

## GÖRÜNTÜ İŞLEME-12.HAFTA

### YAPAY SİNİR AĞLARI

#### 1. Giriş

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; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Bu nedenledir ki, bu konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamış, daha sonraları bilgisayar sistemlerinin gelişimine de paralel olarak bir çok alanda kullanılır hale gelmiştir.

İnsan beyninin çalışma prensibini taklit ederek çalışan bu sistemler, her ne kadar bilgisayar teknolojisi hızlı bir gelişim göstermiş, işlem hızları nano saniyeler mertebesine inmiş olsa da, bırakalım insan beynini, ilkel bir canlı beyninin fonksiyonları dahi baz alındığında, böyle bir organizmanın yanında çok ilkel kalmaktadır. Nano saniyeler bazındaki işlem hızları ile YSA'lar, mili saniyeler mertebesindeki işlen hızları ile işlem yapan insan beyninin işlevselliğinin henüz çok uzağındadır.

Burada kısa bir hatırlatma yapmak gerekirse; insan beyninde yaklaşık  $10^{11}$  sinir hücresinin varlığından bahsedilmekle birlikte, bu sayının bilgisayar ortamında modellenmesi şu an için mümkün görünmemektedir. Fakat karar hızı açısından insan beyni ile henüz yarışamaları bile, YSA'lar yapısalıkları ve hassas eşleştirmelerin başarı ile gerçekleştirebilmeleri ile gün geçtikçe daha fazla uygulama alanı bulmaktadır.

#### 1.1 YSA'ların Avantajları

YSA'lar, uygulanan ağ modeline göre değişik karakteristik özellikler göstermelerine karşın temel birkaç ortak özelliğe sahiptirler.

1. YSA lar bir çok hücreden meydana gelir ve bu hücreler eş zamanlı olarak çalışarak karmaşık işlevleri yerine getirir. Diğer bir deyişle karmaşık işlevler bir çok hücrenin eş zamanlı çalışması ile meydana getirilir. Süreç içerisinde bu hücrelerden her hangi biri işlevini yitirse dahi sistem güvenli bir şekilde çalışmasına devam edebilir.
2. Eğitim esnasında kullanılan nümerik bilgilerden, problemin genel özellikleri elde etmesi ve böylelikle eğitim sırasında kullanılmayan girdiler için de, anlamlı yanıtlar üretebilmesidir.
3. Yapı üzerinde dağılmış belli tipteki non-lineer alt birimler, non lineer problemlerin de çözümünü mümkün kılmaktadır.
4. YSA'lar makina öğrenmesi gerçekleştirebilirler. Yapay sinir ağlarının temel işlevi zaten bilgisayarın öğrenmesini sağlamaktır. Olayları öğrenerek benzer olaylar karşısında mantıklı kararlar verebilirler.
5. Bilgi işleme yöntemleri geleneksel programlamadan farklıdır. Bu nedenle geleneksel programlamanın getirdiği bir çok olumsuzluk ortadan kaldırılabilir
6. Bilgiler ağın tamamında saklanır. Geleneksel programlamada olduğu gibi bilgiler veri tabanları yada dosyalarda belli bir düzende tutulmaz, ağın tamamına yayılarak değerler ile ölçülen ağ bağlantılarında saklanmaktadır. Hücrelerden bazılarının işlevini yitirmesi, anlamlı bilginin kaybolmasına neden olmaz.
7. Dağıtık belleğe sahiptirler. YSA'larda bilgi ağa dağılmış bir şekilde tutulur. Hücrelerin bağlantı ve ağırlık dereceleri, ağın bilgisini gösterir. Bu nedenle tek bir bağlantının kendi başına anlamı yoktur.
8. Örnekleri kullanarak öğrenirler. YSA'nın öğrenebilmesi için örneklerin belirlenmesi, bu örneklerin ağa gösterilerek istenen çıktılara göre ağın eğitilmesi gerekmektedir. Ağın başarısı, seçilen örnekler ile doğru orantılıdır, ağa olay bütün yönleri ile gösterilemezse ağ yanlış çıktılar üretebilir.
9. Daha önce görülmemiş örnekler hakkında bilgi üretebilirler. YSA'lar eğitimleri sırasında kendilerine verilen örneklerden genellemeler çıkarırlar ve bu genellemeler ile yeni örnekler hakkında bilgi üretebilirler.
10. Algılamaya yönelik olaylarda kullanılabilirler. YSA'ların en başarılı oldukları alanlar, algılamaya yönelik uygulama alanlarıdır. Bu alanlarda başarıları kanıtlanmıştır.
11. Örüntü (pattern) ilişkilendirme ve sınıflandırma yapabilirler. YSA'lar kendilerine örnekler halinde verilen örüntüleri kendisi veya diğerleri ile ilişkilendirebilir. Ayrıca kendisine verilen örneklerin kümelenmesi ile, bir sonraki verinin hangi kümeye dahil olacağını karar verilmesi konusunda kullanılabilirler.
12. Örüntü tamamlama yapabilirler. Ağ eksik bilgileri içeren örüntüler verildiğinde eksik bilgilerin tamamlanması konusunda başarılıdırlar.
13. Kendi kendine öğrenebilme ve organize etme yetenekleri vardır. YSA'lar online olarak öğrenebilirler ve kendi kendilerini eğitebilirler.

14. Eksik bilgi ile çalışabilmektedirler. Geleneksel sistemlerin aksine YSA'lar eğitildikten sonra veriler eksik bilgi içerse dahi, çıktı üretebilirler. Bu durum bir performans kaybı yaratmaz, performans kaybı eksik bilginin önemine bağlıdır. Burada bilgilerin önem dereceleri eğitim sırasında öğrenilir.
15. Hata toleransına sahiptirler. YSA'ların eksik bilgilerle çalışabilmeleri ve bazı hücreleri bozulsa dahi çalışabilmeleri, onları hatalara karşı toleranslı yapar.
16. Dereceli bozulma (Graceful degradation) gösterirler. Bir ağ, zaman içerisinde yavaş ve göreceli bir bozulmaya uğrar. Ağlar problemin ortaya çıktığı anda hemen bozulmazlar.

