KTO KARATAY ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS

EBMYL509 - YAPAY SİNİR AĞLARI

FİNAL PROJESİ

GERİYE YAYILIM ALGORİTMASI KULLANILARAK NXOR PROBLEMİNİN ÇÖZÜLMESİ

Ad - Soyad : İsa Alperen Saylar

Öğrenci No : 201300158

Bölüm : Elektrik ve Bilgisayar Mühendisliği Tezli Yüksek Lisans

Akademisyen : Prof. Dr. Novruz Allahverdi

EBMYL509 YAPAY SİNİR AĞLARI

Final Projesi Ödevi için notlar:

Proje raporu oluştururken aşağıdakilere dikkat edilecek işlemler:

1) Proje kapağı;
2) Proje ödevi; UZEM den indirilecek ve raporun 2. Sayfasını oluşturacaktır;
3) Proje sayfası – 20 sayfadan az olmayacaktır; Sisteme yükleme PDF dosyası halinde olacaktır.
4) İçindekiler;
5) Bölüm 1. Çalışmanın amacı, YSA hakkında bilgi (özellikle çok katmanlı YSA, Bu ağlarda geriye yayılmalı eğitim;
6) Bölüm 2. Çok katmanlı geriye yayılıma yöntemi, şekiller çizilecek ve formüller verilerek açıklanacak;
7) Bölüm 3. Geriye yayılım algoritması (http://farabi.sutef.gen.tr/ysa/index.html) ve (Ercan Öztemel-Yapay Sinir Ağları) Bölüm 3-4-5 vs.). Başka kitap, makale kaynaklarına ve internet kaynaklarına da bakmalısınız. Bu bölümde bu algoritmanın blok-şeması tertiplenecektir;
8) Bölüm 4. Geriye yayılım algoritmasının programı yazılacak (herhangi bir görsel dilde), programın bazı önemli çıktıları rapora dahil edilecektir;
9) Bölüm 5. Sonuç. Bu kısımda çalıştınız konuda ne yaptığınız anlatılacak, varsa darboğazlar onların tahlili verilecektir.
10) Kaynaklar. Proje metninde bütün kaynaklar kısmında verilmiş numaraya (Örnek: [Atıf no] şeklinde, Örn. [1], [3] atıf verilmelidir).
11) Tasarladığınız program herkes için aynı, NXOR (NOTXOR) fonksiyonu için uygulanacaktır (4
giriş, tek çıkış, yani ağa 4 örnek seti ayrı ayrı ardışık olarak gösterilecektir). Bunu örneklerle Bölüm 4'te göstermelisiniz.

Her bir öğrencinin kullanacağı veriler aşağıda verilmiştir:

Öğr. No	α	λ
201300153	0.2	0.65
201300154	0.3	0.75
201300158	0.25	0.6
201300159	0.95	0.55
201300162	0.9	0.4
201300163	0.8	0.6
201300164	0.7	0.3
201300167	0.5	0.45
201300286	0.4	0.5

Başarılar!

NOT 1: En son dersimiz 29.01.2021.
NOT 2: Cevap dosyanızın en son PDF formatında sisteme yükleme tarihi. 27.01.2021; Saat 23:59.

İÇİNDEKİLER

1. GİRİŞ

Hızla gelişen teknoloji, gündelik yaşantıyı oldukça etkilemektedir. Özellikle son yıllarda bilgisayar bilimlerinde, yazılım ve donanım alanında çok hızlı gelişmeler yaşanmaktadır. Bilgisayar bilimlerindeki bu ilerleme, insan gibi düşünen ve davranan sistemlerin geliştirilmesine yönelik olarak 1950'li yıllardan beri sürmektedir. Yapay zekâ olarak isimlendirilen bu alan, insan düşünme ve davranışlarını taklide yönelik olduğundan, nöroloji, psikoloji ve mühendislik gibi farklı disiplinleri kapsayan geniş bir alana yayılmıştır.

Yapay Sinir Ağları örnekler ile öğrenebilme ve genelleme yapabilme özellikleri onun tercih edilmesinde büyük katkı sağlamaktadır. Bugün birçok alanda kullanılıp başarılı olduğu saptanmıştır. Yapay Sinir Ağları, beynin çalışma ilkelerinin sayısal bilgisayarlar üzerinde taklit edilmesi fikri ile ortaya çıkmış ve ilk çalışmalara beyni oluşturan nöronların matematiksel olarak modellenmesi üzerinde yoğunlaşmıştır.

1.1 Projenin Amacı

Bu projenin amacı; çok katmanlı algılayıcıların yapısını ve öğrenmesini kullanarak NXOR problemini çözmektir. Çok katmanlı algılayıcılar modeli ile yapay sinir ağındaki istenen girdi ve çıktıları arasındaki ilişkileri öğrenmesini gerçekleştirmek mümkündür. Yapay sinir ağlarının bu tarz problemlerde ne kadar başarılı olduğunu gözlemlemek, yapay sinir ağlarının çalışma mantığını anlamak ve veriler arasındaki bağlantının ağın eğitimindeki önemini kavramak hedeflenmiştir.

1.2 Literatür Taraması

Bu çalışmada çok katmanlı algılayıcıları ağının yapısı ve öğrenme kuralı ile NXOR problemi çözümü yapılmaktadır. Oluşturulan proje çok katmanlı algılayıcıların temeli olup geliştirilerek başka çalışmalarda da kullanılmıştır.

Proje konusuyla ilgili önceden yapılmış çalışmalardan bazıları şunlardır;

- İleri Beslemeli Yapay Sinir Ağlarında Backprogation (Geriye Yayılım) Algoritmasının Sezgisel Yaklaşımı [1].
  Hazırlayanlar: Şahin Bayzan,
- Çok Katmanlı Perseptron Sinir Ağları ile Diyabet Hastalığının Teşhisi [2].
  Hazırlayanlar: İnan Güler
  Üniversite: Gazi Üniversitesi YSA Tabanlı Sistemler için Görsel Bir Arayüz Tasarımı
- Yapay Sinir Ağlarıyla Hisse Senedi Fiyatları ve Yönlerinin Tahmini [3].
  Hazırlayanlar: Muhammed Mustafa Tuncer Çalışkan

1.3 Yapay Sinir Ağlarının Tanımı

Yapay sinir ağları (Artificial Neural Network-ANN) insan beyninin sinir sistemine ve çalışma prensibine dayanan elektriksel bir modeldir. Bir anlamda insan beyninin ufak bir kopyası gibidir. İnsan beyninin öğrenme yoluyla yeni bilgiler üretebilme, keşfedebilme, düşünme ve gözlemlemeye yönelik yeteneklerini, yardım almadan yapabilen sistemler geliştirmek için tasarlanmışlardır, çünkü bu özellikleri, geleneksel programlama algoritmaları ile yaratabilmek imkânsızdır [4].

1.4 Yapay Sinir Ağlarının Kısa Tarihçesi

İlk yapay nöron, 1943 yılında nöropsikiyatrist Warren McCulloch ve bilim adamı Walter Pits tarafından üretilmiştir. Ancak dönemin kısıtlı olanakları nedeniyle, bu alanda çok gelişme sağlanamamıştır. Bundan sonra 1969'da Minsky ve Papert bir kitap yayınlayarak, yapay sinir ağları alanında duyulan etik kaygıları da ortadan kaldırmış ve bu yeni teknolojiye giden yolu açmışlardır. İlk gözle görülür gelişmeler ise 1990'lı yıllara dayanmaktadır [5].

1.5 Yapay Sinir Ağlarının Kullanım Alanları

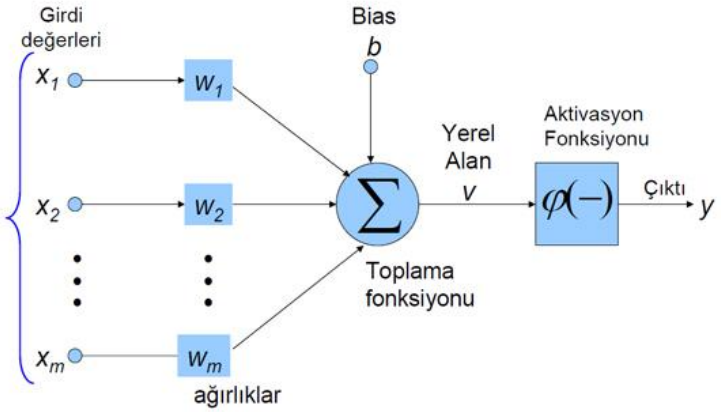
Yapay sinir ağları geniş kullanım alanına sahiptir. Güvenlik, otomotiv, bankacılık, sağlık gibi alanlarda kullanılmaktadır. Genel olarak şöyle sıralayabiliriz;

- **Otomotiv Sektörü:** Yol izleme, Rehberlik, Yol koşullarına göre sürüş analizi…
- **Bankacılık:** Kredi verme uygulamaları, Kredi kartı suçlarının tespiti, İmza tanıma…
- **Uzay Sanayisi:** Uçuş simülasyonları, Otomatik pilot uygulamaları, Uzay mekiği çalışmaları, Manevra denetimi, Uçak motoru sorunlarının erken uyarı sistemi…
- **Elektrik:** Çiplerin bozulma analizi, Lineer olmayan modellemeler…
- **Finans:** Döviz kuru tahminleri, Makro ekonomik tahminler…
- **Sağlık:** Kanser erken teşhis ve tedavisi, EEG - ECG analizleri, Kan analizi, İlaç yan etkileri analizi, Hastalıkların resimlerden tanınması, Solunum hastalıklarının teşhisi…
- **Askeri:** Askeri uçaklarda uçuş yönlerinin belirlenmesi, Silahların doğru yönlendirilmesi, Mayın arama aletleri…
- **Endüstri:** Ürünlerin tasarımı, Ürünlerin kalite kontrolü, Müşteri tahmini analizleri, Konuşmayı yazıya çevirme…

Bunlar dışında sigortacılık, petrokimya, robotik uygulamalar vb. gibi daha birçok alanda Yapay Sinir Ağları kullanılmaktadır [6].

