SİNİR AĞLARI İLE İSTATİSTİKSEL MODELLEME VE BİR UYGULAMA DENEMESİ

Halil ERYILMAZ

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü
İstatistik Anabilim Dalı

Mayıs – 2004

ÖZET

Yüksek Lisans Tezi

SİNİR AĞLARI İLE İSTATİSTİKSEL MODELLEME VE BİR UYGULAMA DENEMESİ

HALİL ERYILMAZ

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
İstatistik Anabilim Dalı

Danışman: Prof.Dr. Ali Fuat YÜZER
2004, 104 sayfa

İnsan beyninin çalışma prensibini taklit eden yapay sinir ağları, herhangi bir problem hakkında girdiler ve çıktılar arasındaki ilişkiyi (doğrusal olsun veya olmasın), elde bulunan mevcut örneklerden genelleme yaparak daha önce hiç görülmemiş veya uygulanmamış olan örneklere kabul edilebilir çözümler üretebilirler.

1990'lı yılların başlangıcında yayınlanan bazı kaynaklarda bazı yapay sinir ağları modelleri ile bazı istatistik tekniklerin benzer hatta bazılarının aynı olduğuna dikkat çekilmiştir. Sonraki çalışmalar ise bunun tesadüfi olmadığını, bu iki alanın birbirleriyle ileri derecede ilişkili olduğunu göstermiştir. Yapay sinir ağı modelleri ile istatistik tekniklerin karşılaştırılması birinin diğerinin geliştirilmesinde önemli olduğunu ortaya çıkarmıştır. Perseptron, çok katmanlı perseptron gibi bazı yapay sinir ağı modelleri istatistiksel uygulamalar için faydalı olabileceği bazı bilim adamları tarafından ileri sürülmektedir.

Bu çalışmanın amacı, yapay sinir ağlarını tanıtmak, bazı yapay sinir ağ modelleri ile bazı istatistiksel teknikler arasındaki ilişkiyi incelemek ve bir uygulama üzerinde yapay sinir ağı modellerin istatistiksel uygulamalardaki kullanılabilirliği ve etkinliğini araştırmaktır.

Anahtar Kelimeler: Yapay Sinir Ağı, Perseptron, Çok Katmanlı Perseptron, Geriye Yayılım Ağı, Lojistik Regresyon

ABSTRACT

Master of Science Thesis

STATISTICAL MODELLING WITH NEURAL NETWORKS AND AN ATTEMPT FOR AN APPLICATON

Halil ERYILMAZ

Anadolu University
Graduate School of Natural and Applied Sciences
Statistics Program

Supervisor: Prof. Ali Fuat YÜZER
2004, 104 pages

Imitiating the principles of human brain, Neural Networks can produce meaningful answers for never seen or never encountered problems via creating generalizations from the present information for the relationships (linear or not) between input and output.

In some sources published early 1990's, it's brought forward that there were some resembles between some neural network models and statistical techniques and even for some cases there was equality. Later studies showed that it was not just pure coincidince but these two scientific areas were related to each other extensively. Comparison neural networks and statistical techniques showed that either is useful for creating better results on the other. A few scientist put forward that some of the neural network models like perceptron or multilayer perceptron were actually very useful for statistical applications.

The aim of this study is to explain neural networks, to show the relationships between neural network models and statistical techniques and using a real life application to show efficiency and applicability of neural networks in statistical problems.

Keywords: Neural Network, Perceptron, Multilayer Perceptron, Backpropagation Net, Logistic Regression

TEŞEKKÜR

Bu çalışmanın her aşamasında değerli ilgi ve yardımlarını esirgemeyen danışman hocam Sayın Prof. Dr. Ali Fuat YÜZER'e, çalışmamın gerçekleşmesinde katkılarıyla bana yön veren hocalarım Sayın Doç. Dr. Memmedağa MEMMEDOV ve Sayın Prof. Dr. Embiya AĞAOĞLU'na, en içten teşekkürlerimi bir borç bilirim.

Ayrıca, bu süre boyunca manevi desteklerini gördüğüm değerli hocalarıma, arkadaşlarıma ve aileme sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Yapay zeka dallarından biri olan yapay sinir ağları, günümüzde sadece yapay zeka çalışmalarında değil, bir çok bilim dalında direkt olarak kullanılmaktadır. Belirli yapay sinir ağı modellerinin bazı istatistiksel tekniklerle olan benzerliği dikkat çekmektedir. [1]

Efsaneye göre kendi ırkını mahveden Zeus'a karşı içinde büyük bir kin ve öfke olan Prometheus, tanrılarını inkar edecek, onları hiçe sayacak ve işleyecekleri kötülüklerle en vahşi hayvanlara bile taş çıkartacak, dünyanın başına bela olacak bir mahluku, insanı yaratarak intikam almaya karar verdi.

Prometheus ilk insanı çamuru göz yaşlarıyla karıştırarak yarattı. Buna aslanın gücünü, tavusun kibrini, tilkinin kurnazlığını ve tavşanın ürkekliğini kattı. Fakat insan çıplaktı, kendisini koruyacak hiç bir şeye sahip değildi. Doğduğu günden itibaren acıları, üzüntüleri ve bitmek bilmeyen ihtiyaçları başlıyordu. İlk insan çiğ meyvalarla, kanlı etlerle beslenip, elbise yerine bitkilerin yapraklarına sarılıyorlardı. Güneşin faydalarını bilmeden kendilerini karanlık oyuklarda saklıyorlardı. Yarattığı mahluklara acıyan Prometheus insanları daha iyi bir şekilde yaşatabilmek, vahşi hayvanlara karşı etkili silahlarla koruyabilmek, toprağı sürmeye yarayacak gerekli aletleri elde edebilmek için onlara madenleri işlemeyi öğretmeye ve ateşi vermeye karar verdi.

Lemnos adasına gitti ve ateşten bir kıvılcım çalarak ilahi bir armağan olarak insanlara götürdü. O günden itibaren insanlar ateşin yardımıyla daha iyi yaşamaya başladılar. Yiyeceklerini pişiriyorlar, soğuk havada ısınıyorlar, karanlık mağaralarda çıralı odunları yakarak birbirlerinin yüzlerini görüyorlardı. Fakat bir süre sonra nerden geldiklerini unutarak kendilerini tanrılarla eşit tutmaya başladılar. Aslında Zeus, onların böyle şımarık davranacaklarını önceden tahmin ettiği için onlara ateşi vermemişti. [2]

Başta Zeus olmak üzere tüm tanrıların öfkesini çeken Prometheus, sadece ateşi çalmakla kalmamış, Olympus'a ait olan zekayı, medeniyetini geliştirmesi için insanoğluna hediye etmiştir. Zeus'un, şımarık davranacakları ve nerden geldiklerini unutup kendisini tanrılara eşit tutacağını önceden tahmin ederek ateşi vermediği insanoğlu, günümüzde sahip olduğu zekayı, yapay zeka (**Artificial Intelligence**) adı altındaki çalışmalarla makinelere vermeye çalışmaktadır.

Yapay zeka araştırmacılarının en büyük hedeflerinden biri milyarlarca nöron (beyin hücresi) içeren insan beyninin çalışmasını taklit eden sistemler yaratmaktır. [3]

Günümüzde yapay zeka hızla gelişmekte olup bir çok dallara ayrılmıştır. Bu dallara,

- **Yapay Sinir Ağları (Artificial Neural Networks)**
- Bulanık Mantık Teorisi
- Genetik Algolaritmalar
- Uzman Sistemler
- Robotik
- v.s. ...

örnek olarak gösterilebilir.

Yapay zeka dallarından biri olan yapay sinir ağları, yapay zeka araştırmalarında önemli bir görev üstlenmektedir. Günümüzde, yapay sinir ağlarının kullanılmasıyla geliştirilen yazılımlar, geleceği nasıl değiştirebileceğini gösterebilmektedir.

İlk yapay sinir ağı modeli 1943 yılında Warren McCulloch ve Walter Pitts tarafından gerçekleştirilmiştir. 1980'li yıllar ise yapay sinir ağları ile ilgili çalışmalar için bir atılım dönemi olmuştur.

Yapay sinir ağları bilgi sınıflama ve bilgi yorumlamanın da içinde bulunduğu çok değişik problemlerin çözümünde kullanılmanın yanı sıra iş hayatı, finans, endüstri ve eğitim alanlarında var olan yöntemlerin yerine başarıyla uygulanmaktadır. [4]

1990'lı yılların başlangıcında yayınlanan bazı kaynaklarda bazı yapay sinir ağları modelleri ile bazı istatistik tekniklerin benzer hatta bazılarının aynı olduğuna dikkat çekilmiştir. Sonraki çalışmalar ise bunun tesadüfi olmadığını, bu iki alanın birbirleriyle ileri derecede ilişkili olduğunu göstermiştir. Yapay sinir ağı modelleri ile istatistik tekniklerin karşılaştırılması birinin diğerinin geliştirilmesinde önemli olduğunu ortaya çıkarmıştır. Perseptron, çok katmanlı perseptron gibi bazı yapay sinir ağı modelleri istatistiksel uygulamalar için faydalı

olabileceği ve aynı şekilde tahmin ölçütü, güven aralıkları, diagnostik yöntemler gibi bazı istatistiksel tekniklerin de yapay sinir ağ uygulamalarına uygulanabileceği bazı bilim adamları tarafından ileri sürülmektedir.

Yapay sinir ağları ve istatistiksel metodoloji arasındaki iletişimin geliştirilmesi her iki alan için de büyük yarar sağlamaktadır. [1]

Bu çalışmada, yapay sinir ağ modellerine değinilecek, bazı istatistiksel teknikler ile bazı yapay sinir ağı modelleri arasındaki ilişki incelenecek ve bir uygulama üzerinde yapay sinir ağı modellerin istatistiksel uygulamalardaki kullanılabilirliği ve etkinliği araştırılacaktır.

2. YAPAY SİNİR AĞLARI

2.1. Yapay Sinir Ağı

Bir yapay sinir ağı, biyolojik sinir ağlarının karakteristiklerine benzeyen karakteristiklere sahip bilgi işleme sistemidir. Yapay sinir ağı modelleri, insanın kavrama özelliği ve biyolojik nöron yapısının matematiksel modellerinin genelleştirilmesi sonucunda aşağıdaki varsayımlara dayandırılarak oluşturulmuşlardır:

- Bilgi işleme nöron adı verilen birçok basit elemanlarda gerçekleşir.
- Sinyaller, nöronlar arasındaki ilişkiyi sağlayan bağlantılar üzerinden gerçekleşir.
- Her bir bağlantı bir ağırlık değerine sahiptir. Biyolojik nöronlarda olduğu gibi, bu ağırlık değerleri gönderilen sinyallerle çarpılır;
- Her nöron, bir çıkış sinyalini belirleyen bir aktivasyon fonksiyonu uygular (genelde bu doğrusal olmayan bir fonksiyondur).

Böylelikle, bir nöron ağı;

- nöronlar ve onlar arasındaki bağlantı (nöron yapısı),
- bağlantılardaki ağırlıkların hesaplanması metotları (eğitim algoritmaları),
- aktivasyon fonksiyonu

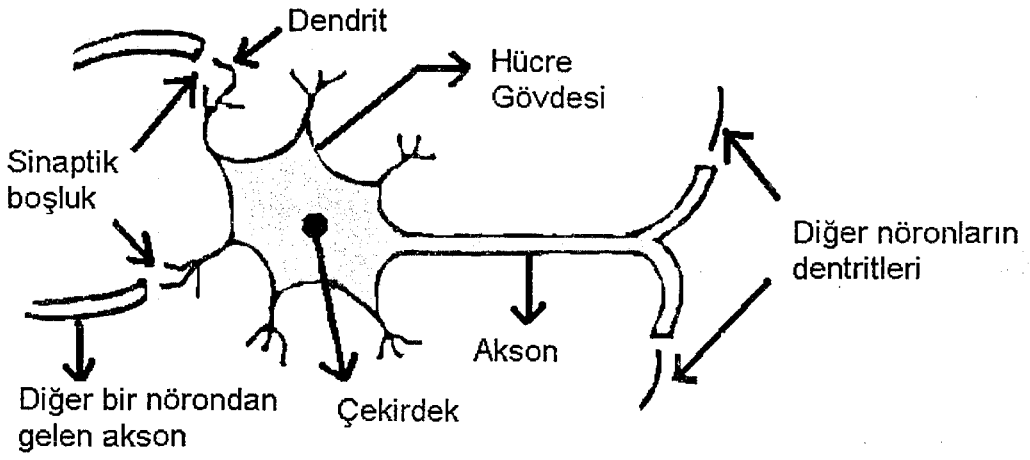
şeklinde tanımlanabilir. [5]

2.2. Biyolojik Sinir Ağları

Günümüzde yapay sinir ağı modelleri, biyolojik sinir sisteminin karmaşıklığını yansıtmaktan henüz çok uzaktır. Ancak bir çok yapay sinir ağı modelinin tasarlanmasında, kısıtlı da olsa biyolojik sinir sisteminin çok basit düzeyde de olsa tanıtılmasında yarar bulunmaktadır. [6]

Şekil 2.1.'de gösterilmiş olan bir biyolojik nöron, üç tip bileşene sahiptir: Dentrit, Soma ve Aksonlar. Dentritler sinyalleri diğer nöronlardan alırlar. Sinyaller, dentritler arasındaki sinaptik boşluklardan kimyasal süreçlerin yardımıyla iletilen elektrik tepileridir. Kimyasal vericilerin hareketi giren

sinyalleri biraz değiştirmektedir. Bu bir anlamda, yapay sinir ağlarındaki ağırlıklara benzemektedir.