## 1.2. YSA'ların Dezavantajları

YSA'ların, pek çok avantajın yanında bazı dezavantajları da vardır. Belli başlı dezavantajları;

1. Donanım bağımlıdır. YSA'ların en önemli sorunu donanım bağımlı olmalarıdır. YSA'ların en önemli özellikleri ve var oluş nedenlerinden birisi olan paralel işlem yapabilme yeteneği, paralel çalışan işlemciler ile performans gösterir.
2. Uygun ağ yapısının belirlenmesinde belli bir kural yoktur. YSA'larda probleme uygun ağ yapısının belirlenmesi için geliştirilmiş bir kural yoktur. Uygun ağ yapısı deneyim ve deneme yanılma yolu ile belirlenmektedir.
3. Ağın parametre değerlerinin belirlenmesinde de belli bir kural yoktur. YSA'larda öğrenme katsayısı, hücre sayısı, katman sayısı gibi parametrelerin belirlenmesinde belirli bir kural yoktur. Bu değerlerin belirlenmesi için belirli bir standart olmamakla birlikte her problem için farklı bir yaklaşım söz konusu olabilmektedir.
4. Öğrenilecek problemin ağa gösterimi önemli bir problemdir. YSA'lar nümerik bilgiler ile çalışabilmektedirler. Problemler YSA'lara tanıtılmadan önce nümerik değerlere çevrilmek zorundadırlar. Burada belirlenecek gösterim mekanizması ağın performansını doğrudan etkileyecektir. Bu da kullanıcının yeteneğine bağlıdır.
5. Ağın eğitiminin ne zaman bitirilmesi gerektiğine ilişkin belli bir yöntem yoktur. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlandığı anlamına gelmektedir. Burada optimum neticeler veren bir mekanizma henüz yoktur ve YSA ile ilgili araştırmaların önemli bir kolunu oluşturmaktadır.
6. Ağın davranışları açıklanamamaktadır. Bu sorun YSA'ların en önemli sorunudur. YSA bir probleme çözüm ürettiği zaman, bunun neden ve nasıl olduğuna ilişkin bir ipucu vermez. Bu durum ağa olan güveni azaltıcı bir unsurdur.

## 1.3. Geleneksel Algoritmalar ile YSA'ların Karşılaştırılması

Geleneksel Algoritmalar	Yapay Sinir Ağları
Çıktılar, koyulan kurallara girişlerin uygulanması ile elde edilir.	Öğrenme esnasında giriş çıkış bilgileri verilerek, kurallar koyulur.
Hesaplama; merkezi, eş zamanlı ve ardışıldır.	Hesaplama; toplu, eş zamansız ve öğrenmeden sonra paraleldir.
Bellek paketlenmiş ve hazır bilgi depolanmıştır.	Bellek ayrılmış ve ağa yayılmıştır.
Hata toleransı yoktur.	Hata toleransı vardır.
Nisbeten hızlıdır.	Yavaş ve donanıma bağımlıdır.
Bilgiler ve algoritmalar kesindir.	Deneyimden yararlanır.

## 1.4. YSA'ların Kullanıldığı Alanlar

Yapay sinir ağları başlıca; Sınıflandırma, Modelleme ve Tahmin uygulamaları olmak üzere, pek çok alanda kullanılmaktadır. Başarılı uygulamalar incelendiğinde, YSA'ların

- çok boyutlu,
- gürültülü,
- karmaşık,
- kesin olmayan,
- eksik,
- kusurlu,
- hata olasılığı yüksek sensör verilerinin olması ve
- problemi çözmek için matematiksel modelin ve algoritmaların bulunmadığı,
- sadece örneklerin var olduğu durumlarda

yaygın olarak kullanıldıkları görülmektedir. Bu amaçla geliştirilmiş ağlar genellikle şu fonksiyonları gerçekleştirmektedirler;

Muhtemel fonksiyon kestirimleri, Sınıflandırma, ilişkilendirme veya örüntü eşleştirme, Zaman serileri analizleri, Sinyal filtreleme, Veri sıkıştırma, Örüntü tanıma, Doğrusal olmayan sinyal işleme, Doğrusal olmayan sistem modelleme, Optimizasyon, Kontrol

YSA'lar pek çok sektörde değişik uygulama alanları bulmuştur. Bunlardan bazıları;

Uzay: Uçuş simülasyonları, otomatik pilot uygulamaları, komponentlerin hata denetimleri vs.

Otomotiv: Otomatik yol izleme, rehber, garanti aktivite analizi, yol koşullarına göre sürüş analizi vs.

Bankacılık: Kredi uygulamaları geliştirilmesi, müşteri analizi ve kredi müracaat değerlendirilmesi, bütçe yatırım tahminleri vs.

Savunma: Silah yönlendirme, hedef seçme, radar, sensör sonar sistemleri, sinyal işleme, görüntü işleme vs.

Elektronik: Kod sırası öngörüsü, çip bozulma analizi, non-lineer modelleme vs.

Eğlence: Animasyonlar, özel efektler, pazarlama öngörüsü vs.

Finans: Kıymet biçme, pazar performans analizi, bütçe kestirimi, hedef belirleme vs.

Sigortacılık: ürün optimizasyonu, uygulama politikası geliştirme vs.

Üretim: üretim işlem kontrolü, ürün dizaynı, makina yıpranmalarının tespiti, dayanıklılık analizi, kalite kontrolü, iş çizelgeleri hazırlanması vs.

Sağlık: göğüs kanseri erken teşhis ve tedavisi, EEG, ECG, MR, kalite artırımı, ilaç etkileri analizi, kan analizi sınıflandırma, kalp krizi erken teşhis ve tedavisi vs.

Petro kimya: arama, verim analizi vs.

Robotik: yörünge kontrol, forklift robotları, görsel sistemler, uzaktan kumandalı sistemler, optimum rota belirleme vs.

Dil: sözcük tanıma, yazı ve konuşma çevrimi, dil tercüme vs.

Telekomünikasyon: görüntü ve data karşılaştırma, filtreleme, eko ve gürültü önümlendirilmesi, ses ve görüntü işleme, trafik yoğunluğunun kontrolü ve anahtarlama vs.

Güvenlik: parmak izi tanıma, kredi kartı hileleri saptama, retina tarama, yüz eşleştirme vs.

Bu örnekler çoğaltılabilir. Görüldüğü gibi YSA'lar günlük hayatımızda farkında olmadığımız pek çok alanda kullanılmaktadır. Gün geçtikçe uygulama alanları genişlemekte ve gelişmektedir.

### 1.5. Yapay Sinir Ağlarının Sınıflandırılması

Yapay sinir ağları işleyiş olarak benzer olmalarına rağmen herhangi bir tasarım ve işleyiş standardı bulunmamaktadır. Nöron dizilimlerine, nöronların ağırlıklarının düzenleme için yapılan hesaplamaların türüne ve zamanına göre yapay sinir ağlarını üç ayrı dalda inceleyebiliriz.

#### Yapılarına Göre Yapay Sinir Ağları

Yapay sinir ağları içerdiği nöronların birbirine bağlantı şekline göre ileri ve geri beslemeli olarak ikiye ayrılır.

**İleri Beslemeli Ağlar**: İleri beslemeli ağlarda nöronlar girişten çıkışa doğru düzenli katmanlar şeklindedir. Bir katmandan sadece kendinden sonraki katmanlara bağ bulunmaktadır. Yapay sinir ağına gelen bilgiler giriş katmanına daha sonra sırasıyla ara katmanlardan ve çıkış katmanından işlenerek geçer ve daha sonra dış dünyaya çıkar.

