# YAPAY SİNİR AĞLARI

3.1.Yapay Sinir Ağlarının Genel Tanımı

3.2. Yapay Sinir Ağlarının Görevi

3.3. Yapay Sinir Ağlarının Genel Özellikleri

3.4. Yapay Sinir Ağları İle Neler Yapılabilir?

3.5. Yapay Sinir Ağlarının Tarihçesi

3.6. Yapay Sinir Ağlarının Yapısı ve Temel Elemanları

3.7. Yapay Sinir Ağı Modelleri

3.8. Eğiticili YSA Modeli

3.9. Eğiticisiz YSA Modeli

# 3. YAPAY SİNİR AĞLARI

## 3.1.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.

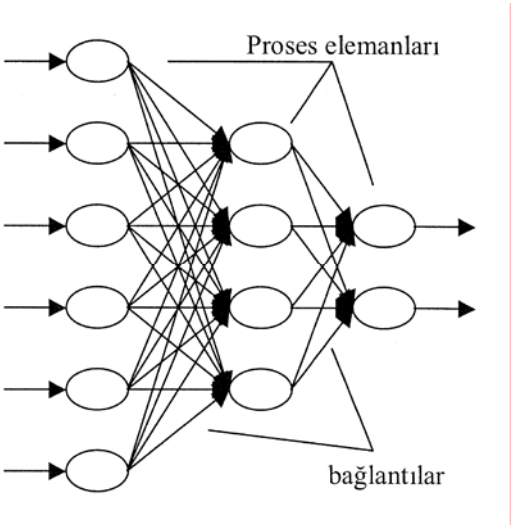
## 3.2. Yapay Sinir Ağlarının Görevi

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
- 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. Proses elemanlarının birbirleri ile bağlanmaları sonucu oluşan ağa yapay sinir ağı denmektedir. Bu ağın oluşturulması biyolojik sinir sistemi hakkındaki bulgulara dayanmaktadır.




**Şekil 3.1.** 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.

### 3.3. 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 zekâ 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. Hâlbuki 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 kararma 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.

### 3.4. 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 birçok problem aslında şekil tanıma problemi haline getirilmekte ve ondan sonra çözülmektedir. Bu nedenle, yapay sinir ağlarının kullanılabileceği birçok 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 sensor 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 birçok 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 çizelgelim 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 imkânsızlaştırmaktadır.

### 3.5. Yapay Sinir Ağlarının 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 tarihsel gelişimine bakıldığında ise 1970 yılının bir dönüm noktası olduğu görülmektedir. Bu tarihten önce birçok 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 zekâ hem de donanım teknolojisindeki gelişmeler ile de desteklenmiştir. Artık bilgisayarlarında öğrenebileceğini herkes kabul etmekte ve bu teknolojiden faydalanmak istemektedir.

### 3.6. Yapay Sinir Ağlarının Yapısı Ve Temel Elemanları

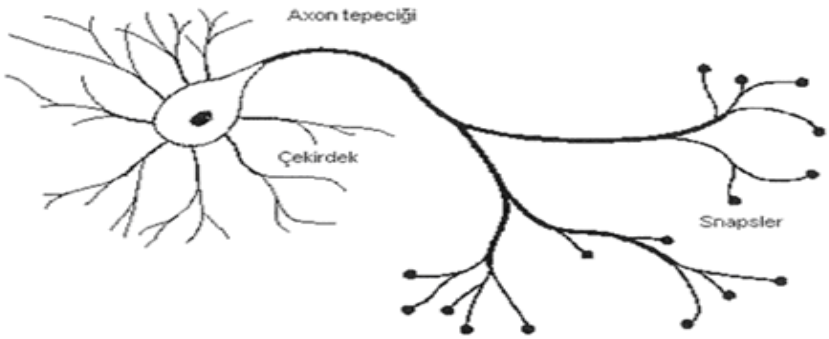
Sinir sistemi birbiri ile iletişim halinde olan sinir hücrelerinden oluşmaktadır. Aşağıda bir sinir hücresinin yapısı açıklanmıştır.

### a) Biyolojik Sinir Hücreleri

Biyolojik sinir ağları beynimizde bulunan birçok 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 1010 adet sinir hücresi ve bunlarında 6x10' 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. Hâlbuki 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 yetenektedir. 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.

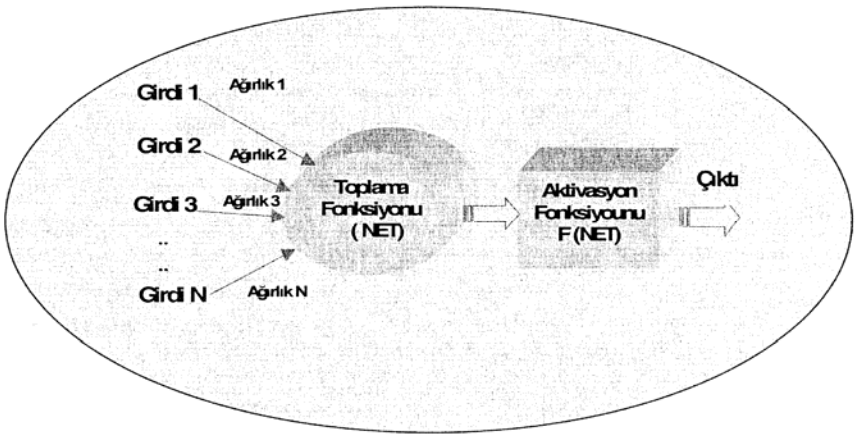



**Şekil 3.2.** Bir biyolojik sinir hücresinin yapısı

Şekil 3.2'de gösterildiği gibi temel bir biyolojik sinir hücresi sinapsler, soma, 0x01 ı ve dentritlerden oluşmaktadır. Sinapslar 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 dendritlere gönderir. Dendritler ise bu sinyalleri sinapslara göndererek diğer hücrelere gönderilir. İki hücrenin birbirleri ile bilgi alış verişi snaptik bağlantılarda neurotransmitterler yolu ile sağlanmaktadır. Şekildeki axon uçlarının her birisi başka bir hücre ile birleşmektedir. Verilen özellikte milyarlarca sinir hücresi bir araya gelerek sinir sistemini oluşturmaktadır. Yapay sinir ağları biyolojik hücrelerin bu özelliklerinden yararlanarak geliştirilmiştir.

### b) 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-3.3). Bunlar:




**Şekil 3.3.** Yapay sinir hücrelerinin yapısı

- **Girdiler:** Bir yapay sinir hücresine (proses elemanına) dış dünyadan bilgilerdir. Bunlar ağm öğrenmesi istenen örnekler tarafından belirlenir. Yapay sinir hücresine dış dünyadan olduğu gibi başka hücrelerden veya kendi kendisinden de bilgiler gelebilir.
- **Ağırlıklar:** Ağırlıklar bir yapay hücreye gelen bilginin önemini ve hücre üzerindeki etkisini gösterir. Şekildeki Ağırlık 1, Girdi l' 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.
- **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.

| **Net giriş** | **Açıklama** |
|---|---|
| *Çarpım*<br>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*<br>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*<br>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*<br>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*<br>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. |

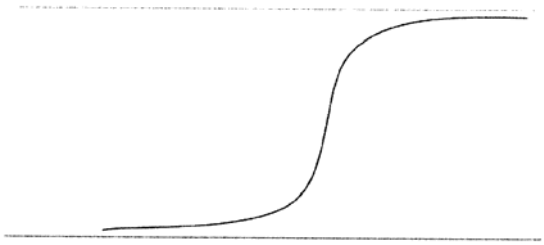
**Tablo 3.1.** Toplama fonksiyonu örnekleri

- **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 3.4.** Sigmoid fonksiyonunun şekilsel gösterimi

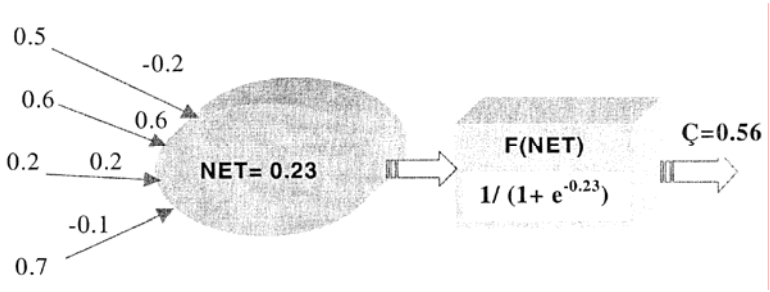
Sigmoid fonksiyonu şekilsel olarak da Şekil-3.4'te gösterilmiştir. Aktivasyon fonksiyonu olarak kullanılacak olan diğer fonksiyonlara örnekler ise Tablo-3.2'de verilmiştir:

| **Aktivasyon fonksiyonu** | **Açıklama** |
|---|---|
| Lineer fonksiyon<br>F(NET)= NET | Gelen girdiler olduğu gibi hücrenin çıktısı olarak kabul edilir. |
| Step fonksiyonu<br>F(NET) = { 1 if NET > eşik_değer<br>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<br>F(NET) = Sin (NET) | Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır. |
| Eşik değer fonksiyonu<br>F(NET) = { 0 if NET<= 0<br>NET if 0<NET< 1<br>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<br>F(NET) = $(e^{NET} + e^{-NET})/(e^{NET} - e^{-NET})$ | Gelen NET girdi değerinin tanjant fonksiyonundan geçirilmesi ile hesaplanır. |

**Tablo 3.2.** Akvitasyon fonksiyonu örnekleri

- **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.

