Yapay Sinir Ağları
(Artificial Neural Networks)

JEOLOJİ MÜHENDİSLİĞİ A.B.D.
ESNEK HESAPLAMA YÖNTEMLERİ-II

DOÇ. DR. ERSAN KABALCI

Yapay Sinir Ağları

- Tarihçe
- Biyolojik nöron
- Matematiksel nöron modeli
- Ağ yapıları
- Yapay sinir ağlarında öğrenme
- Yapay sinir ağı tasarımı ve nöral hesaplama

Yapay Sinir Ağları Tarihçe

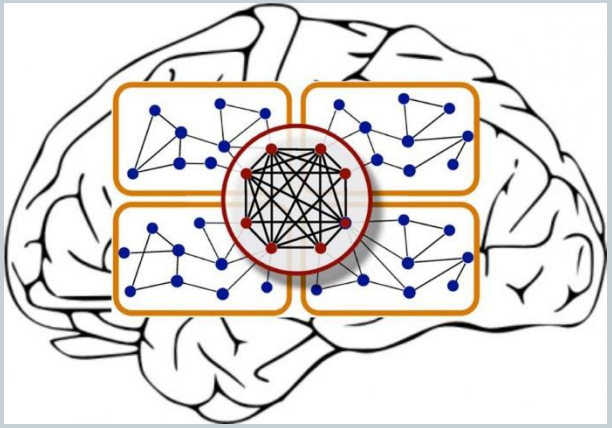
- İlk yapay sinir ağı modeli 1943 yılında, bir sinir hekimi olan Warren McCulloch ile bir matematikçi olan Walter Pitts tarafından gerçekleştirilmiştir.

- McCulloch ve Pitts, insan beyninin hesaplama yeteneğinden esinlenerek, elektrik devreleriyle basit bir sinir ağı modellemişlerdir.

- Genel olarak insan beyninin ya da merkezi sinir sisteminin çalışma prensiplerinin taklit eden bilgi işleme sistemleridir

Yapay Sinir Ağları Tarihçe

- Yapay sinir ağları (YSA), insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri, herhangi bir yardım almadan otomatik olarak gerçekleştirebilmek amacı ile geliştirilen bilgisayar sistemleridir.



Yapay Sinir Ağları Tarihçe

- Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır.

- Bu nedenle, bu konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamış, daha sonraları bilgisayar sistemlerinin gelişimine paralel olarak bir çok alanda kullanılır hale gelmiştir.

Biyolojik Sinir Sistemi

- Biyolojik sinir sistemi, merkezinde sürekli olarak bilgiyi alan, yorumlayan ve uygun bir karar üreten beynin (merkezi sinir ağı) bulunduğu 3 katmanlı bir sistem olarak açıklanır.

Biyolojik sinir sisteminin blok gösterimi

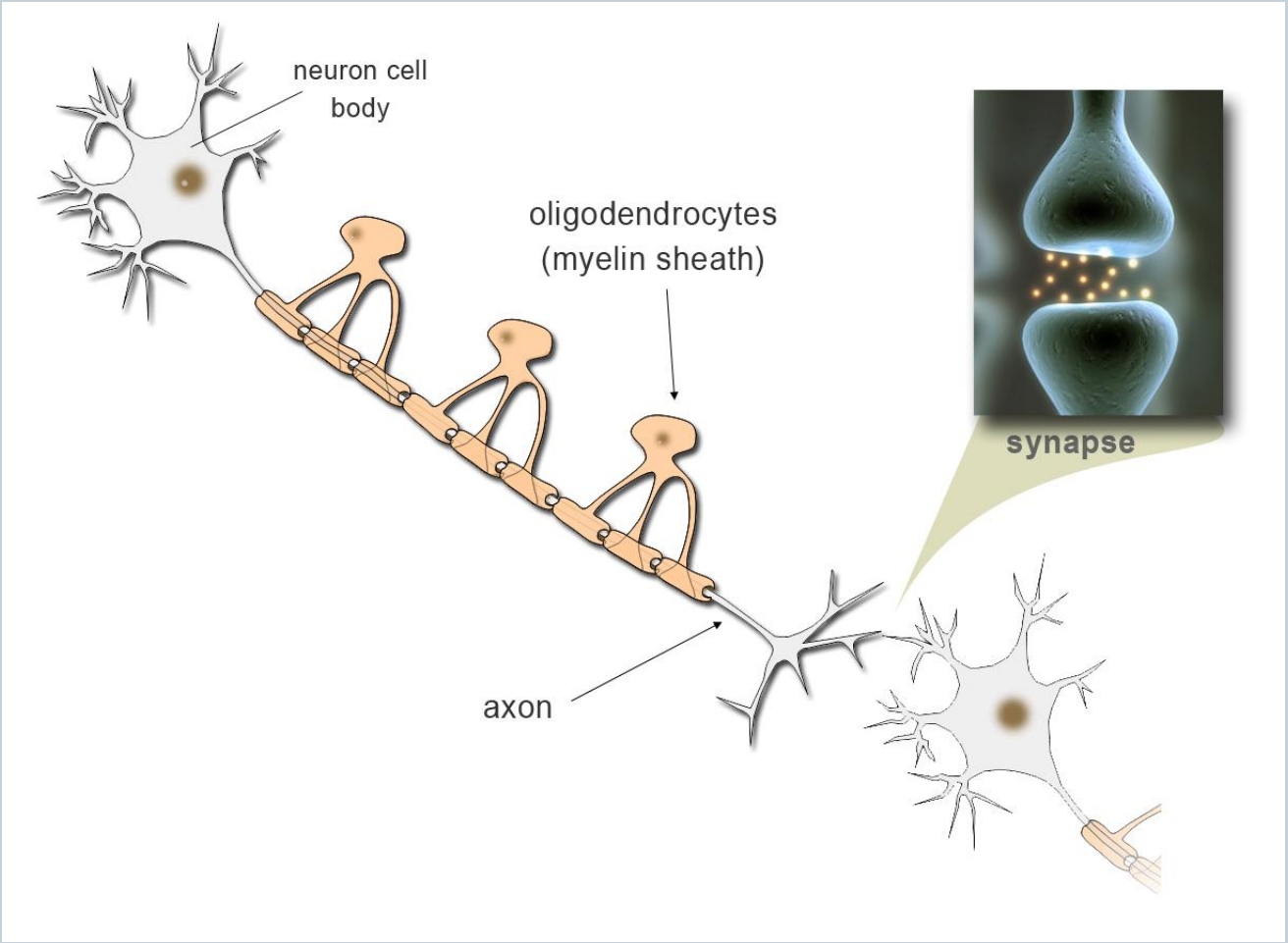
Biyolojik Sinir Sistemi

- Alıcı sinirler (receptor) organizma içerisinden ya da dış ortamlardan algıladıkları uyarıları, beyine bilgi ileten elektriksel sinyallere dönüştürür.

- Tepki sinirleri (effector) ise, beyinin ürettiği elektriksel darbeleri organizma çıktısı olarak uygun tepkilere dönüştürür.

- Merkezi sinir ağında bilgiler, alıcı ve tepki sinirleri arasında ileri ve geri besleme yönünde değerlendirilerek uygun tepkiler üretilir. Bu yönüyle biyolojik sinir sistemi, kapalı çevrim denetim sisteminin karakteristiklerini taşır.

Biyolojik Sinir Sistemi

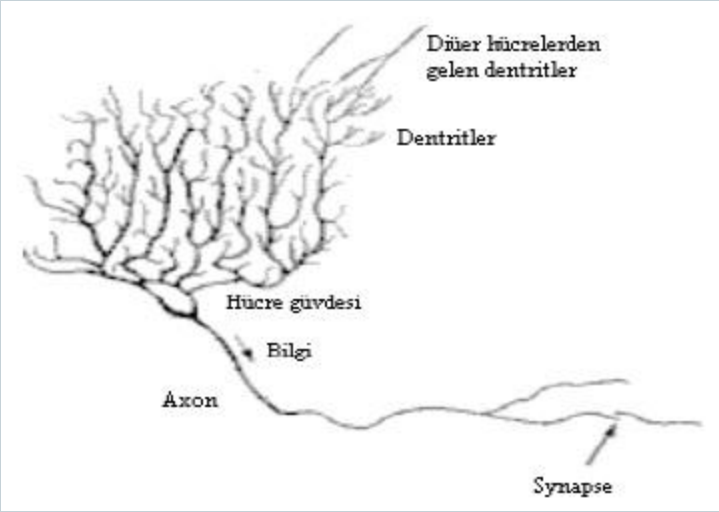


Biyolojik Sinir Sistemi

- Merkezi sinir sisteminin temel işlem elemanı, sinir hücresidir (nöron) ve insan beyninde yaklaşık 10 milyar sinir hücresi olduğu tahmin edilmektedir.
- Sinir hücresi; hücre gövdesi, dendriteler ve aksonlar olmak üzere 3 bileşenden meydana gelir.
- Dendriteler, diğer hücrelerden aldığı bilgileri hücre gövdesine bir ağaç yapısı şeklinde ince yollarla iletir.

Biyolojik Sinir Sistemi

- Aksonlar ise elektriksel darbeler seklindeki bilgiyi hücreden dışarı taşıyan daha uzun bir yoldur.
- Aksonların bitimi, ince yollara ayrılabilir ve bu yollar, diğer hücreler için dendritleri oluşturur. Şekilde görüldüğü gibi axon-dendrite bağlantı elemanı synapse olarak söylenir.



Yapay Sinir Ağları

Biyolojik Sinir sistemi		*Yapay Sinir Sistemi*
Nöron	→	İşlemci eleman
Dentrit	→	Toplama fonksiyonu
Hücre gövdesi	→	Transfer fonksiyonu
Aksonlar	→	Yapan nöron çıkışı
Sinapslar	→	Ağırlıklar

Yapay Sinir Ağları

- YSA'lar; ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir.

- Bir işlem birimi, diğer nöronlardan sinyalleri alır; bunları birleştirir, dönüştürür ve sayısal bir sonuç ortaya çıkartır.

- Genelde, işlem birimleri kabaca gerçek nöronlara karşılık gelirler ve bir ağ içinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır.

Yapay Sinir Ağları

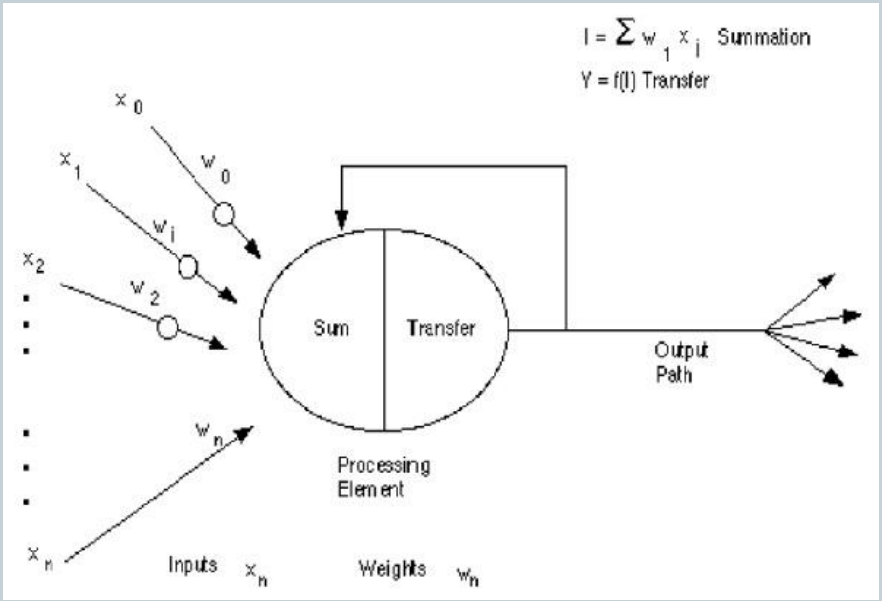
- YSA, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar seklinde düzenlenir.

- Donanım olarak elektronik devrelerle yada bilgisayarlarda yazılım olarak gerçeklenebilir.

- Beynin bilgi isleme yöntemine uygun olarak YSA, bir öğrenme sürecinden sonra bilgiyi toplama, hücreler arasındaki bağlantı ağırlıkları ile bu bilgiyi saklama ve genelleme yeteneğine sahip paralel dağılmış bir işlemcidir.

Yapay Sinir Ağları

- Öğrenme süreci, istenen amaca ulaşmak için YSA ağırlıklarının yenilenmesini sağlayan Öğrenme algoritmalarını ihtiva eder. Şekilde yapay sinir ağı modeli gösterilmektedir.



Yapay Sinir Ağının Özellikleri

- Yukarıda verilen açıklamalardan, YSA'nın hesaplama ve bilgi işleme gücünü, paralel dağılmış yapısından, öğrenebilme ve genelleme yeteneğinden aldığı söylenebilir.

- Genelleme, eğitim ya da öğrenme sürecinde karşılaşılmayan girişler için de YSA'nın uygun tepkileri üretmesi olarak tanımlanır.

- Bu üstün özellikleri, YSA'nın karmaşık problemleri çözebilme yeteneğini gösterir. Günümüzde birçok bilim alanında YSA, aşağıdaki özellikleri nedeniyle etkin olmuş ve uygulama yeri bulmuştur.

Yapay Sinir Ağının Özellikleri

- **Doğrusal** Olmama
  - YSA'nın temel işlem elemanı olan hücre doğrusal değildir. Dolayısıyla hücrelerin birleşmesinden meydana gelen YSA da doğrusal değildir ve bu özellik bütün ağa yayılmış durumdadır. Bu özelliği ile YSA, doğrusal olmayan karmaşık problemlerin çözümünde en önemli araç olmuştur.

- **Öğrenme**
  - YSA'nın arzu edilen davranışı gösterebilmesi için amaca uygun olarak ayarlanması gerekir. Bu, hücreler arasında doğru bağlantıların yapılması ve bağlantıların uygun ağırlıklara sahip olması gerektiğini ifade eder.