1.6 Yapay Sinir Ağları Nasıl Çalışır?

Bir yapay sinir ağı belirli bir amaç için oluşturulur ve insanlar gibi örnekler sayesinde öğrenir. Nasıl insanın öğrenmesi sinaptik boşluklardaki (2 sinir hücresi arasındaki boşluklar) bazı elektriksel ayarlamalar sayesinde oluyorsa, aynı şekilde, yapay sinir ağları da tekrarlanan girdiler sayesinde kendi yapısını ve ağırlığını değiştirir. Yapay sinir ağları aynen canlıların sinir sistemi gibi adapte olabilen bir yapıya sahiptir. Yani içsel ve dışsal uyaranlara göre yapısı değişebilmekte ve bu sayede öğrenebilmektedir. Karar verme aşamasında bağlantı ağırlıkları da devreye girer. İşlem birimleri her ne kadar tek başlarına çalışıyor gibi gözükse de, aslında birçok yapay sinir ağı aynı anda çalışır ve dağınık, paralel hesaplama örneği gösterir. Ancak hesaplamanın duruma göre değişmesi bir başka sorunu da beraberinde getirebilir. Normal bilgisayarlar, ne yapmaları gerektiğini belirleyen kurallar olmadan çalışamaz. Bu sistem ise bir problem karşısında, çözümü kendi kendine bulduğundan, sistemin belli bir durumda ne yapacağı bilinemez. Şekil 1.1'de Yapay Sinir Ağı modeli gösterilmiştir.



Şekil 1.1: Yapay Sinir Ağları Modeli

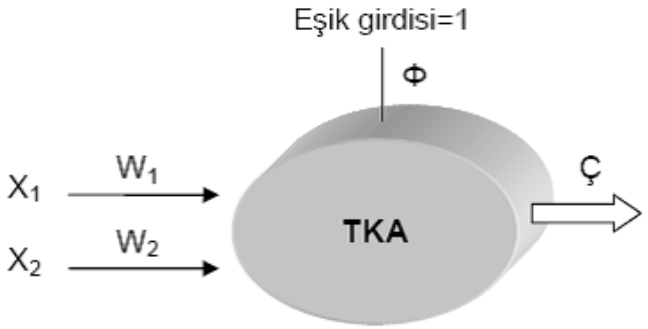
Yapay sinir hücrelerinin ağırlık değerlerinin belirlenmesi işlemine "ağın eğitilmesi" denir. Bir bilgi kaynağından öğrenebilme yeteneği YSA'nın en önemli özelliklerinden biridir. Yapay sinir ağlarında bilgi, ağdaki sinirlerin bağlantılarının ağırlıklarında tutulur. Bu nedenle ağırlıkların nasıl belirleneceği önemlidir. Bilgi tüm ağda saklandığı için bir düğümün sahip olduğu ağırlık değeri tek başına bir şey ifade etmez. Tüm ağdaki ağırlıklar optimal değerler almalıdır. Ağın doğru ağırlık değerlerinin belirlenmesi ile artık olaylar karşısında genellemeler yapılabilir, bu işleme de "ağın öğrenmesi" denir. Buna göre bir ağın eğitilebilir olabilmesi için ağırlık değerlerinin belirli bir kural dahilinde dinamik olarak değiştirilebilir olması gerekmektedir. Ağın performansını ölçmek için yapılan denemelere ise "ağın test edilmesi" denir. Çıkış değeri daima 1 olan Bias, giriş sinyallerinin toplamının sürekli sıfırdan farklı olmasını sağlar çünkü giriş sinyallerinin toplamı sıfır olduğunda öğrenme gerçekleşmez [7].

2. YAPAY SİNİR AĞLARI

Genel de Yapay Sinir Ağları tek katmanlı algılayıcı ve çok katmanlı algılayıcı olurlar.

2.1 Tek Katmanlı Algılayıcılar

Tek katmanlı yapay sinir ağları doğrusal problemlerin çözümünde kullanılıp sadece girdi ve çıktı katmanından oluşmaktadır. Çıktı üniteleri bütün girdi ünitelerine (X) bağlanmaktadır ve her bağlantının bir ağırlığı (W) vardır. Şekil 2.1'de Tek Katmanlı Algılayıcı (TKA) Modeli gösterilmiştir.



Şekil 2.1: Tek Katmanlı Algılayıcı (TKA) Modeli

Bu ağlarda süreç elemanlarının değerlerinin ve dolayısıyla ağın çıktısının sıfır olmasını önleyen bir de eşik değeri (Φ) vardır ve değeri daima 1'dir. Ağın çıktısı ağırlıklandırılmış girdi değerlerinin eşik değeri ile toplanması sonucu bulunur. Bu girdi ile bir aktivasyon fonksiyonundan geçirilerek ağın çıktısı hesaplanır. Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusal bir fonksiyondur ve 1 veya -1 değerlerini almaktadır. Eğer çıktı 1 ise birinci sınıfa -1 ise ikinci sınıfa kabul edilmektedir [8].

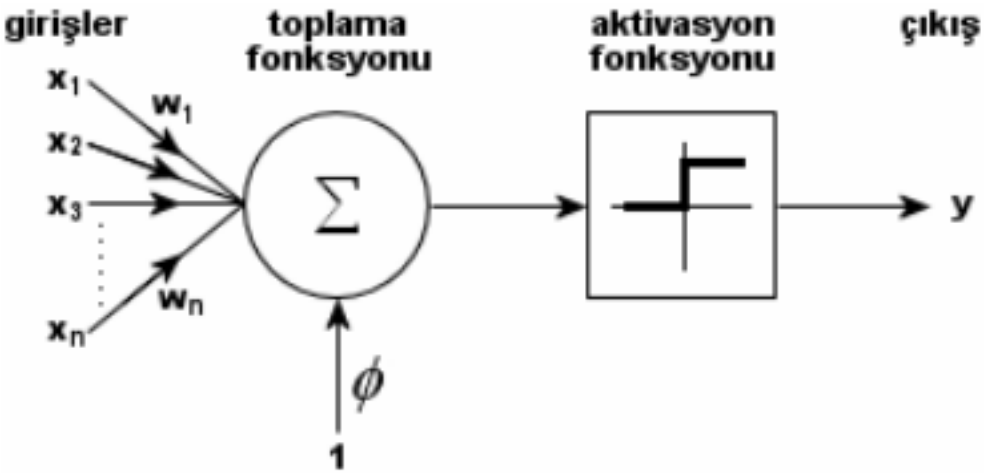
$$f(g)=\begin{cases} 1 & \text{Eğer } Ç>0 \text{ ise} \\ -1 & \text{aksi taktirde} \end{cases}$$

Tek katmanlı algılayıcılarda başlıca iki modelden söz edilebilir. Bunlar;

- Perceptron Modeli
- Adaline/Madaline Modeli

2.1.1 Basit Algılayıcı Modeli (Perceptron)

Perceptron, 1958 yılında Rosenblatt tarafından şekil sınıflandırma amacı ile geliştirilmiştir. "Zeki sistemlerin temel özelliklerinden bazılarını simüle etmek" amacıyla geliştirilen perceptron modeli, bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanır. Ağın çıktısı, girdi değerlerinin ağırlıklı toplamının bir eşik değer ile karşılaştırılması sonucu elde edilir. Toplam eşikten eşit veya büyük ise çıktı değeri 1, küçük ise 0 seçilir. Şekil 2.2'de Basit Algılayıcı Modeli gösterilmiştir

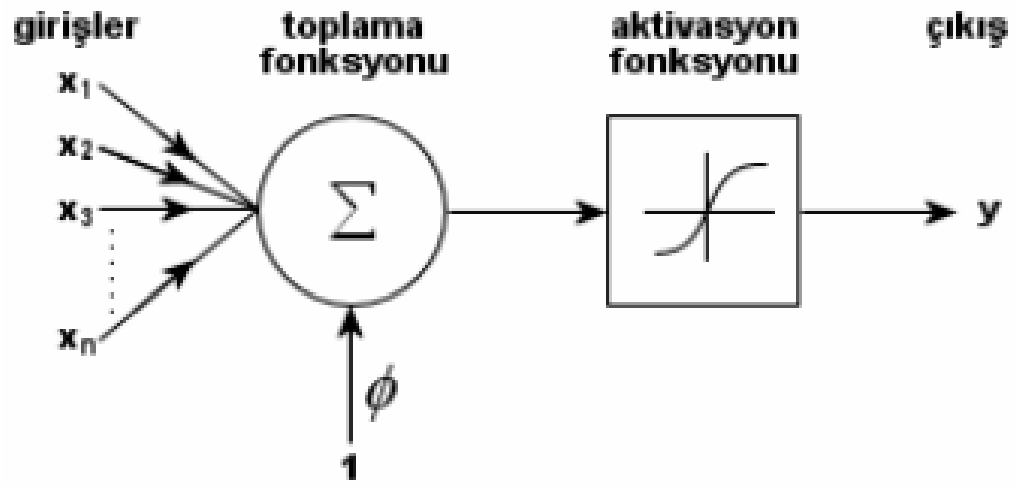


Şekil 2.2: Basit Algılayıcı Modeli

Toplama fonksyonu, ağırlıklı toplam fonksyonu kullanılır. Yüzey ayracının ötelenmesi gereken durumlarda eşik değer ilave edilebilir. Aktivasyon fonksyonu, perceptron sınıflandırma amacı ile geliştirildiğinden, farklı sınıfları temsil edecek değerler üretilmelidir. Sınıflandırma işleminde, klasik küme kuramındaki ait olma ya da ait olmama durumları incelenir. Bulanık mantıkda olduğu gibi bir aitlik derecesi vermek mümkün olmamaktadır. Bu amaçla aktivasyon fonksyonu olarak basamak fonksyon kullanılır. NET girdinin eşik değerden büyük ya da küçük olmasına göre çıkış değeri 0 ya da 1 olarak üretilir. Öğrenme kuralı (Hebb), eğer perceptronun ürettiği çıkış ile hedef değer aynı olursa, ağırlıklarda herhangi bir değişme olmaz. Bu durum sınıf ayracının doğru pozisyonda olduğunu gösterir. Fakat ağ, hedef değerden farklı bir çıkış üretmiş ise, ki bu durumda ağırlıklar değiştirilerek sınıf ayracının pozisyonu değiştirilmelidir, iki durum söz konusu olabilir. Öğrenme katsayısı [0 1] aralığında herhangi bir değer alabilir. Eşik değer benzer şekilde güncellenir [9].