Şekil 2.1. Biyolojik nöronun yapısı

Soma veya hücre gövdesi, giren sinyalleri toplar. Yeterli girdi alındığında (eşik değerden büyük olduğunda) hücre ateşler. Bu ateşleme, nöronun sinyalini aksonundan başka bir nöronun aksonuna aktarması demektir. Genelde hücrenin herhangi bir kısa zaman anında ateşleme yaptığı sanılmakta ve bu ikili sinyal gibi ele alınmaktadır. Ateşlenme frekansı çeşitlenir ve sinyal daha çok şiddetli veya daha az şiddetli sinyal olarak görülebilir.

Belirli bir nöron tarafından sinyalin aktarılması işlemi, nöronların aksonlarının kılıfları arasındaki iyon konsantrasyonun farkından dolayı gerçekleşir. İyonların bulunması daha çok potasyum, sodyum ve klor gerektirir.

Yapay sinir ağı işlem elemanlarının birkaç önemli özelliği, biyolojik nöronların özelliklerini akla getirmektedir:

- Her bir işlem elemanı (nöron), farklı sinyaller alabilir.
- Alınan sinyaller sinapstaki ağırlıklar tarafından değiştirilebilir.
- İşlem elemanları (nöronlar), ağırlıklı girdileri toplar.
- Uygun koşullar altında başka bir ifadeyle yeterli girdi sağlandığında, işlem elemanı (nöron) çıkış sinyalini aktarır .
- Belirli bir nöronun çıkışı diğer birçok nörona gidebilir.

Yapay sinir ağlarının biyolojik nöronlardan aldığı önemli diğer bazı özellikler aşağıda verilmiştir:

- Bilgi işleme yereldir.
- Bellek ikiye ayrılmıştır:

  a) Uzun dönemli bellek (nöronun sinapslarındaki ağırlık değerlerine bağımlı olur).

  b) Kısa dönemli beliek (nöronlar tarafından gönderilmiş daha hızlı sinyallere uymaktadır).
- Bir sinapsın gücü (etkisi) deneyimlerle (eğitimle) değiştirilebilir.
- Nöroaktarıcılar, sinaplar için hızlandırıcı (heyecanlandırıcı) ya da yavaşlatıcı (engelleyici) olabilir.

Yapay sinir ağlarının biyolojik sinir sistemiyle paylaştığı başka bir önemli özellik de hata toleransıdır. Biyolojik sinir sisteminde iki hata toleransı olabilir. Birincisi, girdi sinyallerinde daha önceden görülmemiş bir sinyalle karşılaşılabilir. Buna örnek olarak, daha önceden hiç görülmemiş yada çok uzun süre önce görülmüş bir insanı, bir resmin içinde tanıma yeteneği verilebilir. İkincisi ise; sinir sisteminin kendisine hasar vermesine tahammül edilebilir. Bir insan, 100 milyardan daha fazla nöron ile doğar. Bunların büyük bir kısmı beyindedir ve bir çoğu da öldüklerinde yenilenmezler. Ama birçok nöronun kaybedilmesine rağmen öğrenmeye devam edilir. Sarsıcı nöron kayıplarında bile diğer nöronlar ölen nöronların fonksiyonlarını gerçekleştirmek için eğitilebilirler. Benzer şekilde, Yapay sinir ağlarında, sinir ağındaki küçük hasarlara karşı duyarsız şekilde tasarlanabilirler ve sinir ağı önemli hasarlar sonucunda tekrar eğitilebilir.

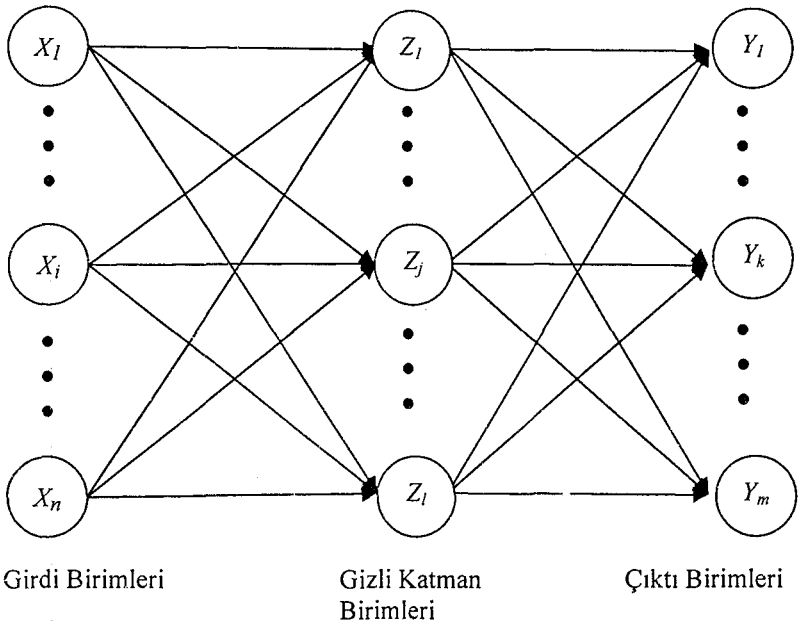
Kullanımı biyolojik nöronların modeli olarak tasarlanmayan yapay sinir ağlarında, biyolojik sinir ağların işlemsel özelliklerinin geliştirilmesi başarısına kalkınmıştır. [5]

2.3. Yapay Sinir Ağının Bileşenleri

Yapay sinir ağı modelleri (Artificial Neural Net Models); Connectionist Modeller, Paralel Dağıtılmış İşleme Modelleri (Paralel Distributed Processing Models) veya Neuronorphic Sistem gibi değişik şekillerde isimlendirilmektedir. Ancak ismi ne olursa olsun tüm bu modeller, biyolojik sinir sisteminin bilinen yapısını göz önüne alarak, yüksek bir performansın elde edilmesini sağlayacak şekilde basit hesaplama elemanlarının yoğun bağlantılarından meydana gelmiştir.

Genel olarak bir yapay sinir ağı modeli Şekil 2.2'de görüldüğü gibi *n* adet katman (layer), her katmanda biyolojik sinir hücrelerine benzer işlevi yerine getiren ve değişik sayılarda olabilen hesaplama elemanları, katmanlar boyunca bu hesaplama elemanları arasındaki yoğun bağlantılardan meydana gelmektedir. Çeşitli yapay sinir ağı modellerinde kullanılan hesaplama elemanları, düğüm (node), birim (unit), işlem elemanı (processing element), yapay sinir hücresi (artificial neuron) veya kısaca nöron olarak isimlendirilmektedir. [6,7]

Bu çalışmada nöron ifadesi kullanılacaktır.



Şekil 2.2. Genel bir yapay sinir ağı modeli

Nöron, bir ağın temel işlem elemanıdır. [8] Ağ içinde yer alan tüm nöronlar bir veya birden fazla girdi alırlar ve tek bir çıktı verirler. Bu çıktı yapay sinir ağının dışına verilen çıktılar olabileceği gibi başka nöronlara girdi olarak da kullanılabilir. [6,7]

Bir yapay sinir ağının en önemli unsurlarından biri de bağlantılardır. Her bir bağlantı aynı zamanda bir ağırlık (weight) değerine sahiptir. Yapay sinir ağı içinde girdilerin nöronlar arasında iletimini sağlayan tüm bağlantılar farklı ağırlık değerine sahiptirler. Kısacası ağırlıklar, sinir ağının bir problemi çözmesi için gerekli olan bilgiyi temsil etmektedir. [7,8]

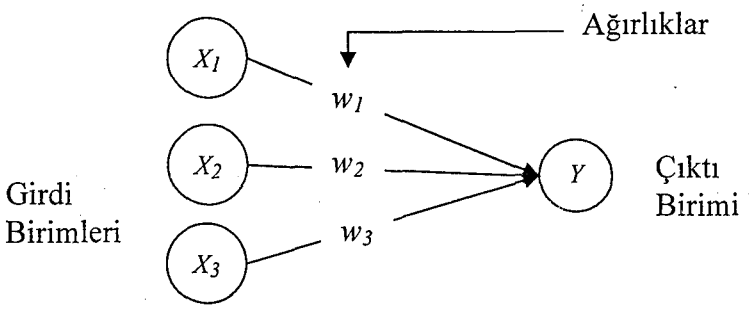
Nöronların aynı doğrultu üzerinde bir araya gelmeleri ile katmanlar oluşmaktadır. [9]

Yapay sinir ağında ilk katman girdi katmanıdır ve dışarıdan verilerin yapay sinir ağına alınmasını sağlar. Diğer katman ise bilgilerin dışarıya iletildiği çıktı katmanıdır. Girdi ve çıktı katmanlarının arasında katman varsa bunlara gizli katman (hidden layer) adı verilir. Bir yapay sinir ağında gizli katman olması gerekmediği gibi, birden fazla gizli katmanda bulunabilir.

Her nöronun bir iç durumu (hali) vardır. Bu durum aktivasyon seviyesi olarak adlandırılır ve alınan giriş değerlerinin bir fonksiyonudur. Genellikle, bir nöron aktivasyonunu bir sinyal olarak diğer nöronlara gönderir. Her aşamada, bir nöron sadece tek bir sinyal gönderir ama bu sinyal aynı anda birden fazla nörona gönderilebilir.

Şekil 2.3.'de örnek olarak gösterilen Y nöronu, X_1, X_2 ve X_3 nöronlarından giriş sinyalleri alır. Bu nöronların aktivasyonları (çıkış sinyalleri) sırasıyla x_1, x_2 ve x_3' tür. Y nöronunu X_1, X_2 ve X_3 nöronları ile bağlayan ağırlıklar ise sırasıyla w_1, w_2 ve w_3'tür. y_in, ağ girişi olmakla birlikte X_1, X_2 ve X_3 nöronlarından Y nöronuna gelen ağırlıklı sinyallerin toplamıdır:

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3 \tag{2.1}$$



Şekil 2.3. Basit bir yapay nöron

Y nöronun aktivasyonu y, ağın giriş değerlerinin bir fonksiyonu olarak tanımlanır ve bu fonksiyon için farklı aktivasyon fonksiyonları kullanılabilir: [5]

$$y = f(y\_in) \tag{2.2}$$

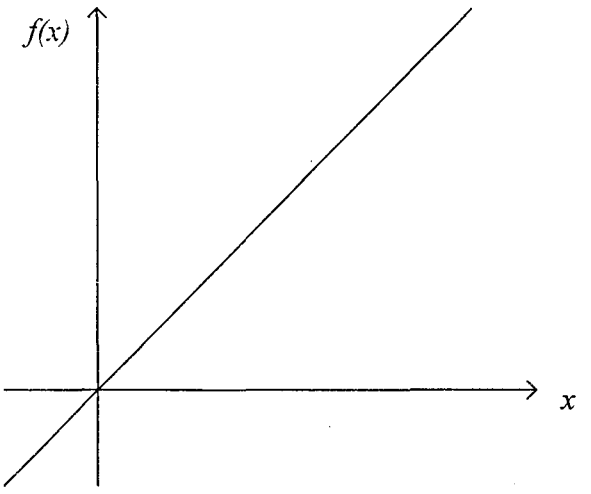
2.4. Aktivasyon Fonksiyonları

Daha önceden de bahsedildiği gibi, bir yapay sinir ağında yapılması gereken en temel işlemler, ağırlıklı girdi değerlerinin toplanması ve bir aktivasyon fonksiyonu uygulanmasıdır.

Yapay sinir ağı uygulamalarında yaygın olarak kullanılan bazı aktivasyon fonksiyonları aşağıda gösterilmiştir:

2.4.1. Özdeşlik Fonksiyonu

Genellikle girdi değerleri için kullanılan aktivasyon fonksiyonu Şekil 2.4.'de gösterilen ve tüm x'ler için $f(x) = x$ olan *özdeşlik fonksiyonu*'dur.