**Örnek:** Yapay sinir hücresinin çalışma prensibi.




**Şekil 3.5.** 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

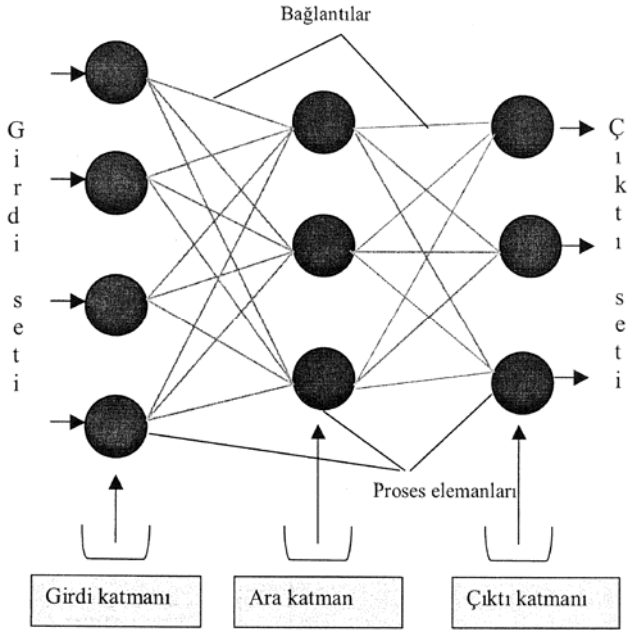
Bir ağdaki bütün proses elemanlarının çıktılarının bu şekilde hesaplanması sonucu ağın girdilere karşılık çıktıları nasıl ürettiği görülür.

### c) 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:

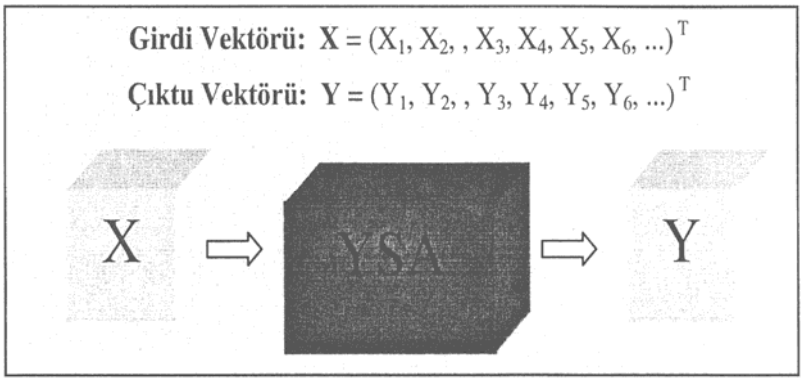
- **Girdi katmanı:** Bu katmandaki proses elemanları dış dünyadan bilgileri alarak ara katmanlara transfer etmekle sorumludurlar. Bazı ağlarda girdi katmanında herhangi bir bilgi işleme olmaz.

- **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 3.6.** Bir yapay sinir ağı örneği

Bu üç katmanın her birinde bulunan proses elemanları ve katmanlar arası ilişkileri şematik olarak Şekil-3.6'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.




**Şekil 3.7.** Yapay sinir ağı girdi, çıktı ilişkisi

Bir YSA, herhangi bir girdi vektörünü çıktı vektörüne nasıl dönüştüğü konusunda bir bilgi vermez. Mühendislik açısından bakıldığında yapay sinir ağları "kara kutu" gibi görülebilirler. Diğer bir deyişle, YSA'nın sonuçları nasıl oluşturduğunu açıklama yeteneği yoktur.

### d) 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.

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.

### e) 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

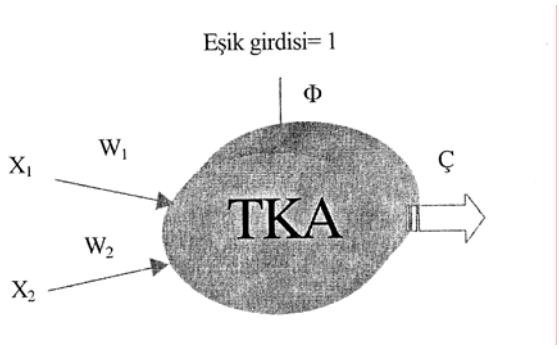
- Çok katmanlı algılayıcılar (hatayı geriye yayma modelleri)
- Vektör Kuantizasyon modelleri (LVQ)
- Kendi kendini organize eden model (SOM)
- Adaptif Rezonans Teorisi modelleri (ART)
- Hopfield ağları
- Counterpropagation ağı
- Neocognitron ağı
- Boltzman makinesi
- Probabilistic ağlar (PNN)
- Elman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

### 3.7. Yapay Sinir Ağı Modelleri

Yapay sinir ağlarının ilk modelleri, tek katmanlı algılayıcı, Perseptron, ve ADALINE/MADALINE yapılarıdır. Bu modeller daha sonra geliştirilen modeller için temel oluşturmaktadır.

**a) 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 3.8'deki ağ iki girdisi ve bir çıktıdan oluşmaktadır. Bu ağlarda proses elemanlarının değerlerinin ve dolayısıyla ağın çıktısının sıfır olmasını önleyen birde eşik değeri (O) vardır. Eşik değerinin girdisi daima l'dir.




**Şekil 3.8.** İki girdi ve bir çıktıdan oluşan en basit TKA model

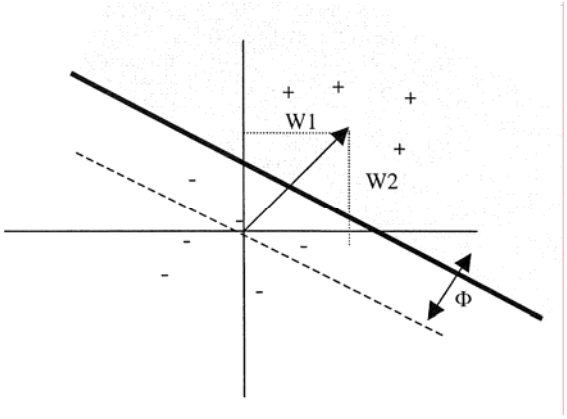
Ağın çıktısı ağırlıklandırılmış girdi değerlerinin eşik değeri ile toplanması sonucu bulunur. Bu girdi değeri bir aktivasyon fonksiyondan geçirilerek ağın çıktısı hesaplanır. Bu, şu şekilde formülize edilmektedir.

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

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 3.9.** Ağırlıkların ve sınıf ayracı olan doğrunun geometrik gösterimi

Sınıf ayracı da denilen bu doğru şu şekilde tanımlanmaktadır.

*W1 .Xl + W2 .X2 + Φ =0*

Buradan *X2 = - ( W1 / W2 ) X1 - Φ / W2* olur.

Benzer şekilde, *X1* = - ( W2 / *W1 ) X2* - Φ / *W2* olarak hesaplanır.

Bu iki formülden hareketle sınıfın ayracı doğrusu çizilebilir. Bu doğrunun geometrik gösterimi ise Şekil 3.9.'da 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;

$Wj\ (t+l)=Wi\ (t)+\Delta\ Wi(t)$

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;

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

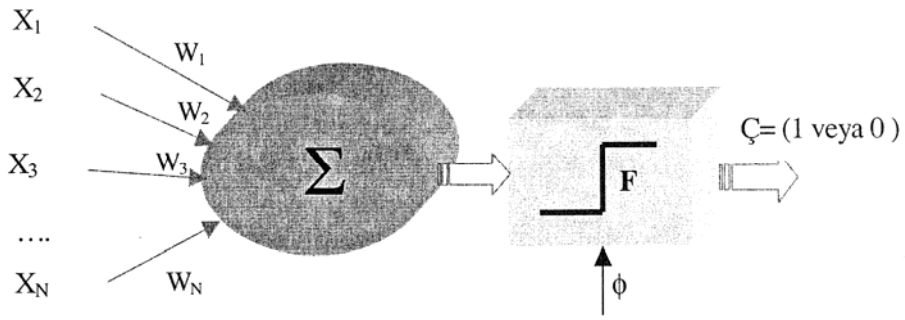
şeklinde değiştirilmesi demektir. Öğrenme sırasında ağırlıklarda olduğu gibi eşik değeri de her iterasyonda $\Delta\Phi$ kadar değiştirilmektedir.

