2
A- Kalıpta soğutma süresi	70 sn	85 sn
B- Tüpün kalıptan çekilmesi	Tam	Yarım
C- Tırnakla kavrama ayarı	Ayarlı	Ayarsız
D- Sehpanın ısı iletimi	Boyalı	Boyasız
E- Dökümhane giriş kapısı	Açık	Kapalı
F- Kalıp boyama süresi	50 sn	40 sn
G- Eriyik sıcaklığı	1450 derece	1430 derece

Görüldüğü gibi bazı değerler tamamen alfanüremik değerlerdir. Bu olayı ÇKA ağına öğretmek için bunları nümerik değerlerle gösterilmek zorundadır. Bu değişik şekillerde yapılabilir. Mesela B faktörü için 1 tam değerini 0'da yarım değerini gösterebilir. Benzer şekilde Tam için 1, yarım içinde 2 değerleri de kullanılabilir. Seçilecek yöntem başarıyı etkileyecektir. O nedenle öğrenmenin gerçekleşmeme durumunda bu gösterim yöntemlerini değiştirmek başarılı çözümler üretilebilir.

Bazı durumlarda ise problem uzayının girdileri tamamen nümerik olmakla beraber örnek yapısının oluşturulması bazı düzenlemeler yapılması gerekmektedir. Mesela bir sonraki gün Amerikan dolarının kaç para olacağını tahmin eden bir ÇKA ağının oluşturulması durumunda girdi örnekleri nasıl olacaktır? Dolar kurunu geçen ay içerisinde aşağıda gösterildiği gibi bir seyir izlediğini kabul edelim.

($ x 10^6)

1	2	3	4	5	6	7	8	9	10
1.43	1.45	1.47	1.48	1.45	1.50	1.51	1.52	1.55	1.49

11	12	13	14	15	16	17	18	19	20
1.52	1.55	1.54	1.58	1.57	1.60	1.61	1.62	1.59	1.62

21	22	23	24	25	26	27	28	29	210
1.63	1.65	1.67	1.64	1.66	1.68	1.69	1.70	1.71	1.71

Benzer şekilde $ kurlarını istenildiği kadar geriye giderek belirlemek söz konusu olabilmektedir. Mesela, TC Merkez Bankasını web sayfasından istenildiği kadar bilgiye ulaşmak mümkün olabilmektedir. Bu durumda oluşturulacak olan ağın kaç tane girdisi olacaktır? Birkaç durum söz konusudur.

A) Ağın 30 tane girdisi olur. Her ay için 30 günlük döviz kurları girdilerin değerlerini oluşturur. Her yıl için toplam 12 adet örnek olacak demektir. Örneğin girdileri günlük döviz kurları çıktısı ise aylık ortalama döviz kuru değeri alınır. Böylece $ kurundaki değişimin aylık etkilenmelerden oluştuğu varsayılarak oluşturulacak ağın günlük etkilenmeleri öğrenmesi zorlaşır.

B) 10 günlük girdiler alınarak örnekleri oluştururken sürekli yeni bir değer (yeni bir günün kuru) eklenip başlangıçtaki değer çıkartılır. Bu istenirse 20 veya 30 gün içinde yapılabilir. Böylece her yeni günün $ kurundaki değişiminin etkisini örnek setine katarak ileri doğru $ kurunun değerinin değişiminin bir resmi çizilmeye çalışılır. Oluşturulacak olan ağın da bunu öğrenmesi beklenir. Yukarıdaki verilen değerler için bu mantıkla oluşturulacak ve 10 adet girdiden oluşacak olan örneklerin ilk 5 tanesi söyle olacaktır.

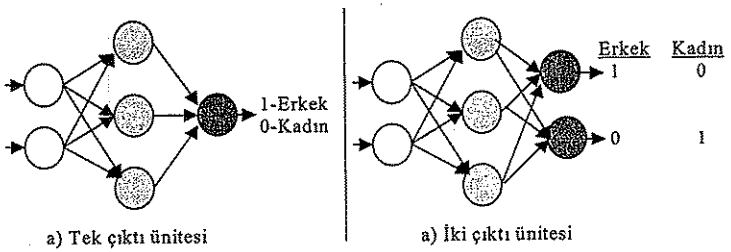
Örnek 1: 1.43, 1.45, 1.47, 1.48, 1.45, 1.50, 1.51, 1.52, 1.55, 1.49
Örnek 2: 1.45, 1.47, 1.48, 1.45, 1.50, 1.51, 1.52, 1.55, 1.49, 1.52
Örnek 3: 1.47, 1.48, 1.45, 1.50, 1.51, 1.52, 1.55, 1.49, 1.52, 1.55
Örnek 4: 1.48, 1.45, 1.50, 1.51, 1.52, 1.55, 1.49, 1.52, 1.55, 1.54
Örnek 5: 1.45, 1.50, 1.51, 1.52, 1.55, 1.49, 1.52, 1.55, 1.54, 1.58

Bu örnekler incelendiğinde ikinci örnekte birinci örneğin ilk değerinin (1.43) atıldığı ve birinci örneğin ikinci değerinin (1.45) ikinci örneğin birinci değeri olduğuna dikkat ediniz. İkinci örneğin en sonuna ise yeni bir $ kuru değerinin [1.52) eklendiğine de dikkat ediniz. Benzer şekilde üçüncü örnekte de ikinci örneğin birinci değeri (1.45) atılıp en sonuna 1.55 değeri eklenmiştir. Bu şekilde yukarıda verilen bir aylık $ kuru değerlerini kullanarak 21 adet örnek üretmek mümkündür.

Yukarıda verilen değişik örneklerden görüldüğü gibi bir ÇKA ağının girdi örneklerini belirlerken değişik gösterimlerin kullanılması mümkündür. ÇKA ağlarının herhangi bir olayı öğrenemediği zaman suçu sadece ağa bağlayıp vazgeçmek yerine girdilerin gösterimlerinde de değişik yöntemler deneyerek çözümler aramak lazımdır. Girdiler kadar çıktılarında nümerik gösterimi de performansı etkileyeceği unutulmamalıdır.

5.9.2.2. Çıktıların Nümerik Gösterimi

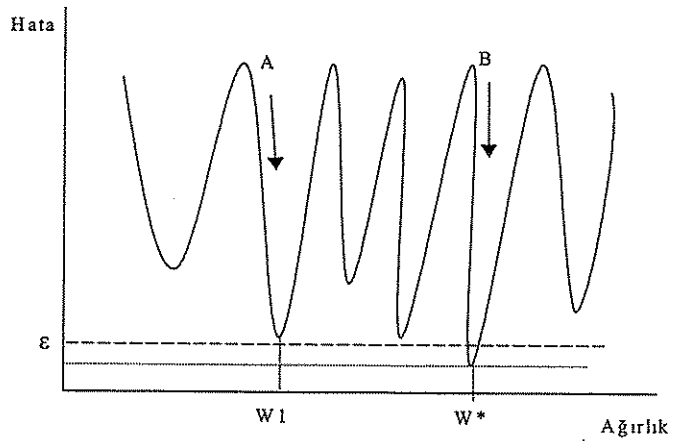
Çıktıların nümerik gösterimi gerçekleştirilmez ise çıktı değerleri ile beklenen değerler arasındaki hatayı bulmak mümkün olmaz. Girdilerde olduğu gibi çıktılarda da nümerik gösterim problemden probleme değişmektedir. Bir problem için birden fazla yöntem kullanarak nümerik gösterim sağlanabilir. Bunların en iyisinin hangisi olduğu bilinmemektedir. Önemli olan uygun olanı bulmaktır. Öğrenemeyen ağlarda tasarımcılar çoğu zaman bu konuyu da göz ardı etmektedirler. Mesela, daha önce verilen kadın ve erkek resimlerini birbirinden ayıran bir ağın çıktısı nasıl olacaktır? Tasarımcı ister ise bir tek çıktı yaparak gösterilen resmin erkek resmi olması durumunda 1 değerini, kadın resmi olması durumunda 0 değerini almasını isteyebilir. Alternatif olarak tasarımcı 2 çıktı ünitesi belirleyerek erkek resimleri için çıktının 1 0 kadın resimleri için ise 0 1 olmasını isteyebilir. İkinci durumda tanıma işleminin sorumluluğu çıktı üniteleri arasında paylaştırılmış olacaktır. Şekil-5.10 bu iki durumu göstermektedir. Bunlardan hangisinin daha iyi olduğunu söylemek mümkün değildir. Bu eğitimin başlaması ile görülebilir. Bazı durumlarda çıktıların değeri nümerik olsa bile onlarında ikili (*binary*) gösterimin kullanılmasını uygun görenler vardır. Örneğinin çıktının 2 olması durum 4 proses elemanı kullanılarak 0 0 1 0 şeklinde gösterilebilir. Problemin durumuna göre çıktıların gösterimini belirlemek gerekir. Performans üzerindeki etkisini unutmamak lazımdır.



Şekil-5.10. Çıktıların sayısal gösterimi

5.9.3. Başlangıç Değerlerinin Atanması

ÇKA ağının proses elemanlarını birbirine bağlayan bağlantıların ağırlıklarının başlangıç değerlerinin atanması da ağın performansı ile yakından ilgilidir. Genel olarak ağırlıklar belirli aralıklarda atanmaktadır. Bu aralık eğer büyük tutulursa ağın yerel çözümler arasında sürekli dolaştığı küçük olması durumunda ise öğrenmenin geç gerçekleştiği görülmüştür. Bu değerlerin atanması için henüz belirlenmiş standart bir yöntem yoktur. Ağırlıkların başlangıç değerlerinin rasgele atanmaları istenmektedir. Tecrübeler–1.0 ile 0.1 arasındaki değerlerin başarılı sonuçlar ürettiğini göstermektedir. Fakat bu tamamen öğrenilmesi istenen problemin niteliğine bağlıdır. Böyle bir genelleme yapmak doğru olmaz. Sadece ÇKA ağı kullanıcılarının bunu bilmeleri ve tecrübelerden faydalanarak denemeleri yapmaları için burada belirtilmektedir. Bu değerler arasında ağırlıkları atayıp hiç öğrenemeyen ağlar da olabilir. Çünkü ağın öğrenmesi daha bir çok parametreye ve öğreneceği olayın karmaşıklığına göre değişmektedir. Özellikle girdi/çıktı arasındaki ilişkinin belirlenmesinin zor olduğu durumlarda öğrenme de güçleşmekte hatta başarısızlıkla sonuçlanabilmektedir. Başlangıç değerleri problemin özünün aranmasına başlandığı noktaları göstermektedir. Öğrenemeyen bir ağın başlangıç değerlerinin değiştirilmesi ağın öğrenmesine neden olabilir. Bazı durumlarda ağın öğrenmesi zor bir olay üzerinde eğitildiğinde, ağın öğrenmesi başlangıç noktalarının değiştirilmesi ile de mümkün olmayabilir. O nedenle her olumsuz eğitimde, başlangıç değerlerini de suçlamamak lazımdır. Diğerleri ile birlikte düşünmek gerekir. Şekil-5.11 başlangıç noktalarının önemini göstermektedir.



Şekil-5.11. ÇKA ağlarında başlangıç noktasının etkisi

Şekilde görüldüğü gibi eğer bir ÇKA ağı öğrenmeye A noktasından başlar ise yerel bir çözüme (W1) takılabilme olasılığı var iken B noktasından başlarsa en iyi çözümü (W*) bulunması daha kolay olmaktadır.

5.9.4. Öğrenme Katsayısı ve Momentum Katsayılarının Belirlenmesi

Başlangıç değerleri kadar öğrenme ve momentum katsayılarının belirlenmesi de ağın öğrenme performansı ile yakından ilgilidir. Öğrenme katsayısı ağırlıkların değişim miktarını belirlemektedir. Eğer büyük değerler seçilirse o zaman yerel çözümler arasında ağın dolaşması ve osilasyon yaşaması söz konusu olmaktadır. Küçük değerler seçilmesi ise öğrenme zamanını artırmaktadır. Tecrübeler genellikle 0.2- 0.4 arasındaki değerlerin kullanıldığını göstermektedir. Fakat bu tamamen ilgili probleme bağlıdır. Bu değerler iyidir demek de doğru olmaz. Bazı uygulamalarda öğrenme katsayısının 0.6 değerini aldığı zaman en başarılı sonuçları verdiği görülmektedir.

Benzer şekilde momentum katsayısı da öğrenmenin performansını etkiler. Momentum katsayısı bir önceki iterasyondaki değişimin belirli bir oranının yeni değişim miktarına eklenmesi olarak görülmektedir. Bu özellikle yerel çözümlere takılan ağların bir sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacı ile önerilmiştir. Bu değerin küçük olması yerel çözümlerden kurtulmayı zorlaştırabilir. Çok büyük değerler ise tek bir çözüme ulaşmada sorunlar yaşanabilir. Tecrübeler bu değerin 0.6-0.8 arasında seçilmesinin uygun olacağını göstermektedir. Fakat bu da kesin denilemez. Problemın niteliğine göre kullanıcının belirlemesinde fayda vardır. Daha küçük değerler ile başarılı sonuçların alındığını gösteren örnekleri görmek de mümkündür.

5.9.5. Örneklerin Ağa Sunulması Şekli

Örneklerin ağa sunulma şekli de öğrenme performansını etkileyebilir. Genel olarak örnekler ağa iki türlü sunulabilirler. Bunlar:

- Sıralı sunum
- Rasgele sunum

Sıralı sunumda örnek setindeki birinci örnek ağa sunulur. Bir sonraki iterasyonda ise sırası ile ikinci, üçüncü sırası ile en sonuncu örneğe örneklerin tamamı ağa sunulur. Sonra tekrar başa dönerek örnek setindeki örnekler tek tek sıra ile ağa tekrar sunulur. Bu işlem öğrenme sağlanıncaya kadar devam eder. Bu tür bir sunuşta örnek setindeki bütün örneklerin ağa gösterilme şansları eşittir.

Rasgele sunumda ise örnekler eğitim seti içinden rasgele seçilirler. Burada da iki durum söz konusudur.

- Seçilen bir örnek tekrar set içine atılıp rasgele yeniden seçim yapılır. Bu durumda bir örneğin peş peşe birden fazla defa seçilme şansı vardır. Öğrenme gerçekleşene kadar böyle devam edilir. Örneklerin ağa gösterilme şansları eşit değildir.
- Rasgele seçilen örnek eğitim içine tekrar atılmaz. Kalanlar arasından rasgele tekrar yeni örnek seçilerek ağa sunulur. Bütün örnekler ağa gösterilince, eğitim seti tekrar içinden rasgele örnekler seçilerek ağa gösterilir. Bir gösterilen örnek bütün set ağa gösterilinceye kadar bekler. Öğrenme sağlanıncaya kadar bu işlem aynı şekilde tekrar eder. Örneklerin ağa gösterilme şansları bu durumda da eşittir.

5.9.6. Ağırlıkların Değiştirilmesi Zamanı

Ağırlıkların değiştirilmesi öğrenme kuralına göre yapılmaktadır. Genel olarak 3 durumda ağırlıkların değiştirilmesine izin verilmektedir. Problemin durumuna göre ağırlıkların ne zaman değiştirileceğine karar vermek gerekir. Doğru zamanlama ağın, öğrenme performansını etkilemektedir. Bu 3 durum şöyle özetlenebilir.

1. *Her örnek ağa gösterildiğinde (pattern based learning):* Bu durumda ağa her örnek gösterildiğinde beklenen çıktı ile ağın gerçekleştirdiği çıktı arasındaki hata bulunur ve bu hata ağın ağırlıklarına daha önce anlatılan öğrenme kuralı gereğince dağıtılır. İkinci örnek ağa sunulduğunda çıktının hatası hesaplanır ve ağırlıklar değiştirilir. Her örnek gösterimi sonucu ağırlıklar değiştirilir.
2. *Belirli sayıda örnek gösterildiğinde (batch based learning):* Bu durumda ağa her örnek gösterildiğinde hatası hesaplanıp ağırlıklar değiştirilmez. Belirli sayıda örnek tek tek ağa gösterilir ve hatalar toplanırlar. İstenen sayıdaki örneğin ağa gösterilmesi bitince toplanan hata ağırlıklara dağıtılır. Genellikle 5-10 örnekten oluşan örnek grupları oluşturulmaktadır. Yani 5 örnek peş peşe ağa gösterilmekte hatalar hesaplanıp toplanmakta ve toplam hata öğrenme kuralına göre ağırlıklara dağıtılmaktadır. Aynı işlemler her örnek grubundaki örneklerin tamamı ağa gösterildikçe tekrarlanmaktadır.

3. *Bütün örnek seti gösterildiğinde (epoch based learning)*: Bu durumda örnek setindeki bütün örnekler ağa tek tek gösterilir. Hatalar hesaplanır ve eğitim setindeki örneklerin tamamının hataları toplandıktan sonra bu hata ağırlıklara dağıtılır. Yani; ağın ağırlık değerleri örneklerin tamamı ağa gösterilmedikçe değiştirilmez. Örnek sayısının az olduğu durumlarda önerilmektedir.

5.9.7. Örneklerin Değerlerinin Ölçeklendirilmesi (*Scaling*)

ÇKA ağlarında girdi ve çıktıların ölçeklendirilmeside ağın performansını yakından etkilemektedir. Çünkü ölçeklendirme örneklerin değerlerinin dağılımını düzenli hale getirmektedir.

5.9.7.1. Girdilerin Ölçeklendirilmesi

Problemin örnekleri toplanırken; bazı durumlarda problem uzayı ile ilgili örnekler farklı ölçekler kullanan ortamlardan toplanmış olabilir. Hepsinin aynı ölçek üzerine indirgenmesi gerekebilir. Bazı durumlarda da problemin girdileri arasında aşırı büyük veya küçük değerler görülebilir. Bunlar yanlışlık sonucu girdi setine girmiş olabilir. NET girdiler hesaplanırken bu değerler aşırı büyük veya küçük değerlerin doğmasına neden olarak ağı yanlış yönlendirebilirler. Bütün girdilerin belirli aralıkta (çoğunlukla 0-1 aralığında) ölçeklendirilmesi hem farklı ortamlardan gelen bilgilerin aynı ölçek üzerine indirgenmesine hem de yanlış girilen çok büyük ve küçük şekildeki değerlerin etkisinin ortadan kalkmasına neden olur. Çünkü bu durumda olası en büyük değer 1 değerini almakta ve ondan büyük değerlerde öğrenme setine 1 değerini alarak girmektedir. En küçük değere ise 0 değeri verilmekte ondan küçük değerlerde yine öğrenme setine 0 değerini alarak girmektedir. Ölçeklendirme değişik şekillerde yapılmaktadır. Bazı araştırıcılar girdi vektörünü normalize etmektedirler. Yani her değeri girdi vektörünün değerine bölerek yeni değerleri bulmaktadırlar. Bu ise şu şekilde verilmektedir.

$$x' = \frac{x}{|X|}$$

Burada x girdi değerini, x' ölçeklendirilmiş yeni girdi değerini, $|X|$ ise girdi vektörünün büyüklük (vektörel) değerini göstermektedir.

Bazı araştırmacılar ise aşağıdaki gibi bir formülasyon kullanmakta ve örnekleri oluşturan değerleri belirli bir aralık içine çekmektedirler.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Burada x girdi değerini, x' girdi değerinin ölçeklendirilmiş halini, x_{min} girdi setindeki olası en küçük değeri x_{max} ise girdi setindeki olası en büyük değeri göstermektedir.

Bazı araştırmacılar ise kendi problemlerine özgü ölçeklendirme yöntemleri geliştirmektedir. Burada önemli olan hangi yöntem kullanıldığından çok girdiler içindeki olumsuz etkileri önleyecek şekilde ölçeklendirme yapmaktır. Her problem için farklı

bir ölçeklendirme yöntemi kullanılabilir. ÇKA ağı tasarımcıları ellerindeki verileri normalize edecek bir yaklaşımı kendileri belirleyebilirler. Bu konuda bir standart koymak doğruda olmaz.

5.9.7.1.1. Çıktıların Ölçeklendirilmesi

ÇKA ağlarında genel olarak *sigmoid* aktivasyon fonksiyonunun kullanıldığı daha önce söylenmişti. Bu fonksiyonun özelliklerinden biriside sadece 0 ve 1 arasında bir değer üretmesinin mümkün olmasıdır. Eğer çıktı değerlerinin 1'den büyük veya 0'dan küçük olması isteniyor ise o zaman *sigmoid* fonksiyonundan faydalanmak mümkün olmaz. Problemin çözümü için sigmoid fonksiyonu kullanmak zorunluluğu veya isteği var ise o zaman beklenen çıktıların 0-1 arasına indirgenmesi ve çıktılar üretildikten sonra orijinal değerlerine dönüştürülmeleri gerekmektedir. Çıktıların ölçeklendirilmesi de yukarıda girdilerin ölçeklendirilmesi için anlatılan yöntemlerin birisi ile yapılabilir. Ağ öğrenme yaptıktan sonra da ölçeklendirilmiş çıktılar üreteceğinden, ağın çıktılarının dış dünyaya verilirken orijinal şekillerine dönüştürülmesi gerekir. Bunun için ölçeklendirme formülünü tersine işletmek lazımdır. Mesela Yukarıda verilen ikinci formül ile ölçeklendirme yapılmış ise orijinal değerlere dönüştürme için o formül tersine çevrilerek şu şekilde kullanılacaktır.

$$x = x'(x_{max} - x_{min}) + x_{min}$$

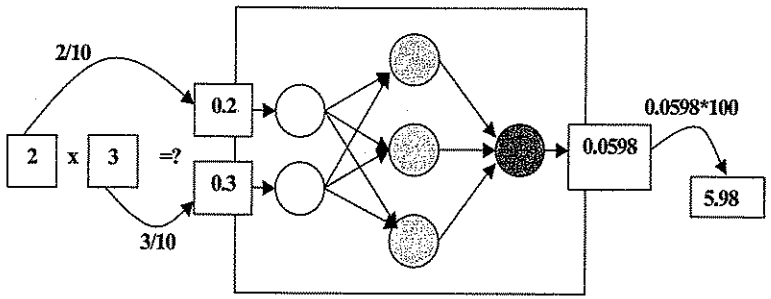
Böylece kullanıcı ağın ürettiği çıktıların sıfırdan küçük veya birden büyük değerler olması sağlanacaktır.

Girdi ve çıktıların ölçeklendirilmesni iyi anlamak için daha önce anlatılan 2 ile çarpma örneğine bakmakta fayda olabilir. Bu problemde ara katman ve çıktı ünitesinde *sigmoid* fonksiyonu kullanılmıştır. Bu ağın çıktısının 1'den büyük bir değer olamaması demektir. 2*4=8 olduğuna göre, ağ 8 sonucunu nasıl üretecektir? Bu amaçla tasarlanan ve eğitilen ağın 2*3=5.98 sonucunu ürettiği söylenmiştir. Bu nasıl olmaktadır? Ağın 1'den büyük değerler üretemeyeceği bilindiğinden hem girdiler hem de çıktılar ölçeklendirilmiştir. Bu ölçeklendirmede girdi setindeki her değer 10 ile bölünerek örnekler oluşturulmuştur. Daha sonra bunların çarpım değerleri ağın beklenen değerleri olmuştur. Ağın ürettiği çıktılar ise 100 ile çarpılarak dış dünyaya gönderilmiştir. Bu problemdeki örnek setinin orijinal değerleri ve ölçeklendirilmiş değerleri Tablo-5.6'da verilmiştir.

Tablo-5.6. İki'li çarpma setinin ölçeklendirilmesi

Orijinal Eğitim Seti		Ölçeklendirilmiş Eğitim Seti	
Girdiler	Çıktı	Girdiler	Çıktı
2 * 2	4	0.2 * 0.2	0.004
2 * 4	8	0.2 * 0.4	0.008
2 * 6	12	0.2 * 0.6	0.012
2 * 8	16	0.2 * 0.8	0.016
2 * 10	20	0.2 * 1.0	0.020

Oluşturulacak olan ÇKA ağına girdi/beklenen çıktı örneği olarak ölçeklendirilmiş örnekler sunulmaktadır. Şekil-5.12 bu durumu göstermektedir.



Şekil-5.12. ÇKA ağına girdilerin ölçeklendirilerek gönderilmesi

Şekilde görüldüğü gibi ağ eğitildikten sonra ağa 2*3=? sorusu 0.2*0.3=? şeklinde sorulmakta ve ağda buna cevap olarak 0.0598 değerini üretmektedir. Daha sonra bu değer 100 ile çarpılarak ağın 2 *3 =5.98 değerini ürettiği söylenmektedir.

5.9.8. Durdurma Kriterleri

ÇKA modelinde ağın eğitilmesi kadar gereğinden fazla eğitilmemesi de önemlidir. Çünkü eğitilmek istenen bir ağ problem uzayına çözüm üretecek ağırlıkları bulduktan sonra eğitime devam edilirse bu ağın ağırlıklarında daha fazla değişikliklere neden olur. Böylece en iyi çözüm üreten bir ağ tekrar daha performansı düşük ağlara veya öğrenemeyen ağlara dönüşebilir. O nedenle ağın eğitiminin ne zaman durdurulması gerektiği konusunda da karar vermek gerekmektedir. Bu da ağın başarısını ve performansını etkilemektedir. Pratikte, genel olarak iki türlü durdurma kriteri kullanılmaktadır.

- *Hatanın belirli bir değerin altına düşmesi halinde eğitimi durdurma:* Bu durumda hatanın bütün eğitim seti için kabul edilebilir bir değerin altına düşmesi kriter olarak alınmaktadır. Burada bir konuya dikkat etmek gerekir. Bazı durumlarda sadece bir iki örnek için hata değerleri çok küçük seviyelere düşmekte bazı örnekler için ise kabul edilme düzeyde kalmaktadır. Bu durumda eğitime devam etmek gerekir. Eğer eğitim durdurulursa, o zaman diğer örneklerde öğrenme olayı gerçekleşmemiş olur. Pratikte bütün örneklerin hatalarının ortalama değerinin belirli bir değerin altına düşmesi de bu konuda uygulamada görülebilmektedir. En sağlıklısı her örnek için hatayı düşürmektir. Bazen bütün örnekleri ağın öğrenmesi mümkün olmayabilir. O zaman tasarımcı bu örnekleri eğitim setinden çıkartabilir veya onların öğrenememesine katlanır. Yani öğrenme performansı %100 olmaz da %98-99 gibi bir düzeyde kalabilir. Bu kabul edilebilir ise bir sorun yok demektir. Kabul edilebilir hatanın miktarı ise problemden probleme değişebilir. ÇKA tasarımcısı bunu kendisi belirler. Bazı örnekler için %5 kabul

edilemez bir hata oranı iken (özellikle insan hayatını ilgilendiren konularda) bazı örnekler için %15-20 hata dahi kabul edilebilir nitelikte olabilmektedir.

- *Ağın belirli bir iterasyon sayısını tamamlaması sonucu eğitimi durdurma:* Bu durumda iterasyon sayısının belirlenmesi önemli olur. Bir kaç deneme yaptıktan ve hata grafiklerini inceledikten sonra ağın eğitilmesi için gereken eğitim iterasyon sayısı saptanabilir. Özellikle hatanın hangi değerlerin altına düşebileceğinin kestirilemediği (yani kabul edilebilir hatanın belirlenemediği) durumlarda bu tür bir durdurma kriteri uygulanabilir. .

5.9.9. Ara Katman Sayısı ve Proses Elemanlarının Sayısının Belirlenmesi

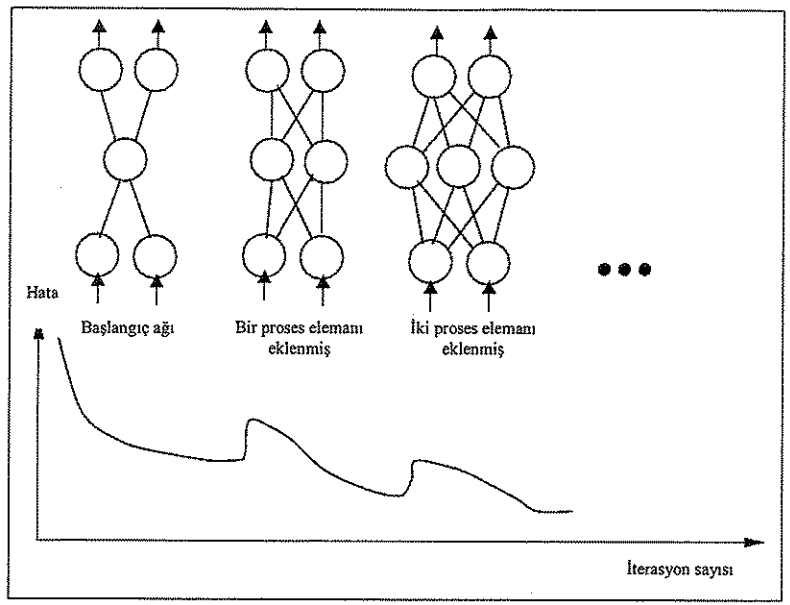
ÇKA modelinde herhangi bir problem için kaç tane ara katman ve her ara katmanda kaç tane proses elemanı kullanılması gerektiğini belirten bir yöntem şu ana kadar bulunmuş değildir. Eğer girdilerin hepsi ikili (*binary*) olursa o zaman bazı yöntemler önerilmekle birlikte bu konudaki çalışmalar deneme yanılma yönteminin etkin olarak kullanıldığını göstermektedir. Ara katman sayısı ve proses elemanı sayları da ağın performansını yakından ilgilendirmektedirler. Tasarımcılar kendi tecrübelerine dayanarak bunları belirler. O nedenle bir problem herhangi bir ağ ile kabul edilebilir hata altında çözüm üretse bile daha iyi bir ağ olur mu diye farklı sayıdaki ara katman ve her ara katmanda farklı sayıda proses elemanları ile denemeler yapmak gerekir. Böylece performansı daha yüksek bir ağ bulmak mümkün olabilir. Bazı durumlarda başlangıçta bir ağ oluşturulup zaman içinde büyütülerek veya küçültülerek istenen ağa ulaşılır. Aşağıda bu durum açıklanmıştır.

5.10. Ağların Büyütülmesi veya Budanması

ÇKA ağlarında problemin çözümü için gerekli en iyi topolojiyi belirlemek mümkün olmadığından deneme yanılma yöntemi kullanılmakta bazen eksik sayıda bazen de fazla sayıda proses elemanı kullanılmaktadır. Gereken sayıda proses elemanı belirlemek için iki yoldan birisi denenmektedir. Bunlar:

- Küçük bir ağdan başlayıp büyük bir ağa doğru eğitim esnasında sürekli proses elaman sayısını artırmak. Bu durum Şekil-5.13'de gösterildiği gibi gerçekleştirilir.
- Büyük bir ağdan başlayıp küçük bir ağa doğru eğitim sırasında sürekli ağı küçültmek ve proses elemanlarını teker teker ağdan çıkartmak. Buna ağın budanması denmektedir.

Şekil-5.13'de büyüyen bir ağ görülmektedir. Başlangıçta 1 ara katman elemanı ile ağın eğitimine başlanmış ve ağın öğrenme hatası belirli bir değere kadar düşürülmüştür. Ağın daha fazla öğrenemeyeceği görüldüğünde, ağa ikinci bir ara katman elemanı eklenmiş ve eğitime devam edilmiştir. Eğitim performansı biraz daha iyileştirilmiştir. Daha sonra 3. eleman eklenmiştir. Bu işlem örneklerin tamamı öğrenilinceye veya tasarımcı öğrenme performansından memnun oluncaya kadar devam edilmekte ve problem için gerekli ara katman proses elemanı sayısı bulunmaktadır.



Şekil 5.13. Büyüyen ağlar

Ağın budanmasında ise tam tersi bir yaklaşım izlenmektedir. Mesela ağın eğitimine ara katmanda 50 adet proses elemanı ile başlanılmakta ve ağ eğitilmektedir. Ağ bu durumda örnekleri öğrenince arak katmanda bulunan elemanlardan bir tanesi eksiltilerek eğitime devam edilmektedir. Eğer ağ bu durumda da örnekleri öğrenebiliyor ise o zaman, ara katmandan bir eleman daha eksiltilir. Bu işlemler ağ öğrenmeyi başardığı sürece devam etmekte ve ağın öğrenemediği nokta bulunduğunda ara katmanda kaç proses elemanı var ise o olay için oluşturulacak ağın topolojisi ona göre belirlenmektedir.

5.11. ÇKA Ağının Uygulama Alanları

ÇKA ağları hayatın hemen hemen her alanında örnekleri görülen bir modeldir. Günümüzde uygulamaların sayısını dahi bilmek mümkün değildir. Genel olarak;

- sınıflandırma
- tahmin etme
- tanıma
- yorumlama
- teşhis etme

problemlerinde başarı ile kullanılmaktadır. Özellikle tıp alanında elektrokardiyografilerin anlaşılması gibi gibi şekillerin hastaların kalp atışlarının yorumlanması örneklerinde de başarılı bir şekilde kullanıldığı görülmektedir.

Değişik alanlarda görülen ÇKA ağının uygulamalarına bazı örnekler vermek gerekirse;

Yatırımların Planlanmasında

Şu ana kadar bu tür problemlerin çözülmesinde daha çok doğrusal tahmin yöntemleri kullanılır iken yapay sinir ağları ile borsadaki bazı şirketlerin durumları takip edilmektedir. Geleceğe yönelik tahminler yapılmasında yapay sinir ağlarından, özellikle ÇKA ağlarından faydalanılmaktadır.

İmza Analizinde

Özellikle bankalarda bulunan müşterilerin imzaları ile yeni işlemlerde karşılaşılan imzaların karşılaştırılmasında özellikle ABD'de yaygın olarak ÇKA ağlarının kullanıldıkları belirtilmektedir. Çünkü ÇKA ağları otomatik hesap işlemlerini yaygın olarak gerçekleştirmeyi sağlamaktadır. Bu konuda Yapay sinir ağları donanımlarının bile gerçekleştirildiği ve ilk yapay sinir ağı *chip*'inin bu amaçla kullanıldığı rapor edilmiştir. [3]

Proses Kontrolünde

ÇKA ağları proseslerin kalite kontrolünde yaygın olarak kullanılmakta ve prosesin çalışması sırasında davranışlarını sürekli gözetleyerek anormal durumların olması halinde operatörleri uyarabilmektedir.

Makinelerin İzlenmesinde

Bu konuda da çok sayıda örnek vermek mümkündür. Mesela, ÇKA ağları, uçakların motorlarının davranışlarının izlenmesinde titreşim ve ses analizleri yaparak durumlarından olası hataları erkenden haber verebilmektedir. Benzer şekilde tren motorlarının izlenmesinde de kullanıldıkları bilinmektedir. imalatta makinelerin hata teşhisinde kullanılmakta oldukları rapor edilmektedir.

Pazarlama

ÇKA ağları pazarlama amacı ile ürünlerin satış trendlerinden tüketici davranışlarına yönelik bilgiler elde edilerek pazarlama stratejileri geliştirmeye yardımcı olacak tahminlerin yapılmasında kullanılmaktadır.

İmalat Sektöründe

Bu sektörde de ÇKA ağlarının uygulamalarına sayısız örnek vermek mümkündür. İşlerin makinelere çizelgelenmesinden kalite kontrole kadar bir çok alanda örnekler görülmektedir.

Kitabın en son bölümünde yapay sinir ağlarının değişik uygulamaları tartışılacaktır. Aşağıda bir ÇKA ağının bir endüstriyel uygulaması bir örnek endüstriyel uygulaması detaylı olarak açıklanmıştır. Bu örnek ÇKA ağının oluşturulması ve eğitilmesi konusunda önemli bilgiler içermektedir.

5.12. Çok Katmanlı Algılayıcı Bir Örnek Uygulama (Endüstriyel Uygulama)

Yukarıda yapay sinir ağlarının özellikle de ÇKA ağlarının mühendislik problemlerinin bir çoğuna çözümler ürettiği belirtilmiştir. Bu uygulamaların gerçekleştirilmesinde ağların belirlenip eğitilmesine kadar her konuda bilgiler verilmiştir. Her hangi bir spesifik endüstriyel problem için yukarıda belirtilen konuların açıklanmasının tasarımcılar açısından faydası olacağı düşüncesi ile aşağıda bir ÇKA ağının problemi nasıl çözdüğü anlatılmıştır.

Günümüz imalat sektöründe Kalitenin tasarım sırasında düşünülmesine yönelik çalışmalar oldukça yaygın olarak yürütülmektedir. Özellikle `Taguchi` [4] tarafından geliştirilen tasarım yöntemi yaygın olarak kullanılmaktadır. Bu bölümde anlatılan örnek ÇKA ağı uygulaması, `Taguchi` yaklaşımına (kalitenin üretim öncesinde tasarlanmasını sağlayan yöntem) alternatif olarak kullanılabilecek ve `Taguchi` metodunun olumsuz yönlerini ortadan kaldıracak bir yaklaşım olarak görülebilir.

5.12.1. Problemin Tanımlanması

Burada anlatılan örnek, piston üreten bir fabrikada piston kalitesinin artırılmasına yönelik çalışmadır. Piston üreten bir firmada proses dökümden başlar ve son ürüne kadar devam eder. Prosesin döküm bölümünde piston yapılmak üzere uzun borular halinde tüpler dökülmektedir. Bu tüpler 8 gömlek olabilecek uzunluktadır. Döküm savurma tekniği ile yapılmaktadır. Döküm işlemleri esnasında yöntem gereği tüplerin dış yüzeyleri ısı iletkenliği çok az olan bir boya ile boyanmaktadır. Döküm yapılıp söküldükten sonra tüplerin dış yüzeylerindeki boya temizlenmektedir. Ancak bilinmeyen nedenlerden ötürü dökülen tüpler üzerinde bir eğim meydana gelmektedir. Bazı tüplerin dış yüzeyleri bu nedenle yeterli oranda temizlenememektedir. Oluşan bu eğime ***sehim*** denmektedir. *Sehimin* önlenmesi için ona neden olan etkin faktörlerin kontrol altında tutulması gerekmektedir.

Yapılan araştırma ve incelemeler neticesinde tüplerde meydana gelen *sehime* etki eden faktörler belirlenmiş ve aşağıda verilmiştir. Tablo-5.7'de gösterildiği gibi her faktörün iki değerinin olması söz konusudur. Yapay sinir ağlarını kullanarak bu 7 faktör için *sehimi* enazlayacak üretim kombinasyonu belirlenecektir. Aslında 2^7 adet yani 128 deney yapılması sonucu en iyi kombinasyonu bulmak mümkündür. Fakat bu deneyleri yapmak maliyet açısından oldukça fazladır. Yapay sinir ağları ise yapılan 8 deneyin sonuçlarını alarak en iyi kombinasyonu bulmaya çalışacaktır. Dahası faktör sayısı ve her faktörün alacağı değerler arttıkça deneylerin hepsini yapmakta imkansızlaşacaktır.

Tablo-5.7. Piston üzerindeki *sehime* neden olan faktörler ve değerleri

Faktörler	Değer 1	Değer 2
A- Kalıpta soğutma süresi	70 sn	85 sn
B- Tüpün kalıptan çekilmesi	Tam	Yarım
C- Tırnakla kavrama ayarı	Ayarlı	Ayarsız
D- Sehpanın ısı iletimi	Boyalı	Boyasız
E- Dökümhane giriş kapısı	Açık	Kapalı
F- Kalıp boyama süresi	50 sn	40 sn
G- Eriyik sıcaklığı	1450 derece	1430 derece

5.12.2. Öğrenme Setinin Oluşturulması

Daha önce belirtildiği gibi ÇKA ağlarında önemli olan problemi en iyi gösterecek örnek setini belirlemektir. Taguchi yaptığı analizler neticesinde geliştirdiği ortogonal dizilere dayanarak oluşturulan örneklerin problem uzayını temsil ettiğini göstermiştir [5].

Yapay sinir ağlarına da bu örnek için tasarlanmış L8 ortogonal dizisine bağlı yapılan deneyler öğrenme seti olarak alınmıştır. Bu diziye göre, sekiz deneyin yapılması gerekmektedir. Yani ÇKA ağı için problem uzayını temsil edebilen 8 örnek belirlenmiştir. L8 ortogonal dizine göre geliştirilmiş örnekler Tablo-5.8'de verilmiştir. Tablo-da her kombinasyon için yapılan deney sonucu oluşan *sehim* miktarı da gösterilmiştir. Her deney ve oluşan *sehim* miktarı bir örnek (girdi/çıktı seti) olarak düşünülmüştür.

Tablo-5.8. L8 Ortogonal dizisine göre belirlenmiş örnekler

Deney	A	B	C	D	E	F	G	Sehim
1	70	Tam	Ayarsız	Boyasız	Açık	50	1450	1.70
2	70	Tam	Ayarsız	Boyalı	Kapalı	40	1430	1.84
3	70	Yarım	Ayarlı	Boyasız	Açık	40	1430	1.60
4	70	Yarım	Ayarıl	Boyalı	Kapalı	50	1450	1.44
5	85	Tam	Ayarlı	Boyasız	Kapalı	50	1430	1.42
6	85	Tam	Ayarlı	Boyalı	Açık	40	1450	1.54
7	85	Yarım	Ayarsız	Boyasız	Kapalı	40	1450	1.22
8	85	Yarım	Ayarsız	Boyalı	Açık	50	1430	1.57

Görüldüğü gibi deneylerde kullanılan bütün değerler sayısal (nümerik) değildir. O nedenle ağın bunları anlaması için nümerik değerlere dönüştürülmesi gerekmektedir. Bu örnekte Her faktörün 1. değeri için 1 ikinci değeri için 2 rakamı o faktörün temsili değeri olarak seçilmiştir. O zaman, eğitim seti Tablo-5.9'daki gibi olacaktır:

Tablo-5.9. Örneklerin nümerik değerler ile gösterimi

Deney	A	B	C	D	E	F	G	*Sehim*
1	1	1	1	1	1	1	1	1.70
2	1	1	1	2	2	2	2	1.84
3	1	2	2	1	1	2	2	1.60
4	1	2	2	2	2	1	1	1.44
5	2	1	2	1	2	1	2	1.42
6	2	1	2	2	1	2	1	1.54
7	2	2	1	1	2	2	1	1.22
8	2	2	1	2	1	1	1	1.57

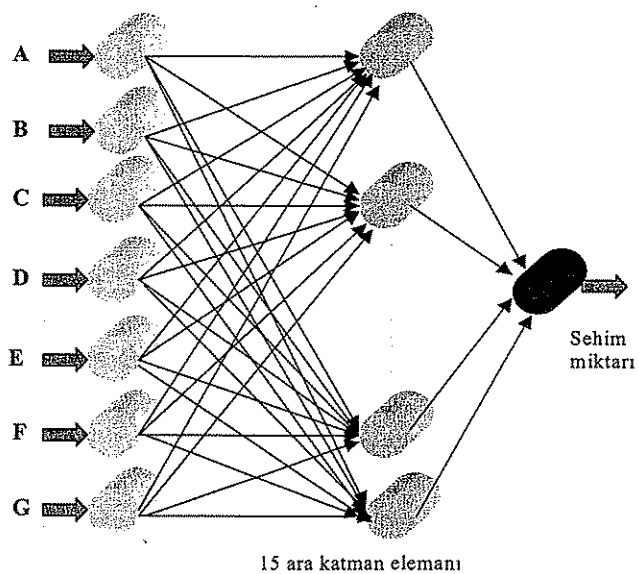
Problemin çözümünde aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılacağından sonuçlar 0-1 arasında çıkacaktır. Yani, ağın sehim miktarı olarak 1'den büyük bir değeri ve 0'dan küçük bir değeri üretmesi mümkün değildir. O nedenle ağın hem girdileri hem de çıktıları 10'a bölünmüş ve ölçeklendirilmiş eğitim seti elde edilmiştir. Bu değerler Tablo-5.10'da verilmiştir. Ağın eğitildikten sonraki çıktı değerleri daha sonra 10 ile çarpılarak tahmin edilen sehim miktarları belirlenmiştir.

Tablo-5.10. Ölçeklendirilmiş örnekler

Deney	A	B	C	D	E	F	G	Sehim
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.170
2	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.184
3	0.1	0.2	0.2	0.1	0.1	0.2	0.2	0.160
4	0.1	0.2	0.2	0.2	0.2	0.1	0.1	0.144
5	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.142
6	0.2	0.1	0.2	0.2	0.1	0.2	0.1	0.154
7	0.2	0.2	0.1	0.1	0.2	0.2	0.1	0.122
8	0.2	0.2	0.1	0.2	0.1	0.1	0.1	0.157

5.12.3. ÇKA Ağının Oluşturulması

Sehimin oluşmasına neden olan faktörlerin sayısı 7 olduğu için 7 giriş ünitesi ve *sehimin* miktarını tahmin eden bir çıktı ünitesi belirlenmiştir. Ara katmanda ise 15 üniteden oluşan bir ağ oluşturulmuştur. Bu ağın topolojisi Şekil-5.14'de verilmektedir.



Şekil-5.14. Önerilen ÇKA modeli

Ağın eğitilmeye başlamadan önce diğer parametreleri de Tablo-5.11'de gösterildiği gibi tanımlanmıştır.

Tablo 5.11. ÇKA ağının parametre değerleri

Parametre	Değeri
Öğrenme katsayısı	0.2
Momentum katsayısı	0.8
Başlangıç değerleri	-0.1 ile 0.1 arasında (rasgele)
Örnek gösterimi	Sıralı
İterasyon sayısı	2000

5.12.4. ÇKA Ağının Eğitilmesi

Ağın eğitilmesi yukarıda anlatılan öğrenme kuralına göre gerçekleştirilmiştir. İlk 1000 iterasyon sonunda ağın öğrenme seti üzerindeki bütün örnekleri 0.02'lik bir hata ile öğrendiği görülmüştür. 2000 iterasyon sonunda ise öğrenme seti üzerinde hiç hatasız öğrendiği görülmüştür. Ağın öğrenip öğrenmediğini test etmek amacı ile gerçek bir deney yapılmış ve onun çıktısı ile ağın çıktısı karşılaştırılmıştır. Daha sonra ağa

128 deneyin hepsi tek tek sorulmuştur. Ağ, $A_2\ B_2C_2\ D_1E_2F_2G_1$ şartlarında deneyin en iyi sonucu yani en az *sehimi* vereceğini göstermiştir. `Taguchi` ile yapılan çalışmada da bu çözüm en iyi üretim kombinasyonu olarak ortaya çıkmıştır. Ağ, bu şartlar altında üretim yapılırsa oluşacak *sehim* miktarının 1.173 olacağını tahmin etmiştir. Gerçekte bu şartlar altında yapılan deneme üretimde ise *sehimin* 1.17 olduğu görülmüştür. ÇKA modeli çözümü 0.003 kadar bir hata ile tahmin edebilmiştir. Aynı sonucun `Taguchi` ile tahmin edilmesi ise *sehimin* 1.228 olacağı tahmin edilmiştir. Buda yaklaşık 0.06'lık bir hatanın oluştuğunu göstermiştir. Bunun nedeni aşağıda tartışılmıştır.