Şekil 2.4. Özdeşlik fonksiyonu

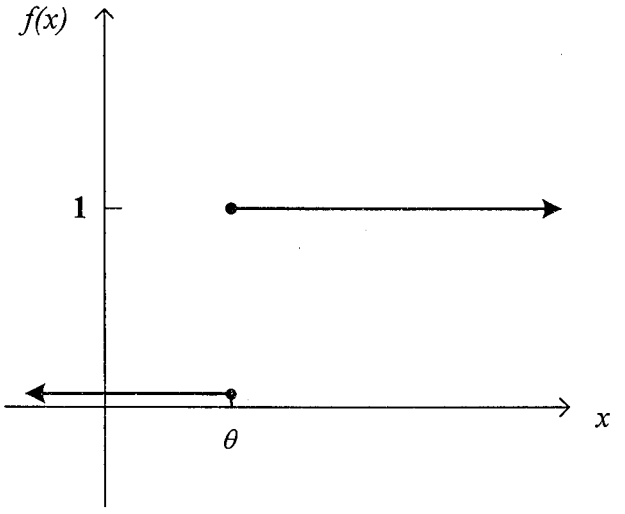
Belirli bir tabakadaki tüm nöronlar için aynı aktivasyon fonksiyonu kullanılabilmektedir. Birçok durumda doğrusal olmayan aktivasyon fonksiyonları kullanılır. Tek katmanlı ağlara göre çok katmanlı ağların avantajlarından yararlanabilmek için, doğrusal olmayan aktivasyon fonksiyonlarının kullanılması gerekmektedir.

2.4.2. θ Eşik Değeri ile Verilen İkili Adım (Heaviside) Fonksiyonu

Şekil 2.5.'te gösterilen θ eşik değeri ile verilen ikili adım fonksiyonu,

$$f(x)=\begin{cases}0 & , \ x<\theta \text{ ise} \\ 1 & , \ x\geq\theta \text{ ise}\end{cases} \tag{2.3}$$

formülü ile verilebilir.



Şekil 2.5. İkili adım fonksiyonu

Tek katmanlı ağlar, genelde, adım fonksiyonunun giriş değerlerini, değeri ikili (0 ya da 1 veya -1 ya da 1) olan çıkış değerlerine çevirmek için kullanılmaktadırlar.

2.4.3. Sigmoid Fonksiyonları

Sigmoid fonksiyonları, oldukça kullanışlı aktivasyon fonksiyonlarıdır. Sigmoid fonksiyonları içinde *lojistik* ve *hiperbolik tanjant* fonksiyonları en yaygın olarak kullanılanlarıdır. Özellikle yapay sinir ağı modellerinden biri olan ve uygulamada sıkça kullanılan geriye beslemeli öğrenme algoritmalarında bu fonksiyonların kullanımı diğerlerine göre daha avantajlıdır. Çünkü fonksiyonun belirli bir noktadaki değeri ile onun türevinin değeri arasındaki ilişki öğrenme zamanındaki hesap yükünü azaltmaktadırlar.

2.4.3.1. Lojistik Fonksiyon

Lojistik fonksiyon, değerleri 0 ile 1 arasında değişen bir sigmoid fonksiyonudur ve yapay sinir ağları için aktivasyon fonksiyonu olarak sıkça

kullanılmaktadır. Fonksiyonun aralık değerini vurgulamak için bu fonksiyona *ikili sigmoid* adı verilmekle birlikte *lojistik sigmoid* adı da verilmektedir.

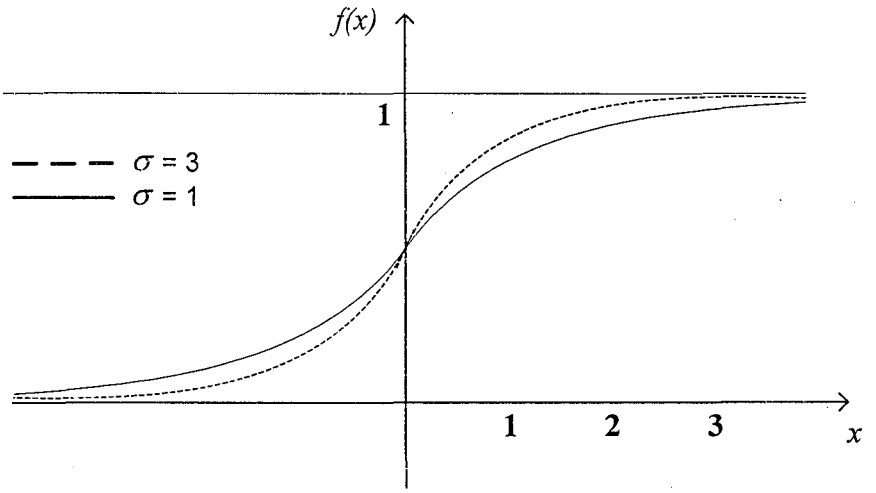
Şekil 2.6'da fonksiyonun adım parametresi olan σ'nın farklı değerleri için ikili sigmoid eğrilerini gösteren bu fonksiyon,

$$f(x)=\frac{1}{1+e^{-\sigma x}} \tag{2.4}$$

veya türevi olan

$$f'(x)=\sigma f(x)\left[1-f(x)\right] \tag{2.5}$$

formülleri ile hesaplanmaktadır.



Şekil 2.6. İkili sigmoid fonksiyonu ($\sigma=1$ ve $\sigma=3$ için)

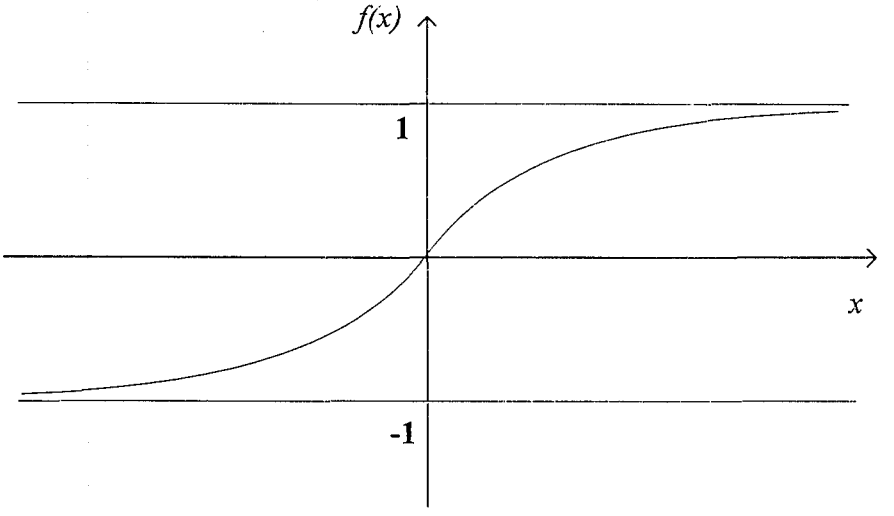
Lojistik sigmoid fonksiyonu istenen değer aralığına göre ölçeklenebilir ve böylece probleme uygun bir fonksiyon haline gelebilir. En yaygın kullanılan aralık, −1 ile 1 aralığıdır. Bu sigmoid fonksiyonu, *iki kutuplu sigmoid* olarak adlandırılır. Şekil 2.7'de $\sigma=1$ için gösterilen iki kutuplu sigmoid fonksiyonu,

$$g(x)=2f(x)-1=\frac{1-e^{-\sigma x}}{1+e^{-\sigma x}} \tag{2.6}$$

ve onun türevi olan

$$g'(x)=\frac{\sigma}{2}[1+g(x)][1-g(x)] \qquad (2.7)$$

formülleriyle hesaplanmaktadır.



Şekil 2.7. İki kutuplu sigmoid fonksiyonu ($\sigma = 1$ için)

2.4.3.2. Hiperbolik Tanjant Fonksiyonu

Hiperbolik tanjant fonksiyonu da istenilen çıkış aralığı (–1 ile 1) arasında ise aktivasyon fonksiyonu olarak kullanılmaktadır. İki kutuplu sigmoid fonksiyonuna çok yakın olan bu fonksiyon,

$$h(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}=\frac{1-e^{-2x}}{1+e^{-2x}} \qquad (2.8)$$

ve ya türevi olan

$$h'(x)=[1+h(x)][1-h(x)] \qquad (2.9)$$

formülleri ile hesaplanmaktadır.

Eğer ikili veri 0 ile 1 arasında değişiyorsa bunu iki kutuplu biçime sokarak iki kutuplu sigmoid ya da hiperbolik tanjant fonksiyonunda kullanmak genelde tercih edilen bir yöntemdir. [5,32]

2.5. Yapay Sinir Ağı Mimarileri

Nöronların katmanlar içindeki yerleşimleri ve diğer katmanlardaki nöronlarla olan bağlanma şekilleri *ağ mimarisi* olarak adlandırılmaktadır. Her nöron ağ girdi katmanına sahiptir ve bu katmandaki nöronların aktivasyon fonksiyonu her nöron için girdi sinyaline eşittir. Başka bir ifadeyle, herhangi bir fonksiyon uygulanmadan direkt olarak girdileri kabul etmektedirler. Örnek olarak, şekil 2.2.'de gösterilen ağ, n adet girdi birimi, l adet gizli birim ve m adet çıkış biriminden oluşmaktadır.

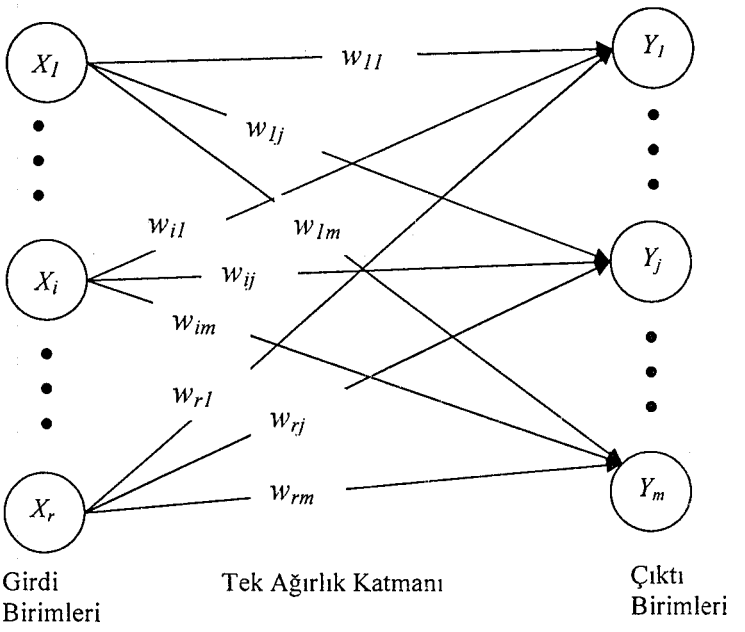
Çoğu kez, nöronların katmanlar şeklinde yerleştiği düşünebilir. Genel olarak, aynı katman içindeki nöronlar aynı davranıştadırlar. Nöronların davranışlarını belirleyen temel faktörler aktivasyon fonksiyonları ve sinyallerin gönderildiği bağlantılar üzerindeki ağırlıklardır. Her katmanın içindeki nöronlar genellikle aynı aktivasyon fonksiyonuna sahiptirler ve diğer nöronlara aynı bağlantı modeliyle bağlanırlar. Bir katmanın içindeki nöronların hepsi tam olarak diğer tüm nöronlara bağlanmış olabileceği gibi bağlanmamış da olabilirler. Eğer bir katmanın içindeki her nöron (örneğin, gizli birimin katmanı) başka bir katmandaki nörona bağlanmışsa, gizli birim her çıkış nöronuna bağlanmış olacaktır.

Yapay sinir ağları, çoğu kez *tek katmanlı* veya *çok katmanlı yapay sinir ağları* olarak sınıflandırılırlar. Bu sınıflandırmaya ek olarak da *rekabetçi katmana sahip yapay sinir ağları* da ilave edilebilir.

Katman sayısını belirlerken, girdi birimi bir katman olarak sayılmaz, çünkü bunlar üzerinde hiçbir hesaplama işlemi yoktur. Bir ağ içindeki katman sayısı nöronları bağlayan ağırlıklı bağlantıların katman sayısına eşittir. Bu duruma örnek olarak şekil 2.2'deki iki adet ağırlık katmanını göstermek mümkündür.

2.5.1. Tek Katmanlı Yapay Sinir Ağı

Tek katmanlı yapay sinir ağlarında bir tane ağırlıklı bağlantı katmanı bulunur. Çoğu kez, birimler sinyalleri alan girdi birimi ve ağın cevabının alınacağı çıktı birimleri olmak üzere ikiye ayrılmaktadır. Tipik bir tek katmanlı ağ şekil 2.8.'de verilmiştir.