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

**b) Basit Algılayıcı Modeli (Perseptron)**

İlk defa 1958 yılında Rosenblat tarafından örüntü (şekil) sınıflandırma amacı ile geliştirilmiştir.

**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 3.10.' da gösterildiği gibidir:




**Şekil 3.10.** 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 3.11. eşik değerinin sıfır ve "a" pozitif değerleri alması durumunda aktivasyon fonksiyonunun konumunu göstermektedir.


**Şekil 3.11.** Eşik değerinin aktivasyon fonksiyonunun konumuna etkisi

**Basit Algılayıcı Öğrenme Kuralı:** Basit algılayıcıların öğrenme kuralı adım adım aşağıda açıklanmıştır:

**Adım,l:** Ağa girdi setini ve ona karşılık olarak beklenen çıktı gösterilir (X,B). Burada birden fazla girdi değeri olabilir. Yani *X= (* xl, *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;

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

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 $\lambda$ *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.

**Örnek:**

**1. örnek:** X1= (xl, *x2)*= (1,0),B1= 1

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

**Ağırlıklar:** *W* = (wl, w2)= (1,2)

**Eşik değeri:** Φ= -1

**Öğrenme katsayısı:** λ =0.5

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

Basit algılayıcının NET girdisi;

NET= wl*xl +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= wl*xl +*w2*x2* = 1*0+2*1= 2

NET > Φ olduğundan Gerçekleşen çıktı, Ç= 1 olacaktır. Ç≠B2 olduğundan ağırlıklar,

$$Wn = Wo - \lambda X$$

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

wl= $w1 - \lambda X1$

wl= 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= *wl*xl* +w2*x2 = 1*1+1.5*0= 1

NET > Φ olduğundan Gerçekleşen çıktı Ç1= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez

**4. iterasyon 2. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*0+1.5*1= 1.5 olarak hesaplanır.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

*Wn = Wo- λX*

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

*w1 = w\ - λ xl*

*w1*= 1-0.5*0=1

w2= *w2 - λ x2*

w2=1.5-0.5* 1 = 1

**5. iterasyon 1. örnek tekrar gösterilirse**

NET= *wl*xl* +w2*x2 = 1*1+1*0= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Ç1= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez.

**6. iterasyon 2. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*0+1*1= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Beklenen çıktıdan farklı olduğundan ağırlıklar;

Wn = Wo-λX

formülü ile yeni ağırlıklar şöyle bulunur.

wl=wl-λxl wl= 1-0.5*0=1

w2= w2-λx2 w2= 1- 0.5 * 1 = 0.5

**7. iterasyon 1. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*1+0.5*0= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Ç1= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez.

**8. iterasyon 2. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*0+0.5*1= 0.5 olur.

NET >Φ olduğundan gerçekleşen çıktı Çl= 1 olacaktır.Beklenen çıktıdan farklı olduğundan ağırlıklar;

Wn = Wo-λX

formülü ağırlıklar şu değerleri alır.

wl= w1 – λ x1

wl= 1-0.5* 0=1

w2= w2 – λ x2

w2= 0.5- 0.5 * 1 = 0

**9. iterasyon 1. örnek tekrar gösterilirse**

NET= wl*xl +w2*r2 = 1*1+0*0= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez.

**10. iterasyon 2. örnek tekrar gösterilirse**

NET= wl1*xl +w2*x2 = 1*0+0.0*1= 0 olur.

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

Wn = Wo- λ X

formülü kullanılarak değiştirilir ve sonuçta şu değerler oluşur.

wl= w1 - λ x1

wl= 1-0.5* 0=1

w2= w2 - λ x2

w2= 0- 0.5 * 1 = -0.5

**11. iterasyon 1. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*1+ (-0.5)*0= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez.

**12. iterasyon 2. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = l*0+(-0.5)*l= -0.5 olur.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Beklenen çıktıdan farklı olduğundan benzer şekilde ağırlıklar;

Wn = Wo- λ X

formülü kullanılarak şu şekilde değiştirilir.

wl= w1 - λ xl

wl= 1-0.5* 0=1

w2= w2 - λ x2

w2=-0.5-0.5* 1=-1

**13. iterasyon 1. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = 1*1+ (-1)*0= 1 olur.

NET > Φ olduğundan Gerçekleşen çıktı Çl= 1 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez

**14. iterasyon 2. örnek tekrar gösterilirse**

NET= wl*xl +w2*x2 = l*0+(-l)*l= -1 olur.

NET = Φ olduğundan Gerçekleşen çıktı Çl= 0 olacaktır. Çl=Bl olduğundan ağırlıklar değiştirilmez. Bundan sonra her iki örnekte doğru olarak sınıflandırılır. Öğrenme sonunda ağırlıklar:

w1 = 1

w2 = -l

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= wl*xl +w2*x2 = 1*1+ (-1)*0= 1 > Φ Çl= 1 =Bl

2. örnek için: NET= wl*xl +w2*x2 = l*0+(-l)*l= -1= Φ Ç2= 0 = B2

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

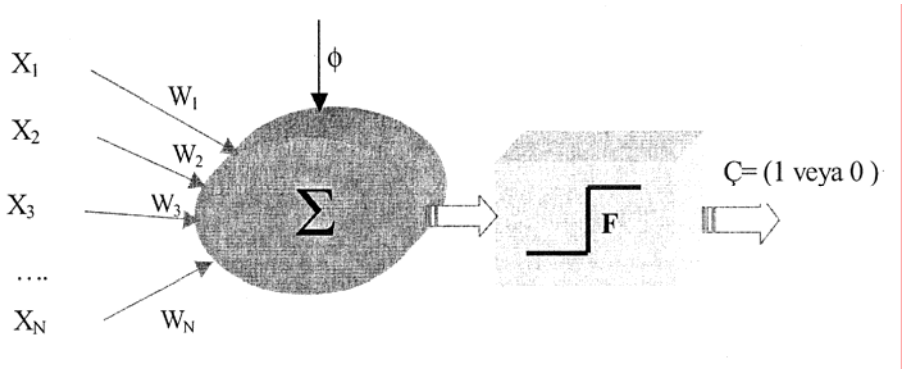
Tek katmanlı algılayıcının en önemli problemi doğrusal olmayan problemlerin çözülmesinde başarılı olmamasıdır. Daha önce belirtildiği gibi, bilinen ZOR problemine çözüm üretemediği için uzun süre yapay sinir ağları araştırmalarının sekteye uğramasına neden olmuştur. Daha sonra çok katmanlı algılayıcıların (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 çözülebilmededir. Fakat eğitim zamanının uzamasına neden olmaktadır.

### c) ADALINE Modeli

ADALİNE widrow ve Hoff [2] tarafından 1959 yılında geliştirilmiş olup adaptif doğrusal eleman (Adaptif Linear Element) ağının kısaltılmış şeklidir. Genel olarak ADALİNE bir proses elemanından (Adaline ünitesi) oluşan bir ağdır.

Bu ağ en küçük ortalamaların karesi (least mean sguare) yöntemine dayanmaktadır. Öğrenme kuralına delta kuralı da denmektedir. Öğrenme kuralı, ağın çıktısının beklenen çıktı değerine göre hatasını enazlayacak şekilde ağın ağırlıklarının değiştirilmesi prensibine dayanır. ADALİNE ünitesinin yapısı Şekil 3.12.'de verilmiştir. ADALINE' nin yapısının tek katmanlı algılayıcıya benzediğine dikkat ediniz. Aradaki fark öğrenme kuralında görülmektedir.




**Şekil 3.12.** Adaline ünitesi

Şekilden de görüldüğü gibi bir ADALİNE ünitesi şu notasyon ile gösterilmektedir: N adet girdiler: XI, X2, X3, ..... XN

Her girdinin ADALİNE elemanı üzerinde etkisinin gösteren ağırlıklar: Wl, W2,W3, ... WN,

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

**Adaline Öğrenme Kuralı:** ADALİNE 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.

ADALİNE ünitesinin net girdisi NET ve çıktısı (Ç) şu şekilde hesaplanmaktadır.

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

$$\text{NET} = \emptyset + X_1 W_1 + X_2 W_2 + X_3 W_3 + \ldots\ldots + X_N W_N$$

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-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.