2.1.2 Adaline Modeli

Widrow ve Hoff tarafından 1959 yılında geliştirilmiştir. Adaline modelinde, perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılabilir. Fakat genellikle hiperbolik tanjant fonksyonu ya da sigmoidal fonksyon 34 kullanılır. Bunun dışında öğrenme kuralı, perceptronda kullanılan Hebb kuralından farklıdır. Şekil 2.3'de Adaline Modeli gösterilmiştir.



Şekil 2.3: Adaline Modeli

Giriş ve hedef vektörleri, perceptronlarda olduğu gibi giriş değerlerinin sayısı için bir kısıtlama yoktur. Giriş vektörü, negatif ya da pozitif herhangi bir değeri içerebilir. Giriş değerlerinin 1'den büyük ya da küçük olması performansı etkilemez. Hedef vektörüne getirilecek kısıtlamalar ise, aktivasyon fonksyonları ile ilişkilidir. Eğer perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılırsa, hedef değerler sadece 0 veya 1 olabilir. Fakat hiperbolik tanjant ya da sigmoidal aktivasyon fonksyonları kullanılırsa, bu durumda hedef vektör, 0 ve 1 aralığındaki değerleri de içerebilir. Toplama fonksyonu, Ağırlıklı toplam fonksyonu kullanılır. Aktivasyon fonksiyonu, adaline modelinde genellikle hiperbolik tanjant fonks,yonu ya da sigmoidal fonksiyon kullanılır. Sigmoidal fonksiyonda β değerini değiştirerek, farklı eğimlerde fonksiyonlar elde etmek mümkündür. Madaline ağları ise birden fazla adaline ünitesinin bir araya gelerek oluşturdukları ağa verilen isimdir.Ağın çıktısı yine ya 1 ve -1 değelerleri ile gösterilmektedir [10].

2.2 Çok Katmanlı Algılayıcılar

Yapay sinir ağlarında girdi ve çıktılar arasındaki ilişkiler doğrusal değilse bu problem tek katmanlı algılayıcı modeller ile öğrenme çözümü gerçekleştiremez. Bu problemde öğrenme gerçekleştirebilmek için daha gelişmiş modele ihtiyaç vardır. Bu öğrenmeyi de çok katmanlı algılayıcı model ile yapmak mümkündür. Çok katmanlı algılayıcı model ile girdi ve çıktılar arasındaki doğrusal olmayan sorun çözülebilir. Buna en iyi örnek ise XOR problemidir. Bu problem doğrusal olmayan özellikte bir ilişkiyi gösterir. Yani çıktılar arasına bir doğru veya doğrular çizerek onları iki veya daha fazla sınıfa ayırmak mümkün değildir. XOR ve NXOR probleminin girdi ve çıktıları Tablo 2.1 ve Tablo 2.2 'de gösterilmiştir.

Tablo 2.1: XOR Problemi Girdi ve Çıktıları

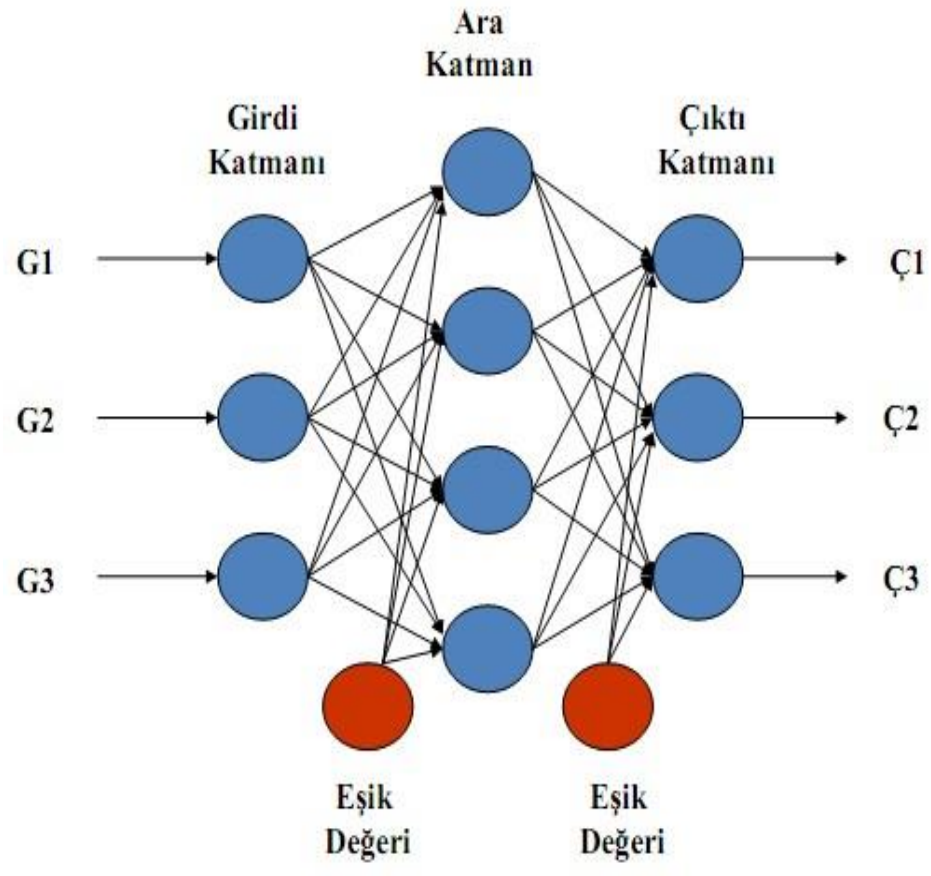
	Girdi 1	Girdi 2	Çıktı
Örnek 1	0	0	0
Örnek 2	0	1	1
Örnek 3	1	0	1
Örnek 4	1	1	0

Günlük hayattaki olayların çoğu doğrusal olmayan nitelikler taşımaktadır. XOR problemi sayesinde de tek katmanlı algılayıcı modellerin bu probleme çözüm üretemediği gösterilmiş olup çok katmanlı algılayıcı modeller geliştirilmiş. Yani XOR problemi yapay sinir ağ araştırmalarında bir kilometre taşı olarak görülmektedir. Çok katmanlı algılayıcı model sayesinde sınıflandırma, tanıma ve genelleme yapma gerektiren problemler için önemli bir araç olmuştur. Çok katmanlı algılayıcılarda Delta Öğrenme Kuralı (geriye yayılım algoritması) kullanılmıştır. Bu kural adaline ve basit algılayıcı modellerindeki öğrenme kuralının geliştirilmiş halidir. Bu öğrenme kuralındaki amaç ağın çıktısı ve beklenen çıktı arasındaki hata oranı en aza indirmek olmuştur [11].

Tablo 2.2: NXOR Problemi Girdi ve Çıktıları

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

Çok Katmanlı Algılayıcı (ÇKA), eğitim aşamasında girdilerin ve bu girdilere karşılık üretilmesi beklenen çıktıların gösterildiği modeldir. Oluşturulan ağ verilen girdiye göre çıktıyı bulmaktadır. Ağ yapısı ise Şekil 2.4'de verildiği üzere 3 farklı katmandan oluşur. Gelen bilgilerin ara katmana iletildiği katman girdi katmanıdır. Ara katman ise bir veya birden çok sayıda olabilir. Burada girdi katmanında alınan bilgiler işlenir. Çıktı katmanında ise ara katmandan alınan bilgilere karşılık her bir girdi için çıktı değerleri saptanır. Tüm katmanlar arasında proses elemanları birbiriyle bağlantılıdır.



Şekil 2.4: Çok Katmanlı Algılayıcı Ağının Topolojik Yapısı

ÇKA'da katmanlar arası ileri ve geri yayılım olarak adlandırılan geçişler bulunur. İleri doğru hesaplama, ağın çıktısı ve hata değeri hesaplanır. Geri doğru hesaplamada ise hesaplanan hata değerinin minimize edilmesi için katmanlar arası bağlantı ağırlık değerleri güncellenir. ÇKA modeli doğrusal perceptrondaki en küçük kareler algoritmasının genelleştirilmesi olan geri yayılım (backpropagation) öğrenme algoritmasını kullanır. Çok katmanlı algılayıcı yapay sinir ağı modelinin çalışma adımları aşağıda açıklanmıştır.