  - YSA' nın karmaşık yapısı nedeniyle bağlantılar ve ağırlıklar önceden ayarlı olarak verilemez yada tasarlanamaz. Bu nedenle YSA, istenen davranışı gösterecek şekilde ilgilendiği problemden aldığı eğitim örneklerini kullanarak problemi öğrenmelidir.

Yapay Sinir Ağının Özellikleri

Genelleme

- YSA, ilgilendiği problemi öğrendikten sonra eğitim sırasında karşılaşmadığı test örnekleri için de arzu edilen tepkiyi üretebilir. Örneğin, karakter tanıma amacıyla eğitilmiş bir YSA, bozuk karakter girişlerinde de doğru karakterleri verebilir yada bir sistemin eğitilmiş YSA modeli, eğitim sürecinde verilmeyen giriş sinyalleri için de sistemle aynı davranışı gösterilebilir.

Uyarlanabilirlik

- YSA, ilgilendiği problemdeki değişikliklere göre ağırlıklarını ayarlar. Yani, belirli bir problemi çözmek amacıyla eğitilen YSA, problemdeki değişimlere göre tekrar eğitilebilir, değişimler devamlı ise gerçek zamanda da eğitime devam edilebilir.
- Bu özelliği ile YSA, uyarlamalı örnek tanıma, sinyal isleme, sistem tanılama ve denetim gibi alanlarda etkin olarak kullanılır.

Yapay Sinir Ağının Özellikleri

Hata Toleransı

- YSA, çok sayıda hücrenin çeşitli şekillerde bağlanmasından oluştuğundan paralel dağılmış bir yapıya sahiptir ve ağın sahip olduğu bilgi, ağdaki bütün bağlantılar üzerine dağılmış durumdadır.
- Bu nedenle, eğitilmiş bir YSA'nın bazı bağlantılarının hatta bazı hücrelerinin etkisiz hale gelmesi, ağın doğru bilgi üretmesini önemli ölçüde etkilemez. Bu nedenle, geleneksel yöntemlere göre hatayı tolere etme yetenekleri son derece yüksektir.

Donanım ve Hız

- YSA, paralel yapısı nedeniyle büyük ölçekli entegre devre (VLSI) teknolojisi ile gerçeklenebilir. Bu özellik, YSA'nın hızlı bilgi işleme yeteneğini artırır ve gerçek zamanlı uygulamalarda arzu edilir.

Yapay Sinir Ağının Özellikleri

Hata Toleransı

- YSA, çok sayıda hücrenin çeşitli şekillerde bağlanmasından oluştuğundan paralel dağılmış bir yapıya sahiptir ve ağın sahip olduğu bilgi, ağdaki bütün bağlantılar üzerine dağılmış durumdadır.
- Bu nedenle, eğitilmiş bir YSA'nın bazı bağlantılarının hatta bazı hücrelerinin etkisiz hale gelmesi, ağın doğru bilgi üretmesini önemli ölçüde etkilemez. Bu nedenle, geleneksel yöntemlere göre hatayı tolere etme yetenekleri son derece yüksektir.

Donanım ve Hız

- YSA, paralel yapısı nedeniyle büyük ölçekli entegre devre (VLSI) teknolojisi ile gerçeklenebilir. Bu özellik, YSA'nın hızlı bilgi işleme yeteneğini artırır ve gerçek zamanlı uygulamalarda arzu edilir.

Yapay Sinir Ağ Modeli

- Tek yönlü işaret kanalları ile birbirine bağlanan işlemsel elemanlardan oluşur.
- Çıkış işareti tek olup kopyalanabilir.
- Daha önce öğrendiği bilgiyi eksik veya bozuk giriş verildiği zaman bile yeniden üretebilir.
- Lineer olmayan bir karakteristiğe sahip olmaları nedeniyle gerçek dünya problemlerine daha doğru çözümler getirebilirler.

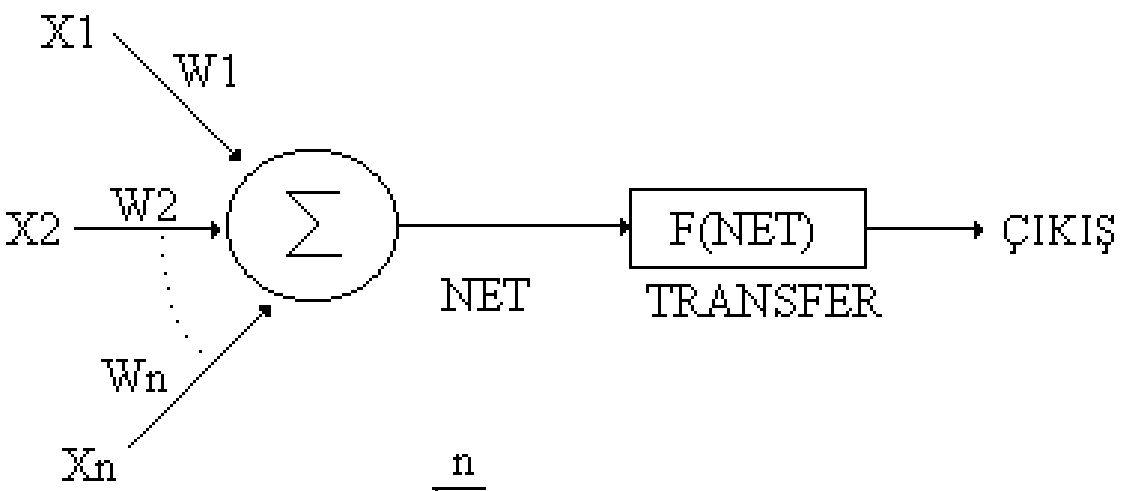
Statik Hücrenin Matematiksel Modeli

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

- Burada; W- hücrenin ağırlıklar matrisini, x- hücrenin giriş vektörünü, v- hücrenin net girişini, y- hücre çıkışını ve $\varphi(v)$- hücrenin aktivasyon fonksiyonunu göstermektedir.

- Denklem den, **x** giriş vektörünün bileşenlerinin dış (geri beslemesiz) girişler olması durumunda hücrenin doğrusal olmayan statik bir işlevi gerçekleştireceği görülmektedir.

Yapay Sinir Ağ Modelinin Birimleri



$$NET = \sum_{i=1}^{n} X_i . W_i$$

ÇIKIŞ = F(NET)
W_i = AĞIRLIKLAR
X_n = GİRİŞLER

Yapay Sinir Ağ Modelinin Birimleri

- İşlem birimleri (nöronlar)
- Aktivasyon durumu
- Birimler arası bağlantı ağırlıkları
- Propagasyon kuralı
- Aktivasyon fonksiyonu
- Offset
- Bilgi toplama metodu
- Çevre

YSA Tipleri

- İleri Beslemeli Yapay Sinir Ağları (İBYSA)
- Geri Beslemeli Yapay Sinir Ağları (GBYSA)
- Çok Katmanlı Yapay Sinir Ağları

İleri Beslemeli Yapay Sinir Ağları

- İleri beslemeli YSA'da, hücreler katmanlar şeklinde düzenlenir ve bir katmandaki hücrelerin çıkışları bir sonraki katmana ağırlıklar üzerinden giriş olarak verilir.

- Giriş katmanı, dış ortamlardan aldığı bilgileri hiçbir değişikliğe uğratmadan orta (gizli) katmandaki hücrelere iletir.

- Bilgi, orta ve çıkış katmanında islenerek ağ çıkışı belirlenir. Bu yapısı ile ileri beslemeli ağlar doğrusal olmayan statik bir işlevi gerçekleştirir.

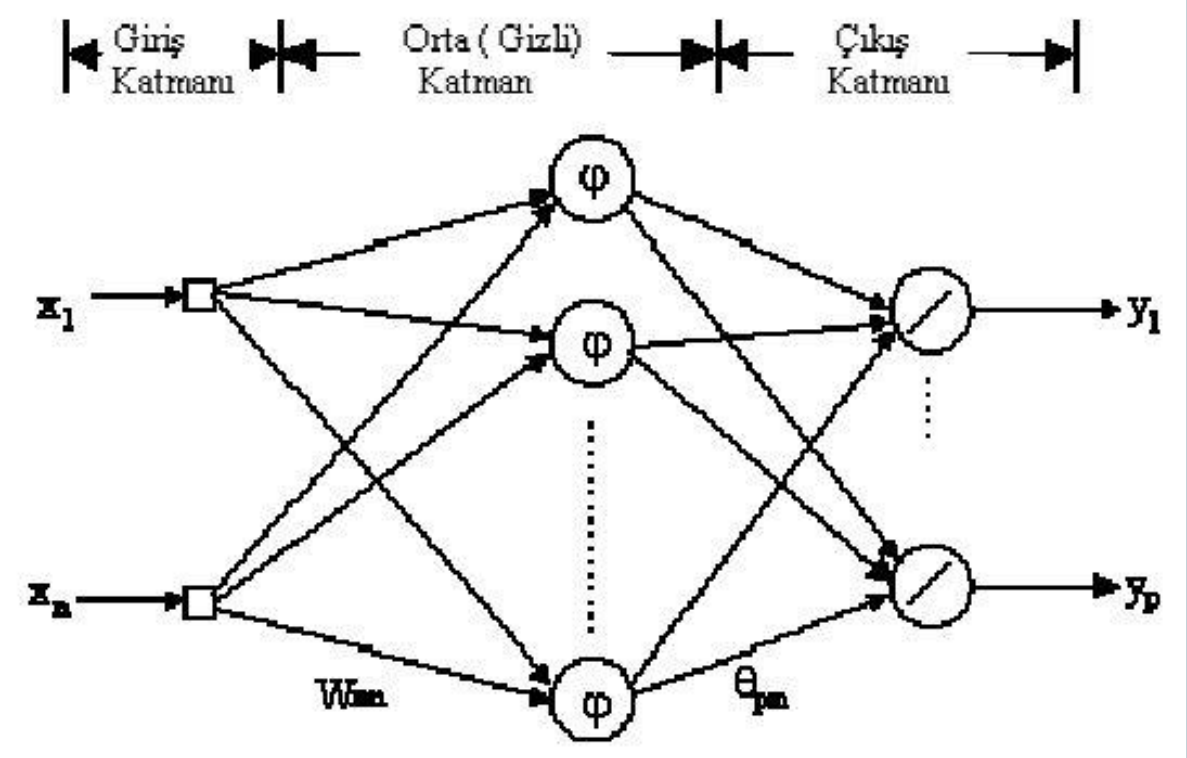
İleri Beslemeli Yapay Sinir Ağları

- İleri beslemeli 3 katmanlı YSA' nın, orta katmanında yeterli sayıda hücre olmak kaydıyla, herhangi bir sürekli fonksiyonu istenilen doğrulukta yaklaştırabileceği gösterilmiştir.

- En çok bilinen geriye yayılım öğrenme algoritması, bu tip YSAların eğitiminde etkin olarak kullanılmakta ve bazen bu ağlara geriye yayılım ağları da denmektedir.

- Şekil de giriş, orta ve çıkış katmanı olmak üzere 3 katmanlı ileri beslemeli YSA yapısı verilmiştir.

İleri Beslemeli Yapay Sinir Ağları



3 katmanlı ileri beslemeli YSA

İleri Beslemeli Yapay Sinir Ağları

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

Geri Beslemeli Yapay Sinir Ağları

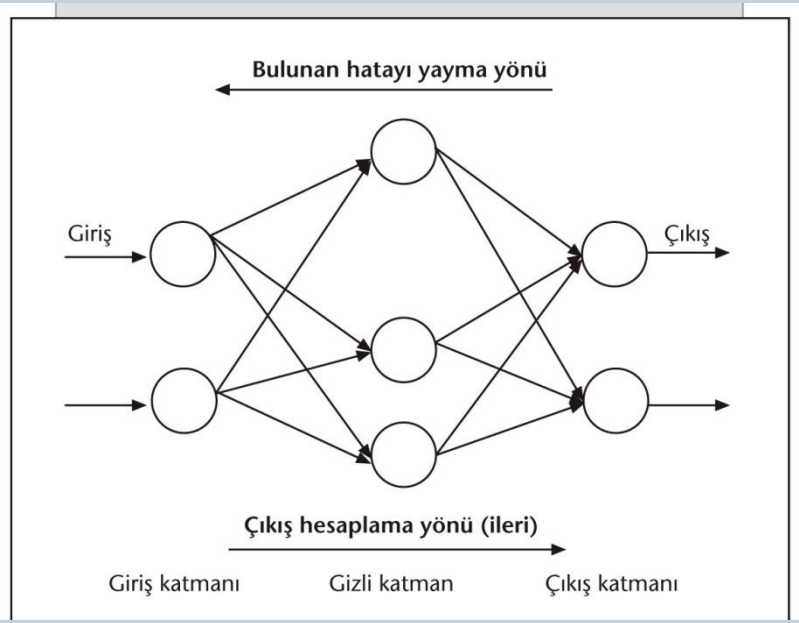
- Geri beslemeli YSA' da, en az bir hücrenin çıkısı kendisine ya da diğer hücrelere giriş olarak verilir ve genellikle geri besleme bir geciktirme elemanı üzerinden yapılır.

- Geri besleme, bir katmandaki hücreler arasında olduğu gibi katmanlar arasındaki hücreler arasında da olabilir.

- Bu yapısı ile geri beslemeli YSA , doğrusal olmayan dinamik bir davranış gösterir.

Geri Beslemeli Yapay Sinir Ağları

- Dolayısıyla, geri beslemenin yapılış sekline göre farklı yapıda ve davranışta geri beslemeli YSA yapıları elde edilebilir. Bu nedenle, bu bölümde bazı geri beslemeli YSA yapılarında örnekler verilecektir. Şekilde iki katmanlı ve çıkışlarından giriş katmanına geri beslemeli bir YSA yapısı görülmektedir.