**Wy = We + α (B-Ç) X**

Diğer bir deyişle her hangi bir t anında,

**Wi(t) = Wi(t-l) + α *E*Xi**

olacaktır. Bu formülde W(t) ağırlıkların t zamanındaki yeni değerlerini, W (t-l) ağırlıkların değişmeden önceki (t-l. zamandaki) değerlerini, a 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.

**Φy= Φe + α (B-Ç)**

Burada, Φy yeni eşik değerini, Φe değiştirilmeden önceki eşik değerini göstermektedir.

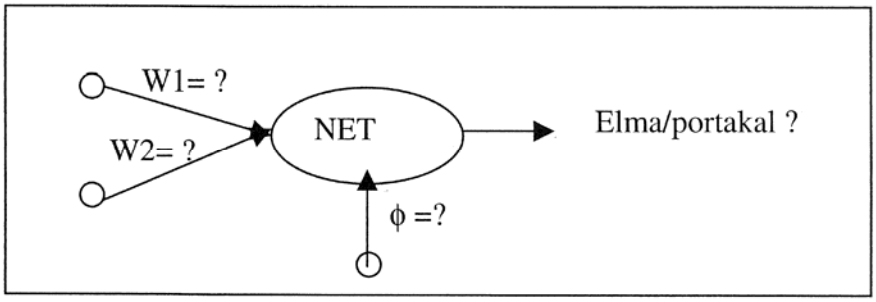
**Örnek:** ADALİNE ünitesinin çalışma prensibini göstermek için şu örnek verilebilir. Bir meyve üreticisi firmanın elma ve portakalların ambara geldiklerinde karışmasının önlemek için bir makine yapmak istediğini varsayınız. Bu amaçla bir yapay sinir ağının kurulabilmesi nasıl mümkün olacaktır?

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 Çl= -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 3.13.).




**Şekil 3.13.** İki girdili bir adaline ünitesi

Problemin çözümü için ağırlık değerleri ( Wl, W2 ve eşik değerinin değerleri başlangıçta rasgele atanmaktadır). Bunun aşağıdaki gibi olduğu varsayılsın.

W1=0.3

W2= 0.2

α=0.5

Φ=0.1.

**1. iterasyon:** Bu iterasyonda öncelikle birinci girdi vektörünün ağa gösterilmesi sonucu ağın çıktısı hesaplanır.

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

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

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 = -l olduğundan ağın ağırlıklarının değiştirilmesi gerekmektedir. Beklenen ve gerçekleşen çıktılar arasındaki hata E ile gösterilirse, E = B-Ç= -l -l = -2 olur.

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

Wy = We + α (B- Ç) X

Wy=[0.3, 0.2]+ 0.5* (-2 ) [l ,0]

Wy=[0.3, 0.2]+ (-l ) [l ,0]

Wy= [0.3 - 1, 0.2 - 0] =[ -0.7, 0.2 ]

Eşik değer ünitesinin ağırlığı da benzer şekilde değiştirilir.

Φy= Φe + α (B-Ç)

Φy = 0.1+ 0.5* (-2) = -0.9

Böylece öğrenmede birinci iterasyon tamamlanmış olur.

**2. 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.

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

NET = -0.7*0+0.2*1-0.9 = – 0.9 < 0 => Ç=-1

Bu örnek için B = 1 olması gerektiğinden ortaya çıkan hata; E = B-Ç = l-(-l)= 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]

Φy = -0.9 + 0.5*(2) = 0.1

şeklinde hesaplanır.

**3. iterasyon:**

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

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

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

**4. iterasyon:**

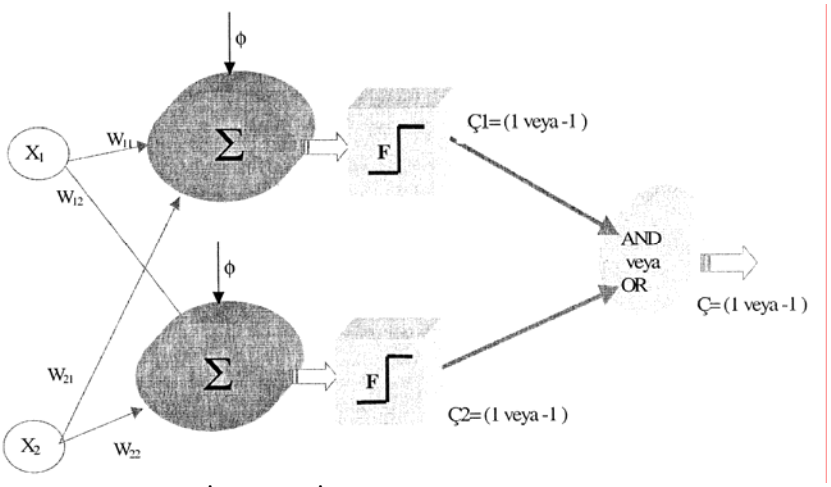
$$NET = [-0.7,\ 1.2] \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 0.1$$

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

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.

### d) MADALINE Modeli

MADALİNE ağları birden fazla ADALİNE ünitesinin bir araya gelerek oluşturdukları ağa verilen isimdir. MADALİNE ile ilgili ayrıntılı açıklamalar Widrow ve Lehr [4] tarafından verilmiştir. MADALİNE 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 3.14.' de iki adaline ünitesinden oluşan bir MADALİNE gösterilmiştir.




**Şekil 3.14.** İki ADALİNE ağından oluşan MADALINE ağı

MADALİNE' nin öğrenme kuralı ADALİNE 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 ADALİNE ünitelerinin 1 değerini üretmesi sonucu MADALINE ağının çıktısı 1 olur. Aksi halde -1 (veya 0) değerini alır. OR sonlandırıcısı olması durumunda ADALİNE ünitelerinin birisinin 1 üretmesi MADALINE ağının çıktısının 1 olması için yeterlidir.

### 3.8. Eğiticili YSA Modeli

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.

#### a) 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 3.3.'de gösterildiği gibidir. Basit algılayıcı ve ADALİNE ile bu problemi çözmek mümkün olmamıştır.

| Girdi 1 | Girdi 2 | Çıktı |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

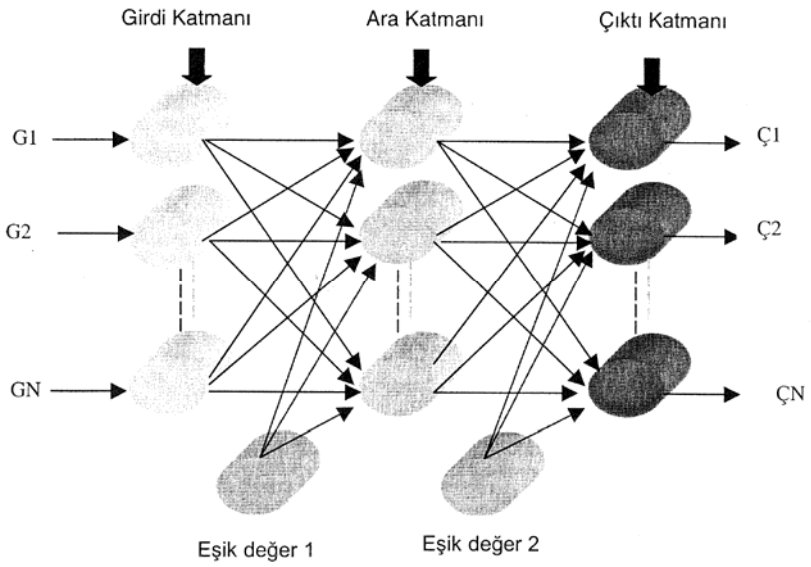
**Tablo 3.3.** XOR problemi

Bu problem, hemen hemen her yapay sinir ağının anlatan her kitapta örnek olarak verilmektedir. Bundan çok yaygın olarak bahsedilmesinin nedeni şöyle açıklanabilir. Minsky özellikle basit algılayıcı (perseptrori) 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 ADALİNE 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.

**b) ÇKA Modelinin Yapısı**

ÇKA ağlarının yapısı Şekil 3.15.'de gösterildiği gibidir. Şekilden de görüldüğü gibi ÇKA ileriye doğru bağlantılı ve 3 katmanda oluşan bir ağdır. Bunlar:




**Şekil 3.15.** ÇKA modeli

- **Girdi katmanı:** Dış dünyadan gelen girdileri (Gl, 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 katma-nmdaki 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 (Çl, 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.