Örnek uzayının toplanması: Bu adımda ağın çözmesi istenen problem için daha önce gerçekleşmiş örnek uzayı bulunur. Ağın eğitilmesi için örnek uzayı toplandığı gibi ağın test edilmesi içinde örnek uzayının toplanması gerekmektedir. Ağın eğitimi esnasında test örnek uzayı ağa hiç gösterilmez. Eğitim safhasında eğitim örnek uzayındaki örnekler ağa tek tek gösterilerek ağın problemi öğrenmesi sağlanır. Eğitim safhasının sonunda test uzayındaki örnekler ağa gösterilerek ağın performansı ölçülür. Daha önce hiç gösterilmeyen örnekler karşısındaki ağın başarısı eğitim safhasında ağın ne kadar iyi öğrenip öğrenmediğini ortaya koymaktadır.

Ağın topolojisinin belirlenmesi: Bu adımda ağda kaç tane girdi ünitesi, kaç tane ara katman, ara katmanlarda kaçar tane nöron ve çıktı katmanında kaç tane çıktı elemanı olması gerektiği belirlenmektedir.

Öğrenme safhasındaki değişkenlerin belirlenmesi: Bu adımda ağın öğrenme katsayısı, nöronların toplama ve transfer fonksiyonları, momentum katsayısı vb. değişkenler belirlenmektedir.

Ağırlık katsayılarının başlangıç değerlerinin tayin edilmesi: Bu adımda nöronları birbirine bağlayan ağırlık katsayılarının ve eşik değer ünitesinin ağırlık katsayılarının başlangıç değerlerinin tayini yapılmaktadır. Genellikle ilk değerler rastgele atanır.

Örnek uzayından örneklerin seçilmesi ve ağa gösterilmesi: Bu adımda ağın eğitime başlaması ve aşağıda anlatılan delta öğrenme kuralına göre ağırlık katsayılarını değiştirmesi için ağa girdi-çıktı değerleri belirli bir düzende gösterilir.

Öğrenme safhasındaki ileri hesaplamaların yapılması: Bu adımda bir önceki adımda anlatıldığı şekilde ağa sunulan girdi değerleri için ağın çıktıları hesaplanır.

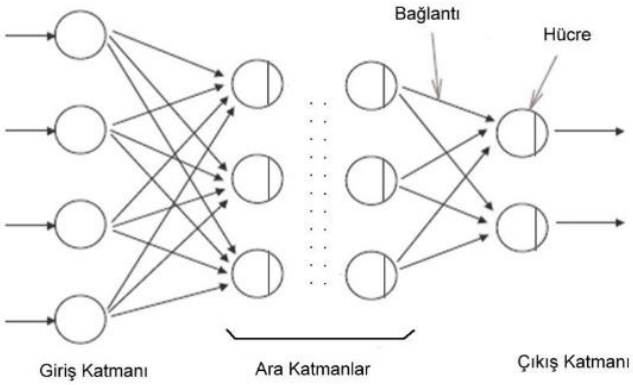
Gerçekleşen çıktı ile beklenen çıktının karşılaştırılması: Bu adımda ağın ürettiği hata değerleri hesaplanır

Ağırlık katsayılarının değiştirilmesi: Bu adımda delta öğrenme kuralındaki geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

Bu adımlar beklenen çıktı değerleri ile gerçekleşen çıktı değerleri arasındaki fark kabul edilebilir düzeye ininceye kadar devam ettirilir. Beklenen çıktı değerleri ile gerçekleşen çıktı değerleri arasındaki fark bir eşik değerinin altına düştüğünde çok katmanlı algılayıcı yapay sinir ağı modelinin öğrenmesi tamamlanmış demektir.Çok Katmanlı algılayıcı yapay sinir ağı modeli öğrenme Kuralı Delta Öğrenme Kuralı'dır. İki Aşamadan oluşmuştur.

2.2.1 İleri Doğru Hesaplama

İleri doğru hesaplama sinir ağları tek yönlü sinyal akışı için izin verir. Ayrıca, ileri doğru hesaplamada Şekil 2.5'de verildiği gibi sinir ağları çoğu katmanlar halinde organize edilmektedir.



Şekil 2.5: İleri Doğru Hesaplama Yapay Sinir Ağı Modeli

İleri doğru hesaplama yapay sinir ağında, 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 ara (gizli) katmandaki hücrelere iletir. Bilgi, ara ve çıkış katmanında işlenerek ağ çıkışı belirlenir. En çok bilinen algoritma olan geriye yayılım öğrenme algoritması, bu tip yapay sinir ağların eğitiminde etkin olarak kullanılmaktadır. Ağa, hem örnekler hem de örneklerden elde edilmesi gereken çıktılar (beklenen çıktılar) 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 [12].

İleri Doğru Hesaplama Aşaması: Bu aşama örnek uzayındaki bir örneğin girdi katmanından ağa gösterilmesi ile başlar. Girdi katmanında herhangi bir bilgi işleme olmadığından gelen girdiler hiçbir değişikliğe uğramadan ara katmana gönderilir. Girdi katmanındaki m. nöronun çıktısı şu şekilde olmaktadır.

$$Ç_m^i = G_m$$

Ara katmandaki her nöron girdi katmanındaki bütün nöronlardan gelen bilgileri bağlantı ağırlıklarının (A1, A2……) etkisi ile alır. Önce ara katmandaki nöronlara gelen net girdi (NET^a) hesaplanır.

$$NET_j^a = \sum_{m=1}^{n} A_{mj} Ç_m^i$$

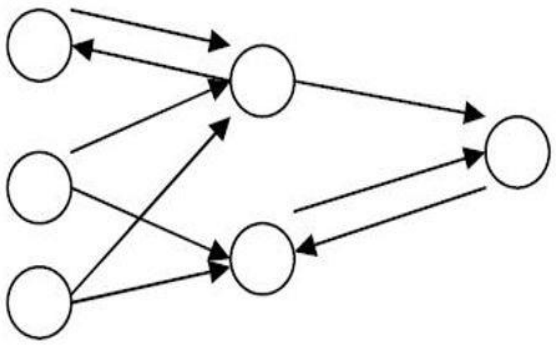
Bu formülde A_{mj}m. girdi katmanı elemanını j. Ara katman elemanına bağlayan bağlantının ağırlık değerini göstermektedir. j. Ara katman elemanının çıktısı ise bu net girdinin transfer fonksiyonundan geçirilmesi ile hesaplanır. Kullanılan transfer fonksiyonu türevi alınabilir bir fonksiyon olmalıdır. Eğer transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa çıktı,

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

Olacaktır. Burada geçen βj ara katmanda bulunan j. Elemana bağlanan eşik değer elemanının ağırlığını göstermektedir. Ara katmanın bütün nöronları ve çıktı katmanının nöronlarının çıktıları aynı şekilde kendilerine gelen net girdinin hesaplanması ve transfer fonksiyonundan geçirilmesi sonucu belirlenir. Çıktı katmanındaki nöronların çıktıları bulununca ileri doğru hesaplama aşaması tamamlanmış olur [13].

2.2.2 Geriye Doğru Hesaplama

Geriye doğru hesaplama Yapay Sinir Ağları'nda, en az bir hücrenin çıkışı kendisine ya da diğer hücrelere giriş olarak verilir ve genellikle geri doğru hesaplama, bir geciktirme elemanı üzerinden yapılır. Geriye doğru hesaplama, bir katmandaki hücreler arasında olduğu gibi katmanlar arasındaki hücreler arasında da olabilir. Bu yapısı ile geri doğru hesaplama, doğrusal olmayan dinamik bir davranış gösterir. Dolayısıyla, geri doğru hesaplama yapılış şekline göre farklı yapıda ve davranışta geri beslemeli YSA yapıları elde edilebilir. Aşağıda bulunan Şekil 2.6'da iki katmanlı ve çıkışlarından giriş katmanına geri beslemeli bir YSA yapısı görülmektedir [14].



Şekil 2.6: Geriye Doğru Hesaplama Yapay Sinir Ağı Modeli

Geriye Doğru Hesaplama Aşaması: Bu aşamada ağa sunulan girdiler için ağ tarafından üretilen çıktı değerleri beklenen çıktı değerleri (B1, B2 ….) ile karşılaştırılır. Aradaki fark hata olarak kabul edilir. Hedef bu hata değerinin azaltılmasıdır. Dolayısıyla bu hata değeri ağın ağırlık değerlerine dağıtılarak bir sonraki iterasyonda hatanın azaltılması temin edilir. Çıktı katmanındaki k. Nöron için oluşan hata (Ek);

$$E_k = B_k - Ç_k$$

Olarak bulunur. Bu bir nöron için oluşan hatadır. Çıktı katmanı için oluşan toplam hata (TH) şöyle hesaplanır;

$$TH = \frac{1}{2}\sum_k E_k^2$$

Toplam hatayı en aza indirmek için bu hatanın kendisine neden olan nöronlara dağıtılması gerekmektedir. Bu ise nöronların ağırlıklarını değiştirmek demektir. Ağırlık değiştirme işlemi çıktı katmanı – ara katman arasında olmak üzere ve ara katmanlar veya ara katman – girdi katmanı arasında olmak üzere ayrı ayrı anlatılacaktır.

Çıktı katmanı – ara katman arasındaki ağırlık değiştirilmesi işleminde çıktı katmanındaki m. nörona ara katmandaki j. Nöronu bağlayan bağlantının ağırlığındaki değişim miktarına ΔA^a denildiğinde t. iterasyonda ağırlığın değişim miktarı aşağıdaki formül ile hesaplanır.