**Geri Beslemeli Yapay Sinir Ağları**: Geri beslemeli yapay sinir ağlarında ileri beslemeli olanların aksine bir hücrenin çıktısı sadece kendinden sonra gelen hücrenin katmanına girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir hücreye de girdi olarak bağlanabilir.

Bu yapı ile geri beslemeli yapay sinir ağları doğrusal olmayan dinamik bir davranış göstermektedir. Geri besleme özelliğini kazandıran bağlantıların bağlantı şekline göre geri aynı yapay sinir ağıyla farklı davranışta ve yapıda geri beslemeli yapay sinir ağları elde edilebilir.

#### Öğrenme Algoritmalarına Göre Yapay Sinir Ağları

Yapay sinir ağlarının verilen girdilere göre çıktı üretebilmesinin yolu ağı öğrenmesidir. Bu öğrenme işleminin de birden fazla yöntemi vardır. Yapay sinir ağları öğrenme algoritmalarına göre danışmanlı, danışmansız ve takviyeli öğrenme olarak üçe ayrılır.

**Danışmanlı Öğrenme**: Danışmanlı öğrenme sırasında ağa verilen giriş değerleri için çıktı değerleri de verilir. Ağ verilen girdiler için istenen çıkışları oluşturabilmek için kendi ağırlıklarını günceller. Ağı çıktılar ile beklenen çıktılar arasındaki hata hesaplanarak ağı yeni ağırlıkları bu hata payına göre düzenlenir.

Hata payı hesaplanırken ağı bütün çıktıları ile beklenen çıktıları arasındaki fark hesaplanır ve bu farka göre her hücreye düşen hata payı bulunur. Daha sonra her hücrenin kendine gelen ağırlıkları günceller.

**Danışmansız Öğrenme**: Danışmansız öğrenmede ağ öğrenme sırasında sadece örnek girdiler verilmektedir. Herhangi bir beklenen çıktı bilgisi verilmez. Girişte verilen bilgilere göre ağ her bir örneği kendi arasında sınıflandıracak şekilde kendi

kurallarını oluşturur. Ağ bağlantı ağırlıklarını aynı özellikte olan dokuları ayırabilecek şekilde düzenleyerek öğrenme işlemini tamamlar.

**Destekleyici Öğrenme:** Bu öğrenme yaklaşımında ağın her iterasyonu sonucunda elde ettiği sonucun iyi veya kötü olup olmadığına dair bir bilgi verilir. Ağ bu bilgilere göre kendini yeniden düzenler. Bu sayede ağ herhangi bir girdi dizisiyle hem öğrenerek hem de sonuç çıkararak işlemeye devam eder.

Örneğin satranç oynayan bir yapay sinir ağı yaptığı hamlenin iyi veya kötü olduğunu anlık olarak ayırt edememesine rağmen yine de hamleyi yapar. Eğer oyun sonuna geldiğinde program oyunu kazandıysa yaptığı hamlelerin iyi olduğunu varsayacaktır ve bundan sonraki oyunlarında benzer hamleleri iyi olarak değerlendirerek oynayacaktır.

### Öğrenme Zamanına Göre Yapay Sinir Ağları

Yapay sinir ağları öğrenme zamanına göre de statik ve dinamik öğrenme olarak ikiye ayrılır.

**Statik Öğrenme:** Statik öğrenme kuralıyla çalışan yapay sinir ağları kullanmadan önce eğitilmektedir. Eğitim tamamlandıktan sonra ağı istenilen şekilde kullanılabilir. Ancak bu kullanım sırasında ağın üzerindeki ağırlıklarda herhangi bir değişiklik olmaz.

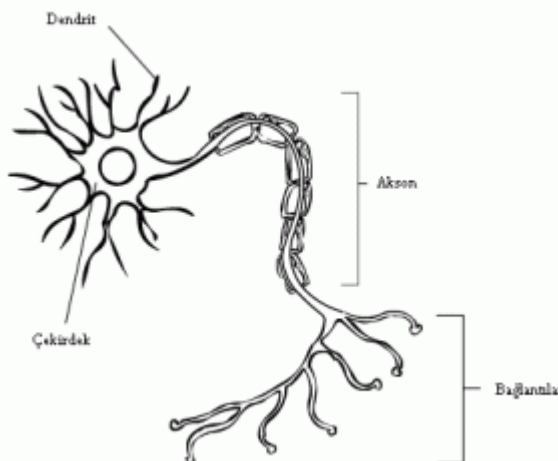
**Dinamik Öğrenme:** Dinamik öğrenme kuralı ise yapay sinir ağlarının çalıştığı süre boyunca öğrenmesini öngörerek tasarlanmıştır. Yapay sinir eğitim aşaması bittikten sonra da daha sonraki kullanımlarında çıkışların onaylanmasına göre ağırlıklarını değiştirerek çalışmaya devam eder.

## 2. YSA'nın Yapısı

### 2.1. Biyolojik Sinir Hücresinin Yapısı

Bir insanın beyinde yaklaşık olarak 10 milyar sinir hücresi ve bu nöronların birbirleriyle yaptığı bağlantı sayısının ise 60 trilyon olduğu tahmin edilmektedir. Bu sinirler girdi bilgilerini duyu organlarından alırlar. Daha sonra alıcı (taşıyıcı) sinirler bu sinyalleri işleyip bir sonraki sinire aktararak sinyalin merkezi sinir sistemine kadar ulaşmasını sağlar. Merkezi sinir sistemi bu sinyalleri alıp yorumladıktan sonra tepki sinyallerini üretir. Bu sinyaller de tepkilerin oluşacağı organlara tepki sinirleri vasıtasıyla iletilir. Bu sayede duyu organlarından gelen bilgilere karşı tepki organlarına uygun işaretler sinir sistemi vasıtasıyla yollar.

Yapay sinir ağları biyolojik sinir ağlarının modellemesi olduğu için, öncelikle biyolojik sinir sisteminin yapısına bakmak gerekir. Biyolojik sinir sisteminin temel yapı taşı olan nöronların yapısı dört ana bölümden oluşmaktadır; dendrit, akson, çekirdek ve bağlantılar. Dendritlerin sinir hücresinin ucunda bulunan ve ağaç kökü görünümüne sahip bir yapıya sahiptir. Dendritlerin görevi bağlı olduğu diğer nöronlardan veya duyu organlarından gelen sinyalleri çekirdeğe iletmektir. Çekirdek dendrit tarafından gelen sinyalleri bir araya toplayarak ve aksona iletir. Toplanan bu sinyaller akson tarafından işlenerek nöronun diğer ucunda bulunan bağlantılara gönderilir. Bağlantılar ise yeni üretilen sinyalleri diğer nöronlara iletir.