### c) 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.

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

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

**İleri Doğru Hesaplama:** Bu safhada bilgi işleme eğitim setindeki bir örneğin Girdi Katmanından (Gl, 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 $Ç^{i}_{k}$ şu şekilde belirlenir.

$$Ç_k^i = G_k$$

Ara katmandaki her proses elemanı girdi katmanındaki bütün proses elemanlarından gelen bilgileri bağlantı ağırlıklarının (Al, Al...) etkisi ile alır. Önce ara katmandaki proses elemanlarına gelen net girdi ( $NET^{a}_{j}$ ) 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 ($B_j$, ara katmanda bulunan j. elemana bağlanan eşik değer elemanın ağırlığını göstermektedir. Bu eşik değeri ünitesinin çıktısı sabit olup l' 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, (Çl, Ç2,...) bulununca ağın ileri hesaplama işlemi tamamlanmış olur.

**Geriye Doğru Hesaplama:** Ağa sunulan girdi için ağın ürettiği çıktı ağın beklenen çıktıları (#1, J52,...) 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 karekökü 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 $\Delta A^a$ 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 $\lambda$ öğrenme katsayısını, a 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 $\delta_m$ 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.

$$A^a_{jm}(t) = A^a_{jm}(t-1) + \Delta A^a_{jm}(t)$$

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 $\beta^ç$ ile gösterilirse; bu ünitenin çıktısının sabit ve 1 olması nedeni ile değişim miktarı,

$$\Delta\beta^ç_m(t) = \lambda\delta_m + \alpha\Delta\beta^ç_m(t-1)$$

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

$$\beta^ç_m(t) = \beta^ç_m(t-1) + \Delta\beta^ç_m(t)$$

şeklinde hesaplanacaktır.

**<u>Ara katmanlar arası veya ara katman girdi katmanı arasındaki ağırlıkların değiştirilmesi</u>**

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) $\Delta A^i$ ile gösterilirse değişim miktarı;

$$\Delta A^i_{kj}(t) = \lambda\delta^a_j Ç^i_k + \alpha\Delta A^i_{kj}(t-1)$$

olacaktır. Buradaki hkta terimi δa ise şöyle hesaplanacaktır.

$$\delta^a_j = f'(NET)\sum_m \delta_m A^a_{jm}$$

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

$$\delta^a_j = Ç^a_j(1-Ç^a_j)\sum_m \delta_m A^a_{jm}$$

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 $\beta^a$ 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.

**d) ÇKA Ağının Çalışma Prosedürü**

ÇKA ağlarının çalışması şu adımları içermektedir:

**1.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.

**2.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.

**3.Öğrenme parametrelerini belirlenmesi:** Ağın öğrenme katsayısı, proses elemanlarının toplama ve aktivasyon fonksiyonları, momentum katsayısı gibi parametreler bu adımda belirlenmektedir.

**4.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.

**5.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.

**6.Öğrenme sırasında ileri hesaplamaların yapılması:** Yukarıda anlatıldığı şekilde sunulan girdi için ağın çıktı değerleri hesaplanır

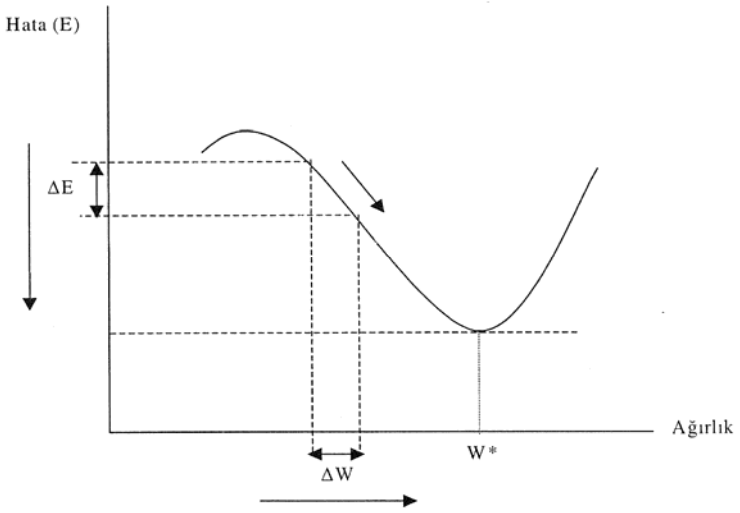
**7.Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması:** Ağın ürettiği hata değerleri bu adımda hesaplanır.

**8.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.

**e) 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 zekâ 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 3.16.'da gösterildiği gibi bir hata uzayının olduğu varsayılsın. Şekildeki W* en az hatanın olduğu ağırlık vektörünü göstermektedir.

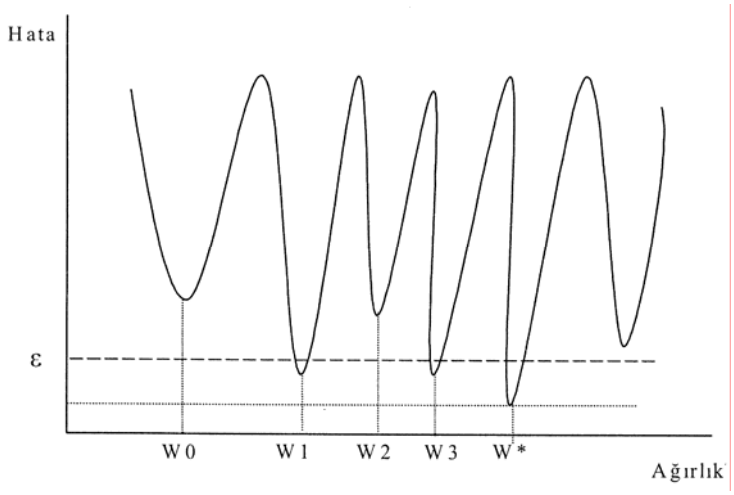



**Şekil 3.16.** Öğrenmenin hata uzayındaki gösterimi

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 3.17. 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) s kadar hatayı kabul etmektedirler (tolerans değeri). Tolerans değeri altındaki her hangi bir noktada olay öğrenilmiş kabul edilmektedir. Şekildeki W0 ve Wl çözümlerinin hataları kabul edilebilir hata düzeyinin üzerinde olduğundan bu çözümler kabul edilemez çözümlerdir. Bunlara **yerel çözümler** denilmektedir. Wl 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 3.17.** Çok boyutlu hata uzayı

Şekil 3.17. 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 Wl ağırlıklarını bulur hatayı daha aşağıya düşüremez. Bu aslında yerel bir

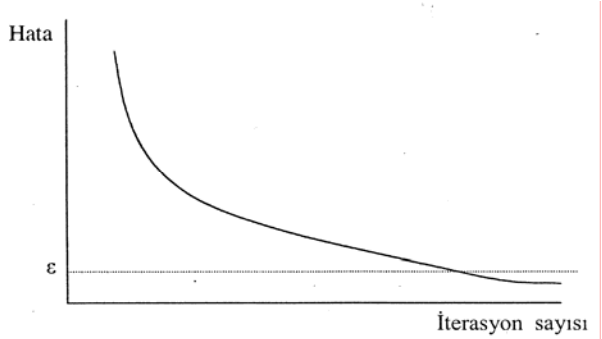
çözümdür. En iyi çözüm değildir. Hata kabul edilebilir düzeye indiğinden yerel en iyi bir çözüm olarak görülebilir. Global çözümün bulunması da mümkün olabilir. Bu tamamen ağın tasarımına, örneklerin niteliğine ve eğitim sürecine bağlıdır.

Bazı durumlarda ağın takıldığı lokal sonuç kabul edilebilir hata düzeyinin üstünde kalabilir (Mesela Şekil 3.18.'deki 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 3.18.** Öğrenen bir ÇKA ağının öğrenme eğrisine örnek

Ağın öğrenmesinin gösterilmesinin en güzel yol hata grafiğini çizmektir. Öğrenen bir ağ için her îterasyonda oluşan hatanın grafiği çizilirse Şekil 3.18.'de gösterildiği şekle benzer bir hata grafiği oluşur. Burada hatanın zaman içerisinde düştüğü görülür.

Belirli bir iterasyondan sonra hatanın daha fazla azalmadığı görülür. Bu ağm 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.

### f) 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 topolojisi 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 ($\lambda$) ve momentumun ($\alpha$) katsayılarının ise şu şekilde belirlendiği varsayılsın,

$\lambda = 0.5$

$\alpha = 0.8$

sse = 0.03

**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 0.75262l]

Eşik değeri ağırlıkları şöyle olsun.

$\beta^a$= [0.341332 -0.115223]

$\beta^c$ =[-0.993423]

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

Birinci örnek G1=0, G2=0 ve B=0 olarak belirlenmiştir.