Çok Katmanlı Algılayıcı
(Multilayer Perceptron)

JEOLOJİ MÜHENDİSLİĞİ A.B.D.
ESNEK HESAPLAMA YÖNTEMLERİ-II

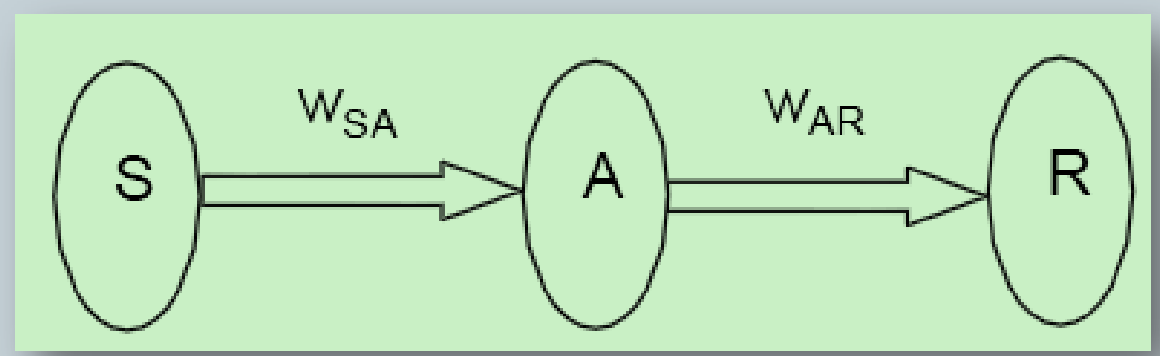
DOÇ. DR. ERSAN KABALCI

Perceptron

- Rosenblatt (1962):
  - İlk olarak görsel algıyı modellemede (retina) kullanıldı
  - Bir ileri beslemeli ağın üç parçası vardır :

duyular(sensory), eşleştirme(association), cevap(response)

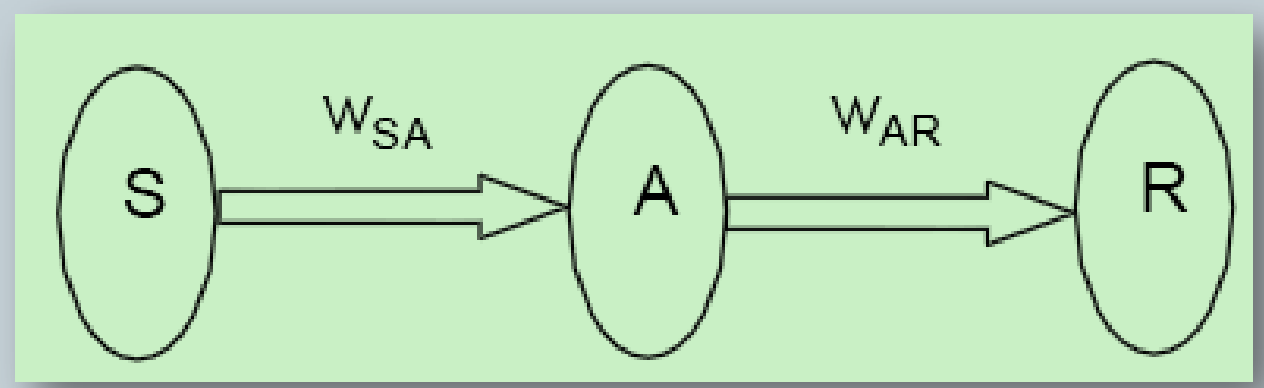
  - Öğrenim sadece A birimlerinden R birimlerine ağırlıklar üzerinde meydana gelir (S birimlerinden A birimlerine olan ağırlıklar sabittir).



Perceptron

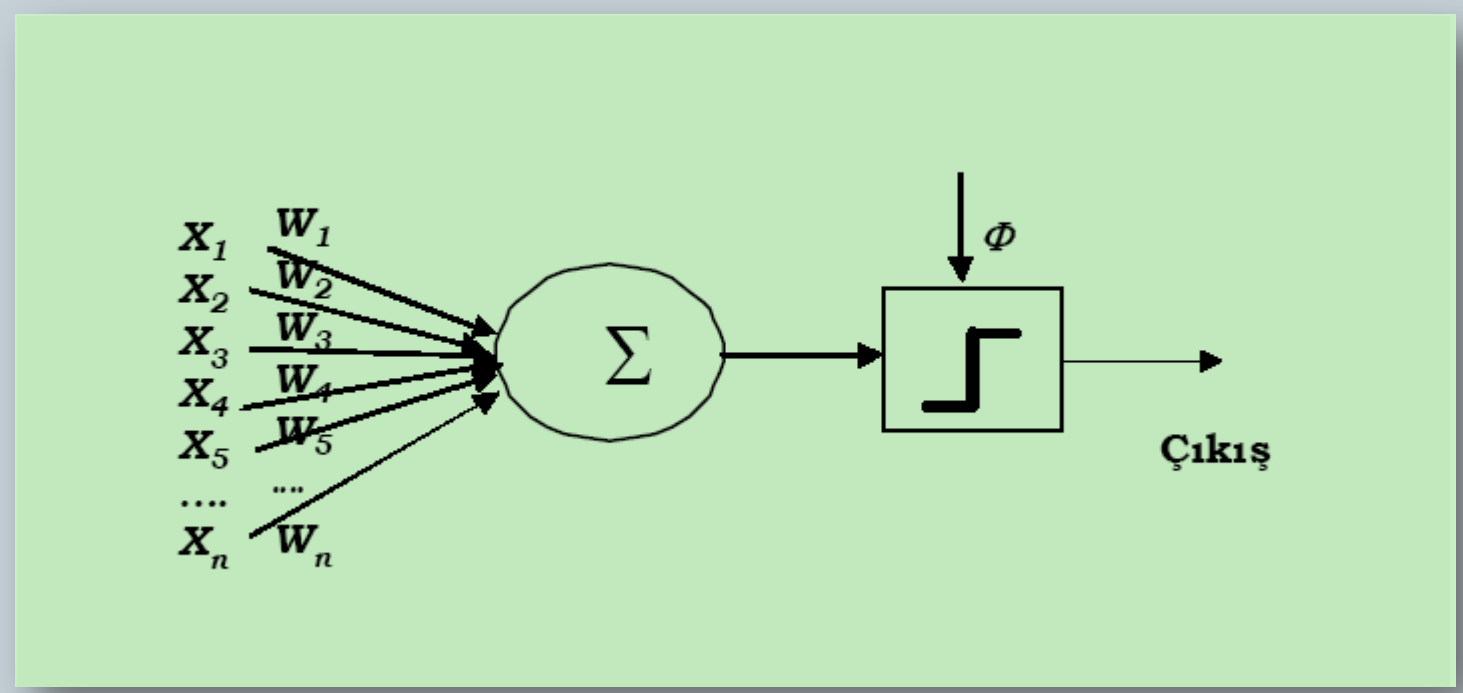
- Her bir ***R*** birimi n tane A biriminden giriş alır.

- Verilen bir eğitim örneği için (s:t), hedef çıktıdan (t) farklı bir y çıktısı hesap edilirse A ve R arasında ağırlıklar değiştirilir.



Perceptron

- Perceptronlar, son derece sınırlı olmalarına karşın en eski sinir ağlarından biridir. Perceptron, bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanır.



Basit bir perceptron yapısı

Perceptron Nerelerde Kullanılır ?

- Perceptron doğrusal bir fonksiyonla iki parçaya bölünebilen problemlerde kullanılabilir.
- Bu problemlere AND, OR, NOT durumları örnek olarak verilebilir.

Terimler

- *Epoch :* Olası girdiler için ağırlıkların güncellenme sayısına denir.

- *Error:* Çıktı değeriyle bizim fonksiyondan beklediğimiz değer arasındaki farktır. Örneğin, eğer biz çıktı olarak 0 bekleyip de 1 aldığımızda hata (error) değeri -1‘ dir.

- *Target Value, T :* Perceptrondan öğrenmesini beklediğimiz değerdir. Örneğin, eğer AND fonksiyonuna [1,1] girdisini verirsek bekleyeceğimiz sonuç 1’dir.

Terimler

- *Output , O :* Perceptron'un verdiği çıktıdır.
- *Xi:* Nörona verilen girdi
- *Wi:* Xi'nci girdinin ağırlık değeri
- *LR (Learning rate):* Bir perceptron'un hedefe varması için gereken adım büyüklüğü.

Öğrenme Algoritması

Epoch hata ürettiği sürece

{

Girdilere göre çıktıyı hesapla

Error= T -O

If Error<> 0 then

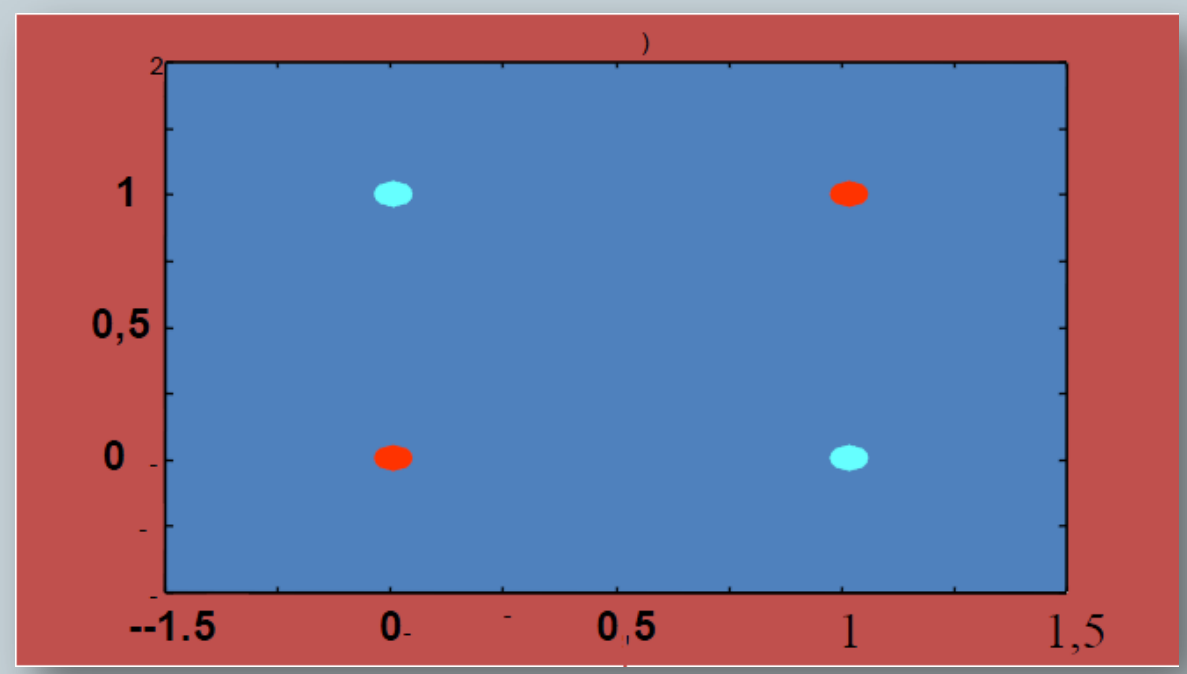
i= Wi+ LR * Xi* Error

End If

}

XOR Problemi

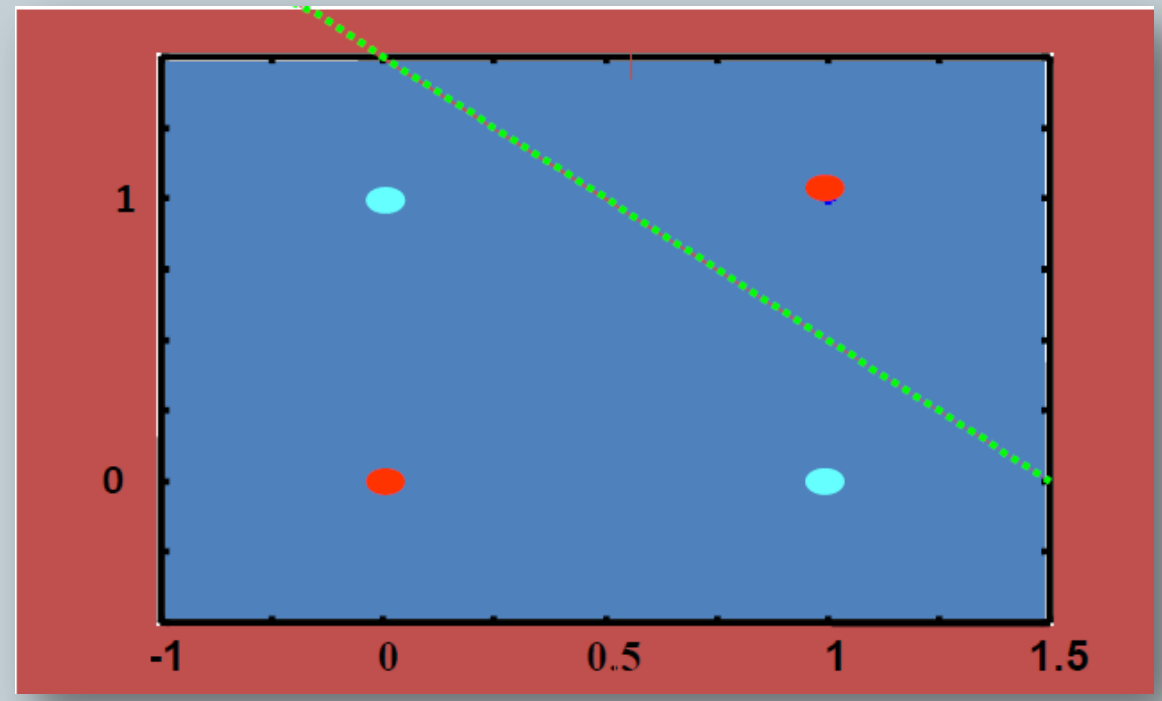
- Perceptronlar XOR problemi gibi doğrusal olarak sınıflandırılamayan problemleri çözümünde başarısızdır.



Kırmızı : Lojik 0 Mavi : Lojik 1

XOR Problemi

- Diğer bir deyişle çıktıların arasına bir doğru veya doğrular çizerek onları iki veya daha fazla sınıfa ayırmak mümkün değildir.