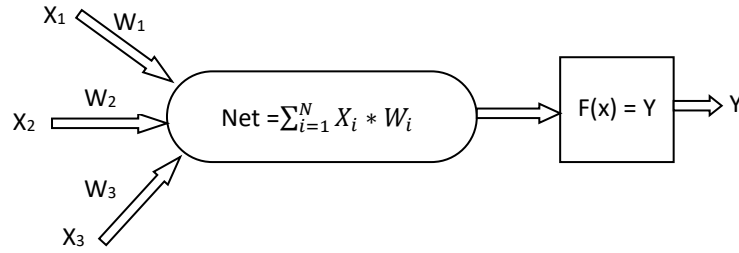
Şekil 1. Sinir hücresi ve Sinir sisteminin çalışma prensibi.

### 2.2. Yapay Sinir Hücresinin Yapısı

Yapay sinir hücreleri de biyolojik sinir hücrelerine benzer yapıdadır. Yapay nöronlar da aralarında bağ kurarak yapay sinir ağlarını oluştururlar. Aynı biyolojik nöronlarda olduğu gibi yapay nöronların da giriş sinyallerini aldıkları, bu sinyalleri toplayıp işledikleri ve çıktılarını ilettikleri bölümleri bulunmaktadır.

Bir yapay sinir hücresi beş bölümden oluşmaktadır;

- Girdiler
- Ağırlıklar
- Toplama Fonksiyonu (Birleştirme Fonksiyonu)
- Aktivasyon fonksiyonu
- Çıktılar



Şekil 2. Yapay Sinir Hücresinin yapısı.

**Girdiler:** Girdiler nöronlara gelen verilerdir. Girdiler yapay sinir hücresine bir diğer hücreden gelebileceği gibi direk olarak dış dünyadan da gelebilir. Bu girdilerden gelen veriler biyolojik sinir hücrelerinde olduğu gibi toplanmak üzere nöron çekirdeğine gönderilir.

**Ağırlıklar:** Yapay sinir hücresine gelen bilgiler girdiler üzerinden çekirdeğe ulaşmadan önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe iletilir. Bu sayede girdilerin üretilen çıktı üzerindeki etkisi ayarlanabilmektedir. Bu ağırlıkların değerleri pozitif, negatif veya sıfır olabilir. Ağırlığı sıfır olan girdilerin çıktı üzerinde herhangi bir etkisi olmamaktadır.

**Toplama Fonksiyonu (Birleştirme Fonksiyonu):** Toplama fonksiyonu bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri toplayarak o hücrenin net girdisini hesaplayan bir fonksiyondur.

Bazı durumlarda gelen girdilerin değeri dikkate alınırken bazı durumlarda ise gelen girdilerin sayısı önemli olabilmektedir. Bir problem için en uygun toplama fonksiyonu belirlenirken geliştirilmiş bir yöntem yoktur. Genellikle deneme yanılma yoluyla toplama fonksiyonu belirlenmektedir. Bazen her hücrenin toplama fonksiyonunun aynı olması gerekmez. Bu konulara karar vermek tasarımcıya aittir.

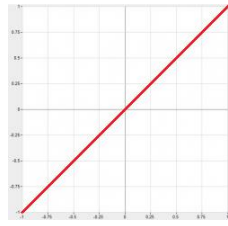
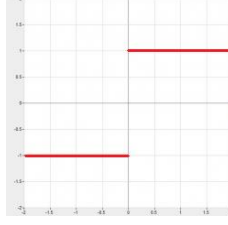
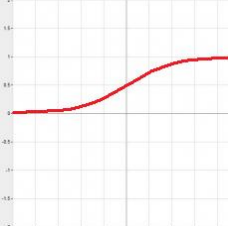
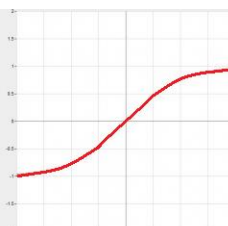
Tablo 1. Bazı Toplama Fonksiyonları

Toplam $Net = \sum_{i=1}^N X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $Net = \prod_{i=1}^N X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $Net = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $Net = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $Net = \sum_{i=1}^N \text{Sgn}(X_i * W_i)$	n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kümülatif Toplam $Net = \text{Net}(\text{eski}) + \sum_{i=1}^N X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

**Aktivasyon Fonksiyonu:** Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Aktivasyon fonksiyonu genellikle doğrusal olmayan bir fonksiyon seçilir. Yapay sinir ağlarının bir özelliği olan

“doğrusal olmama” aktivasyon fonksiyonlarının doğrusal olmama özelliğinden gelmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta ise fonksiyonun türevinin kolay hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplamaların yavaşlamaması için türevi kolay hesaplanır bir fonksiyon seçilir. Günümüzde en yaygın olarak kullanılan “Çok katmanlı algılayıcı” modelinde genel olarak aktivasyon fonksiyonu olarak “Sigmoid fonksiyonu” kullanılır.