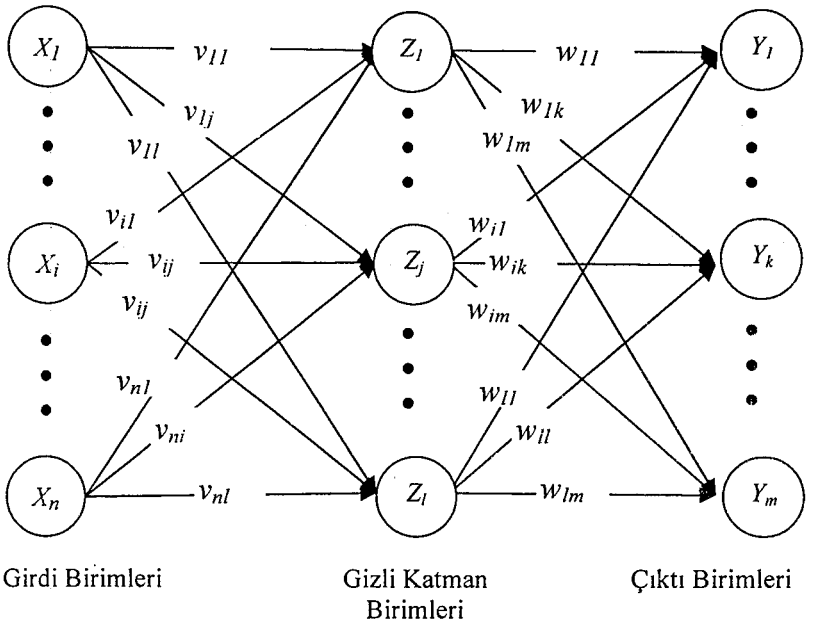


Şekil 2.8. Tek katmanlı bir yapay sinir ağı

Tek katmanlı yapay sinir ağları için bir çıkış birimini etkileyen ağırlıklar başka bir çıkış birimini etkilemezler. Örnek sınıflandırma problemleri için bu mimari kullanılabilir. Örnek ile ilişkili üretilmiş olan girdi sinyalleri sayesinde örnek için cevabın çıktı birimlerinden alınmasını sağlar. Sinir ağından alınacak cevabın yorumuna göre, aynı mimarili ağ farklı problemlerin çözümü için kullanılabilir.

2.5.2. Çok Katmanlı Yapay Sinir Ağları

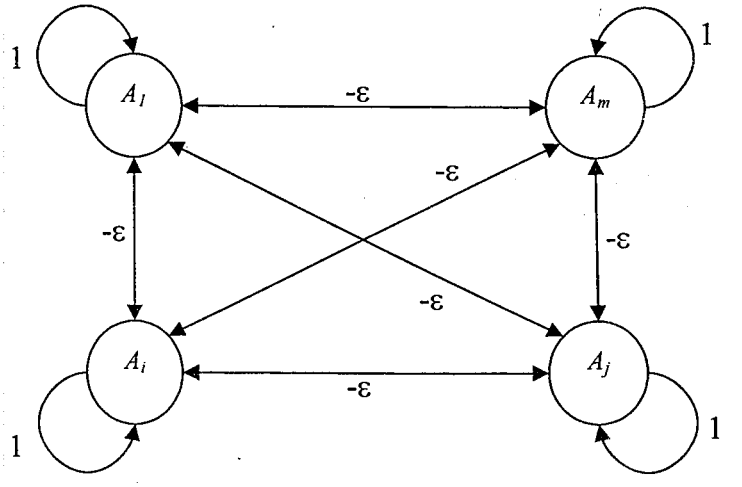
Çok katmanlı yapay sinir ağlarında, girdi birimleri ile gizli birimlerin arasında bir yada birden fazla katman bulunmaktadır. Genellikle, gizli ve çıktı birimleri arasında ağırlıklı bağlantı katmanı bulunan bu tip ağ mimarileri tek katmanlı ağ mimarilerine göre daha karmaşık problemleri çözebilir. Eğitilmeleri çok daha zor olabilen çok katmanlı ağlar, bazı tek katmanlı ağların problemi tam olarak çözecek şekilde eğitilemediği durumlarda son derece başarılı olabilirler. [5] Tipik bir tek katmanlı ağ şekil 2.9.'da gösterilmiştir:



Şekil 2.9. İki katmanlı bir yapay sinir ağı

2.5.3. Rekabetçi Katmana Sahip Yapay Sinir Ağları

Bir rekabetçi katman *(competitive layer)* çok fazla sayıdaki yapay sinir ağlarının bir parçasının şeklidir. Genellikle, rekabetçi katmanda nöronlar arasındaki bağlantılar ağın mimari diyagramında gösterilmez. Rekabetçi bağlantılar, $-\epsilon$ ağırlıklarına sahiptirler. Rekabetçi katmana bir örnek Şekil 2.10'da gösterilmiştir: [4,5]



Şekil 2.10. Rekabetçi katmana sahip bir yapay sinir ağı

2.6. Yapay Sinir Ağının Eğitilmesi (Ağırlıkların Ayarlanması)

Birbirlerinden farklı yapay sinir ağı karakteristiklerini birbirinden ayıran önemli bir etken, onların eğitilmeleri esnasında bağlantı ağırlıkları değerlerinin ayarlanması için kullanan yöntemlerdir. Bunun için genel olarak iki öğrenme yöntemi vardır. Bu öğrenme yöntemleri aşağıda açıklanmaktadır.

2.6.1. Danışmanlı Öğrenme (Supervised Learning)

Bu tip öğrenmede, yapay sinir ağlarına örnek olarak bir doğru çıktı verilir. İstenilen ve gerçek çıktı arasındaki farka (hataya) göre nöronlar arasındaki bağlantıların ağırlıklarını en uygun çıktıyı elde etmek için sonradan düzenlenebilir. Bu sebeple danışmanlı öğrenme algoritmasının bir "danışmana (öğretmene)" ihtiyacı vardır. Widrow-Hoff tarafından geliştirilen delta kuralı ve Rumelhart ve McClelland tarafından geliştirilen genelleştirilmiş delta kuralı ve ya geriye beslemeli öğrenme algoritması danışmanlı öğrenme algoritmalarına örnek olarak verilebilir.

En basit yapay sinir ağları, verilen bir kategoriye ait olan ya da olmayan bir örneği sınıflamak için tasarlanan ağlardır. Bu tip ağlarda çıkış değeri genelde iki değerlidir (1 ya da –1 gibi). Bu çıkış değerleri verilen kategoriye ait ise 1, değilse –1 değerini alır. Bir sonraki bölümde, örnek sınıflaması için tasarlanmış bir kaç basit tek katmanlı ağ incelecektir. Bu ağlar kontrollü öğrenme algoritmasını kullanmaktadırlar. Basit sınıflandırma problemleri için bu algoritma ve ağ mimarisi yeterlidir ama daha karmaşık olanlar için çok katmanlı ağ mimarisi ve geri beslemeli öğrenme tekniklerinin kullanılması gerekmektedir.

Örnek ilişkilendirme, haritalama problemlerinin diğer bir biçimidir. Sadece evet veya hayır çıktısı beklenmektedir. Bir yapay sinir ağı, girdi değerlerine uygun olan ve *ilişkili hafıza* (*associative memory*) olarakda adlandırılan çıktı değerlerini vermesi için eğitilir. Eğer istenilen çıktı vektörü girdi vektörüyle aynıysa, yapay sinir ağı, *otomatik ilişkili hafıza* (auto*associative memory*) olarak adlandırılır. Eğer çıktı vektörü girdi vektöründen farklı ise, *farklı türden ilişkili hafıza* (*heteroassociative memory*) olarak adlandırılır. Öğrenme sonunda, ilişkili hafıza daha önceden öğrendiği bir girdi değerini gördüğünde, bu girdiye uygun

olan örneği hemen anımsayabilir. İlişkili hafızalar ilerideki bölümlerde bahsedilecek olan hem ileri beslemeli hem de geri beslemeli ağ yapısındadır.

2.6.2. Danışmansız Öğrenme (Unsupervised Learning)

Danışmansız öğrenmede ağ, girdi olarak verilen örnekten elde edilen çıktı bilgisine göre sınıflandırmayı kendi kendine geliştirmektedir. Bu öğrenme algoritmalarında, istenilen çıktı değerinin bilinmesine gerek yoktur. Öğrenme sürecinde sadece giriş bilgileri verilir. Ağ daha sonra bağlantı ağırlıklarını aynı özellikleri gösteren desenler oluşturmak üzere ayarlar. Kohonen'in kendiliğinden organize olabilen haritaları (Kohonen self-organizing maps) ve adaptif rezonans teorisi (adaptive resonance theory), danışmansız öğrenmeye örnek olarak verilebilir. [10]

2.7. Yapay Sinir Ağlarının Gelişim Tarihçesi

Geliştirilmiş olan eski yapay sinir ağı mimarileri ve algoritmalar günümüzde hala kullanılmaktadır ve bu alandaki gelişmeler öncekilerle önemli derecede ilişkilidir. Bu yüzden yapay sinir ağlarının gelişmesinde önemli rolü olan bilim adamları ve araştırmaları hakkında kısa bir bilgi aşağıda verilmiştir:

- **1940'lı Yıllar : Yapay Sinir Ağı Çalışmalarının Başlangıcı**

McCulloch-Pitts Nöronları : Warren McCulloch ve Walter Pitts ilk sinir ağlarıyla ilgilendiklerinde, bu ağların ne çeşit uygulamaları yerine getireceğini tasarladılar. Ağa basit nöronlar eklediklerinde, ağın hesaplama gücünün arttığını gördüler. McCulloch-Pitts ağırlık değerlerini kullanarak, basit mantık fonksiyonlarını sinir ağına sundular. Bu modelde farklı nöronlar farklı fonksiyonları gerçekleştirir. Herhangi bir çıkışı üretmek için, sinir ağına yerleştirilen nöronlar mantıksal fonksiyonları birleştirerek yapılandırılabilir. Ağ üzerinde bir nörondan diğerine bilgi akışı zamanı, yani sinyalin geçişi, bir birim zamanı gibi kabul edilir. Zaman gecikmesi, bazı psikolojik işlemlerin modellemesini sağlar. Sıcak ve soğuğun algılanması buna örnek olabilir.

Sinir ağına giren değer eşik değerinden büyükse, nöron aktif hale geçer, aksi takdirde ise aktif hale geçmez. Bu fikir McCulloch-Pitts nöronlarının bir özelliği gibi günümüzde de aynı mantıkla kullanılmaktadır.

McCulloch ve Pitts'in çalışmaları önemini hale korumaktadır ve bu çalışmalar, çevirme ve döndürme, karakter tanıma gibi alanlarda oldukça yaygın olarak kullanılmaktadır.

Hebb Öğrenmesi : Donald Hebb, McGill üniversitesinde bir psikologdur ve yapay sinir ağları için ilk öğrenme yasasını 1949 yılında tasarlamıştır. Hebb'e göre, eğer iki nöron eş zamanlı olarak aktif oluyorsa, bu iki nöron arasındaki bağlantının gücünün artırılması gerekir. Bu genel ifade bilgisayar simülasyonunun yapılmasını mümkün kılmıştır. Kohonen'in ve Anderson'un korelasyon matrisi fikri yukarıdaki fikirle yakından ilişkilidir.

- **1950 ve 1960'lı Yıllar : Yapay Sinir Ağların İlk Gelişim Dönemi**

Günümüzde yapay sinir ağı yaklaşımı geleneksel hesaplama yöntemlerine bir alternatif gibi görünse de, bu alan "modern hesaplamanın babası" olan John Von Neuman'ın ilgisini büyük ölçüde çekmiştir. Neuman beynin modellemesi üzerinde hızlı şekilde çalışmıştır. Johnson ile Brown (1988) ve Anderson ile Rosenfeld (1988) ilk yapay sinir ağları araştırmacılarından Warren McCulloch ve Von Neumann arasındaki etkileşimi tartışmışlardır ve Von Neuman'ın gelecekte ne tür bilgisayarlar geliştireceği görüşü üzerinde mutabakat sağlamışlardır.

Perseptronlar : Block, Minsky ve Papert (1958) ve Frank Rosenblatt (1962), birkaç araştırmacı ile birlikte, perseptronlar adı verilen çok geniş bir yapay sinir ağı sınıfını tanıtmış ve geliştirmişlerdir. En genel perseptron, diğer nöronlarla ağırlıklı bir bağlantısı bulunan girdi biriminden oluşmaktaydı ve bu bağlantılar arasındaki ağırlıklar ayarlanabiliyordu. Perseptron öğrenme kuralı, Hebb öğrenmesinden daha güçlü olan tekrarlı ağırlık ayarlama yöntemini kullanır. Perseptron öğrenme kuralı problemi çözmek için gerekli olan ağırlıkları bulabilir. Rosenblatt'ın 1962'de yaptığı çalışmada perseptronların birçok tipini tanımlamıştır. McCulloch, Pitts ve Hebb tarafından geliştirilen nöronlardaki gibi perseptronlar da eşik çıkış fonksiyonunu kullanırlar.

İlk elde edilen başarılar sonucunda, bu konuyla daha çok kişi ilgilenmeye başlamıştır. Fakat uygun koşullar altında tekrarlı öğrenme yaklaşımının matematiksel ispatı, perseptron tipli ağların neyi öğreneceğini dikkate aldıktan sonra getirilen sınırlamaların tespit edilmesi sonucunda kanıtlanmıştır.