5.12.5. Sonuçların Tartışılması

Piston gömleklerindeki eğilme (*sehim*) miktarını enazlamak için ÇKA modeli kurulmuş ve başarılı bir şekilde çözüm üretmiştir. Geleneksel yöntemler ile (*Taguchi* yöntemi) aynı problem çözülmüş aynı sonuçlar elde edilmesine rağmen geleneksel yöntemler %5 hata yaparken ÇKA ağı yaklaşık %0.25 gibi çok küçük bir hata ile doğru sonucu bulmuştur.

Bunun en önemli nedeni *Taguchi*'nin faktörler arasındaki ilişkiyi doğrusal kabul etmesidir. ÇKA için bu ilişkinin türü önemli değildir ve örneklerde kendisini gösterecektir. Doğrusal olan ilişki de doğrusal olmayan ilişkilerde örneklerden öğrenilebilir.

Ayrıca ÇKA ağı Taguchinin yakaladığı faktör etkilerini de yakalamıştır. Mesela F faktörü her iki yöntem tarafından da en önemsiz faktör olarak tespit edilmiştir. Öyle ki sadece F faktörünün değerlerinin değişmesi durumunda yapılan sehim miktarı tahminleri hemen hemen birbirinin aynıdır.

Taguchinin en etkin faktör olarak belirlediği A faktörünü ÇKA da en etkin faktör olarak belirlemiştir. Öyle ki, A_1 değerlerini alan örneklerin %80'i tahmin değerini 1.5 mm'den daha büyük verirken A_2 değerlerini alan örneklerin %80'i tahmini sehim değerini 1.5 mm altında vermektedir.

Bunlara ek olarak *Taguchi* faktörler arası ilişkilerin bilindiğini, eğer bilinmiyorsa yok olduğunu varsaymaktadır. Halbuki ÇKA modeli için bu yine örneklerde vardır. Etki varsa da yoksa da zaten örneklerde gizlidir. Böyle bir kabul yapılmasına gerek kalmamaktadır. Bu şu anlama gelir. Eğer faktörler arası ilişki varda bilinmiyorsa *Taguçhi*'nin başarı şansı zayıflayacaktır. ÇKA ağı ise böyle bir durumdan olumsuz etkilenmeyecektir. Dahası bu tür etkileşimler günlük hayatta kaçınılmazdır. Bu da ÇKA modelinin başarısının önemini göstermektedir. Mesela bu örnekte faktörler arası bir ilişki olmadığın ön görülmüştür. Fakat deney sonuçları incelendiğinde A ve B faktörleri ile A ve G faktörlerinin birbirlerini etkilediği belirlenmiştir. *Taguchi*'nin tahminindeki hatanın büyüklüğünün altında bunu görmemiş olması da yatabilir.

Yukarıdakilere ek olarak `tagucih` her faktörün belirli sayıda değer almasını istemektedir. Genellikle her faktörün 2 değeri olması gerektiğini bazı durumlarda 3 ve 4 olabileceğini belirtmektedir. Halbuki ÇKA modeli için her faktör istenildiği kadar farklı değerler alabilir. Eğer bu değerlerin temsil edildiği deneyler yapılırsa o zaman ÇKA modeli başarılı sonuçlar elde edebilir. Bu ise özellikle deneysel tasarım çalışmalarında oldukça önemli bir sorunu ortadan kaldırmış olacaktır.

Son olarak *Tagucih* metodunun faktör sayıları arttıkça başarısının düştüğü belirtilmektedir [6]. Önerilen ÇKA modeli için böyle bir sıkıntı söz konusu olmadığı gibi faktör sayılarının artmasında başarılı olması önemini bir kat daha artıracaktır. Ancak seçilecek örnek sayısının tüm problem uzayını kapsayacak şekilde seçilmesi gerekmektedir.

5.13. Özet

Yapay sinir ağlarının günümüzde en yaygın olarak kullanılan modeli çok katmanlı algılayıcı (ÇKA) ağlarıdır. Bu ağlar, özellikle mühendislik problemlerinin %95'ine çözüm üretebilecek nitelikte bir ağdır. ÇKA ağları XOR problemini çözebilmek için yapılan çalışmalar neticesinde ortaya çıkmıştır. Bu ağlar 3 katmandan oluşurlar

- ***Girdi katmanı***: Dış dünyadan bilgileri alır. Bu katmanda herhangi bir bilgi işleme olmaz.
- ***Ara katmanlar***: Girdi katmanından gelen bilgileri işlerler. Bir adet ara katman ile birçok problemi çözmek mümkündür. Eğer ağın öğrenmesi istenilen problemin girdi/çıktı arasındaki ilişkisi doğrusal olmaz ve karmaşıklık artarsa birden fazla sayıda ara katmanda kullanılabilir.
- ***Çıktı katmanı***: Ara katmandan gelen bilgileri işleyerek ağa girdi katmanından sunulan girdi için ağın ürettiği çıktıyı bulur. Bu çıktı dış dünyaya iletilir.

Girdi ve çıktı katmanlarında kaç tane proses elemanının olması gerektiğine probleme bakılarak karar verilir. Ara katman sayısı ve her ara katmandaki proses elemanı sayısının kaç olması gerektiğini gösteren bir yöntem yoktur. Bu deneme yanılma yolu ile belirlenmektedir. Girdi katmanındaki proses elemanlarının her birisi ara katmandaki proses elemanlarının hepsine bağlıdır. Onlarda çıktı katmanındaki proses elemanlarının hepsine bağlıdır. Bilgi akışı girdi katmanından ara katmana oradan da çıktı katmanına ileri doğrudur.

ÇKA ağının eğitilmesi "genelleştirilmiş delta kuralı"na göre gerçekleşmektedir. ÇKA ağları öğretmenli öğrenme stratejileri kullandıklarından eğitim sırasında hem girdiler hem de o girdilere karşılık ağın üretmesi gereken çıktılar ağa gösterilirler. Kullanılan öğrenme kuralının felsefesi eğitim sırasında ağın ürettiği çıktılar ile üretmesi gereken (beklenen) çıktılar arasındaki farkın (hatanın) ağın ağırlıklarına dağıtılarak zaman içinde bu farkın en aza indirgenmesidir. Öğrenme sırasında önce girdiler ağa sunularak bu girdilere karışıklık gelen çıktılar üretilir. Bu işleme ileri doğru hesaplama denir. Daha sonra üretilen çıktı ile beklenen çıktı karşılaştırılarak aradaki hata geriye doğru dağıtılarak ağırlıklar değiştirilirler. Buna da geriye doru hesaplama denmektedir.

ÇKA ağlarının olayları öğrenmesini etkileyen faktörler şunlardır.

- Örneklerin seçilmesi
- Girdi ve çıktıların ağa sunulması
- Girdi ve çıktıların sayısal gösterimi
- Ağırlıkların başlangıç değerlerinin atanması

- Öğrenme ve momentum katsayılarının belirlenmesi
- Örneklerin ağa sunulması
- Ağırlıkların değiştirilme zamanları
- Girdi ve çıktıların ölçeklendirilmesi
- Durdurma kriterinin belirlenmesi
- Ağların büyütülmesi ve budanması

ÇKA ağlarının tasarımcıları bu faktörleri dikkatlice değerlendirmeli ve problemin çözümü için en uygun yaklaşımı kullanmalıdırlar. Bölüm içinde değişik yaklaşımlar örnekler ile anlatılmıştır.

ÇKA ağlarının eğitim performansını ölçmek için eğitim bittikten sonra ağın eğitim sırasında görmediği örnekler ağa gösterilerek bunlar hakkında ağın kararına bakılır. Eğer ağ görmediği örneklere doğru cevaplar üretiyorsa o zaman performansı iyidir ve layı öğrenmiştir denir.

ÇKA ağlarının mühendislik problemlerindeki başarılı uygulamaları dikkatleri üzerine çekmektedir. Özellikle;

- Sınıflama
- Tahmin etme
- Tanıma
- Yorumlama
- Teşhis etme

Konularında tıp biliminden finans dünyasına kadar bir çok alanda başarılı uygulamalar görülmüştür. Bölüm içinde `Taguchi`'nin kalite tasarın metoduna alternatif olarak geliştirilen bir ÇKA ağı tanıtılmıştır. Problemin tanıtılması ve ağın eğitilmesi sonucu `Taguchi`'nin yaklaşık %6 hata ile tahmin ettiği üretim kombinasyonunu ÇKA ağı %25 gibi çok küçük bir hata ile daha başarılı bir şekilde tahmin etmiştir. ÇKA ağının problem hakkında ön bilgi istememesi, doğrusal olmayan ilişkileri dikkate alması gibi yararlarının olduğu da görülmüştür; bu, geleneksel yöntemlere alternatif olacağını da göstermektedir.

5.14. Kaynakça

[1] Minsky M. ve Papert S., (1969), Perceptrons. The MIT Press.

[2] Rumelhart D.E, hinton G.E, Williams R.J, (1986), "Learning representations by backpropogation errors", Nature, vol. 323, pp 533-536.

[3] Prof. Leslie Smith'in web adresinde rapor edilmiştir. Centre for Cognitive and Computational Neuroscience ,Department of Computing and Mathematics University of Stirling, lss@cs.stir.ac.uk .

[4] Taguchi g., (1986), Introduction to quality engineering: Designing quality into product and processes, Asian Productivity Organization, Tokyo, Japonya.

[5] Phadke M.S. (1986), Quality Engineering using robust Design, Prentice Hall, USA.

[6] Pignatiello J.J, Ramberg J.S. (1991-1992), Top ten triumphs and tragedies of Genichi Taguchi, Quality Engineering, Vol 4., Sayı 2, pp 221-225.

BÖLÜM 6

YAPAY SİNİR AĞI MODELİ
(DESTEKLEYİCİ ÖĞRENME) - LVQ MODELİ

Bu bölüme kadar özellikle öğretmenli öğrenme konusunda geliştirilmiş modeller tanıtılmıştır. O modellerde eğitim sırasında ağa hem girdi değerleri hem de o girdi değerleri için üretilecek çıktı değerinin ne olması gerektiği konusunda bilgiler verilmektedir. Bazı durumlarda ağa çıktının ne olduğunu vermek mümkün olamamaktadır; fakat, ağın üretmiş olduğu çıktının doğru veya yanlış olduğu belirtilebilmektedir. Destekleyici öğrenme olarak belirlenen bu yöntemi kullanan modellerin bir tanesi de doğrusal vektör parçalama modeli diyebileceğimiz LVQ (*Linear Vektor Quantization*) modelidir. Bu bölümde bu model ayrıntılarıyla ele alınmaktadır.

6.1. LVQ Ağının Özellikleri

LVQ ağı Kohonen tarafından 1984 yılında geliştirilmiştir[1]. Temel felsefesi *n* boyutlu bir vektörü bir vektörler setine haritalamaktır (uydurmaktır). Aslında bir vektörün belirli sayıda vektörler ile gösterimi amaçlanmaktadır. Öğrenme ile de girdi vektörünün hangi vektör seti tarafından temsil edilmesi gerektiği bulunmak 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.

LVQ ağları genel olarak sınıflandırma problemlerinin çözümünde kullanılmaktadırlar. Çıktılardan sadece birisi 1 diğerleri 0 değerlerini almaktadırlar. Çıktı değerinin 1 olması girdinin ilgili çıktının temsil ettiği sınıfa ait olduğunu göstermektedir.

Eğitim sırasında girdilerin sınıflara ayrılması en yakın komşu (*nearest neigbour*) kuralına göre gerçekleştirilmektedir. Girdi vektörü ile referans vektörleri arasındaki en kısa mesafe aranmakta ve girdi vektörünün en kısa mesafede bulunan vektör grubuna ait olduğu varsayılmaktadır. Ağın ağırlıklarını değiştirmek yolu ile gidileri doğru sınıflara ayıracak referans vektörleri belirlenmektedir. Kullanılan öğrenme stratejisi destekleyici (*reniforcement learning*) öğrenmedir. Çıktı değerlerinin belirlenmesinde ise ***"kazanan herşey alır"*** (*winner takes all*) stratejisi uygulanmaktadır. Ağ eğitilirken her iterasyonda ağın ürettiği çıktının değeri yerine sadece doğru olup olmadığı söylenir. Sadece girdi vektörüne en yakın olan vektör'ün (kazanan vektör) değerleri (ağın bu vektöre ait ağırlıkları) değiştirilir.

Diğer ağlarda olduğu gibi LVQ ağında da ağın ağırlıkları öğrenme katsayısına göre değiştirilmektedir.

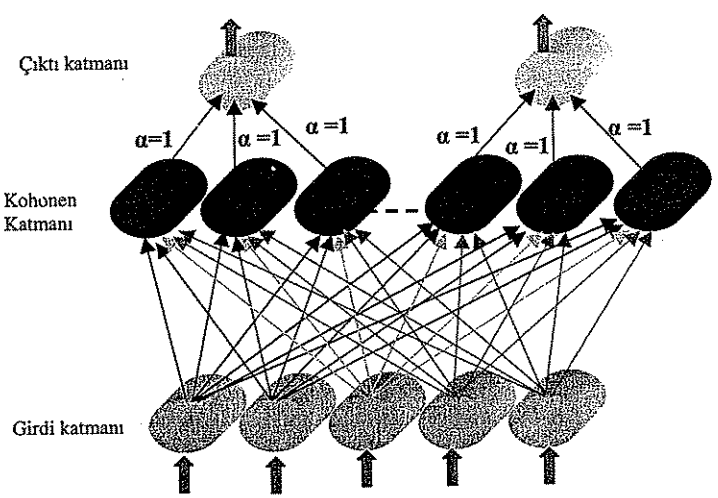
Kullanılan öğrenme katsayısının zaman içerisinde sıfır olacak şekilde monoton olarak azaltılması istenmektedir.

LVQ ağının öğrenme hızı da ÇKA gibi ağlara göre daha yüksektir. Yani LVQ ağları bir olayı ÇKA ağlarından daha kısa zamanda öğrenebilmektedir.

6.2. LVQ Ağının Yapısı

ÇKA'larda olduğu gibi LVQ ağları da 3 katmandan oluşmaktadır. Şekil-6.1 bu katmanları göstermektedir. Bunlar:

- ***Girdi katmanı:*** Bu katmanda bilgi işleme olmaz. Dış dünyadan alınan örneklerin ağa gösterilmesi bu katmanda sağlanmaktadır. Gelen bilgiler girdi vektörünü oluşturur.
- ***Kohonen katmanı*** *(ara katmanda denmektedir).* Bu katmanda girdi setine en yakın olan ağırlık vektörü belirlenmektedir. Bu katmandaki her eleman bir referans vektörünü göstermektedir. Girdi vektörü, girdi katmanı ile *Kohonen* katmanı arasındaki ağırlıkların oluşturduğu referans vektörlerine haritalanmaktadır.
- ***Çıktı katmanı***: Bu katmanda ise girdinin ait olduğu sınıf belirlenir.



Şekil-6.1. LVQ ağının topolojik yapısı

Şekilden de görüldüğü gibi LVQ ağları girdi katmanı ile *Kohonen* katmanı arasında tam bağlantılı, *Kohonen* katmanı ile çıktı katmanı arasında ise kısmi bağlantılıdır. Yani Girdi katmanındaki her proses elemanı *Kohonen* katmanındaki her proses elemanına bağlıdır. *Kohonen* katmanındaki proses elemanları ise sadece çıktı katmanındaki bir tek proses elemanına bağlıdır. *Kohonen* katmanı ile çıktı katmanı arasındaki ağırlıklar (α) sabit olup 1'e eşittir. Yani bu ağırlıkla değişmezler. Sadece girdi katmanı ile *Kohonen* katmanı arasındaki ağırlıklar değiştirilirler. Öğrenme bu ağırlıklar üzerinden gerçekleştirilir. *Kohonen* katmanında kaç tane proses elemanı var ise o kadar referans vektörü oluşacaktır. ***Referans vektörü*** girdi değerlerini *Kohonen* katmanındaki proses elemanlarına bağlayan bağlantıların ağırlık değerlerinden oluşur. Dolayısı ile referans vektörünün eleman sayısı girdi katmanındaki eleman sayısı kadardır. Bu konu ileride yeniden tartışılacaktır (bkz. Şekil-6.4)

Kohonen ve çıktı katmanlarındaki proses elemanlarının çıktıları ikili (*binary*) değerler olup sadece bir proses elemanının çıktısı 1 diğerlerinin ki ise 0'dır. *Kohonen* katmanında proses elemanları birbirleri ile yarışırlar. Yarışı kazanan proses elemanını çıktısı 1 değerini alır diğerlerinin çıktısı ise 0 olur. Hangi proses elemanının yarışı kazandığına öğrenme kuralına göre karar verilir. *Kohonen* katmanında hangi proses elemanının çıktısı 1 olursa onun bağlı olduğu çıktı katmanındaki proses elemanının çıktısı 1 değerini alır diğerlerinin çıktısı da 0 olur. Böylece ağa sunulan bir girdi için çıktı katmanında sadece bir proses elemanının çıktısı 1 olmakta ve girdi vektörü o çıktının gösterdiği sınıfın üyesi kabul edilmektedir. Öğrenme ile girdi için doğru sınıfın belirlenmesi sağlanmaktadır.

6.3. LVQ Ağının Çalışma Prosedürü

LVQ ağlarını kullanmak için şu prosedürü izlemek gerekmektedir:

1. Örneklerin belirlenmesi
2. Ağın topolojisinin belirlenmesi (girdi ve çıktı sayısının belirlenmesi, referans vektör sayısının belirlenmesi)
3. Ağın öğrenme parametrelerinin belirlenmesi (öğrenme katsayısı ve istenen sabit değerlerin belirlenmesi)
4. Ağırlıkların başlangıç değerlerinin atanması
5. Öğrenme setinden bir örneğin ağa gösterilmesi
6. Kazanan proses elemanının bulunması
7. Ağırlıkların değiştirilmesi
8. Bütün örnekler doğru sınıflandırılıncaya kadar yukarıdaki adımları (5-7) tekrar etmek...

Bir LVQ ağının performansı doğru sayıda referans vektörünün belirlenmesi, ağırlıkların başlangıç değerleri ve öğrenme katsayısının belirlenmesi ile yakından ilgilidir. Bunları belirlerken tasarımcıların çok dikkatli olmaları ve tecrübelerini kullanmaları gerekmektedir.

6.3.1. LVQ Ağının Öğrenme Kuralı

LVQ ağının öğrenme kuralına *Kohonen* öğrenme kuralı da denmektedir. Öğrenme kuralı, *Kohonen* tabakasındaki proses elemanlarının birbirleri ile yarışmaları ilkesine dayanır. Yarışma girdi vektörü ile ağırlık vektörleri (referans vektörler) arasındaki öklid (*euclid*) mesafesinin hesaplanmasına dayanmaktadır. Hangi proses elemanın referans vektörü girdi vektörüne en yakın ise o yarışmayı kazanmaktadır. Girdi vektörü X ile referans vektörü A arasındaki mesafe d ile gösterilirse; i. prosess elemanının mesafesi şu şekilde hesaplanmaktadır.

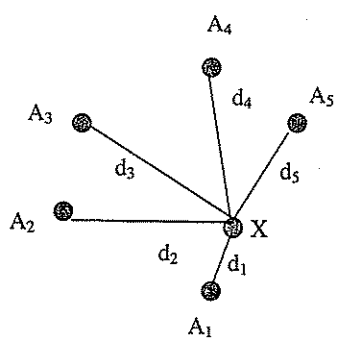
$$d_i = \|A_i - X\| = \sqrt{\sum_j (A_{ij} - x_j)^2}$$

Burada A_{ij} ve x_j ağırlık vektörü ve girdi vektörünün j. değerlerini göstermektedir. girdi vektörü ile referans vektörlerinin hepsinin aralarındaki mesafesi tek tek hesap edildikten sonra hangi proses elemanın referans vektörü girdi vektörüne en yakın ise o yarışmayı kazanmaktadır. Öğrenme sırasında, sadece girdi katmanını bu proses elemanına bağlayan ağırlık değerleri değiştirilir. Diğer ağrılıklar değiştirilmezler. Kazanan proses elemanı için iki durum söz konusudur.

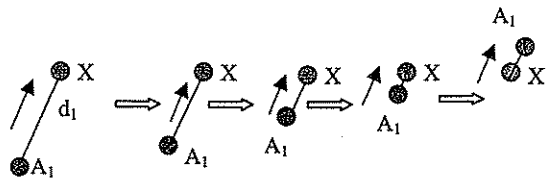
a) Kazanan proses elemanı doğru sınıfın bir üyesidir. Bu durumda ilgili ağırlıklar girdi vektörüne biraz daha yaklaştırılırlar. Bu, aynı örnek ağa tekrar gösterildiğinde yine aynı proses elamanının kazanması için yapılmaktadır. Bu durumda, ağırlıkların değiştirilmesi şu formüle göre yapılmaktadır.

$$A_y = A_e + \lambda(X - A_e)$$

Burada λ öğrenme katsayısıdır. Zaman içersini de sıfır değerini alacak şekilde monoton olarak azaltılır. Bunun nedeni girdi vektörünün referans vektörüne çok yaklaştığında durması ve aksi yönde tekrar uzaklaşmaması içindir. Aksi taktirde ters yönde tekrar uzaklaşma olacaktır. Bu durum Şekil-6.2 ve Şekil-6.3'te gösterilmiştir.



Şekil-6.2. Girdi vektörüne en yakın ağırlık (referans vektörü (A_1))



Şekil-6.3. Ağırlık vektörünün girdi vektörüne yaklaşması

Şekil-6.2'de görüldüğü gibi,girdi vektörüne en yakın vektör A_1 vektörüdür. Bu vektörün sürekli X vektörüne yaklaştırılması zaman içinde onu geçerek ters yönde uzaklaşması anlamına gelecektir. Şekil-6.3'te bu durum vurgulanmak istenmektedir. Görüldüğü gibi A_1 vektörü sürekli X vektörüne yaklaşmaktadır. Belirli bir süre sonra bu iki vektör birbirine çok yakın (bazen üst üste) olmakta ve daha sonra A_1 vektörü tekrar X vektöründen uzaklaşabilmektedir. O nedenle, girdi ve ağırlık vektörleri birbirine çok yakın olduğunda ağırlıkların değişmemesi için öğrenme katsayısı sıfır değerine indirilmektedir. Burada tasarımcıların çok dikkatli olması gerekir. Öğrenme katsayısının ne çok erken nede çok geç sıfıra indirgenmemesi lazımdır. Eğitim süreci çok ayrıntılı incelenerek öğrenmenin ne zaman durdurulacağına karar verilmelidir.

b) Kazanan proses elemanı yanlış sınıftandır. Bu durumda ağırlık vektörü girdi vektöründen uzaklaştırılır. Bir daha aynı örnek geldiğinde aynı proses elemanı kazanmasın diye bu uzaklaştırma yapılır. Şu formül ile ağırlıklar değiştirilir.

$$A_y = A_e - \lambda(X - A_e)$$

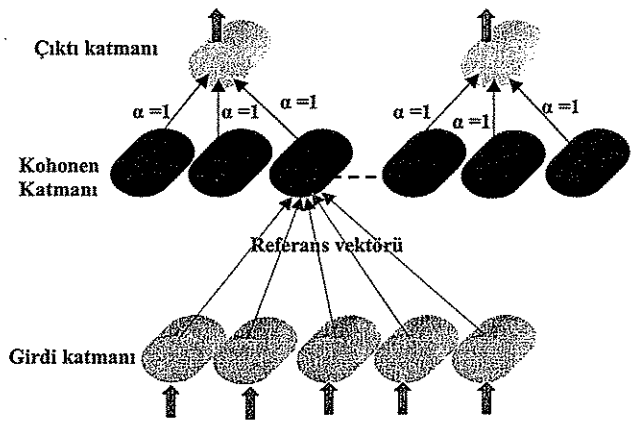
Öğrenme katsayısının zaman içinde sıfır değerini alacak şekilde monotonik olarak azalması burada da geçerlidir.

Şu ana kadar anlatılan algoritma ve tanıtılan model standart LVQ olarak bilinmektedir. Yukarıda anlatılan yöntemin en önemli dezavantajı daha önce belirtildiği gibi öğrenme katsayısının zamanında sıfır değerini almaması durumunda yani fazla eğitim yapılması durumunda ağın öğrendiklerini unutması ve doğru ağırlık değerlerinden uzaklaşmasıdır. Ayrıca bazı problemlerde sürekli aynı referans vektörü yarışmayı kazanmaktadır. Bu da, ağın esnekliğini ortadan kaldırmaktadır. Diğer bir sorun ise sınıflandırmayı yaparken iki sınıfın tam ortasında veya sınırlara çok yakın bulunan vektörlerin hangi sınıfa gireceklerinin belirlenememesidir. Bu problemleri önlemek için LVQ ağı farklı şekillerde değiştirilmiş ve öğrenme kuralı sürekli geliştirilmiştir. Aşağıda değişik LVQ ağlarına örnekler verilecektir.

6.3.2. LVQ Ağının Eğitilmesi

LVQ ağının eğitilmesinde amaç her iterasyonda girdi vektörüne en yakın referans vektörünü bulmaktır. Referans vektörleri *Kohonen* katmanındaki proses elemanlarını girdi katmanındaki proses elemanlarına bağlayan ağırlık değerleridir. Şekil-6.4'te *Kohonen* katmanındaki 3. proses elemanını girdi katmanına bağlayan referans vektörü görülmektedir.

Öğrenme esnasında sadece referans vektörlerinin ağırlık değerleri değiştirilir. Değişimin ne şekilde olacağı yukarıda öğrenme kuralı anlatılırken belirtilmiştir. Şekil-6.4 aynı zamanda girdi katmanı ile çıktı katmanı arasındaki ağırlıkların (α) sabit ve 1 değerine sahip olduklarını da göstermektedir. Bu ağırlıklar eğitim sırasında da değiştirilmezler.



Şekil-6.4. Referans vektörü örneği

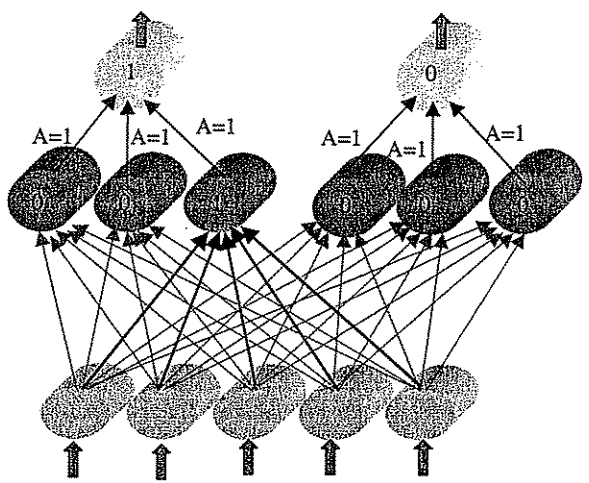
Öğrenme sırasında girdi katmanından gösterilen bir örnek sonucu referans vektörlerinin ağırlık değerleri kullanılarak girdi vektörü ile arasındaki mesafeleri hesaplanır. Bu mesafelerden en küçük değeri hangi proses elemanına ait referans vektörü üretiyor ise o proses elemanının çıktısı 1 diğerlerininki 0 olur. Yani *Kohonen* katmanındaki her proses elemanın çıktısı $Ç^k$ ise,

$$Ç_i^k = \begin{cases} 1 & \text{Eğer i. proses elemanı yarışı kazanırsa} \\ 0 & \text{Aksi halde} \end{cases}$$

Kohonen katmanındaki proses elemanlarının çıktıları bu proses elemanları çıktı katmanına bağlayan ağırlık değerleri ile çarpılarak ağın çıktısı hesaplanır. Yani,

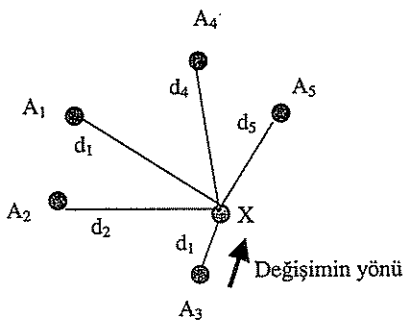
$$Ç_i = \sum_j Ç_j^k \alpha_{ki}$$

olacaktır. Bu *Kohonen* katmanında yarışmayı kazanan proses elemanına bağlı olan çıktı elemanın değerinin 1, diğerlerinin değerinin 0 olması anlamına gelmektedir. Burada *Kohonen* katmanındaki proses elemanlarının sadece bir tane çıktı elemanına bağlı olduğu unutulmamalıdır. Şekil 6.5'de bu durum bir örnek ile gösterilmektedir. Örnekte girdi katmanından gelen bilgileri *Kohonen* katmanındaki 3. proses elemanının kazandığı varsayılmaktadır. Kazanan referans vektörü A_3 ile gösterilmektedir. Ağ 5 girdi 6 *Kohonen* katmanı elemanı ve 2 çıktı elemanından oluşmaktadır. *Kohonen* katmanındaki 3. eleman çıktı katmanındaki 1. elemana bağlı olduğundan çıktı katmanında 1. eleman 1, diğerleri ise 0 değerini almaktadır. Ağın çıktıları belirlendikten sonra yapılacak iş çıktının doğru sınıflandırılıp sınıflandırılmadığın sorgulamaktır. Bu sorunun cevabına göre *Kohonen* katmanındaki 3. elemanı girdi katmanına bağlayan ağırlıklar değiştirilir.



Şekil-6.5. LVQ öğrenme prosedürünün geometrik gösterimi

Bulunan sonuç eğer doğru ise o zaman Şekil-6.6'de gösterildiği gibi referans vektörü (A_3) ok yönünde girdi vektörüne (X) yaklaştırılır. Öğrenme setindeki bütün örnekler doğru sınıflandırılıncaya kadar bu işlemler tekrarlanır. Hepsi doğru sınıflandırılınca öğrenme gerçekleştirilmiş olur. Bazı durumlarda girdi setinden bir iki (çok az sayıda) adet öğrenilemeyebilir. Bu örneklerin yapısından kaynaklanabilir. Bu durumda da ağ öğrenemedi denilemez. Önemli olan tasarımcının eğitim sonucunda öğrenilen sınıflandırmadan memnun olması ve test seti sınıflandırılmasının kabul edilebilir olmasıdır.



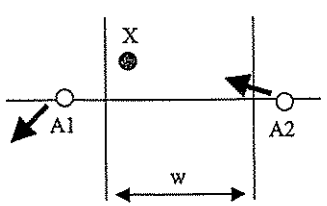
Şekil-6.6. LVQ öğrenmede ağırlıkların değişim yönü
(*Kohonen* katmanındaki 3. eleman doğru sınıflandırdığından referans vektörü girdi vektörüne yaklaştırılmaktadır)

6.4. LVQ2 Ağı

LVQ2 ağı yine *Kohonen* tarafından geliştirilmiştir [2]. LVQ2 standart LVQ modelinin uygulanması sonucunda elde edilen çözümüm iyileştirilmesi amacı ile geliştirilmiştir. Özellikle sınıfların sınır değerlerindeki yanlış sınıflandırmaları önlemeye çalışır. Standart LVQ'nun aksine LVQ2 ağı eğitim sırasında aynı anda 2 referans vektörünün ağırlıklarını değiştirmeyi önermektedir. Bu iki vektöre A_1 ve A_2 denirse, bunların ağırlıklarının değiştirilmesi için iki koşulun sağlanması gerekmektedir. Bunlar:

1. A_1 girdi vektörüne en yakın ağırlık vektörü, A_2 ise ondan sonraki en yakın ağırlık vektörüdür. A_1 yanlış, A_2 ise doğru sınıftandır.
2. Girdi vektörü A_1 ve A_2 vektörlerinin arasında merkezi olarak belirlenmiş bir aralık içerisinde kalmaktadır.

Bu durum Şekil-6.7'te gösterilmektedir. Şekildeki w, A_1 ve A_2 vektörleri arasında belirlenmiş aralığın genişliğini göstermektedir. X ise girdi vektörüdür. w aralığı tasarımcı tarafından belirlenmektedir.



Şekil-6.7. LVQ2 prosedürünün geometrik gösterimi

Yukarıda anlatılan iki durumun sağlanması halinde A_1 ve A_2 vektörlerinin ikisinin de ağırlıkları değiştirilir. Bu ağırlık vektörlerinin yeni değerleri (A_{1y} ve A_{2y}) şu şekilde hesaplanır.

$$A_{1y} = A_{1e} - \lambda(X - A_{1e})$$
$$A_{2y} = A_{2e} + \lambda(X - A_{2e})$$

Burada A_{1e} ve A_{2e} ağırlık vektörlerinin değişmeden önceki değerlerini göstermektedir. Girdi vektörünün iki ağırlık vektörü arasında belirlenen bir aralıkta olup olmadığına ise şu şekilde karar verilir. Eğer,

$$Min(d_1 / d_2, d_2 / d_1) > s \quad ve$$
$$s = (1 - w) / (1 + w)$$

ise o zaman girdi vektörü iki vektörün arasında belirlenen aralıkta demektir. Burada:

d_1 A_1 ağırlık vektörünün girdi vektörüne olan mesafesi;
d_2 ise A_2 ağırlık vektörünün girdi vektörüne olan mesafesidir.

Bir problemi LVQ ağı kullanarak bilgisayara öğretmek için o problemin öncelikle standart LVQ ağı kullanarak eğitmekte ondan sonra LVQ2 ağını kullanarak eğitime devam etmekte fayda vardır. Standart LVQ sınıfları belirlemekte LVQ2 ise sınırlarda bulunan vektörleri doğru sınıflara ayırmak için sınıfların sınırlarını güncellemektedir. LVQ2 ağının standart LVQ ile birlikte kullanılması öğrenmenin hızını ve performansını artırabilmektedir.

6.5. Cezalandırma Mekanizmalı LVQ

Standart LVQ modelinde bazı ağırlık vektörleri çok sık üst üste kazanmakta ve bu da ağın esnekliğini bozmaktadır. Diğer ağırlık vektörleri referans olma niteliklerini kaybetmektedirler. Belirlenen her ağırlık vektörü girdinin bir yönünü göstermesi gerekirken sadece bir tanesi bütün sorumluluğu üstlenmektedir. Bu problemden kurtulmak için Desiono [3] tarafından önerilen cezalandırma mekanizması ile ağın eğitimi sırasında sürekli kazanan ağırlık vektörü cezalandırılmakta ve sürekli peş peşe kazanması önlenmektedir. Bu cezalandırma mekanizması ağırlık vektörünün girdi vektöründen olan mesafesine b kadar değerin eklenmesi ile gerçekleşmektedir. Eklenecek b değeri ilgili ağırlık vektörünün yarışmayı kaç defa kazandığına bağlı olarak belirlenmektedir. i. proses elemanına ait ağırlık vektörünün girdi vektöründen uzaklık mesafesine eklenecek miktar (b_i) şu şekilde hesaplanmaktadır:

$$b_i = C\left(p_i + \frac{1}{N}\right)$$

Burada C sabit bir değeri göstermekte olup kullanıcı tarafından belirlenmektedir. N *Kohonen* katmanındaki proses elemanı sayısını göstermektedir. p_i ise i. proses elemanının yarışmayı kazanma olasılığıdır. Bu olasılık başlangıçta $1/N$ olarak belirlenmekte ve daha sonra şu kurala göre sürekli güncellemektedir.

$$p_i^y = p_i^e + B(y_i - p_i^e)$$

Burada p^y, proses elemanının kazanma olasılığının yeni değerini, p^e ise proses elemanının değişmeden önceki kazanma olasılık değerini göstermektedir. B verilerdeki dalgalanmaları önlemek amacı ile geliştirilmiş sabit bir sayıdır. Kullanıcı tarafından atanmaktadır. Bu değerler deneme yanılma yolu ile bulunmaktadırlar. Literatürde öneriler olmakla birlikte hangi değerin alınacağı konusu tamamen probleme göre değişir ve kullanıcını kendisinin belirlemesi gerekmektedir. Bu değerleri belirlemek için farklı değerler ile denemeler yapmak faydalı olur. y_i ise çıktı değerini göstermektedir; y_i değeri

$$y_i = \begin{cases} 1 & \text{Eğer i. proses elemanı yarışı kazanırsa} \\ 0 & \text{Aksi halde} \end{cases}$$

Cezalandırma mekanizmalı LVQ ağında, referans vektörlerine eklenecek ilave b değerleri bütün proses elemanları için belirlendikten sonra girdi vektörü ile ağırlık vektörü arasındaki hesaplanan mesafelere bu b değerleri eklenmektedir. Yani,

$$d_i^y = d_i^e + b_i$$

olmaktadır. Burada d^e girdi vektörü ile ağırlık vektörü arasındaki gerçek mesafeyi , d^y ise kazanma olasılığına dayalı yeni mesafe değerini göstermektedir. Yeni mesafeler hesaplandıktan sonra *Kohonen* katmanındaki proses elemanları arasındaki yarışma bu yeni değerler dikkate alınarak yapılmaktadır. Böylece fazla kazanan proses elemanının mesafesi daha fazla artırılıp cezalandırılarak sürekli kazanması önlenmekte ve diğerlerine de zaman içinde kazanma şansı tanınmaktadır. Bu cezalandırma mekanizmasının özellikle ağın ezberlemesini önlediği söylenebilir.

6.6. LVQ-X Modeli

Yukarıda anlatılan modeller iyice incelendiğinde her iterasyonda sadece bir ağırlık vektörünün değiştirildiği görülmektedir. Her ne kadar LVQ2 ağı iki ağırlık vektörünün değerlerini değiştirmekte ise de bu çok az rastlanan durumlarda kendini göstermekte ve sadece tam sınırlardaki ağırlık vektörleri için etkili olmaktadır. Bunların aksine, LVQ-X ağının öğrenme kuralına göre eğitim sırasında her iterasyonda çoğunlukla iki ağırlık vektörünün ağırlıkları değiştirilmektedir. Bu hem öğrenme hızını artırmakta ve öğrenme zamanını kısaltmakta hem de ağın genelleme yeteneğini artırmaktadır. LVQ-X ağının öğrenme kuralı `Oztemel` [4] tarafından geliştirilmiş olup her iterasyonda yarışmayı kazanan iki tane proses elemanı belirlemektedir. Bunlar,

- ***Global kazanan***: Girdi vektörüne en yakın ağırlık vektörüne sahip olan proses elemanını göstermektedir.
- ***Yerel kazanan*** : Bu ise doğru sınıf içerisinden girdi vektörüne en yakın olan ağırlık vektörüne sahip proses elemanını göstermektedir.

LVQ-X'in felsefesi şöyledir. Eğer global kazanan proses elemanı doğru sınıf içerisinde değil ise onun ağırlık vektörü o girdi vektöründen uzaklaştırılır. Aynı zamanda doğru

sınıf içindeki en yakın olan proses elemanının (yerel kazanan) ağırlık vektörü de girdi vektörüne yaklaştırılır. Bu doğru proses elemanın daha sonraki iterasyonlarda kazanmasına şans vermektedir. Eğer global kazanan ve yerel kazanan aynı proses elemanı ise o zaman sadece tek bir ağırlık vektörü değiştirilmekte ve kazanan proses elemanının ağırlık vektörü girdi vektörüne yaklaştırılmaktadır. Ağırlık vektörlerinin değerleri şu şekilde değiştirilmektedir.

a) Global kazanan ve yerel kazanan aynı proses elemanı ise o proses elemanına ait referans vektörü için;

$$A_y = A_e + \lambda(X - A_e)$$

b) Global kazanan ve yerel kazanan farklı proses elemanları ise;

Yerel kazanan proses elemanının referans vektörü için;

$$A_y = A_e + \lambda(X - A_e)$$

Global kazanan proses elemanının referans vektörü için;

$$A_y = A_e - \lambda(X - A_e)$$

Geliştirilecek bir LVQ ağı uygulamasında LVQ-X , LVQ2 ve Cezalandırılmış LVQ modellerinin birlikte kullanılması sonucu öğrenmenin performansı oldukça iyi bir değere ulaşmaktadır. Önce LVQ-X öğrenme kuralı cezalandırma mekanizması ile birlikte uygulanmakta ve eğitim seti öğrenildikten sonra LVQ2 ile sınır değerleri iyileştirilmektedir. Eğitilmiş LVQ ağlarının kullanılmasında ağırlıklarda her hangi bir değişiklik yapılmadığından ağların farklı öğrenme kuralları ile eğitilmiş olmasının bir farkı yoktur.

6.7. LVQ Ağının Uygulama Alanları

LVQ ağları genel olarak sınıflandırma yapmak amacı ile geliştirilmişlerdir. Örüntü tanıma ve birbirinden farklı şekilleri sınıflandırmak, birbirinden farklı nesneleri tanımak vb. gibi konularda mühendislik bilimleri başta olmak üzere hayatın her alanında uygulamalarını görmek mümkündür. Çok hızlı sonuç üretiyor olmaları ve performanslarının çok yüksek olması diğer ağlara göre bu ağları tercih sebebi olmaktadır.

Aşağıda kalite kontrol amacı ile geliştirilmiş bir LVQ ağı uygulanması ayrıntılı olarak anlatılmıştır. Ağın oluşturulması, örnek setinin belirlenmesi açıklanmış ve problem yukarıda anlatılan değişik öğrenme kuralları kullanılarak LVQ ağına öğretilmiştir. Sonuçların bir karşılaştırması da yapılmıştır.

6.8. LVQ – Endüstriyel Bir Örnek Uygulama (Örüntü Tanıma)

Günümüzde endüstriyel kuruluşların en önemli sorunlarından birisi kalite kontroldür. İstatistiksel süreç kontrolü yaklaşımı ile üretim hattında kalite problemleri oluşmadan önceden belirlenerek hataların oluşmaması için önlemler almak ve kalite problemlerinin oluşmasını önlemek amaçlanmaktadır. Bunu gerçekleştirmek kolay olmamaktadır.

Çünkü bu konuda gerekli önlemleri almak yoğun proses bilgi ve tecrübesi gerektirmektedir. Üretim süreci hakkında otomatik bilgiler üretecek sistemlere ihtiyaç vardır. Bu bölümde bu konuda geliştirilmiş LVQ ağına dayalı bir uygulama açıklanacaktır. LVQ ağı prosesin davranışlarını öğrenerek üretim süreci hakkında etkin kararlar verebilmektedir.

6.8.1. Problemin Tanımlanması

İstatistiksel kalite kontrolün temel felsefesi hataların oluşmadan hataya neden olacak faktörleri ortadan kaldırarak kaliteli ürünler üretmektir. Bunu başarmak için üretim süreci sürekli izlenmektedir. İmalat sektöründe yapılan çalışmalar üretilen ürünlerin özellikleri aynı olsa ve aynı makinelerde üretilseler bile, kesinlikle birbirinin aynı olmadığını göstermiştir. Ürünlerin üretilmesindeki değişiklikler iki nedenden ötürü oluşmaktadır. Bunlar:

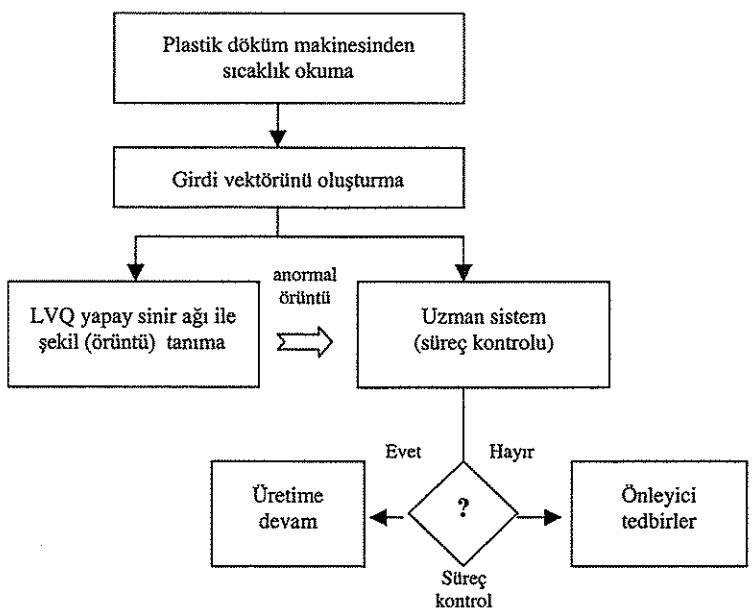
a) kontrol edilebilir faktörlerden kaynaklanan değişim
b) kontrol edilemez faktörlerden kaynaklanan değişim

İstatistiksel kalite kontrolü kontrol edilemez faktörler ile uğraşmaz çünkü havanın nem oranı gibi kontrol edilemeyen bir faktörü üzerinde kararlar vermek ve bu faktörleri değiştirmek hem mümkün değildir hem de anlamsız olur. Önemli olan kontrol edilebilir faktörler üzerindeki değişimleri ortadan kaldırarak onları kontrol altında tutmaktır. Bu faktörlerdeki değişimlerin ürünün özelliklerinde (spesifikasyonlarında) değişiklik ve kalite sorunlarına neden olması durumunda önceden bazı önlemler almak mümkündür. Bu sorunlar oluşmadan gerekli önleyici tedbirler almak için ürünün bir kalite karakteristiğini belirlemek ve üretim hattındaki süreçlerin bu karakteristik kapsamında izlenmesi ile mümkün olmaktadır. Bu kalite karakteristiği sürekli ölçülerek kontrol şemaları denilen şemalar üzerinde bu ölçülen değerler yorumlanmakta ve yerleştirilerek süreç hakkında kararlar verilmektedir.

Bu yorumlar önceden belirlenmiş bazı kuralları kullanarak yapılmakla birlikte kontrol şeması üzerinde oluşan şekillerin yorumlanması ile de gerçekleştirilmektedir. Kontrol şemaları üzerinde değişik şekiller oluşabilmektedir. Örneğin herşeyin normal olması durumunda şema üzerinde normal dağılıma uygun bir şekil görülmektedir. Ölçülen kalite karakteristiğinin ortalama değerinde zaman ile bir yükselme söz konusu olursa o zaman şema üzerinde yükselen bir trend görülmektedir. Her şeklin oluşması durumunda (mesela trend olması durumunda) alınması gereken önlemler belirlidir. Önemli olan şeklin tanınmasıdır. Otomatik olarak şekilleri tanıyan bir sistem kurulması halinde önleyici tedbirleri operatörlere bildirmek ve gerekeni yapmalarını istemek mümkün olabilir. O nedenle şekillerin otomatik olarak tanınması imalat sektörü için çok önemlidir.