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

Bu formülde α momentum katsayısını, λ öğrenme katsayısını gösterir. Momentum katsayısı çok katmanlı algılayıcı yapay sinir ağı modelinin öğrenmesi esnasında yerel bir optimum noktaya takılıp kalmaması için ağırlık değişim değerinin belirli bir oranda bir sonraki değişime eklenmesini sağlarken öğrenme katsayısı ağırlıkların değişim miktarını sağlar. Formüldeki δ_m m. çıktı ünitesinin hatasını göstermektedir. Aşağıdaki şekilde hesaplanır.

$$\delta_m = f'(\text{NET}).E_m$$

Bu formüldeki f'(NET) transfer fonksiyonunun türevidir. Transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa

$$\delta_m = Ç_m(1-Ç_m).E_m$$

Olur. Ağırlıkların t. iterasyondaki yeni değerleri aşağıdaki gibi olacaktır.

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

Aynı şekilde eşik değer ünitesinin de ağırlıklarının değiştirilmesi gerekmektedir. Bu amaçla önce değişim miktarı hesaplanır. Eğer çıktı katmanındaki nöronların eşik değer ağırlıkları $\beta^{ç}$ olursa değişim miktarı

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

Olur. t. iterasyondaki eşik değerin yeni ağırlık değeri

$$\beta_m^{ç}(t) = \beta_m^{ç}(t\text{-}1) + \Delta\beta_m^{ç}(t)$$

Olacaktır. Ara katmanlar veya ara katman – girdi katmanı arasındaki ağırlık değiştirilmesi işleminde ağırlıkların değişim miktarı ΔA^{i} ile gösterildiğinde

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

Olur. Hata değeri δ^{a} ise aşağıdaki gibi hesaplanır.

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

Eğer transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa hata değeri aşağıdaki gibi olacaktır.

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

Hata değeri bulunduktan sonra ağırlıkların yeni değeri

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

Olarak hesaplanır. Aynı şekilde eşik değer ünitesinin yeni ağırlıkları da yukarıda anlatıldığı şekilde hesaplanır. Ara katmanlar için eşik değer ağırlıkları β^{a} ile gösterildiğinde değişim miktarı aşağıdaki gibi hesaplanır.

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

Olur. t. iterasyonda ağırlıkların yeni değerleri aşağıdaki gibi olacaktır.

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

Sonuç olarak ağırlıkların yeni değerleri bir iterasyon için hem ileri hem de geriye hesaplamaları yapılmış olarak tamamlanmış olacaktır. Bundan sonra ikinci bir örnek ağa verilerek sonraki iterasyona başlanır ve ağırlıklar yeniden hesaplanır. Bu işlemler ağın öğrenmesi tamamlanana kadar tekrarlanır [13].

3. GERİYE YAYILIM ALGORİTMASI

Yapay Sinir Ağları'nda genelde "geri yayılım algoritması" kullanılır. Geri yayılım algoritmasının adımlarını şöyle sıralayabiliriz;

1) İlk olarak başlangıç ağırlıkları 0'a yakın rasgele küçük değerler seçilir.

2) Aktifleşmenin hesaplanması.

2.1) Ağa ilk giriş değerleri (kombinasyonu) verilir.

2.2) Gizli katmandaki her bir nöron için aktifleşme aşağıdaki ifadelere göre hesaplanır:

$$net_j = \sum_{i=1}^{n} x_i w_{ij} \qquad F_j = \frac{1}{1+exp(-net_j)}$$

2.3) Çıkış katmanındaki nöronlar için aktifleşme aynen gizli katmandaki gibi hesaplanır. Adım 2.2'de hesaplanmış olan çıkış değerleri, bu ifadelerde çıkış nöronunun girişleri olacaktır.

3) Hatanın hesaplanması.

3.1) Her bir çıkış nöronu için geriye yayılma hatasının hesaplanması:

$$d_j = F_j\,(1\text{-}F_j)\,(T_j - F_j)$$

Burada F_j gizli katmandaki j. nöronun 2.2 adımında hesaplanmış olan çıkış değeridir.

3.2) Bu adımda önce verilmiş bütün çıkış hedefleri ile elde edilmiş çıkış değerleri kıyaslanır. Eğer bu değerler farklı ise hesaplama devam eder (3.3'e git). Farklı değilse hesaplama bitmiştir. Hesaplamanın bitmesi için karar şu formülle verilir:

$$E = (1/2)\sum_j (Tj\text{-}Fj)^2$$

Bütün giriş kombinasyonları için uygun çıkışların E değeri "0" veya bu değer önceden verilmiş bir hata değerinden eşit veya küçük olduğunda eğitim bitirilir.

3.3) Gizli katmanlardan en son katmandaki nöronlar için hatanın hesaplanması:

$$\delta_j = F_j(1-F_j)\sum_k \delta_k w_{kj}$$

Burada F_j bir önceki gizli katmandaki (eğer varsa) nöronların çıkış değerleridir. Eğer gizli katman bir tane ise, o zaman F_j yerine x_j yani uygun giriş değerleri kullanılır. Böylece 3.3 adımı bütün gizli katmanlar için tekrar edilir.

4) Yeni ağırlıkların hesaplanması.

4.1) Bütün katmanlar için ağırlık değerleri yeniden hesaplanır. Formülleri şu şekildedir;

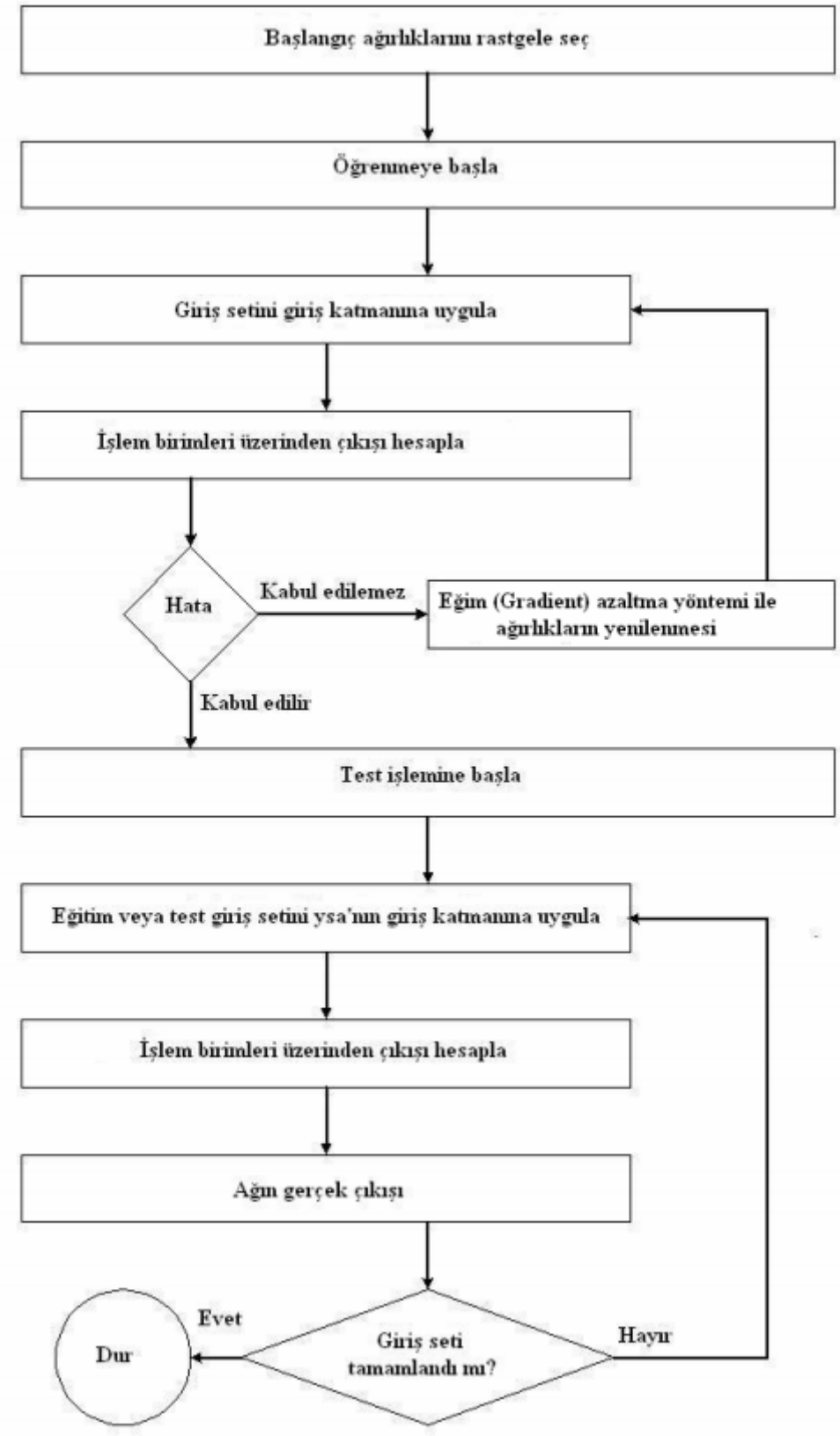
$$Dw_{ji} = a \,.\, d_i \,.\, x_j$$

$$w_{ji}\,(t+1) = w_{ji}\,(t) + Dw_{ji}$$

Burada a - eğitim hızı ($0 < a < 1$, örneğin a=0,3); d_i - i. birimin hatasıdır.

4.2) 2. adıma geri dön.

5) İşlemin sonlandırılması [15].



Şekil 3.1: Geriye Yayılım Algoritması Blok Şeması [16].