İkinci Girdi kümesi için hatalı

XOR Problemi

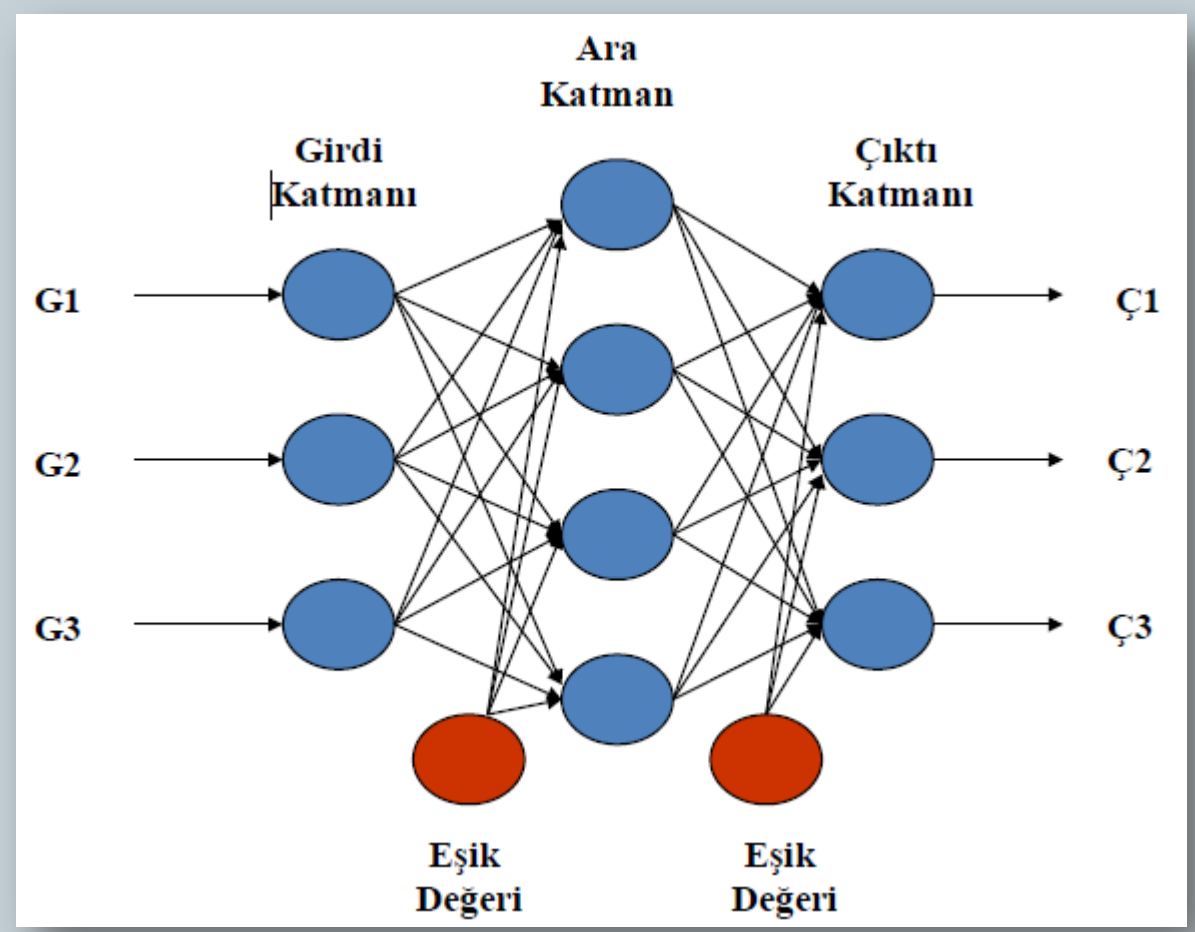
- XOR problemini çözmek için yapılan çalışmalar sonucu çok katmanlı algılayıcı modeli geliştirilmiştir.
- Rumelhart ve arkadaşları tarafından geliştirilen bu hata yayma modeli veya geriye yayılım modeli (back propogation) denilmektedir.
- Bunu özellikle sınıflandırma, tanıma ve genelleme yapmayı gerektiren problemler için çok önemli bir çözüm aracıdır. Bu model *"Delta Öğrenme Kuralı"* denilen bir öğrenme yöntemini kullanmaktadır.

XOR Problemi

- Bu kural aslında ADALINE ve basit algılayıcı modelinin öğrenme kurallarının geliştirilmiş bir şeklidir.

- Temel amacı ağın beklenilen çıktısı ile ürettiği çıktı arasındaki hatayı aza indirmektir. Bunu hatayı ağa yayarak gerçekleştirdiği için bu ağa hata yayma ağı denmektedir.

Çok Katmanlı Ağ Modelinin Yapısı



Çok Katmanlı Ağ Modelinin Yapısı

Girdi Katmanı: Dış dünyadan gelen girdileri alarak ara katmana gönderir. Bu katmanda bilgi işleme olmaz. Gelen her bilgi geldiği gibi bir sonraki katmana gider.

Her proses elemanın sadece bir tane girdisi ve bir tane çıktısı vardır. Yani, girdi katmanındaki her proses elemanı bir sonraki katmanda bulunan proses elemanlarının hepsine bağlanır.

Ara Katmanı: Ara katmanlar girdi katmanından gelen bilgileri işleyerek bir sonraki katmana gönderir.

Çok katmanlı bir ağda birden fazla ara katman ve her katmanda birden fazla proses elemanı bulunabilir.

Çok Katmanlı Ağ Modelinin Yapısı

Çıkış Katmanı: Ara katmandan gelen bilgileri işleyerek ağa girdi katmanından verilen girdilere karşılık ağın ürettiği çıkışları 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 elemanının bir çıktısı vardır.

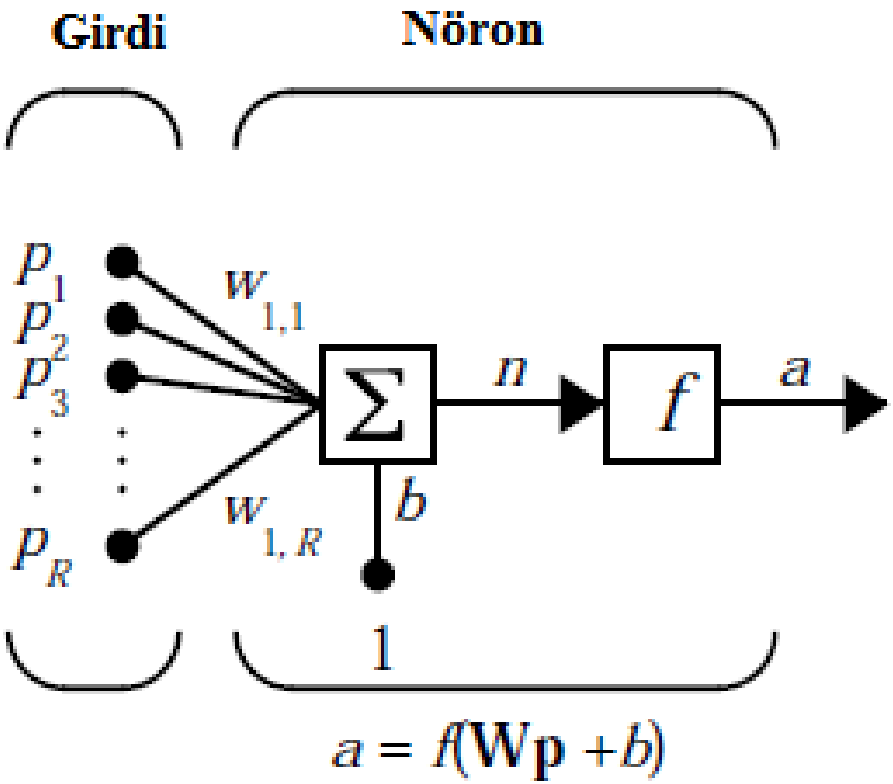
Çok Katmanlı Ağ Modelinin Yapısı

- Çok katmanlı ağ öğretmenli öğrenme stratejisini kullanır. Ağa, hem örnekler hem de örneklerden elde edilmesi gereken çıktılar verilmektedir.

- Sistem, 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.

Çok Katmanlı Ağ Hücresi



n : net giriş toplamı

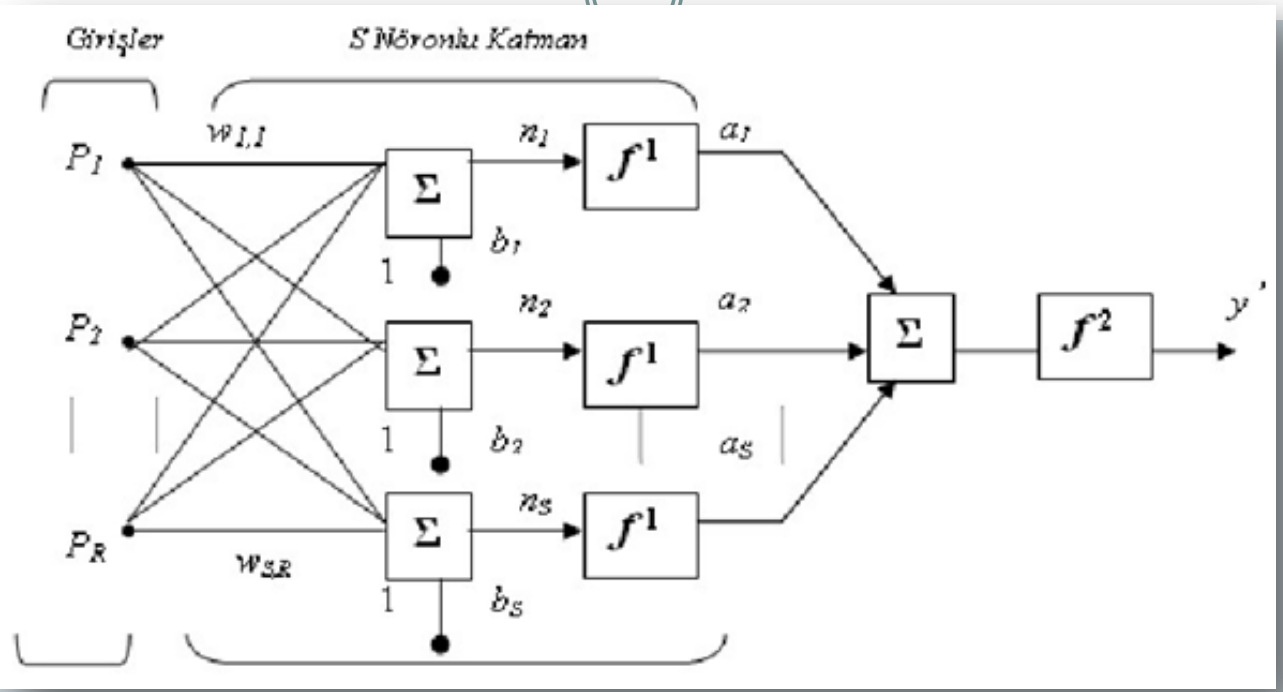
a : çıkış

Σ : Toplam fonksiyonu.

f : Aktivasyon fonksiyonu.

- Sigmoid
- Tanh
- Lineer

Çok Katmanlı Ağ Hücresi



Bir çok giriş için genellikle bir nöron yeterli olmayabilir. Paralel işlem yapan birden fazla nörona ihtiyaç duyulduğunda katman kavramı devreye girmektedir. S tane nöronun tek bir katmanı şekilde gösterilmiştir. Burada her giriş bir nörona bağlıdır.

Çok Katmanlı Ağın Öğrenme Kuralı

- Çok katmanlı ağın öğrenme kuralı en küçük kareler yöntemine dayalı "*Delta Öğrenme Kuralı*"nın genelleştirilmiş halidir. Bu yüzden "*Genelleştirilmiş Delta Kuralı*" olarak da isimlendirilmektedir.

- 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.

Çok Katmanlı Ağın Öğrenme Kuralı

"*Genelleştirilmiş Delta Kuralı*" iki aşamadan oluşur:

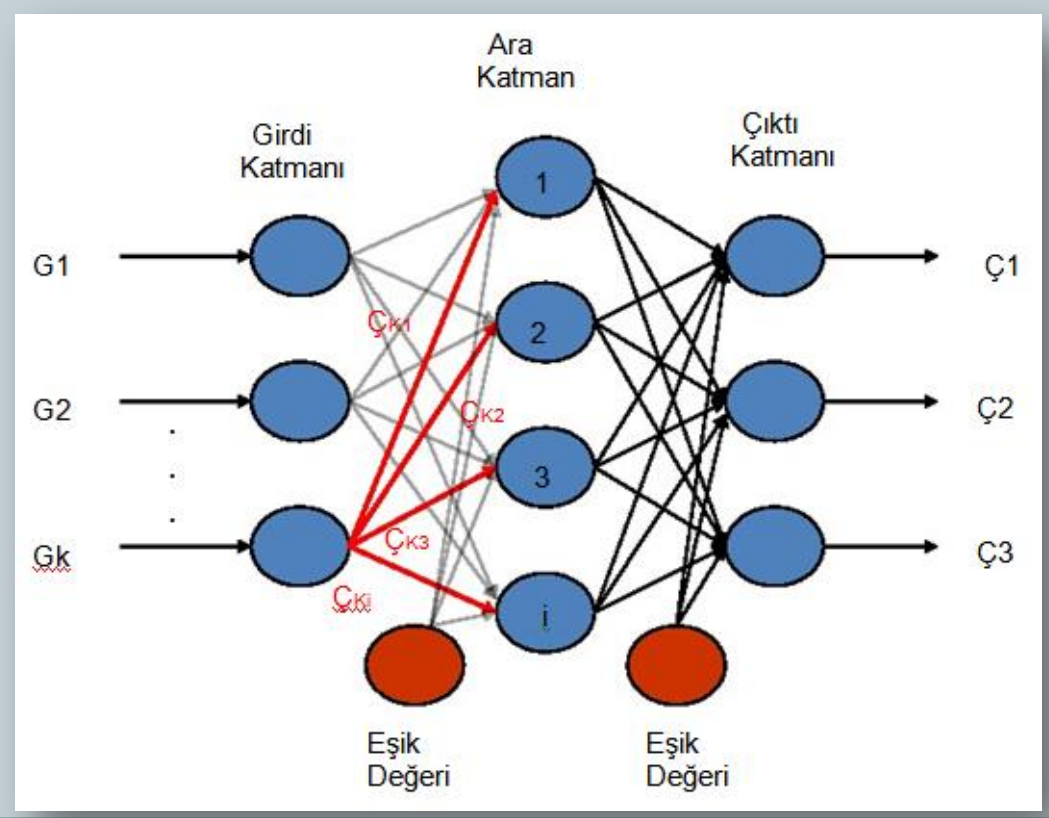
- *İleri doğru hesaplama(Feed Forward)*
- *Geri doğru hesaplama (Back Propogation)*

I) İleri Doğru Hesaplama:

Bu safhada bilgi işleme eğitim setindeki bir örneğin Girdi Katmanından ağa gösterilmesi ile başlar. Gelen girdiler hiç bir değişiklik olmadan ara katmana gönderilir.