Bu açıklamalardan sonra aşağıda araba üretimi yapan bir firma için jonta üreten bir üretim hattında jontaların enjeksiyon preslerinde dökülmesi sırasında sıcaklığın kontrol edilmesine yönelik geliştirilen bir kontrol şemalarının şekillerini tanıma sistemi (örüntü tanıma sistemi) anlatılacaktır. Bu amaçla geliştirilen bir uzman sistem XPC olarak adlandırılmıştır. Uzman sistem prosesin kısa dönemli davranışını kontrol

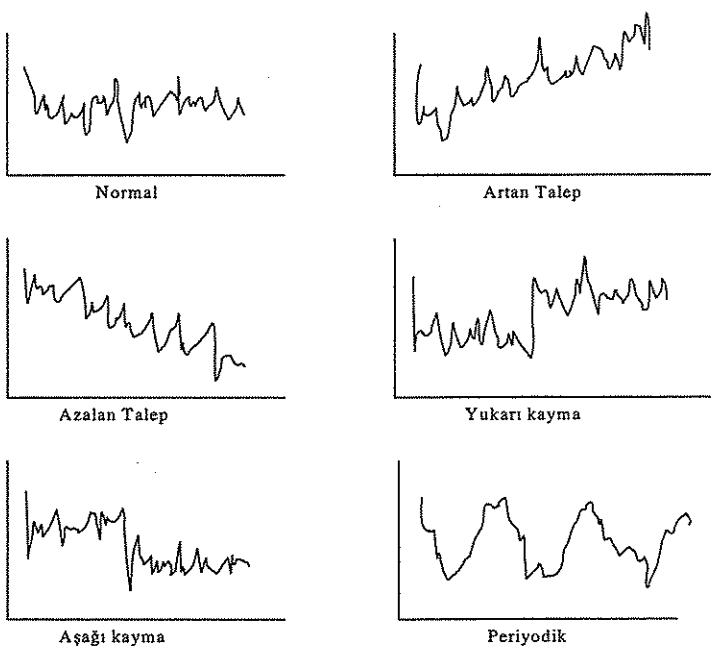
edebilmektedir. Uzun dönemli davranışlarını otomatik olarak algılamak için LVQ ağından faydalanılmaktadır. LVQ ağları kontrol şemaları üzerinde oluşan şekilleri yorumlayarak uzun dönemli davranışlar hakkında bilgi sahibi olmaktadır. Geleneksel yöntemlerden oldukça fazla doğruluk oranında çalışan bir sistem geliştirilmiştir. Geliştirilen sistem makinelerde sürekli ölçülen sıcaklık değerlerin alarak sürecin kontrol altında olup olmadığını kontrol etmektedir. Eğer ölçüm değerleri normal dağılıma uygun hareket ediyorsa bir sorun olmamakta aksi durumda ise bir uzman sistem ile önleyici tedbirler önerilmektedir. Geliştirilen XPC sistemin yapısı Şekil-6.8'de gösterilmiştir. Makine üzerinden sıcaklıklar hem otomatik olarak okunabilmekte hem de el ile girilebilmektedir. Her gelen yeni sıcaklık değeri önceki değerler ile birleştirilerek girdi vektörü oluşturulmaktadır. Girdi vektörünün boyutu 60'dır. Yani sürekli son 60 ölçüm (sıcaklık değeri) dikkate alınmaktadır. Girdi vektörü hem LVQ ağına hem de uzman sisteme girdi olarak gönderilmektedir. Uzman sistem sürecin kısa dönemli davranışlarını (son 15 ölçümü dikkate alarak) yorumlar iken LVQ ağı da yukarıda belirtildiği gibi sürecin uzun dönemli davranışını yorumlamaktadır. Bu bölümün amacı LVQ ağının uygulanmasını anlatmak olduğundan burada uzman sistem konusu anlatılmayacaktır. Sadece yapay sinir ağı uygulaması tanıtılacaktır. Sistemin tamamı ile ilgili bilgiler [5] nolu referansta bulunabilir.



Şekil-6.8. Zeki istatistiksel kalite kontrol sistemi elemanları

XPC sisteminin LVQ şekil tanıma modüle Şekil-6.9'de verilen 6 adet şekli tanıyabilmektedir. Diğer bir deyişle ölçülen sıcaklıkların bu altı sınıftan hangisine ait olduğunu belirlemektedir. Bu şekiller:

- **Normal**: Bu şekil, üretim hattında herşeyin normal olduğunu, üretimdeki değişikliğin sadece kontrol altında tutulamayan faktörlerden kaynaklandığını göstermektedir.
- **Trend:** Kalite karakteristiğinde (burada ürünün üretildiği makinenin sıcaklığında) ortalama değerin zaman içinde artması veya azalması durumudur.
- **Kayma:** Kalite karakteristiğinin ortalama değerinde ani bir artış veya azalmanın söz konusu olduğu durumdur.
- **Periyodik**: Kalite karakteristiğinde zaman içinde birbirini tekrar eden periyodik şekillerin oluşması durumudur.



Şekil-6.9. XPC sisteminde LVQ ağının tanıdığı kontrol şeması üzerindeki şekiller

6.8.2. Öğrenme Setinin Oluşturulması

XPC programı, bir endüstriyel kuruluşta uygulandığından ilgili kuruluşta Şekil-6.9'da gösterilen şekillere ait örnekler araştırılmıştır. Geçmişte doldurulmuş kontrol şemalarında yeterli oranda örnek bulunamadığından örneklerin benzetim yolu ile üretilmesi kararlaştırılmıştır. Bunun için her şekli üreten matematik formülasyon belirlenmiştir. Bu konuda ayrıntılı bilgiler `Pham` ve `Oztemel` [5] tarafından açıklanmıştır.

6.8.2.1. Normal Şeklin Üretilmesi

Şekil-6.9'da örneği gösterilen ***"normal"*** şekiller şu formül kullanılarak oluşturulmuştur.

$$y(t) = \mu + r(t) * \sigma$$

Bu formülde $y(t)$ *t.* sıcaklık değerini, μ beklenen ortalama sıcaklık değerini, $r(t)$ normal dağılıma uygun olarak üretilmiş *t.* rassal değeri, σ ise ölçülen sıcaklık değerlerinin ortalamadan sapmalarını gösteren standart sapmadır. Bu örnek uygulamada ortalama sıcaklık değeri ve standart sapma için şu değerler alınmıştır.

$$\mu = 80 \qquad \sigma = 5$$

Burada bir konuya dikkatleri çekmekte yarar vardır. Bilgisayarlar genel olarak düzgün (*uniform*) dağılıma uygun rassal değerler üretmektedirler. O nedenle bu uygulamada bilgisayarın ürettiği değerleri normal dağılıma uygun hale getirmek için bir dönüşüm fonksiyonu kullanılmıştır. Bu fonksiyon

$$r(t) = \sqrt{-2\ln(u)}\cos 2\pi u$$

şeklindedir. Bu fonksiyondaki $r(t)$ normal dağılıma uygun *t.* değeri, *u* ise bilgisayar tarafından düzgün dağılıma uygun üretilmiş rassal değeri göstermektedir.

6.8.2.2. Artan veya Azalan Trendin Üretilmesi

Şekil-6.9'da örneği gösterilen ***"azalan ve artan trend"*** şekilleri şu formülü kullanarak oluşturulmuştur.

$$y(t) = \mu + r(t) * \sigma \mp g * t$$

Bu formüldeki *g* oluşturulacak trendin eğiminin hangi aralıklarda değişeceğini, $r(t)$ is normal dağılıma uygun üretilmiş rassal değeri göstermektedir. Bu örnekte *g* değerleri 0.2 ile 0.5 arasında değişen ve trendi olan değerlerdir. Bu uygulamada her örnek için *g* değerleri de verilen aralıkta rasgele seçilmiştir.

6.8.2.3. Yukarı veya Aşağı Doğru Kaymanın Üretilmesi

Şekil-6.9'da örneği gösterilen "***aşağı ve yukarı doğru kayma***" şekilleri şu formülü kullanarak oluşturulmuştur.

$$y(t) = \mu + r(t) * \sigma \mp k * s$$

Bu formüldeki k kayma pozisyonunu göstermektedir. *k* katsayısı kayma pozisyonundan önce 0, ondan sonra ise 1 değerini almaktadır. Formüldeki *s* ise kayma miktarının göstermektedir. Bu uygulamada her örnek için *s* değeri 7.5 ile 20 arasında rasgele değerler seçilmiştir. Benzer şekilde *r(t)* normal dağılıma uygun üretilmiş rassal değerleri göstermektedir.

6.8.2.4. Periyodik Şeklin Üretilmesi

Şekil-6.9'da örneği verilen "***aşağı ve yukarı doğru kayma***" şekilleri şu formülü kullanarak oluşturulmuştur.

$$y(t) = \mu + r(t) * \sigma \mp \alpha * \sin(2\pi / T)$$

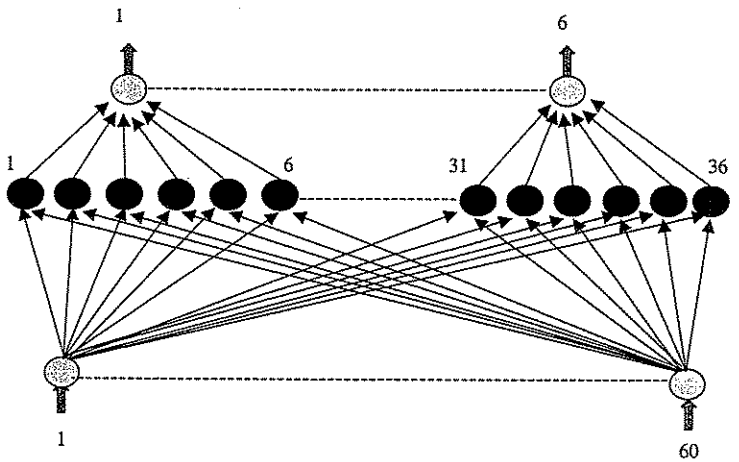
Bu formülde *r(t)* normal dağılıma uygun rassal değerleri, α periyodik değişimlerin boyutunu göstermektedir. Bu örnekte α değerleri 15 ve daha küçük değerler arasından her örnek için rasgele alınmıştır. *T* ise toplam veri sayısı olup bu uygulamada 60 olarak alınmıştır. Yani her bir örüntü örneği 60 sıcaklık değerinden oluşmaktadır.

Yukarıda anlatılan formüller kullanılarak her birinden 250 adet olmak üzere toplam 1500 örnek oluşturulmuştur. Bu örnekler tek tek gözden geçirildikten sonra yanıltıcı özellikte olanlar ve rasgele değerlerden dolayı her iki sınıfta da olabilecek nitelikte üretilmiş olan örnekler ayıklanmış ve onların yerine yeni örnekler oluşturulmuştur. listeye alınmamıştır. Bu 1500 örnekten oluşan set ikiye ayrılmıştır.

- Eğitim seti: her şekilden 83 örnek olmak üzere toplam 498 örnekten oluşmaktadır.
- Test seti: her şekilden 167 örnek olmak üzere toplam 1002 örnekten oluşmaktadır.

6.8.3. LVQ Ağının Oluşturulması

Bu uygulama için geliştirilecek olan LVQ ağı, kontrol şeması üzerindeki 6 şekli tanınması istendiğinden bu 6 şekli gösteren 6 çıktı ünitesi belirlenmiştir. Her bir örüntü 60 sıcaklık değerinin ölçülmesi ile oluşturulduğundan girdi ünitelerin sayısının 60 olmasına karar verilmiştir. *Kohonen* katmanında değişik sayılar denenmiş ve her şekli 6 ünitenin temsil etmesinin uygun olacağı görülmüştür. Şekil-6.10'da gösterilen topolojide bir ağ kurulmuştur.



Şekil-6.10. Zeki kalite sistemi (XPC) için LVQ ağı topolojisi

Şekilden görüldüğü gibi girdi katmanındaki 60 proses elemanının hepside *Kohonen* katmanındaki bütün elemanlara bağlıdır. *Kohonen* katmanındaki her 6 ünite ise 1 çıktı katmanı elemanına bağlıdır. Şekildeki rakamlar her katmandaki proses elemanı sayısını göstermektedir. Oluşturulan ağın diğer parametreleri Tablo-6.1'de gösterildiği gibi belirlenmiştir.

Tablo-6.1. Zeki kalite sistemi için geliştirilen LVQ ağının parametreleri

Parametre	Değeri
Girdi Ünitesi sayısı	60
Çıktı ünitesi sayısı	6
Her kategori (şekil) için belirlenen *Kohonen* katmanı proses elemanı sayısı	6
Kohonen katmanı proses elemanı sayısı	36
Başlangıç değerleri	-0.1 ile 0.1 aralığında rasgele değerler
Girdilerin ölçeklendirilmesi	0-1 arasında
Öğrenme katsayısı	0.05
B sabit değeri	0.0001
C sabit değeri	10

6.8.4. LVQ Ağının Eğitilmesi

Geliştirilen model değişik LVQ yaklaşımları ile eğitilmiştir. Elde edilen sonuçlar şu şekilde özetlenebilir.

Standart LVQ

Yapılan ilk denemelerde problemin öğrenilmesi çok zor olmuş ve sonuçlar istenilen başarı düzeyine çıkamamıştır. LVQ ağının daha önce belirtilen bütün sorunları bu örnekte yaşanmıştır. *Kohonen* katmanında bazı proses elemanlarının çok sık kazandığı görülmüştür. Başlangıç değerlerinin atanmasının da ağın performansı üzerindeki etkisi çok önemli görülmüştür. Ağırlıklara tesadüfi olarak atayınca öğrenmesi mümkün olamamıştır. Öğrenme performansı %60'ın altında olmuştur. Sorunları çözmek için bazı yöntemler denenmiştir. Öğrenme setindeki bazı girdi vektörleri ağırlık vektörlerinin başlangıç değerleri olarak atanmıştır. Bu durum öğrenme performansının artışına neden olmuştur. Öğrenmeye devam edilmiş ve tüm örnekler 70 defa ağa gösterildikten sonra (70 *epoch* sonra) öğrenme setinde ağın performansı %95. 18 test setinde ise ağın performansı %92.31 olmuştur. Yani 1002 tane test örneğinden yaklaşık 923 tanesi doğru olarak sınıflandırılmıştır. Öğrenmeye devam edilse bile, bu sonucu daha fazla iyileştirmek mümkün olmamıştır.

Burada öğrenme ve test setleri karşısında ağın performansı şu şekilde hesaplanmıştır.

$$\text{Öğrenme Performansı} = \frac{\text{Öğrenme setinde doğru sınıflandırılan örnek sayısı}}{\text{Öğrenme setindeki toplam örnek sayısı}} \times 100$$

$$\text{Test Performansı} = \frac{\text{Test setinde doğru sınıflandırılan örnek sayısı}}{\text{Test setinde toplam örnek sayısı}} \times 100$$

LVQ2

LVQ2 ağını eğitirken de ağırlıkların başlangıç değerlerinin belirlenmesinin performans üzerindeki etkisi çok önemli olmuştur. Performansın artmasına neden olduğundan, eğitim setindeki ilk örnekler ağırlık vektörlerine (referans vektörlerine) başlangıç değerleri olarak atanması sağlanmış ve ağın öğrenmesi gerçekleşmiştir. 4 *epoch* (her örneğin 4'er defa gösterilmesi) sonrasında LVQ2 metodunun daha fazla öğrenemediği görülmüştür. Çünkü LVQ2'nin şartları daha fazla sağlanamaz olmuştur. Bu durumda ağın öğreıme performansı %94.31 iken test setinin performansı %89.62 olmuştur. Ağın önce standart LVQ ile 70 *epoch* eğitilmesi ve sonra LVQ2 ile eğitilmesi uygulanması sonucu ise eğitim performansında bir artış görülmüştür. Öğrenme performansı

%96.18 olurken test seti üzerindeki performansı %92.61'e yükselmiştir. Eğitim ve test setleri üzerindeki ağın performansları yukarıda belirtilen formüller kullanılarak hesaplanmıştır.

Ceza Uygulanmış LVQ

Bu öğrenme metodu ile, Standart LVQ yöntemine bir cezalandırma uygulanması sonucu öğrenme performansı yükselmiştir. Bu, aynı zamanda başlangıç değerlerinin ağın performansı üzerindeki etkisini de düşürmüştür. Eğitim setinden örneklerin değerlerinin ağırlık vektörlerine başlangıç değeri olarak atanması zorunluluğu ortadan kalkmıştır. Aynı topoloji ile oluşturulan ağ cezalandırma metodunun uygulanması ile öğrenme setinde %95.98 performansa ulaşırken test setinde bu değer %92.71 olmuştur. Karşılaştırmayı yapabilmek için bu ağda yine her örnek 70 defa ağa gösterilinceye kadar (70 *epoch*) eğitilmiştir. Ağın başlangıç değerleri değiştirilerek denemeler yapılmış ve performans değerlerinin belirtilen oranlar civarında oynadığı görülmüştür.

LVQ-X

Şu ana kadar anlatılan LVQ ağlarının hiç birisi öğrenme setindeki örneklerin hepsini öğrenememiştir. Sadece LVQ-X ağının öğrenme kuralı ile öğrenme %100 gerçekleştirmiştir. Yani LVQ-X ağı kendisine sunulan örnek setini öğrendikten sonra, öğrenme setindeki örneklerin hepsi bu ağ tarafından doğru bir şekilde sınıflandırılmıştır. Üstelik diğer ağlardan daha kısa zamanda ve daha iyi performans ile öğrenebilme yeteneğinin olduğu görülmüştür. Ağın öğrenebilmesi için başlangıç değerlerine bağlılık da ortadan kaldırılmıştır. Ağ hem örneklerin değerlerinin başlangıç değerler olarak ağırlıklara atanması ile denenmiş hem de rasgele değerlerin ağırlıklara atanması ile denenmiştir. Performansın her iki durumda da birbirine çok yakın olduğu görülmüştür. Her örneğin 10 defa gösterilmesi (10 *epoch*) sonucunda ağ, öğrenme setinin %99.39'unu test setinin ise % 96.30'unu doğru sınıflandırmıştır. Öğrenmeye devam edilmiş ve 20 *epoch* sonra öğrenme setindeki örneklerin %100'ü test setindeki örneklerin ise %97.70'i doğru olarak sınıflandırılmıştır. Bu, LVQ-X ağının diğerlerine göre performansının daha yüksek olduğu anlamına gelmektedir. Bunun nedeninin birden fazla referans vektörünün aynı anda değiştirilmesi ve her iterasyonda mutlaka doğru sınıflama yapacak ağırlık vektörlerinin girdi vektörüne yaklaştırılmasının olduğu söylenebilir.

6.8.5. Sonuçların Tartışılması

Yapılan araştırmalar ve bu bölümde anlatılan endüstriyel uygulama LVQ ağlarının endüstriyel problemlere çözümler üretebilecek yetenekte olduklarını göstermiştir. Fakat her durumda standart algoritmanın kullanılması ile istenilen performans değerini yakalamak mümkün olmayabilir. Burada anlatılan örnekte de bu durum tecrübe edilmiştir. O nedenle değişik LVQ ağları denenmiş ve özellikle LVQ-X algoritmasının uygulanmasının başarılı sonuçlar üreteceği görülmüştür. Yukarıda anlatılan sonuçların özetleri bir tablo halinde şu şekilde verilebilir (bkz. Tablo-6.2). LVQ-X metodunun cezalandırma mekanizması ile birlikte kullanılmasının performansı daha da artıracağını söylemek mümkündür.

Tablo-6.2. Farklı LVQ ağlarının karşılaştırılması

LVQ algoritması	İterasyon sayısı	Öğrenme Performansı (%)	Test Performansı (%)
Standard LVQ	70	95.18	92.31
LVQ2	4	94.31	89.62
Standard LVQ + LVQ2	74	96.18	92.61
Ceza uygulamalı LVQ	70	95.98	92.71
LVQ-X	20	100.00	97.70

Burada şekillerin geleneksel yöntemler ile sınıflandırılıp sınıflandırılamayacağının düşünülüp düşünülmediği sorulabilir. Kontrol şemaları üzerinde oluşan şekiller genel olarak gürültü içermektedirler. O nedenle bu şekillerin geleneksel istatistiksel yöntemler ve diğer sezgisel yöntemler ile sınıflandırılması zordur. Yapay sinir ağlarının genelleştirme yeteneklerinin bu gürültüleri şeklin kendisinden ayırarak şekilleri sınıflandırmaya yardımcı olduğu yapılan deneyler ile açık olarak görülmektedir. Bu çalışma daha önce anlatılan ÇKA modeli ile de gerçekleştirilmiş öğrenmenin %100 test performansının ise %95 civarında olduğu görülmüştür. Standart LVQ algoritmanın performansının düşük olması problemin LVQ ile sınıflandırılamayacağı anlamına gelmez. Çünkü diğer LVQ yöntemleri de standart LVQ algoritmasına çok ufak değişiklikler yapılarak elde edilmişlerdir. O nedenle, LVQ-X algoritması %97 gibi bir performansa ulaşılarak ÇKA ağından da başarılı sınıflandırma yapılmıştır. Geleneksel istatistiksel yöntemler ile yapılan denemelerde ise performansın %94'ün üzerine çıkarılamadığı görülmüştür.

6.9. Özet

LVQ ağları destekli öğrenme stratejisini kullanırlar. Eğitim sırasında ağa sadece öğrenilmesi istenen girdiler verilmekte ve ağın çıktıyı kendisi üretmesi istenmektedir. Ağ çıktısını ürettikten sonra, ağa sunulan girdi vektörüne karşılık gelen ağın ürettiği çıktının doğru veya yanlış olup olmadığı söylenmektedir. LVQ ağının temel felsefesi ağa sunulan girdi vektörünü problem uzayını temsil eden referans vektörlerinden birisine haritalamaktır. LVQ ağı, 3 katmandan oluşmaktadır. Bunlar:

- Girdi katmanı
- *Kohonen* katmanı
- Çıktı katmanı

Girdi katmanındaki her eleman *Kohonen* katmanındaki her elemana bağlıdır. Girdi katmanından, *Kohonen* katmanına bağlantıların ağırlıkları bir referans vektörünü oluştur. Öğrenme sırasında sadece bu referans vektörlerinin değerleri (ağırlık değerleri) değiştirilirler. Her iterasyonda sadece tek bir vektörün değerleri değiştirilir. Öğrenmenin başarısı ise bu vektörlerin başlangıç değerleri ile yakından ilgilidir. *Kohonen* katmanındaki elemanların her biri çıktı katmanındaki sadece tek bir elemana bağlıdır.

Kohonen katmanı ile Çıktı katmanı arasındaki ağırlıklar sabit olup değerleri 1'dir. Bu ağırlıkların değerleri eğitim sırasında değiştirilmez. LVQ ağlarında öğrenme girdi vektörü ile referans vektörleri arasındaki öklid mesafesine dayanmaktadır. *Kohonen* katmanındaki her eleman bir referans vektörünü göstermekte olup birbirleri ile yarılırlar. Öklid mesafesi en kısa olan eleman yarışmayı kazanır. Yarışmayı kazanan elemanın çıktısı 1 diğerleri ise 0 değerini alır. Yarışmayı kazanan çıktı elemanı ilgili girdinin sınıfını gösterir. Eğer girdi doğru sınıflandırılmış ise ilgili referans vektörü girdi vektörüne yaklaştırılır. Aksi halde ise uzaklaştırılır. Yaklaştırma ve uzaklaştırma öğrenme katsayısı ile gerçekleştirilmektedir. LVQ ağının en önemli problemi aynı vektörün çok sık kazanması ve ağın öğrenme performansının düşük olmasıdır. Bu sorun zamanla öğrenmenin ters yönde öğrenmemeye dönüşmesine neden olmaktadır. Bunu önlemek için öğrenme katsayısı zaman içinde azaltılarak sıfır değerine kadar düşürülmesi ve öğrenme tamamlanınca sıfır değerini alarak öğrenmeyi durdurması sağlanmaktadır. Bu sorunu çözmek için yeterli olmadığından, öğrenme algoritması da geliştirilmiştir. LVQ2 öğrenme yöntemi sınıflar arasında kalan örnekleri sınıflandırabilecek yeteneğe sahiptir. Bunun yanında çok kazanan proses elemanlarını cezalandırarak diğer proses elemanlarında şans verilmesi için başka bir yöntem önerilmiştir. Diğer bir yöntem de ise, eğitim sırasında her iterasyonda birden fazla referans vektörünün ağırlıklarını değiştirmek ve yanlış sınıflandırılan örnekler referans vektörlerinden uzaklaştırılırken doğru sınıfın bir örneği referans vektörüne yaklaştırılmaktadır. LVQ-X adı verilen bu yöntem hem başlangıç değerlerine bağlılığı ortadan kaldırmış, hem öğrenme %100 gerçekleştirilmiş, hem eğitim performansı yükselmiş hem de öğrenmenin hızı artmıştır. Bu avantajlar XPC adı verilen zeki istatistiksel kalite kontrol sisteminde gerçekleştirilen örnek uygulamada LVQ-X algoritması, sadece diğer LVQ ağlarından değil aynı zamanda hem ÇKA ağından hem de geleneksel istatistiksel yöntemlerden daha başarılı bir öğrenme gerçekleştirmiştir.

6.10. Kaynakça

[1] Kohonen T. (1984), Self organization and associative memory, Springer Verlag, New York.

[2] Kohonen T., Barna G., Chrisley R. (1987), Statistical pattern recognition with neural networks: Benchmarking studies, Proc. Of the Int. Joint Conf. On Neural Networks, San Diago, California, Vol. 1, pp. I-66 – I-68.

[3] Desiono D., (1987) Addind a conscience to competitive learning, Proc. Of the Int. Jonit Conference on Neural Networks, San Diago, California, Vol. 1, pp. I-117 – I-124.

[4] Öztemel E. (1992), integrating expert system and neural networks for intelligent on-line statistical process control, Universtiy of Wales, College of Cardiff, Doktor tezi.

[5] Pham D.T., Öztemel E. (1996), "Intelligent Quality Control", Springer Verlag.

BÖLÜM 7

YAPAY SİNİR AĞI MODELİ
(ÖĞRETMENSİZ ÖĞRENME)
ADAPTİF REZONANS TEORİ (ART) AĞLARI

Bu bölüme kadar öğretmenli ve destekli öğrenmeye dayalı ağlar örneklerle ele alındı. Bu bölümde ise öğretmensiz öğrenmeye dayalı ART ağlarına değinilecektir. Bu konuyu iyi anlamak için öncelikle hafıza ve bilgilerin hafızada saklanması kavramları açıklanacaktır. Çünkü burada ağ dışarıdan herhangi bir destek almaksızın örneklere bakarak bilgileri kendisi keşfetmek ve hafızasında saklamak durumundadır.

7.1. Hafıza (Bellek) Kavramı

ART ağlarında öğrenme doğru bilgilerin belirlenerek hafızaya alınması anlamına gelmektedir. Öğrenme sırasında kullanılan örneklerden öğrenilen bilgilere dayanarak daha sonra görülmemiş örnekler hakkında yorumlar yapılabilmektedir. Bilgilerin hafızada saklanması ve hafızada tutulması ise iki şekilde olmaktadır:

- ***Kısa Dönemli Hafıza** (KDH):* Bilgilerin geçici olarak tutulduğu ve zaman içerisinde yok olduğu ve yerlerine başka bilgilerin saklandığı hafızadır. Bu insanda da böyledir. Çoğu insan dün akşam ne yediğini hatırlamayabilir. Çünkü bu bilgiler kısa dönemli hafızada tutulur ve zamanla başka olayların etkisi ile unutulurlar.
- ***Uzun Dönemli Hafıza** (UDH):* Bilgilerin sürekli tutulduğu ve kolay kolay unutulmadığı hafızadır. Bilginin silinmesi için çok uzun zamanın geçmesi gerekebilmektedir. Örneğin aradan yıllar geçmesine rağmen çok sevdiğimiz bir arkadaşımızı görürsek onu hemen hatırlarız. Aradan geçen yıllar onunla ilgili bilgilerin hafızada tutulmasını önleyememektedirler. Hepimiz ilk yaş günü partimizi, ilk okula gidişimizi, ilk mezun oluşumuzu, evlilik törenimizi vb. bir çok olayı hiç unutamayız. Bilginin uzun dönemli tutulabilmesi için o bilginin bizim için öneminin olması gerekmektedir. Çünkü aradan geçen seneler bazı arkadaşlarımızın unutulmamasına neden olmakta iken bazılarının ise kısa süre sonra hatırlanamaz hale getirmektedir. Komşusunun ilk doğum günü partisini hatırlayan kaç kişi vardır. Ancak o partide çok önemli bir olay olmuş ise hiç unutulmayabilir. Ben etrafımda ölen komşulardan hiç birini hatırlamaz iken bir tanesini hiç unutamıyorum. Çünkü onun öldüğü gün dışarıdan geldiğimde herkes bizim balkonda ve evin etrafında toplanmış ve ben bizim aileden birisine bir şey oldu zannetmiştim. Ama babamın teyzesi öldüğü

halde yakın akrabam olmasına rağmen bende o kadar etki yapmamıştır. Bilgilerin hafızada kalma süresi beyinde oluşturdukları etkiye göre de değişmektedir.

ART ağlarında bilgiler ileride anlatılacağı gibi hem kısa dönemli hem de uzun dönemli hafızada saklanmaktadır. Yani bazı bilgiler öğrenilirken bazı bilgiler de unutulmaktadır. Zaman içerisinde öğrenilen bilgiler ağın ilgili olay hakkında karar vermesine neden olmaktadır.

7.2. ART Ağları

ART ağları Grosberg'in 1976 yılında biyolojik beynin fonksiyonlarına yönelik olarak yaptığı çalışmalar neticesinde ortaya çıkmıştır [1,2]. Kendisi çalışmaları neticesinde beynin çalışmasını açıklayacak bir model önermiştir. Bu modelin 3 temel özelliği vardır. Bunlar:

❶ ***Normalizasyon:*** Bu, özellikle biyolojik sistemlerin çevredeki büyük değişikliklere karşı adaptif olduklarının durumu göstermektedir. Örneğin insanın çok fazla gürültülü bir ortamda bir süre sonra gürültüden rahatsız olmaması sisteme adapte olunduğunu ve çevredeki olayların normalize edildiğini göstermektedir. Tren yoluna yakın yaşayan insanların gürültü anlayışlarında tren sesi gürültü sayılmamaktadır. Günde defalarca geçen tren kimseyi rahatsız etmez iken, başka küçük bir gürültü herkesin dikkatini çekebilmektedir. Bazı insanlar tren yoluna yakın yaşayanların sese karşı duyarsızlıklarını anlatmakta güçlük çekmektedirler. Halbuki bu insanlarında bazıları minibüs yoluna yakın yerde oturmakta ve korna sesinden etkilenmemektedirler.

❷ ***Ayrıştırabilme***: İnsanın karar verebilmesinde ve olayları yorumlayabilmesinde çevredeki olaylar arasında var olan fakat görülmesi zor farklılıkları ayrıştırmak çok önemlidir. Bazen küçük ayrıntılar hayati öneme sahip olabilir. Uyuyan bir aslan ile saldırıya hazır bir aslanın fark edilmesinin önemi açık olarak ortadadır. Yatan bir aslanın vücudunun hareketlerinin fark edilmesi çok önemli bir tehlikeyi önleyebilir. Biyolojik sistemlerin böyle ayrıntıları fark etmeleri çok önemli bir özellikleridir.

❸ ***Ayrıntıların saklandığı kısa dönemli hafıza***: Belirlenen farklılıklar ve çevresel olaylar davranışlara neden olmadan önce hafızada saklanmakta ve daha sonra eyleme dönüşmektedir. Bu uzun dönemli hafızada değişikliklere neden olmaktadır. Hafızadaki her olay uzun süre etkili olmamakla beraber, sürekli aynı şeyleri tekrar etmek sonucu olaylar unutulmaz hale gelebilmektedir. Karşılaşılan ani olaylar karar vermede öncelikli olabilmektedir. Fakat uzun karar vermede uzun dönemli hafızadaki bilgiler daha etkili olmaktadır. Anlık kararlar bazen olumsuz sonuçlar doğurmaktadır.

Grosberg, bu özelliklerden yola çıkarak arkadaşları ile Adaptif Rezonans Teorisi ağı (ART) adını verdiği yapay sinir ağları setini oluşturmuştur. ART ağları daha sonra geliştirilen öğretmensiz öğrenme ağlarının geliştirilmesine de temel olmuştur. Grosberg'in önerdiği ART ağlarının en temel özelliği sınıflandırma problemleri için geliştirilmiş olmalarıdır.

Sınıflandırma günümüzde en çok karşılaşılan problemlerin başında gelmektedir. Çevremizdeki bir çok nesneyi de bizler sınıflandırmış durumdayız. Örneğin hayvanları cinslerine göre (köpek, kedi, at vb.) sınıflandırmışızdır. Kedi cinsinden ise bir çok farklı örnek görmek mümkündür. Kedi sınıflaması ile kedilere ait genel özellikler dile getirilmektedir. Bu genel özellikler düşünülerek kediler ile ilgili yorumlar yapılabilmektedir. Örneğin bütün kediler tırmalama özelliğine sahiptir diyoruz. Benzer şekilde endüstriyel kuruluşlarda makineler üzerinde oluşacak olan hataları sınıflandırmakta ve her sınıfa giren hatalar için o sınıfa özel çözümler üretilmektedir. Meslekler değerlendirilirken de sınıflandırılmaktadır. Mühendislik problemlerinin çoğu da sınıflama problemi haline getirilerek çözülmektedir. Sınıflandırma hayatımızda bu kadar önemli bir yer tuttuğundan sınıflandırma yapabilen yapay sinir ağları da önemli bir yer tutmaktadır. ART ağları bu amaçla geliştirilmiş ve başarılı bir şekilde kullanılabilen bir yöntem olarak bilim dünyasında kabul görmüştür. Bir önceki bölümde anlatılan LVQ ağları da sınıflandırma için kullanılmaktadır. ART ağlarının LVQ ağlarından farkı ise yapılacak olan sınıflandırma ile ilgili olarak ağa herhangi bir bilginin verilmeyişidir. ART ağları sınıflandırmayı kendi başlarına yapmaktadırlar.

ART ağlarının bilim dünyasında bu derece önemli olmasının nedenlerinden birisi de, nedeni yapısal olarak insan beyninin davranışları ve sinir sistemi hakkında bilinen (veya varsayılan) bulgular üzerine kurulmuş olmalarıdır. Bu ağlar, memeli hayvanların beyinlerinin çalışma prensipleri dikkate alınarak geliştirilmiştir. Beynin kullandığı sezgisel yaklaşımları matematik bir modele dönüştürmelerinden dolayı da bu ağlar oldukça yoğun bir ilgi görmüştür.

7.3. ART Ağlarının Diğer Yapay Sinir Ağlarından Farkları

ART ağlarının diğer yapay sinir ağlarından farklılıklarının iyi anlaşılması da bu ağlardan daha iyi faydalanabilmek için önemlidir. Bilinen diğer ağlardan özellikle en yaygın olarak kullanılan çok katmanlı algılayıcılardan temel farklılıkların bazılarını, Grosberg [3] şu şekilde özetlemektedir:

- ART ağları gerçek zamanlı olarak oldukça hızlı ve kararlı bir şekilde öğrenme yeteneklerine sahiptirler. Bu yetenek bir çok ağda yoktur. ART ağları bu özellikleri ile gerçek zamanlı kullanılabilen donanımla da desteklenerek gerçek zamanlı öğrenebilen bilgisayarların oluşmasına yardımcı olmaktadırlar.
- Gerçek zamanda ortam genel olarak durağan değildir. Olayların oluşumu her an beklenmedik olaylar ile değişebilmektedir. Bununda ötesinde gerçek zamanlı olaylar sürekli devam etmektedir. ART ağları bu durağan olmayan dünyada sınırsız karmaşıklık altında çalışabilme yeteneğine sahiptirler. Diğer ağların çoğu ise durağan olarak çevrimdışı (*off-line*) öğrenip çalışırlar. Esneklikleri yoktur. Ortama anında uyum sağlamaları çok sınırlıdır.
- ART ağları beklenen çıktıları bir öğretmenden almak yerine kendi kendine öğrenmeye çalışır.
- ART ağları ağa sunulan farklı nitelikteki ve değişik durumlardaki örnekler karşısında kendi kendilerine kararlı (*stabil*) bir yapı oluşturabilirler. Ağa sunulan,

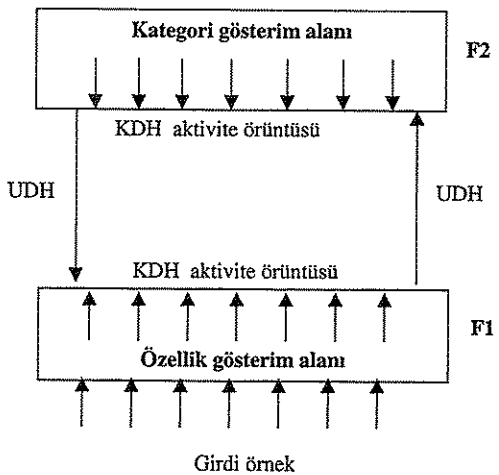
yeni bir girdi geldiği zaman ya bilinen sınıfların kodlarına (sınıflarına) ulaşabilecek şekilde ağda iyileşmeler yapılır yada yeni kod (sınıf) oluşturulur. Bu ağın büyümesine de neden olabilir ve ağın bütün kapasitesini kullanana kadar devam eder.

- ART ağları çevredeki olayları sürekli öğrenmeye devam eder. Uzun dönemli hafızada bulunan ağırlıklar sürekli olarak gelen girdi değerlerine göre değişmeye devam ederler.
- ART ağları girdi değerlerini otomatik olarak normalize ederler. Çok fazla ve oldukça düşük orandaki gürültülerin girdi işaretindeki etkileri ortadan kaldırılmış olur.
- ART ağlarında hem aşağıdan yukarı hem de yukarıdan aşağıya ağırlık değerleri vardır. Özellikle yukarıdan aşağıya ağırlıklar sınıfları temsil etmektedirler. Bunları ağ kendisi girdilere bağlı olarak otomatik olarak belirlemektedir. Bu ağırlıklar aynı sınıftan olan bütün örneklerin ortak yönlerini içermektedir. Bu ağırlıklardan oluşan örüntülere *kritik özellik örüntüleri* denmektedir [4]. Yukarıdan aşağıya ağırlıklar ağın öğrendiği beklentileri (beklenen girdi temsilcileri) göstermektedir. Bu değerler aşağıdan yukarı gelen bilgiler ile karşılaştırılarak eşleme yapılır. Aşağıdan yukarı gelen bilgiler ile karşılaştırma kısa zamanlı hafızada (KDH) oluşmaktadır. Aşağıdan yukarı ve yukarıdan aşağı ilişkiler bir ART ağında kapalı çevrimi tamamlamaktadır.
- Bu kapalı çevrimden dolayı Yukarıdan aşağı ağırlıklar KDH'da yapılan karşılaştırma ile Kazanç faktörünü kullanarak aynı kategoride olmayan girdilerin o kategoriye girmesini önlemektedir. Böylece kategoriyi gösteren ağırlıkların gerçek zamanlı gelen farklı bir girdiden etkilenmeleri önlenmektedir. Böyle bir kontrol yapılmaması gelen her girdi değerinin ağırlıkları değiştirerek önceden öğrenilen bilgilerin kayıp olmasına neden olacaktır. ART bu özelliği ile sürekli öğrenmeyi desteklemekte ve önceden öğrenilenler ancak aynı gruptaki başka örneklerin yeni özellikleri olunca değiştirilmektedir. Bu özellik ise yakın eşleme (*approximate match*) olarak bilinmektedir.
- ART ağlarının yakın eşleme özelliğinden dolayı hem hızlı hem de yavaş öğrenebilme yetenekleri vardır. Hızlı öğrenme Uzun Dönemli Hafızada (UDH) bir denemede yeni bir dengenin (*equilibrium*) oluşturulması ile gerçekleştirilir. Yavaş öğrenme ile bir dengenin oluşması için birden çok denemenin yapılması durumu kastedilmektedir. Halbuki çok katmanlı algılayıcılar gibi ağlarda osilasyonları önlemek için özellikle yavaş öğrenme zorunluluğu vardır.

1976 yılından bugüne kadar değişik ART ağları tanımlanmıştır. Bunlar arasında ART1, ART2, ART3 [7], ARTMAP[8], Fuzzy ART [9,10] gibi ağları saymak mümkündür. Bu ağların hepsi aslında aynı temel felsefeye dayanmakta ve çok az farklılıklar göstermektedir. En yaygın olarak kullanılan ART1 ve ART2 ağlarıdır. O nedenle bu bölümde bu iki ağ ayrıntılı olarak tanıtılacaktır. Diğerleri ile ilgili bilgiler ise verilen kaynaklarda bulunabilir. Öncelikle ART ağların genel yapısı ve çalışma ilkesi anlatılacaktır; daha sonra bu modeller tanıtılacaktır.

7.4. ART Ağlarının Yapısı

Adaptif Rezonans Teorisi (ART) ağları genel olarak iki katmandan oluşmaktadır. Bu katmanlar F1 ve F2 olarak isimlendirilmiştir. F1 katmanı girdinin özelliklerini gösterirken F2 katmanı kategorileri (ayrıştırılmış sınıfları) göstermektedir. Bu iki katman birbirlerine UDH ile bağlanmaktadırlar. Girdi bilgileri F1 katmanından alınır ve sınıflandırma ise F2 katmanında yapılır. Modelin genel yapısı Şekil-7.1'de verilmektedir.



Şekil-7.1. ART ağının genel yapısı

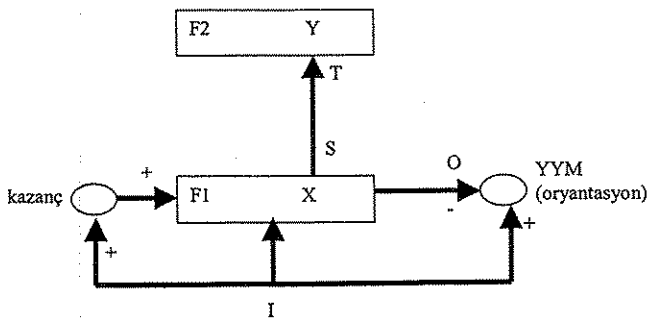
ART ağlarında girdiler direkt olarak sınıflandırılmazlar. Öncelikle girdilerin özellikleri incelenerek F1 katmanının aktivasyonu belirlenir. UDH'da ki bağlantı değerleri ile gelen bilgiler kategorilere ayrılarak F2 katmanına gönderilir. F2 katmanındaki sınıflandırma ile F1 katmanından gelen sınıflandırma birbirleri ile eşleştirilerek, eğer örnek belirlenmiş bir sınıfa uyuyorsa o kategoride gösterilir. Aksi taktirde, ya yeni bir sınıf oluşturulur veya girdinin sınıflandırılması yapılmaz.

7.5. ART Ağlarının Çalışma Prensibi

Daha önce belirtildiği gibi ART ağları F1 katmanından gelen bilgileri F2 katmanındaki kategorilere eşleştirmektedir. Bu eşleşme sağlanamaz ise yeni bir kategori oluşturulmaktadır. ART ağlarının çalışması iki yönlü olmaktadır:

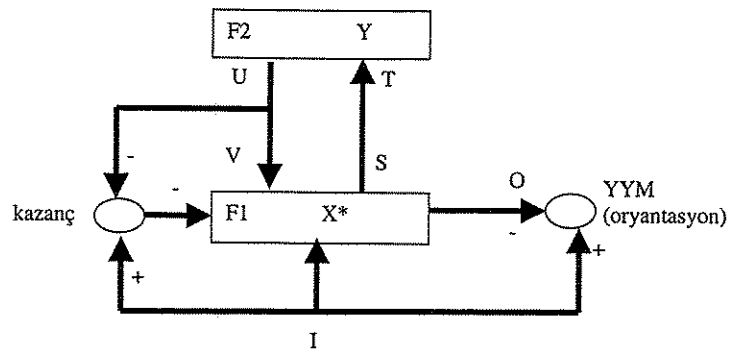
- Aşağıdan yukarı (F1 den F2'ye) bilgi işleme
- Yukarıdan aşağı (F2 den F1'e) bilgi işleme

ART ağlarında aşağıdan yukarı bilgi işleme prensibi Şekil-7.2'de gösterilmiştir. Şekilden de görüldüğü gibi, bir girdi örüntüsü (I) ağa gösterilir. Bu örüntü hem F1 katmanında KDH da X aktivite örüntüsünü oluşturur hem de oryantasyon sistemini veya diğer bir deyişle yeniden yerleştirme modülünü (YYM) aktif etmek üzere bir işaret (*signal*) gönderir. Benzer şekilde oluşturulan X örüntüsü hem YYM'ne bir men-edici (*inhibitory*) işaret (O) göndermekte hem de F1 katmanından bir çıktı örüntüsü (S) oluşturmaktadır. S sinyali F2 katmanına giden bir girdi örüntüsüne (T) dönüştürülür. Bu girdi örüntüsü ise F2 katmanının çıktısı olan örüntüyü (Y) oluşturur. Bu aynı zamanda ağında çıktısıdır. Bu şekilde aşağıdan yukarı (F1 katmanından F2 katmanına) bilgi işleme tamamlanmış olur.



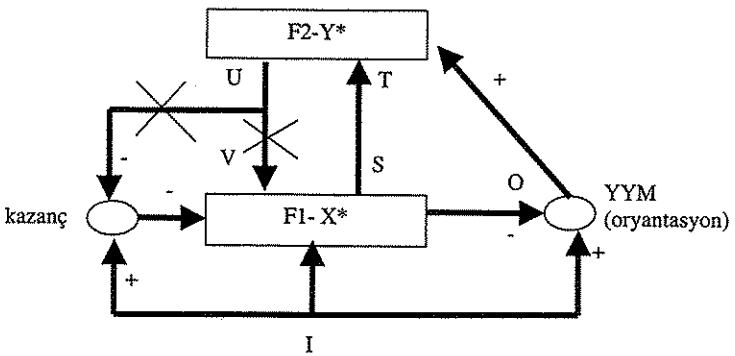
Şekil-7.2. ART ağında çıktı oluşturma süreci (aşağıdan yukarı)

Yukarıdan aşağı bilgi işleme de, benzer şekilde, Şekil-7.3'de gösterildiği gibi yapılmaktadır. Bu durumda, F2 katmanında oluşturulan çıktı örüntüsü yukarıdan aşağıya bir sinyal (U) gönderir. Bu sinyal daha sonra beklenen şablon örüntüye (V) dönüştürülür. Aynı zamanda kontrol faktörü (kazanç) için men-edici (*inhibitory*) bir işaret üretir. Bundan sonra şablon örüntünün girdi örüntüsü ile eşlenip eşlenemeyeceği sınanır. Eğer böyle bir eşleşme mümkün değilse o zaman F1 katmanında yeni bir KDH örüntüsü (X*) oluşturulur. Bu örüntü oryantasyon sistemindeki men-edici işaretin etkisini azaltır.