4. PROJENİN YAPIMI

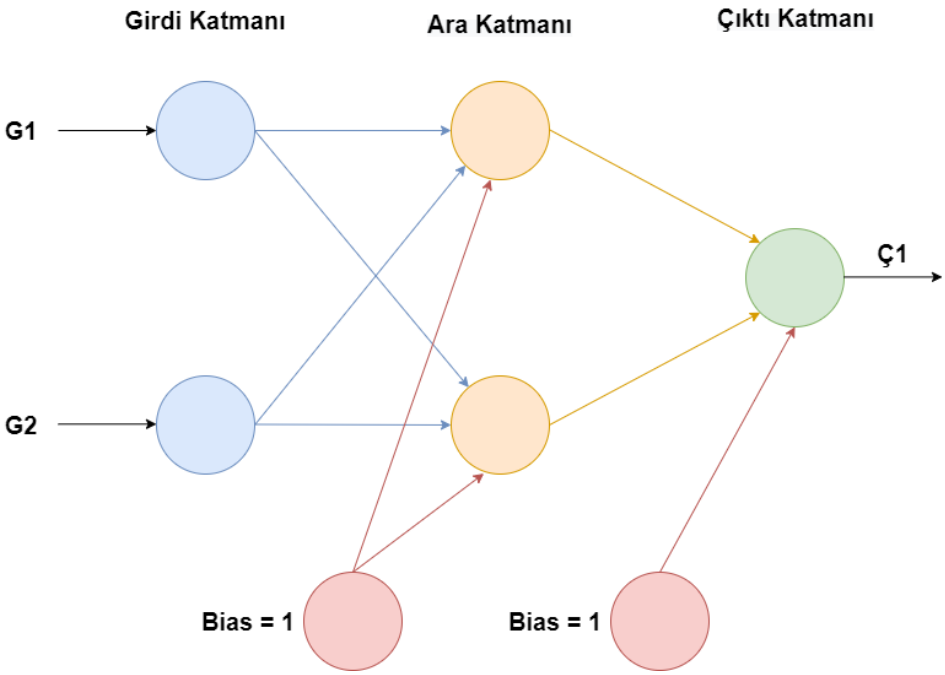
4.1 Projenin Tanınması ve Yapay Sinir Ağı Modelinin Oluşturulması

Yapay Sinir Ağı modellerinden biri olan Çok Katmanlı Algılayıcı Ağı modeli seçilmiştir. Bu model XOR probleminin çözümünde kullanılır. Projemizde NXOR problemini XOR problemindeki yöntemlerle çözmeye çalışacağız. NXOR probleminde 4 giriş parametresi, 1 tane ise çıktı değeri olacağı saptanmıştır. Yani 4 örnek seti de ayrı ayrı ardışık olarak ağa gösterilecek beklenen çıktılarla ağın çıktısı belirlenecek. Hata oranı hesaplanacak. Ağın giriş parametreleri ve beklenen çıktılar Tablo 4.1'de gösterilmiştir [17].

Tablo 4.1: NXOR Problemi Girdi ve Çıktıları

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

Buna göre tasarlanacak olan YSA modelinin temsili diyagramı Şekil 4.1.'de gösterilmiştir.



Şekil 4.1.: YSA Modelinin Temsili Diyagramı

4.2 NXOR Probleminin Çözülmesi

4.2.1 NXOR Probleminin Çözülmesi 1.Adım

Tablo 4.1'de gösterildiği gibi NXOR problemi için 4 örnek veri seti vardır. Bunlar 1 ve 0 değerlerinden oluşmaktadır. Her örnek için girdiler ve beklenen çıktı gösterilmiştir.

4.2.2 NXOR Probleminin Çözülmesi 2.Adım

Şekil 4.1'de gösterildiği gibi ağın topolojik yapısı belirlenmiştir. Çok Katmanlı Algılayıcı ağının 2 girdi ünitesi, 1 çıktı ünitesi 1 adet de eşik değer ünitesi olacaktır.1 ara katman ve 2 tanede ara katman proses elamanı bu problemi çözecektir. Bunlar için yazılan Python kodu da Şekil 4.2'de gösterilmiştir.

```
### 2.ADIM - 2 Girdi 1 Çıktı, 1 Ara katmadan ve 2 tane ara katman processi
# Input datasets
inputs = np.array([[int(self.girdi1.text()), int(self.girdi2.text())]])

if (int(self.girdi1.text()) == 0 and int(self.girdi2.text()) == 0):
    expected_output = np.array([[1]])
elif (int(self.girdi1.text()) == 1 and int(self.girdi2.text()) == 0):
    expected_output = np.array([[0]])
elif (int(self.girdi1.text()) == 0 and int(self.girdi2.text()) == 1):
    expected_output = np.array([[0]])
elif (int(self.girdi1.text()) == 1 and int(self.girdi2.text()) == 1):
    expected_output = np.array([[1]])
else:
    expected_output = np.array([[0]])

inputLayerNeurons = 2
hiddenLayerNeurons = 2
outputLayerNeurons = 1
```

Şekil 4.2: Python Kodu

4.2.3 NXOR Probleminin Çözülmesi 3.Adım

Oluşturulan ağ için aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmıştır. Öğrenme katsayısı 0.25 ve Momentum katsayısı 0.6 olarak belirlenmiştir.

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)
```

Şekil 4.3: Sigmoid Fonksiyonu Python Kodu

```
### 3.ADIM - Öğrenme Parametrelerinin Belirlenmesi
lr = 0.6
momentum = -0.25
print("Öğrenme Kat Sayısı", lr)
print("Momentum", momentum)
```

Şekil 4.4: Öğrenme ve Momentum Katsayısı Python Kodu

4.2.4 NXOR Probleminin Çözülmesi 4.Adım

Oluşturulan ağ için ağırlık vektörleri ve başlangıç değerlerinin belirlenmesi için yazılan kod Şekil 4.5'de gösterilmiştir.

```
### 4.ADIM - Ağırlık Başlangıç Değerlerinin Atanması
# Random weights and Bias=1

# Girdi Katmanı ile Ara Katman Arasındaki Rastgele Atanmış Ağırlıklar ve Bias Değeri
hidden_weights = np.random.uniform(size=(inputLayerNeurons, hiddenLayerNeurons))
hidden_bias = np.random.uniform(low=1, high=1, size=(1, hiddenLayerNeurons))

print("Initial hidden weights: ", end='')
print(*hidden_weights)

# Çıktı Katman ile Ara Katman arasındaki Rastgele Atanmış Çıktı Ağırlıklar ve Bias Değeri
output_weights = np.random.uniform(size=(hiddenLayerNeurons, outputLayerNeurons))
output_bias = np.random.uniform(low=1, high=1, size=(1, outputLayerNeurons))


print("Initial output weights: ", end='')
print(*output_weights)
```

Şekil 4.5: Ağırlık Başlangıç Değerlerinin Atandığı Python Kodu

4.2.5 NXOR Probleminin Çözülmesi 5.Adım

Birinci örnek G1 = 0, G2 = 0 ve B=1 olarak belirlenmiş ağa gösterilmiş ve ileri doğru hesaplama başlamıştır. Ara katman ünitesinin NET girdileri ve ara katman ünitelerinin çıktıları hesaplanmıştır. Daha sonra çıktı katmanındaki proses elemanının NET girdisi ve ağın çıktısı hesaplanmıştır. Bunun için yazılan kod Şekil 4.6'de gösterilmiştir.

```
## İLERİYE DOĞRU HESAPLAMA

# Ara Katman Ünitelerinin NET Girdilerinin Hesaplanması
hidden_layer_activation = np.dot(inputs, hidden_weights)
hidden_layer_activation += hidden_bias

# Ara Katman Ünitelerinin Çıktı Değerlerinin Hesaplanması
hidden_layer_output = sigmoid(hidden_layer_activation)

#Çıktı Katmanındaki Proses Elemanlarının NET Girdisi Hesaplanması
output_layer_activation = np.dot(hidden_layer_output, output_weights)
output_layer_activation += output_bias

# Çıktı Katmanı Ünitesinin Çıktı Değeri Hesaplanması
predicted_output = sigmoid(output_layer_activation)
```

Şekil 4.6: İleri Doğru Hesaplama Python Kodu

Beklenen çıktıya göre ağın hatasının hesaplandığı, bu hatanın geriye doğru yayılması sonucu ara katman ile çıktı katmanı ve girdi katmanı ile ara katman arasındaki ağırlıkların değişim miktarlarının hesaplandığı geriye doğru hesaplama evresi kod Şekil 4.7'de gösterilmiştir.