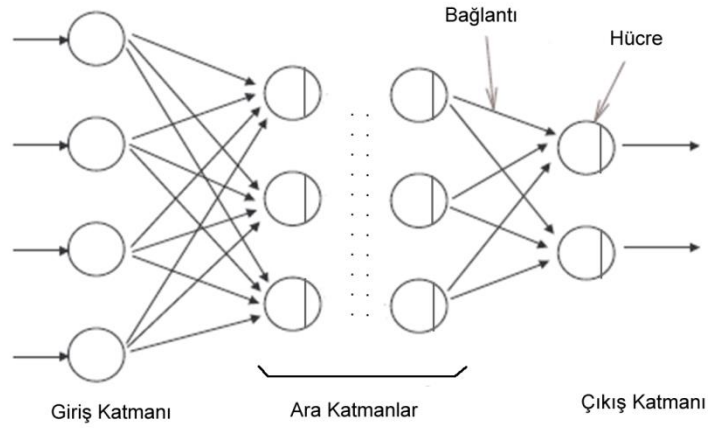
Tablo 2. Aktivasyon Fonksiyonları

Doğrusal (Linear) Aktivasyon Fonksiyonu		$F(Net) = A * Net$ (A sabit bir sayı)	Doğrusal problemler çözmek amacıyla aktivasyon fonksiyonu doğrusal bir fonksiyon olarak seçilebilir. Toplama fonksiyonundan çıkan sonuç, belli bir katsayı ile çarpılarak hücrenin çıktısı olarak hesaplanır.
Adım (Step) Aktivasyon Fonksiyonu		$F(Net) = \begin{cases} 1 & \text{if } Net > \text{Eşik Değer} \\ 0 & \text{if } Net \leq \text{Eşik Değer} \end{cases}$	Gelen Net girdinin belirlenen bir eşik değerinin altında veya üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerini alır.
Sigmoid Aktivasyon Fonksiyonu		$F(Net) = \frac{1}{1 + e^{-Net}}$	Sigmoid aktivasyon fonksiyonu sürekli ve türevi alınabilir bir fonksiyondur. Doğrusal olmayışı dolayısıyla yapay sinir ağı uygulamalarında en sık kullanılan fonksiyondur. Bu fonksiyon girdi değerlerinin her biri için 0 ile 1 arasında bir değer üretir.
Tanjant Hiperbolik Aktivasyon Fonksiyonu		$F(Net) = \frac{e^{Net} + e^{-Net}}{e^{Net} - e^{-Net}}$	Tanjant hiperbolik fonksiyonu, sigmoid fonksiyonuna benzer bir fonksiyondur. Sigmoid fonksiyonunda çıkış değerleri 0 ile 1 arasında değişirken hiperbolik tanjant fonksiyonunun çıkış değerleri -1 ile 1 arasında değişmektedir.
Eşik Değer Fonksiyonu		$F(Net) = \begin{cases} 0 & \text{if } Net \leq 0 \\ Net & \text{if } 0 < Net < 1 \\ 1 & \text{if } Net \geq 1 \end{cases}$	Gelen bilgilerin 0 dan küçük-eşit olduğunda 0 çıktısı, 1 den büyük-eşit olduğunda 1 çıktısı, 0 ile 1 arasında olduğunda ise yine kendisini veren çıktılar üretilebilir.
Sinüs Aktivasyon Fonksiyonu		$F(Net) = \sin(Net)$	Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır.

**Hücrenin Çıktısı:** Aktivasyon fonksiyonundan çıkan değer hücrenin çıktı değeri olmaktadır. Bu değer ister yapay sinir ağının çıktısı olarak dış dünyaya verilir isterse tekrardan ağın içinde kullanılabilir. Her hücrenin birden fazla girdisi olmasına rağmen bir tek çıktısı olmaktadır. Bu çıktı istenilen sayıda hücreye bağlanabilir.

### 2.3. Yapay Sinir Ağının Yapısı

Yapay sinir ağları yapay sinir hücrelerinin birbirine bağlanmasıyla oluşan yapılardır. Yapay sinir ağları üç ana katmanda incelenir; Giriş Katmanı, Ara (Gizli) Katmanlar ve Çıkış Katmanı.



Şekil 3. Yapay Sinir Ağı'nın Yapısı.

**Giriş Katmanı:** Yapay sinir ağına dış dünyadan girdilerin geldiği katmandır. Bu katmanda dış dünyadan gelecek giriş sayısı kadar hücrenin bulunmasına rağmen genelde girdiler herhangi bir işleme uğramadan alt katmanlara iletilmektedir.

**Ara (Gizli) Katman(lar):** Giriş katmanından çıkan bilgiler bu katmana gelir. Ara katman sayısı ağdan ağa değişebilir. Bazı yapay sinir ağlarında ara katman bulunmadığı gibi bazı yapay sinir ağlarında ise birden fazla ara katman bulunmaktadır. Ara katmanlardaki nöron sayıları giriş ve çıkış sayısından bağımsızdır. Birden fazla ara katman olan ağlarda ara katmanların kendi aralarındaki hücre sayıları da farklı olabilir. Ara katmanların ve bu katmanlardaki nöronların sayısının artması hesaplama karmaşıklığını ve süresini arttırmasına rağmen yapay sinir ağının daha karmaşık problemlerin çözümünde de kullanılabilmesini sağlar.

**Çıkış Katmanı:** Ara katmanlardan gelen bilgileri işleyerek ağın çıktılarını üreten katmandır. Bu katmanda üretilen çıktılar dış dünyaya gönderilir. Geri beslemeli ağlarda bu katmanda üretilen çıktı kullanılarak ağın yeni ağırlık değerleri hesaplanır.

### 3. Yapay Sinir Ağlarının Eğitilmesi

İnsan beyni doğumdan sonraki gelişme sürecinde çevresinden duyu organlarıyla algıladığı davranışları yorumlar ve bu bilgileri diğer davranışlarında kullanır. Yaşadıkça beyin gelişir ve tecrübelenir. Artık olaylar karşısında nasıl tepki göstereceğini çoğu zaman bilmektedir. Fakat hiç karşılaşmadığı bir olay karşısında yine tecrübesiz kalabilir. Yapay sinir ağlarının öğrenme sürecinde de dış ortamdan girişler alınır, aktivasyon fonksiyonundan geçirilerek bir tepki çıkışı üretilir. Bu çıkış yine tecrübeyle verilen çıkışla karşılaştırılarak hata bulunur.

Çeşitli öğrenme algoritmalarıyla hata azaltılıp gerçek çıkışa yaklaşılmaya çalışılır. Bu çalışma süresince yenilenen yapay sinir ağının ağırlıklarıdır. Ağırlıklar her bir çevrimde yenilenerek amaca ulaşılmaya çalışılır. Amaca ulaşmanın veya yaklaşmanın ölçüsü de yine dışarıdan verilen bir değerdir. Eğer yapay sinir ağı verilen giriş-çıkış çiftleriyle amaca ulaşmış ise ağırlık değerleri saklanır.

Ağırlıkların sürekli yenilenerek istenilen sonuca ulaşılan kadar geçen zamana öğrenme adı verilir. Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler verilip, sinir ağı çıkışıyla gerçek çıkış yaklaşımı incelenir. Eğer yeni verilen örneklere de doğru yaklaşıyorsa sinir ağı işi öğrenmiş demektir. Sinir ağına verilen örnek sayısı optimum değerden fazla ise sinir ağı işi öğrenmemiş, ezberlemiş demektir. Genelde eldeki örneklerin %80 ağı verilip ağı eğitilir. Daha sonra kalan %20 lik kısım verilip ağı davranışı incelenir. Böylece ağı testi yapılmış olur.

**1. Örneklerin toplanması:** Ağı öğrenmesi istenilen olay için daha önce gerçekleşmiş örneklerin bulunması adıdır. Ağı eğitilmesi için örnekler toplandığı gibi (eğitim seti) ağı test edilmesi için de örneklerin (test seti) toplanması gerekmektedir. Eğitim setindeki örnekler tek tek gösterilerek ağı olayı öğrenmesi sağlanır. Ağı olayı öğrendikten sonra test setindeki örnekler gösterilerek ağı performansı ölçülür. Hiç görmediği örnekler karşısındaki başarısı ağı iyi öğrenip öğrenmediğini ortaya koyar.

**2. Ağı topolojik yapısının belirlenmesi:** Öğrenilmesi istenen olay için oluşturulacak olan ağı topolojik yapısı belirlenir. Kaç tane girdi ünitesi, kaç tane ara katman, her ara katmanda kaç tane proses eleman kaç tane çıktı eleman olması gerektiği bu adımda belirlenmektedir.

**3. Öğrenme parametrelerinin belirlenmesi:** Ağı 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. Başlangıç genellikle rasgele değerler atanır. Daha sonra ağı uygun değerleri öğrenme sırasında kendisi belirler.