Şekil-7.3. ART ağında çıktı oluşturma süreci (yukarıdan aşağı)

Oluşturulan X* sinyali oryantasyon sisteminde O işaretinin men-edici etkisini azaltarak YYM'nün (oryantasyon modülünün) F2 katmanına bir sinyal göndermesini sağlar. Bu işaret F2 katmanında Y* örüntüsünü oluşturur. Böylece, Şekil-7.4'te gösterildiği gibi girilen I örüntüsü için doğru sınıfı gösteren Y* çıktısı üretilmektedir.

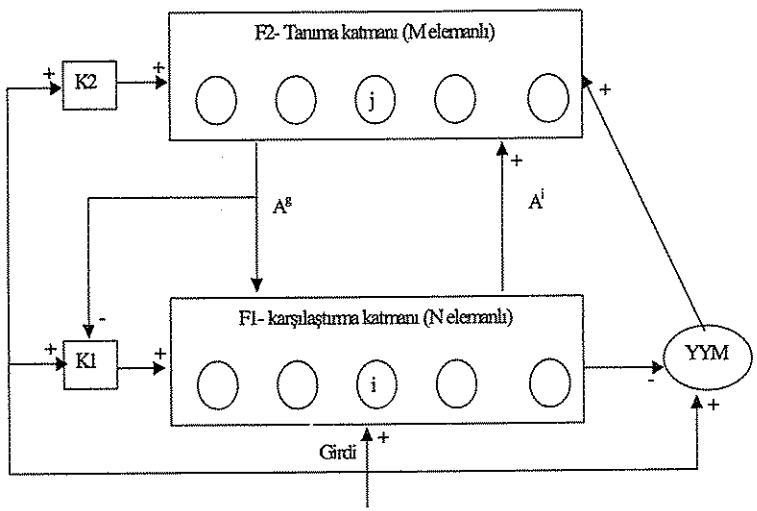


Şekil-7.4. ART ağında yeni bir sınıf oluşturma

Eğer üretilen V şablon örüntüsü ile girdi örüntüsü eşleşir ise o zaman sadece yukarıdan aşağı o girdinin sınıfını gösteren ağırlıklar değiştirilir. Bu değiştirme öğrenme kuralına göre gerçekleştirilir. Her ART modelinin öğrenme kuralı ayrıdır. Aşağıda ART1 ve ART2 ağlarının öğrenme kuralları tanıtılmıştır.

7.6. ART1 Ağı

ART1 ağı sadece ikili (*binary*) girdiler ile çalışan ve en basit ART ağının örneğidir. Geliştirilen ilk ART ağıda denilebilir. Yukarıda belirtildiği gibi iki katmandan oluşmaktadır. F1 katmanını *karşılaştırma katmanı*, F2 katmanı ise *tanıma katmanı* olarak isimlendirilmiştir. F1 katmanındaki bütün proses elemanları F2 katmanındaki proses elemanlarının tamamına bağlanmıştır. Bu bağlantılar sürekli değerlerden oluşan UDH bağlantılarıdır (burada A^i ile gösterilmiştir). Bu bağlantıların özellikleri ileri doğru bağlantılar olmalıdır. Aynı zamanda F2 katmanından F1 katmanına geriye doğru ikili değerleri olan bağlantılar vardır (burada A^g ile gösterilmiştir). Modelin K1 ve K2 olarak isimlendirilen iki tane kazanç modülü ve bir tanede yeniden yerleştirme modülü (YYM) vardır. Bu modüle bazı kaynaklarda oryantasyon modülü de denmektedir. ART1 ağının geometrik gösterimi Şekil-7.5'de verilmiştir:



Şekil-7.5. ART1 modelinin geometrik gösterimi

Karşılaştırma katmanındaki her proses elemanının 3 girdisi vardır. Bunlar:

- Girdi örneğinin bir elemanı (aşağıdan yukarı)
- Geri besleme değeri (yukarıdan aşağıya)
- Kazanç değeri (K1 den gelen değer)

F1 katmanındaki proses elemanı bu üç girdiden en az ikisinin aktif olması durumunda çıktı olarak 1 değerini oluşturur. Buna *2/3 kuralı* denilmektedir.

F2 katmanındaki proses elemanları ise kendilerine gelen net girdiyi hesaplarlar. Proses elemanları birbirleri ile yarışmaktadırlar. Genellikle en fazla çıktıyı üreten proses elemanı yarışmayı kazanmaktadır.

K2 girdi örneğindeki bütün elemanlar arasında mantıksal VEYA (*OR*) operatörüdür. Sadece yeni bir sınıfın oluşmasında kullanılır.

K1 modülü girdi örüntüsünün değerlerini hafızada bulunan şablon değerler ile eşleştirmede kullanılmaktadır. Gelen bilgilerin aşağıdan yukarı veya yukarıdan aşağıya olup olmadığının bilinmesini sağlar. Aşağıdan gelen değerler bu kazanç değerini aktif yukarıdan gelenler ise pasif yaparlar.

YYM ise girdi vektörü ile F1 katmanı çıktısı (çıktı vektörü) arasındaki fark belirli bir değeri aşması durumunda F2 katmanına bir sinyal göndermekle yükümlüdür. İki vektörün arasındaki fark *benzerlik katsayısı* denilen bir değer ile karşılaştırılır. Bu katsayı girdi vektörünün hafızada bulunan çıktı vektörlerine uygunluğunu belirler. Aradaki fark benzerlik katsayısından küçük ise o zaman benzerlik yok demektir. Bu durumda girdi vektörü için yeni bir sınıf oluşturmak ve hafızaya koymak gerekir. ART1 ağının öğrenmesi bu mantık ile çalışmaktadır. Benzerlik katsayısı aşağıda açıklanmış olup kullanıcı tarafından belirlenmektedir.

7.6.1. ART 1 Ağının Eğitilmesi ve Öğrenmesi

ART konusunda yukarıdaki açıklamaları dikkate alarak öğrenme olayı basitleştirilmiş ve aşağıda adım adım açıklanan öğrenme kuralı geliştirilmiştir.

Adım 1: *Ağırlıklara başlangıç değerleri atanır;*

F1 katmanından ağa sunulan N adet örnek, F2 katmanında ise M adet çıktı proses elemanı olduğu varsayılsın. Burada $M \geq N$ 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 aşağıda anlatılacak olan benzerlik katsayısının önemi artmaktadır. Daha önce belirtildiği gibi bu katsayı girdilerin hangi sınıfın üyesi olduğunu belirlemede kullanılmaktadır. ART ağlarının başlangıç değerlerinin atanmasında da iki durum söz konusudur. Bunlar:

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

$$A^g_{ji} = 1 \qquad \text{j: } 1,2,3,.....M$$

i: 1,2,3.....n

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 proses elemanı sayısı n olmak üzere ise şu şekilde atanmaktadır.

$$A^i_{ij} = \frac{1}{1+n}$$

Adım 2: *Benzerlik katsayısının (ρ) değeri belirlenir;*

Benzerlik katsayısı ρ ile gösterilmekte olup 0 ile 1 arasında bir sayıdır. Bu katsayı iki vektörün aynı sınıfın elemanı sayılabilmesi için birbirlerine ne kadar benzemesi gerektiğini belirler. Örneğin, bu değerin 0.8 olması benzerliğin %80 olması aynı sınıfın elemanı olmak için yeterli görülüyor demektir. Yani %100 benzerlik yerine %80 benzerlik kabul edilmektedir. %20'lik bir farklılığa izin verilmektedir. Bunun altında bir benzerlik aynı grubun elemanı olmak için yeterli görülmemektedir.

Benzerlik katsayısı ne kadar büyük alınırsa farklı sınıf sayısı da o kadar çoğalır. Bu sayı düşük alınırsa o zaman sınıf sayısı da az olur. Çünkü benzerlik için vektörler arasındaki farklılığa daha çok izin verilmektedir. Böylece daha çok örnek aynı sınıfın üyesi olabilmektedir. Yalnız burada bir noktaya dikkatleri çekmekte yarar vardır. Sınıf sayısının az olması durumunda ağın öğrenme performansı düşük olabilir. Sınıf sayısının az veya çok olmasından çok öğrenilmesi istenilen olayı doğru temsil edebilen sınıf sayısını oluşturacak şekilde benzetim katsayısını belirlemektir.

Adım 3: *Girdi setinden bir örnek ağa gösterilir;*

Girdi setindeki örnek X ($x1$,$x2$,.....xn) vektörü olarak (her bir elemanı x_i olarak tanımlanmış şekilde) ağa gösterilir.

Adım 4: *F2 katmanındaki proses elemanlarının çıktılarının hesaplanması;*

F2 katmanındaki her proses elemanının çıktı değeri şu şekilde hesaplanmaktadır:

$$y_j' = \sum_i^N A_{ij}^i(t)x_i \qquad \text{j:1,2,.....M}$$

Adım 5: *Kazanan elemanın seçilmesi;*

Kazanan eleman en büyük (maksimum) çıktıya sahip proses elemanıdır. Bu elemanın sahip olduğu ağırlık vektörüne en uygun sınıf (katagori) gösterim vektörü denmektedir. Bunun *k.* proses elemanı olduğu varsayılırsa kazanan elemanın çıktısı,

$$y_k^* = \max\cdot(y_j')$$

olacaktır. Burada * işareti kazanan elemanı temsil etmektedir. Bu elemanın ağırlık vektörü girdi vektörü ile karşılaştırılarak benzetim katsayısına göre uygunluk sınaması yapılacaktır.

Adım 6: *Uygunluk testinin yapılması;*

Burada kazanan elemanın ağırlık vektörünün ile girdi vektörünü temsil edip edemeyeceğine karar verilmektedir. Bunun için önce girdi vektöründe bulunan 1 sayısı ($s1$) belirlenir.

$$s1 = |X| = \sum x_i$$

Daha sonra kategori gösterimi vektörü (kazanan elemanın ağırlık vektörü) ile girdi vektörünün uyuştuğu 1 sayısı ($s2$) bulunur. Bunu veren formül ise şöyledir:

$$s2 = \left|A_k^g . X\right| = \sum A_{jk}^g x_i$$

Eğer, $\frac{S2}{S1} \geq \rho$ ise o zaman iki vektör birbirinin benzeri kabul edilir. Yani kategori gösterim vektörü girdi vektörünü temsil edebiliyor demektir. Bu durumda ağırlıklar şu şekilde değiştirilir.

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

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

Eğer; $\frac{S2}{S1} < \rho$ ise o zaman maksimum çıktıyı veren proses elemanının çıktısı 0 olarak atanır ve ondan sonraki maksimum çıktıyı veren (ikinci en büyük çıktıyı üreten) proses elemanı seçilerek 4üncü adımdan itibaren işlemler tekrar edilir. Girdi vektörü bu sınıfın elemanı olarak atanamaz ise üçüncü en büyük çıktıyı veren proses elemanının sınıf (kategori) gösterim vektörü ile benzerliğine bakılarak benzerlik olması durumunda ağırlıklar değiştirilir. Başlangıçta birinci örnek birinci sınıfın temsilcisi olarak atanır. İkinci örnek birinci ile aynı sınıfından ise (benzer ise) birinci sınıfın elemanı sayılır. Yoksa bu örnek ikinci sınıfın temsilcisi olur. Böylece örneklerin hepsi ya var olan sınıflardan birisine girer ve ağırlık değerleri değiştirilir. Ya da yeni bir sınıfın temsilcisi olur. En kötü olasılık ile N örnek için N adet sınıf oluşturulabilir *(N=M)*. Bu şekilde yukarıdaki işlemler bütün girdi vektörleri sınıflandırılıncaya kadar devam eder...

7.7. ART2 Ağı

ART2 modeli 1986 ve 1987 yıllarında Carpenter ve Grosberg [5,6] tarafından geliştirilmiştir. ART1 algoritmasının geliştirilmiş halidir. ART1 algoritması sadece ikili (*binary*) girdiler ile çalışabilmektedir. ART2 ağı ise hem ikili hem de sürekli girdi değerleri ile çalışabilme yeteneğine sahiptir. Her iki modelin arasındaki en önemli yapısal fark ART 2 ağında F1 katmanında 3 alt sistemin bulunmasıdır. Bunlar:

- Aşağıdan yukarı (F1 katmanından F2 katmanına) girdi değerlerini okuyan alt sistem
- Yukarıdan aşağıya (F2 katmanından F1 katmanına) girdileri okuyan alt sistem
- Bu ikisini birbirleri ile eşleştiren alt sistem

7.7.1. ART2 Ağının Yapısı

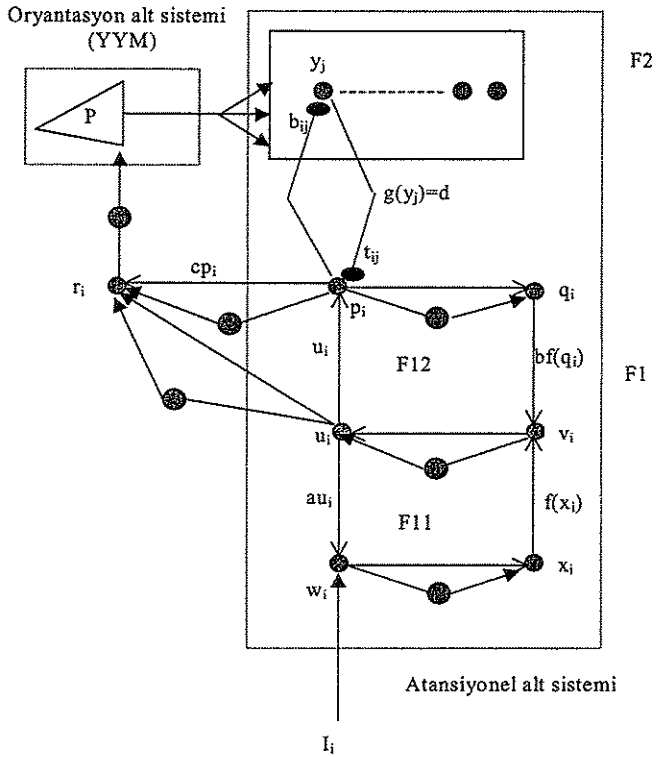
ART2 ağının yapısı Şekil-7.6'da verilmiştir. Bu ağ da ART 1 ağı gibi iki alt sistemden oluşmaktadır:

- Oryantasyon (Yeniden Yerleştirme Modülü) alt sistemi
- Atansiyonel alt sistemi

Atansiyonel alt sistemi ise F1 ve F2 olmak üzere iki katmandan oluşmaktadır. Bunlardan F1 katmanı ***"özellik gösterim katmanı"*** F2 ise ***"kategori veya sınıf gösterim katmanı"*** olarak belirlenmiştir. F1 katmanında iki tür proses elemanı vardır. Bunlar:

- *Doğrusal elemanlar*: Ağın girdilerini toplayan elemanlardır. Şekil üzerindeki p_i, v_i, w_i bu elemanları göstermektedir.
- *Paralel elemanlar*: Bunlar ise ağın girdilerini normalize eden elemanlardır. Şekildeki q_i, x_i, u_i bu elemanları göstermektedir.

F1 ve F2 katmanları arasındaki bağıntılar uzun dönemli hafızada (UDH) tutulan değerler, proses elemanlarının çıktıları ise kısa dönemli hafızada tutulan değerlerdir. Ağın hem F1 hem de F2 katmanlarında kısa dönemli hafızası vardır. Oryantasyon (Yeniden yerleştirmek modülü) katmanı ise ART1 ağında olduğu gibi sadece yukarıdan aşağı F1 girdilerinin daha önce oluşturulmuş olan sınıflara uymadığı zamanlarda aktif olmaktadır. Aksi halde çalıştırılmamaktadır. ART2 ağında da F2 katmanında çıktı ünitesi olarak en fazla eğitim setindeki örnek sayısı kadar eleman kullanılabilir. Bu her girdi örneğinin mutlaka bir sınıfın elemanı olmasını sağlamak içindir. ART1 ağında olduğu gibi ART2 ağında da kategori (sınıf) sayısı sınırlı tutulabilir. Örneğin bütün örnekler 5 sınıfta gruplandırılabilir. Bunun için benzerlik katsayısının belirlenmesinde esnek davranılması gerekir. Bu örnek sayısının çok fazla olduğu durumlarda uygulanacak bir strateji olabilir. Binlerce örnek için binlerce çıktı ünitesinin kullanılması pratik olmayabilir. Özellikle süreç kontrolü veya makine hatası izleme amaçlı geliştirilen ağlarda çıktıların (hata türlerinin) sınıflandırılıp örnekleri bu sınıfların hatayı en iyi şekilde karakterize etmesini sağlamak pratik olarak uygulanabilir. Böylece çevrimiçi (*on-line*) çalışabilir sistemler oluşturulabilir. Bu durumda belki her sınıf için yüzlerce örnek toplayıp ağı eğitmek gerekecektir; ancak, 5-10 hata sınıfı (çıktı grubu) belirlenecektir. Bu hataların teşhis edilmesi ise çevrimiçi hata önlemeyi mümkün kılabilecektir.



Şekil-7.6. ART2 ağının topolojik yapısı

Şekil-7.6'da gösterildiği gibi, ağa sunulan girdi değerleri I vektörü ile gösterilmektedir. F1 katmanının w, x, u, v, p, ve q birimleri (üniteleri) vardır. F2 katmanında ise y birimi vardır. Aşağıdan yukarı b ile yukarıdan aşağıda t ile gösterilen ağırlık değerleri bulunmaktadır. Girdilerin ağın belirlediği vektörler ile eşleştirilmesi sonucu birbirlerine benzerliklerini kontrol eden mekanizma ise r ile gösterilmektedir. Bu birimlerin aktivasyon değerlerinin hesaplanması aşağıda gösterilmiştir.

7.7.2. ART2 Ağının Çalışma Prensibi

ART2 ağının öğrenme kuralı bir hipotez testine dayanmaktadır. Ağa yeni bir girdi örneği sunulduğunda, ağ bu girdi örneğini oluşturulmuş olan sınıflardan birisine koyup koyamayacağını belirlemektedir. Eğer bu mümkün ise o zaman öğrenme başlar ve atansiyonel alt sistem aktif olur. Eğer bu mümkün değil ise o zaman ağ, o girdi için yeni bir sınıf oluşturur (oryantasyon alt sistemi aktif olur) ondan sonra öğrenme başlar.

Dış dünyadan girdi değerleri F1 katmanına sunulmaktadır. Girdiler alındıktan sonra F1 katmanındaki proses elemanlarının aktivasyon değerleri aşağıda gösterildiği gibi hesaplanır. Daha sonra bu ünitelerin değerlerine dayanarak F2 katmanındaki proses elemanlarının aktivasyon değerleri hesaplanır. F2 katmanındaki en yüksek değeri üreten proses elemanı yarışmayı kazanan proses elemanı olarak belirlenir. Bu proses elemanının yukarıdan aşağı ağırlıkları F1 katmanındaki u ve p elemanlarının değerleri güncellenir ve ağa sunulan girdinin yukarıdan aşağı oluşturulan ağırlıklar ile eşleşmesi yapılır. Eğer bu eşleşme mümkün değil ise girdi için yeni bir kategori oluşturulur. Bu çalışma prensibi öğrenme kuralı olarak aşağıda ayrıntılı olarak açıklanmıştır.

7.7.3. ART2 Ağının Öğrenme Kuralı

ART2 Ağın öğrenmesinde iki filtre vardır. Bunlardan birincisi yukarıdan aşağı adaptif bir filtredir. F1 ve F2 katmanları arasındaki ağırlık değerlerinin değiştirilmesini sağlamaktadır. Burada girdi örneklerinin belirlenmiş kategorilerin hangisine ait olacakları belirlenir. İkinci filtre ise aşağıdan yukarı çalışan bir filtredir. Buda öğrenmenin hızını artırmak amacı ile geliştirilmiştir. Bu filtre paralel arama ve şekil uydurma görevini üstlenmektedir. Buraya kadar anlatılanlar ışığında geliştirilen ART2 ağının öğrenme algoritması şu adımlardan oluşmaktadır.

Adım 0: *Ağın parametrelerinin değerlerinin atanması;*

Bu parametreler a, b, θ, c, d, e, α, ρ değerleridir.

Bunlardan a, b, c parametreleri F1 katmanında kullanılan sabit ağırlıklardır. Kullanıcı belirlemektedir. Genellikle a=10, b=10 ve c=0.1 alınmaktadır.

d değeri ise F2 katmanında kazanan proses elemanının aktivasyon değerini göstermekte olup genel olarak 0.9 değeri kullanılır. Bu değerin 1'e yakın olması önemlidir. Böylece eşleşme sırasında uygunluk aralığı genişlemiş olur. Yalnız bu değeri belirlerken bir sınırlama vardır. Oda,

$$\frac{cd}{1-d} \geq 1$$

olmasıdır. Yukarıda belirtilen e değeri 0 ile bölünmeyi önlemek için kullanılan çok küçük bir sayıyı göstermektedir. Sıfır olarak alınmasında bir sorun yoktur. θ değeri ise bir eşik değeri olup,

$$\theta = \frac{1}{\sqrt{n}}$$

olarak belirlenmektedir. Buradaki n girdi ünitesindeki proses elemanı sayısını göstermektedir. α değeri ise öğrenme katsayısı olup sıfır ile bir arasında bir sayı değerini almaktadır. ρ ise benzerlik katsayısı olup 0 ile 1 arasında bir sayıdır. Benzerlik katsayısı ART1 ağında anlatılan fonksiyonun aynısını burada da yerine getirmektedir. Hatırlanacağı gibi, bu sayının büyük olması benzerliğin fazla olması demektir. Oda sınıf sayısını artırmaktadır. Eğer bu sayı küçük alınırsa o zaman girdiler ile ağın ağırlık değerlerinin birbirlerine benzerlikleri daha kolay olacağından sınıf sayısı az olacaktır.

Burada dikkatli olmakta yarar vardır. Eğitilen ağın performansı oluşturulan sınıfların olayı temsil etme yeteneği ile doğru orantılıdır. Sınıf sayısının yeterli olmayıp az olması durumunda ağın performansı olumsuz yönde etkilenebilir. Sınıf sayısının çok fazla olması ise pratik olarak uygulamayı zorlaştırabilir. Pratik hayatta benzerlik katsayısının 0.7 ile 1 arasında bir değer alması önerilmektedir. Tasarımcılar problemi en iyi şekilde temsil edecek bir sınıflandırmaya yol açacak bir katsayı değeri belirlemelidirler.

Benzerlik katsayısının belirlenmesinin yanı sıra ağın diğer parametrelerinin değerleri de belirlenir ve ağın aşağıdan yukarı ve yukarıdan aşağı ağırlıklarının başlangıç değerleri atanır. Yukarıdan aşağı ağırlıkların değerlerini atarken ilk sunulan örnek için bir sınıf açılmasını sağlamak amacı ile mümkün olduğu kadar düşük değerleri seçmek gerekir. Pratik hayatta yukarıdan aşağı ağırlıklar için başlangıç değerlerinin,

$t_{ji} = 0$ i: 1,2,....n - girdi sayısı

j: 1,2,....M – çıktı sayısı

olması önerilmektedir. Aşağıdan yukarı ağırlıkların başlangıç değerlerinin atanması içinde şu eşitsizliğin sağlanması önemlidir.

$$b_{ij} \leq \frac{1}{(1-d)\sqrt{n}}$$

Bu ağırlıkların büyük değerler atanması daha fazla sayıda sınıf oluşturulmasını sağlar. Eşitsizliği bozmamak koşulu ile değerler rasgele atanabilir

Parametre değerleri ve ağırlıkların başlangıç değerleri belirlendikten sonra, öğrenme adımları başlayabilir. Aşağıdaki adımlar örnek setindeki örneklerin tamamı için tekrarlanmalı ve her örnek için bütün adımlar gerçekleştirilmelidir.

Adım 1: *Önekleri ağa göstermek;*

Örnek setinden bir örnek alınarak ağa gösterilir. Bu örnek n boyutlu I vektörü ile gösterilirse $I = (I1, I2, ...In)$ olacaktır. Bu vektör F1 katmanından aa gösterilir.

Adım 2: *F1 katmanı proses elemanlarının aktivasyon değerlerinin belirlenmesi;*

F1 katmanındaki proses elemanlarının aktivasyonları şöyle belirlenir.

$u_i = 0;$

$w_i = I_i;$

$x_i = w_i / \|w\| + e$

$q_i = 0;$

$p_i = 0;$

$v_i = f(x_i)$

Buradaki f fonksiyonu doğrusal ve doğrusal olmayan bir fonksiyon olabilir. Hem yukarıdan aşağı hem de aşağıdan yukarı bilgi işlemede kullanılmaktadır. Doğrusal olması durumunda,

$$f(x_i) = \begin{cases} 0 & \text{Eğer } 0 <= x < \theta \text{ ise} \\ x & \text{Eğer } x >= \theta \text{ ise} \end{cases}$$

doğrusal olmaması durumunda ise,

$$f(x_i) = \begin{cases} \dfrac{2\ x^2\ \theta}{x^2 + \theta^2} & \text{Eğer } 0 <= x < \theta \text{ ise} \\ x & \text{Eğer } x >= \theta \text{ ise} \end{cases}$$

şeklinde hesaplanır.

Adım 3: *F2 katmanındaki proses elemanlarının değerlerini hesaplanası;*

F2 katmanında bulunan her proses elemanın çıktı değerleri şu şekilde hesaplanmaktadır.

$$y_j = \sum_i t_{ji} p_i$$

Adım 4: *Kazanan proses elemanının belirlenmesi;*

F2 katmanında M proses elemanı var ise ve kazanan proses elemanı da y^* ile gösterilir ise,

$$y^* = max\,(y_j)\ j=1,2...M$$

olur. Kazanan proses elemanının j. eleman olduğu varsayılırsa; çıktı değeri y_J, aşağıdan yukarı ve yukarıdan aşağı ağırlık değerleri ise sırası ile b_{iJ} ve t_{Ji} olacaktır.

Adım 5: *r değerinin hesaplanması;*

Yukarıdan aşağı ağırlıklar ile girdi vektörünün karşılaştırılması için kullanılan r değeri şu formül ile hesaplanmaktadır:

$$r_i = \frac{u_i + cp_i}{e + \|u\| + \|p\|}$$

Burada c kazanç parametresi olup kullanıcı tarafından belirlenmektedir.

$$u_i = \frac{v_i}{e + \|v\|} \qquad p_i = u_i + \sum_j g(y_j) t_{Ji}$$

olacaktır. Burada sadece kazanan elemanın aşağı doğru ağırlıklarının kullanıldığına (j) dikkat ediniz. Bu formüllerdeki t_{Ji} kazanan proses elemanının yukarıdan aşağı ağırlık değerlerini göstermektedir. $g(y_j)$ ise F2 katmanındaki en fazla aktivasyonu olan proses elemanını (kazanan proses elemanı) göstermektedir.

$$g(y_j) = \begin{cases} d & \text{F2 katmanında kazanan eleman için } 0<d<1 \\ 0 & \text{Aksi durumda} \end{cases}$$

Adım 6: *Eşleştirmenin uygunluğunun belirlenmesi;*

Burada girdi vektörü ile hafızada saklı bulunan ve yarışmayı kazanan proses elemanının kategori gösterim vektörünün eşleştirilmesi kastedilmektedir.

Eğer, $\|r\| < \rho - e$ olursa, girdi vektörü kazanan olamayacak demektir. Bu durumda F2 katmanındaki bir sonraki en yüksek çıktı değeri olan (bir sonraki kazanan) proses elemanı için r değeri hesaplanır. Bu bütün sınıflar kontrol edilinceye kadar devam eder. Eğer hiç bir sınıf bulunmaz ise yeni bir sınıf açılarak bu örneğin yeni bir sınıf temsil ettiği varsayılır.

Eğer herhangi bir aşamada, $\|r\| \geq \rho - e$ ise o zaman girdi örneği için bir sınıf var demektir. Bu durumda ağırlıklar ve F1 katmanındaki proses elemanlarının değerleri güncellenir. Güncelleme aşağıda şekilde yapılmaktadır:

$$w_i = I_i + au_i$$

$$p_i = u_i + \sum_j g(y_j) t_{Ji}$$

$$x_i = \frac{w_i}{e + \|w\|}$$

$$v_i = f(x_i) + bf(q_i)$$

$$q_i = \frac{p_i}{e + \|p\|}$$

Bu aşamada kazanan proses elemanının ağırlıkları da değiştirilir. Hem aşağıdan yukarı hem de yukarıdan aşağı ağırlıklar şu şekilde değiştirilirler:

$$t_{Ji} = \alpha d\ u_i + [1 + ad\,(d-1)]\, t_{Ji}$$

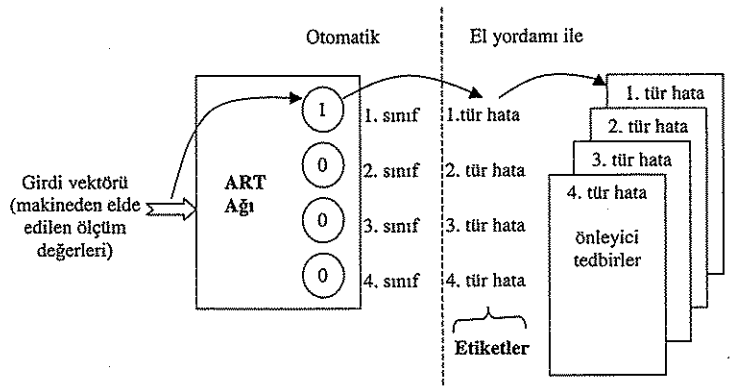
$$b_{iJ} = \alpha d u_i + [1 + ad\,(d-1)]\, b_{iJ}$$

Yukarıdaki adımlar bütün örnek setindeki örnekler sınıflanıncaya kadar devam eder. Bazı durumlarda belirlenen iterasyon sayısı tamamlanıncaya kadar eğitim sürdürülür.

7.8. ART Ağlarında Etiketlendirme

ART ağları öğretmensiz öğrenme gerçekleştirdiklerinden eğitim setindeki örnekler ağ tarafından otomatik olarak sınıflandırılmaktadır. Eğitilen ağa daha sonra görmediği örnekler sorulunca, ağ kendisi bu örneğin kendisinin belirlediği sınıflardan hangisine gireceğine karar verir. Örneklerin sınıflarının bulunmasına rağmen hangi sınıfın neyi temsil ettiği bilinmemektedir. Grupların neyi temsil ettikleri öğrenme bittikten sonra tasarımcı tarafından belirlenir. Buna ***sınıfın etiketlenmesi*** denir. Bu konuyu bir örnek ile açıklamak gerekirse, hata teşhisi yapmak üzere eğitilmiş olan bir ağ için 4 çıktı grubunun (hata sınıfının) olduğu varsayılsın. Oluşturulan ağın 4 tane çıktı ünitesi olacaktır. Bu ağ örnekler üzerinde eğitildikten ve 4 tür hatayı teşhis edecek duruma geldikten sonra ağ kullanıma alındığında kendisine gösterilen bir örneğin (hata türünün) sadece 4 sınıftan hangisine ait olduğunu söyleyecektir. Bu sınıfın hangi tür hata olduğu ise bilinmemektedir. Sadece bu sınıfa ait örneklerin hepsinin temsil ettiği hata türü

aynıdır. Sınıfların hangi hata türünü gösterdiklerini belirlemek için tasarımcı incelemeler yaparak sınıfların etiketlerini belirler. Böylece ağ görmediği örnekler gelince önce grubunu belirler ve o grubun etiketine göre dış dünyaya örnek hakkında bilgiler verebilir. Hatanın türü belirlendikten sonra o hatayı önleyici tedbirler kullanıcıya sunulabilir. Şekil-7.7 bu durumu göstermektedir.



Şekil-7.7. Art ağında etiketlendirme

Etiketlendirme ağın eğitilmesi kadar önemlidir. Ağı eğitmenin amacı belirli sorunları çözmektir. Sorunu tespit etmek kadar sorunu çözmekte önemlidir. Ağın eğitilmesinde bir öğretmen kullanılmamaktadır. Çıktıların sınıfları temsil ettiği bilinmektedir. Etiketlendirme ile hangi sınıfın ne anlama geldiğinin belirlenmektedir. Bu ise sorunların tespit edilmesi ve gerekli çözüm önerilerinin oluşturulması demektir. Etiketlendirme o nedenle çok önemlidir. Bazı araştırmacılar bunu öğrenmeye müdahale olarak görse de bu ağın öğretmenli öğrenme yaptığı anlamına gelmez.

7.9. ART1 – Bir Örnek Uygulama -Grup Teknolojisine Dayalı İmalat Uygulaması

Daha önce anlatılan ağlarda olduğu gibi ART ağları da endüstriyel problemlere çözümler üretebilmektedir. Aşağıda Grup teknolojisinde gerçekleştirilen bir uygulama anlatılacaktır.

7.9.1. Problemin Tanımlanması

Grup teknolojisi imalat sistemlerinin en yaygın biçimde kullanılan yaklaşımlarından birisidir. Bu yaklaşımın verimliliği artırdığı, üretim maliyetlerini düşürdüğü, özellikle kütle üretimini çok rahat desteklediği bilinmektedir. Grup teknolojisinin temel felsefesi birbirine benzer aynı makineleri kullanılarak üretilen parçaları ve prosesleri belirleyerek bunları üretecek makineleri bir arada toplamaktır. Bu amaçla çeşitli gruplama algoritmaları geliştirilmiştir [11]. Bu algoritmaların farklılıkları performanslarının farklı olmasından kaynaklanmaktadır. Üretilecek parça ve kullanılan

makine sayısı arttıkça üretilen çözümlerin performansı düşmektedir. Bu algoritmalara alternatif olarak yapay sinir ağlarından yararlanan bir çalışma yapılmış ve ART algoritması kullanılarak makinelerin sınıflandırılması gerçekleştirilmiştir [12]. Yapay sinir ağlarının bilinen geleneksel algoritmalar ile üretilen sonuçlardan daha yüksek performansta sonuçlar ürettikleri görülmüştür.

Grup teknolojisinde genel olarak, seri üretim yapan bir fabrikalarda benzer ürünleri üreten makineleri bir araya getirmek istenmektedir. Oluşturulan makine gruplarına makine hücreleri denilmektedir. Her makine hücresi için atölyede bulunan işler ve bu işleri işleyecek makinelerin sınıflandırılmasının yapılması düşünülmektedir. Bu sınıflamanın maliyetleri en azlayacak şekilde yapılması gerekmektedir. Bazı makineler bütün parçaları işleyebilmekte bazıları ise sadece belirli parçaları işleyebilmektedir. Makineler arasında ara stokların azalması ve parçaların atölye içinde en az yol katederek üretim hattından çıkması oluşturulacak olan makine hücrelerinin doğru oluşturulmasına bağlıdır.

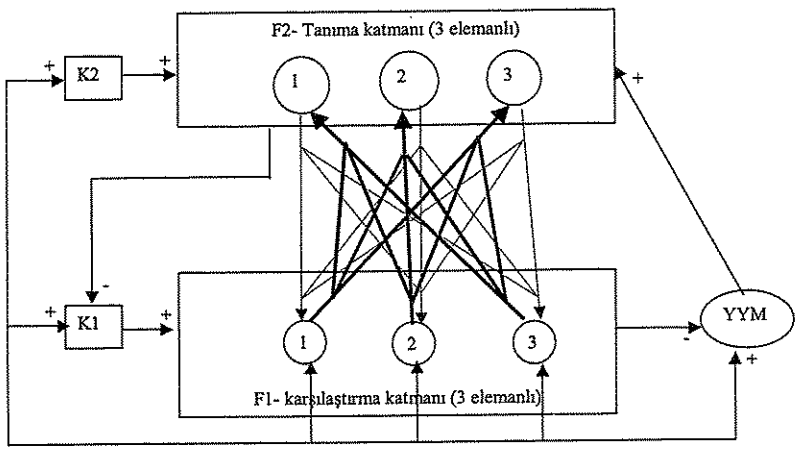
Aşağıda ART1 ağlarının grup teknolojisinde kullanılmasına yönelik geliştirilmiş bir model açıklanacaktır. Geliştirilen ART1 ağından makine/parça arasındaki ilişkileri gösteren örnekleri alarak, dış dünyadan herhangi bir yardım almadan makine/parça sınıflandırmasını otomatik olarak yapması istenmektedir. Problemin nasıl çözüldüğünün daha iyi anlaşılması için burada 3 makine ve 3 parçadan oluşan küçük bir problem için tasarlanmış ART1 ağı açıklanmıştır. Mamafih, istenilen sayıda makine ve parça için ART1 ağının uygulanması mümkündür. Modelin uygulanmasında parça ve makine sayısında herhangi bir sınırlama yoktur. Kullanılan öğrenme kuralında da bir değişiklik gerekmektedir. Özellikle çok sayıda makine ve parçanın söz konusu olması durumunda yapay sinir ağlarının, özellikle ART ağının, avantajlarını daha açık olarak görmek mümkündür.

7.9.2. Problemin Modelinin Oluşturulması

Gösterim amaçlı düşünülen ve 3 makine ve 3 parçadan oluşan problemde makine/parça ilişkilerini gösteren matris şu şekilde verilmiştir.

$$\begin{array}{c} \\ M1 \\ M2 \\ M3 \end{array} \begin{array}{c} \begin{array}{ccc} P1 & P2 & P3 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

Bu matristen anlaşıldığına göre birinci parça hem birinci hem de ikinci makinede işlenebilmektedir. İkinci parça ise sadece ikinci makinede işlenebilmektedir. Üçüncü parça ise hem birinci hem de üçüncü makinede işlenebilmektedir. Problemde makinelerin aynı sınıfın üyesi olup olmadıklarını belirlemeye yarayan benzerlik katsayısı ρ=0.8 olarak alınmıştır. Yani %80 civarında bir benzerlik aynı grubun elemanı olmak için yeterli görülmektedir. Oluşturulan ART1 ağı Şekil-7.8'de gösterilmiştir. Şekildeki F1 ve F2 katmanları arasındaki koyu renkli bağlantılar aşağıdan yukarıya açık renkli bağlantılar ise yukarıdan aşağı ağırlıkları göstermektedir.



Şekil-7.8. Oluşturulan ART 1 ağının gösterimi

7.9.3. Oluşturulan Ağın Eğitilmesi

Oluşturulan makine/parça matrisinin her satırı bir örnek olarak ağa gösterilecektir. Birincisi ağa gösterildiği zaman, daha önce herhangi bir sınıflandırma yapılmadığından ağ otomatik olarak birinci sınıfı oluşturacak ve birinci makineyi birinci sınıfın üyesi olarak belirleyecektir. İkinci örnek geldiği zaman bunu birinci örnek ile karşılaştıracak ve eğer birbirlerine benzer bulursa (hesaplanacak benzerlik katsayısı 0.8 üzerinde ise) ikinci makineyi de birinci sınıfa koyacaktır. Eğer benzer bulmaz ise ikinci makineyi ikinci sınıfın temsilcisi olarak belirleyecektir. Bütün makineler sınıflandırılıncaya kadar bu çalışma devam eder. Bu süreç aşağıda gösterilmiştir. Eğitim süresince geliştirilen her iterasyonda yapılacak olan işlemler açıklanmıştır.

1. İterasyon

Ağın eğitilmesine başlanan birinci iterasyonda aşağıdaki hesaplamalar yapılır.

1. Adım: *Ağırlıklara başlangıç değerleri atanması;*

F1 katmanındaki proses elemanı sayısı N=3

F2 katmanındaki proses elemanı sayısı M=3

Benzerlik katsayısı ρ=0.8

F2 ve F1 katmanları arasındaki geriye doğru ağırlıkların bütün değerleri başlangıçta 1 değerini alır. Yani,

$$A_{ji}^{g} = 1 \qquad j: 1,2,3 \ldots M \quad i: 1,2,3 \ldots N$$

A^g matris şeklinde yazılırsa,

$$A^g = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

bu matrisin her satırı bir kategori gösterim vektörü olarak düşünülebilir.

F1 ve F2 katmanları arasındaki ileri doğru ağırlıkların başlangıç değerleri ise, F1 katmanındaki proses elemanı sayısı N olmak üzere ise şu şekilde atanmaktadır.

$$A_{ij}^{i} = \frac{1}{1+N} = \frac{1}{1+3} = 0.25 \qquad j: 1,2,3 \ldots M \quad i: 1,2,3 \ldots N$$

2. Adım: *Girdi setinden bir örnek ağa gösterilmesi;*

Girdi setindeki örnek X= (1,0,1) ağa gösterilir.

3. Adım: *F2 katmanındaki proses elemanlarının çıktılarının hesaplanması;*

F2 katmanındaki her proses elemanının çıktı değerleri sırası ile şu şekilde hesaplanmaktadır.

$$y_1' = \sum_{i}^{3} A_{ij}^{i}(t) x_i = 0.25.1 + 0.25.0 + 0.25.1 = 0.5$$

$$y_2' = \sum_{i}^{3} A_{ij}^{i}(t) x_i = 0.25.1 + 0.25.0 + 0.25.1 = 0.5$$

$$y_3' = \sum_{i}^{3} A_{ij}^{i}(t) x_i = 0.25.1 + 0.25.0 + 0.25.1 = 0.5$$

4. Adım: *Kazanan elemanın seçilmesi;*

F2 katmanındaki proses elemanlarının çıktı (y) değerlerinin hepsi eşit ve 0.5 olduğundan F2 katmanındaki elemanlardan bir tanesi rasgele yarışmayı kazanmış sayılır. Bu örnekte 1. proses elemanı kazanan eleman olarak seçilmiştir. Yani,

$$y_k^* = \max(y_j) \Rightarrow y_1^* = 0.5$$

5. Adım: *Uygunluk testinin yapılması;*

Burada kazanan elemanın ağırlık vektörü ile girdi vektörünün birbirlerine benzerliği kontrol edilmektedir. Bunun için önce girdi vektöründe bulunan 1 sayısı ($s1$) belirlenir.

$$s1 = |X| = \sum x_i = 1 + 0 + 1 = 2$$

Bundan sonra kategori gösterimi vektörü ile girdi vektörünün uyuştuğu 1 sayısı ($s2$) bulunur. Bunu veren formül ise şöyledir:

$$A_{11}{}^{g} = [1 \quad 1 \quad 1]$$

olduğundan,

$$s2 = \left| A_{11}^{g}.X \right| = \sum A_{j1}^{g} x_i = 1.1+0.1+1.1 = 1+0+1 = 2$$

$$\frac{S2}{S1} = \frac{2}{2} = 1 \geq \rho$$

olduğundan bu örneğin birinci kategori gösterim vektörünün gösterdiği 1.sınıfın bir üyesi olur. Hem ileriye doğru hem de geriye doğru ağırlıklar aşağıdaki gibi değiştirilir.

$$A_{11}^{g}(t+1) = A_{11}^{g}(t)x_1 = 1.1 = 1$$

$$A_{21}^{g}(t+1) = A_{21}^{g}(t)x_2 = 1.0 = 0$$

$$A_{31}^{g}(t+1) = A_{31}^{g}(t)x_3 = 1.1 = 1$$

Benzer şekilde,

$$A_{11}^{i}(t+1) = \frac{A_{11}^{g}(t)x_1}{0.5 + \sum_{i=1}^{3} A_{11}^{g}(t)x_i} = \frac{1.1}{0.5+1.1+1.0+1.1} = 0.4$$

$$A_{21}^{i}(t+1) = 0.0$$

$$A_{31}^{i}(t+1) = 0.4$$

Böylece birinci iterasyon tamamlanmış ve birinci makine birinci grubun elemanı olarak atanmıştır. Çünkü benzerlik katsayısından büyük bir benzerlik ortaya çıkmıştır.

2. İterasyon

İkinci iterasyonda işlemlere 2. adımdan başlayarak devam edilir.

2. Adım: *Girdi setinden bir örnek ağa gösterilmesi;*

Girdi setindeki örnek X= (1,1,0) ağa gösterilir.

3. Adım: *F2 katmanındaki proses elemanlarının çıktılarının hesaplanması;*

F2 katmanındaki her proses elemanının çıktı değeri birinci iterasyondakine benzer şekilde hesaplanmaktadır.

$$y_1' = \sum_{i}^{3} A_{ij}^{i}(t)x_i = 0.4$$

$$y_2' = \sum_{i}^{3} A_{ij}^{i}(t)x_i = 0.5$$

$$y_3' = \sum_{i}^{3} A_{ij}^{i}(t)x_i = 0.5$$

4. Adım: *Kazanan elemanın seçilmesi;*

İkinci ve üçüncü proses elemanlarının çıktı değeri en yüksek olup birbirine eşit olduklarından bunların arasından 2. proses elemanı (rasgele) kazanan olarak alınmıştır. Yani,

$$y_k^{*} = max(y_j) \Rightarrow y_2^{*} = 0.5$$

5. Adım: *Uygunluk testinin yapılması;*

Girdi vektöründeki 1 sayısı 2 (s1=2) tanedir. Sınıf (kategori) gösterim vektöründe girdi ile uyuşan 1 sayısı da yine 2'dir ($s2$=2). $A_2{}^{g} = [1 \quad 1 \quad 1]$ olduğundan benzerlik katsayısı ile mukayese yapılırsa;

$$s1 = |X| = 2$$

$$s2 = \left| A_2^{g}.X \right| = 2$$

$$\frac{S2}{S1} = \frac{2}{2} = 1 \geq \rho$$

sonucu elde edilir. Bu durumda ilgili ağırlıkların değiştirilmesini gerektirir. Yeni ağırlıklar şöyle belirlenir.

$$A_{12}^{g}(t+1) = A_{12}^{g}(t)x_1 = 1.1 = 1$$

$$A_{22}^{g}(t+1) = A_{22}^{g}(t)x_2 = 1.1 = 1$$

$$A_{32}^{g}(t+1) = A_{32}^{g}(t)x_3 = 1.0 = 0$$

Benzer şekilde;

$$A_{12}^{i}(t+1) = 0.4$$

$$A_{22}^{i}(t+1) = 0.4$$

$$A_{32}^{i}(t+1) = 0.0$$

Böylece ikinci iterasyon tamamlanmış ve ikinci makine ikinci proses elemanının temsil ettiği sınıf ile benzerlik gösterdiğinden 2. sınıfın elemanı olarak seçilmiştir.

3. İterasyon

Üçüncü iterasyonda işlemlere yine 2. adımdan başlanarak adımların hepsi benzer şekilde yerine getirilir.

2. Adım: *Girdi setinden bir örnek ağa gösterilir;*

Girdi setindeki üçüncü örnek X= (0,0,1) ağa sunulur.