ADALINE : Bernard Widrow ve onun öğrencisi olan Ted Hoff, perseptron öğrenme kuralıyla çok yakın ilişkisi olan bir öğrenme kuralı geliştirdiler. Perseptron kuralı, sinirsel birimin cevabının yanlış olduğu durumlarda, bağlantı ağırlıklarını ayarlar ve bu çalışma, giriş-çıkış numunesinin sınıflandırılmasını göstermektedir. Delta kuralı, çıktı birimi ile ağ girdisi arasındaki farkı azaltmak için ağırlıkları değiştirir. Bu da, en küçük karesel hataya yol açar. Psikolog olan Rosenblatt'ın ve elektrik mühendisi olan Widrow ve Hoff'un geliştirmiş oldukları modellerin benzerliği, sinir ağlarının doğasının disiplinler arası olduğunun bir kanıtıdır. Öğrenme kuralındaki farklılık önemsiz olmasına rağmen, ağa genelleştirme yeteneği kazandırmıştır. Tek katmanlı bir ağ için kullanılan Widrow-Hoff Öğrenme Kuralı, çok katmanlı ağlar için geri beslemenin habercisi olmuştur.

Widrow ve öğrencisinin çalışması, bazen bir yapay sinir ağı çalışması gibi bazen de adaptif doğrusal sistemlerin araştırılması çalışması olarak literatüre geçmiştir. ADALINE ismi, Adaptive Linear Neuron or Adaptive Linear Systems in yorumlanması sonucunda oluşmuştur. Adapte olabilen anten sistemleri için sinir ağlarından ADALINE'nın bir çok ilginç uygulaması mevcuttur. Kamyonun yüklenmesi, boşaltılması, süpürge dengelenmesi gibi kontrol problemleri, bunlara örnek olarak verilebilir. MADALINE'lar, ADALINE'ların çok katmanlı ağlara yaygınlaştırılmış halidir.

- **1970'li Yıllar : Yapay Sinir Ağların Durgun Yılları**

Minsky ve Papert'in perseptronun eksikliklerini göstermelerine rağmen yapay sinir ağları üzerindeki araştırmalar devam etmiştir. Güncel uzman liderlerden bir çoğu 1970'lerde çalışmalarını yayımlamaya başlamışlardır.

Kohonen : Teuvo Kohonen,1972 yılında yapmış olduğu ilk çalışmasında, birleştirici hafızalı sinir ağlarıyla ilgilenmiştir. On yıl sonraki çalışması,

kendiliğinden organize olabilen nitelik haritalarının gelişimiydi ve bu haritalar, kümelenmiş birimler için topolojik bir yapıyı kullanmıştır. Bu ağlar Fince ve Japonca kelimeler için konuşma tanımada ve müzikal bestelerde uygulanmıştır.

Anderson : James Anderson, birleştirici hafızalı ağlar üzerinde çalışmaya başlamıştır. 'Kutudaki Beyin Durumu' fikrini geliştirmiştir. Daha kararlı bir çıktı almak için, önceki modellerdeki doğrusal çıktıyı kuvvetlendirmiştir. Bu tür uygulamalara örnek olarak, tıbbi teşhis ve çarpım tablosunun öğrenilmesi verilebilir.

Grossberg : Stephen Grossberg'in, birçok iş arkadaşı ile birlikte inanılmaz derecede verimli ve üretken bir kişiliği vardır. Klimasauskas 1989 yılında, Grossberg' in 1967 ile 1988 arasında yaptığı 146 yayını listelemiştir. Grossberg'in matematiksel ve biyolojik çalışmaları çok geniş bir çevre tarafından bilinmektedir.

Carpenter : Stephen Grossberg ile birlikte Gail Carpenter, Adaptive Resonance Theory olarak adlandırılan ve kendiliğinden organize olabilen yapay sinir ağları modelini geliştirdiler. Adaptif rezonansa sahip yapay sinir ağı, ikili giriş örnekleri ve sürekli giriş değerleri için kullanılmaktadır. [11]

- **1980'li Yıllar : Yapay Sinir Ağlarına Yenilenen İlgi**

Geriye Yayılım (Backpropagation) : 1970'lerdeki durgun yılların iki sebebi vardı. Basit sinir ağları, XOR mantık fonksiyonu olarak, basit haritalama problemlerinin çözümünü bulmada başarısızlığa uğruyordu ve çok katmanlı bir ağın eğitimi için genel bir yöntem yoktu. 1960'larda, çıktı birimlerindeki hatalar hakkında geri yayılım bilgisinin gizli birimlere gönderilmesi duyulmuştu, fakat bu fazla bilinmiyordu. Bu yöntem, David Parker ve LeCun tarafından, birbirlerinden bağımsız bir biçimde bulunmuştu. Bu optimal kontrol teorisinde daha önce kullanılan algoritmaya çok benzerdir. Parker'ın çalışması, Paralel Dağıtılmış İşlem Gurubu üyelerinden San Diego'daki California üniversitesinde psikolog olarak çalışan David Rumelhart ve Carnegie-Mellon üniversitesinden James McClelland'ın ilgisini çekmiştir ve onlar bu araştırmayı inceleyerek yayınlamışlardır.

Hopfield Ağları : Sinir ağlarının gelişiminde katkısı olan önemli bilim adamlarından birisi de California Teknoloji Enstitüsü'nden fizikçi John Hopfield'dir. David Tank ile birlikte Hopfield, uyarlanabilir aktivasyonlar ve sabit ağırlıklara dayalı birkaç yapay sinir ağı modelini geliştirmişlerdir. Bu ağlar birleştirici hafızaya sahip ağlar olarak hizmet edebilir ve seyyar satıcı problemi gibi kısıtlandırılmış yeterlilik problemlerinde kullanılabilir. Amerikalı bilim adamları Tank ve Hopfield'ın yayınları, yapay sinir ağlarına olan ilginin oldukça artmasına neden olmuştur ve onlara Nobel ödülünü kazandırmıştır. Söz konusu yayınlar, insanların yaptığı işleri yapabilen makinelerin üretilmesi ile ilgiliydi.

Yeni Biliş Ağları : Kunihiko Fukushima ve çalışma arkadaşları, karakter tanıma ile ilgili uzmanlaştırılmış birkaç yapay sinir ağı mimarisi geliştirmişlerdir. Bu yapay sinir ağlarına bir örnek olarak neocognitrion verilebilir. Cognitron olarak adlandırılan daha önceki kendiliğinden organize olabilen yapay sinir ağları, karakter pozisyonunu ve karakterlerin rotasyon bozukluklarını tanımada başarısızdı. Ancak bu eksiklik neocognitron yöntemiyle düzeltildi.

Boltzmann Makinesi : Birkaç araştırmacı deterministik olmayan sinir ağlarını incelemek istiyordu. Bu tip yapay sinir ağlarında, aktivasyonların ve ağırlıkların değişmesi olasılık yoğunluk fonksiyonu temelinde yapılmaktadır. Söz konusu ağlar, Bayes'in karar teorisi ve benzetişim tavlama gibi klasik fikirleri de içermektedir.

Donanım Gerçekleştirmesi : Yapay sinir ağları üzerindeki ilginin artmasının başka bir nedeni de hesaplama yeteneği güçlü olan donanımların ortaya çıkmasıdır. Yapay sinir ağlarıyla, optik sinir ağları uygulamaları geliştirilmektedir.

California Teknoloji Enstitüsünden Carver Mead, mikroçiplerin tasarımı için üretilen yazılımın bulucusu olmakla birlikte, sinirsel devre çalışmalarının da lideridir.

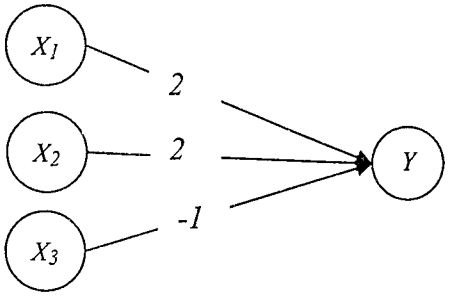
Brown üniversitesinden Nobel ödüllü Leon Cooper, çok katmanlı ilk ağlardan biri olan azaltılmış coulomb enerji ağı (reduced coulomb energy network)'nı tanıtmıştır. [9,10]

2.8. McCulloch-Pitts Nöronları

İlk yapay sinir ağ modeli olan McCulloch-Pitts nöronları, 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.

McCulloch-Pitts ağlarının aktivasyonları ikilidir. Herhangi bir zaman anında nöron ya ateşler ve "1" aktivasyonuna sahip olur, ya da ateşlemez ve "0" aktivasyonuna sahip olur. Bu nöronlar, direkt olarak ağırlıklandırılmış yollarla biribirine bağlıdır. Bu yollar üzerindeki ağırlıklar pozitifse bağlantı yolu hızlandırıcı, değilse yavaşlatıcıdır. Belirli bir nörondaki tüm hızlandırıcı bağlantılar aynı ağırlıklara sahiptirler. Ayrıca her bir nöron sabit eşik değerine sahiptir. Nöronun ağ girişi, bu eşik değerinden büyük ise nöron aktif hale geçer. Bununla birlikte yavaşlatma mutlak hale gelsin diye eşik değeri ayarlanır, başka bir ifadeyle "0" olmayan herhangi bir yavaşlatıcı giriş nöronun ateşlenmesini önleyecektir.

McCulloch-Pitts ağlarına basit bir örnek şekil 2.11'de gösterilmiştir. Bu ağda, X_1 ve X_2'den Y'ye giden bağlantılar hızlandırıcıdır. Bu hızlandırıcılar aynı birime doğru gittiği için aynı pozitif ağırlık değerine sahiptirler. Y için eşik değeri 4'tür. Sadece yavaşlatıcı bağlantı sinyali "0" değerini aldığında Y aktif hale geçer, diğer hallerde Y aktif hale geçmez. X birimlerinde Y'ye her sinyalin geçişi bir zaman adımı kadar sürmektedir. Y'nin t zamanındaki aktivasyonu X_1, X_2 ve X_3'ün, $t-1$ zamanındaki aktivasyonlarıyla belirlenir. [5,11]

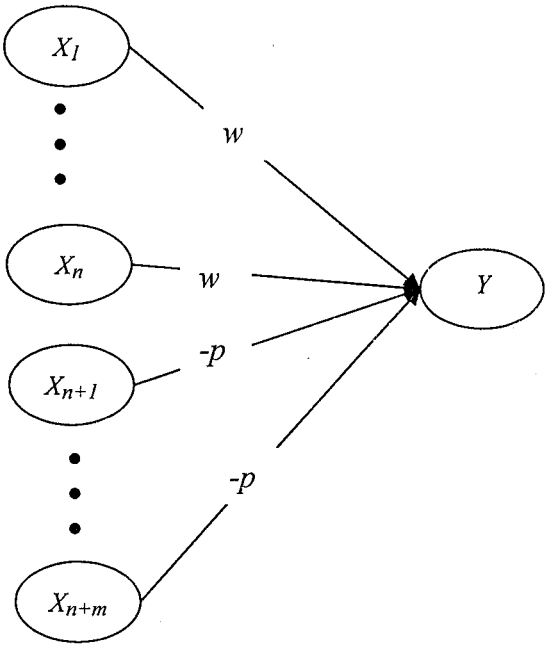


Şekil 2.11. Basit bir McCulloch-Pitts nöronu

2.8.1. McCulloch-Pitts Nöronlarının Mimarisi

Bir McCulloch-Pitts Y nöronu genel olarak, birkaç sayıdaki X nöronlarından sinyal alır. Her bir bağlantı yolu ya hızlandırıcı başka bir ifadeyle ağırlığı $w > 0$ ya da yavaşlatıcı başka bir ifadeyle ağırlığı $-p$ ($p > 0$)'dır.

Şekil 2.12'de McCulloch-Pitts nöronlarının genel bir mimarisi gösterilmiştir:



Şekil 2.12. McCulloch-Pitts Y nöronunun mimarisi

Şekil 2.12.'de gösterilen ağda, $X_1,...,X_n$ birimleri hızlandırıcı sinyalleri, $X_{n+1},...,X_{n+m}$ birimleri ise yavaşlatıcı sinyalleri Y nöronuna iletir. Y için aktivasyon fonksiyonu aşağıdaki şekildedir:

$$f(y\_in) = \begin{cases} 0 & , \ y\_in < \theta \quad \text{ise} \\ 1 & , \ y\_in \geq \theta \quad \text{ise} \end{cases} \tag{2.10}$$

y_in, Y birimine giren toplam giriş sinyalleri, θ ise eşik değeridir. Yavaşlatmanın olması için $\theta > nw - p$, en az k tane hızlandırıcı giriş alındığında aktif hale geçmesi için $kw \geq \theta > (k-1)w$ eşitsizliklerinin sağlanması gerekir.

Belirli bir birim içindeki tüm hızlandırıcıların ağırlıkları aynı olmasına rağmen farklı iki birimin hızlandırıcıları farklı olabilir.