```
## GERİYE DOĞRU HESAPLAMA

# Beklenen Çıktıya Göre Ağın Hatasının Hesaplanması
error = expected_output - predicted_output
# Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Değişim Miktarının Hesaplanması
d_predicted_output = error * sigmoid_derivative(predicted_output)

# Ara Katman Hatasının Hesaplanması
error_hidden_layer = d_predicted_output.dot(output_weights.T)
# Girdi Katmanı ile Ara Katmanı Arasındaki Ağırlıkların Değişim Miktarının Hesaplanması
d_hidden_layer = error_hidden_layer * sigmoid_derivative(hidden_layer_output)
```

Şekil 4.7: Geriye Doğru Hesaplama Python Kodu

Değişim miktarları kullanılarak girdi, ara ve çıktı katmanları arasındaki ağırlıkların yeniden hesaplandığı ve bu ağırlıklara göre ağın çıktısının üretildiği geriye doğru hesaplama evresi kod Şekil 4.8'de gösterilmiştir.

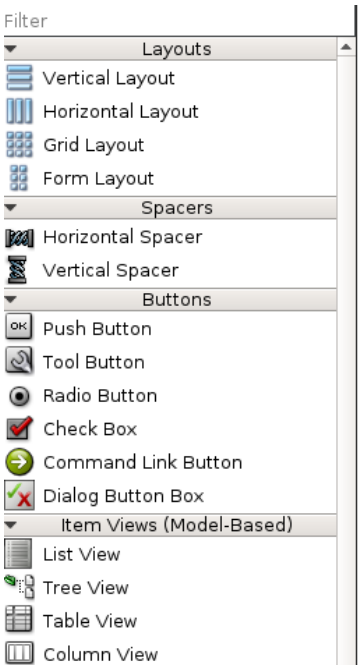
```
# Bu Değişim Miktarları Kullanılarak Girdi Katmanı, Ara Katman ve Çıktı Katmanı Arasındaki Ağırlıkların Yeniden Hesaplanması
output_weights += hidden_layer_output.T.dot(d_predicted_output) * lr
output_bias += np.sum(d_predicted_output, axis=0, keepdims=True) * lr

hidden_weights += inputs.T.dot(d_hidden_layer) * lr
hidden_bias += np.sum(d_hidden_layer, axis=0, keepdims=True) * lr

print("Final output weights: ", end='')
print(*output_weights)

print("Final hidden weights: ", end='')
print(*hidden_weights)