3. Adım: *F2 katmanındaki proses elemanlarının çıktılarının hesaplanması;*

F2 katmanındaki her proses elemanının çıktısının değeri benzer şekilde hesaplanmaktadır.

$$y_1^{'} = \sum_i^3 A_{ij}^i(t)x_i = 0.4$$

$$y_2^{'} = \sum_i^3 A_{ij}^i(t)x_i = 0.0$$

$$y_3^{'} = \sum_i^3 A_{ij}^i(t)x_i = 0.25$$

4. Adım: *Kazanan elemanın seçilmesi;*

Birinci proses elemanın çıktı değeri en yüksek olduğundan bu proses elemanı kazanan olarak alınır. Yani;

$$y_k^* = \max(y_j) \Rightarrow y_1^* = 0.4$$

5. Adım: *Uygunluk testinin yapılması;*

Önceki örneklerde gösterildiği gibi s1 ve s2 değerleri hesaplanarak benzetim katsayısı ile karşılaştırılır.

$$s1 = |X| = 1$$

$$s2 = \left|A_2^g . X\right| = 1$$

$$\frac{S2}{S1} = \frac{1}{1} = 1 \geq \rho$$

olduğundan bu makinede yine 1. sınıfın bir makinesi olarak sınıflandırılır. Birinci iterasyonda $A_1^9 = [1 \quad 0 \quad 1]$ olmuştu. Bu vektörün ağırlık değerleri yeniden değiştirilir.

$$A_{11}^g(t+1) = A_{11}^g(t)x_1 = 1.0 = 0$$

$$A_{21}^g(t+1) = 0$$

$$A_{31}^g(t+1) = 1$$

Benzer şekilde,

$$A_{11}^i(t+1) = 0.0$$

$$A_{21}^i(t+1) = 0.0$$

$$A_{31}^i(t+1) = 0.66$$

olacaktır. Bundan sonra birinci örnek ağa tekrar gösterildiğinde birinci sınıfın bir üyesi olarak sınıflandırıldığı görülür. Böylece bütün parça/makine sınıfları belirlenmiş olur.

Sınıflandırılacak başka makine kalmadığından iterasyon tamamlanmış olur. Ağırlıkları tekrar değiştirmenin bir anlamı olmayacaktır.

Sonuç olarak 1. ve 3. makineler birinci sınıfın elemanları olup aynı yere konulacak 2. makine ise ayrı bir sınıf olarak farklı bir yere konulacaktır. Böylece iş akışı düzenli olarak yürütülecek ve ara stoklar sorun oluşturmayacaktır.

7.9.4. Sonuçların Tartışılması

Örnekte de görüldüğü gibi problem oldukça basit bir algoritma ile çözülebilmektedir. Burada modelin anlaşılabilmesi için çok küçük boyutlu bir problem çözülmüştür. Makine ve parça sayılarının artması durumunda geleneksel yöntemlerin performansı düşmektedir. Yani önerdikleri çözümde hem parçalar atölye içinde daha fazla dolaşmakta hem de maliyetler artmaktadır. Yapay sinir ağlarının bu durumlarda oldukça başarılı sonuçlar ürettiği görülmüştür. Kusiak ve Chung [12] tarafından yapılan çalışmadan geleneksel yöntemler yapay sinir ağları çözümü ile karşılaştırılmış ve yapay sinir ağlarının başarısı ispatlanmıştır.

7.10. Özet

Bu bölümde öğretmensiz öğrenme stratejisi kullanarak öğrenen ART ağları tanıtılmıştır. Bu ağların en önemli özellikleri gerçek zamanlı çalışabilmeleri ve çevrimiçi (*on-line*) öğrenebilmeleridir. Yeni durumlara adapte olabilmekte ART ağları oldukça güçlüdürler. Bunu sağlayabilmek için bir taraftan öğrenirken bir taraftan da unutabilmektedirler. Bir ART ağı, gösterim katmanı (F1) ve kategori katmanı (F2) olmak üzere iki katmandan oluşmaktadır. Bunların yanında, ağa sunulan girdilerin sınıflandırılamaması durumunda yeni bir kategori oluşturmaya yardım eden oryantasyon modülü vardır. F1 ve F2 katmanları arasında hem aşağıdan yukarı hem de yukarıdan aşağı ağırlık vektörü vardır. Eğitim sırasında bu ağırlıklar örneklere bakılarak değiştirilirler. Girdiler F1 katmanından ağa sunulur ve yukarı doğru ağırlıklar ile F2 katmanından çıktı değerleri hesaplanır. F2 katmanında en yüksek çıktı değerini oluşturan proses elemanı yarışmayı kazanan proses elemanı olarak 1 değerini diğerleri ise 0 değerini alır. Bu elemana bağlı ağırlıklar değiştirilir. Kazanan elemanın ilgili girdi vektörünün sınıfını göstermesi için ona bağlı hafızadaki vektör ile girdi vektörünün birbirine benzemesi gerekmektedir. Bu benzerliği benzerlik katsayısı denilen bir katsayı ile karar verilir. Eğer iki vektör birbirine benzer bulunursa girdi vektörü o sınıfın elemanı

sayılır ve ağırlıklar girdi vektörüne göre güncellenir. Arada benzerlik olmaz ise o zaman oryantasyon sistemi o girdi vektörü için yeni bir sınıf oluşturur. Bu nedenle bir ART ağında örnek sayısı kadar sınıf sayısı oluşturulabilir. Değişik ART ağları geliştirilmiştir. ART1 ağı geliştirilmiş ilk ART ağıdır. Bu ağ sadece ikili (*binary*) değerlerden oluşan girdi vektörlerini kabul eder. ART2 ağları ise sürekli değerleri de kabul etmektedirler. Benzer şekilde ART3, Fuzzy ART, ARTMAP gibi başka modellerde geliştirilmiştir. Bu modellerin eğitim performansları benzetim katsayısı ile yakından ilgilidir. Bu katsayının küçük olması sınıf sayısını azaltmakta, büyük olması ise sınıf sayısını artırmaktadır. Çünkü sayının büyük olması daha fazla benzerlik istenmesidir. İstenilen benzerlik düzeyi yakalanamayınca da yeni sınıflar oluşturulmaktadır. ART ağları endüstriyel problemlerde özellikle sınıflandırma amaçlı kullanılmakta ve oluşturulan etiketlendirme mekanizması ile sorunlara çözümlere önerilebilmektedir. Grup teknolojisi gibi makine hücrelerinin belirlenmesinde de geleneksel yöntemlerden daha etkili çözümler üretebilmektedirler.

Bu ağlardan özellikle ART2 ağının en önemli sorunlarından birisi birçok parametreyi kullanıcının belirlemesinin gerekmesidir. Bunlar ise ancak uzun denemeler ve deneyimler sonucunda doğru olarak belirlenebilmektedir.

7.11. Kaynakça

[1] Grossberg, S. (1976a). Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors, Biological Cybernetics, 23, 121-134.

[2] Grossberg, S. (1976b). Adaptive pattern classification and universal recoding I-I: Feedback, expectation, olfaction, and illusions, Biological Cybernetics, 23, 187-2002.

[3] Grosberg S. (1988) Nonlinear Neural Networks, Principles, mechanisms, and architectures, Neural Networks, Vol.1, pp.17-6.

[4] Carpenter G.A., Grosberg S., (1987), A massively parallel architecture for self organizing neural pattern recognition machine, Compuet Vision, Graphics and Image Processing, Vol. 37, pp. 54- 115.

[5] Carpenter G.A, Grosberg S., (1986), Category learning and adaptive pattern recognition: A neural network model, Proc. of the 3rd Army Conference on Applied Mathematics and Computing, ARO, report 86-1, pp. 37-56.

[6] Carpenter G.A., and S. Grossberg, (1987) "ART2: Self-organisation of stable category recognition codes for analog input patterns,"Applied Optics, vol. 26, pp. 4919-4930.

[7] Carpenter G.A, Grsoberg S, (1990) "ART3: Hierarchical search using chemical transmitters in self –organizing pattern recognition architectures", Neural Networks, Vol 3, pp. 129-152.

[8] Carpenter, G.A., Grossberg, S. and Reynolds, J.H. (1991a), "ARTMAP: Supervised real-time learning and classification of non-stationary data by a self-organizing neural network", Neural Networks 4, 565-588.

[9] Huang, J., Georgiopoulos, M. and Heileman, G.L. (1995), "Fuzzy ART properties", Neural Networks 8, 203-213.

[10] Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H. and Rosen, D.B. (1992), "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", IEEE Transactions on Neural Networks 3, 698-712.

[11] Burbidge J.L., (1975), Introduction to group technology, Heineman, London,

[12] Kusiak A., Chung Y., (1991), "GT/ART: Using neural networks to form machine cells", Manufacturing Review, Vol 4, No 4, 293-301.

BÖLÜM 8

GERİ DÖNÜŞÜMLÜ (RECURRENT) AĞLAR (ELMAN AĞI) VE DİĞER YAPAY SİNİR AĞI MODELLERİ

Daha önce ele alınan ağların en temel özelliklerinden birisi de ileri beslemeli ağlar olmalarıdır. Yani, bu ağlardaki proses elemanları çıktılarının bir sonraki katmana veya dış dünyaya gönderirler. Bu elemanların çıktılarının geri dönüşümü yoktur. Çıktıları proses elemanlarına geriye tekrar girdi olarak kullanılmamaktadır. Bu bölümde ise öncelikle geri dönüşü olan ağlar tanıtılacak ve bunlardan `Elman` ağı ayrıntılı olarak anlatılacaktır. Daha sonra yapay sinir ağının yaygın olarak kullanılan diğer modelleri kısa kısa açıklanacaktır.

8.1. Geri Dönüşümlü Ağlar

Geri dönüşümlü ağlarda, ağın proses elemanlarının çıktıları yine ağa belirli bir şekilde geri gönderilerek girdi olarak kullanılmaktadır. Dinamik sistemlerin modellenmesinde ve öğrenilmesinde geri dönüşümlerin olması özellikle zaman gecikmelerini (*time delays*) dikkate almak için önemlidir. Bu amaçla değişik modeller geliştirilmiştir [örnekler için bkz.1-3].

Geri dönüşümlü ağlar iki şekilde olabilir:

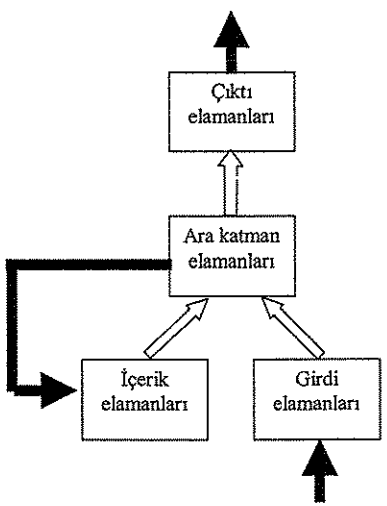
- ***Tam geri dönüşümlü ağlar***: Bu ağlar gelişigüzel ileri ve geri bağlantıları olan ağlardır. Bu bağlantıların hepsi eğitilebilirdir.
- ***Kısmi geri dönüşümlü ağlar***: Bu ağlarda ağın proses elemanlarına ek olarak içerik (*context*) elemanları vardır. Bu ağlar, temelde ileri beslemeli bir ağdır. İleri bağlantılar eğitilebilirler. Geri dönüşüm sadece içerik elemanları üzerinde yapılır ve bu bağlantılar eğitilemezler. İçerik elemanları ara katman elemanlarının geçmiş durumlarını hatırlamak için kullanılırlar. Ağın çıktısı hem önceki durumlara hem de ağın o andaki durumuna bağlı olarak oluşturulmaktadır. Geçmiş durumları hatırlayabilmeleri bu ağlara dinamik hafızaya sahip olma özelliği kazandırmaktadır.

Geri dönüşümlü ağların arasındaki en basit yapıya sahip olan ve kullanılması en kolay ağ `Elman` [1] ağıdır. Bu bölümde bu ağ ayrıntılı olarak açıklanacaktır.

8.1.1. Elman Ağı ve Yapısı

Elman ağı daha önce anlatılan çok katmanlı algılayıcı ağının öğrenme kuralına göre öğrenmektedir. Elman ağının yapısı Şekil-8.1'de gösterilmiştir. Şekilden de görüldüğü gibi Elman ağının 4 çeşit proses elemanı vardır:

- Girdi elemanları
- Ara katman elemanları
- Çıktı elemanları
- İçerik (*context*) elemanları



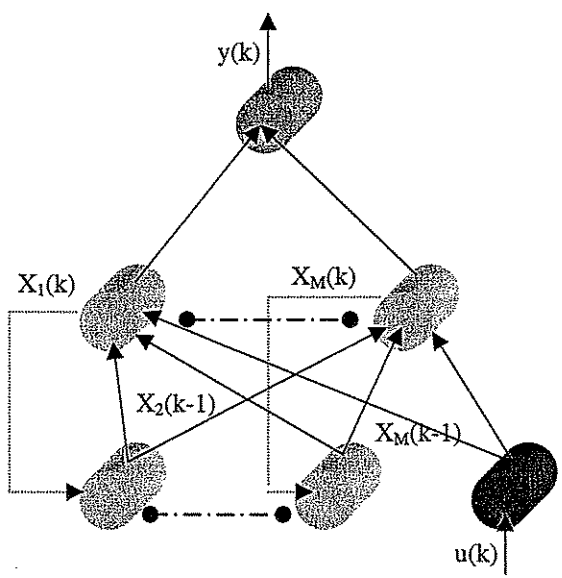
Şekil-8.1. Elman ağı yapısı

Bunlardan girdi ve çıktı elemanları dış dünya ile etkileşim halindedir. Girdi elemanları dış dünyadan bilgileri alır ara katmanlara iletirler. Çok katmanlı algılayıcılarda olduğu gibi, Elman ağında da girdi elemanlarının bilgi işleme özellikleri yoktur. Girdileri olduğu gibi ara katman elemanlarına gönderirler. Çıktı elemanları ise ağın çıktısını dış dünyaya iletirler. Çıktı ünitelerinin bilgi işleme fonksiyonları doğrusaldır. Sadece kendilerine gelen bilgileri toplarlar. Ara katman elemanları ise hem doğrusal hem de doğrusal olmayan aktivasyon fonksiyonlarına sahip olabilirler. İçerik elemanları ara katman elemanlarının önceki aktivite değerlerini hatırlamak için kullanılmaktadır. Bu elemanlar bir adım gecikmeyi (*one step time delay*) içermektedirler. Bir önceki iterasyondaki aktivasyon değerlerini bir sonraki iterasyona girdi olarak taşırlar. Şekildeki ileri beslemeli bağlantıların ağırlıkları (boş oklar) eğitim sırasında değiştirilebilirler. Geri dönüşümlerin (içerik elemanlarının) bağlantı ağırlıkları ise (dolu oklar) sabittir. Bu ağırlıklar değiştirilmezler. Geri dönüşümlerin ağırlıkları sabit olduklarından Elman ağına kısmı geri dönüşümlü ağ denilebilir.

`Elman` ağındaki, her hangi bir *t* zamanında hem *t* zamanındaki girdi değerleri hem de ara katmanların *t*-1. zamandaki (önceki) aktivite değerleri ağa girdi olarak verilirler. Ağın girdileri belirlendikten sonra ağ artık ileri beslemeli bir çok katmanlı algılayıcıya dönüşmektedir. Bu girdiler kullanılarak ileri doğru ağın çıktıları belirlenir. Bunu belirlemek için Bölüm 5'te ele alındığı gibi standart Çok Katmanlı Algılayıcı öğrenme kuralı kullanılmaktadır. Bu ileri doğru hesaplamadan sonra ağın ara katmanlarının aktivasyon değerleri geriye doğru içerik elemanlarına girdi olarak gönderilir ve orada bir sonraki iterasyonda kullanılmak üzere saklanır.

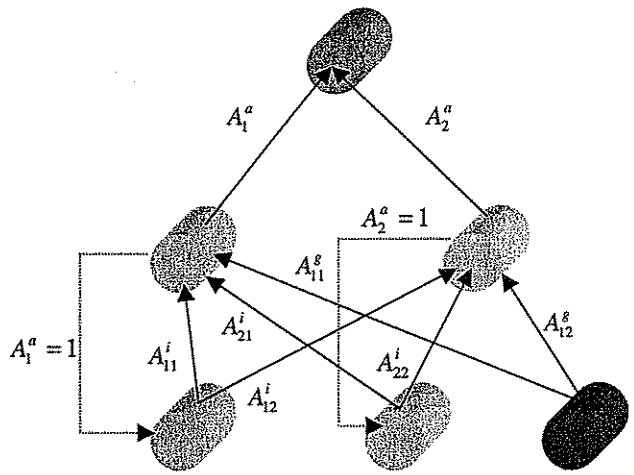
Başlangıçta ara katmanların aktivasyon değerleri bilinmediğinden içerik elemanlarının başlangıç değerlerinin belirlenmesi gerekir. Bunun için, genel olarak bir ara katmanın alabileceği maksimum değerin yarısı içerik elemanlarının başlangıç girdi değerleri olarak atanır. Eğer *sigmoid* fonksiyonu kullanılacak ise genellikle bu elemanlar başlangıçta girdi olarak 0.5 değerini alırlar. Hiperbolik Tanjant fonksiyonu için ise 0 değeri başlangıç girdi değerleri olarak atanmaktadır.

Şekil-8.2'de 1 girdi *M* ara katman elemanı ve 1 çıktıdan oluşan `Elman` ağının daha ayrıntılı bir şekilde gösterimi verilmiştir. Şekildeki *u(k), k.* zaman dilimindeki dış dünyadan gelen girdiyi; *y(k), k.* zaman diliminde üretilen çıktıyı; *x(k)* ise, *k.* zaman dilimindeki ara katman elemanlarının çıktılarını; *x(k-1)* ise bir önceki zaman diliminde ara katman elemanlarının çıktılarını göstermektedir.



Şekil-8.2. Elman ağının ayrıntılı gösterimi

Şekil-8.3'te 1 girdi 2 ara katman ve 1 çıktı elemanından oluşan bir Elman ağının ağırlıkları gösterilmiştir. Şekilde girdi elemanı ile ara katman arasındaki ağırlıklar A^g ile içerik elemanı ile ara katman arasındaki ağırlıklar A^i, A^a ile gösterilmiştir. Benzer şekilde ara katmandan içerik elemanlarına geri dönüşüm bağlantı ağırlıkları ise A^s ile gösterilmiştir. Bu ağırlıklar değiştirilmez ve değerleri sabit olup 1'e eşittir.



Şekil-8.3. Elman ağı bağlantı ağırlıkları

8.1.2. Elman Ağının Öğrenmesi

Yukarda belirtildiği gibi Elman ağının öğrenmesi Çok Katmanlı Algılayıcılarda anlatılan Genelleştirilmiş Delta öğrenme kuralına göre gerçekleşmektedir. Ara katmanda bulunan elemanlara gelen net girdi değeri girdi katmanındaki elemanın girdi değeri (u) ile ağırlığının (A^g) çarpılıp toplanması sonucu bulunan değerlere içerik elemanından gelen bağlantı değerlerinin ara katmanlarının (A^i) bir önceki aktivite değerleri ile çarpılıp eklenmesi sonucu bulunur. Bu değer bir fonksiyondan geçirilerek (f) ara katman elemanlarının çıktısı (x) bulunur. Yani herhangi bir k zaman diliminde kullanılan aktivasyon fonksiyonunun *sigmoid* olması durumunda ara katman elemanlarının çıktıları,

$$x_i(k) = \frac{1}{1+e^{-NET_i(k)}}$$

şeklinde hesaplanacaktır. Burada hesaplanan NET girdi ise yukarıda belirtildiği gibi ara katmanlardan gelecek olan geri beslemeler dikkate alınarak hesaplanır. Bilinen ÇKA öğrenme kuralında hesaplanan NET girdi değerindeki farklılık buradadır. Yani Elman ağında,

$$NET(k) = A^g u(k) + A^i x(k-1)$$

olacaktır. Bu formül açık olarak yazılırsa;

$$NET_i(k) = \sum_{j=1}^{N} A_{ji}^g u(k) + \sum_{i=1}^{M} A_{ij}^i x(k-1)$$

formülü elde edilir. Burada N girdi elemanı sayısını M ise ara katman elemanının sayısının göstermektedir.

ÇKA ağlarına olduğu gibi burada da kullanılan aktivasyon fonksiyonları türevi alınabilir fonksiyonlar olmak koşulu ile kullanıcı tarafından seçilir ve genelleştirilmiş delta kuralına göre hem proses elemanlarının çıktıları hesaplanır, hem de ağırlıkların değişim miktarları belirlenir. Ağın çıktısı ise çıktı katmanına gelen net değerin doğrusal fonksiyondan geçirilmesi ile belirlenir. Yani, çıktı katmanındaki proses elemanlarının aktivasyon fonksiyonları doğrusal olduğundan, herhangi bir k zaman diliminde çıktı katmanındaki çıktı elemanın değeri, A^a ağırlık değerleri ile ara katman elemanlarının çıktıları (x) kullanılarak;

$$y(k) = A^a(k)x(k)$$

şeklinde hesaplanır. k. zaman diliminde çıktı katmanındaki j. elemanın çıktısı

$$y_j(k) = \sum_{i=1}^{M} A_i^a(k) X_i(k)$$

olacaktır. Böylece ağın ileri doğru bilgi işlemesi tamamlanmış olur. Ağa sunulan her örnek için beklenen çıktı (B) belirli olduğundan, k. zaman diliminde j. çıktı elemanında oluşan hata (E_j)

$$E_j = B_j(k) - y_j(k)$$

olacaktır. Bu hata, genelleştirilmiş delta kuralına göre ağın değişebilen ağırlıklarına dağıtılır. Önce bu hata değeri çıktı fonksiyonunun türevi ile çarpılarak ağırlıklara dağıtılacak hata oranları (δ) belirlenir. Çıktı fonksiyonunun *sigmoid* olması durumunda k. zaman diliminde ağırlıklara dağıtılacak olan hata şu şekilde belirlenir.

$$\delta(k) = y(k) - [1 - y(k)]E(k)$$

Ağın ağırlıklarının değiştirilmesi de Bölüm 5'te anlatıldığı gibi benzer şekilde değiştirilir. Bunun için önce ağırlıkların değişim miktarı hesaplanır. Bu değişim miktarı ağırlıklara eklenir. Bölüm 5'te formüller açıklandığından burada tekrar edilmeyecektir. Elman ağında ağırlıkların değiştirilmesinde herhangi bir farklı durum yoktur. Burada dikkat edilmesi gereken geri dönüşüm ağırlıklarının değerlerinin sabit olduğu ve değiştirilmeyeceğidir. Yani ağırlıkların değerleri değiştirilirken geri dönüşüm ağlarını 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 elemanı olarak düşünülmesi halinde Elman ağının bir ÇKA ağı ile aynı olduğuna dikkat ediniz.

8.1.3. Geri Dönüşümlü Ağların Uygulama Alanları

Elman ağı özellikle birinci dereceden doğrusal sistemleri modellemekte başarılı bir şekilde uygulanmaktadır. Bu ağlar, zamana bağlı olayları işleyebilme ve önceki zamanlarda elde edilen sonuçları bir sonraki zamanlara taşıyabilme yeteneği ile özellikle konuşma anlama ve ses tanıma problemlerinde etkin olarak kullanılmaktadır [4]. Benzer şekilde bu ağların olayların zamana bağlı ilişkilerini dikkate almaları, olayların bugün ki gidişatlarına göre gelecekte nasıl olacaklarının tahmin edilmesinde de etkin olarak geri beslemeli ağlar, özellikle de Elman ağı, kullanılmaktadır. Çünkü zamana bağlı olaylarda öğrenme olayında geçmişten geleceğe olayların gidişatının dikkate alınması gerekmektedir.

8.2. Diğer Yapay Sinir Ağı Modelleri ve Son Araştırmalar

Bu bölüme kadar yapay sinir ağları ve öğrenmeleri konusunda ayrıntılı bilgiler verilmiştir. Sadece 3 model ayrıntılı olarak tanıtılabilmiştir. Yapay sinir ağlarında çok sayıda değişik model geliştirilmiştir. Bu bölümde bu modellerin bazılarına genel bir bakış yapılacaktır.

8.2.1. Diğer Yapay Sinir Ağı Modelleri

1940'lı yıllarda başlayan yapay sinir ağları günümüze kadar oldukça önemli gelişmelere neden olmuştur. Yapılan çalışmalar sürekli değişik modellerin ortaya çıkmasına ve çözülemeyen sorunlar teker teker çözülmeye başlamıştır. Bölüm 3'ten hatırlanacağı gibi, bir ağın sahip olduğu proses elemanlarının yapısı, aktivasyon fonksiyonu, toplama fonksiyonu, ağın kullandığı öğrenme kuralı ve stratejisi, ağın topolojisi gibi faktörler farklı ağ modellerinin oluşmasına neden olmuştur. Bu bölümde, özellikle bilim dünyasında yaygın olarak kullanılan şu ağlar kısaca gözden geçirilecektir.

- *Hopfield* ağı
- *Counterpropogation* ağı
- *Cognitron* ağı
- SOM

Bunların dışında BAM, RBNN, PNN, *Boltzman* makinesi, Optik sinir ağları, Gerçek zamanlı geri dönüşüm ağları vb. gibi başka modeller de geliştirilmiştir. Bunlar o veya bu şekilde buraya kadar anlatılan modellerin geliştirilmeleri sonucunda elde edilmiştir.

8.2.2. *Hopfield* Ağı

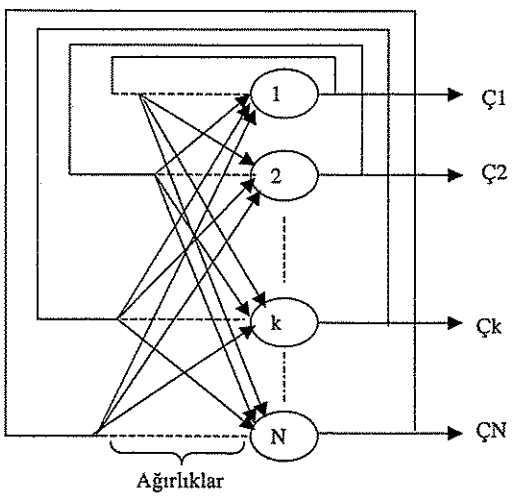
Hopfield ağı tek katmanlı ve geri dönüşümlü bir ağdır. Proses elemanlarının tamamı hem girdi hem de çıktı elemanlarıdır. Ağın bağlantı değerleri bir enerji fonksiyonu olarak saklanmaktadır. *Hopfiel* tarafından geliştirilen *Hopfield* ağı hakkında ayrıntılı bilgi [5] nolu referansta bulunabilir...

Günümüzde geliştirilmiş iki tür *Hopfield* ağı ardır:

- Kesikli (*discrete*) *Hopfield* Ağı: Bu ağlar çağrışımlı bellek (*associative memory*) olarak kullanılırlar.
- Sürekli (*continuous*) *Hopfield* Ağı: Bu ağlar ise daha çok kombinatoriyel optimizasyon problemlerinin çözümünde kullanılmaktadırlar.

8.2.2.1. Kesikli *Hopfield* Ağı

Bu ağdaki her hücrenin iki değeri vardır. Hücre *on* (+1) veya *off* (-1) olabilir. *N* proses elemanından oluşan ağ Şekil-8.4'de verilmiştir:



Şekil-8.4. *Hopfiel* ağının yapısı

Bir proses elemanının *t.* zamandaki girdisi *G(t)* olarak gösterilirse bu;

$$G_k(t) = \sum_{j \neq k}^{N} A_{jk} Ç_j(t-1) - \theta_k$$

formülü ile hesaplanmaktadır. Burada kullanılan *A* ağırlık değerini, *Ç(t-1)* proses elemanının bir önceki zaman dilimindeki çıktısını θ ise sabit eşik değerini göstermektedir.

Aynı proses elemanının çıktısı; *Ç(t)* ise şöyle hesaplanır:

$$Ç(t) = \text{sgn}(G(t))$$

Buradaki *sgn signum* fonksiyonunu göstermektedir. Yani,

$$Ç_k(t) = \begin{cases} +1 & \text{Eğer } G_k(t) > U_k \\ -1 & \text{Eğer } G_k(t) > U_k \\ Ç_k(t-1) & \text{Aksi halde} \end{cases}$$

Burada kullanılan U değeri de bir eşik değeridir. Pratikte 0 değeri seçilmekle beraber böyle bir zorunluluk yoktur. Çağrışımlı bellekte de olduğu gibi *Hopfield* ağının eğitilmesinde de ÇKA ağlarında olduğu gibi iki faz vardır:

- Ağırlıkları belirleme ve saklama fazı
- Bilgilere ulaşma fazı.

Ağırlıkların Belirlenmesi ve Bellekte Saklanması

Ağırlıkların belirleme fazı ağın öğrenme aşamasını göstermektedir. Ağın eğitimi bir defada öğrenme prensibine göre aşağıdaki formül kullanılarak gerekleşmektedir:

$$A_{ij} = \begin{cases} \frac{1}{N}\sum_{p=1}^{M} \chi_i^p \chi_j^p & \text{Eğer } j \neq k \text{ ise} \\ 0 & \text{Aksi halde} \end{cases}$$

Burada *M* öğrenilecek (saklanacak) örnek sayısının göstermektedir. χ ise bir örneğin *i.* ve *j.* elemanının değerlerini göstermektedir. Bu ağırlıklar hesaplandıktan sonra sabitlenirler. Dikkat edilirse burada A_{ij} ile A_{ji} ağırlıkları aynıdır. Bu da oluşturulan Ağırlık matrisinin simetrik bir matris olması demektir.

Ağın kullanılabilmesi için durağan *(stable)* hale gelmiş olması gerekmektedir. Bu ise şu şekilde sağlanabilir.

Bilgilerin Çağrılması

Bu fazda, ağa daha önce görmediği yani eğitim setinde olmayan bir örnek gösterilir. Bu örnek eksik bilgiler içerebilir. Ağın görevi bu eksiklikleri belirlemek ve örneğin tamamını hafızadan bulmaktır. Bunun için verilen örnek ağa sunulur ve ağın iterasyonlar yaparak durağan hale gelmesi beklenir. Ağ durağan hale gelince ürettiği çıktı, ağın kendisine gösterilen girdiye ürettiği cevabı olarak görülür. Girdi örneği *X* (*X*1, *X*2, *X*3,.....*X*N) başlangıç değerlerine atanmak üzere ağın iterasyonları yukarıda da gösterilen şu formüle göre devam eder.

$$Ç_k(t+1) = \text{sgn}\left(\sum_{j=1,\, j\neq k}^{N} A_{jk} Ç_j(t) - \theta_k\right)$$

Bu formülün çalışabilesi için girdi vektörü, ağın başlangıç çıktı değerleri olarak atanır. Yani,

Ç(0) = X = (X1, X2, X3,...XN)

olarak alınır. Ağın durağan hale gelmesi bir enerji fonksiyonunun değerinin en azlanması demektir. Bu enerji fonksiyonu;

$$E(t) = -\frac{1}{2}\sum_{i=1,\, j=1, j\neq i}^{N} A_{ij} Ç_i(t) Ç_j(t) + \sum_{i=1}^{N} Ç_i(t)\theta_i$$

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

8.2.2.2. Sürekli *Hopfield* Ağı

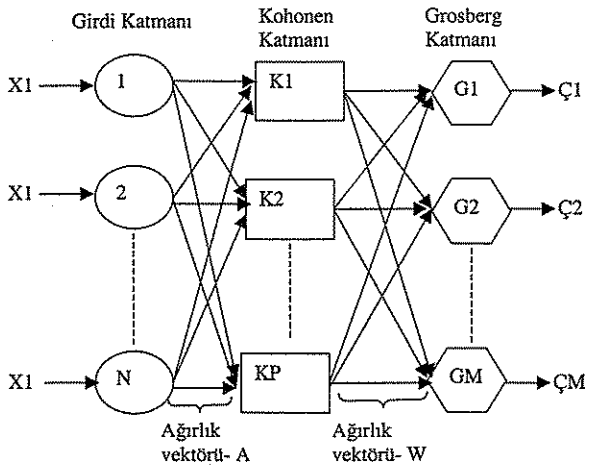
Bu ağlar kesikli hopfield ağlarının aynısıdır. Aradaki fark ise *signum* fonksiyonunun yerine *sigmoid* fonksiyonunun kullanılmasıdır. Bu durumda ağın çıktı değerleri 0-1 arasında sürekli değerler olabilmektedir.

Hopfield ağının en önemli uygulamalarından birisi geleneksel optimizasyon algoritmaları ile çözümü mümkün olmayan veya 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. Bu problemin *Hopfield* ağı ile çözülmesi [6] nolu referansta ayrıntılı olarak anlatılmıştır.

8.2.3. *Counterpropogation* Ağı

Bu ağlar `Robert Hect-Nielsen` tarafından geliştirilmiştir. Ayrıntılı bilgiler [7] nolu referansta verilmiştir. ÇKA ağına benzer bir yapısı vardır. Fakat ÇKA ağından daha çok hızlı öğrenebilmektedir. Özellikle uzun eğitim zamanlarına tolerans tanıtmayan problemlerin çözülmesinde etkin olarak kullanılmaktadırlar.

Counterpropogation ağları bilinen `Kohonen` ve `Grosberg` algoritmalarının birleştirilmesi ile oluşturulmuştur. Bu ağ `Kohonen` ve `Grosberg` ağlarının olumlu yanlarını bir araya getirmiştir. Ağın girdileri hem sürekli değerler hem de ikili değerler olabilir. Ağ eğitildikten sonra, eksik ve kısmen yanış değerler içeren girdiler için doğru çıktı vektörlerini oluşturabilmektedir. Bu özelliği ile özellikle şekil tanıma problemlerinde etkin olarak kullanılan bir ağdır. Şekil-8.5 bir *Counterporpogation* ağının yapısını göstermektedir.



Şekil-8.5. *Counterpropogation* ağının genel yapısı

Şekilden görüldüğü gibi *Counterpropogation* 3 katmandan oluşmaktadır:

- ***Girdi katmanı:*** ÇKA ağlarında olduğu gibi sadece bilginin transfer edilmesini sağlar. Herhangi bir bilgi işleme olmaz.
- ***Kohonen katmanı:*** Bu katman girdi vektörünün sınıflandırılmasını sağlamaktadır.
- ***Grosberg katmanı***: Bu katman ağın çıktısını belirler.

Diğer ağlar gibi *Counterpropogation* ağıda hem eğitim modunda hem de normal modda çalışır. Eğitim modunda ağın ağırlıkları değiştirilirken normal modda verilen girdilere karşılık gelecek çıktı değerleri belirlenir.

8.2.3.1. *Kohonen* Katmanın Çalışması

Kohonen katmanındaki proses elemanları birbirleri ile yarışırlar ve yarışmayı kazanan prosess elemanı 1 değerini alırken diğerleri 0 değerini alırlar.

Girdi vektörü X, n elemandan oluşan bir vektör olursa ve ağırlık vektörü de A ile gösterilirse; *Kohonen* katmanındaki bir proses elemanına gelen NET bilgi,

$$NET_j = \sum_{i=1}^{N} X_i A_{ij}$$

şeklinde hesaplanır. *Kohonen* katmanındaki bütün proses elemanlarının NET girdileri hesaplandıktan sonra bunların en büyüğüne sahip olan proses elemanı yarışmayı

kazanan eleman olarak belirlenir ve bu elemanın çıktısı 1 diğerlerininki ise 0 olarak atanır. Bu çıktı değerleri daha sonra *Grosberg* katmanına girdi olarak gönderilir.

8.2.3.2. *Grosberg* Katmanının Çalışması

Grosberg katmanındaki proses elemanlarının NET girdileri *Kohonen* katmanından gelen bilgilere göre hesaplanır. p elemandan oluşan *Kohonen* katmanının çıktısı K vektörü ($k1$, $k2$,..kp) ile *Kohonen* katmanı ile *Grosberg* katmanı arasındaki ağırlıklar da W vektörü ile gösterilirse, *Grosberg* katmanındaki bir proses elemanının NET girdisi,

$$NET_r = \sum_{j=1}^{P} K_r w_{rj}$$

olmaktadır. *Kohonen* katmanındaki elemanların sadece birisi 1 diğerleri 0 çıktı değerin alacağından aslında bu formül, sadece *Kohonen* katmanındaki kazanan eleman ile *Grosberg* katmanındaki elemanların arasındaki ağırlıkların hesaba katılacağını göstermektedir.

8.2.3.3. *Kohonen* Katmanını Eğitilmesi

Bunun için öncelikle ağın ağırlıklarına başlangıç değerlerinin atanması gerekmektedir. *Kohonen* katmanı ağırlıkların (girdi katmanı ile *Kohonen* katmanı arasındaki ağırlıkların) başlangıç değerlerinin belirlenmesi için farklı stratejiler uygulanmaktadır. Bunlardan bazıları şunlardır:

- Başlangıç değerleri olarak rasgele değerleri atamak. Bu durumda rasgele değerlerden oluşan ağırlık vektörlerinin normalize edilmesi gerekmektedir. Normalizasyonu şu formüle göre yapmak mümkündür.

  $$X_i' = \frac{X_i}{\sqrt{x_1^2 + x_2^2 + ... + x_n^2}}$$

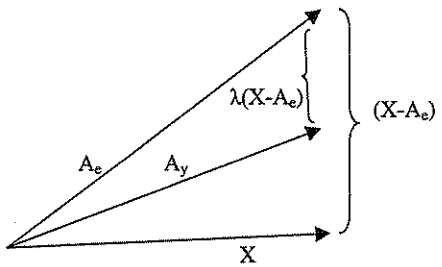
  burada xi girdi vektörünün i. elemanını göstermektedir.
- Örnek setinden bazı örnekler ağırlıkların başlangıç değerleri olarak seçilebilirler.
- Bütün ağırlık değerlerine sabit olarak $1/\sqrt{n}$ değeri atanır. Burada n girdi elemanı sayısını göstermektedir.

Ağırlıkların başlangıç değerlerinin atanmasından sonra eğitime başlanabilir. Girdi vektörü ağa gösterilir ve *Kohonen* katmanındaki proses elemanlarının NET girdileri yukarıda anlatıldığı gibi hesaplanarak kazanan eleman belirlenir. Sadece kazanan elemanın ağırlıkları değiştirilir. Ağırlıkların yeni değerleri şu şekilde hesaplanır.

$$A_y = A_e + \lambda(X - A_e)$$

Buradaki X girdi vektörünü A ağırlık vektörünün değiştirilmeden önceki (A_e) ve sonraki (A_y) değerlerini, λ ise öğrenme katsayısını gösterir. Ağırlıkların bu şekilde

değiştirilmesi öğrenme devam ettiği sürece tekrarlanır. *Kohonen* katmanındaki kazanan eleman için ağırlıkların değişimi şematik olarak Şekil-8.6'da gösterilmiştir.



Şekil-8.6. *Kohonen* katmanındaki ağırlık değişimi

8.2.3.4. *Grosberg* Katmanının Eğitilmesi

Grosberg katmanının öğrenmesinde oldukça basit bir yöntem uygulanmaktadır. Girdi vektörü ağa gösterildikten sonra önce *Kohonen* katmanı çıktıları ondan sonrada *Grosberg* katmanı çıktısı (yani ağın çıktısı) yukarıda anlatıldığı gibi hesaplanır. Ağın çıktısı *Kohonen* katmanındaki kazanan elemanın ağırlıkları ile hesap edildiğinden sadece bu ağırlıkların değiştirilmesi söz konusudur. Diğer ağırlıklar değiştirilmezler. Ağırlıkların değiştirilmesi şu formüle göre yapılmaktadır.

$$W_{yeni} = W_{eski} + \beta(Ç - W_{eski})K$$

Burada *K*, *Kohonen* katmanındaki elemanların çıktısıdır. Sadece bir tanesi 1 diğerleri sıfır olduğundan bu formül sadece kazanan elemanın ağırlıklarının değişmesine olanak verir. *Ç* ağın çıktı vektörüdür. β ise 0-1 arasında bir değer olup zamanla azalan bir katsayıdır. Bu formül örnek setindeki örneklerin ortalama değerlerinin belirlenmesine ve ağın o değerler etrafında durağanlaşmasına neden olacaktır.

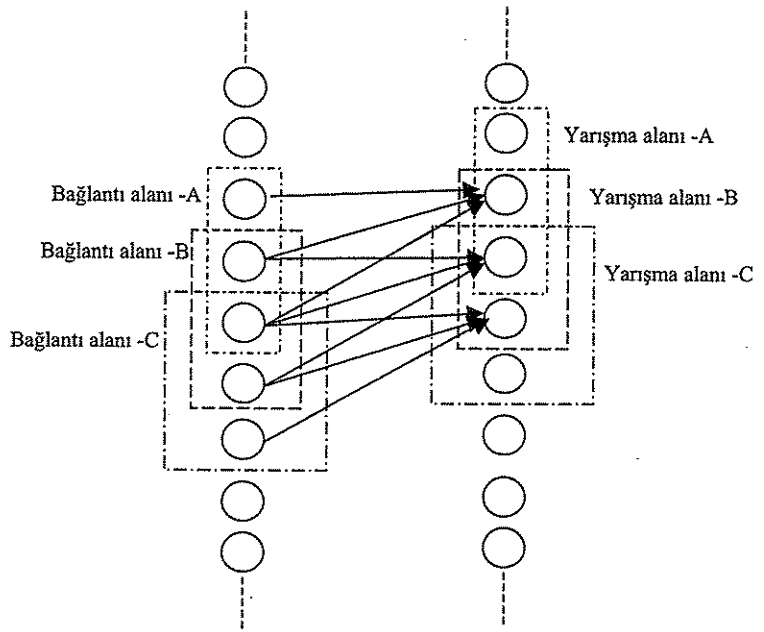
Counterpropogation ağının uygulamalarına örnekler [7] nolu referansta verilmiştir.

8.2.4. *Cognitron* ve *Neocognitron* Ağları

8.2.4.1. *Cognitron* Ağı

Cognitron ağı Fukushima tarafından insan beyninin görsel sisteminin görevini üstelenmek amacı ile geliştirilmiştir. Ayrıntıları [8] nolu referansta açıklanmıştır. *Cognitron* ağıda *Conuterpropogation* ağı gibi iki katmandan oluşmaktadır. Birinci katmanında bağlantı alanları oluşmaktadır. Birbirine yakın ve komşu olan proses elemanları bir bağlantı alanının içinde düşünülmektedir. İkinci katman ise yarışma alanlarına bölünmüştür. Yarışma alanının her birinde bulunan proses elemanları bağlantı alanlarından gelen bilgilere göre yarışmaktadırlar. Her elemana bağlantı alanlarında

bulunan yakın proses elemanları bağlanmıştır. Şekil-8.7 bağlantı, yarışma alanları ve proses elamanlarının bağlantılarını göstermektedir.



Şekil-8.7. *Cognitron* ağında bağlantı ve yarışma alanları

Bu ağda uyarıcı (*excitatory*) elemanlar ve men-edici (*inhibitory*) elemanlar olmak üzere 2 tür proses elemanı vardır. Bir elemanının çıktısı kendisine gelen uyarıcı ve men-edici sinyallere göre belirlenmektedir.

Uyarıcı Proses Elemanları

Uyarıcı elemanların çıktıları uyarıcı sinyallerin men-edici işaretlere oranına göre belirlenmektedir. Bir proses elemanına gelen toplam uyarıcı işaret (*E*) birinci katmandaki uyarıcı elemanlardan gelen işaretlerin ağırlıkları ile çarpılıp toplanması sonucu bulunur. Bu şu formül ile verilmektedir:

$$E = \sum_i \alpha_i u_i$$

Burada α uyarıcı proses elemanından gelen ağırlık değerini u ise o elemanın çıktısını göstermektedir. Benzer şekilde bir proses elemanına gelen men-edici işaretlerin

toplamı (I) birinci katmandaki men edici sinyallerin ağırlıkları ile çarpılıp toplanması sonucu bulunur. Bu,

$$I = \sum_i b_i v_i$$

şeklinde gösterilir. Burada b birinci katmandaki men edici elemanlardan gelen bağlantının ağırlığını v ise men-edici elemanların çıktılarını göstermektedir.

Cognitron ağında ağırlıklar sadece pozitif değerler aldıklarını belirtmekte yarar vardır. Bir proses elemanına gelen NET bilgiler şu şekilde hesaplanır:

$$NET = [(1+E)/(1+I)]-1$$

Proses elemanının çıktısı ise şöyle hesaplanır:

$$Ç = \begin{cases} NET & \text{Eğer NET} >= 0 \\ 0 & \text{Eğer NET} < 0 \end{cases}$$

Eğer NET>0 ise o zaman;

$$Ç = (E-I)/(1+I)$$

olacaktır. Men-edici işaretin ihmal edilebilir düzeyde çok küçük olması durumunda,

$$Ç = (E-I)$$

şeklinde yazılabilir. Ağırlıklar da negatif değerler olmaması sürekli artmalarına neden olmaktadır. Yüksek değerlerdeki uyarıcı ve men-edici işaretlerin sınırlandırılması için uyarıcı işaretlerin men-edici işaretlerin oranı kullanılarak çıktı hesaplanır. Bu ise şu formül ile verilmektedir:

$$Ç = (E/I)-1 \quad \text{Eğer E ve I'nın değeri 1'den çok büyük ise}$$

Eğer hem uyarıcı hem de men-edici işaretlerin aynı oranda (X miktarında) artıyorsa; o zaman,

$$E = pX$$
$$I = qX$$

olarak hesaplanır. Burada p ve q sabit değerler olup şu eşitliği sağlamaktadırlar:

$$Ç = [(p-q)/2q][1+\tanh[\log(pq)/2]]$$

Men-Edici Proses Elemanları

Congintronda birinci katmanda men-edici proses elemanları vardır. Eğitim sırasında bu elemanlara gelen ağırlıklar değiştirilmezler. Bu ağırlıkların değerleri bir men-edici elemana gelen ağırlıkların toplam 1 olacak şekilde belirlenirler. Bu kısıtlama ile bir men-edici elemanın çıktısı (O) o elemanın bağlı olduğu uyarıcı elemanlarının çıktılarının ağırlıklandırılmış toplamıdır. Yani,

$$O = \sum_i s_i Ç_i$$
$$\sum_i s_i = 1$$

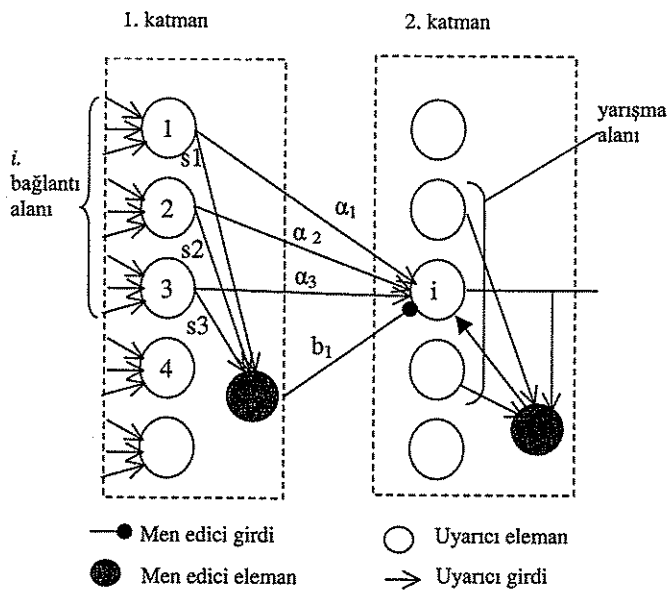
Buradaki s men-edici ağırlıkları göstermekte olup toplamı 1'e eşittir.