Çok Katmanlı Ağın Öğrenme Kuralı

Girdi katmanındaki k. Proses elemanının çıktısı $Ç_k^i$ şu şekilde belirlenir: $Ç_k^i = G_k$



Çok Katmanlı Ağın Öğrenme Kuralı

- Ara katmandaki her proses elemanı girdi katmanındaki bütün proses elemanlarından gelen bilgileri bağlantı ağırlıklarının (A1,A2,....) etkisi ile alır.

- Önce ara katmandaki proses elemanlarına gelen net girdi (NET_j^a) şu formül kullanılarak hesaplanır:

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

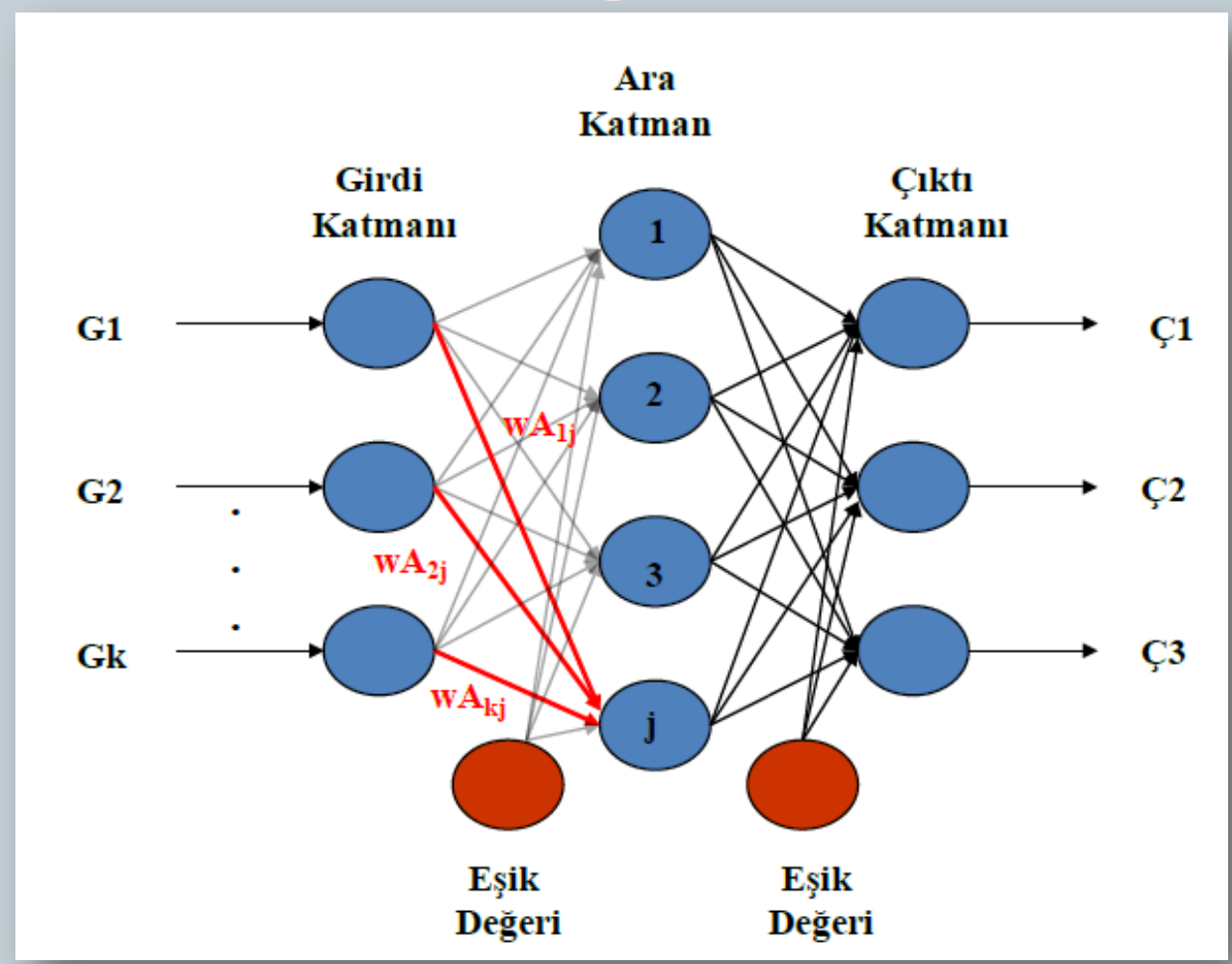
Çok Katmanlı Ağın Öğrenme Kuralı

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

Burada A_{kj} k. girdi elemanını j. ara katman elemanına bağlayan bağlantının ağırlık değerini göstermektedir.

J. ara katman elemanının çıktısı ise bu net girdinin aktivasyon fonksiyonundan geçirilmesiyle hesaplanır.

Çok Katmanlı Ağın Öğrenme Kuralı



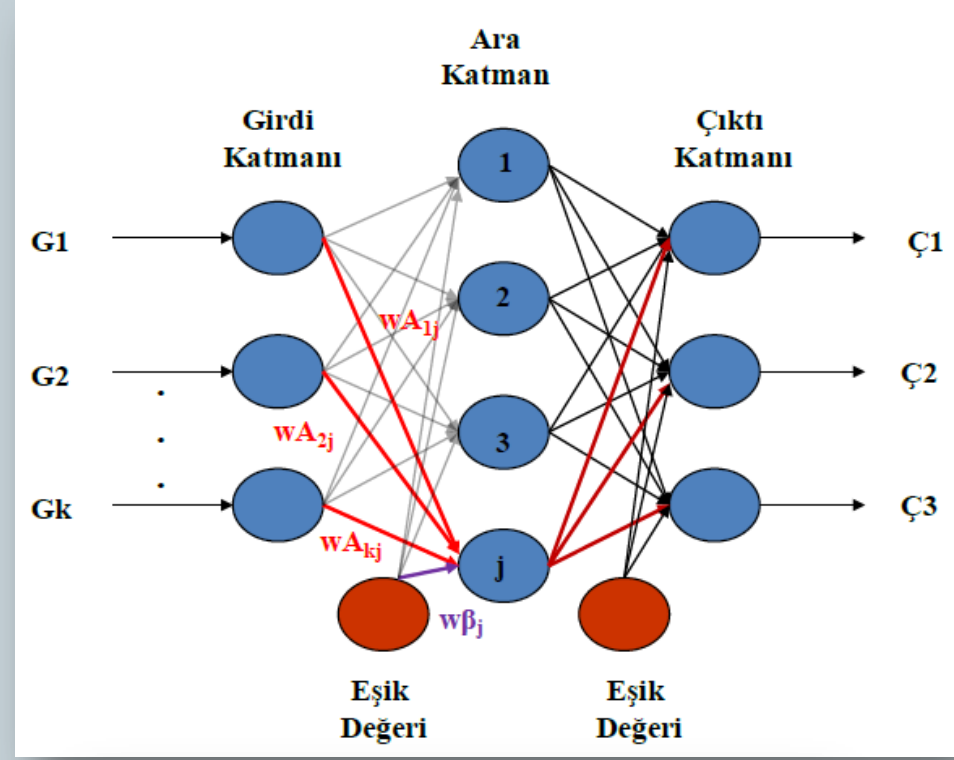
Çok Katmanlı Ağın Öğrenme Kuralı

- Sigmoid fonksiyonu kullanılması halinde çıktı:

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

- Burada β_j , ara katmanda bulunan j. elemana bağlanan eşik değer elemanının ağırlığını göstermektedir. Bu eşik değer ünitesinin çıktısı sabit olup 1'e eşittir. Eğitim sırasında ağ bu değeri kendisi belirlemektedir.

Çok Katmanlı Ağın Öğrenme Kuralı



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

- Ara katmanın bütün proses elemanları ve çıktı katmanının proses elemanlarının çıktıları aynı şekilde kendilerine gelen NET girdinin hesaplanması ve sigmoid fonksiyonundan geçirilmesi sonucu belirlenirler.

- Çıktı katmanından çıkan değerler bulununca ağın ileri doğru hesaplama işlemi tamamlanmış olur.

II) Geriye Doğru Hesaplama

- Ağa sunulan girdi için ağın ürettiği çıktı ağın beklenen çıktıları ile karşılaştırılır. Bunların arasındaki fark hata olarak kabul edilir. Amaç bu hatanın düşürülmesidir.

- Bu hata, ağın ağırlık değerlerine dağıtılarak bir sonraki iterasyonda hatanın azaltılması sağlanır.

- Çıktı katmanındaki m. proses elemanı için oluşan hata E_m ;

$$E_m = B_m - Ç_m$$

II) Geriye Doğru Hesaplama

- Yukarıdaki hata, bir proses elemanı için oluşan hatadır. Çıktı katmanı için oluşan toplam hatayı (TH) bulmak için bütün hataların toplanması gerekir.

$$TH=1/2(\sum E^2_m)$$

- Toplam hatayı en azlamak için bu hatanın kendisine neden olan proses elemanlarına dağıtılması gerekmektedir.

II) Geriye Doğru Hesaplama

- Ağın ağırlıklarını değiştirmek için 2 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 katmandaki j. Proses elemanı çıktı katmanındaki m. Proses elemanına bağlayan bağlantının ağırlığındaki değişim miktarına ΔA^a denirse; herhangi bir t zamanında ağırlığın değişim miktarı şöyle hesaplanır:

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

- Burada λ öğrenme katsayısını, α momentum katsayısını göstermektedir.

Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Değiştirilmesi

- Momentum katsayısı ağın öğrenmesi esnasında yerel bir optimum noktaya takılıp kalmaması için ağırlık değişim değerinin belirli bir oranda bir sonraki değişime eklenmesini sağlar.

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

- Yine yukarıdaki formül dikkate alındığında δ_m ise m. Çıktı ünitesinin hatasını göstermektedir

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

Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Değiştirilmesi

- f’(NET) aktivasyon fonksiyonunun türevidir. Sigmoid fonksiyonun kullanılması durumunda

$$\delta_m = Ç_m(1-Ç_m)\, E_m$$

- Değişim miktarı hesaplandıktan sonra ağırlıkların t.iterasyondaki yeni değerleri:

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

Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Değiştirilmesi

- Benzer şekilde eşik değer ünitesinin de ağırlıklarını değiştirmek gerekmektedir. Çıktı katmanında bulunan proses elemanlarının eşik değer ağırlıkları β^{ς} ile gösterilirse; bu ünitenin çıktısı sabit ve1olması nedeni ile değişim miktarı:

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

olacaktır. Eşik değerin t. İterasyonundaki ağırlığının yeni değeri ise ;

$$\beta_m^{\varsigma}(t) = \beta_m^{\varsigma}(t-1) + \Delta\,\beta_m^{\varsigma}(t)$$

şeklinde hesaplanacaktır.

Ara Katmanlar Arası veya Ara Katman Girdi Katmanı Arasındaki Ağırlıkların Değiştirilmesi

- Ara katman ile çıktı katman arasındaki ağırlıkların değişiminde her ağırlık için sadece çıktı katmanındaki bir proses elemanının hatası dikkate alınmıştır. Oysaki bu hataların oluşmasında girdi katmanı ve ara katman arasındaki ağırlıkların payı vardır.
- Girdi katmanı ile ara katman arasındaki ağırlıkların değişimi ΔA^i ile gösterilirse değişim miktarı:

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

Ara Katmanlar Arası veya Ara Katman Girdi Katmanı Arasındaki Ağırlıkların Değiştirilmesi

Yine buradaki hata terimi δ^a şöyle hesaplanacaktır:

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

Aktivasyon fonksiyonu olarak sigmoid fonksiyonun kullanılması durumunda;

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

Ara Katmanlar Arası veya Ara Katman Girdi Katmanı Arasındaki Ağırlıkların Değiştirilmesi

Ağırlıkların yeni değerleri;

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

Benzer şekilde eşik değer ünitesinin de ağırlıklarını değiştirmek gerekmektedir. Ara katman eşik değer ağırlıkları β^a ile gösterilirse değişim miktarı;

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

Ara Katmanlar Arası veya Ara Katman Girdi Katmanı Arasındaki Ağırlıkların Değiştirilmesi

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 doğru hesaplamaları yapılarak tamamlanmış olacaktır.

Çok Katmanlı Ağın Çalışma Şekli

Çok Katmanlı Ağın Çalışma Şekli

Örneklerin toplanması: Ağın çözmesi istenen olay için daha önce gerçekleşmiş örneklerin bulunması adımıdır. Ağın eğitilmesi için örnekler toplandığı gibi (eğitimseti), ağın test edilmesi içinde örneklerin (testseti) toplanması gerekmektedir.