**5. Öğrenme setinden örneklerin seçilmesi ve ağa gösterilmesi:** Ağın öğrenmeye başlaması ve Öğrenme kuralına uygun olarak ağırlıkları değiştirmesi için ağa örnekler belirli bir düzeneğe göre gösterilir.

**6. Öğrenme sırasında ileri hesaplamaların yapılması:** Sunulan girdi için ağın çıktı değerleri hesaplanır.

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

**8. Ağırlıkların değiştirilmesi:** Geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

**9. Öğrenmenin tamamlanması:** İleri beslemeli sinir ağı öğrenmeyi tamamlayınca, yani gerçekleşen ile beklenen çıktılar arasındaki hatalar kabul edilir düzeye ininceye kadar devam eder.

Ağın kendisine gösterilen girdi örneği için beklenen çıktıyı üretmesini sağlayacak ağırlık değerleri başlangıçta rastgele atanmakta ve ağa örnekler gösterildikçe ağırlıklar değiştirilerek istenen değerlere ulaşması sağlanmaktadır. İstenen ağırlık değerlerinin ne olduğu bilinmemektedir. Bu nedenle YSA'nın davranışlarını yorumlamak ve açıklamak mümkün olmaz.

Bazı durumlarda ağın takıldığı yer hata düzeyinin üstünde kalabilir. Bu durumda ağın olayı öğrenmesi için bazı değişiklikler yapılarak yeniden eğitilmesi gerekir. Bunlar; Başlangıç değerlerinde değişiklik yapılabilir. Ağ topolojisinde değişiklikler yapılabilir. Ağın parametrelerinde değişiklikler yapılabilir. Ağa sunulan verilerin gösterimi ve örneklerin formülasyonu değiştirilerek yeni örnek seti oluşturulabilir. Öğrenme setindeki örneklerin sayısı artırılabilir veya azaltılabilir.

İleri beslemeli sinir ağının yerel sonuçlara takılıp kalmaması için momentum katsayısı geliştirilmiştir. Ağların 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 sebep 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 sürmektedir.

Ağın öğrenmesinin gösterilmesinin en güzel yolu hata grafiğini çizmektir. Her iterasyonda oluşan hatanın grafiği çizilirse hatanın zaman içinde düştüğü gözlemlenebilir. Belirli bir iterasyondan sonra hatanın daha fazla azalmayacağı görülür. Bu ağın öğrenmesinin durduğu ve daha iyi bir sonuç bulunamayacağı anlamına gelir. Eğer elde edilen çözüm kabul edilemez ise o zaman ağ yerel bir çözüme takılmış demektir.

### Katsayıların Belirlenmesi

İleri beslemeli sinir ağında bağlantıların **ağırlık değerlerinin** başlangıçta belirlenmesi ağın performansını yakından ilgilendirmektedir. Genel olarak ağırlıklar belirli aralıklarda atanmaktadır. Bu aralık eğer büyük tutulursa yerel çözümler arasında sürekli dolaştığı küçük olması durumunda ise öğrenmenin geç gerçekleştiği görülmüştür. Bu değerlerin atanması için henüz standart bir yöntem yoktur. Tecrübeler **0.1 ile 1.0** arasındaki değerlerin başarılı sonuçlar ürettiğini göstermektedir. Fakat bu tamamen öğrenilmesi istenen problemin niteliğine bağlıdır.

Başlangıç değerleri kadar öğrenme ve momentum katsayılarının belirlenmesi de ağın öğrenme performansı ile yakından ilgilidir. **Öğrenme katsayısı ( $\lambda$ ) 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 göstermiştir. Bazı uygulamalar öğrenme katsayısının 0.6 değerini aldığı zaman en başarılı sonuçları verdiği göstermiştir. Bu durum tamamen probleme bağlıdır.

Benzer şekilde momentum katsayısı da öğrenmenin performansını etkiler. **Momentum katsayısı ( $\alpha$ ) bir önceki iterasyondaki değişimin belirli bir oranının yeni değişim miktarına eklenmesidir.** Bu özellikle yerel çözümlere takılan ağların sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacıyla ile önerilmiştir. Bu değerlerin küçük olması yerel çözümlerden kurtulmayı zorlaştırabilir. Çok büyük değerler ise çözüme ulaşmada sorunlar yaşatabilir. Tecrübeler momentum katsayısının **0.6-0.8** arasında seçilmesinin uygun olacağını göstermiştir. Problemin niteliğine göre kullanıcının almasında fayda vardır.



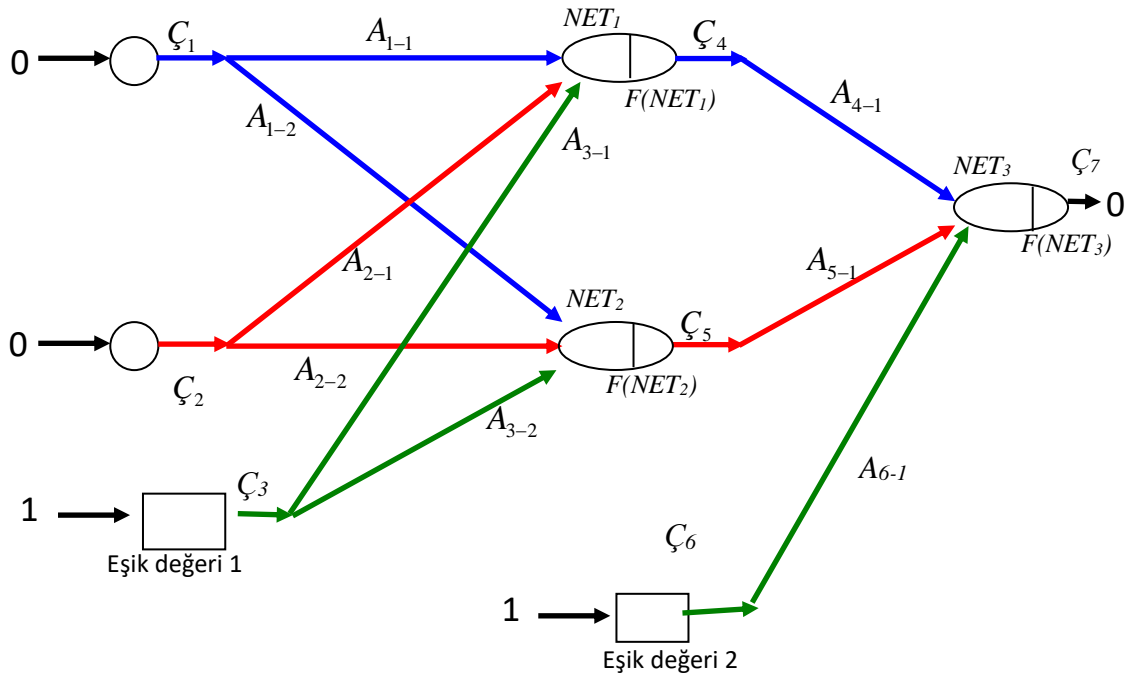
#### 4. Örnek Problem

Xor Problemini YSA ile çözen bir program yazınız ve çözüm adımlarını gösteriniz.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