8.2.4.2. *Cognitronun* Eğitilmesi

Cognitron ağında sadece uyarıcı ağırlıklar değiştirilmektedir. Bir uyarıcı proses elemanının komşularından daha kuvvetli bir uyarıcı işaret üretmesi durumunda ağırlıklarının değiştirilmesi kastedilmektedir. Şekil-8.8'de gösterilen bu durumdaki ağırlıkların (α) değişim miktarı şu şekilde hesaplanmaktadır.

$$\Delta\alpha_i = q s_j u_j$$

Burada s birinci katmandaki men-edici elemandan men-edici elemana gelen ağırlık değeridir. u birinci katmandaki elemanın çıktısını göstermektedir. α uyarıcı ağırlık değerini, q ise öğrenme katsayını göstermektedir. Şekil-8.8 bir *Cognitron* ağını ve ağırlıkları göstermektedir.



Şekil-8.8. *Cognitron* ağı ve katmanları

Şekil-8.8'te görülen ikinci katmandaki *i.* elemana gelen ağırlık değerinin (*b*) değişimi ise şu formül ile verilmektedir.

$$\Delta b_i = q(\sum_j a_j u_j) / 2O_i$$

Eğer yarışma katmanında herhangi bir eleman yarışmayı kazanamaz ise o zaman ağırlıkları değişimi şu şekilde olmaktadır.

$$\Delta a_i = q' s_. u_j$$

$$\Delta b_i = q' O_i$$

Buradaki q' pozitif bir öğrenme katsayısı olup q değerinden daha küçük bir değerdir.

Şekil-8.8'de ikinci katmanda bulunan elemanların komşu elemanlardan men-edici işaretleri gönderen bir men-edici eleman vardır. Bu men-edici eleman komşu hücrelerden yarışma alanında ki konmuş elemanlardan gelen bilgileri toplar ve ilgili proses elemanına men-edici işareti gönderir. Bu elemana men-edici proses elamanı denilirse, girdisi (*MG*) şöyle hesaplanır:

$$MG = \sum_i g_i P_i$$

Burada *Pi* ikinci katmandaki *i.* proses elemanının çıktısını, g_i ise ikinci katmandaki men-edici elemana gelen bağlantının ağırlık değerini göstermektedir. *g* değerleri toplam 1 oiacak şekilde belirlenmektedir. Men-edici proses elemanının çıktısı ise şöyle belirlenmektedir.

$$\dot{C_i} = [(1 + C_i)/(1 + MG)] - 1$$

Bu men-edici sinyallerin temel görevi ikinci katmandaki proses elemanlarından sadece birisinin sürekli yarışmayı kazanmasını önlemek ve hepsinin eğitilmesini sağlamaktır.

8.2.4.3. ***Neocognitron***

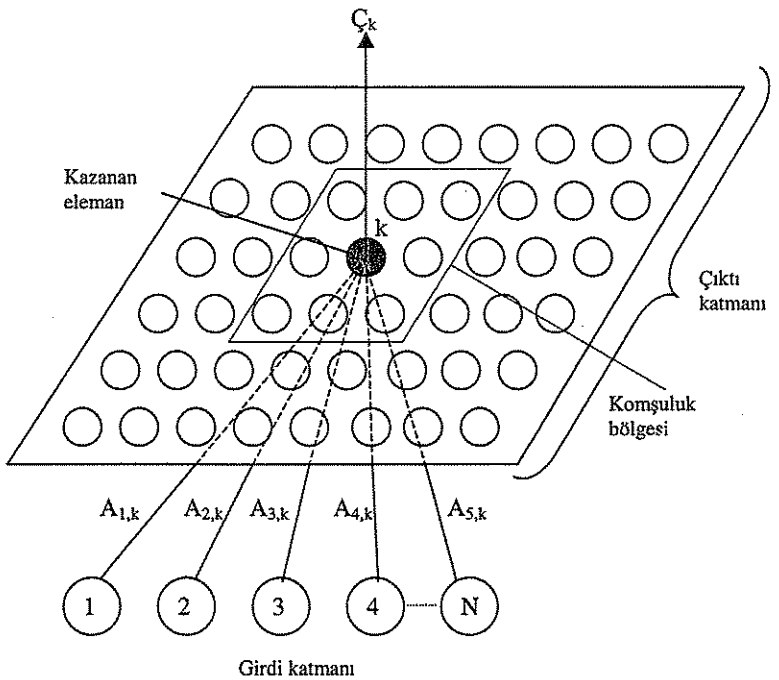
Fukushima ve çalışma arkadaşları daha sonra *cognitron* modellerini geliştirerek *neocognitron* modelini oluştururlar. Temel yapısı benzer olmakla birlikte Bu model *cognitron* modelinden daha güçlü bir ve etkin bir modeldir. Resimlerin rotasyonları, bozulmalar, aynı şeklin değişik pozisyonlarda gösterilmesi gibi durumlarda daha doğru tanıma yapabilmektedir. İnsan görme sistemini dağa gerçekçi bir şekilde modellemektedir. Bu modelin ayrıntılı açıklaması [9] nolu referansta verilmektedir.

8.2.5. **SOM**

SOM ağları Kohonen tarafından geliştirilmiştir. Genel olarak sınıflandırma yapmak için kullanılmaktadırlar. Bu ağların girdi vektörlerini sınıflandırmak ve girdi vektörlerinin dağılımını öğrenebilme yetenekleri çok yüksektir. Ayrıntılı bilgiler [10] nolu referansta bulunabilir.

Bu ağların en temel özelliği olayları öğrenmek için bir öğretmen veya ağın üretmesi gereken çıktıların ağa söylenme zorunluluğunun olmamasıdır. Özellikle beklenen çıktıların belirlenemediği problemler için kullanılmaktadır.

Yapısal olarak da bu ağlar diğerlerinden farklıdır. Ağ girdi ve çıktı katmanından oluşmaktadır. Çıktı katmanı 2 boyutlu bir düzlemi göstermektedir. Proses elemanları bu düzlem üzerine dağılmış vektörleri gösterirler. SOM ağları yarışmayı kazanma ve kazanan elemanın 1 diğerlerinin 0 değerini alması ilkesine dayanmaktadır. Bir girdi verildiğinde çıktı uzayında yarışmayı kazanan ve onun etrafındaki komşuları eğitim sırasında ağırlıklarını değiştirmektedir. Som ağının anlatılan bu yapısı Şekil-8.9'da verilmiştir.



Şekil-8.9. SOM ağının gösterimi

Şekilde sadece kazanan eleman ile girdi elemanlarının arasındaki bağlantılar gösterilmiştir. Aslında girdi elemanları çıktı elemanlarının tamamına bağlıdır.

8.2.5.1. SOM Ağının Eğitilmesi

Herhangi bir t zamanında örnek setinden bir örnek ağa gösterilir. Girdi vektörü X ve ağırlık vektörü A normalize edilmiş olmalıdır. Çıktı elemanlarından kazana eleman bulunur. Bunun için iki yoldan birisi kullanılmaktadır:

❶ Her elemanın çıktısı ($Ç$) ağırlıklarla girdilerin çarpımının toplamı ile bulunur. Yani,

$$Ç_i = \sum_i A_{io} X_i$$

Bu çıktı değerlerinden en yüksek değere sahip olan proses elemanı yarışmayı kazanmaktadır. Bu eleman k. eleman olması durumunda,

$$Ç_k = 1$$
$$Ç_i = 0 \quad i=1.2... \text{ ve } i \neq k$$

❷ *Euclid* mesafesi (d) kullanılarak girdi vektörüne en yakın ağırlık vektörüne sahip elemanı kazanan elemandır. İki vektör arasındaki mesafe,

$$d_j = \|X - A_j\|$$

şeklinde hesaplanır. Her çıktı elemanı için bu mesafeler hesaplanır ve en küçük mesafe değerine sahip eleman kazanan elemana olarak belirlenir.

Kazanan eleman belirlendikten sonra bu eleman ve komşularının ağırlıkları şu formüle göre değiştirilmektedir.

$$A(t+1) = A(t) + \lambda g(i,k)(X(t) - A(t))$$

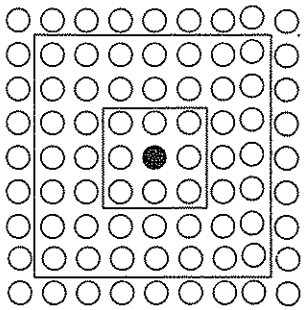
Burada λ öğrenme katsayısıdır. Öğrenme anında zamanla küçültülmektedir. $g(i,k)$ ise komşuluk fonksiyonudur. i ve k elemanlarının komşuluklarını belirler. $i=k$ durumunda $g(i,k)=1$ olur. Bu fonksiyon zaman içerisinde azalan bir fonksiyondur. Genel olarak $g(i,k)$,

$$g(k) = (\exp(-\|d_i - d_k\|^2 / (2\sigma^2))$$

şeklinde verilmektedir. Formülde, d_i ve d_k i. ve k. elemanların pozisyonunu gösteren vektörler, σ ise komşuluk alanının genişliğin göstermektedir. Zaman içersinde azalmaktadır.

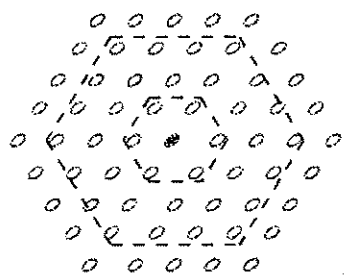
Bir proses elemanının komşularını (onunla beraber) ağırlıkları değişecek olanları belirlemede iki yöntem kullanılmaktadır:

> Proses elemanının etrafındaki elemanları bir kare/dikdörtgen içinde belirlemek: Bu Şekil-8.10'de gösterilmiştir. Şekildeki kırmızı dolu elemanın kazanan eleman olduğu varsayılırsa komşuları dörtgen içinde gösterilen elemanlardır.



Şekil-8.10. Dörtgen komşuluk alanı

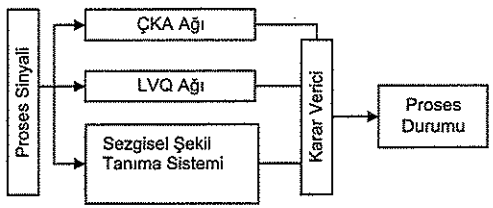
- Proses elemanının etrafındakileri bir çokgen içinde belirlemek: Bu Şekil-8.11'de gösterilmiştir. Şekildeki kırmızı dolu elemanın kazanan eleman olduğu varsayılırsa komşuları dörtgen içinde gösterilen elemanlardır.



Şekil-8.11. Çokgen komşuluk alanı

8.2.6. Karma ve Bileşik Ağlar

Günümüzde problemlerin karmaşıklığı nedeni ile bazı durumlarda tek bir yapay sinir ağı kullanmak yerine birden fazla yapay sinir ağını kullanmak daha verimli sonuçların oluşmasına neden olmaktadır. Bazı durumlarda Yapay sinir ağları geleneksel yöntemler (istatistik ve optimizasyon yöntemleri gibi) ile birlikte kullanılarak geliştirilen sistemlerin performansları yükseltilmektedir. `Öztemel` birden bir imalat sisteminde kalite kontrol şemalarındaki şekillerin tanınması için bir sistem geliştirmiş ve bu sistemde hem LVQ, hem ÇKA hem de geleneksel istatistik yöntemlerini birlikte kullanmıştır. Şekil-8.12. bu sistemi şematik olarak göstermektedir. Tek tek sistemlerin performansları %95 civarında iken birleşik ve karma sistemin performansı %99.7'lere kadar çıkmıştır. Bu sistemin ayrıntıları [11] nolu referansta bulunabilir.



Şekil-8.12. Karma yapay sinir ağı modeli

Şekil-8.12'de gösterilen sistemde, proses üzerinde yapılan ölçümler ile oluşturulan işaret hem ÇKA ağını hem LVQ ağına hem de sezgisel şekil tanıma sistemine girdi olarak gitmektedir. Bu sistemlerin hepside proses işaretine bakarak prosesin kontrol altında olup olmadığını belirlemekte ve kararlarını karar vericiye göndermektedirler. Karar verici, sistemlerin kararlarını birleştirerek prosesin durumunu belirlemektedir. Böylece sistemlerin birisi prosesin durumunu belirlemede hata yapsa bile diğerleri doğru analiz yaptığından karar verici çoğunluğun kararına uysa dahi doğru karar verebilmektedir. Karar verici ÇKA, LVQ ağı ve sezgisel yöntemin çıktılarının ortalamasını alarak ta proses ile ilgili karar verebilir. Problemin türüne göre ortak bir karar verme mekanizması oluşturulabilir. Karma sistemi oluşturmak için sadece 3 ayrı sistem olması koşulu yoktur. İstenilen sayıda ağ ile karma sistem oluşturulabilir. Bir sonraki bölümde özellikle birleşik ağlar ayrıntılı olarak açıklanmıştır.

8.3. Özet

Geri dönüşümlü ağlarda proses elemanlarının çıktıları sadece ileri doğru değil aynı zamanda geri doğruda gönderilmektedir. Genel olarak tam geri dönüşümlü ve kısmi geri dönüşümlü ağlardan söz etmek mümkündür. Elman ağı kısmı geri dönüşümlü ağlara verilecek en güzel örnektir. *Elman* ağı bilinen ÇKA ağlarına benzer bir şekilde tasarlanmış ve Genelleştirilmiş Delta kuralına göre eğitilirler. Bu ağlarda ÇKA ağlarından farklı olarak Çıkarım elemanları denilen proses elemanları vardır. Bu elemanların görevi ara katman elemanlarının çıktılarını ağa tekrar girdi olarak göndermektir. Ağın çıktısının hesaplanması ile ilgili formüller bölüm içinde verilmiştir. Ara katman elemanlarını içerik elemanlarına bağlayan bağlantıların ağırlık değerleri sabit olup 1'e eşittir. O nedenle ağın eğitimi sırasında beklenen çıktı ile gerçekleşen çıktı arasındaki hatanın ağa yayılmasında bu bağlantılar dikkate alınmadıklarından ÇKA ağının öğrenmesi aynı şekilde burada da uygulanabilmektedir.

Günümüzde yaygın olarak kullanılan diğer ağlara örnekler ise şunlardır:

- *Hopfield* ağı
- *Conterpropogation* ağı
- *Cognition* ve *Neocognition*
- SOM ağı

Hopfield ağının en temel özelliği proses elemanlarının hepsinin birbirine bağlı olmasıdır. Ağ bağlantı değerleri bir enerji fonksiyonu olarak saklanmaktadır. İki türlü model vardır. Kesikli ve sürekli *Hopfield* ağları. Bu ağlar özellikle geleneksel yöntemler ile çözülmesi zor optimizasyon problemlerinin çözülmesinde kullanılabilmektedirler.

Counterpropogation ağı ise *Kohonen* ve *Grosberg* öğrenme kurallarının birleştirilmesi ile oluşturulmuş bir ağdır. Her iki öğrenme kuralında avantajlarını bir arada bulunmaktadır. Bu ağlarda özellikle şekil tanıma problemlerinde başarılı bir şekilde kullanılabilmektedir.

Cognitron ve *neocognitron* ağları ise daha çok insan beyninin görsel sisteminin görevini üstlenmek üzere geliştirilmiş ağlardır. Bu ağlarda iki katmandan oluşmaktadır. Birinci katman bağlantı alanları ikinci katman ise yarışma alanlarına bölünmüştür. Bu ağlarda uyarıcı ve men edici olmak üzere iki tür proses elemanı vardır. Bir elemanın çıktısı kendisine gelen uyarıcı ve men edici sinyallere göre belirlenmektedir. Bu ağlarda daha çok tanıma problemleri için kullanılmaktadır.

SOM öğretmensiz öğrenme yapabilen ve sınıflandırma problemlerinde başarılı bir şekilde uygulanan bir ağdır. Ağın girdi ve çıktı katmanları vardır. Çıktı katmanındaki proses elemanları birbirleri ile yarışmakta ve yarışmayı kazanan eleman girdinin sınıfını göstermektedir. Ağın çıktı katmanı bir düzlem şeklinde düşünülmekte ve kazanan her proses elemanının komşularını gösteren komşuluk bölgesi oluşturulmaktadır. Komşuluk alanı içindeki bütün elemanlar kazanan eleman ile birlikte düşünülmektedir.

Bölüm içinde son olarak karma ve birleşik ağlara dikkatler çekilmiştir. Burada birden fazla ağın bir problemi çözebilmek için birlikte hareket etmeleri sağlanmaktadır. Bir sonraki bölümde özellikle bileşik ağların nasıl tasarlandığı ve problemlere nasıl çözümler ürettikleri gösterilecektir.

8.4. Kaynakça

[1] Elman J.L., (1990), Finding structure in time, Cognitive Science, Vol. 14, pp. 179-211.

[2] Pearlmutter, B.A. (1989). Learning state space trajectories in recurrent neural networks. Proceedings of the International Joint Conference on Neural Networks, Washington, D.C., II-365.

[3] Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. Institute for Cognitive Science Report 8604. University of California, San Diego.

[4] Elman, J.L. (1991b). Distributed representations, simple recurrent networks, and grammatical structure. Machine Learning, 7, 195-225.

[5] Hopfield J. J., (1982), Neural networks and physical systems with emergent collective computational abilities," Proceedings of National Academy of Science, vol. 79, (USA), pp. 2554-2558.

[6] Hopfield J.J. and Tank D.W., (1985), "Neural computation of decisions in optimization problems", Biological Cybernetics, 52, pp. 141-152.

[7] Hect-Nielsen R., (1988), "Applications of counterpropogation networks", Neural Networks, 1, pp.131-139.

[8] Fukushima K., (1975), "Cognitron- A self organizing multi-layered neural network", Biological Cyberetics, 20, pp. 121-136.

[9] Fukushima K, Miyake S., (1982), " Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position", Pattern recognition, 15(6), 455- 469.

[10] Kohonen, T., (1984). Self-Organization and Associative Memory. Berlin: Springer-Verlag.

[11] Öztemel E. (1992), "Integrating expert systems and neural networks for on-line intelligent statistical process control ", Doktora tezi, University of Wales, College of Cardiff, UK.

BÖLÜM 9

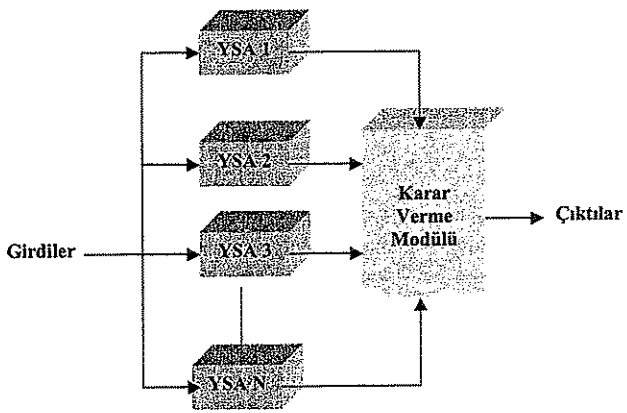
BİRLEŞİK YAPAY SİNİR AĞLARI

Bu bölüme kadar değişik yapay sinir ağlarının topolojileri, kullandıkları öğrenme stratejileri ve öğrenme kuralları ele alındı. Buraya kadar anlatılan ağların hepsinin tek başlarına problemleri çözmek üzere geliştirildikleri söylenmiştir. Yapay sinir ağlarının en iyi çözümü garanti etmediklerinde yine önemli bir özelliği olarak ortaya konuldu. Bu aslında önemli bir özelliktir. Yapay sinir ağlarının bağlantı ağırlıklarının başlangıç değerleri, eğitimde kullanılan örneklerin ağa sunuluş şekli, kullanılan öğrenme parametrelerinin belirlenen değerleri, öğrenmenin gerçekleştirildiği iterasyon sayısı vb. gibi bazı faktörler elde edilen sonuçların performansını yakından ilgilendirmektedir. Aynı ağ bu faktörlerin farklı değerlerine göre farklı sonuçlar üretebilmektedir. Örneğin sadece farklı başlangıç değerlerinden başlaması durumunda iki ağ eğitilse bir ağın tanıdığı bir örüntüyü diğer ağ tanıyamamaktadır. Ağların performansları eşit olsa bile birinin doğru sonuç ürettiği bir örnek için diğerinin farklı bir sonuç ürettiği görülebilmektedir. Bir problem için üretilen sonuçların deneme yanılma sonucu elde edilmesi en iyi topolojinin bulunmasını da zorlaştırmaktadır. Her türlü topolojiyi denemek mümkün değildir. Bu nedenlerden dolayı problemlere daha iyi sonuçlar üretebilmek için birden fazla ağın eğitilerek birlikte kullanılması ve bunların bir sinerjisini oluşturarak problemlere çözüm üretmek için birden fazla ağın birlikte kullanıldığı sistemler geliştirilmektedir.

Birden fazla ağın aynı probleme çözüm üretmek üzere geliştirildiği bu sistemlere ***"birleşik ağlar"*** denmektedir. Birleşik ağlar birden fazla uzmanın bir probleme çözüm üretmesine benzetilebilir. Bu uzmanların her birisi ilgili probleme farklı bir açıdan bakmakta ve hepsinin görüşleri bir araya getirildiğinde onların sinerjisi ile daha iyi ve doğru sonuç ortaya konulmaktadır. Birleşik ağlarda da birden fazla ağın her birisi olayın farklı bir yönünü öğrenebilmekte ve hepsinin kararları bir araya getirilerek ortak bir karar oluşturulmaktadır. Oluşturulan birleşik ağın performansı sistemi oluşturan ağların her birisinden tek tek daha yüksektir. Bu bölümde birleşik ağların yapısı, eğitilmesi ve ürettikleri çıktıların nasıl birleştirildiği konusu anlatılacaktır.

9.1. Birleşik Ağların Yapısı

Birleşik ağların en temel özelliği birden fazla ağın birlikte aynı problemi çözmek üzere eğitilmesi ve ortak bir kararın oluşturulmasıdır. Şekil-9.1 bir birleşik ağ sistemini göstermektedir.



Şekil-9.1. Birleşik sinir ağlarının elemanları

Şekilden görüldüğü gibi N adet yapay sinir ağı (YSA) bir araya gelerek aynı problemi (girdileri) çözmekte ve sonuçlarını Karar Verme Modülüne göndermektedir. Burada gelen bütün kararlar incelenerek bileşik ağın kararı oluşturulmakta ve dış dünyaya iletilmektedir.

Bileşik ağ oluşturmak için en az iki ağın bir araya gelmesi yeterlidir. Verilen kararların daha rahat ortak bir karara dönüştürülmesi için 3 adet YSA ağının kullanılması önerilmekle beraber böyle bir zorunluluk yoktur. Bazı durumlarda ağa sunulan bir örnek bir ağ tarafından tanınmaz iken diğer ağ tarafından tanınmaktadır. Üçüncü bir ağın kararı bu durumda önemli olmaktadır. Bu ağın kararı hangi ağa yakın ise o zaman birleşik ağın kararı o yönde olacaktır.

Şekilde gösterilen ağların hepsinin aynı yapay sinir ağı modeli olması gerekmez. Farklı YSA modelleri aynı problemi çözebilir. Mesela şekilleri sınıflandırma probleminde CKA, LVQ ağları bir araya gelerek bir bileşik ağ oluşturabilirler. Benzer şekilde aynı modelden oluşan bir bileşik sistem oluşturulsa da bu ağların topolojik olarak eşit olması gerekmez. Bir ağın ara katmanı bir tane olurken diğerinin iki tane olabilir. Kullanılan ağların ara katman sayıları farklı olabilir. Bunun yanında aynı problem üzerinde karar verebilmek için ağların girdi ve çıktılarının aynı olması gerekir. Burada bir noktaya daha dikkatleri çekmek gerekmektedir. Bileşik sistemlerin sadece yapay sinir ağlarından oluşması da gerekmez. İstatistiksel sistemler, uzman sistemler, klasik öğrenme sistemleri de yine bileşik sistemlerin bir parçası olabilirler. Burada önemli olan bu sistemlerinde aynı girdi hakkında fikir üretebilmeleridir.

Bileşik ağların bir özelliği de hepsinin aynı örnek üzerinde karar vermeleridir. Bu ağların birleştirilmesinin amacı zaten aynı probleme farklı açıdan bakılmasıdır. Farklı problemlere çözüm üreten ağların bir araya gelmesinin bir anlamı olmaz. Piyasada bu tür sistemlerde vardır. Ama onların tek bir sistem olması söz konusu değildir. Bu durumdaki ağlar büyük bir sistemin parçaları olarak görülebilir. Çünkü her ağın ürettiği çıktı bağımsız olarak sistem içindeki görevini yerine getirmektedir. Bileşik sistemler tam tersine aynı probleme çözüm üreten ağların bir araya gelerek oluşturdukları sistemlerdir. Ağların kendilerinin ürettiklerin bağımsız olarak kullanılması söz konusu değildir. Aslında ağların çıktıları sistemin çıktısını oluşturmak için girdi olarak kullanılmaktadır.

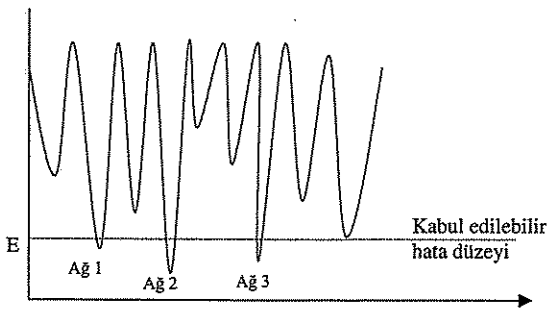
Dikkatleri çeken diğer bir önemli noktada ağların birbirleri ile direkt ilişkisinin olmamasıdır. Her ağ girdileri bağımsız olarak işlemekte diğer ağların kararlarını etkileyecek bir girişimde bulunmamaktadır. Kararların ortak olarak değerlendirilmesinde ağların etkisi görülmektedir.

9.2. Birleşik Ağların Eğitilmesi ve Test Edilmesi

Bileşik ağların eğitilmesi ve test edilmesinde herhangi özel bir algoritmaya gerek yoktur. Her ağ birbirinden bağımsız eğitilmekte ve test edilmektedir. Her ağ kendi öğrenme kuralına göre eğitilmektedir. Bileşik ağlar, eğitimini tamamlamış ve test edilmiş ağlardan oluşmaktadır. Eğer bir ağ eğitilmemiş ve test edilmemiş ise bileşik sistemde yer almamalıdır. Performansı onaylandıktan sonra ağların kullanılması daha iyi olur. Burada önemli olan bazı noktalar vardır. Bunları şu şekilde sıralamak mümkündür.

- Bileşik sisteme ait olan ağların ağırlıklarının ve a parametrelerinin (ara katmanı sayısı, her katmandaki eleman sayısı, öğrenme katsayısı vb. gibi) farklı olabilir.
- Ağları eğitmek için aynı örnek seti kullanılırsa bunlar ağlara farklı sıralarda sunulabilir. Büyük bir örnek seti parçalara bölünerek her ağ farklı eğitim setinde eğitilebilir.
- Test setine gelince mümkün olduğu sürece bütün ağları aynı test seti üzerinde test etmekte yarar vardır. Böylece ağların olayı öğrenip öğrenmedikleri birbirleri ile kıyaslanarak belirlenebilir.

Şekil-9.2 bir olayın hata uzayını iki boyutlu olarak göstermektedir. Görüldüğü gibi problemin hata uzayında hata değerleri bazen çok değerlere düşmekte bazen de oldukça büyük değerlerde kalmaktadır. Yapay sinir ağlarının görevi olayı öğrenerek en az hatalı veya kabul edilebilir bir düzeyde hata (şekilde E noktası) ile sonuçları verebilmesidir.



Şekil-9.2. Farklı ağların farklı hata düzeylerinde eğitilmesi

Şekilden de görüldüğü gibi ağlar farklı şekilde ve farklı parametreler ile aynı problem için eğitilirler ise o zaman farklı hata düzeylerini üretebilmektedirler. Ağlar tarafından hiç görülmemiş bir örnek bu ağlara sunulsa her ağ farklı sonuçlar üretebilir. Birisi örneği tanırken hiç hata yapmaz iken diğeri hiç tanımayabilir. Birleşik ağların temel görevi birbirlerinden farklı düşünen ağların bir örnek üzerindeki ortak kararını oluşturmaktır. Böylece üzerinde karar verilmesi güç örnekler için dahi ortak kararlar oluşturulabilir.

9.3. Ortak Karar Verme Modülü

Yukarıda belirtildiği gibi Karar Verme Modülü problemin nihai çözümünü hesaplamaktadır. Problemin girdileri bileşik sistemin elemanı olan her ağa bağımsız olarak sunulmakta ve ağların çıktıları belirlenmektedir. Bu çıktılar bir araya getirerek birleşik sistemin kararı yani ağların ortak kararı belirlenmektedir. Ortak kararın verilmesi ağlar arasında sağlanan ***"oy birliği"*** anlayışına dayanmaktadır. Tasarımcılar kendi sistemleri için ortak karar verme modülünü tecrübelerine dayanarak istedikleri şekilde geliştirebilirler. Önemli olan farklı ağlardan gelen sonuçları tek bir sonuca indirgemektir. 3 ağdan oluşan ve sınıflandırma yapmak üzere tasarlanmış bir birleşik ağ sistemi için aşağıdaki gibi bir karar verme modülü önerilebilir. Bu ağların çıktılarının 5 elemandan oluştuğu ve her ağın çıktılarının ise Tablo-9.1'de gösterildiği gibi verildiği varsayılsın.

Tablo-9.1. Birleşik ağların çıktıları

Ağ	Çıktı 1	Çıktı 2	Çıktı 3	Çıktı 4	Çıktı 5
YSA 1	A1	B1	C1	D1	E1
YSA 2	A2	B2	C2	D2	E2
YSA 3	A3	B3	C3	D3	E3
Eşik 1	T1				
Eşik 2	T2				

Buradaki çıktılar girdi kümesine (*set*) karşılık gelen sınıfları göstermektedir. Burada 5 sınıf vardır. Tablodaki değerler ağların kararlarını göstermektedir. Her ağ girdiyi sınıflandırmaktadır. Eğer çıktı değerinin 1 olması girdinin o sınıfa ait olmasını 0 olması ise onun gösterdiği sınıfa ait olmadığını göstermektedir.

Ortak kararın hesaplanması için ağlardan gelen ortak görüşü belirleyecek eşik değerleri belirlenmiştir. 3 ağdan oluşan bu sistemde iki adet eşik değerinden bahsetmek mümkündür. Bunlardan T1, ağların tek bir karar üzerinde hem fikir olup olmadıklarını belirler. T2 ise tek bir sonuç üzerinde hem fikir olunmaması durumunda en üzerinde en fazla hem fikir olunan sonucu belirlemek için kullanılmaktadır.

Yukarıdaki bilgiler ile oluşturulan bir bileşik ağın ortak kararı oluşturması şu şekilde olacaktır:

Adım 1: Ağlardan gelen çıktıları birbirleri ile eşlenerek toplanır.

$A = A1 + A2 + A3 + A4 + A5$
$B = B1 + B2 + B3 + B4 + B5$
$C = C1 + C2 + C3 + C4 + C5$
$D = D1 + D2 + D3 + D4 + D5$
$E = E1 + E2 + E3 + E4 + E5$

Elde edilen toplam değerlerinden en büyüğü (AA) belirlenir.

$Ç_x = max(A,B,C,D,E)$

Eğer $Ç_x > T1$ ise o zaman $Ç_x = 1$ değerini alır. Diğerleri ise 0 değerini alarak ortak karar belirlenmiş ve bir tane 1 değeri olan sonuç oluşturulmuş olur. Bu ise ağlara sunulan girdilerin, 1 değerinin gösterdiği sınıfa ait olduğunu göstermektedir. T1 değerinden büyük birden fazla proses elemanının çıktısı olursa veya hiç bir çıktı değerinin T1 değerinden büyük bir değere oluşamaması durumunda bir sonraki adım uygulanabilir.

Adım 2: Ağların çıktıları ikişer ikişer toplanır. Bileşik sistemde sadece 3 adet ağ olduğundan her bir çıktı için 3 adet ikili toplam değeri oluşacaktır. Mesela birinci çıktı için şu değerler belirlenecektir. Yani,

$Ç_{12} = A1+A2$
$Ç_{13} = A1+A3$
$Ç_{23} = A2+A3$

olacaktır. Bu ikili toplama işlemi bütün çıktılar için gerçekleştirilir. Her çıktının en yüksek değeri (bu ikili toplamların en yüksek değeri- C_{1x}) belirlenir.

$Ç_{1x} = max(Ç_{12}, Ç_{13}, Ç_{23})$

Buradaki örnekte 5 adet ikili toplama sonucu maksimum değer elde edilecektir. Bunları $Ç_{1x}$, $Ç_{2x}$, $Ç_{3x}$, $Ç_{4x}$, $Ç_{5x}$ olursa bunlarında en büyüğü hesaplanır. Buna $Ç_{xx}$ denilirse o zaman

$Ç_{xx} = max(Ç_{1x}, Ç_{2x}, Ç_{3x}, Ç_{4x}, Ç_{5x})$ olacaktır.

Eğer, $Ç_{xx}$ > T2 ise o zaman $Ç_{xx}$ = 1 olacak ve diğerleri 0 değerini alacaktır. Bu durum ağa sunulan girdilerin 1 değerini üreten çıktı elemanının gösterdiği sınıfa ait oldukları kararı verilecektir.

Adım 3: Yapılan bu değerlendirmeler neticesinde T2 değerinden den büyük bir değer elde edilmeyebilir. Bu durumda da elde edilen ikili toplam değerlerinin ortalaması alınarak ağın çıktıları belirlenir. Yani;

$Ç1 = (Ç_{1x}/2, Ç_{2x}/2, Ç_{3x}/2, Ç_{4x}/2, Ç_{5x}/2)$ olacaktır.

Ağın ortak kararını verilmesinde Adım 3'e ulaşılması, ağın ilgili girdi üzerinde ortak bir kararının oluşturamadığı anlaşılır. Aslında bu pek karşılaşılan bir durum değildir. Ağların en az iki tanesi ilgili girdi üzerinde karar oluşturabilmektedir.

9.4. Birleşik Ağların Uygulanması

Yukarıda anlatılanlar ışığında birleşik ağlar oluşturularak daha önce 6. Bölümde anlatılan kontrol şemalarının üzerindeki örüntülerin (şekillerin) tanınması örneği birleşik ağlara öğretilmiştir. Sonuçlar incelenince birleşik sistemlerin öğrenme performansına katkıları açık olarak görülmüştür. Bu kapsamda 3 tür birleşik sistem oluşturulmuştur. Bu sistemler hakkındaki ayrıntılı bilgiler [1] veya [2] nolu referanslarda bulunabilir.

1. Birleşik ÇKA Ağı Sistemi

Bu sistem 3 adet ÇKA ağından oluşmaktadır. Sisteme dahil edilen ÇKA ağları yine 6. Bölümde anlatılan eğitim ve test setleri kullanılarak eğitilmiştir. Eğitim setindeki her örnek 200 defa ağlara (200 *epoch*) gösterilmiştir. Öğrenme katsayısı olarak 0.3 ve momentum katsayısı olarak da 0.8 değerleri kullanılmıştır. Eğitim setindeki örnekler ağlara rasgele gösterilmiştir. Ağların ağırlıklarının başlangıç değerleri –1 ila 1 arasında değişen rasgele değerlerden oluşturulmuştur. Bu koşullarda eğitilen ağların sonuçlarına yukarıda anlatılan karar verme mekanizması uygulandığında elde edilen sonuçlar Tablo-9.2'de gösterilmiştir.

Tablo-9.2. ÇKA bileşik ağı sonuçları

Ağ	Sonuçlar (%) (performans)	Nihai sonuç (%) (performans)
ÇKA 1	95.2	
ÇKA 2	94.7	
ÇKA 3	94.8	
Birleşik ağ		96.8

Görüldüğü gibi bileşik sistem tek tek ağlardan daha iyi bir performans sergilemektedir. Ağların hepsi %95 civarında bir performans sergiler iken bileşik sistemin performansı %97'lere ulaşmıştır. Bu örnekte ağlar benzer şekilde farklı öğrenme setleri kullanarak da eğitilmiş ve performans %97.1'e yükseltilmiştir. Bir imalat endüstrisinde kalite

kontrolü amacı ile kullanılan bu sistemin üretilecek ürünün kalitesine katkısı da benzer şekilde daha fazla olacaktır. Üretilen ürünün insan hayatı ile ilgili olması durumunda ise bu kalitenin kontrol edilmesindeki bu performans daha fazla önem kazanmaktadır.

2. Birleşik LVQ Sistemi

6. Bölümde bu probleme LVQ ağlarının çözüm üretmesi ayrıntılı olarak anlatılmıştır. Birleşik LVQ ağının oluşturulmasında da yine aynı örnekler kullanılmıştır. Ağırlık değerleri –0.1 ile 0.1 arasında rasgele atanmıştır. Örnekler ağlara sıralı bir şekilde gösterilmiştir. Öğrenme katsayısı 0.05 alınarak zaman içinde 0.01 değerine kadar inmesi sağlanmıştır. Burada LVQ-X öğrenme kuralı kullanılmıştır. Her ağ farklı sayıda öğrenme iterasyonuna göre öğreninceye kadar eğitilmiştir. Elde edilen sonuçlar Tablo-9.3'te verilmiştir.

Tablo-9.3. LVQ birleşik ağı sonuçları

Ağ	Öğrenme iterasyon sayısı	Sonuçlar (%) (performans)	Nihai sonuç (%) (performans)
LVQ 1	20	97.7	
LVQ 2	30	97.4	
LVQ 3	70	88.4	
Birleşik ağ			99.0

Görüldüğü gibi ağın kalite problemlerini yakalama performansı %99'a kadar çıkartılmıştır. Tek bir ağ ile bu performansı %97 civarında tutabilmek mümkün olabilmiştir. Ağların ortak kararlarına bakıldığında ise performansın %99 olduğu görülmektedir.

3. Karma Birleşik Ağ Sistemi

Daha sonra hem ÇKA ve hem de LVQ ağlarından oluşan birleşik sistemler geliştirilmiştir. 2 ÇKA 1 LVQ ağının (Bileşik ÇKA+LVQ) birlikte kullanılması, 1 ÇKA 2 LVQ ağının (bileşik LVQ+ÇKA) birlikte kullanılması denenmiştir. Bu çalışmada eldeki örneklerin tanınması için bir de istatistiksel yöntemleri kullanan sezgisel yöntem geliştirilmiş ve bileşik sistemin bir parçası olarak denenmiştir. Sezgisel yöntem tek başına örneklerin (test setindeki) %94.8'ini doğru sınıflandırmış ve kalite problemi olan örnekleri belirlemiştir. Oluşturulan ağlar aynı veriler üzerinde hem de farklı eğitim setleri üzerinde eğitilmiştir. Elde edilen sonuçların hepsi toplu olarak Tablo-9.4'te verilmiştir.

Bu tabloda kullanılan ÇKA ağlarından eni iyisinin performansının %95.2, LVQ ağlarından en iyisinin performansının da %97.7 olduğunu belirtmekte yarar vardır. Görüldüğü gibi LVQ ağlarının sezgisel sistem ile birlikte kullanılmasında en iyi performansı gösteren bir bileşik sistem oluşturulmuş ve performans %99.5'e yükselmiştir. Bu aslında hiç bir ağın tek başına ulaşabileceği bir sonuç değildir. Böyle bir sisteme sahip bir imalat sisteminde şemaların üzerindeki şekillerin yorumlanması ile hemen hemen bütün kalite problemlerini belirlemek mümkün olacaktır. Bu performansın test kümesi

üzerinden elde edildiğine dikkatleri çekmek gerekir. Bütün ağlar öğrenme kümesini %100 öğrenebilmektedirler.

Sonuçta elde edilen %95 başarının neticesinde test kümesindeki 1002 adet şekilden doğru sınıflandırılamayanlar incelendiğinde şemalar üzerindeki şekillerin insan gözü ile ayrıştırılması ve sınıflandırılmasının bile zor olduğunun görüldüğü rapor edilmiştir.

Tablo-9.4. Birleşik ağları genel sonuçları

Birleşik ağ	Aynı eğitim kümesi ile performans (%)	Farklı eğitim kümesi ile performans (%)
Birleşik ÇKA	96.8	97.1
Birleşik LVQ		99.0
Birleşik ÇKA + LVQ	97.9	98.2
Birleşik LVQ + ÇKA	99.1	98.5
Birleşik ÇKA+ sezgisel	97.7	98.2
Birleşik LVQ + sezgisel		**99.5**
Birleşik ÇKA+LVQ+ sezgi.	98.9	99.0

9.5. Karma Sistemler (Uzman sistem + Yapay Sinir ağları)

Yapay sinir ağlarının yukarıda gösterildiği gibi sezgisel yöntemler ile başarılı sonuçlar olması araştırmacıların dikkatlerini çektikten sonra yapay zeka teknolojilerinden özellikle uzman sistemler ile birlikte çalışabilmeleri de incelenmiştir. Uzman sistemlerin ve yapay sinir ağlarının her ne kadar birbirinin rakibi gibi görünse de aslında bu iki teknolojinin birbirinin tamamlayıcısı olduğu görülmüştür. Karma sistemler adı verilen bu sistemlerde yapay sinir ağları ve uzman sistemler iki şekilde bir arada kullanılmaktadır.

- *Fonksiyonel olarak birliktelik*: Bu tür bir birliktelikte büyük bir problem alt parçalara bölünmekte ve parçaların her biri ya yapay sinir ağına yada uzman sisteme çözdürülmektedir. Her teknoloji kendisinin en iyi olduğu alanlarda çözüm önerileri sunmaktadır. Burada uzman sistem ile yapay sinir ağı gerekli görüldüğünde birbirleri ile bilgi alış verişlerinde bulunmaktadırlar. Genellikle uzman sistem sistemin bütününün kontrolü altında tutmakta ve yapay sinir ağını yönlendirmektedir. Bu aslında uzman sistemlerin sonuçların nasıl oluştuğunu açıklama yeteneklerinin olmasından kaynaklanmaktadır. Bu konuda ayrıntılı bilgiler ve örnekler [3-5] nolu referanslarda bulunabilir.
- *Yapısal birliktelik:* Bu tür birlikteliklerde genel olarak yapay sinir ağı bir uzman sistemin içine gömülmüştür. Her iki sistemde aynı problemi çözmek üzere geliştirilmişlerdir. Uzman sistemin bazı sorumlulukları yapay sinir ağına yüklenmiştir. Yapay sinir ağı uzman sistemin bilgi tabanı veya çıkarım mekanizmasının rolünü üstlenebilmektedir. Bu durumda yapay sinir ağının birbirine bağlı proses elemanları kuralların EĞER / O ZAMAN ilişkisini bağlantının ağırlık değeri ise ilgili kuralın belirsizlik katsayısını göstermektedir. Bu tür karma sistemler hakkındaki bilgiler ise [6-8] nolu referanslarda görülebilir.

Karma sistemlerin uygulanmasında da yine başarılı sonuçlar elde edildiği görülmüştür. Kaynakçada verilen [1] veya [2] nolu referansta bu konuda başarılı bir uygulama ayrıntılı olarak anlatılmaktadır.

9.6. Özet

Yapay sinir ağlarının parametrelerinin değişik olması ve başlangıç değerlerinin değiştirilmesi neticesinde eğitilmeleri durumunda farklı sonuçlar ürettikleri görülmüştür. Bazı durumlarda tanıyamadıkları bir şekli diğer durumlarda öğrendikleri fark edilmiştir. Bunun neticesinde aynı problemi eğitine birden fazla ağın eğitilmesinin ve sonuçlarının bir araya getirilerek ortak bir kararın oluşturulmasının mümkün olabileceği görülmüştür. Birleşik ağlar denilen bu sistemler kendilerini oluşturan ağların her birinden performansı daha yüksek sonuçları üretilmesini sağlamaktadırlar. Bileşik ağlarda sistemin içinde bulunan ağlar kadar ortak karar vermeyi sağlayan mekanizmada önemlidir. Ağlar eğitilirken ve test edilirken diğerlerinden bağımsız olarak düşünülmektedir. Bütün ağlar eğitilip performansları onaylandıktan sonra bir araya getirilmektedir. Sistemde kullanılan ağların her hangi bir modelden seçilmesi gibi bir zorunluluk yoktur. Önemli olan birleşik sistemde bulunan ağların aynı problem üzerinde eğitilmiş olmaları ve çıktılarının formatının ortak olmasıdır. Birleşik ağ içinde bulunan ağların topolojileri, kullandıkları toplama ve aktivasyon fonksiyonları, ara katman sayısı ve her ara katmanda bulunan eleman sayıları farklı olabilir. Girdi ve çıktı ünitelerinin sayılarının eşit olması gerekmektedir. Ağların birbirlerini ve karar vermelerini etkilemeleri söz konusu değildir. Karar verme mekanizması kendisine gelen ağların bağımsız kararlarını "oy birliği" mantığında bir yaklaşım ile birleştirerek ağlara sunulan girdi hakkında bileşik sistemin kararını oluşturur. Geliştirilmiş birleşik sistemlerde sadece yapay sinir ağlarını kullanmak zorunlu değildir. Sisteme geleneksel yöntemleri de katmak olasıdır. Ağın öğrendiği olay hakkında geliştirilecek bir geleneksel ve sezgisel yöntem oluşturulabilirse o da sistemin bir parçası gibi değerlendirilebilir. Yapılan denemelerde yapay sinir ağlarının sezgisel yöntemler ile birlikte kullanılmasının sistemin performansını artırmada önemli bir katkısının olduğu görülmüştür. Ağların tek tek performanslarının %97 civarında olması durumunda geliştirilen bir bileşik sistemin performansının %99.5'lere çıkması bu ağların önemini ortaya koymaktadır.

9.7. Kaynakça

[1] Öztemel E. (1992), "Integrating expert systems and neural networks for on-line intelligent statistical process control ",Doktora tezi, University of Wales, College of Cardiff, UK.

[2] Pham D.T, Öztemel E. (1996), Intelligent Quality Systems, Springer Verlag,

[3] Handelman D.A, Lane S.H. Gelfand J.J. (1990), "integrating neural networks and knowledge based systems for intelligent robotic control", IEEE control systems magazine, ss. 77-87.

[4] Garcia R. P. (1991), "A rule and neural net basef hybrid expert system for electrical transformer price estimation", Proc. Of the world congress on expert systems, Orlando, Florida, Vol. 1, 16-19, ss. 524-521.

[5] Woodcock N., Hallam N.J, Picton P.D., Hopgood A.A. (1991), " integration of ultrasonic images of weld defects using a hybrid system", Proc. Neuronimes'91, 4th Int. Conf. on Neural Networks and their applications, Nimes, 4-8 Kasım, ss. 593-605.