Bu basit nöronlar kullanılarak herhangi bir mantıksal fonksiyon modellenebilir. Basit mantıksal fonksiyonları gerçekleştirebilmek için, nöronun bağlantı ağırlıkları, aktivasyon fonksiyonunda kullanılan eşik değeri ile birlikte ayarlanır. [11]

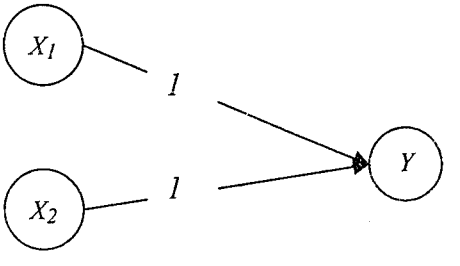
Örnek 2.1.

VE mantıksal fonksiyonu girdi değerlerinin her ikisi doğru ise doğru, aksi takdirde yanlış cevabını verir. Doğru "1" ve yanlış "0" ile gösterilirse, dört eğitim girişi ve bunlara uygun hedef çıkış değerleri çizelge 2.1.'deki gibi olacaktır:

Çizelge 2.1. Dört eğitim girişi ve bunlara uygun *VE* fonksiyonu için girdi ve hedef değerleri

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Şekil 2.13.'te *VE* mantıksal fonksiyonunun gerçekleştiren bir McCulloch-Pitts nöronu gösterilmiştir. Burada *Y* birimi üzerindeki eşik değeri 2'dir. Bu nöron şu şekilde çalışmaktadır: X_1, X_2 girdi birimlerinden dahil olan sinyaller *Y* nöronuna bağlandıkları linkler üzerindeki ağırlıklarla çarpılır ve toplamı "2" eşik değeri ile karşılaştırılır. *Y* nöronuna giren sinyal, 2'den küçük ise çıktı sinyali "0", 2'ye eşit ve ya büyükse "1" olur. Bu şekilde *VE* fonksiyonunu gerçekleştirir.



Şekil 2.13. *VE* fonksiyonunu gerçekleştiren bir McCulloch-Pitts nöronu

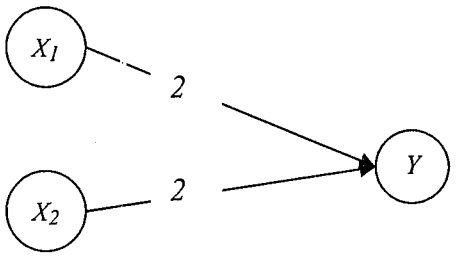
Örnek 2.2.

VEYA mantıksal fonksiyonu girdi değerlerinden herhangi biri doğru olduğunda doğru cevabını, girdilerden her ikisi de yanlış olduğunda yanlış cevabını verir. Doğru "1" ve yanlış "0" ile gösterilirse, dört eğitim girişi ve bunlara uygun hedef çıkış değerleri çizelge 2.2.'deki gibi olacaktır:

Çizelge 2.2. Dört eğitim girişi ve bunlara uygun *VEYA* fonksiyonu için girdi ve hedef değerleri

x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

Şekil 2.14.'te *VEYA* mantıksal fonksiyonunun gerçekleştiren bir McCulloch-Pitts nöronu gösterilmiştir. Burada Y birimi üzerindeki eşik değeri 2'dir. Bu nöronun çalışması da örnek 2.1.'deki AND fonksiyonu için izlenen yöntemle aynıdır.



Şekil 2.14. *VEYA* fonksiyonunu gerçekleştiren bir McCulloch-Pitts nöronu

2.9. Tek Katmanlı Yapay Sinir Ağ Modelleri

Tek katmanlı yapay sinir ağları örnek sınıflandırma, tanıma, örnek ilişkilendirme ve bunun gibi diğer problemlerin çözülmesinde kullanılabilir.

Örnek sınıflandırma problemlerinde, her giriş vektörü (örnek, numune) belirli sınıflara ait olabilir ya da olmayabilir. Basit olarak, bir sınıfa üye olma sorusu göz önünde bulundurulur. Çıkış birimi için +1 cevabının alınmasıyla örneğin o sınıfa üye olduğu, -1 cevabı alınırsa, örneğin o sınıfa üye olmadığı belirlenir. Bu tip durumlarda, her bir sınıf için bir çıkış birimi vardır. Örnek

tanımlama, örnek tanıma uygulamasının bir çeşididir. Örneklerin ilişkili hatırlanması ise daha farklıdır.

Tek katmanlı sinir ağlarının eğitilmesinde üç önemli yöntem vardır:

- Hebb Kuralı
- Perseptron Öğrenme Kuralı
- Delta Kuralı

Gerçek dünyada karşılaşılan birçok problem, daha karmaşık mimarileri ve karmaşık eğitim kurallarını gerektirir ve genel olarak, tek katmanlı yapay sinir ağları bu tip problemleri çözmede yeterli değildir. Ancak şartlar bu ağları kullanmak için elverişli ise, doğru sonuçlar alınabilmesi mümkündür. [5,9]

2.9.1. Hebb Kuralı

Hebb kuralı, bir yapay sinir ağı için, en eski ve en basit öğrenme kuralı olarak bilinir. Hebb, öğrenmenin, sinaps uzunluklarını (ağırlıkları) değiştirerek meydana geleceğini önermiştir. Hebb'e göre, eğer birbiri ile bağlı iki nöronun her ikisi de aynı zamanda "aktif" ise, bu nöronlara uygun ağırlıkların artırılması gerekmektedir. Benzer olarak, eğer her iki nöron aynı zamanda "pasif" ise, ağırlıkların artırılması gerekir. Bu durumda, daha güçlü bir öğrenme şekli meydana gelir. Hebb kuralı, geliştirilmiş hesaplama yeteneğinden dolayı sıkça kullanılır. Geliştirilmiş Hebb kuralı ile tek katmanlı ileri beslemeli bir sinir ağının eğitilmesi bir Hebb ağını anlatır. Hebb kuralı, diğer özgül ağları eğitmek için de kullanılabilir. Tek katmanlı bir yapay sinir ağında birbiri ile bağlantılı nöronlardan bir tanesi giriş birimi, bir tanesi de çıkış birimi olacaktır (hiçbir giriş birimi birbiri ile bağlanmadığı için, herhangi bir çıkış birimleri de birbiri ile bağlı değildir).

2.9.1.1. Hebb Eğitim Algoritması

Örnek sınıflandırmada bir çıktı biriminin olması yeterlidir. Ancak genel olarak bir çok problemlerde, örneğin ilişkilendirme problemlerinde birden fazla çıktı birimlerinin olması durumu ortaya çıkmaktadır. Hebb ağında kullanılan tek katmanlı ağ n sayıda girdi biriminin m sayıda ise çıktı biriminin olduğunu varsayıldığında ağ, ikili veya kutupsal vektörler olarak temsil edilebilen

numuneler için kullanılabilir. Herhangi i girdi birimi ile j çıktı birimine uygun ağırlığı w_{ij}, $i = 1,...,n; j = 1,...,m$ olduğu durumda, Hebb eğitim algoritmasını aşağıdaki gibi yazılabilir:

Adım 0 Ağırlıkların ilk değerlerini ata:

$$w_{ij} = 0,\ i = 1,...,n; j = 1,...,m$$

Adım 1 Her bir s, t hedef çıkış çifti için, 2-4 adımlarını uygula.:

Adım 2 Eğitim girdileri için aktivasyonları ata:

$$x_i = s_i \quad i = 1,...,n$$

Adım 3 Çıkış birimlerine hedef aktivasyonları ata:

$$y_j = t_j \quad j = 1,...,m$$

Adım 4 Yeni ağırlıkları ayarla:

$$w_{ij}(yeni) = w_{ij}(eski) + x_i y_j \qquad i = 1,...,n; j = 1,...,m$$

Sapmayı ayarla:

$$b_j(yeni) = b_j(eski) + y_j$$

Burada, sapma, herhangi bir "birim" den gelen bir ağırlık gibi ayarlanır. Bu birimin çıkış sinyali daima 1'dir.

Ağırlık güncellemesi aynı zamanda aşağıdaki vektörel formda da verilebilir:

$$\mathbf{w}\ (\text{yeni}) = \mathbf{w}\ (\text{eski}) + \mathbf{x}y \tag{2.11}$$

Ağırlık değişimi terimi $\Delta\mathbf{w}$ kullanılarak şu şekilde yazılabilir:

$$\Delta\mathbf{w} = \mathbf{x}y \tag{2.12}$$

Bu durumda,

$$\mathbf{w}\ (\text{yeni}) = \mathbf{w}\ (\text{eski}) + \Delta\ \mathbf{w} \tag{2.13}$$

olur. Ayrıca, sınıflandırma problemlerinde çıktı katmanında bir sayıda birimin olması yeterlidir, başka bir ifadeyle $m = 1$ kabul edilebilir.

Öğrenme için Hebb kuralını uygulamanın birkaç yöntemi vardır. Yukarıdaki algoritma, eğitim grubu boyunca sadece bir tane yola gerek duymaktadır. [5,32]

2.9.1.2. Hebb Ağının Uygulaması

Eğitim örneklerinin seçimi, hangi problemlerin Hebb kuralı ile çözüleceğinin kararında önemli rol oynamaktadır. *VE* fonksiyonunun girişlerinin ikili yada iki kutuplu olmasına bağımlı olarak Hebb kuralı farklı sonuçlar verir. Girişler ve hedef değerleri iki kutuplu olduğunda hem girdi biriminin hem de çıkış biriminin aynı anda aktif oldukları ya da aynı anda pasif oldukları durumlarda Hebb algoritması ağırlıkların değiştirilmesine imkan verir.

Örnek 2.3.

VE mantık fonksiyonu için Hebb kuralı aşağıdaki şekilde uygulanır:

VE mantık fonksiyonu iki kutuplu girdiler ve bipolar hedefler için çizelge 2.3'deki gibi ifade edilebilir:

Çizelge 2.3. *VE* fonksiyonu için iki kutuplu girdiler ve hedefler

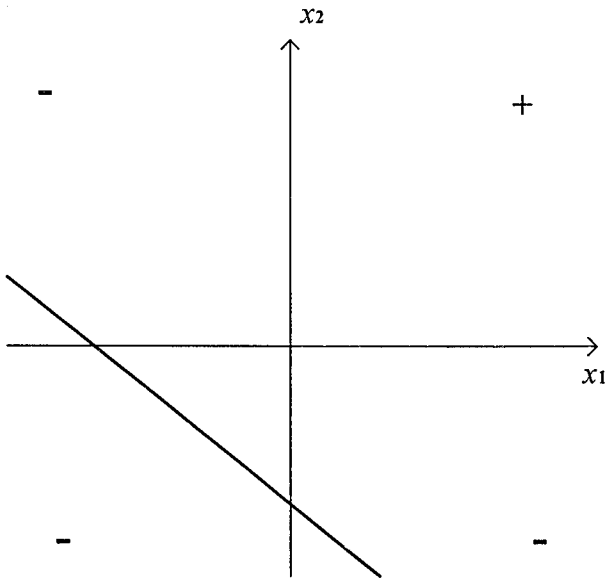
GİRİŞ			HEDEF
x_1	x_2	*1*	
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Verilen mantık fonksiyonuna uygun olan tek katmanlı ağda iki girdi birimi, bir çıktı birimi ve bir sapma olacaktır. Bu yapay sinir ağında Hebb algoritmasını kullanarak, ilk giriş için elde edilen ağırlıklar çizelge 2.4.'de verilmiştir:

Çizelge 2.4. İlk giriş sonucunda elde edilen ağırlıklar

GİRİŞ			HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	*1*		Δw_1	Δw_2	Δb	w_1	w_2	b
							0	0	0
1	1	1	1	1	1	1	1	1	1

Bu tabloya uygun ayırma doğrusu $x_2 = -x_1 - 1$ doğrusudur ve bu doğru şekil 2.11'de gösterilmiştir. Bu grafik, ilk giriş değeri için ağın doğru cevabı verebileceğini göstermektedir.