**Çözüm:**

**İLERİ DOĞRU HESAPLAMA**



Rasgele alınan ağırlık (A) değerleri :

$$A_{1-1}= 0,129952 \quad A_{2-1}= -0,923123 \quad A_{3-1}= 0,341232 \quad A_{4-1}= 0,164732$$

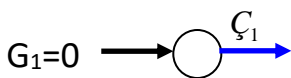
$$A_{1-2}= 0,570345 \quad A_{2-2}= -0,328932 \quad A_{3-2}= -0,115223 \quad A_{5-1}= 0,752621$$

$$A_{6-1}= -0,993423$$

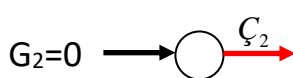
**Çözüm :**

İleriye doğru hesaplama yöntemi;

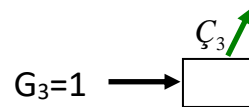
İlk giriş değerleri, ilk katman olan girdi katmanında çıkış değerlerine eşittir. Giriş katmanındaki çıkış değerlerini  $\check{C}_1$ ,  $\check{C}_2$  'i yi ve eşik değeri 1'i hesaplayalım



$$\check{C}_1 = G_1 \Rightarrow \check{C}_1 = 0$$

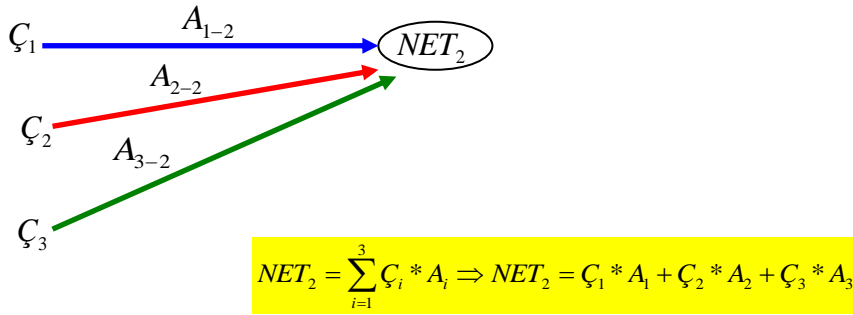
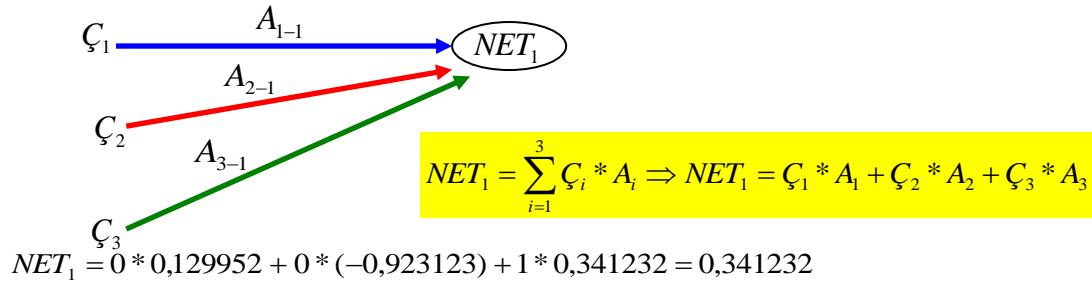


$$\check{C}_2 = G_2 \Rightarrow \check{C}_2 = 0$$



$$\check{C}_3 = G_3 \Rightarrow \check{C}_3 = 1$$

İkinci katman olan ara katmana giriş değerlerini hesaplamak için giriş katmanından gelen çıkış değerleri ile ağırlık değerlerin çarpımlarını toplayarak buluruz. Aşağıda iki ara katmanın giriş değerleri hesaplanmıştır.



Ara katmanların çıkış değerleri F(NET) formülüyle hesaplanmaktadır. İki ara katmanın çıkış değerlerini Ç4, Ç5 'i ve eşik değeri 2'yi hesaplayalım.



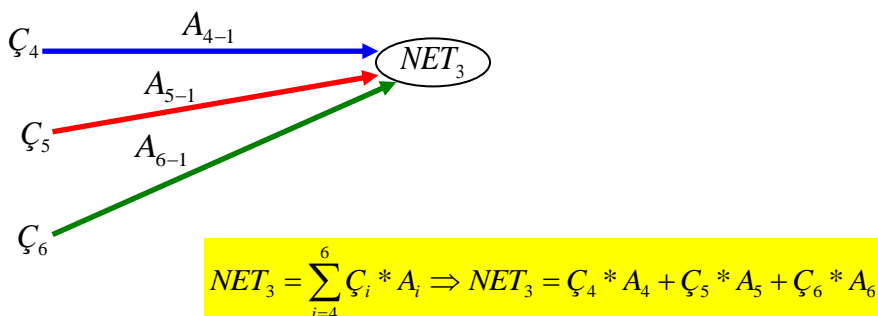
$$C_4 = F(NET_1) = \frac{1}{1 + e^{-NET_1}} = \frac{1}{1 + e^{-0,341232}} = 0,584490$$

$$C_5 = F(NET_2) = \frac{1}{1 + e^{-NET_2}} = \frac{1}{1 + e^{0,115223}} = 0,471226$$

Eşik değeri 2 hesaplanması

$$C_6 = G_6 \Rightarrow C_6 = 1$$

Üçüncü katman olan çıktı katmanının giriş değeri ara katmanın giriş değerinin hesaplanması ile aynıdır. Çıktı katmanının giriş değerini hesaplayalım;



$$NET_3 = 0,584490 * 0,164732 + 0,471226 * 0,752621 + 1 * (-0,993423) = -0,542484$$

Problemin çıkış değerini F(NET) formülü ile hesaplayalım;

$$\text{NET}_3 \xrightarrow{C_7} C_7 = F(NET_3) = \frac{1}{1 + e^{-NET_3}} = \frac{1}{1 + e^{542484}} = 0,367610$$

Beklenen çıktı 0 olduğuna göre **Ağın hatasını** hesaplayalım;  $E_m = 0 - 367610 = -0,367610$

Ağın hatası beklediğimizden büyük olduğundan şimdi geriye doğru hesaplama yöntemi kullanalım;