Ağın topolojik yapısının belirlenmesi: Öğrenilmesi istenen olay için oluşturulacak olan ağın yapısı belirlenir. Kaç tane girdi ünitesi, kaç tane ara katman, her ara katmanda kaç tane hücre elemanı ve kaç tane çıktı elemanı olması gerektiği belirlenmektedir.

Çok Katmanlı Ağın Çalışma Şekli

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

Ağın başlangıç değerlerinin atanması: Hücre elemanlarını bir birine bağlayan ağırlık değerlerinin ve eşik değere başlangıç değerinin atanması

Çok Katmanlı Ağın Çalışma Şekli

Öğrenme setinden örneklerin seçilmesi ve ağa gösterilmesi: Ağın öğrenmeye başlaması, öğrenme kuralına uygun olarak ağırlıkların değiştirilmesi için ağa örneklerin gösterilmesi.

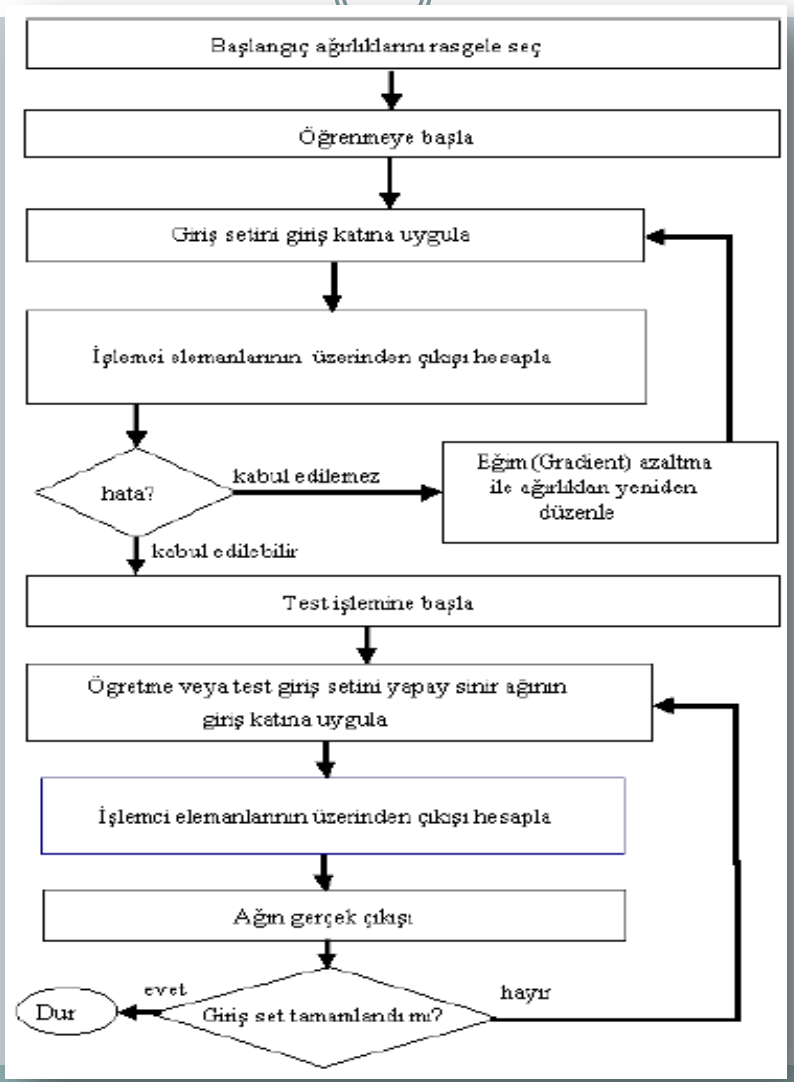
Öğrenme sırasında ileri hesaplamaların yapılması: Verilen girdi için ağın çıktı değerinin hesaplanması.

Çok Katmanlı Ağın Çalışma Şekli

Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması: Ağın ürettiği hata değerlerinin hesaplanması.

Ağırlıkların değiştirilmesi: Geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi

Çok Katmanlı Ağın Çalışma Şekli



Ağın Eğitilmesi

- Ağın kendisine gösterilen girdi örneği için beklenen çıktıyı üretmesini sağlayacak ağırlık değerleri bulunmaktadır.

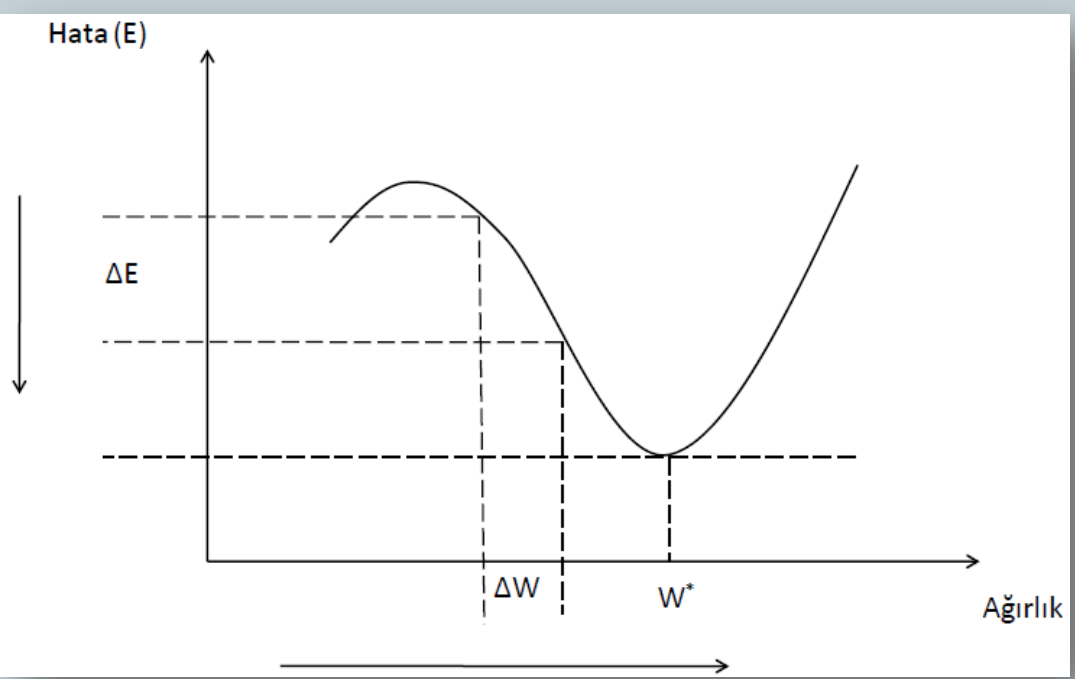
- Başlangıçta bu değerler rastgele atanmakta ve ağa örnekleri gösterdikçe ağın ağırlıkları değiştirilerek zaman içerisinde istenilen değerlere ulaşması sağlanmaktadır.

Ağın Eğitilmesi

- İstenilen ağırlık değerlerinin ne olduğu ise bilinmemektedir. Bu nedenle YSA'nın davranışlarını yorumlamak ve açıklamak mümkün olmamaktadır.

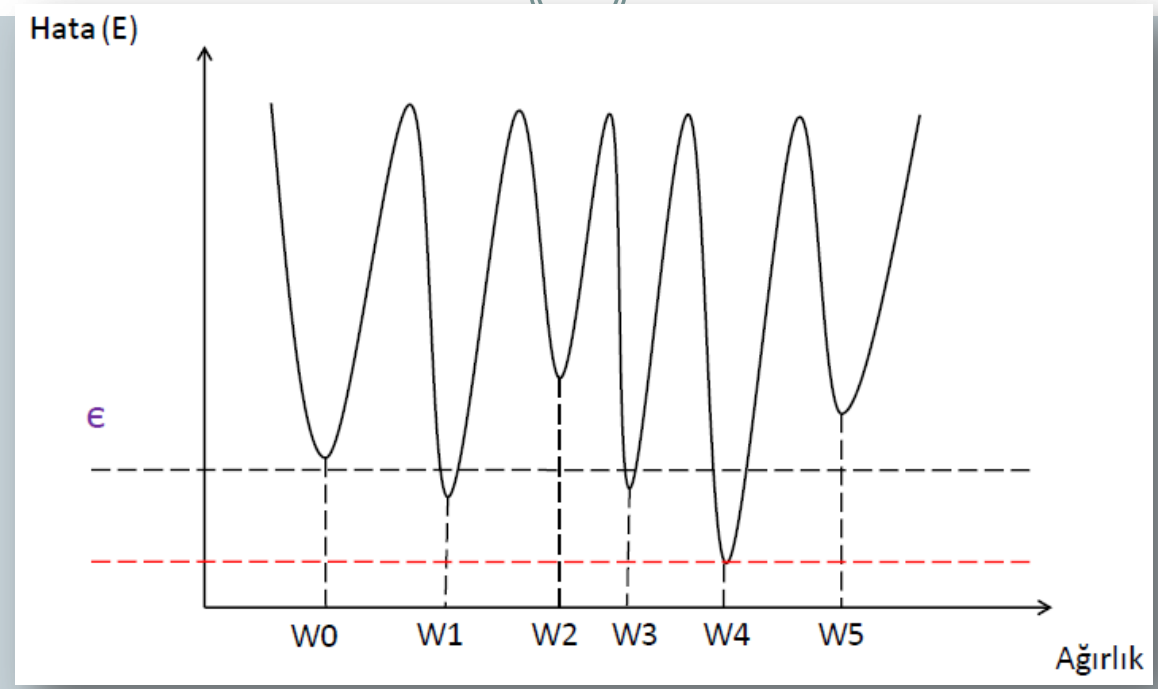
- Bunun temel nedeni ise, bilginin ağ üzerinde dağıtılmış olması ve ağırlık değerlerinin kendi başlarına her hangi bir anlam göstermemeleridir.

Öğrenmenin Hata Uzayındaki Gösterimi



- Ağın W* değerine ulaşması istenmektedir. Her iterasyonda ΔW kadar değişim yaparak hata düzeyinde ΔE kadar bir hatanın düşmesi sağlanır

Öğrenmenin Hata Uzayındaki Gösterimi



- Problemin çözümü için en az hatayı veren ağırlık vektörü W* olmasına rağmen pratikte bu hata değerini yakalamak mümkün olmayabilir. Bu çözüm ağın sahip olabileceği *en iyi çözümdür.*

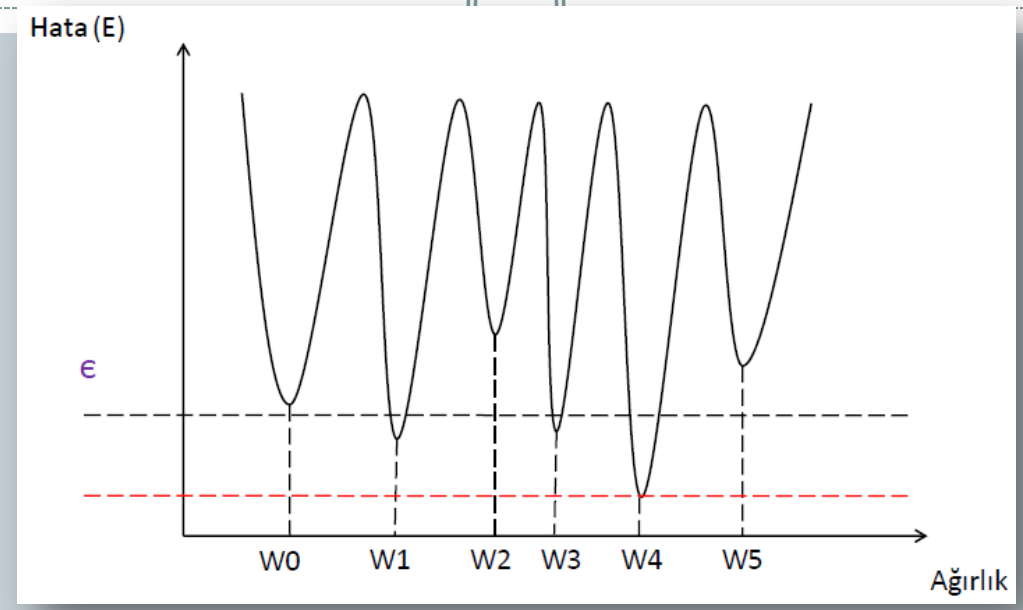
Ağın Eğitilmesi

- Bazen farklı bir çözüme takılabilmekte ve performansı daha iyileştirmek mümkün olmamaktadır.
- Bu nedenle kullanıcılar ağların performanslarında ϵ kadar hatayı kabul etmektedirler. Bu tolerans değerinin altındaki her hangi bir nokta da olay öğrenilmiş kabul edilir.
- En iyi çözümün bulunamamasının nedeni aşağıdakilerden biri veya birkaçı olabilir:

Ağın Eğitilmesi

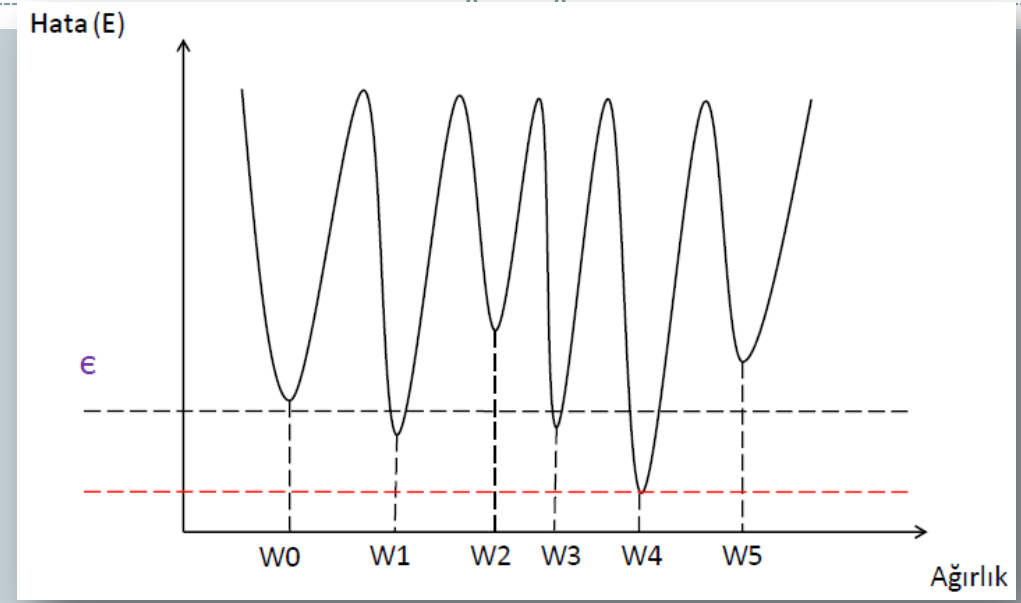
- Problem eğitilirken bulunan örnekler problem uzayını %100 temsil etmeyebilir.
- Oluşturulan çok katmanlı ağ için doğru parametreler seçilmemiş olabilir.
- Ağın ağırlıkları başlangıçta tam istenildiği şekilde belirlenmemiş olabilir.
- Ağın topolojisi yetersiz seçilmiş olabilir.