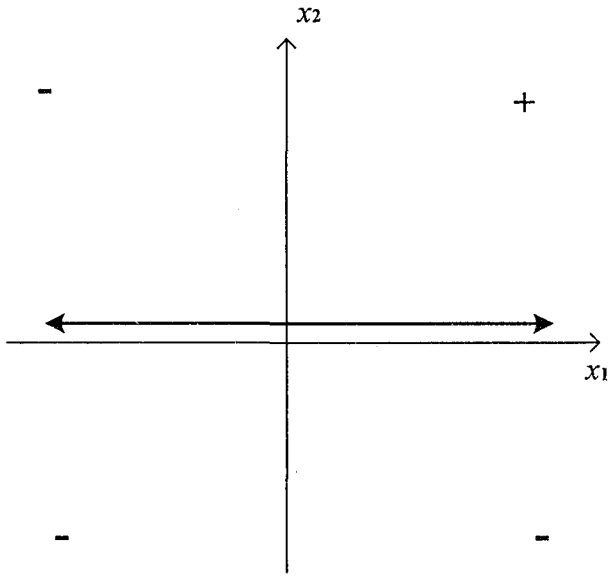
Şekil 2.15. Hebb kuralını kullanan *VE* fonksiyonu için iki kutuplu giriş ve hedeflerde ilk giriş çiftinden sonraki karar sınırı

İkinci giriş vektörü ve hedef değeri için Hebb algoritması çizelge 2.5.'deki sonuçları vermektedir:

Çizelge 2.5. İkinci giriş sonucunda elde edilen ağırlıklar

GİRİŞ			HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	1		Δw_1	Δw_2	Δb	w_1	w_2	b
							1	1	1
1	-1	1	-1	-1	1	-1	0	2	0

Bu tabloya uygun ayırma doğrusu $x_2 = 0$ doğrudur ve bu doğru şekil 2.16'da gösterilmiştir:



Şekil 2.16. Hebb kuralını kullanan *VE* fonksiyonu için iki kutuplu giriş ve hedeflerde ikinci giriş çiftinden sonraki karar sınırı

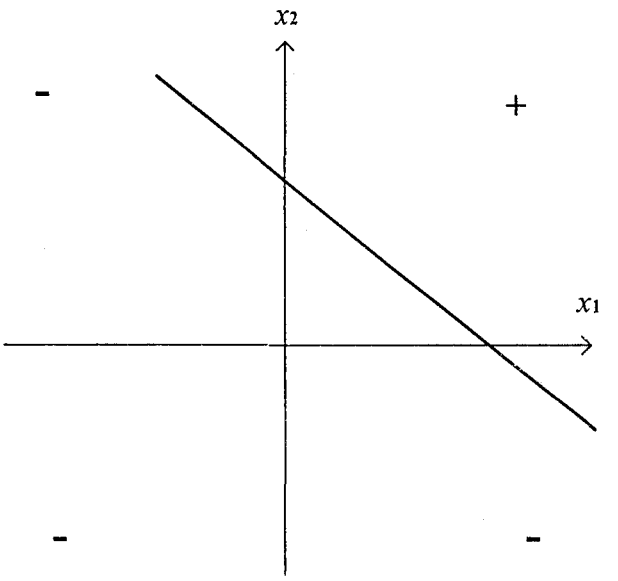
Şekil 2.16.'deki grafik, (0,2,0) ağırlığında ağın ilk (1,1) ve (1,-1) girişleri için doğru cevabı vereceğini göstermektedir.

Üçüncü giriş vektörü ve hedef değeri için Hebb algoritması çizelge 2.6.'deki sonuçları vermektedir:

Çizelge 2.6. Üçüncü giriş sonucunda elde edilen ağırlıklar

GİRİŞ			HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	1		Δw_1	Δw_2	Δb	w_1	w_2	b
							0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1

Bu tabloya uygun ayırma doğrusu $x_2 = -x_1 + 1$ doğrusudur ve bu doğru şekil 2.17'de gösterilmiştir.



Şekil 2.17. Hebb kuralını kullanan *VE* fonksiyonu için iki kutuplu giriş ve hedeflerde üçüncü giriş çiftinden sonraki karar sınırı

Şekil 2.17.'deki grafik, ilk üç girdi değeri için ağın doğru cevabı üreteceğini göstermektedir.

Son giriş vektörü ve hedef değeri için Hebb algoritması çizelge 2.7.'deki sonuçları vermektedir:

Çizelge 2.7. Son giriş sonucunda elde edilen ağırlıklar

GİRİŞ			HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	1		Δw_1	Δw_2	Δb	w_1	w_2	b
							1	1	-1
-1	1	1	-1	1	-1	-1	2	2	-2

Bu tabloya uygun ayırma doğrusu $x_2 = -x_1 + 1$ doğrusudur. Ağırlıkların değişmesine rağmen ayırma çizgisi bir öncekiyle aynıdır. Böylece karar bölgesinin grafiği Şekil 2.17'deki gibidir.

VE mantıksal fonksiyonu için giriş ve hedef ikili (0,1) değerler olarak kabul edilirse Hebb algoritması olumlu bir sonuç vermemektedir. Bu durum, eğitim algoritmalarında veri tiplerinin önemini göstermektedir.

Hebb kuralını kullanan basit örneklerden bir tanesi de karakter tanımadır. Verilen bir "X" ile "O" karakterini birbirinden ayırt eden bir ağ aşağıdaki örnekte verilmiştir.

2.9.2. Perseptron

Perseptronlar, yapay sinir ağlarının öğrenebilir niteliğini taşıyan ilk modelidir. Hebb kuralından daha yetenekli bir öğrenme kuralıdır. Perseptron tekrarlı öğrenme algoritmasıdır ve çözümün varlığı durumunda yakınsama niteliğine sahiptir. Bu, perseptron modelinin en önemli niteliklerinden biridir.

Rosenblatt (1962) ve Minsky-Papert (1969, 1988) tarafından çeşitli perseptron modelleri tanımlanmıştır. Orijinal perseptronlar, duyumsal birimler, birleştirici birimler ve cevap birimleri olmak üzere nöronların üç durumuna sahiptirler. Örneğin, bir basit perseptron duyumsal ve birleştirici birimler için ikili aktivasyon, cevap birimi için ise +1, 0, veya -1 değerlerini üreten aktivasyon uygulayabilir.

Sınıflandırma problemlerinde eşik değerli aktivasyon fonksiyonu kullanılır:

$$f(y\_in) = \begin{cases} -1, & y\_in < -\theta \text{ ise} \\ 0, & -\theta \le y\_in \le \theta \text{ ise} \\ 1, & y\_in > \theta \text{ ise} \end{cases} \qquad (2.14)$$

Çıktı biriminin aktivasyonu $y = f(y\_in)$ şeklinde hesaplanır.

Birleştirici birimden cevap birimine giden bağlantıların ağırlıkları perseptron öğrenme kuralı ile ayarlanır. Her eğitim girişi için, sinir ağı, çıkış biriminin cevabını hesaplar. Daha sonra sinir ağı, bu örnek için çıkış değeri ile hedeflenen çıkış arasındaki farkı karşılaştırarak bir hata oluşup oluşmadığını tespit eder. Yapay sinir ağı, hesaplanmış çıkış değeri "0" ve hedef değeri "-1" olan örnek için hatayı ayırt edemez, buna karşıt olarak hesaplanmış çıkış değeri "+1" ve hedef değeri "-1" olan örnek için hatayı ayırt edebilir. Bu durumlarda, hedef verinin işareti yönünde ağırlıkların işareti değiştirilmelidir. Bununla birlikte çıkış birimine "0" olmayan sinyaller gönderen bağlantıların ağırlıkları ayarlanmalıdır. Eğer belirli bir eğitim giriş örneğinde hata oluşuyorsa, ağırlıklar

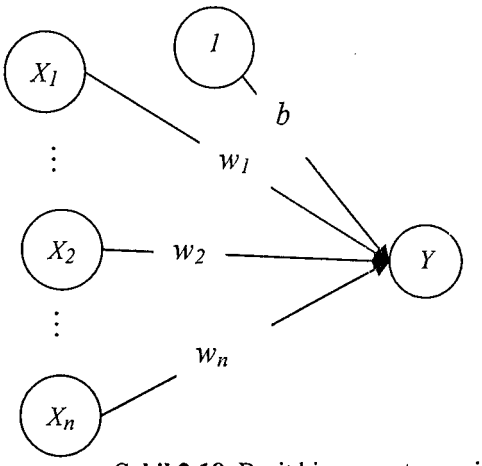
$$w_i(yeni) = w_i(eski) + \alpha t x_i \qquad (2.15)$$

formülüne göre değiştirilmelidir.

Burada hedef değeri t ya "+1" ya da "-1"dir ve α öğrenme oranı katsayısıdır. Eğer hata oluşmadıysa ağırlıklar değiştirilmemelidir. Eğitim işlemi hata oluşmayıncaya kadar devam etmelidir. Perseptron öğrenme kuralı yakınsama teoremine göre, eğer ağda tüm eğitim örnekleri için uygun ağırlıkların varlığına izin veriiirse, bu ağırlıklar, eğitim sürecinde bütün eğitim örnekleri için elde edilebilir. Bu kuralın amacı, ağın tam olarak doğru cevap veremediği eğitim örnekleri için ağırlıkları ayarlamaktır. Ayrıca, eğitim sonunda bu ağ sınırsız sayıdaki eğitim adımları için ağırlıkların değerlerini bulmalıdır. [5,10]

2.9.2.1. Perseptronun Mimarisi

Şekil 2.18.'de perseptronun mimarisi gösterilmiştir. Burada $X_1,...,X_n$ girdi birimleri, Y çıktı birimi ve 1 sapma sinyalidir. b sapma ağırlığı, w_i (i = 1,...,n) ağırlıklardır.



Şekil 2.18. Basit bir perseptron mimarisi

2.9.2.2. Sınıflandırma Problemleri için Perseptron Algoritması

Sınıflandırma problemlerinde, sinir ağının görevi tüm giriş örneklerinin belirli bir sınıfa ait olup olmamasını belirlemektir. Sınıfa ait olma çıkışın "+1" değerine, ait olmama ise çıkışın "-1" değerine uygun olmasıyla belirlenir. Sınıflandırma işlemi yapılabilmesi için ağ, tekrarlı bir teknik ile eğitilir. Girdi ve hedefler ikili ve ya iki kutuplu olabilir. θ eşik değeri tüm birimler için değişmezdir. Sapma ve eşik değerinin her ikisinin aynı zamanda kullanılmasına ihtiyaç duyulmaktadır. Bu işlemin algoritması aşağıda verilmiştir. Bu algoritma, ağırlıkların başlangıç değerlerine ve öğrenme oranına tam olarak duyarlı değildir.

Adım 0 Ağırlıklar ve sapmalara başlangıç değerleri ata.

(Ağırlıkları ve sapma değeri kolaylık için "0" olarak alınabilir.)

Öğrenme oranı olan α $(0 < \alpha \leq 1)$'yı ayarla.

(kolaylık için, α, 1'e eşitlenebilir.)

Adım 1 Durma koşulu yanlış iken , adım 2-6' yı uygula.

Adım 2 Her bir **s:t** öğrenme çifti için, 3-5 adımlarını uygula.

Adım 3 Giriş birimlerinin aktivasyonlarını ayarla.

$$x_i = s_i \quad i = 1, \ldots, n$$

Adım 4 Her çıktı birimi için aktivasyonları hesapla.

$$y\_in_j = b_j + \sum_i x_i w_{ij}. \quad j = 1,...,m:$$

$$f(y\_in) = \begin{cases} -1, & y\_in < -\theta \text{ ise} \\ 0, & -\theta \leq y\_in \leq \theta \text{ ise} \\ 1, & y\_in > \theta \text{ ise} \end{cases}$$

Adım 5 Ağırlıkları ve sapmaları ayarla.

eğer $t_j \neq y_j$ ise,

$$b_j(yeni) = b_j(eski) + t_j$$
$$w_{ij}(yeni) = w_{ij}(eski) + t_j x_i.$$

eğer $t_j = y_j$ ise,

$$b_j(yeni) = b_j(eski)$$
$$w_{ij}(yeni) = w_{ij}(eski)$$

Adım 6 Durma koşulu:

Eğer adım 2'de hiç bir ağırlık değişmezse dur; aksi durumda devam et.

Algoritmada çıktı birimlerinin sayısı m = 1 olabilir. Örneğin, mantıksal fonksiyonları gözden geçirirken çıktı biriminin sayısının bir olduğu kabul edilir. Eğitimden sonra, ağ her bir eğitim vektörünü doğru şekilde sınıflandırır.

Sınıflandırma ile ilgili perseptron eğitim algoritmasında, bir ayırma doğrusu yerine, pozitif cevaplar bölgesini sıfır cevaplar bölgesinden ayıran $w_1x_1 + w_2x_2 + b > \theta$ doğrusu ve negatif cevaplar bölgesini sıfır cevaplar bölgesinden ayıran $w_1x_1 + w_2x_2 + b < -\theta$ doğrusu olmak üzere iki ayırma doğrusu vardır. [5]

Örnek 2.4.

Perseptron öğrenme kuralını kullanan ikili girişli ve iki kutup çıkışlı *VE* mantıksal fonksiyonunu göz önüne alındığında, eğitim verisi çizelge 2.8.'deki gibidir:

Çizelge 2.8. *VE* fonksiyonu için ikili girdiler ve iki kutuplu hedefler

GİRİŞ			HEDEF
x_1	x_2	*1*	
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

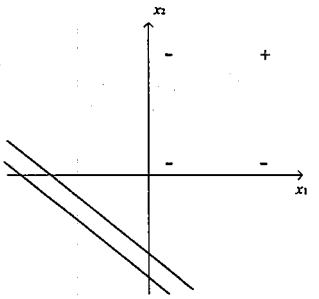
Tek katmanlı bir yapay sinir ağının, problemi çözebilmesi için, gerekli olduğunda, ayarlanabilir bir sapma dahil edilir. İşlem kolaylığı için bu sapma değeri α = 1 olarak alınır ve başlangıç ağırlıkları ve sapma algoritmada belirtildiği gibi "0" olarak ayarlanır. Bununla birlikte, eşik değerinin rolünü göstermek için $\theta = 0{,}2$ olarak alınmıştır.