### GERİYE DOĞRU HESAPLAMA

Ara katman ile çıktı katmanı arasındaki ağırlıkların değişim miktarını hesaplayalım;

$$\text{NET}_3 \xrightarrow{C_7}$$

**Değişim Miktarı Formülü;**

(Değişim miktarı = ağın çıktısı \* (1-ağın çıktısı) \* hata miktarı)

$$\delta_m = C_7 (1 - C_7) * E_m$$

$$\delta_m = 0,367610 * (1 - 0,367610) * (-0,367610) = -0,085459$$

**Ağırlık Değişim Miktarı Formülü;**

(Ağırlık değişim miktarı = öğrenme katsayısı \* değişim miktarı \* ilk ünitenin çıktısı + momentum katsayısı \* bir önceki ağırlık değişim miktarı)

$$\Delta A_{jm}(t) = \lambda * \delta_m * C_{jm} + \alpha * \Delta A_{jm}(t-1)$$

$$\Delta A_{4-1}(t) = \lambda * \delta_m * C_4 + \alpha * \Delta A(t-1) = 0,5 * (-0,085459) * 0,584490 + 0,8 * 0$$

$$= -0,024875$$

$$\Delta A_{5-1}(t) = \lambda * \delta_m * C_5 + \alpha * \Delta A(t-1) = 0,5 * (-0,085459) * 0,471226 + 0,8 * 0$$

$$= -0,020135$$

$$\Delta A_{6-1}(t) = \lambda * \delta_m * C_6 + \alpha * \Delta A(t-1) = 0,5 * (-0,085459) * 1 + 0,8 * 0$$

$$= -0,042730$$

Ağırlıklardaki bu değişimlerden sonra ara katman ile çıktı katmanı arasındaki **yeni ağırlıkları** hesaplayalım ( $A_{4-1}$ ,  $A_{5-1}$ ,  $A_{6-1}$ );

(Yeni ağırlık değeri = Eski ağırlık değeri + Ağırlık değişim miktarı)

$$A_k(t) = A_k(t-1) + \Delta A_k(t)$$

$$A_{4-1}(t) = A_{4-1}(t-1) + \Delta A_{4-1}(t) = 0,164732 - 0,024875 = 0,139757$$

$$A_{5-1}(t) = A_{5-1}(t-1) + \Delta A_{5-1}(t) = 0,752621 - 0,020135 = 0,732486$$

$$A_{6-1}(t) = A_{6-1}(t-1) + \Delta A_{6-1}(t) = -0,993423 - 0,042730 = -1,036153$$

**Ağırlıklardaki değişim** miktarı ile şimdide girdi katmanı ile ara katman arasındaki ağırlıkları hesaplayalım.

Ara katmandaki hata oranları ve değişim miktarları;

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

$$\delta_4^a = \zeta_4^a (1 - \zeta_4^a) \sum_1 \delta_m A_{4-1}^a(t-1)$$

$$\delta_4^a = 0,584490 * (1 - 0,584490) * (-0,085459) * 0,164732 = -0,0034190$$

$$\delta_5^a = \zeta_5^a (1 - \zeta_5^a) \sum_1 \delta_m A_{5-1}^a(t-1)$$

$$\delta_5^a = 0,471226 * (1 - 0,471226) * (-0,085459) * 0,752621 = -0,0160263$$

$$\Delta A_{jm}(t) = \lambda * \delta_m * \zeta_{jm} + \alpha * \Delta A_{jm}(t-1)$$

$$\Delta A_{1-1}(t) = 0,5 * (-0,003419) * 0 + 0,8 * 0 = 0$$

$$\Delta A_{1-2}(t) = 0,5 * (-0,003419) * 0 + 0,8 * 0 = 0 \quad \text{DİKKAT: -0,0034 LÜ SAYI BURADA -0.016 ŞEKLİNDE OLMALI}$$

$$\Delta A_{2-1}(t) = 0,5 * (-0,016026) * 0 + 0,8 * 0 = 0$$

$$\Delta A_{2-2}(t) = 0,5 * (-0,016026) * 0 + 0,8 * 0 = 0 \quad \text{DİKKAT: -0,0034 LÜ SAYI BURADA -0.016 ŞEKLİNDE OLMALI}$$

$$\Delta A_{3-1}(t) = 0,5 * (-0,003419) * 1 + 0,8 * 0 = -0,0017095$$

$$\Delta A_{3-2}(t) = 0,5 * (-0,016026) * 1 + 0,8 * 0 = -0,0080132$$

Girdi katmanı ile Ara katman arasındaki ağırlıklar;

$$A_k(t) = A_k(t-1) + \Delta A_k(t)$$

$$A_{1-1} = 0,129952 + 0 = 0,129952$$

$$A_{1-2} = 0,570345 + 0 = 0,570345$$

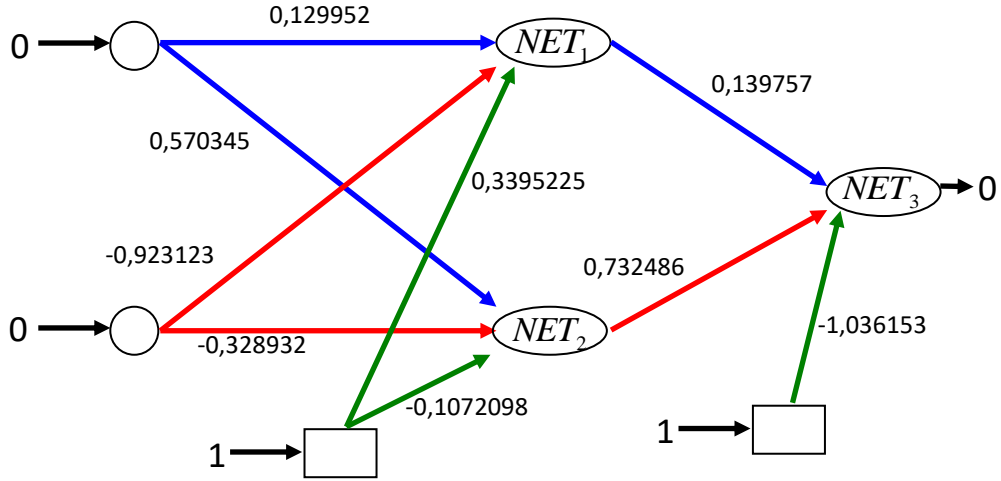
$$A_{2-1} = (-0,923123) + 0 = -0,923123$$

$$A_{2-2} = (-0,328932) + 0 = -0,328932$$

$$A_{3-1} = 0,341232 + (-0,0017095) = 0,3395225$$

$$A_{3-2} = -0,115223 + (-0,0080132) = -0,1072098$$

Birinci iterasyon bittikten sonraki problemin durumu aşağıdaki gibidir;



**Kaynak:** Burada kullanılan örnek (Prof. Dr. Ercan Öztemel, Yapay Sinir Ağları, İstanbul: Papatya Yayıncılık, 2012) kitabından alınmıştır.

YSA İŞLEMİ YAPAN GENEL BİR PROGRAM EKLENECEK.