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

Netl = (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= (l*-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.

$$\beta_{1}^{i}(t) = 0.341232 - 0.017095 = 0.3242038$$

$$\beta_{2}^{i}(t) = 0.115223 - 0.081325 = -0.0350905$$

Birinci iterasyon bittikten sonra ikinci iterasyon başlayacaktır. Bu kez ikinci örnek ağa gösterilir. G1=0, G2=l ve B=l olacaktır. Yukarıdaki işlemler aynı şekilde tekrar edilir. Bu iterasyonlar bütün çıktılar doğru cevap verinceye kadar devam etmelidir.

Bu ağırlıklar ile girdiler ağa tekrar gösterildiğinde o zaman Tablo 3.4'de gösterilen sonuçlar elde edilir. Bu sonuçlar ağın problemi çok düşük hatalar ile çözebilecek şekilde öğrendiğini göstermektedir.

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

**Tablo 3.4.** XOR problemini öğrendikten sonra ağın ürettiği çözümler ve hata oranları

### g) 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.

### h) 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.

**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.

**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.

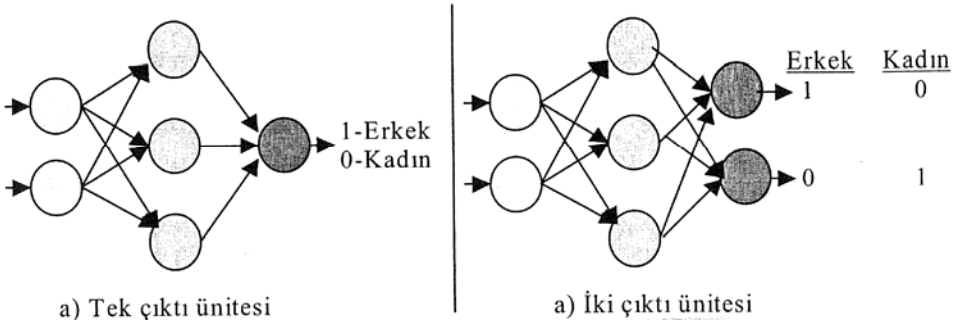
**Girdi Değerlerinin Nümerik Gösterimi:** Ağa gösterilen girdi değerlerini ağın anlayabilmesi için nümerik olma zorunluluğu problemin girdilerinin nümerik gösterimini gerektirmektedir. Bu ise her zaman kolay olmamakta ve problem tasarımcısını zor durumda bırakabilmektedir. Çünkü bu güne kadar geliştirilmiş 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.

**Tablo 3.5.** Örneklerdeki sayısal ve alfa nümerik 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 |

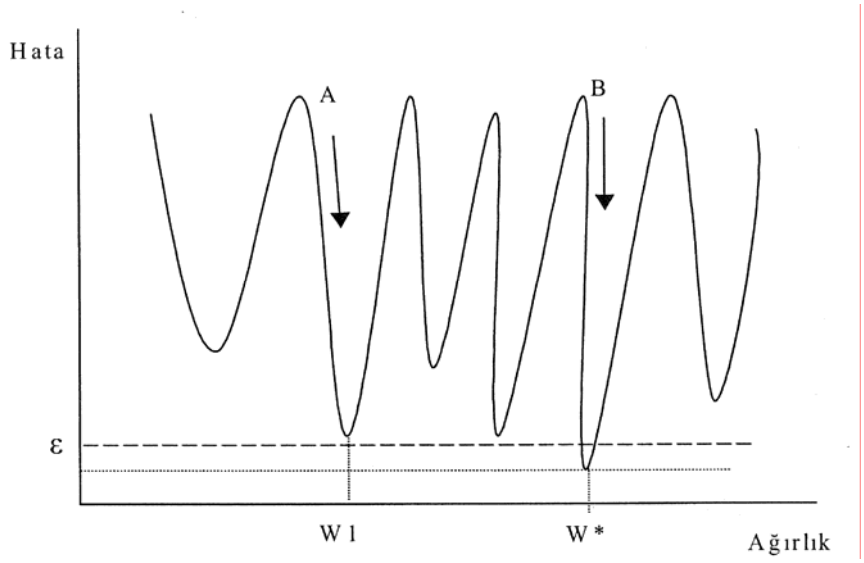
Görüldüğü gibi bazı değerler tamamen alfa nümerik değerlerdir. Bu olayı ÇKA ağına öğretmek için bunları nümerik değerlerle gösterilmek zorundadır. Bu değişik şekillerde yapılabilir. Mesela B faktörü için 1 tam değerini O'da yarım değerini gösterebilir. Benzer şekilde Tam için 1, yarım içinde 2 değerleri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

**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 alasında paylaştırılmış olacaktır. Şekil 3.19. bu iki durumu göstermektedir. Bunlardan hangisinin daha iyi olduğunu söylemek mümkün değildir. Bu eğitimin başlaması ile görülebilir.




**Şekil 3.19.** Çıktıların sayısal gösterimi

**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. Diğerleri ile birlikte düşünmek gerekir. Şekil 3.20. başlangıç noktalarının önemini göstermektedir.




**Şekil 3.20.** ÇKÂ ağlarında başlangıç noktasının etkisi

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

**Öğrenme Katsayısı ve Momentum Katsayılarının Belirlenmesi:** 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.

Benzer şekilde momentûm katsayısı da öğrenmenin performansını etkiler. Momentûm 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.

**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.

**Ağırlıkların Değiştirilmesi Zamanı:** Ağırlıkların değiştirilmesi öğrenme kuralına göre yapılmaktadır. 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. îstenen 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.
3. *Bütün örnek seti gösterildiğinde (epoch based leanıing):* 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.

**Örneklerin Değerlerinin Ölçeklendirilmesi (Scaling):** ÇKA ağlarında girdi ve çıktıların ölçeklendirilmeside ağın performansını yakından etkilemektedir. Çünkü ölçeklendirme örneklerin değerlerinin dağılımını düzenli hale getirmektedir.

- **Girdilerin Ölçeklendirilmesi:** 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. 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.

Burada önemli olan hangi yöntem kullanıldığından çok girdiler içindeki olumsuz etkileri önleyecek şekilde ölçeklendirme yapmaktır. Bu konuda bir standart koymak doğru olmaz.

- **Çıktıların Ölçeklendirilmesi:** Çıktıların ölçeklendirilmesi de yukarıda girdilerin ölçeklendirilmesi için anlatılan yöntemlerin birisi ile yapılabilir. 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. 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.

**Tablo 3.6.** 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 |

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.

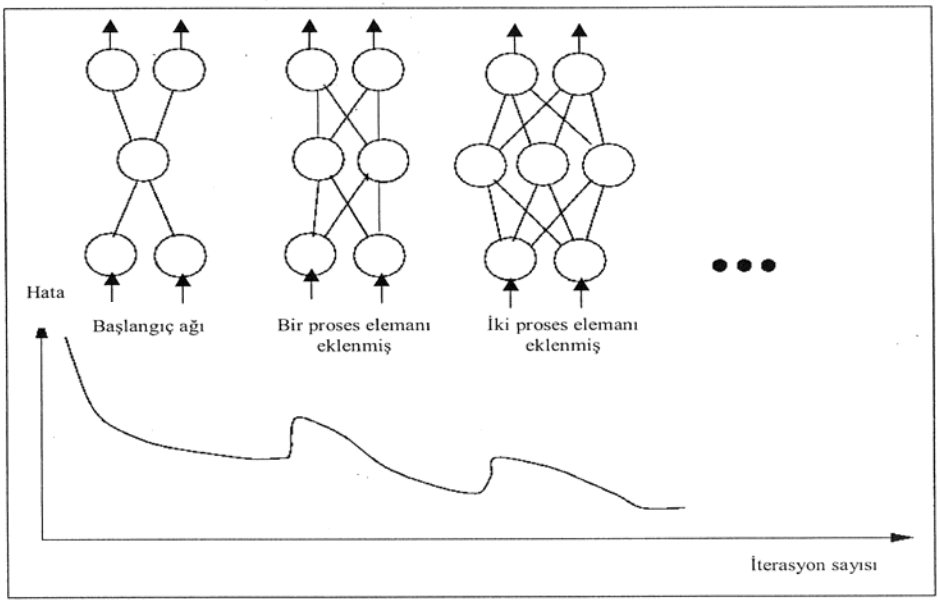
**Durdurma Kriterleri:** ÇKA modelinde ağın eğitilmesi kadar gereğinden fazla eğitilmemesi de önemlidir. Pratikte, genel olarak iki türlü durdurma kriteri kullanılmaktadır.

- *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. Öğrenme performansı %100 olmaz da %98-99 gibi bir düzeyde kalabilir. 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:* Hatanın hangi değerlerin altına düşebileceğinin kestirilemediği (yani kabul edilebilir hatanın belirlenemediği) durumlarda bu tür bir durdurma kriteri uygulanabilir.