Ağın Eğitilmesi



W0,W2 yerel çözümler olup tolerans değerinin üzerinde kaldığı için kabul edilemezler. Yine W1,W3,W4 ve W5 de yerel çözümler olup kabul edilebilirler. Fakat en iyi çözüm W5'dir. Fakat her zaman bu çözüme ulaşmak az önce sayılan nedenlerden ötürü mümkün olmayabilir.

Ağın Eğitilmesi



- Bazı durumlarda ağın takıldığı yerel sonuç kabul edilebilir hata düzeyinin üstünde kalabilir. Örneğin aşağıdaki şekilde w0 ağırlığının bulunması ve hatanın daha fazla azaltılmasının mümkün olmaması gibi.

Ağın Eğitilmesi

Bu durumda ağın olayı öğrenmesi için bazı değişiklikler yapılarak yeniden eğitilmesi gerekmektedir. Bu değişikler şunlar olabilir:

- Başka başlangıç değerleri kullanılabilir.
- Topolojide değişiklikler yapılabilir (Ara katman sayısını arttırmak, proses elemanı sayısını arttırmak veya azaltmak)

Ağın Eğitilmesi

- Parametrelerde değişiklik yapılabilir. (Farklı fonksiyonların seçilmesi, öğrenme ve momentum katsayılarının değiştirilmesi)
- Problemin gösterimi ve örneklerin formülasyonu değiştirilerek yeni örnek seti oluşturulabilir.
- Öğrenme setindeki örneklerin sayısı arttırılabilir veya azaltılabilir.
- Öğrenme sürecinde örneklerin ağa gösterilmesi.

Momentum Katsayısı

- Çok katmanlı ağların yerel sonuçlara takılıp kalmaması için momentum katsayısı geliştirilmiştir.

- Bu katsayının iyi kullanılması yerel çözümleri kabul edilebilir hata düzeyinin altına çekebilmektedir.

- Çok katmanlı ağların diğer bir sorunu ise öğrenme süresinin çok uzun olmasıdır.

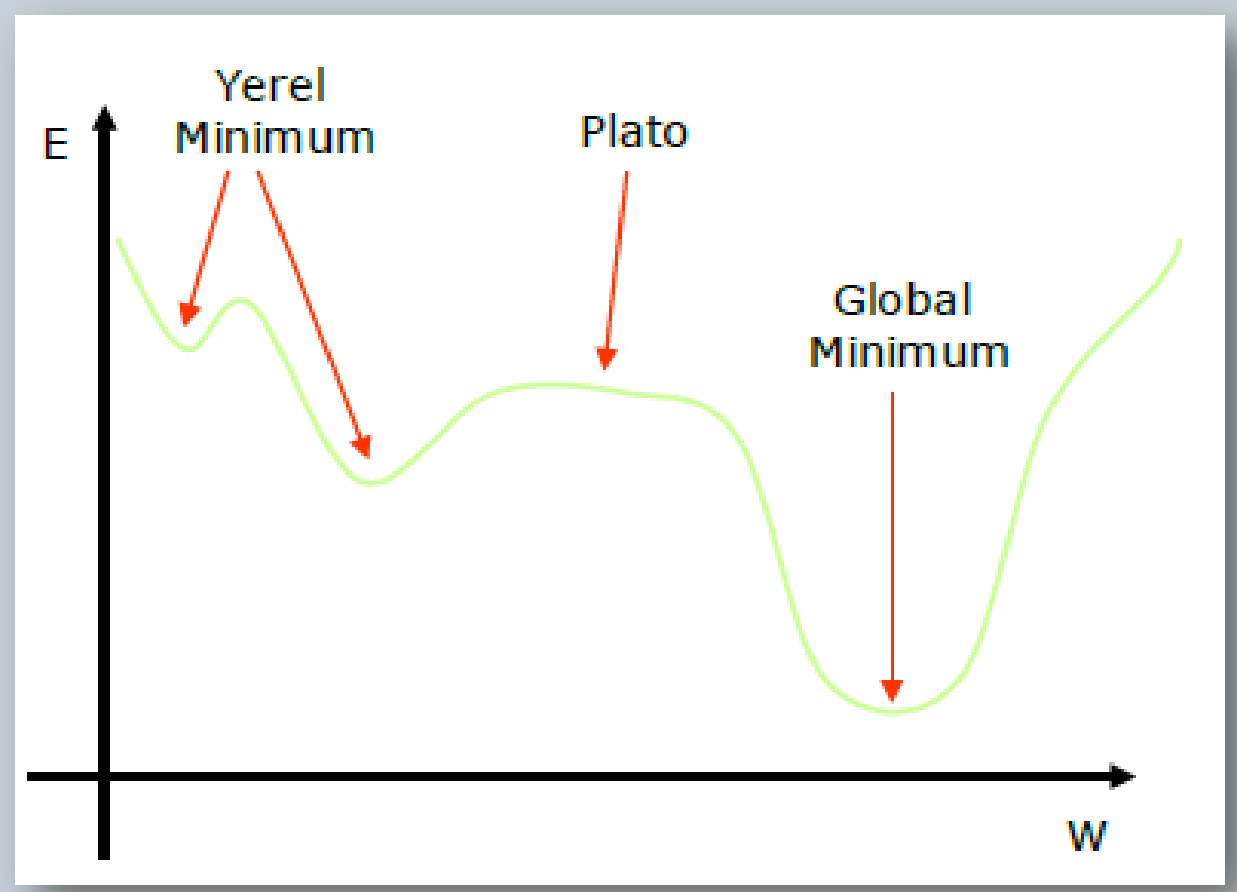
Momentum Katsayısı

- Ağırlık değerleri başlangıçta büyük değerler olması durumunda ağın yerel sonuçlara düşmesi ve bir yerel sonuçtan diğerine sıçramasına neden olmaktadır.

- Eğer ağırlıklar küçük aralıkta seçilirse o zamanda ağırlıkların doğru değerleri bulması uzun sürmektedir.

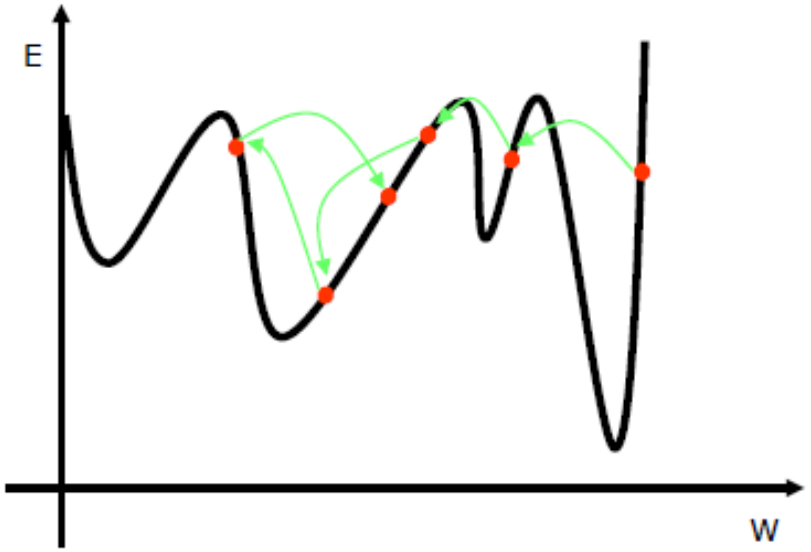
- Momentum katsayısı, yerel çözümlere takılmayı önler. Bu değerin çok küçük seçilmesi yerel çözümlerden kurtulmayı zorlaştırır. Değerin çok büyük seçilmesi ise tek bir çözüme ulaşmada sorunlar yaratabilir.

Momentum Katsayısı

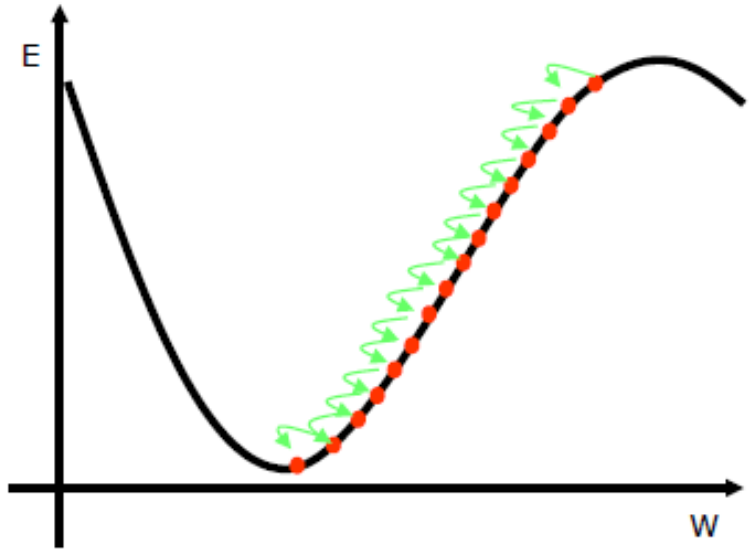


Öğrenme Katsayısı

Öğrenme katsayısı ağırlıkların değişim miktarını belirler.

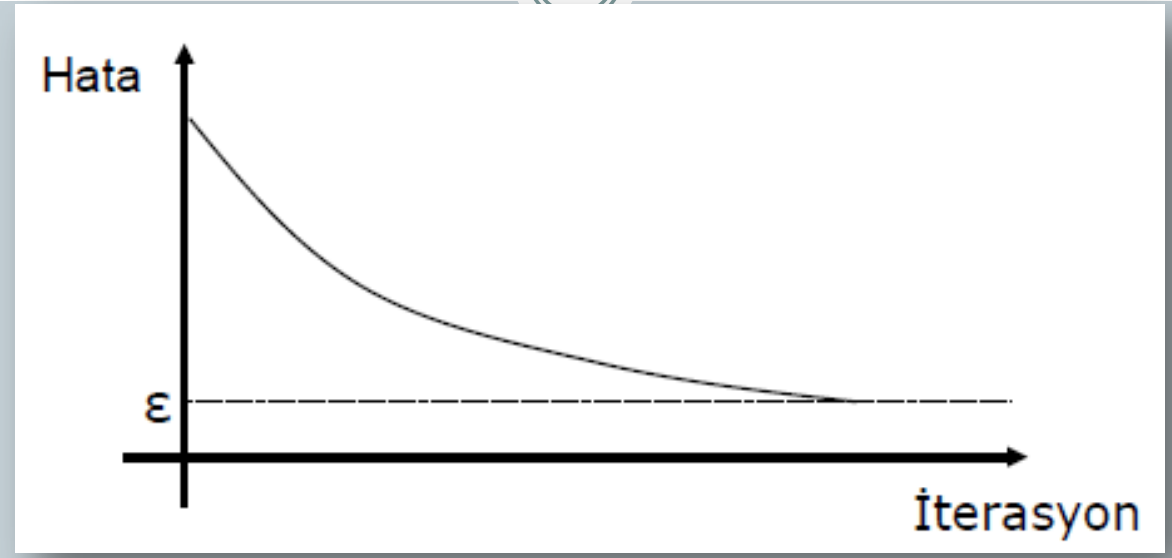


Eğer öğrenme katsayısı gereğinden büyük olursa problem uzayında rasgele gezinme olur. Bunun da ağırlıkları rasgele değiştirmekten farkı olmaz.



Eğer öğrenme katsayısı çok küçük olursa çözüme ulaşmak daha uzun sürer.

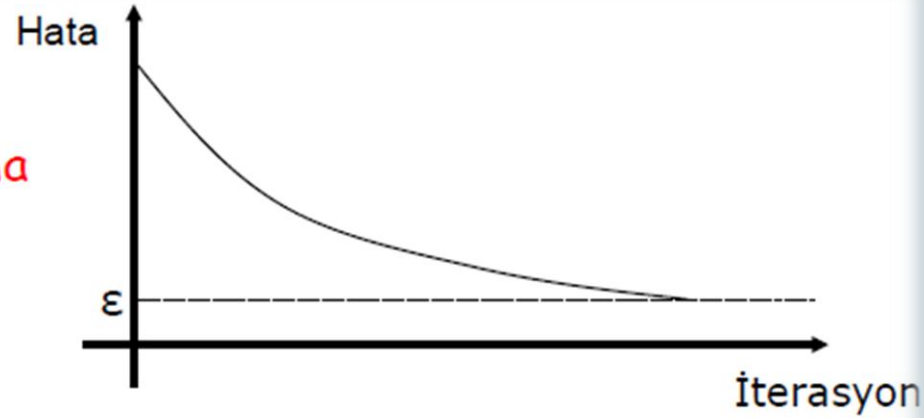
Ağın Hatası



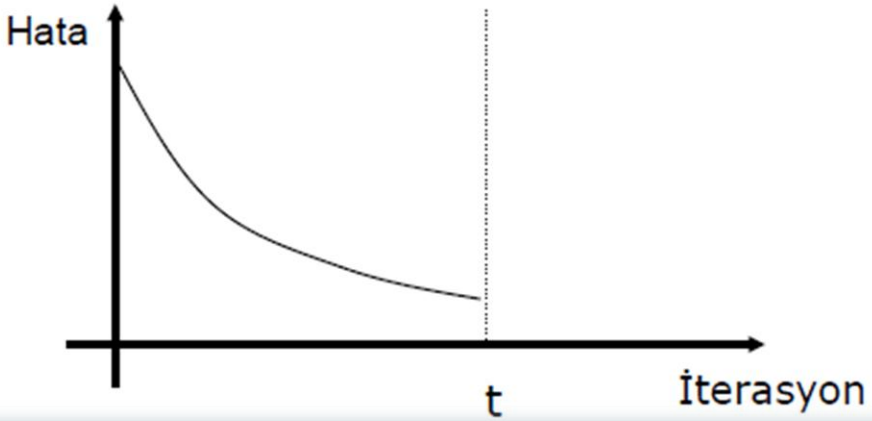
- Belirli bir iterasyondan sonra hatanın daha fazla azalmadığı görülür. Bu ağın öğrenmesini durdurduğu ve daha iyi bir sonuç bulunamayacağı anlamına gelir. Eğer elde edilen çözüm kabul edilemez ise o zaman ağ yerel bir çözüme takılmış demektir.