[6] Fu L.M ve Fu L. C. (1990), "Mapping rule based systems into neural architecture", Knowledge based systems, 3 (3), ss. 48-58.

[7] Sun R. (1994), "Integrating rules and connectionism for robust commonsense reasoning, John Wiley and Sons, New York.

[8] Sun R. (1992), "Connectionist models of rule based reasoning", AISB Quarterly, Vol 79, 1992, ss. 21-24.

BÖLÜM 10

YAPAY SİNİR AĞLARI DONANIMI

Günümüzde yapay sinir ağları uygulamalarının çoğu yazılım teknolojisi olarak görülmektedir. Belirli bir modelin yazılımı gerçeklenmekte ve seri bilgisayarlarda çalıştırılarak sorunların çözülmesi istenmektedir. Yapay sinir ağlarının paralellik gibi bazı özelliklerinin gösterilebilmesi için özel donanım teknolojisine ihtiyaç vardır. Yapay sinir donanımları ticari olarak genellikle şu alanlarda kendini göstermektedir:

- Optik karakter tanıma
- Ses tanıma
- Trafik izleme
- Veri madenciliği ve filtreleme

Yapay sinir ağları için özel donanım geliştirilmesinin bir çok yararı vardır. Bunlar arasında şunları saymak mümkündür.

- **Hız:** Bir çok uygulamada sistemin karar verme hızı çok önemlidir. Günümüzdeki en hızlı seri işlemciler bile özellikle çok sayıda girdi parametrenin gerektiği ve ağın boyutlarının çok büyük olması durumunda gerçek zamanlı kullanıma ve öğrenmeye uygun olmamaktadırlar. Sistemin gerçek zamanlı kullanılması için özel bir donanım ile sistemin hızının artırılması istenebilir. Örneğin birbirine paralel bağlanan işlemciler yardımıyla uygulamanın hızı artırılabilir.
- **Güvenirlilik:** Özel donanım sayesinde sistemin güvenirliliği artırılabilir. Özellikle donanım hatalarının kontrol altına alınması sağlanacağından sisteme olan inanç artacaktır.
- **Özel çalıştırma koşulları:** Donanımın probleme uygun bir şekilde tasarlanması özellikle sistemin boyutları, ağırlığı gibi konularda en uygun durumun seçilmesini sağlar.
- **Güvenlik:** Özel donanım ile sistemin güvenliği e koruması da daha rahat kontrol altına alınmaktadır.

Yapay sinir ağlarının verimli bir şekilde kullanılması için geliştirilecek olan özel donanımların geliştirilmesi çok kolay olmamaktadır. Özellikle resimlerin metin haline dönüştürülmesi ve girdi olarak sisteme tanıtılması zordur. Ayrıca donanımı geliştirmek için gereken zaman da oldukça uzundur. O nedenle, bir donanım geliştirilmeden veya satın alınmadan düşünülmesi gereken sistemin donanım maliyeti, bu donanım üzerinde çalışacak olan yazılımların maliyeti ve geliştirilme zamanlarıdır. Satın alınacak veya geliştirilecek sistemin bu çatıdan katlanılabilir olması ve neticesinde ondan elde edilecek olan yarara değmesi gerekir.

Değişik şekillerde yapay sinir ağı donanımlarının geliştirildiğini görmek mümkündür. Hangi donanımın daha iyi olacağı ağın öğreneceği probleme göre değişmektedir. Yapay sinir ağlarını geliştirmeye yönelik çalışmaların merkezinde VLSI yongaları (*chips*) bulunmaktadır. Yapay sinir ağları donanımlarının iki yönlü geliştirildiği görülmektedir:

- ***Genel amaçlı donanımlar***: Bunlar nöröbilgisayarlar gibi her problem için uygulanabilir donanımlardır.
- ***Özel amaçlı donanımlar:*** Bunlar ise belirli bir spesifik problemi çözmek için geliştirilmiş ve belirli bir amaç için kullanılan donanımlardır.

Piyasada bu iki yaklaşıma dayanarak geliştirilmiş ve kullanılmış donanım çeşitlerine örnekleri şu şekilde sıralamak mümkündür.

- **Nörobilgisayarlar:** Bu bilgisayarlar tamamen yapay sinir ağı teknolojisine dayanarak geliştirilmiştir. Genellikle çok hız gerektiren ve büyük boyutlardaki problemlerin çözülmesi için geliştirilmişlerdir. *Siemens* tarafından geliştirilmiş *Synapse* 1 nöröbilgisayarı bir ana bilgisayara ethernet kartı ile bağlanıp çalışabilen bir bilgisayardır. 8 tane MA-16 *sistolik* dizi yongasını kullanmak. Bu bilgisayarın performansı 25 MHz ile saniyede 3.2 milyar çarpım (16 bit * 16 bit) ve toplama işlemi yapabilecek bir güce sahiptir.
- **PC hızlandırıcılar ve kartları:** Bunlar bilinen bilgisayarlara takılabilen kartlardır. Bunlar sistemin hızını artırmaklar beraber daha çok küçük boyutlardaki problemlerin çözülmesinde tavsiye edilmektedirler. Nörobilgisayarlardan daha ucuzdurlar. IBM'in geliştirdiği ZISC PCI kartları saniyede her birisi 8 bit elemanda oluşan 64 elemandan oluşan 165000 örüntüyü işleme yetenekleri vardır.
- **Yongalar (*Chips*)**: Bunlar yukarıdaki iki tür donanımı oluşturmak için kullanılan sistemler olabilecekleri gibi probleme özel makinelerin (Örneğin makinelerin yanında kullanılacak aparatların) oluşturulmasında kullanılırlar.
- **Gömülü mikroişlemciler**: Bunlar belirli bir yapay sinir ağı uygulamasını koşan fakat klavye, ekran, disk gibi aksamları olmayan bilgisayarlar olarak görülebilir.

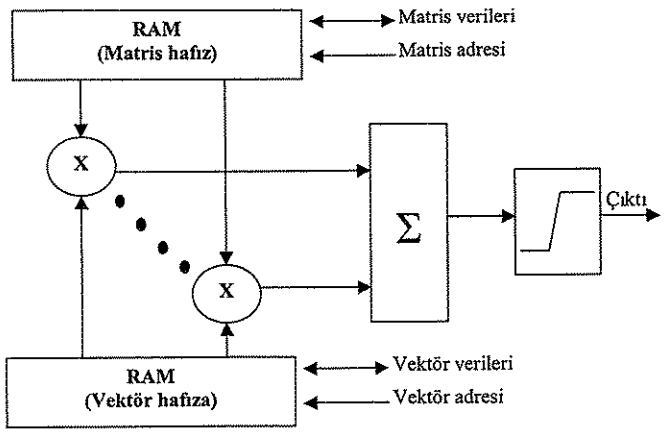
Yukarıdaki donanımların uygulamalarında ise 3 grupta toplamak mümkündür. Her kategoride de öğrenme ve tanıma fonksiyonların gerçekleştirmek için değişik donanım yapıları görülmektedir. Bunlar:

- **Dijital uygulamalar**
- **Analog uygulamalar**
- **Karma sistemler**

10.1. Dijital Yapay Sinir Ağı Donanımları

Dijital yapay sinir ağı donanımlarında, ağ (*network*) üzerindeki bütün hesaplamalarda kullanılan değerler büyüklüğü belirlenmiş ikili (*binary*) vektörlerdir. Dijital teknolojinin verilerin gürültüden ayrılmış olması, ağırlıkları saklamak içim RAM kullanılması, çarpma ve toplama işlemlerinde duyarlılığın (doğruluk derecesinin) artması ve özellikle

de mevcut sistemlere entegrasyonunun sağlanmasındaki kolaylıklar gibi olumlu yanları vardır. Şekil-10.1 bir dijital yapay sinir ağı donanımının yapısı gösterilmektedir.



Şekil-10.1. Dijital yapay sinir ağı yonga yapısı

Dijital yapay sinir ağları, ağa girdi olarak verilen bilgilerdeki gürültüye karşı analog sistemlere göre daha toleranslıdırlar. Hem sabit hem de değişken ağırlık matrislerini saklayabilirler. Programlanabilir elemanları içerebilirler. Bunların yanında bazı olumsuz yanları da vardır. Özellikle, çarpma ve ekleme işlemlerindeki yavaşlık ve dış dünyadan gelen bilgiler genellikle analog bilgilerdir. Onların dijital bilgilere dönüştürülmesi için dönüştürücülere ihtiyaç olması dezavantajlara örnek olarak verilebilir. Uygulanan dijital teknolojiler arasında şunları saymak mümkündür.

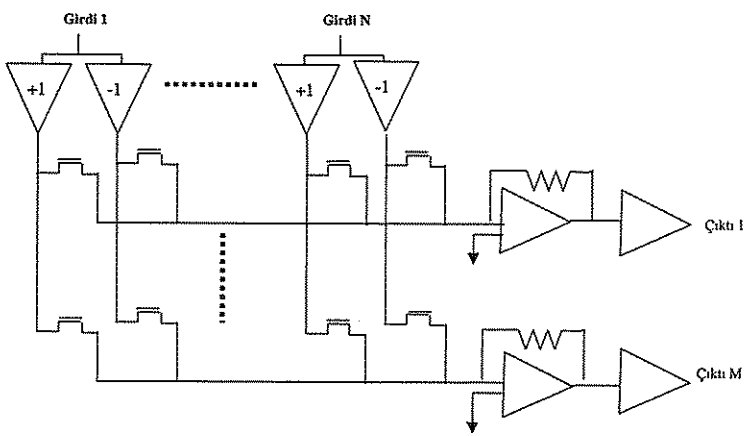
- **Dilim yapıları (*slice architectures*):** Dilim yapısı geleneksel dijital işlemcilerdeki bit dilimlerine benzeyen yapay sinir ağı yapılarıdır. İstenilen boyutta yapay sinir ağı oluşturacak geliştirme blokları sunarlar. Bu konuda *"Philips Lneuro chip"*, *"The Micor devices"* MD120 ve *Neuralogic* NLX-420 işlemcileri örnek olarak gösterilebilir.
- **Çoklu-İşlemciler (*Multi-Processors*)**: Bir yonga üzerine birden fazla işlemcinin konulması sonucu ortaya çıkan işlemcilerdir. Genel olarak bu kapsamda iki grup prosesör görülmektedir. Bunlardan birisi SIMD (*Single Instruction Multiple Data*) olarak bilinen yongalar ve sistolik dizilerdir (*systolic arrays*). SIMD yongalarında, bütün işlemciler aynı komutu farklı veriler kullanarak işletirler. *Sistolik* dizilerde ise her işlemci hesaplamanın bir kısmını gerçekleştirir ve sonuçlarını dizideki bir sonraki işlemciye göndermektedir. SIMD yongalarına *Inova* N64000 (64 işlem elemanı olan), *the* HNC 100NAP (4 proses elemanı) ve *Siemens* MA-16 (hızlı matris operasyonları yapabilen) örnek olarak verilebilir.
- **Radyal temelli fonksiyon (RBF) ağları:** RTF ağları genel olarak verilere dayanarak belirlenmiş etki alanlarını belirleyen vektörleri kullanırlar. Her bir vektör

çok büyük boyutlu (*hyper-dimentional*) bir uzayın bir boyutu olarak düşünülebilir. Geliştirilen donanım bu vektörleri işleyebilecek işlemcilerden oluşmaktadır. Geliştirilmiş RTF donanımlarına örnek olarak, IBM ZISC (*Zero Instruction Set Computer*) yongası ve Nestor Ni1000 yongası sayılabilir.

- Bu sınıflara girmeyen bazı digital yapay sinir ağı yongaları vardır. *Micro Circuit Engineering* şirketinin MT19003 NISP RISC tabanlı bir işlemcisidir. ÇKA için geliştirilmiştir. *Hitachi* tarafından geliştirilmiş *Hopfield* ve ÇKA ağları için geliştirilmiş yongalar da vardır.

10.2. Analog Yapay Sinir Ağı Donanımları

Analog sistemlerin özellikle hızları onları cazip hale getirmektedir. Fakat çarpma ve toplama işlemlerinde duyarlılıkları dijital sistemler kadar iyi değildir. Sıcaklık ve ısı şartlarına, imalat sırasında kullanılan toleranslara bağlı olarak çok küçük sinyalleri sınırlamaktadır. Diğer bir problem ise, ağırlık değerlerinin uzun süreli saklanmasının zorluğudur. Analog ağların tasarlanmasında kullanılan yaklaşım nörömorfik tasarımdır. Bu tasarımda, elektronik devre daha önce anlatılan biyolojik sinir hücreleri ve *sinaps*'ların davranışlarını mümkün olduğunda taklit etmeye çalışır. Şekil-10.2 temel bir analog yapay sinir ağı yonga yapısını göstermektedir:



Şekil-10.2. Temel analog yapay sinir ağı yonga yapısı

Analog yapay sinir ağı donanımları, doğrusal ve doğrusal olmayan aygıtlardan oluşan elektronik devrelerdir. Transistörler, resistörler ve kapasitörler bu aygıtlarda kullanılmaktadır. Bu şekilde geliştirilmiş *Sinaptik Silikon Retina* sistemi resim işlemede biyolojik *Retina*'nın fonksiyonları taklit ederek başarılı sonuçlar göstermiştir. Bu konuda verilebilecek diğer bir örnek ise, *Intel* 8017NW ETANN eğitilebilen bir donanım olup 64 hücre ve 10280 ağırlık değerini saklayabilecek güçteki nöröbilgisayardır.

10.2.1. Analog Sistemlerin Avantajları ve Dezavantajları

Analog yapay sinir ağları donanımları incelendiğinde şu yararlarının olduğu görülmektedir:

- Basit bir transistor yapay sinir ağlarının temel fonksiyonlarından birisi olan çarpma işlemini kolayca gerçekleştirebilir. Aynı işi dijital olarak yapmak için birçok transistöre ihtiyaç vardır.
- Yapay sinir ağlarının diğer bir özelliği ise toplamadır. Basit bir kablo ile birçok çarpandan gelen akımları toplayabilir. Bunu dijital olarak yapmak içinde bir çok transistöre ihtiyaç vardır.
- Kapasitörleri doldurmak için herhangi bir güç israfı yapılmamaktadır.
- Yapay sinir ağı yongası ağırlık matrisinin saklanmasını gerektirir. Eğer yonga adaptif bir algoritmayı çalıştıracak ise o zaman ağırlık değerlerinin elektriksel olarak değişken olmaları gerekir. Analog ağlar ağırlık matrisini saklamak ve dinamik olarak kullanmak bakımından herhangi bir güçlük göstermezler. Bu sistemler hem sabit hem de değişken değerleri saklayabilirler.
- Analog sistemleri hazır sistemleri kullanarak tasarlamak kolaydır.
- Analog sensörlerin çoğu analog çıktılar üretirler. O nedenler analogtan dijitala dönüştürücülerin kullanılmasına gerek yoktur.

Analog yapay sinir ağları üzerinde yapılan araştırmalar şu dezavantajların olduğunu göstermektedir.

- Analog yapay sinir ağlarının ısı karşısında duyarlıdır. Isı değişiklikleri yonga fonksiyonlarını etkileyebilir.
- Gürültü analog sistemlerin diğer bir problemidir. Yapay sinir ağları girdi voltajlarına duyarlı olduklarından bu voltajdaki en küçük bir değişiklik ve verilerdeki çok küçük bir gürültü yonga davranışlarını etkiler.
- Paralel Analog yapay sinir ağlarını test etmek kolay değildir.
- Resistör ve kapasitörleri geliştirmek kolay değildir. 256 x 256 girdi ve çıktısı olan bir ağ için 512 uç (pin) gerektirmektedir. Bu ise oldukça pahalıdır.
- Önceden belirlenmiş ağırlık matrisleri ile seri üretim yapmak çok zordur.

10.3. Karma Tasarımlar

Bu donanım sistemleri hem dijital hem de analog sistemlerin birleştirilmesi ile elde edilen donanımlardır. Bu sistemlerin amacı her iki yaklaşımında avantajlarını bir araya getirmektedir. Genel olarak dış dünya ile ilişki dijital sinyaller ile gerçekleştirilmektedir. Dahili prosesler ile ya tamamen ya da kısmen analog devreler ile sağlanmaktadır. Karma sistemlere örnek, *Bellcore* CLNN-32 ve AT&T ANNA yongalarıdır.

10.4. Yapay Sinir Ağı Donanımlarının Performanslarının Ölçülmesi ve Karşılaştırılmaları

Yapay sinir ağlarının donanımlarını karşılaştırmak ve performanslarını değerlendirmek için şu ana kadar geliştirilmiş bir kıyaslayıcı henüz geliştirilmemiştir. Çünkü ağın sahip olduğu topoloji, modelin yapısı ve öğrenme kuralı donanımın da performansını etkilemektedir. Buna rağmen, yapay sinir ağlarına dayalı donanımların performanslarını ölçmek için kullanılan en yaygın ölçüt saniyede gerçekleştirilen işlem (*connection per sec*) sayısıdır. Bu ağa bilgiler sunulduktan sonra saniyede gerçekleştirilen işlem sayısını göstermektedir. Bir problemi öğrenme zamanı da yine ölçülebilecek ve performansı gösteren bir faktör olarak görülebilir.

Yapay sinir ağı donanımlarını birbirleri ile karşılaştırmak için öncelikle üretilen ürünlerin fabrikasyon özelliklerine bakmak gerekir. Fakat bu verileri bulmak kolay değildir. O nedenle, burada donanımları birbirleri ile karşılaştırmaya neden olacak faktörler daha çok ürünlerin bilinen özellikleri açısından olacaktır. Bunlar:

- ***Teknoloji***: Analog, dijital, karma
- ***Girdiler:*** giriş sayısı, girdi değerlerinin duyarlılığı
- ***Çıktılar:*** Çıktı eleman sayısı, çıktı değerlerinin duyarlılığı
- ***Matris hafıza:*** kapasite, organizasyon, teknoloji
- ***Vektör hafıza:*** kapasite, organizasyon, teknoloji
- ***Fonksiyon işleme***: matematik işlemlerin sabit veya programlanabilir şekilde gerçekleştirilmesi
- ***Saniyede bağlantı***: Her saniyede gerçekleştirilen işlem sayısı
- ***Run-time adresleme***: matris ve vektör hafızalarının yonga çalışırken nasıl adreslendiği
- ***Transfer fonksiyonu***: Hesaplamaların nasıl güncellendiği.

10.5. Özet

Yapay sinir ağlarında yapılan çalışmalar ve bu kitapta anlatılan modeller genel olarak yazılım olarak geliştirilmekte ve seri makinelerde çalıştırılmaktadırlar. Bu ağların başarı bir şekilde uygulanmaları ve paralel çalışabilme özelliklerini göstermek için özel donanımlara ihtiyaç vardır. Bu konuda oldukça yoğun çalışmalar yapılmakta ve ticari sistemler oluşturulmaktadır. Bu donanımlar incelendiğinde genel olarak 3 tür donanım oluştuğu görülmektedir. Bunlar:

- Dijital yapay sinir ağı donanımları
- Analog yapay sinir ağı donanımları
- Karma donanımlar

Bunların her birisinin kendine göre avantaj ve dezavantajları vardır. Tasarımcılar, zaman, maliyet, ve problemin niteliğine bakarak kendileri için en uygun donanımı seçebilirler.

BÖLÜM 11

YAPAY SİNİR AĞLARI UYGULAMALARINA GENEL BİR BAKIŞ

Daha önceki bölümlerde yapay sinir ağları ve değişik modeller tanıtılmıştır. Değişik endüstriyel uygulamalar ayrıntılı olarak açıklanmıştır. Herhangi bir uygulama için ağ seçimi anlatılmıştır. Ağların avantaj ve dezavantajları tekrar gözden geçirilmiştir.

11.1. Yapay Sinir Ağları Uygulama Alanları

Yapay sinir ağlarının uygulamaları gözden geçirildiğinde binlerce uygulamanın yapıldığı ve başarılı sonuçların elde edildiği görülebilir. Uygulamalar o kadar yaygındır ki, bunların listesini çıkartmak hemen hemen mümkün değildir. Örneğin, 1997 yılında pazarlanan, *Caere Inc.* tarafından üretilen Optik karakter okuma sisteminin yılda 3 milyon $'dan fazla para kazandırmaktadır. Benzer şekilde aynı yıl, HNC firması tarafından pazarlanan ve kredi kartlarının haksız yere kullanımlarını ortaya çıkartan **Falcon** isimli yapay sinir ağı sistemi yılda 23 Milyon $'dan fazla para kazandırmıştır. *Wall Street Journal* 1998 yılında *Sensöry Inc.* Firması tarafından geliştirilen ses tanıma sistemindeki yonganın (*chip*) 5$'a mal olduğunu ve bir milyondan fazla sayıda satıldığını rapor etmiştir.

Çalışmalar eskiden laboratuarlarda yürütülmekte ve veriler benzetim yolu ile elde edilmekte iken artık yapay sinir ağları günlük hayatımızın vazgeçilmez bir parçası haline gelmiştir. Ağların eğitilmesinde gerçek örnekler kullanılmaktadır. Örnek bulmak eskisi kadar zor olmamaktadır. Evimizdeki aletlerden elimizdeki cep telefonlarına kadar bir çok alanda yapay sinir ağlarının uygulamalarını görmek mümkündür. Burada her türlü uygulamadan örnek vermek imkansızdır. O nedenle burada, yapay sinir ağları uygulamalarının bir sınıflandırılması yapılacaktır. Bu kapsamda, yapay sinir ağları uygulamaları:

- Endüstriyel uygulamalar
- Finansal uygulamalar
- Askeri ve savunma uygulamaları
- Sağlık uygulamaları
- Diğer alanlardaki uygulamalar

Şeklinde sınıflandırılarak incelenebilir. Bu alanlardaki uygulamalar incelendiğinde yapay sinir ağlarının genel olarak şu fonksiyonları gerçekleştirmek için uygulandıkları görülmektedir:

- ***Tahmin:*** Bu amaçla kullanılan yapay sinir ağları, ağa sunulan bilgilerden yararlanılarak karşılık gelen çıktı değerini tahmin ederler. Hava tahmini, borsada hisselerin değerlerinin tahmini, döviz kurlarının tahmini gibi örnekler vermek mümkündür.
- ***Sınıflandırma:*** Bu amaçla kullanılan yapay sinir ağları kendilerine sunulan bilgileri kategorize etmek görevini üstlenirler. Bir makine üzerinde görülen hataların sınıflandırılması buna örnek olarak verilebilir.
- ***Veri ilişkilendirme***: Bu amaçla eğitilen ağlar ağa sunulan verilerin hatalı ve eksik olup olmadığını belirlerler. Öğrendikleri bilgiler ile eksik olan bilgileri tamamlarlar. Eksik bir resmin tamamlanması bu konuda örnek olarak verilebilir.
- ***Veri filtreleme***: Bu amaçla eğitilen ağlar, birçok veri arasından uygun verileri belirleme görevini yerine getirirler. Telefon konuşmalarındaki gürültüleri asıl konuşmalardan ayıran ağlar bu konudaki uygulamalara örnek olarak verilebilir.
- ***Tanıma ve eşleştirme***: Değişik şekil ve örüntülerin tanınması, eksik, karmaşık, belirsiz bilgilerin işlenerek eşleştirme ve tanıma fonksiyonları gerçekleştirilebilir. Daha önce örneği verilen kalite kontrol şemaları üzerindeki şekilleri tanıyan ağ, bu konuda örnek olarak verilebilir.
- ***Teşhis***: Bu amaçla geliştirilen ağlar sistemlerin olumsuzluklarının ortaya konulması ve problemlerin teşhis edilmesi işlemini yerine getirirler. Makinelerin, süreçlerin arazi durumlarının ve hatalarının teşhis edilmesi buna örnek olarak verilebilir. Tıp alanında da bu tür sistemler yaygın olarak geliştirilmektedir.
- ***Yorumlama***: Bir olay hakkında toplanan örneklerden elde edilen ve eğitim sonucu oluşturulan bilgileri kullanarak yeni olayların yorumlanması işlemleri bu kapsamda düşünülmektedir. Bir olay hakkında toplanan verilerin yorumlanarak istatistiksel dağılımlarının belirlenmesi bu konuda örnek olarak verilebilir.

Yapay sinir ağlarının yukarıdaki açıklamalar ışığında değişik uygulama alanlarına örnekler kaynaklarda verilmiştir. Bunlardan bazıları aşağıda listelenmiştir.

11.1.1. Endüstriyel Uygulamalar

Yapay sinir ağlarının sayısız endüstriyel uygulaması vardır. Bunlardan bazıları şu şekilde sıralanabilir:

- Yapay sinir ağları bir endüstriyel proseste fırınların ürettiği gaz miktarını tahmini
- İmalatta, ürün tasarımı, proses ve makinelerin bakımı ve hataların teşhisi görsel kalite kontrolü
- Kimyasal proseslerin dinamik modellenmesi
- Otomobillerde otomatik rehber sisteminin geliştirilmesi
- Robotlarda görme sistemleri ve *mainpulatör*lerin kontrol edilmesi
- Cep telefonlarında ses ile çalışabilme
- Araba pistonlarının üretim şartlarının belirlenmesi
- Elektronik yonga hata analizleri
- Optimizasyon çalışmaları (üretim planlama ve kontrol çalışmalarında)
- Müşteri tatmini ve Pazar verilerinin değerlendirilmesi ve analiz edilmesi
- Kömür güç istasyonları için çevrimiçi (*on-line*) karbon akımı ölçülmesi
- İşlerin makinelere atanması ve çizelgeleme
- Gezgin satıcı problemini

Bu listeyi uzatmak mümkündür. Görüldüğü gibi endüstriyel uygulamalar oldukça yaygın bir alanı kapsamaktadır. İmalattan güç istasyonlarına kadar bir çok alanda bu ağlar kullanılmaktadır.

11.1.2. Finansal Uygulamalar

Yapay sinir ağları finans dünyasında da oldukça yaygın olarak kullanılmaktadır. Kullanım alanlarına örnekler şu şekilde listelenebilir.

- Makro ekonomik tahminler
- Borsa benzetim çalışmaları endekslerinin tahmin edilmesi
- Kredi kartı hilelerinin tespiti
- Kredi kartı kurumlarında iflas tahminleri
- Banka kredilerinin değerlendirilmesi
- Emlak kredilerinin yönetilmesi
- Döviz kuru tahminleri
- Risk analizleri

11.1.3. Askeri Uygulamalar

Yapay sinir ağlarının sivil hayattaki uygulamaları kadar askeri alandaki uygulamaları da dikkatleri çekmektedir. Bunlar arasında şunlar örnek olarak sayılabilir:

- Hedef tanıma ve takip sistemleri
- Yeni sensörlerin performans analizleri

- Radar ve görüntü sinyalleri işleme
- Sensör fizyonu
- Askeri uçakların uçuş yörüngelerinin belirlenmesi (optimizasyonu)
- Mayın detektörleri

11.1.4. Sağlık Uygulamaları

Yapay sinir ağlarının insan beyni çalışmaları ile yakın ilişki içinde olması tıp ve sağlık alanında da uygulamaların gelişmesine neden olmuştur. Bunlardan bazı örnekler ise şöyle sıralanabilir:

- Solunum hastalıklarının teşhisi
- EEG ve ECG analizleri
- *Transplant* zamanlarının optimizasyonu
- Hastalıkların teşhisi ve resimlerden tanınması
- *Karidovascular* sistemlerin modellenmesi ve teşhisi
- Tıbbi resim işleme
- CTG izleme
- Hamile kadınların karınlarındaki çocukların kalp atışlarının izlenmesi
- Yumurtalık kanserinin *immunoterapik* izlenmesi
- Üroloji uygulamaları (prostat analizleri, sperm analizleri)

11.1.5. Diğer Alanlar

Yukarıda verilenlere ek olarak daha birçok alanda yapay sinir ağları uygulamalarını görmek mümkündür. Bunlardan bazıları şöyle sıralanabilir:

- Sigorta poliçelerinin değerlendirilmesi
- Uçak parçalarının hata teşhislerinin yapılması
- Petrol ve gaz aramasının yapılması
- Hava alanlarında bomba dedektörleri ve uyuşturucu koklayıcıları
- Rotalama sistemleri (kamyon rotalarının optimizasyonu)
- Resim işleme, segmentasyon, restorasyon
- Karakter, el yazısı ve imza tanıma sistemleri
- Şekil sıkıştırma
- Veri madenciliği
- İnsani davranışlar sergileyen çocuk oyuncaklarının geliştirilmesi
- Kömür ve yemeklerdeki nem oranın tahmin edilmesi
- Büyük inşaat projelerinde (baraj inşaatı gibi) maliyetlerin tahmin edilmesi

Yukarıda açıklanan konularda değişik alanlardaki yapay sinir ağları uygulamalarına değişik örnekler bölümün sonunda verilen referanslarda bulunabilir.

11.2. Herhangi Bir Uygulama İçin Ağ Seçimi

Ağların hangi alanlarda kullanılabileceğinin bilinmesi kadar hangi problem için hangi ağın daha uygun olacağının bilinmesi de önemlidir. Yukarıdaki bazı alanlarda hangi ağların daha başarılı olarak uygulandıkları Tablo-11.1'de verilmiştir:

Tablo-11.1. Ağların başarılı oldukları alanlar

Kullanım Amacı	Ağ Türü	Ağın Kullanımı
Tahmin	• ÇKA	Ağın girdilerinde bir çıktı değerinin tahmin edilmesi
Sınıflandırma	• LVQ • ART • *Counterpropagation* • Olasılık Sinir ağları (PNN)	Girdilerin hangi sınıfa ait olduklarının belirlenmesi
Veri ilişkilendirme	• *Hopfield* • *Boltzmann Machine* • *Bidirectional associative Memory* (BAM)	Girdilerin içindeki hatalı bilgilerin bulunması ve eksik bilgilerin tamamlanması

Görüldüğü gibi her ağın iyi olduğu kullanım alanları vardır. Bu alanları belirleyerek uygulamalar geliştirmek başarılı sonuçları elde etmeye neden olur. Bazı durumlarda yanlış ağ seçimi yüzünden haftalarca ağını eğitemeyen ve yapay sinir ağlarının becerisinin bazı olaylar için yetersiz olduğunu iddia edenler görülmektedir. Bu doğru bir yaklaşım değildir. Doğru ağ, doğru örnek seti ve doğru bir öğrenme algoritmasının çözemeyeceği problem yok denecek kadar azdır.

11.3. Yapay Sinir Ağı Uygulamalarının Avantajları

Yapay sinir ağlarının uygulamaya alınmasının arkasında haklı gerekçeler ve yararlar vardır. Bunları şu şekilde sıralamak mümkündür.

- Yapay sinir ağları matematik olarak modellenmesi mümkün olmayan veya zor olan karmaşık problemleri çok rahat modelleyerek çözebilmektedir.
- Yapay sinir ağlarını kullanarak problemleri başarılı bir şekilde çözebilmek için problemin çok iyi modellenmesi gerekmektedir. Bu modelleme, problemi çözebilmek için sadece söz konusu olay ile ilgili örneklerin belirlenip toparlanmasına yardımcı olacaktır. Örneklerin dışında herhangi bir ön bilgiye ihtiyaç yoktur. Örnek bulmak bilgi bulmaktan çok daha kolaydır. Yeter ki modele uygun örnekler olsun. Bunlar da genellikle bir sorun oluşturmaktadır.
- Gerçek dünyada olaylar ve olayların arkasındaki değişik faktörlerin birbirleri ile ilişkilerini ve birbirinin üzerindeki etkileri gerçek hayatta bilmek zordur. Yapay sinir ağları bu ilişkileri otomatik olarak örneklerden öğrenirler. Kullanıcıların bu ilişkileri bilmesi ve ağa söylemesi beklenmemektedir. Geleneksel

yöntemlerde bu ilişkileri belirlemek veya yok saymak gerekmektedir. Bu özellik belki de yapay sinir ağlarının en önemli avantajlarından birisidir.

- Benzer şekilde, gerçek dünya olayları ve bu olayların arkasındaki faktörlerin birbirleri ile ilişkileri doğrusal olmaz ise bu ilişkileri modellemek çok zordur. O nedenle gerçek hayatta problemleri çözmek için bazı varsayımlar yapmak gerekmektedir. Buda modellenen sistemin gerçek sisteme uygunluğunu azaltmakta ve gerçek sistemin davranışlarını kontrol altına almayı zorlaştırmaktadır. Özellikle insan hayatının söz konusu olduğu yerlerde bu sebepten dolayı geleneksel sistemleri kullanmak sorun olabilmektedir. Yukarıda anlatılanların aksine, yapay sinir ağları için ise ilişkilerin doğrusal olup olmaması önemli değildir. Çünkü ilişkileri gerçekleşen örnekler üzerinde kendiliğinden hesaba katılmaktadır. O nedenle, bu ilişkilerin modellenmesi sorunu geleneksel sistemdeki kadar zor değildir. Örnekler gerçek sistemi temsil etmektedirler. Onun için bu örnekleri kullanarak öğrenen ağlar tarafından verilen kararlar daha gerçekçi olmaktadır.
- Yapay sinir ağları uygulamaları hem pratik hem de maliyet bakımından daha ucuzdurlar. Sadece örneklerin belirlenmesi ve basit bir program problemi çözmek için yeterli olabilmektedir.
- Yapay sinir ağları zaman bakımından da çok verimli çalışırlar. Örneklerin bulunması, ağların oluşturulması, olayı öğrenmesi, gerçek zamanda kullanıma alınması çok kısa bir zaman diliminde mümkün olabilmektedir. Aynı zamanda yapay sinir ağlarının çalışması da geleneksel sistemlerden daha hızlıdır.
- Yapay sinir ağları yeni bilgilerin ortaya çıkması ve ortamda bazı değişikliklerin olması durumunda yeniden eğitebilirler. Bazı ağların eğitilmesine de gerek yoktur. Kendileri ortama uyumu öğrenerek gerçekleştirebilirler.
- Yapay sinir ağlarının paralel çalışabilmeleri onların gerçek zamanlı kullanımlarının kolaylaştırmaktadır.

11.4. Yapay Sinir Ağı Uygulamalarının Dezavantajları

Yapay sinir ağlarının oluşturulmasında ve kullanılmasında avantajlar yanında bazı dezavantajlarda vardır. Bunların bazıları şöyle listelenebilir.

- Yapay sinir ağlarının oluşturulmasında, model seçilmesinde, ağın topolojisinin belirlenmesinde bir kurallar seti yoktur. Kullanıcının tecrübesine dayalı olarak belirlenmektedir.
- Problemlerin yapay sinir ağı ile çözülebilmesi için örneklerin tasarlanması için bir kurallar seti yoktur. Problem sahibi kendi tecrübesine göre örnekleri formulize etmektedir. Daha önce anlatıldığı gibi aynı problem değişik şekillerde gösterilebilmekte ve her gösterimin kendisine göre performansı da değişmektedir. Doğru gösterimi bulmanın yolu yine tecrübeler ile sınırlıdır.
- Ağın davranışlarını açıklaması mümkün değildir. Bu ise ağa olan güveni azaltmakta ve özellikle insan hayatı ile ilgili olan problemlerde sonuçların neden verilmediğinin açıklanamaması kullanım alanlarını sınırlandırmaktadır.
- Eğitimin gerçekleştirilmesi uzun zamanlar alabilmektedir.

- Problemlere optimum sonuçlar garanti etmez. Üretilen sonuçların optimum olduğunu iddia etmek doğru değildir. İyi sonuçlardan birisidir denilebilir. Geleneksel yöntemler optimum çözümler üretirler.
- Örneklerin bulunmasının güç olduğu durumlarda ve problemi doğru temsil eden örneklerin bulunmaması durumunda problemlere sağlıklı çözümler üretebilmek mümkün olamamaktadır.

11.5. Yapay Sinir Ağı Simulatörleri

Yapay sinir ağları ile ilgili değişik modeller ve endüstriyel uygulamalar verildi. Yapay sinir ağlarının bilgisayar programları yazılmış ve birçok modeli çözebilen hazır sistemler geliştirilmiştir. Yapay sinir ağı simulatörü denilebilen bu sistemler kullanıcıdan sadece örnek seti, test setini hazırlamasını ve ağın modelini belirleyerek ilgili modeller için gereken parametre ve sabit değerlerin (öğrenme katsayısı gibi) belirlemesini istemektedir. Böylece sistem problemi çözecek ağı otomatik olarak oluşturmakta ve ağı eğitebilmektedir. Bu sistemler aynı zamanda ağın hatasının da bir grafiğini çizerek eğitimin zaman içindeki iyileşmesini göstermektedir.

Pazarda sayısız yapay sinir ağı simulatörü bulmak mümkündür. Bunların hepsini burada ele almak mümkün değildir. Burada örnek olması bakımından bir kaç tanesi ve ulaşım adresleri verilecektir (bkz. Tablo-9.2). Sistemler hakkında ilgili web sayfalarından ayrıntılı bilgiler almak mümkün olabilir. Bazı sitelerde ise birden fazla simulatör tanıtılmaktadır. Örneğin,

http://www.cs.cofc.edu/~manaris/ai-education-repository/neural-n-tools.html

adresinde birkaç simulatör hakkında bilgi bulunabilir...

Tablo-9.2. Yapay sinir ağları simulatörleri ve adresleri örnekleri

Simulatör Adı	Web Adresi
neurosolutions	http://www.nd.com
Matlap-Neural Network Tool *box*	http://www.mathworks.com
NeurDS	ftp://gatekeeper.dec.com/pub/DEC
Mactivation	ftp://ftp.cs.colorado.edu/pub/cs/misc/.
Xerion	ftp://ftp.cs.toronto.edu/pub/xerion/
Brain Wave	http://www2.psy.uq.edu.au/~brainwav/Manual/WhatIs.html
PDP++	http://www.cnbc.cmu.edu/PDP++/manual/pdp-user_20.html
Z-Solutions	http://www.zsolutions.com/software.htm
STATSOFT	www.statsoftinc.com

11.6. Yapay Sinir Ağları Bilgi Kaynakları

Yapay sinir ağları ile ilgili olarak her yıl çok sayıda sempozyumlar düzenlenmekte kitap ve makaleler yazılmaktadır. Bu konuya ola ilgi her geçen gün daha çok artmaktadır. Yapay sinir ağı hakkında bilgi almak isteyen ve en çok sorulan soruların ve cevaplarının tutulduğu bir adres vardır. Orada sadece sorulara cevap verilmemekte aynı zamanda yapılan sempozyumlar, yayınlanan kitaplar, geliştirilen bilgisayar programları vb. gibi konularda da sürekli bilgiler verilmekte ve site güncellemektedir. Bu bilgilere şu adresten ulaşmak mümkündür.

ftp://ftp.sas.com.pub/neural/FAQ.html

Ayrıca IEEE Yapay sinir ağları *Konsilinin* web adresinden de (http://www.ieee.org/nnc/index.html) bilgiler almak mümkündür.

Bunların dışında web adreslerinin sayısı o kadar çoktur ki burada listelemenin bir anlamı yoktur. Bu iki adresten istenilen bilgileri içeren değişik web adreslerine ulaşmak mümkündür.

11.7. Özet

Bu bölümde yapay sinir ağlarının uygulamalarına genel bir bakış yapılmıştır. Genel olarak uygulamaların şu alanlarda görüldüğü belirtilmiştir.

- Endüstriyel uygulamalar
- Finansal uygulamalar
- Askeri ve savunma uygulamaları
- Sağlık uygulamaları

Bu alanlarda yapay sinir ağının bu alanlarda teşhis, sınıflandırma, tahmin, kontrol, veri ilişkilendirme, veri filtreleme, yorumlama gibi alanlarda kullanılmaktadır. Hangi problem için hangi ağın daha uygun olduğunu belirlemek için ağların özellikleri ile problemlerin özelliklerini karşılaştırmak gerekir. Her ağın çok iyi olduğu bazı özellikleri vardır. Örneğin ÇKA ağlarının tahmin problemlerinde çok iyi sonuçlar ürettiği bilinmektedir. Bu özelliklere göre problemler için uygun ağlar belirlenebilir.

Yapay sinir ağlarının bazı avantaj ve dezavantajları vardır. Bölüm içinde bunlar listelenmiştir. Yapay sinir ağlarının avantajlarından yararlanmak ve dezavantajlarını önleyecek şekilde sistemler oluşturmak için çalışmalar yapmak gerekmektedir.

Yapay sinir ağlarının geliştirilmesi diğer sistemlere göre oldukça kolaydır. Piyasada ticari olarak satılan simulatörler vardır. Bu sistemlere sadece örnekler verilmekte ve ağın topolojisi belirlenmektedir. Öğrenme işimi simulatör kendisi yapmaktadır. Bu simulatörlerin test etme yetenekleri de vardır. Öğrenen ağın performansını da belirleyebilmektedir.

Yapay sinir ağları ile ilgili olarak bir çok kaynaktan ayrıntılı bilgiler almak mümkündür. Aşağıda yapay sinir ağlarının uygulamaları başta olmak üzere bu konuda okuyucuya yardımcı olacak kaynaklardan bazıları listelenmiştir.

Kaynakça

[1] Hopfield, J. J. and Tank, D. W. 1985, "Neural computation of decisions in optimization problems," *Biological Cybernetics* , Vol. 52, pp. 141-152.

[2] Sharda, R. 1994, "Neural networks for the MS/OR analyst: An application bibliography," *Interfaces*, Vol. 24, No. 2, pp. 116-130.

[3] Wilson, R.L. and Sharda, R. 1992, "Neural networks," *OR/MS Today*, Vol. 19, No. 4, pp. 36-42.

[4] Klimasauskas, C. C. 1989, *The 1989 Neuro-Computing Bibliography* , The MIT Press, Cambridge, MA.

[5] Burke, L. L. and Ignizio, J. P. 1992, "Neural networks and operations research: An overview," *Computers and Operations Research*, Vol. 19, No. 3/4, pp. 179-189.

[6] Widrow, B.; Rumelhart, D.E.; and Lehr, M.A. 1994, "Neural networks: Applications in industry, business, and science," *Communications of the ACM*, Vol. 37, No. 3, pp. 93-105.

[7] Ripley, B.D. 1993, "Statistical Aspects of Neural Networks," Proceedings of Invited Lectures for SemStat, Sandbjerg, Denmark.

[8] Kuan, C.M. and White, H. 1994, "Artificial neural networks: An econometric perspective," *Econometric Reviews*, Vol. 13, No. 1, pp. 1-91.

[9] Wilson, R.L. and Sharda, R. 1994, "Bankruptcy prediction using neural networks," *Decision Support Systems*, Vol. 11, No. 5, pp. 545-557.

[10] Fletcher, D. and Goss, E. 1993, "Forecasting with neural networks: An application using bankruptcy data," *Information and Management*, Vol. 24, No. 3, pp. 159-167.

[11] Coats, P.K. and Fant, L.F. 1991, "A neural network approach to forecasting financial distress," *Journal of Business Forecasting*, Vol. 10, No. 4, pp. 9-12.

[12] Coats, P.K. and Fant, L.F. 1993, "Recognizing financial distress patterns using a neural network tool," *Financial Management*, Vol.22,No.3, pp.142-155.

[13] Collins, A. and Evans, A. 1994, "Aircraft noise and residential property values," *Journal of Transport Economics and Policy*,Vol.28, No. 2, pp. 175-197.

[14] Altman, E.I.; Marco, G.; and Varetto, F. 1994, "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience)," *J. of Banking and Finance*, Vol. 18, No. 3, pp. 505-529.

[15] Salchenberger, L. M.; Cinar, E. M.; and Lash, N. A. 1992, "Neural networks: A new tool for predicting thrift failures," *Decision Sciences*, Vol. 23, No. 4, (July/Aug.), pp. 899-916. Also reprinted in *Neural Networks in Finance and Investing* , ed. R. Trippi and E. Turban, Probus Publ. Co., Chicago, IL, 1993.

[16] Tam, K. Y. and Kiang, M. Y. 1992, "Managerial applications of neural networks: The case of bank failure predictions," *Management Science*, Vol. 38, No. 7, (July), pp. 926-947. Also reprinted in *Neural Networks in Finance and*

Investing , ed. R. Trippi and E. Turban, Probus Publishing Company, Chicago, IL, 1993.

[17] Collins, J.M. and Clark, M.R. 1993, "An application of the theory of neural computation to the prediction of workplace behavior: An illustration and assessment of network analysis," *Personnel Psychology*, Vol. 46, No. 3, pp. 503-524.

[18] Gorr, W.L.; Nagin, D.; and Szczypula, J. 1994, "Comparative study of artificial neural network and statistical models for predicting student grade point averages," *International Journal of Forecasting*, Vol. 10, No. 1, pp. 17-34.

[19] Hardgrave, B.C.; Wilson, R.L.; and Walstrom, K.A. 1994, "Predicting graduate student success: A comparison of neural networks and traditional techniques," *Computers and Operations Research* , Vol. 21, No. 3, pp. 249-263.

[20] Tsukuda, J. and Baba, S. 1994, "Predicting Japanese corporate bankruptcy in terms of financial data using neural network," *Computers and Industrial Engineering*, Vol. 27, No. 1-4, pp. 445-448.

[21] Udo, G. 1993, "Neural network performance on the bankruptcy classification problem," *Computers and Industrial Engineering* ,Vol.25,No.1-4,pp.377-380.

[22] Piramuthu, S.; Shaw, M.J.; and Gentry, J.A. 1994, "A classification approach using multi-layered neural networks," *Decision Support Systems*, Vol. 11, No. 5, pp. 509-525.

[23] Trippi, R.R. and DeSieno, D. 1992, "Trading equity index futures with a neural network," *Journal of Portfolio Management* , Vol. 19, No. 1, pp. 27-33.

[24] Yoon, Y.; Swales, G.; and Margavio, T.M. 1993, "A comparison of discriminant analysis versus artificial neural networks," *Journal of the Operations Research Society*, Vol. 44, No. 1, pp. 51-60.

[25] Kattan, M.W.; Adams, D.A.; and Parks, M.S. 1993, "A comparison of machine learning with human judgement," *Journal of Management Information Systems*, Vol. 9, No. 4, pp. 37-57.

[26] Proctor, R.A. 1991, "An expert system to aid in staff selection: A neural network approach," *International Journal of Manpower*,Vol.12,No.8, pp.18-21.

[27] Kryzanowski. L.; Galler, M.; and Wright, D.W. 1993, "Using artificial neural networks to pick stocks," *Financial Analysts Journal*,Vol.49,No.4, pp.21-27.

[28] Collins, A. and Evans, A. 1994, "Aircraft noise and residential property values," *Journal of Transport Economics and Policy*, Vol.28, No.2, pp.175-197.

[29] A Quang Do and Grudnitski, G. 1992, "A neural network approach to residential property appraisal," *Real Estate Appraiser*, Vol. 58, No. 3, pp. 38-45.