print("\nAğın ürettiği çıktı:  ", end='')
print(*predicted_output)
```

Şekil 4.8: Geriye Doğru Hesaplama Python Kodu

Birinci örnek bittikten sonra en son ağırlıklar tekrar kullanılarak ikinci örnek de ağa gösterilir. Bu adımlar kalan örnekler içinde yapılır. Her bir örnek için ağın çıktısı ve hata oranları hesaplanır.

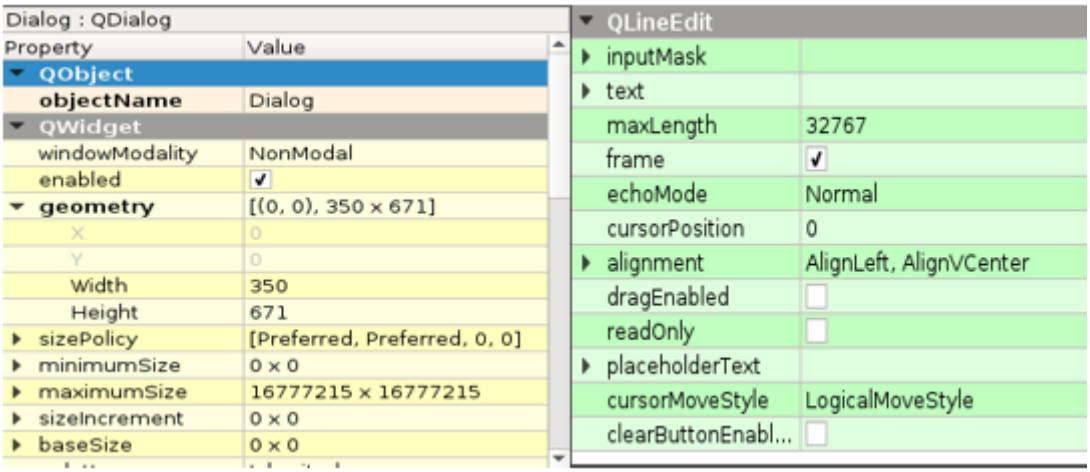
4.3 NXOR Problemi Uygulaması İçin Arayüz Tasarlanması

NXOR problemini oluştururken terminal ekranını kullandım. Bu terminal ekranına, kullanıcıların kullanımını kolaylaştıracak bir arayüz tasarlamam gerekiyordu. Python kütüphanesi olan PYQT5 kütüphanesini Şekil 4.9'daki tasarım özellikleri ile kullandım.



Şekil 4.9: PYQT5 Tasarım

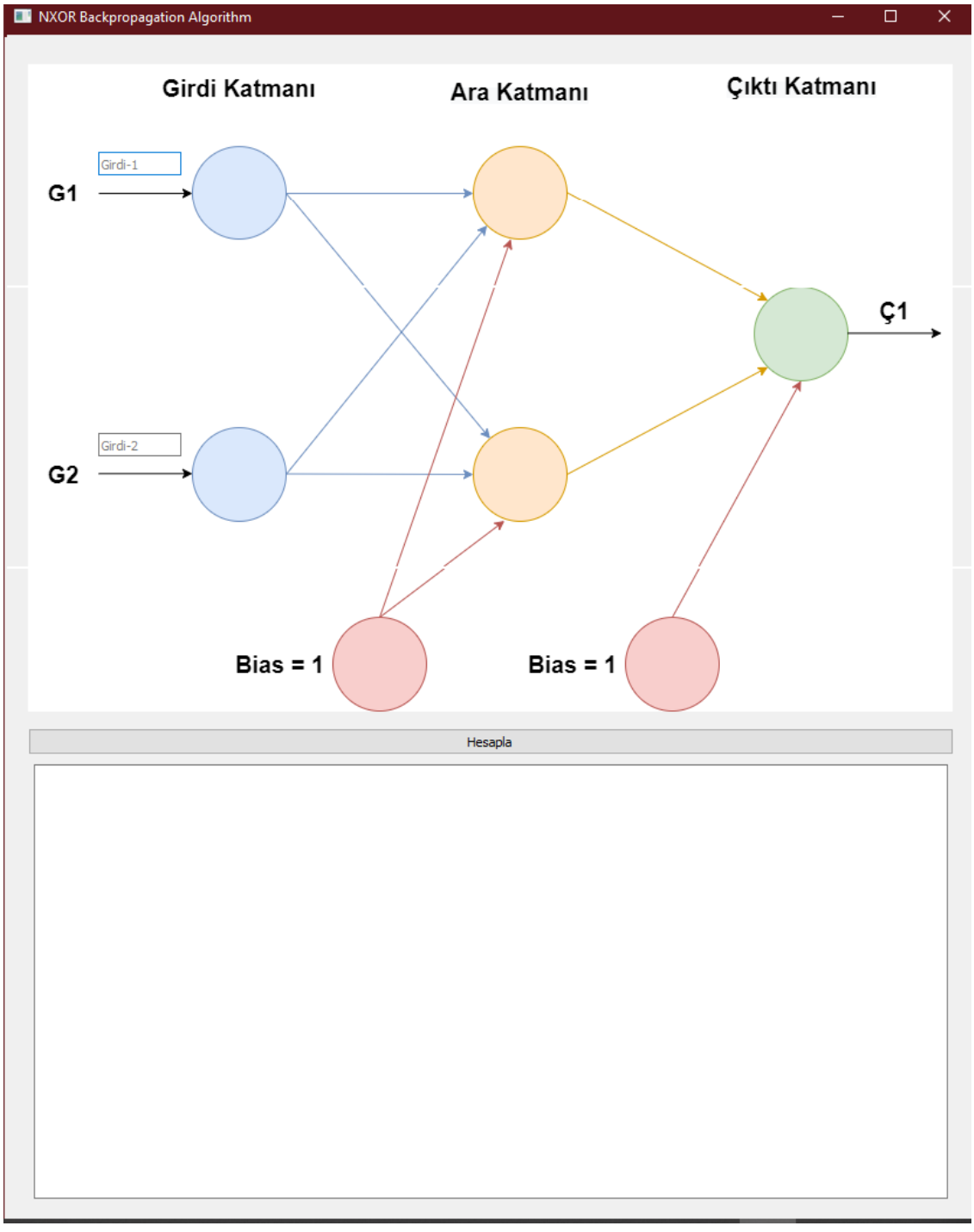
PYQT5 kütüphanesin de Şekil 4.10'daki gibi layouts, spacers, buttons, item views, item widgets, containers, input widgets gibi özellikleri vardır.



Şekil 4.10 PYQT5 Özellikleri

Her nesnenin özellikleri de ayarlanabiliyor. Object, enabled, font, size, text, eachmode, icon gibi.

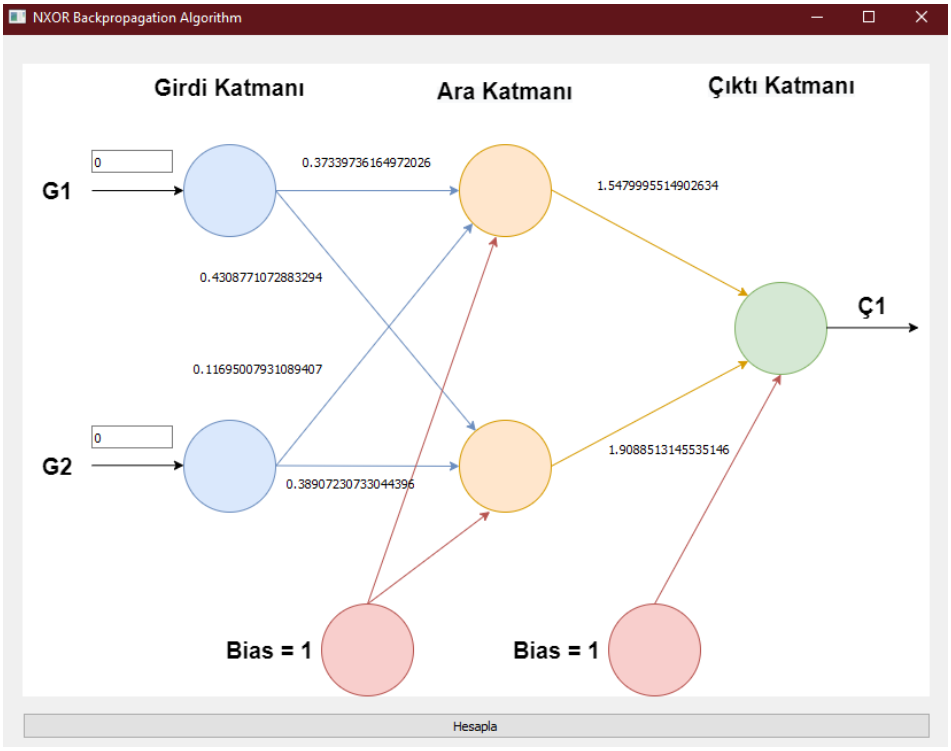
Arayüz de kullanıcı girdileri girebilecek ve hesapla butonuna basınca her işlem adımı aşağıdaki beyaz alan olan TextBrowser da gösterilecek. Arayüzdeki çok katmanlı algılayıcı modelinin üstünde ise katmanlar arasındaki ağırlıkların yeniden hesaplanmış halleri oklar üzerinde yazılmış olacak. Bu arayüz tasarımı Şekil 4.11'de gösterilmiştir.



Şekil 4.11 Arayüz Tasarımı

4.4 NXOR Problemi Sonuçları

Birinci örnek G1 = 0, G2 = 0 ve B=1 olarak belirlenmişti ve ağa gösterildikten sonraki adımlar ve sonuçlar aşağıdaki Şekil 4.12 ve Şekil 4.13'de gösterilmiştir.



Şekil 4.12 Birinci Örnek Girdi Sonuçları

Öğrenme Kat Sayısı : 0.6

Momentum : -0.25

Girdi Katmanı ile Ara Katman Arasındaki rastgele Atanmış Ağırlıklar [[0.37339736 0.43087711]
[0.11695008 0.38907231]]

Çıktı Katman ile Ara Katman arasındaki Rastgele Atanmış Çıktı Ağırlıklar[[1.54799955]
[1.90885131]]

İLERİYE DOĞRU HESAPLAMA

Ara Katman Ünitelerinin NET Girdilerinin Hesaplanması[[1.26071487 1.34323358]]

Ara Katman Ünitelerinin Çıktı Değerlerinin Hesaplanması[[0.77914914 0.7930212]]

Çıktı Katmanındaki Proses Elemanlarının NET Girdisi Hesaplanması[[5.10748345]]

Çıktı Katmanı Ünitesinin Çıktı Değeri Hesaplanması[[0.99398511]]

GERİYE DOĞRU HESAPLAMA

Beklenen Çıktıya Göre Ağın Hatasının Hesaplanması[[0.00601489]]

Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Değişim Miktarının Hesaplanması[[3.59613408e-05]]

Girdi Katmanı ile Ara Katmanı Arasındaki Ağırlıkların Değişim Miktarının Hesaplanması[[9.57903312e-06 1.12671674e-05]]

Bu Değişim Miktarları Kullanılarak Ara Katman ile Çıktı Katmanı Arasındaki Ağırlıkların Yeniden Hesaplanması[[1.54799955]
[1.90885131]]

Bu Değişim Miktarları Kullanılarak Girdi Katman ile Ara Katmanı Arasındaki Ağırlıkların Yeniden Hesaplanması[[0.37339736 0.43087711]
[0.11695008 0.38907231]]

Ağın ürettiği çıktı: [0.99398511]

Şekil 4.13 Birinci Örnek Girdi Sonuçları

Tüm ağırlıklar ile girdiler ağa tekrar gösterildiğinde Tablo 4.2'deki tabloda gösterilen sonuçlar elde ediliyor. Bu sonuçlar da ağın çıktısı ve ne kadar hata ile çözdüğünü gösteriyor.

Tablo 4.2 NXOR Problemi İçin Ağın Çıktıları ve Hata Oranları

Girdi 1	Girdi 2	Beklenen Çıktı	Ağın Çıktısı	Hata
0	0	0	0.99398511	0.00601489
1	0	1	0.00564921	-0.00564921
0	1	1	0.00577015	-0.00577015
1	1	0	0.99434905	0.00565095

5. SONUÇ

Sonuç olarak; projenin amacı XOR probleminin yapay sinir ağlarında tek katmanlı algılayıcı modellerle çözülemeyeceğine ve çok katmanlı algılayıcı modelin yapısının temelini oluşturduğu görülmektir. Çok katmanlı algılayıcı model sınıflandırma, tanıma ve genelleme yapma gerektiren problemler için önemli bir araç olduğu gözlenmektedir**.** Çok katmanlı algılayıcı ağlar sayesinde XOR probleminde kullanılan geri yayılım algoritmasının adımlarını tek tek öğrenerek ve kullanarak NXOR problemi çözülmüştür. Bu öğrenme kuralındaki amaç hata oranını en aza indirmek olmuştur. NXOR problemindeki dört örnek setin ağ çıktıları beklenen çıktılarla karşılaştırılmış ve hata oranları gözlenmiştir. Bu programı yazarken Python kullanılmış ve Pyqt5 kütüphanesinden yararlanarak arayüz tasarlanmıştır.

KAYNAKÇA

[1] Şahin Bayzan, Meriç Çetin, Alper Uğur – "İleri Beslemeli Yapay Sinir Ağlarında Backprogation (Geriye Yayılım) Algoritmasının Sezgisel Yaklaşımı", https://www.academia.edu/6712812/İleri_Beslemeli_Yapay_Sinir_Ağlarında_Backpropagation_Geriye_Yayılım_Algoritmasının_Sezgisel_Yaklaşımı, Erişim: 1 Ocak 2021

[2] İnan Güler, Elif Übeyli – "Çok Katmanlı Perseptron Sinir Ağları ile Diyabet Hastalığının Teşhisi", https://dergipark.org.tr/en/pub/gazimmfd/issue/6668/89367, Erişim: 1 Ocak 2021

[3] Devran Deniz – "Yapay Sinir Ağlarıyla Hisse Senedi Fiyatları ve Yönlerinin Tahmini" https://www.researchgate.net/profile/Devran_Deniz2/publication/318913240_Yapay_Sinir_Aglariyla_Hisse_Senedi_Fiyatlari_ve_Yonlerinin_Tahmini/links/5984e0b3aca27266ad9a28d2/Yapay-Sinir-Aglariyla-Hisse-Senedi-Fiyatlari-ve-Yoenlerinin-Tahmini.pdf, Erişim: 1 Ocak 2021

[4] Selahattin Yavuz, Muhammet Deveci – "İstatistiksel Normalizasyon Tekniklerinin Yapay Sinir Ağın Performansına Etkisi", http://iibf.erciyes.edu.tr/dergi/sayi40/ERUJFEAS_Jun2012_167to187.pdf, Erişim: 1 Ocak 2021

[5] Selahattin Yavuz, Muhammet Deveci – "İstatistiksel Normalizasyon Tekniklerinin Yapay Sinir Ağın Performansına Etkisi", http://iibf.erciyes.edu.tr/dergi/sayi40/ERUJFEAS_Jun2012_167to187.pdf, Erişim: 1 Ocak 2021

[6] Yücel Gönül, Şahin Ulu, Abdülkadir Bucak, Abdülkadir Bilir – "Yapay Sinir Ağları ve Klinik Araştırmalarda Kullanımı" http://geneltip.org/upload/sayi/89/GTD-00718.pdf, Erişim: 1 Ocak 2021

[7] Prof. Dr. Ercan Öztemel – "Yapay Sinir Ağları Ders Kitabı", İstanbul, Papatya Yayıncılık, Erişim: 1 Ocak 2021

[8] Derin Öğrenme – "Yapay Sinir Ağları" http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/, Erişim: 1 Ocak 2021

[9] M. Orkun ÖĞÜCÜ – "Yapay Sinir Ağları ile Sistem Tanıma" https://polen.itu.edu.tr/bitstream/11527/5230/1/4200.pdf Erişim: 1 Ocak 2021

[10] M. Orkun ÖĞÜCÜ – "Yapay Sinir Ağları ile Sistem Tanıma" https://polen.itu.edu.tr/bitstream/11527/5230/1/4200.pdf Erişim: 1 Ocak 2021

[11] Prof. Dr. Ercan Öztemel – "Yapay Sinir Ağları Ders Kitabı", İstanbul, Papatya Yayıncılık, Erişim: 1 Ocak 2021

[12] Derin Öğrenme – "Yapay Sinir Ağları", http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/, Erişim: 1 Ocak 2021

[13] Murat LORTOĞLU – "EPGA Tabanlı Yapay Sinir Ağı Kullanılarak Buğday Türlerinin Sınıflandırılması", http://acikerisim.karatay.edu.tr:8080/xmlui/bitstream/handle/20.500.12498/1592/569998.pdf?sequence=1&isAllowed=y, Erişim: 1 Ocak 2021

[14] Derin Öğrenme – "Yapay Sinir Ağları" http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/, Erişim: 1 Ocak 2021

[15] Prof. Dr. Novruz Allahverdi – "Yapay Sinir Ağları Ders Notları", http://novruzallahverdi.karatay.edu.tr/kategori/yapay-sinir-aglari, Erişim: 1 Ocak 2021

[16] Ali ÖZDEMİR, Melih İNAL– "Melez Tek Bir İşlem Birimli Yapay Sinir Ağı Hücresinin Otomatik Öğrenme Veritabanlarına Uygulanması", http://dspace.kocaeli.edu.tr:8080/xmlui/bitstream/handle/11493/927/197939.pdf?sequence=1&isAllowed=y, Erişim: 1 Ocak 2021

[17] Prof. Dr. Ercan Öztemel – "Yapay Sinir Ağları Ders Kitabı", İstanbul, Papatya Yayıncılık, Erişim: 1 Ocak 2021