Ağın Hatası

Hatanın belli bir değerin altına düşmesi sonucu durma



Belirli sayıda iterasyondan sonra durma



Çok Katmanlı Ağın Performansının Ölçülmesi

- Bir yapay sinir ağının performansı denilince öğrenme yeteneğinin ölçülmesi anlaşılır. Ağın performansı ağa daha önce hiç görmediği test örnekleri gösterilince bu örnekler karşısında ürettiği doğru cevaplar oranı şeklinde ölçülür.

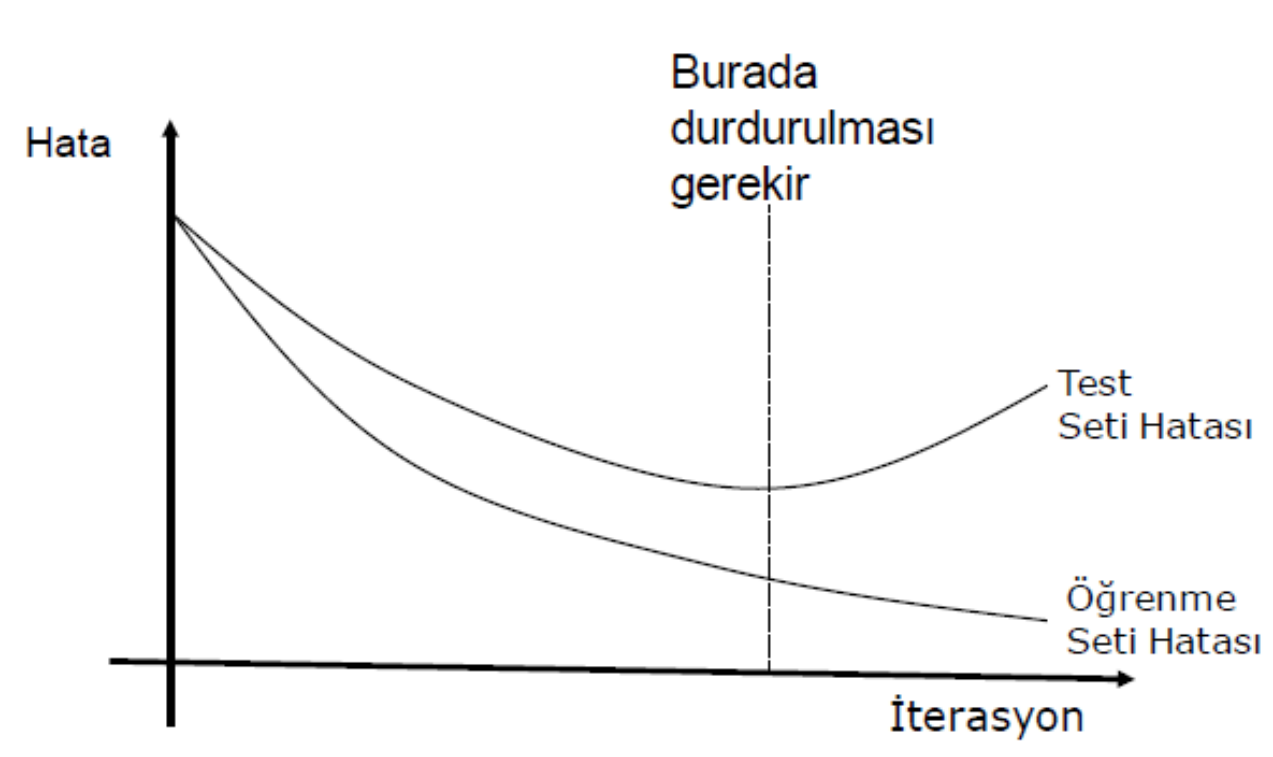
$$\text{Performans} = \frac{\text{Test setinde bulunan örneklere verilen doğru cevap sayısı}}{\text{Test setinde bulunan toplam örnek sayısı}} \times 100$$

Ağın Ezberlemesi

- Bazı durumlarda eğitilen ağ eğitim setindeki bütün örneklere %100 doğru cevap üretmesine rağmen test setindeki örneklere doğru cevaplar üretememektedir. Bu durumda ağın öğrenmediği fakat öğrenme setini ezberlediği görülmektedir.

- Ağ gereğinden fazla eğitilirse problemi öğrenmek yerine verileri ezberler. Buda ağın genelleme yapamamasını ve hatalı sonuçlar üretmesine neden olur.

Ağın Ezberlemesi



Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Örneklerin seçilmesi: Seçilen örnekler problem uzayını temsil edebilecek nitelikte olması gerekir. Çok katmanlı ağ tasarımcısı, problem uzayının her bölgesinden ve uzayı temsil eden örnekler seçmesi gerekir.

Girdi ve çıktıların ağa gösterilmesi: Problem uzayı sayısal verilerden oluşmuyorsa, bu örnekleri sayısal olarak temsil etmek gerekir.Bu dönüştürme çeşitli şekillerde olabilmekte ve bu da ağın performansını etkilemektedir.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Girdilerin sayısal gösterimi: Ağa sunulan girdilerin sayısal duruma dönüştürülmesi her zaman kolay olmamaktadır. Bu da tasarımı zorlaştırabilmektedir.

Çıktıların sayısal gösterimi: Girdilerde olduğu gibi çıktılarda da sayısal gösterim problemden probleme değişmektedir. Bir problem için bir den fazla yöntem kullanılarak sayısal gösterim sağlanabilir. Bunların en iyisinin hangisi olduğu bilinmemektedir. Önemli olan uygun olanı bulmaktır.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Başlangıç değerinin atanması: Başlangıç değerinin atanması performansı etkilemektedir. Genel olarak ağırlıklar belirli aralıkta atanmaktadır.

- Bu ağırlıklar büyük tutulursa ağın yerel çözümler arasında sürekli dolaştığı,
- Küçük olması durumunda ise, öğrenmenin geç gerçekleştiği görülmüştür.
- Tecrübeler,-1.0 ile 0.1 arasındaki değerlerin başarılı sonuçlar ürettiğini göstermektedir.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Öğrenme ve momentum katsayısının belirlenmesi: Öğrenme katsayısı daha öncede belirtildiği gibi ağırlıkların değişim miktarını belirlemektedir.

- Büyük değerler seçilirse, yerel çözümler arasında ağın dolaşması diğer bir deyişle osilasyon yaşaması mümkündür.
- Küçük değerler seçilirse öğrenme zamanı artmaktadır.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

- Momentum katsayısı, bir önceki iterasyon değişiminin belirli bir oranının yeni değişim miktarını etkilemesidir.

- Bu özellikle yerel çözüme takılan ağların bir sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacı ile geliştirilmiştir.

- Momentum katsayısı, bir önceki iterasyon değişiminin belirli bir oranının yeni değişim miktarını etkilemesidir.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Bu özellikle yerel çözüme takılan ağların bir sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacı ile geliştirilmiştir.

- Değerin küçük olması yerel çözümlerden kurtulmayı zorlaştırır.
- Çok büyük değerler ise bir çözüme ulaşmada sorunlar yaşanabilir.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Örneklerin ağa sunulması: İki farklı yaklaşım söz konusudur.

i. **Sıralı Sunum:** Örnek setindeki bütün örneklerin ağa gösterilme şansı eşittir.

ii. **Rastgele Sunum:** Seçilen bir örnek tekrar set içerisine dahil edilip rastgele seçim yapılır. Örneklerin ağa gösterimi eşit değildir.

- Rastgele seçilen örnek eğitim setine tekrar dahil edilmez. Örneklerin ağa gösterilme şansı eşittir.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Ağırlıkların değiştirilme zamanı: İki farklı yaklaşım söz konusudur. Ağırlıkların değiştirilmesi öğrenme kuralına göre yapılmaktadır.

Genel olarak 3 durumda ağırlıkların değiştirilmesine izin verilmektedir.

- Her örnek ağa gösterildiğinde
- Belirli sayıda örnek gösterildiğinde
- Bütün örnek seti gösterildiğinde

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

Girdi ve Çıktıların ölçeklendirilmesi: Ölçeklendirme değişik şekillerde yapılmaktadır.

- i. Girdilerin Ölçeklendirilmesi: Öncelikli olarak girdi vektörü normalize edilir.

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

- Ardından örnekleri oluşturan değerler belirli bir aralık içerisine çekilirler.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

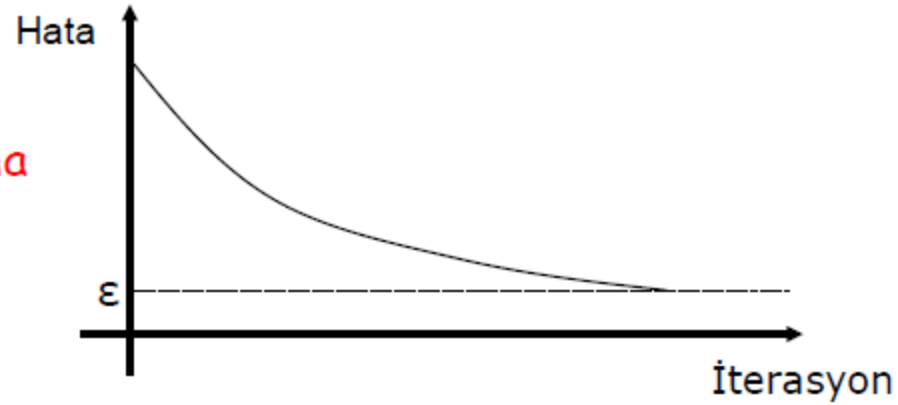
- Çıktıların ölçeklendirilmesi: Çıktıların ölçeklendirilmesi için kullanılan yöntemler girdilerin ölçeklendirilmesi için kullanılan yöntemlerle aynı olabilir.

Ağ öğrenme yaptıktan sonra ölçeklendirilmiş çıktı üreteceğinden ağ çıktıları dış dünyaya verilirken orijinal şekle dönüştürülmesi gerekir. Bunun için ölçeklendirme formülünün tersini kullanmak gerekir.

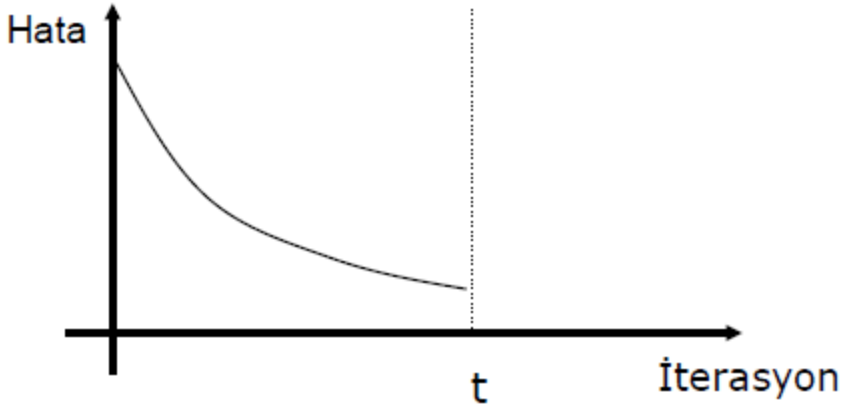
Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

- Durdurma kriterinin belirlenmesi:

Hatanın belli bir değerin altına düşmesi sonucu durma



Belirli sayıda iterasyondan sonra durma



Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

- Ara katmanların ve her ara katmandaki hücre elemanlarının sayısının belirlenmesi:
- Çok katmanlı ağ modelinde herhangi bir problem için kaç tane ara katman ve her ara katmanda kaç tane hücre elemanı kullanılması gerektiğini belirten bir yöntem yoktur.

- Bu konudaki çalışmalar deneme yanılma yönteminin etkin olduğunu göstermektedir. Bazı durumlarda başlangıçta bir ağ oluşturup zaman içinde büyütülerek veya küçültülerek istenilen ağa ulaşılır.

Çok Katmanlı Ağlarda Dikkat Edilmesi Gereken Noktalar

- Ağların büyütülmesi veya budanması:

Küçük bir ağdan başlayıp büyük bir ağa doğru eğitim esnasında sürekli hücre elemanlarını arttırmak.

Büyük bir ağdan başlayıp küçük bir ağa doğru eğitim sırasında sürekli ağı küçültmek ve hücre elemanlarını teker teker ağdan çıkarmak.

XOR Problemi ve Çözümü

- Tek katmanlı YSA'lar doğrusal problemlerin çözümü için uygun iken, doğrusal olmayan bazı problemlerin çözümü için uygun değildir. Buna en temel örnek ise XOR problemidir.

- XOR probleminin çözümünde geriye yayılımlı çok katmanlı ağlardan yararlanılır.

- Bu ağlar, danışmanlı öğrenme kuralları kullanılarak eğitilirler ve problem durumunu öğrenerek yeni problemlere çözüm yolları getirirler.