**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. Tasarımcılar kendi tecrübelerine dayanarak bunları belirler. Bazı durumlarda başlangıçta bir ağ oluşturulup zaman içinde büyütülerek veya küçültülerek istenen ağa ulaşılır. Aşağıda bu durum açıklanmıştır.

**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 3.21.'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 3.21.** Büyüyen ağlar

İşlemler 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.

### 3.9. Eğiticisiz YSA Modeli

Bu bölümde ise öğretmensiz öğrenmeye dayalı ART (Adaptif Rezonans Teori (Art) Ağları) ağlarına değinilecektir. Bu konuyu iyi anlamak için öncelikle hafıza ve bilgilerin hafızada saklanması kavramları açıklanacaktır.

#### a) 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.

**b) ART Ağları**

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

1. **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 duyarsızlıklarım anlatmakta güçlük çekmektedirler. Hâlbuki bu insanlarında bazıları minibüs yoluna yakın yerde oturmakta ve korna sesinden etkilenmemektedirler.
2. **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.
3. **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 ağlan 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 birçok 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.

### c) 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* şu şekilde özetlemektedir:

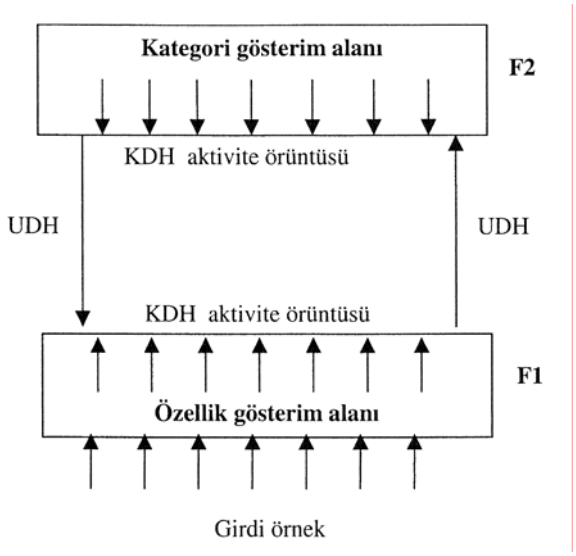
- ART ağları gerçek zamanlı olarak oldukça hızlı ve kararlı bir şekilde öğrenme yeteneklerine sahiptirler. Bu yetenek birçok ağda yoktur. ART ağları bu özellikleri ile gerçek zamanlı kullanılabilen donanımla da desteklenerek gerçek zamanlı öğrenebilen bilgisayarların oluşmasına yardımcı olmaktadırlar.

- 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 ya da 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. 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. Hâlbuki 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 , ARTMAP, Fuzzy ART 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.

### d) ART Ağlarının Yapısı

Adaptif Rezonans Teorisi (ART) ağları genel olarak iki katmandan oluşmaktadır. Bu katmanlar Fİ ve F2 olarak isimlendirilmiştir. Fİ katmanı girdinin özelliklerini gösterirken F2 katmanı kategorileri (ayrıştırılmış sınıfları) göstermektedir. Bu iki katman birbirlerine UDH ile bağlanmaktadırlar. Girdi bilgileri Fİ katmanından alınır ve sınıflandırma ise F2 katmanında yapılır. Modelin genel yapısı Şekil 3.22. 'de verilmektedir.




**Şekil 3.22.** ART ağının genel yapısı

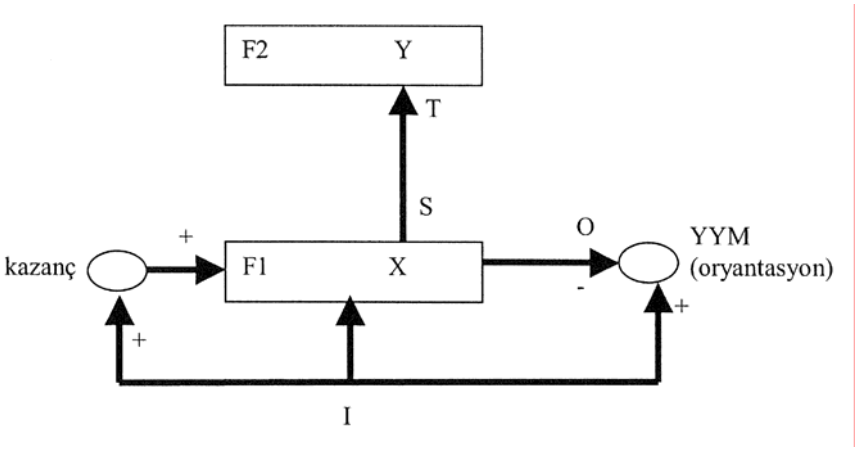
ART ağlarında girdiler direkt olarak sınıflandırılmazlar. Öncelikle girdilerin özellikleri incelenerek Fİ katmanının aktivasyonu belirlenir. UDH'da ki bağlantı değerleri ile gelen bilgiler kategorilere ayrılarak F2 katmanına gönderilir. F2 katmanındaki sınıflandırma ile F1 katmanından gelen sınıflandırma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

### e) ART Ağlarının Çalışma Prensibi

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

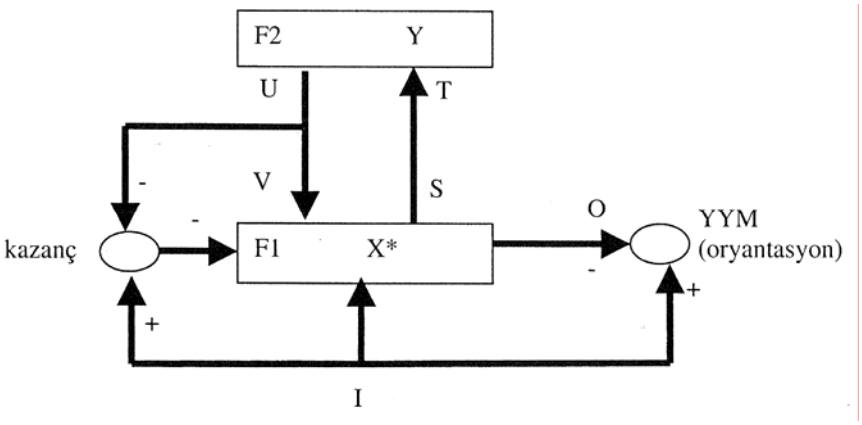
- Aşağıdan yukarı (F1 den F2'ye) bilgi işleme
- Yukarıdan aşağı (F2 den F1 'e) bilgi işleme

ART ağlarında aşağıdan yukarı bilgi işleme prensibi Şekil 3.23.'de gösterilmiştir. Şekilden de görüldüğü gibi, bir girdi örüntüsü (I) ağa gösterilir. Bu örüntü hem Fİ 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.



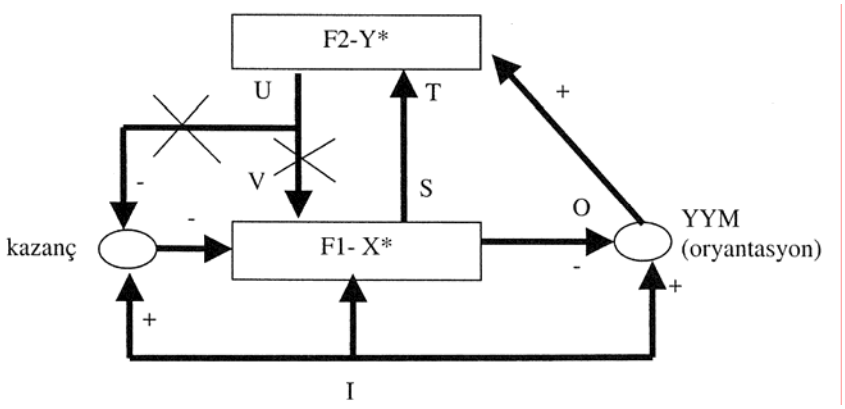

**Şekil 3.23.** ART ağında çıktı oluşturma süreci (aşağıdan yukarı)

Yukarıdan aşağı bilgi işleme de, benzer şekilde, Şekil 3.24'te 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 3.24.** 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 3.25'te gösterildiği gibi girilen I örüntüsü için doğru sınıfı gösteren Y* çıktısı üretilmektedir.