[30] Balakrishnan, P. V.; Cooper, M.; Jacob, V. S.; and Lewis, P. A. 1996, "Comparative performance of the FSCL neural net and k-means algorithm for market segmentation," *European Journal of Operational Research*, In Press.

[31] Curry, B. and Moutinho, L. 1993, "Neural networks in marketing: Modelling consumer responses to advertising stimuli," *European Journal of Marketing*, Vol. 27, No. 7, pp. 5-20.

[32] Dasgupta, C.G.; Dispensa, G.S.; and Ghose, S. 1994, "Comparing the predictive performance of a neural network model with some traditional market response models," *International Journal of Forecasting*, Vol. 10, No. 2, pp. 235-244.

[33] Proctor, R.A. 1992, "Marketing decision support systems: A role for neural networks," *Marketing Intelligence and Planning*, Vol. 10, No. 1, pp. 21-26.

[34] McKim, R.A. 1993, "Neural network applications for project management: Three case studies," *Project Management Journal* , Vol. 24, No. 4, pp. 28-33.

[35] Hansen, J. V.; McDonald, J.B.; and Stice, J.D. 1992, "Artificial intelligence and generalized qualitative-response models: An empirical test on two audit decision-making domains," *Decision Sciences* , Vol. 23, No. 3, pp. 708-723.

[36] Hill, T.; Marquez, L.; O'Connor, M.; and Remus, W. 1994, "Artificial neural network models for forecasting and decision making," *International Journal of Forecasting*, Vol. 10, No. 1, pp. 5-15.

[37] Wilson, R.L. 1994, "A neural network approach to decision alternative prioritization," *Decision Support Systems* , Vol. 11, No. 5, pp. 431-447.

[38] Kuan, C.M. and Liu, T. 1992, "Forecasting exchange rates using feedforward and recurrent neural networks," Faculty Working Paper No. 92-0128, University of Illinois at Urbana-Champaign.

[39] Bansal, A.; Kauffmann, R.J.; and Weitz, R.R. 1993, "Comparing the modeling performance of regression and neural networks as data quality varies: A business value approach," *Journal of Management Information Systems*, Vol. 10, No. 1, pp. 11-32.

[40] Archer, N.P. and Wang, S. 1993, "Application of the backpropagation neural network algorithm with monotonicity constraints for two-group classification problems," *Decision Sciences* , Vol. 24, No. 1, pp. 60-75.

[41] Curram, S.P. and Mingers, J. 1994, "Neural networks, decision tree induction and discriminant analysis: an empirical comparison," *Journal of the Operations Research Society* , Vol. 45, No. 4, pp. 440-450.

[42] Liang, T.P.; Chandler, J.S.; Han, I.; and Roan, J. 1992, "An empirical investigation of some data effects on the classification accuracy of probit, ID3, and neural networks," *Contemporary Accounting Research*, Vol. 9, No. 1, pp. 306-328.

[43] Patuwo, E.; Hu, M.H.; and Hung, M.S. 1993, "Two-group classification using neural networks," *Decision Sciences* , Vol. 24, No. 4, pp. 825-845.

[44] Subramanian, V.; Hung, M.S., and Hu, M.Y. 1993, "An experimental evaluation of neural networks for classification," *Computers and Operations Research*, Vol. 20, No. 7, pp. 769-782.

[45] Wang, S. 1995, "The unpredictability of standard back propagation neural networks in classification applications," *Management Science*, Vol. 41, No. 3, pp. 555-559.

[46] Hurrion, R.D. 1993, "Representing and learning distributions with the aid of a neural network," *Journal of the Operational Research Society*, Vol. 44, No. 10, pp. 1013-1023.

[47] Mangiameli, P.; Chen, S.K.; and West, D. 1996, "A comparison of SOM neural network and hierarchical clustering methods," *European Journal of Operational Research*, In Press.

[48] Palocsay, S.W.; Stevens, S.P.; Brookshire, R.G.; Sacco, W.J.; Copes, W.S.; Buckman, R.F.; and Smith, J.S. 1996, "Using neural networks for trauma outcome evaluation," *European Journal of Operational Research*, In Press.

[49] Wang, S. 1994, "Neural Networks for monotonic nonlinear models," *Computers and Operations Research*, Vol. 21, No. 2, pp. 143-154.

[50] Wu, F.Y.; and Yen, K.K. 1992, "Applications of neural network in regression analysis," *Computers and Industrial Engineering*, Vol. 23, No. 1-4, pp. 93-95.

[51] Caporaletti, L.E.; Dorsey, L.E.; Johnson, J.D.; and Powell, W.A. 1994, "A decision support system for in-sample simultaneous equation systems forecasting using artificial neural systems," *Decision Support Systems*, Vol. 11, No. 4, pp. 482-495.

[52] Chu, C. and Widjaja, D. 1994, "Neural network system for forecasting method selection," *Decision Support Systems* , Vol. 12, No. 1, pp. 13-24.

[53] Damodran, P.S.; Kolli, S.S.; and Alexander, S.M. 1993, "The investigation of new approaches to business data analysis," *Computers and Industrial Engineering*, Vol. 25, No. 1-4, pp. 545-548.

[54] Lee, J.K. and Jhee, W.C. 1994, "A two-stage neural network approach for ARMA model identification with ESCAF," *Decision Support Systems*, Vol. 11, No. 4, pp. 461-479.

[55] Lee, T.H.; White, H.; and Granger, C.W.J. 1993, "Testing for neglected nonlinearity in time series models," *Journal of Econometrics*, Vol. 56, pp. 269-290.

[56] Anderson, J.A.; Pellionisz, A.; and Rosenfeld, E. (Eds) 1990, *Neurocomputing 2: Directions for research*, The MIT Press.

[57] Tank, D. W. and Hopfield, J. J. 1986, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and linear programming circuit," *IEEE Transaction on Circuits and Systems*, Vol. CAS33, No. 5 (May), pp. 533-541.

[58] Dagli, C. H.; Kumara, S. R. T.; and Shin, Y. C. 1991, *Intelligent Engineering Systems Through Artificial Neural Networks*, ASME Press, Fairfield, NJ.

[59] Ramanujam, J. and Sadayappan, P. 1988, "Optimization by neural networks," in *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, Vol. 2, pp. 325332.

[60] Looi, C. 1992, "Neural network methods in combinatorial optimization," *Computers and Operations Research*, Vol. 19, No. 3/4, pp. 191-208.

[61] Wang, J. and Chankong, V. 1991, "Neurally inspired stochastic algorithm for determining multi stage multi attribute sampling inspection plans," *Journal of Intelligent Manufacturing* , Vol. 2, No. 5 (October), pp. 327336.

[62] de Ca.valho, Luis A. V. and Barbosa, V. C. 1989, "Towards a stochastic neural model for combinatorial optimization," in *Proceedings of the IEEE International Joint Conf. on Neural Networks*, Washington D.C., Vol.2, pp.587.

[63] Chakrapani, J. and Skorin-Kapov, J. 1992, "A connectionist approach to the quadratic assignment problem," *Computers and Operations Research*, Vol. 19, No. 3/4, pp. 287-295.

[64] DeSilets, L.; Golden, B.; Kumar, R.; and Wang, Q. 1992, "A neural network model for cell suppression of tabular data," Working paper, College of Business and Management, University of Maryland, College Park, MD.

[65] Glover, F. 1994, "Optimization by ghost image processes in neural networks," *Computers and Operations Research* , Vol. 21, No. 8, pp. 801-822.

[66] Jagota, A. 1996, "An adaptive, multiple restarts neural network algorithm for graph coloring," *European Journal of Operational Research*, In Press.

[67] Lee, B. W. and Sheu, B. J. 1990, "Combinatorial optimization using competitive Hopfield neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Washington D.C., Vol. 2, pp. 627630.

[68] Satake, T.; Morikawa, K.; and Nakamura, N. 1994, "Neural network approach for minimizing the makespan of the general job-shop," *International Journal of Production Economics*, Vol. 33, No. 1-3, pp. 67-74.

[69] Wacholder, E. 1989, "A neural network based optimization algorithm for the static weapon target assignment problem," *ORSA Journal on Computing*, Vol. 1, No. 4 (Fall), pp. 232246.

[70] Angeniol, B.; Croix Vaubois, G.; and Le Texier, J. 1988, "Self organizing feature maps and the travelling salesman problem," *Neural Networks*, Vol. 1, No. 4, pp. 289-293.

[71] Bigus, J. P. and Goolsbey, K. 1990, "Integrating neural networks and knowledge based systems in a commercial environment," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Washington D.C., Vol. 2, pp. 463466.

[72] Burke, L. L. and Damany, P. 1992, "The guilty net for the traveling salesman problem," *Computers and Operations Research*, Vol.19, No.3/4, pp.255-266.

[73] Dagli, C. H.; Ashouri, M. R.; Leininger, G.; and McMillin, B. 1990, "Composite stock cutting pattern classification through neocognition," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Washington D.C., Vol. 2, pp. 587590.

[74] Hegde, S. U.; Sweet, J. L.; and Levy, W. B. 1988, "Determination of parameters in a Hopfield/Tank computational network," in *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, Vol. 2, pp. 291298.

[75] Jeffries, C.; and Niznik, T. 1994, "Easing the conscience of the guilty net," *Computers and Operations Research* , Vol. 21, No. 9, pp. 961-968.

[76] Xu, X. and Tsai, W. T. 1990, "An adaptive neural network algorithm for the traveling salesman problem," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Washington D.C., Vol. 2, pp. 716719.

[77] Zhang, L. and Thomopoulos, S. C. A. 1989, "Neural network implementation of shortest path algorithm for traffic routing in communication networks," in

Proceedings of the IEEE International Joint Conference on Neural Networks, Washington D.C., Vol. 2, pp. 591.

[78] Cichocki, A.; Unbehauen, R.; Weinzeirl, K.; and Holzel, R. 1996, "A new neural network for solving linear programming models," *European Journal of Operational Research*, In Press.

[79] Culioli, J. and Protopopecu, V. 1990, "Neural network models for linear programming," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* Washington D.C., Vol. 1, pp. 293296.

[80] Wang, J. and Chankong, V. 1992, "Recurrent neural networks for linear programming: analysis and design principles," *Computers and Operations Research*, Vol. 19, No. 3/4, pp. 297-311.

[81] Lin, Y; Austin, L.M.; and Burns, J.R. 1992, "An intelligent algorithm for Mixed-Integer programming models," *Computers and Operations Research*, Vol. 19, No. 6, pp. 461-468.

[82] Kennedy, M. P. and Chua, L. O. 1988, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, Vol. 35, No. 5 (May), pp. 554562.

[83] Reklaitis, G. V.; Tsirukis, A. G.; and Tenorio, M. F. 1990, "Generalized Hopfield network and nonlinear optimization," in *Advances in Neural Information Processing Systems*, Vol. 2 (February), ed. D. Touretzky, Morgan Kaufmann Publishers, San Mateo, CA, pp. 355.

[84] Nygard, K. E.; Juell, P.; and Kadaba, N. 1990, "Neural networks for selecting vehicle routing heuristics ," *ORSA Journal on Computing*, Vol. 2, No. 4 (Fall), pp. 353364.

[85] Potvin, J.-Y; Shen, Y; and Rousseau, J.-M. 1992, "Neural networks for automated vehicle dispatching," *Computers and Operations Research*, Vol. 19, No. 3,4, pp. 267-276.

[86] Malakooti, B.; and Zhou, Y.Q. 1994, "Feedforward neural networks for solving discrete multiple criteria decision making problems," *Management Science*, Vol. 40, No. 11, pp. 1542-1561.

[87] Wang, J. 1994, " Neural network approach to modeling fuzzy preference relations for multiple criteria decision making," *Computers and Operations Research*, Vol. 21, No. 9, pp. 991-1000.

[88] Wang, J. and Malakooti, B. 1992, "A feedforward neural network for multiple criteria decision making," *Computers & Operations Research*, Vol. 19, No. 2, pp. 151-167.

[89] Wang, S. and Archer, N.P. 1994, "A neural network technique in modeling multiple criteria multiple person decision making," *Computers and Operations Research*, Vol. 21, No. 2, pp. 127-142.

[90] Sastri, T; English, J.; and Krishnamurthi, M. 1989, "Modeling a markovian decision process by neural network," *Computers and Industrial Engineering*, Vol. 17, No. 1-4, pp. 464-468.

[91] Foo, Y. S. and Takefuji, Y. 1988, "Stochastic neural networks for jobshop scheduling: Parts 1 and 2," in *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, CA, Vol. 2, pp. 275290.

[92] Dagli, C.; and Sittisathanchai, S. 1993, "Genetic neuro-scheduler for job shop scheduling," *Computers and Industrial Engineering* , Vol. 25, No. 1-4, pp. 267-270.

[93] Gulati, S.; Iyengar, S. S.; Toomarian, N.; Protopopescu, V.; and Barhen, J. 1987, "Nonlinear neural networks for deterministic scheduling," in *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, CA, Vol. 4, pp. 745752.

[94] Hellstrom, B. J. and Kanal, L. N. 1990, "Asymmetric meanfield neural networks for multiprocessor scheduling," Working paper #UMIACSTR9099, University of Maryland, College Park, MD.

[95] Johnston, M.D.; and Adorf, H.-M. 1992, "Scheduling with neural networks - The case of the Hubble Space Telescope," *Computers and Operations Research*, Vol. 19, No. 3,4, pp. 209-240.

[96] Kovacic, M. 1993, "Timetable construction with Markovian neural network," *European Journal of Operational Research* , Vol. 69, No. 1, pp. 92-96.

[97] Mausser, H.E.; and Magazine, M.J. 1996, "Comparison of neural and heuristic methods for a timetabling problem," *European Journal of Operational Research*, In Press.

[98] Poliac, M. O.; Lee, E. B.; Slagle, J. R.; and Wick, M. R. 1987, "A crew scheduling problem," in *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, Vol. 4, pp. 779786.

[99] Rabelo, L. C.; Alptekin, S.; and Kiran, Ali S. 1990, "Synergy of artificial neural networks and knowledgebased expert systems for intelligent FMS scheduling," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* San Diego, CA, Vol. 1, pp. 359366.

[100] Sabuncuoglu, I; and Gurgun, B. 1996, "A neural network model for scheduling problems," *European Journal of Operational Research*, In Press.

[101] Tanaka, T.; Canfield, J.R.; Oyanagi, S.; and Genchi, H. 1989, "Optimal task assignment using neural network," in Proceedings of the *IEEE International Joint Conference on Neural Networks*, Washington D.C., Vol. 2, pp. 591.

[102] Vaithyanathan, S. and Ignizio, J. P. 1992, "A stochastic neural network for resource constrained scheduling," *Computers and Operations Research*, Vol. 19, No. 3/4, pp. 241-254.

[103] Yu, T. L. 1990, "Time table scheduling using neural network algorithms," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* San Diego, CA, Vol. 1, pp. 279284.

[104] Zhou, D. N.; Cherkassky, V.; Baldwin, T. R.; and Hong, D. W. 1990, "Scaling neural network for jobshop scheduling," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* San Diego, CA, Vol. 3, pp. 889894.

[105] Burke, L. L. and Rangwala, S. 1991, "Tool condition monitoring in metal cutting: A neural network approach," *Journal of Intelligent Manufacturing*, Vol. 2, No. 5 (October), pp. 269280.

[106] Chakraborty, K.; and Roy, U. 1993, "Connectionist models for part-family classifications," *Computers and Industrial Engineering*, Vol. 24, No. 2, pp. 189-198.

[107] Dagli, C. and Lammers, S. 1989, "Possible applications of neural networks in manufacturing," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* Washington D.C., Vol. 2, pp. 605.

[108] Kao, Y. and Moon, Y. B. 1991, "A unified group technology implementation using the backpropagation learning rule of neural networks," *Computers and Industrial Engineering*, Vol. 20, No. 4, pp. 425437.

[109] Kaparthi, S. and Suresh, N.C. 1994, "Performance of selected part-machine grouping techniques for data sets of wide ranging sizes and imperfection," *Decision Sciences*, Vol. 25, No. 4, pp. 515-539.

[110] Kaparthi, S.; Suresh, N.C.; and Cervaney, R.P. 1993, "An improved neural network leader algorithm for part-machine grouping in group technology," *European Journal of Operational Research*, Vol. 69, No. 3, pp. 342-356.

[111] Ko, T.J.; Cho, D.W.; and Jung, M.Y. 1995, "On-line monitoring of tool breakage in face milling using a self-organized neural network," *Journal of Manufacturing Systems*, Vol. 14, No. 2, pp. 80-90.

[112] Kumara, S. R. T.; and Kamarthi, S. V. 1991, "Function-to-structure transformation in conceptual design: An associative memory paradigm," *Journal of Intelligent Manufacturing*, Vol. 2, pp. 281-292.

[113] Lee, S. and Chen, H. 1991, "Design optimization with back propagation neural networks," *Journal of Intelligent Manufacturing*, Vol. 2, No. 5 (October), pp. 293304.

[114] Lee, S. and Park, J. 1991, "Neural computation for collision free path planning," *Journal of Intelligent Manufacturing*, Vol.2, No.5 (Oct.), pp. 315326.

[115] Liao, T.W.; and Lee, K. S. 1994, "Integration of a feature-based CAD system and an ART1 neural model for GT coding and part family forming," *Computers and Industrial Engineering* , Vol. 26, No. 1, pp. 93-104.

[116] Madey, G. R.; Weinroth, J.; and Shah, V. 1992, "Integration of neurocomputing and system simulation for modeling continuous improvement systems in manufacturing," *Journal of Intelligent Manufacturing*, Vol. 3, pp. 193-204.

[117] Malave, C. O. and Ramachandran, S. 1991, "Neural network based design of cellular manufacturing systems," *Journal of Intelligent Manufacturing*, Vol. 2, No. 5 (October), pp. 305314.

[118] Romano, R.; Maimon, O.; and Furst, M. 1989, "Neural net implementation for assigning a product to a production line," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* Washington D.C., Vol. 2, pp. 577.

[119] Sieger, D.B.; and Badiru, A.B. 1993, "An artificial neural network case study: Prediction versus classification in a manufacturing pplication," *Computers and Industrial Engineering*, Vol. 25, No. 1-4, pp. 381-384.

[120] Smith, K.; Palaniswami, M.; and Krishnamoorthy, M. 1996, "Traditional heuristic versus hopfield neural network approaches to a car sequencing problem," *European Journal of Operational Research*, In Press.

[121] Sunil, E. V. T.; Shin, Y. C.; and Kumara, S. R. T. 1990," Machining condition monitoring via neural networks," in *Monitoring and Control of Manufacturing Processes*, eds. S. Y. Liang and T. Tsolo, ASME Press, Fairfield, NJ, pp. 85-95.

[122] Wang, J. 1994, " A neural network approach to multiple objective cutting paramater optimization based on fuzzy preference information," *Computers and Industrial Engineering*, Vol. 25, No. 1-4, pp. 389-392.

[123] Wang, Q.; Sun, X.; Golden B.L.; and DeSilets, L. 1993, "A neural network model for the wire bonding proces," *Computers and Operations Research*, Vol. 20, No. 8, pp. 879-888.

[124] Zhou, Y. and Malakooti, B. 1991, "Inprocess regressions and adaptive neural networks for monitoring and supervising machining operations," Technical report #91182, Case Western Reserve University, Cleveland, OH.

[125] Benjamin, C.O.; Chi, S.-C.; Gaber, T.; and Riordan, C.A. 1995, "Comparing BP and ART II neural network classifiers for facility location," *Computers and Industrial Engineering* , Vol. 28, No. 1, pp. 43-50.

[126] Gaber, M.T.; and Benjamin, C.O. 1992, "Classifying U.S. manufacturing plant locations using an artificial neural network," *Computers and Industrial Engineering*, Vol. 23, No. 1-4, pp. 101-104.

[127] Vaithyanathan, S.; Burke, L.I.; and Magent, M.A. 1996, "Massively parallel analog Tabu search using neural networks applied to simple plant location problems," *European Journal of Operational Research*, In Press.

[128] Alpaydin, E.; Altinel, K.I.; and Aras, N. 1996, "Parametric distance functions vs. nonparametric neural networks for estimating road travel distances," *European Journal of Operational Research*, In Press.

[129] Yuan, Y; and Wang, L. 1996, "A neural network approach to solve the stable matching problem," *European Journal of Operational Research*, In Press.

[130] Steiger, D.M.; and Sharda, R. 1996, "Analyzing mathematical models with inductive learning networks," *European Journal of Operational Research*, In Press.

[131] Cheng, C.-S. 1995, "A multi-layer neural network model for detecting changes in the process mean," *Computers and Industrial Engineering*, Vol. 28, No. 1, pp. 51-61.

[132] Hwarng, H. B.; and Hublee, N. F. 1993, "X control chart pattern identification through efficient off-line neural network training," *IIE Transactions*, Vol. 25, No. 3, pp. 27-40.

[133] Pugh, G.A. 1989, "Synthetic neural networks for process control," *Computers and Industrial Engineering*, Vol. 17, No. 1-4, pp. 24-26.

[134] Anandalingam, G.; Mathieu, R.; Pittard, C. L.; and Sinha, N. 1989, "Artificial intelligence based approaches for solving hierarchical optimization problems," in *Impact of Recent Computer Advances on Operations Research*, eds. R. Sharda et al, North Holland, New York, pp. 289301.

[135] Denton, J.W.; and Hung, M.S. 1996, "A comparison of non linear optimization methods for supervised learning in multilayer feedforward neural networks," *European Journal of Operational Research*, In Press.

[136] Denton, J. W. and Hung, M. S. 1992, "Second order training methods for supervised learning in neural networks," paper presented at the TIMS/ORSA Joint National Meeting, Orlando, FL.

[137] Hung, M.S.; and Denton, J.W. 1993, "Training neural networks with the GRG2 nonlinear optimizer," *European Journal of Operational Research*, Vol. 69, No. 1, pp. 83-91.

[138] Shawe-Taylor, J. S. and Cohen, D. A. 1990, "Linear programming algorithm for neural networks," *Neural Networks* , Vol. 3, No. 5, pp. 575-582.

[139] Bennet, K. P. and Mangasarian, O. L. 1992a, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software*, Vol. 1, pp. 23-34.

[140] Bennet, K. P. and Mangasarian, O. L. 1992b, "Neural network training via linear programming," in *Advances in Optimization and Parallel Computing*, ed. P. M. Paradalos, North Holland, Amsterdam, pp. 56-67.

[141] Roy, A. and Mukhopadhyay, S. 1991, "Pattern classification using linear programming," *ORSA Journal on Computing*, Vol.3, No.1 (Winter), pp. 6680.

[142] de Werra, D. and Hertz, A. 1989, "Tabu search techniques: A tutorial and an application to neural networks," *OR Spektrum*, Vol.11 No.3, pp.131-141.

[143] Jones, K. L.; Lustig, I. J.; and Kornhauser, A. L. 1990, "Optimization techniques applied to neural networks: Line search implementation for backpropagation," in *Proceedings of the IEEE International Joint Conference on Neural Networks,* San Diego, CA, Vol. 3, pp. 933940.

[144] Bennet, K. P. 1992, "Decision tree construction via linear programming," in *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Conference*, pp. 97-101.

[145] Baek, W.; and Ignizio, J.P. 1993, "Pattern classification via linear programming," *Computers and Industrial Engineering* , Vol. 25, No. 1-4, pp. 393-396.

Papatya Programlama Dili Kitapları

Papatya Yayıncılık, programlama dilleri konusunda nitelikli ve özgün birçok kitaba sahiptir. Programlama dilleri konusunda yeni yeni kitaplar hazırladığı gibi var olan kitaplarını da güncellemektedir. Programlama dilleri ve bu konudaki gelişmeler Papatya Yayıncılık kitaplarıyla yakından izlenebilir. İşte yayınevimizin programlama üzerine kitapları:

- **C Programlama Dili**
  Dr. Rifat ÇÖLKESEN
  376 sayfa (8. basım), 16,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **Uygulamalı C Programlama Dili**
  Bora TUNÇER
  344 sayfa, 16,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **JAVA ve Yazılım Tasarımı**
  Altuğ B. ALTINTAŞ
  688 sayfa, 18,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **C ile Programlama Sanatı Algoritmalar (Yeni Başlayanlar için)**
  Dr. Cengiz UĞURKAYA
  336 sayfa, 16,5x21 cm^2, 80 gr. 1. hamur kağıt.
- **C++ ile Programlama Sanatı Algoritmalar (Yeni Başlayanlar için)**
  336 sayfa, 16,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **JAVA ile Programlama Sanatı Temel Algoritmalar (Yeni Başlayanlar için)**
  336 sayfa, 16,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **Linux Altında Programlama**
  M Ali VARDAR
  288 sayfa, 16,5x24 cm^2, 80 gr. 1. hamur kağıt.
- **Veri Yapıları ve Algoritmalar (Programlama ve Yazılım Mühendisliğinde)**
  Dr. Rifat ÇÖLKESEN
  424 sayfa, 18,5x24 cm^2, 90 gr. 1. hamur kağıt.
- **ve diğerleri için: www.papatya.info**

C Programlama Dilini Öğrenmenin En Kolay Yolu

İşte C PROGRAMLAMA DİLİ

Dr.Rifat ÇÖLKESEN

Bu kitap C dilini öğretmek üzere tasarlanmış olup X3J11 grubunun belirlediği standart olan ANSI C temel alınmıştır; standart C içerisinde tanımlı bütün C fonksiyonları ayrıntılı olarak ele alınarak, aynı zamanda bir başvuru kitabı olma özelliği kazandırılmıştır. Kitap, hangi programlama diliyle olursa olsun program geliştiren yazılımcıların başvuru kitabı gibi yararlanabileceği; ve, Üniversitelerin Bilgisayar, Elektronik, Elektrik, Fizik, Matematik ve Endüstri bölümlerinde okutulan C programlama dersi için ders kitabı olabilecek niteliktedir.

Kitap herbiri farklı konulara odaklanmaya çalışılan 16 Bölüm ve 2 Ekten oluşmaktadır. C dilinin bu kitapla öğrenilmesi için bütün bölümler sırayla okumalı, örnek olarak verilen programlar dikkatle incelenmeli ve sorular yanıtlanmaya çalışılmalıdır. Verilen örnekler ANSI C'yi destekleyen herhangi bir derleyicide derlenebilir şekildedir.

Kitabın ilk dört bölümünde, C'yi yeni öğrenenler için değişkenler, sabitler, operatörler ve program denetim deyimleri gibi C dilinin temel yapı taşları verilmektedir; 5. Bölüm ise C fonksiyon yapısını ayrıntılı olarak ele almakta ve Bölüm 6'da Standart C'de tanımlı bütün fonksiyonlar açıklanmaktadır. Sırasıyla 7, 8, 9, 10, 11 ve 12. Bölümlerde Diziler, İşaretçiler (*pointers*) Katarlar (*strings*) ve Yapısal Veri Tipleri ele alınmıştır. Onüçüncü bölümde ise Disk Dosyaları üzerinde durulmuştur; dosyanın açılması, dosyaya erişilmesi ve kapatılması için kullanılan standart fonksiyonlar ve UNIX işletim sisteminin disk dosyası için kullanılan sistem çağrıları da bu bölümde verilmiştir. Ondördüncü bölümde esnek ve taşınabilir program tasarımına imkan veren önişlemçi komutları, onbeşinci bölümde ise seri haberleşme için gerekli program yapısı ele alınmıştır. Onaltıncı bölümde, gerçeklenmesi C derleyicisi hazırlayanlara bırakılan, bir standart belirlenmeyen Grafik Fonksiyonlar üzerinedir. Bu bölümde grafikle ilgili bazı kavramlar verilmiş ve Turbo C 2.0 paketinin grafik fonksiyonları tanıtılmıştır.

Eklerde ise, önce bir C programının nasıl derleneceği/çalıştırılacağı üzerinde durulmuş ve ardından ASCII çizelge verilmiştir. Bir C programının PC tabanlı bir işletim sisteminde ve UNIX işletim sisteminde nasıl derleneceği ve çalıştırılacağı açıklanmıştır.

Bölüm 10'da büyükçe sayılabilecek 3 tane örnek verilmiş ve adım adım açıklanmıştır; hemen hemen C dilinin tüm özellikleri kullanılmaya çalışılmıştır. Aynı uygulama, dizi, bağlantılı liste ve ikili ağaç modeline göre uygulanarak adım adım açıklanmış ve aralarındaki farklara değinilmiştir. Bu içeriğiyle kitap, C dilini öğrenmek isteyenler ve C dilinde başvuru kitabına ihtiyaç duyacak programcılar ve sistem yöneticileri için ciddi bir kitap niteliğindedir.

Bilgisayar Programlama ve Yazılım Mühendisliğinde

VERİ YAPILARI VE ALGORİTMALAR

Dr. Rifat ÇÖLKESEN

Bu kitap, program geliştiren, matematik ve mühendislik problemlerini bilgisayar ortamında çözmek isteyen, iş dünyasına yönelik yazılım tasarımları yapan her düzeyden programcı veya yazılımcılar için ciddi bir başvuru kitabıdır. Kitap, aynı zamanda, üniversitelerin bilişimle ilgili bölümlerinde okutulan Veri Yapıları ve Algoritmalar dersleri için bir ders kitabı özelliğindedir. Program ve yazılım tasarımında, ciddi bir bakış açısı yakalamak isteyenlere önerilir...

Veri Yapıları ve Algoritmalar, program tasarımında çoğu zaman eksikliği hissedilen önemli bir konu; yalnız başına bir programlama dili bilmek, program geliştirmeye yetmemektedir. Bu kitap, C programlama diline dayanılarak çeşitli veri yapıları ve modellerini ele almakta, onlara ait programın algoritmik ifadesini incelmekte ve örneklerle açıklamaktadır; bütün bunlara ek olarak, tasarımı konularına uygun olarak adım adım açıklanmış ve Network yazılımı ve Veri modeli program tasarımında yapılması gereken aşamalar sistem analizi ve de açıklanmıştır.

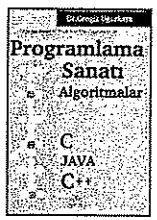
Program tasarımında en önemli konu, ele alınan uygulamaya en uygun veri modelinin belirlenmesi, veri yapısının tanımlanması ve programın algoritmik olarak ifade edilmesidir. Veri yapısı, verinin veya bilginin bellekte tutulma şeklini ve düzenini gösterir. Veri modeliyse, verilerin birbirleriyle ilişkisel ve sırasal durumunu gösterir. Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yaklaştırılarak veya yeni veri modelleri tanımlanması yapılarak çözülebilmektedir. Uygun bir veri modeliyle çözüme gidilemeyen problemler, çoğu zaman ya çözümsüz kalmakta veya bellek yetmiyor, bilgisayarın hızı yetmiyor gibi sebeplerle yarım bırakılmaktadır. Uygulamada, her problem, doğası gereği en uygun bir veri modeline sahiptir.

Kitap, sırasıyla şu bölümleri kapsamaktadır: Bilgisayar Yazılım Dünyası, Program/Yazılım Geliştirme Süreci, Algoritmik Yaklaşımda C Dili Esnekliği ve Özellikleri, Veri Yapıları, Veri Modelleri, Algoritmalar ve Tasarım Yaklaşımları, Program Çalışma Hızı ve Bellek Gereksinimi, Sıralama Algoritmaları, Arama Algoritmaları, Özel Amaçlı Algoritmalar, Bağlantılı Listeler ve Uygulamaları, Ağaçlar ve Uygulamaları, Graf'lar ve Uygulamaları, Veritabanı İlişkisel Veri Modeli ve SQL, Network Yazılımı ve Veri Modeli, Adaptif Programlama.

Bilgisayar Programcılığı Temelleri & Yeni Başlayanlar için

PROGRAMLAMA SANATI ALGORİTMALAR

Dr. Cengiz UĞURKAYA



Bu eser program tasarımına yeni başlayanlar için programlama temelleri ve mantığını kazandırmak amacıyla hazırlanmıştır. Bilindiği gibi programlama dilleri öğrenilmeden önce programlama tekniği ve algoritma tasarımı mantığı öğrenilmedir. Programlama Sanatı Algoritmalar adlı bu kitabımız öğrencilerimize bol bol akış şeması örnekleri vererek kendi kendilerine programlama yapabilme ve algoritma oluşturabilme becerisi ve yeteneği kazandırmaya çalışmaktadır. Çeşitli programlama dilleri için ayrı ayrı baskıları vardır...

Bu kitabın C, JAVA ve C++ olmak üzere 3 ayrı baskısı vardır; herbiri farklı bir programlama diliyle başlangıç yapmak isteyen genç program tasarımcısı adaylarını hedeflemiştir. Dolayısıyla ileride hangi programlama dili kullanılacaksa ona ait Programlama Sanatı Algoritmalar kitabı kullanılabilir. Her 3 uyarlama içinde de örnekler ve metinsel ifadeler aynı olup ilgili dilin özelliğine göre ek konular da vardır.

Güçlü bir temel kazanmak program tasarımı için en önemli başlangıç adımıdır denilebilir.

Programlama tekniğini ve mantığını öğrenmek ve bol örneklerle geliştirmek isteyen öğrencilerimize yararlı bir kaynak özelliğindedir.

Üniversitelerde ilk yıllarda verilen "Bilgisayar Programlama", "Programlama Teknikleri", "Programlamaya Giriş" ve "Programlama Mantığı" gibi derslerin ana kitabı veya yardımcı kitabı olma özelliğindedir.

Programlama teknikleri ve algoritma tasarımı konusu aşağıdaki ana başlıklar ışığında öğretilmeye çalışılmıştır:

- Program Tasarımında Temel Kavramlar
- Programlama Dilleri ve C/JAVA/C++
- Algoritma Tasarımı ve Akış Şemaları
- Programlama Diline Giriş: C, JAVA veya C++
- Programlama Dili Temelleri: C, JAVA veya C++
- Programlama Dillerinde Veri Yapıları
- Çevrimli ve Rekürsif Programlama Teknikleri
- Temel Uygulamalar
- Dizi ve Matris Uygulamaları
- Matematiksel Uygulamalar
- İstatistiksel Uygulamalar
- İşletim Sistemine Dayalı Uygulamalar
- Programda Bellek ve Zaman Maliyetleri
- Bol Bol Soru

Papatya NETWORK Kitapları

Papatya Yayıncılık, bilgisayar ağları konusunda nitelikli ve özgün birçok kitaba sahiptir; bu konuda yeni yeni kitaplar hazırladığı gibi var olan kitaplarını da gelişmelere göre güncellemektedir. Network konusunda teknolojik gelişmeler Papatya Yayıncılık kitaplarıyla yakından izlenebilir. İşte yayınevimizin halihazırda basılı olan kitapları:

- **Bilgisayar Haberleşmesi ve Ağ Teknolojileri**
  Dr.Rifat ÇÖLKESEN ve Prof.Dr.Bülent ÖRENCİK
  3. basım: 448 sayfa, 18,5x24 cm^2, 90 gr. 1. hamur kağıt.
- **Veri Haberleşmesi Temelleri**
  Yasin KAPLAN
  1. basım: 352 sayfa, 16,5x24 cm^2, 85 gr. 1. hamur kağıt.
- **Network TCP/IP ve UNIX El Kitabı**
  Dr.Rifat ÇÖLKESEN
  3. basım: 260 sayfa, 15x21 cm^2, 90 gr. 1. hamur kağıt.
- **Veri Haberleşmesi Kavramları**
  Yasin KAPLAN
  2. basım: 208 sayfa, 15x21 cm^2, 90 gr. 1. hamur kağıt.
- **Veri Haberleşmesi/Network Uygulamaları**
  Yasin KAPLAN
  1. basım: 240 sayfa, 16,5x24 cm^2, 85 gr. 1. hamur kağıt.
- **Bilgisayar Ağları**
  Dr.B.Demir ÖNER
  1. basım: 312 sayfa, 18,5x24 cm^2, 90 gr. 1. hamur kağıt.
- ve hazırlanan diğerleri...

OSI Başvuru Modeli, Hata Sezme ve Düzeltme Teknikleri: CRC/LRC, Asenkron Seri İletişim, Ağ Katmanı, Veri Bağı Katmanı, Fiziksel Katman, Kodlama Teknikleri, İletim Hattı Teorisi, İletişim Kanalı Başarım Hesabı, Ağ Güvenliği.

LAN Teknolojileri, WAN Teknolojileri, 802.x ailesi, CSMA/CD, Ethernet, Yüksek Hızlı Ethernet, Jetonlu Halka, Jetonlu Yol, ATM, FDDI, ISDN: BRI/PRI, xDSL: ADSL/VDSL, *Frame Relay*, X.25, E1/E3, Kanallı E1/E3, *Intranet/Extranet*, vLAN, *Router* Konfigürasyonu

TCP/IP protokol kümesi, İnternet Uygulamaları, IP Yönlendirme, Soket Programlama, IP adresleri ve Altağlar, Sunucu sistemler, bilgisayarların TCP/IP tabanı ağa eklenmesi, İnternet Hizmet Programlarının Kurulması

Yapısal Kablolama, RJ45 Sonlandırması, Cat 5 Kablo, Fiber Optik Kablo, ISDN kablo bağlantısı, RS-232 seri kablo bağlantısı, Kablolama Üzerine Standartlar, Çapraz bağlantı, Kablolama Standartları.

Yönlendirici (*Router*), Anahtar (*Switch*), *Network* Kartı, HUB, Güvenlik Duvarı (*Firewall*), Modem, Erişim Sunucu (*Access Server*), Ortam Dönüştürücüler, İnternet Erişim Paylaştırıcı ve diğer konular Papatya Yayıncılık *Network* kitaplarında...

Bilgisayar Haberleşmesi Konusunda Temel Bilgiler

NETWORK/VERİ HABERLEŞMESİ TEMELLERİ

Yasin KAPLAN

OSI modeli esas alınarak hazırlanan kitap, üniversite öğrencileri ve bilgisayar ağları ile profesyonel olarak ilgilenen mühendisler için hazırlanmıştır. Sayısal iletimin temelleri, hata tespiti ve onarma/düzeltme teknikleri gibi başlangıç konularla yapılan bir girişi takiben, seri arayüzler, modemler, SDLC ve türevleri, X.25, PPP, ISDN, Frame Relay ve ATM konuları ele alınmıştır.

Kitapta, ağırlıklı olarak değinilen WAN kavramlarının yanı sıra Ethernet gibi. LAN kavramlarına da yer verilmiştir. Ayrıca ayrıntılı bir TCP/IP ve IP yönlendirme bölümü bulunmaktadır. H.323 Video & Voice over IP ve IPv6 gibi güncel başlıklar hakkında derin bilgilere yine bu kitap üzerinden ulaşabilirsiniz. E1 sayısal iletim ortamı hakkında çok detaylı bir bölüm bulunmaktadır.

- OSI Başvuru Modeli
- İletim Hattı Teorisi
- Sayısal Kodlama
- Hata Denetimi/Düzeltimi
- Seri Haberleşme ve RS-232 Arayüzü
- Modemler
- Synchronous Data Link Control, SDLC ve Türevleri
- X.25 Protokol Kümesi
- E1 İletim Şekli
- ISDN *(BRI ve PRI)*
- Frame Relay
- PPP (Point to Point Protocol)
- ATM Temelleri
- xDSL *(ADSL, SDSL, IDSL)*
- Ethernet ve Türleri
- TCP/IP
- IP Sürüm 6 *(IPv6)*
- IP Trafiğini ATM Ağları Üzerinde Taşımak İçin Yöntemler
- IP-IPX Yönlendirme Temelleri
- IP Yönlendirme Protokolleri
- Telnet Protokolü
- Dosya Aktarım Protokolü *(FTP)*
- H.323 Standardı*(Voice and Video over IP)*
- RFC'ler

PAPATYA YAYINCILIK
Temel Bilimler/Teknik Kitaplarımız

- **Yapay Sinir Ağları** ⇨ Prof.Dr.Ercan ÖZTEMEL - *elinizdeki kitap!*
- **Türkiye Bilişim Ansiklopedisi** ⇨ Başeditörler: ÖREN, ÜNEY, ÇÖLKESEN
- **Sistem Analizi ve Tasarımı** ⇨ Prof.Dr.Oya KALIPSIZ, A. BUHARALI, G. BİRİCİK
- **Yazılım Mühendisliği** ⇨ Dr. Erhan SARI
- **C++ ve Nesneye Yönelik Programlama** ⇨ Dr. Erhan SARI
- **İnternet Teknolojileri ve İntranet Uygulamaları** ⇨ Türker CAMBAZOĞLU
- **JAVA ve Yazılım Tasarımı** ⇨ Altuğ B. ALTINTAŞ
- **Programlama Sanatı Algoritmalar (C ile)** ⇨ Dr.Cengiz UĞURKAYA
- **Programlama Sanatı Algoritmalar (JAVA ile)**
- **Programlama Sanatı Algoritmalar (C++ ile)**
- **Veri Yapıları ve Algoritma Temelleri** ⇨ Dr.Sefer KURNAZ
- **Veri Yapıları ve Algoritmalar (Bilgisayar Prog. ve Yazılım Müh.)** ⇨ Dr.Rifat ÇÖLKESEN
- **Bilgisayar Ağları** ⇨ Dr.B.Demir ÖNER
- **Bilgisayar Haberleşmesi ve Ağ Teknolojileri** ⇨ Dr.Rifat ÇÖLKESEN ve Prof.Dr.Bülent ÖRENCİK
- **Veri Haberleşmesi Temelleri** ⇨ Yasin KAPLAN
- **NETWORK/Veri Haberleşmesi Uygulamaları** ⇨ Yasin KAPLAN
- **Veri Haberleşmesi Kavramları** ⇨ Yasin KAPLAN
- **NETWORK TCP/IP ve UNIX El Kitabı** ⇨ Dr.Rifat ÇÖLKESEN
- **C Programlama Dili** ⇨ Dr.Rifat ÇÖLKESEN
- **Uygulamalı C Programlama Dili** ⇨ Bora TUNÇER
- **GTK+/GNOME Programlama** ⇨ M. Ali VARDAR
- **LINUX Altında Program Geliştirme** ⇨ M. Ali VARDAR
- **Lojik Devre Tasarımı** ⇨ Dr.Taner ARSAN ve Dr.Rifat ÇÖLKESEN
- **Enerji Sistemleri** (Cilt I, III) ⇨ Prof.Dr. Nariman ŞERİFOĞLU
- **Enerji Sistemleri** (Cilt II) ⇨ Prof.Dr. Nariman ŞERİFOĞLU ve Prof.Dr. Oğuz SOYSAL
- **Elektromagnetik Dalga Teorisi Çözümlü Problemleri** ⇨ G. UZGÖREN, A. BÜYÜKAKSOY
- **Elektromagnetik Alan Teorisi Çözümlü Problemleri** ⇨ G. UZGÖREN, A. BÜYÜKAKSOY
- **ISI Transferi: Teorik ve Uygulamalı** ⇨ Prof.Dr. Tuncay YILMAZ
- **Lineer CEBİR** ⇨ Prof.Dr. Veli ŞAHMUROV ve Prof.Dr. Gökhan UZGÖREN
- **Lineer CEBİR Uygulamaları** ⇨ Prof.Dr. Veli ŞAHMUROV ve Prof.Dr. Gökhan UZGÖREN
- **Matematik Analizi ve Uygulamaları** ⇨ Prof.Dr. Elimhan MAHMUDOV
- **Diferansiyel Denklemler Teorisi** ⇨ (Prof.Dr.) E.HASANOV, G.UZGÖREN, A.BÜYÜKAKSOY
- **ÖSS Matematik Çözümlü Soru Bankası** ⇨ Erhan SARI

PAPATYA YAYINCILIK
Sosyal/İşletme/Hukuk Kitaplarımız

- **Satışçılara Öneriler** ⇨ Doç.Dr. Erdoğan TAŞKIN
- **Kurumsal Yönetim** ⇨ Dr. Veysel KULA
- **Satış Teknikleri Eğitim** ⇨ Doç.Dr. Erdoğan TAŞKIN
- **Fonksiyonları Açısından İşletme ve Yönetim** ⇨ Dr. Mehmet ÖZTÜRK
- **Satış Yönetimi Eğitimi** ⇨ Doç.Dr. Erdoğan TAŞKIN
- **İşletme Yönetiminde Eğitim ve Geliştirme** ⇨ Doç.Dr. Erdoğan Taşkın
- **Müşteri İlişkileri Eğitimi** ⇨ Doç.Dr. Erdoğan Taşkın
- **Öğrenen Organizasyon ve Rekabet Üstünlüğü** ⇨ Salim ÇAM
- **Politik Pazarlama** ⇨ Dr. Ahmet TAN
- **Politika'da Niye Kaybediyorlar? Nasıl Kazanırlar?** ⇨ Dr. Ahmet TAN
- **İnsan Kaynakları: Kişi ve Kurumlara Öneriler** ⇨ Meltem YAMAN
- **İnternet ve Hukuk** ⇨ Ali Osman ÖZDİLEK

Dil/Türk Dili Kitaplarımız

- **Türk Dilinde Çatı** ⇨ Prof.Dr.Ömer DEMİRCAN
- **Biçimbilim: Temel Kavramlar** ⇨ N. Engin UZUN
- **Türkiye Türkçesinde Biçimbirimler** ⇨ Oya ADALI
- **Türkiye Türkçesinde Kök-Ek Bileşmeleri** ⇨ Prof.Dr.Ömer DEMİRCAN
- **Türkçe Üzerine: Denemeler ve Eleştiriler** ⇨ Yusuf ÇOTUKSÖKEN
- **Dilin İşlevleri ve İletişim** ⇨ Dr. Veysel KILIÇ
- **Uygulamalı Türk Dili-II** ⇨ Yusuf ÇOTUKSÖKEN
- **Uygulamalı Türk Dili-I** ⇨ Yusuf ÇOTUKSÖKEN
- **Uygulamalı Türk Dili - TEK CİLT** ⇨ Yusuf ÇOTUKSÖKEN
- **Türkiye Türkçesinin Sözdizimi** ⇨ Neşe ATABAY, Sevgi ÖZEL ve Ayfer ÇAM
- **Sözcük Türleri** ⇨ Neşe ATABAY, Dr.İbrahim KUTLUK ve Sevgi ÖZEL

Tıp/Sağlık Kitaplarımız

- **9 Ay 10 Gün: Gebelik Süreci ve Bebeğin Gelişimi** ⇨ Op.Dr. Kağan KOCATEPE
- **28 Gün/Kadın Olmak** ⇨ Op.Dr. Kağan KOCATEPE
- **Çocuğumuz ve Sağlıklı Beslenmesi** ⇨ Dr. İsmail TUNÇDOĞAN ve Dr. C. Ahmet TUNÇDOĞAN
- **Çocuk Sağlığı Kılavuzu** ⇨ Dr.Doğu ERKER
- **Gazlı Sindirim Bozuklukları ve Doğal Tedavisi** ⇨ Dr. N. Suthi ATMACA

DİZİN