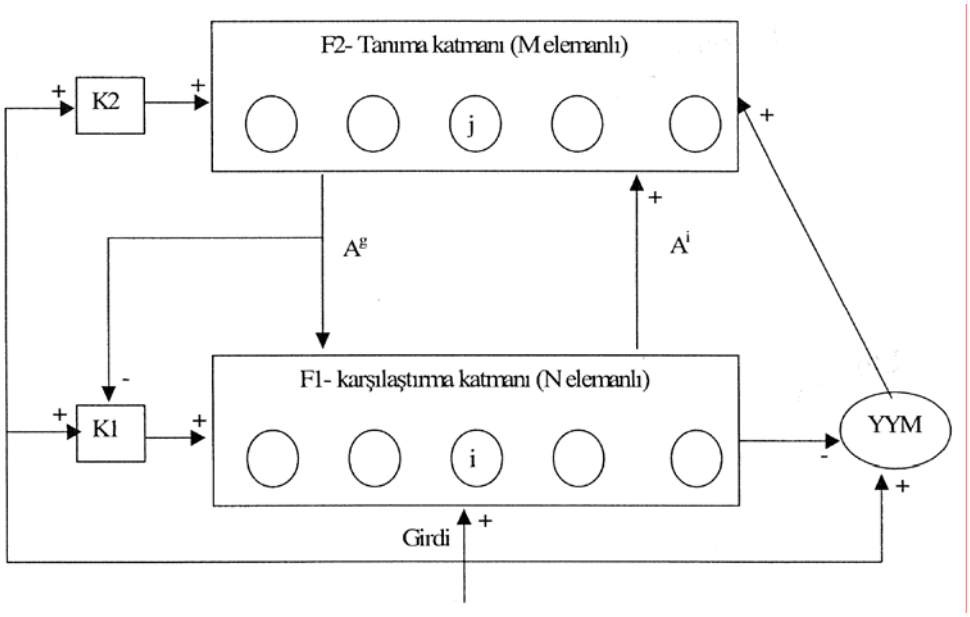

**Şekil 3.25.** ART ağında yeni bir sınıf oluşturma

Eğer üretilen V şablon örüntüsü ile girdi örüntüsü eşleşir işe 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.

### f) 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 katmam, 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 Fİ 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 3.26.'da verilmiştir:




**Şekil 3.26.** 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.

**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≥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:

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

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.

$$A_{ij}^{i} = \frac{1}{1+n}$$

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

Benzerlik katsayısı p 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 (kategori) gösterim vektörü denmektedir. Bunun k. proses elemanı olduğu varsayılırsa kazanan elemanın çıktısı,

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

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.

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

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, S2/S1 ≥ p ise o zaman iki vektör birbirinin benzeri kabul edilir. Yani kategori gösterim vektörü girdi vektörünü temsil edebiliyor demektir. Bu durumda ağırlıklar şu şekilde değiştirilir.

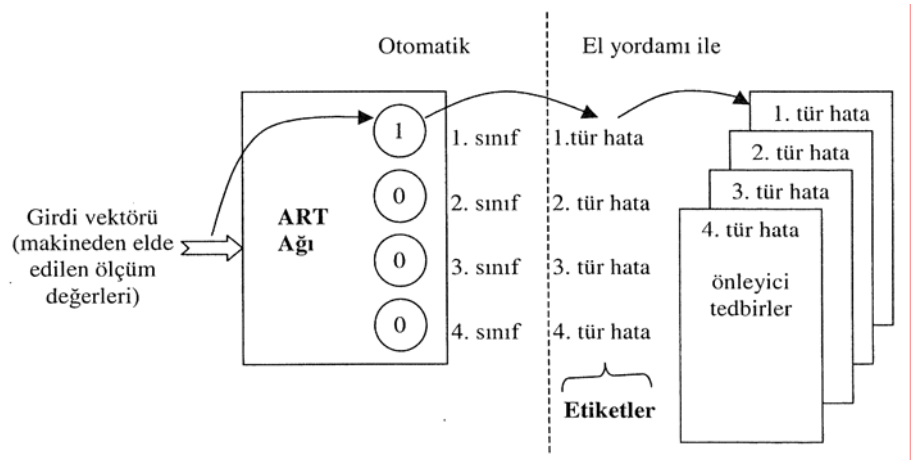
$$A_{jk}^g(t+1) = A_{jk}^g(t) x_i$$

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

Eğer; S2/S1 < p 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 4üneü 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.

**g) 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 3.27. bu durumu göstermektedir.




**Şekil 3.27.** 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.

**ÖRNEK:** ART1 - Bir Grup Teknolojisine Dayalı İmalât Uygulaması

Daha önce anlatılan ağlarda olduğu gibi ART ağları da endüstriyel problemlere çözümler üretebilmektedir. Aşağıda Grup teknolojisinde gerçekleştirilen bir uygulama anlatılacaktır.

- **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. 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. 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 kat ederek üretim hattından çıkması oluşturulacak olan makine hücrelerinin doğru oluşturulmasına bağlıdır.

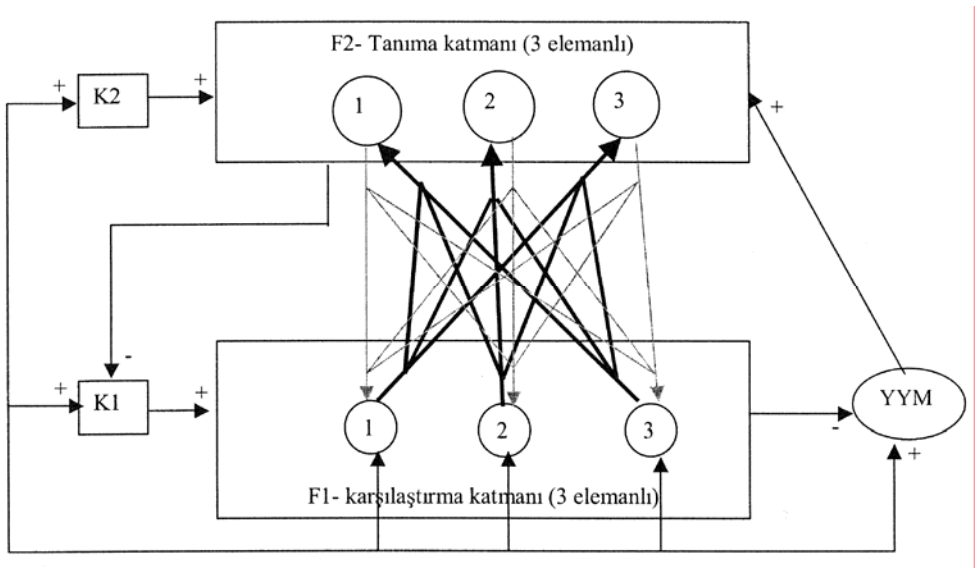
  Aşağıda ART1 ağlarının grup teknolojisinde kullanılmasına yönelik geliştirilmiş bir model açıklanacaktır. Geliştirilen ART1 ağından makine/parça arasındaki ilişkileri gösteren örnekleri alarak, dış dünyadan herhangi bir yardım almadan makine/parça sınıflandırmasını otomatik olarak yapması istenmektedir. 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.

- **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 is& hem birinci hem de üçüncü makinede işlenebilmektedir. Problemde makinelerin aynı sınıfın üyesi olup olmadıklarını belirlemeye yarayan benzerlik katsayısı p=0.8 olarak alınmıştır. Yani %80 civarında bir benzerlik aynı grubun elemanı olmak için yeterli görülmektedir. Oluşturulan ART1 ağı Şekil 3.27' de gösterilmiştir. Şekildeki F1 ve F2 katmanları arasındaki koyu renkli bağlantılar aşağıdan yukarıya açık renkli bağlantılar ise yukarıdan aşağı ağırlıkları göstermektedir.




**Şekil 3.27.** Oluşturulan ART 1 ağının gösterimi

- **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 p=0.8

F2 ve F1 katmanları arasındaki geriye doğru ağırlıkların bütün değerleri başlangıçta 1 değerini alır. Yani,

$$A_{ji}^{g} = 1 \qquad j: 1,2,3 \ldots M \qquad i: 1,2,3 \ldots N$$

$A^g$ 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.

$$A_{ij}^{i} = \frac{1}{1+N} = \frac{1}{1+3} = 0.25 \qquad j: 1,2,3 \ldots M \qquad i: 1,2,3 \ldots N$$

**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). $A_2^g$ = [l 1 l ] olduğundan benzerlik katsayısı ile mukayese yapılırsa;

$$s1 = |X| = 2$$

$$s2 = |A_2^g . X| = 2$$

$$\frac{S2}{S1} = \frac{2}{2} = 1 \geq \rho$$

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:** 3. iterasyonda işlemlere yine 2. adımdan başlanarak adımların hepsi 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.

$$s1 = |X| = 1$$
$$s2 = |A_2^{g}.X| = 1$$
$$\frac{S2}{S1} = \frac{1}{1} = 1 \geq \rho$$

olduğundan bu makinede yine 1. sınıfın bir makinesi olarak sınıflandırılır. Birinci iterasyonda $A_1^9$ = [l 0 l] 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.