Şayet hata oluşuyorsa ağırlık değişimi $\Delta w = t(x_1,\ x_2,\ 1)$, aksi takdirde "0" olur. Perseptron algoritmasının aşamaları yerine getirilirse, sonuç 10 aşamada gerçekleşir. Bu hesaplama aşamaları çizelge 2.9.'da gösterilmiştir:

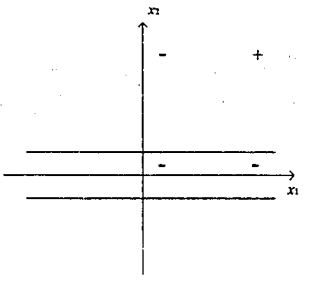
Çizelge 2.9. *VE* fonksiyonu için ikili girdiler ve iki kutuplu hedefler için perseptron uygulaması

GİRİŞ			NET	ÇIKIŞ	HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	1				Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	-1	-1	0	0	0	0	0	-1
2. aşama									0	0	-1
1	1	1	-1	-1	1	1	1	1	1	1	0
1	0	1	1	1	-1	-1	0	-1	0	1	-1
0	1	1	0	0	-1	0	-1	-1	0	0	-2
0	0	1	-2	-1	-1	0	0	0	0	0	-2
3. aşama									0	0	-2
1	1	1	-2	-1	1	1	1	1	1	1	-1
1	0	1	0	0	-1	-1	0	-1	0	1	-2
0	1	1	-1	-1	-1	0	0	0	0	1	-2
0	0	1	-2	-1	-1	0	0	0	0	1	-2
4. aşama											
1	1	1	-1	-1	1	1	1	1	1	2	-1
1	0	1	0	0	-1	-1	0	-1	0	2	-2
0	1	1	0	0	-1	0	-1	-1	0	1	-3
0	0	1	-3	-1	-1	0	0	0	0	1	-3
5. aşama									0	1	-3
1	1	1	-2	-1	1	1	1	1	1	2	-2
1	0	1	-1	-1	-1	0	0	0	1	2	-2
0	1	1	0	0	-1	0	-1	-1	1	1	-3
0	0	1	-3	-1	-1	0	0	0	1	1	-3
6. aşama									1	1	-3
1	1	1	-1	-1	1	1	1	1	2	2	-2
1	0	1	0	0	-1	-1	0	-1	1	2	-3
0	1	1	-1	-1	-1	0	0	0	1	2	-3
0	0	1	-3	-1	-1	0	0	0	1	2	-3
7. aşama									1	2	-3
1	1	1	0	0	1	1	1	1	2	3	-2
1	0	1	0	0	-1	-1	0	-1	1	3	-3
0	1	1	0	0	-1	0	-1	-1	1	2	-4
0	0	1	-4	-1	-1	0	0	0	1	2	-4
8. aşama									1	2	-4
1	1	1	-1	-1	1	1	1	1	2	3	-3
1	0	1	-1	-1	-1	0	0	0	2	3	-3
0	1	1	0	0	-1	0	-1	-1	2	2	-4
0	0	1	-4	-1	-1	0	0	0	2	2	-4
9. aşama									2	2	-4
1	1	1	0	0	1	1	1	1	3	3	-3
1	0	1	0	0	-1	-1	0	-1	2	3	-4
0	1	1	-1	-1	-1	0	0	0	2	3	-4
0	0	1	-4	-1	-1	0	0	0	2	3	-4
10. aşama									2	2	-4
1	1	1	1	1	1	0	0	0	2	3	-4
1	0	1	-2	-1	-1	0	0	0	2	3	-4
0	1	1	-1	-1	-1	0	0	0	2	3	-4
0	0	1	-4	-1	-1	0	0	0	2	3	-4

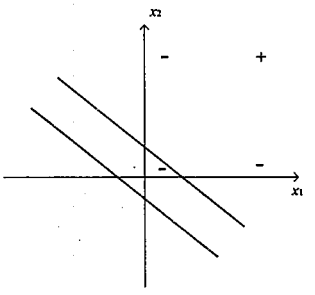
Bu aşamalar esnasında girişler sonucunda elde edilen bazı ayırma çizgileri şekil 2.19.'da verilmiştir:



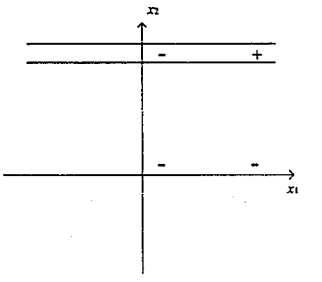
Birinci aşama
Birinci giriş için ayırma çizgileri:
$x_1 + x_2 + 1 = 0,2$ ve $x_1 + x_2 + 1 = -0,2$



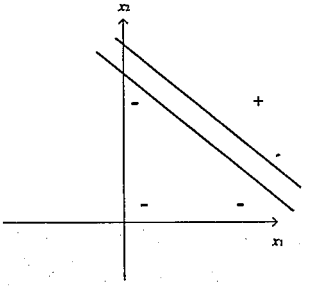
Birinci aşama
İkinci giriş için ayırma çizgileri:
$x_2 = 0,2$ ve $x_2 = -0,2$



İkinci aşama
Birinci giriş için ayırma çizgileri:
$x_1 + x_2 = 0,2$ ve $x_1 + x_2 = -0,2$



İkinci aşama
İkinci giriş için ayırma çizgileri:
$x_2 - 1 = 0,2$ ve $x_2 - 1 = -0,2$



Onuncu aşama
Son giriş için ayırma çizgileri:

$$x_2 = -\frac{2}{3}x_1 + \frac{7}{5} \text{ ve } x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$$

Şekil 2.19. Örnek 2.4. için elde edilen bazı ayırma çizgileri

Son aşamadan sonra şekil 2.20.'de gösterildiği gibi, sınır doğrusu olan pozitif cevaplar $2x_1 + 3x_2 - 4 > 0,2$ eşitsizliğinin, ve sınır doğrusu olan negatif cevaplar ise $2x_1 + 3x_2 - 4 < -0,2$ eşitsizliğinin verdiği noktalarla belirlenir.

Örnek 2.5.

Perseptron öğrenme kuralını kullanan iki kutup girişli ve iki kutup çıkışlı *VE* mantıksal fonksiyonunu göz önüne alındığında, eğitim verisi çizelge 2.10.'deki gibidir:

Çizelge 2.10. *VE* fonksiyonu için iki kutuplu girdiler ve iki kutuplu hedefler

GİRİŞ			HEDEF
x_1	x_2	*1*	
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Örnek 2.4.'de sonuç on aşamada gerçekleşti. Örnek 2.5.'te girdiler ve hedefler iki kutuplu değerlerdir. Giriş ve hedef verileri iki kutuplu olduğunda sonuç iki aşamada gerçekleşir ve bu aşamalardan biri sonucu doğrulayan test aşamasıdır. Eşik değeri θ = 0,2, öğrenme oranı $\alpha = 1$, başlangıç ağırlıklar ve başlangıç sapma değeri 0 olarak ayarlandığında elde dilecek hesaplama sonuçları çizelge 2.11.'de gösterilmiştir:

Çizelge 2.11. *VE* fonksiyonu için iki kutuplu girdiler ve iki kutuplu hedefler için perseptron uygulaması

GİRİŞ			NET	ÇIKIŞ	HEDEF	AĞIRLIK DEĞİŞİMLERİ			AĞIRLIKLAR		
x_1	x_2	*1*				Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	-1	1	1	1	-1	-1	1	-1	0	2	0
-1	1	1	2	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1
2. aşama									1	1	-1
1	1	1	1	1	1	0	0	0	1	1	-1
1	-1	1	-1	-1	-1	0	0	0	1	1	-1
-1	1	1	-1	-1	-1	0	0	0	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

İkinci aşamada Δw=0'dır, başka bir ifadeyle ilk aşamadan sonra sistem tam olarak eğitilmiştir. Birinci aşamada doğru sonuç elde edilse bile onun kanıtlanması için ikinci aşamanın yapılması önemlidir.

Örnek 2.4. ve örnek 2.5.'ten görüleceği gibi verilerin tipi, yapay sinir ağı algoritmalarında önemlidir. İki kutuplu veriler, ikili verilere göre daha hızlı sonuç verir. Bu sadece perseptron algoritması için değil, diğer bir çok yapay sinir ağı algoritmaları içinde geçerlidir.

2.9.2.3. Perseptron Öğrenme Kuralı Yakınsama Teoremi

Perseptron öğrenme kuralı aşağıdaki şekildedir:

Belirlenmiş sonlu P sayıda $,x(p)\ \ p=1,...,P$ eğitim vektörleri "+1" veya "-1" değerlerini ala bilen $t(p),\ \ p=1,...,P$ hedef değerleri ile ilişkilidir ve $y=f(y\_in)$ aktivasyon fonksiyonu,

$$f(y\_in)=\begin{cases}-1\ , & y\_in<-\theta \ \text{ise}\\ 0\ , & -\theta\le y\_in\le\theta \ \text{ise}\\ 1\ , & y\_in>\theta \ \text{ise}\end{cases} \qquad (2.16)$$

şeklindedir ve ağırlıklar aşağıdaki şekilde güncellenir:

Eğer $y\neq t$ ise $w(yeni)=w(eski)+tx$

değilse ağırlıklarda değişme yapılmaz.

Perseptron öğrenme kuralı yakınsama teoremi ise aşağıda verildiği gibidir:

Teorem: Eğer tüm p numuneleri ve belirli bir w^* ağırlık vektörü için $f(\mathbf{x}(p).w^*)=t(p)$ ise, bu durumda herhangi bir w_0 başlangıç ağırlık vektörü için perseptron öğrenme algoritması, sonlu adım sonucunda tüm eğitim örnekleri için doğru sonucu veren bir ağırlık vektörüne yaklaşır. Bu ağırlık vektörünün tek olması ve w^* olması zorunlu değildir.

Bu teoremin anlamı; eğer problem ayrılabilen ise, başka bir ifadeyle problemin çözümü var ise perseptron algoritması bu problemi çözebilir. Bu

teorem, perseptron kuralının en önemli özelliklerinden biridir ve Hebb kuralı karşısındaki avantajını göstermektedir. [5,10,32]

2.9.3. Delta Kuralı

Delta kuralı, Widrow ve Hoff (1960) tarafından ADALINE için ortaya atılmış olan iteratif bir öğrenme sürecidir. Delta kuralında, tüm girdi numuneleri için çıktı ve hedef farkları karelerinin toplamının, başka bir ifadeyle, toplam hatanın küçültülmesine hedeflenmiştir. Uygun algoritmalarda her numune için gradient vektörünün ters yönünde ağırlıkların güncellenmesi yapılır. Bu durumda delta kuralı, nöron bağlantılarının ağırlıklarını, ağ girişi (y_in) ve ağın hedef çıkışı (t) arasındaki farkı en aza indirgeyecek şekilde değiştirir. Amaç, tüm eğitim numunelerinin hatalarını en aza indirgemektir. Ağırlık düzeltmeleri, çok sayıdaki eğitim numunesi ile beraber biriktirilebilir ve bu yığın güncelleştirilmesi olarak adlandırılır.

Bu kural, örnek ilişkilendirmede doğrusal bağımsız olan numuneler için de kullanılabilir. Doğrusal bağımsız olan girdi haritaları, tek katmanlı ağlar kullanılarak çözülebilir. Buna rağmen basit öğrenme kurallarında karşılaşılan hat karışmasından kaçınmak için delta kuralına ihtiyaç duyulur. Numunelerin doğrusal bağımsız olmadığı durumlarda delta kuralı en küçük kareler çözümünü üretecektir.

2.9.3.1. Geliştirilmiş Delta Kuralı

Widrow ve Hoff'un ileri sürdüğü orjinal delta kuralında çıktı biriminin aktivasyon fonksiyonu olarak özdeşlik fonksiyonu kullanılmaktadır. Genel olarak geliştirilmiş delta kuralı ele alınacaktır. Bu durumda çıktı katmanının aktivasyon fonksiyonu herhangi bir diferansiyellenebilir aktivasyon fonksiyonunu kullanabilecektir.

n girdi birimi ve m çıktı birimi olan tek katmanlı yapay sinir ağında çıktı katmanının aktivasyon fonksiyonunu $f(x)$ ise, herhangi bir j. çıktı birimine girdi sinyali,

$$y\_in_j = b_j + \sum_i x_i w_{ij} \quad , j = 1,...,m \tag{2.17}$$

olursa, o birimin aktivasyonu $y_j = f(y\_in_j)$ şeklinde hesaplanacaktır. Bu durumda, bir girdi numunesi için hata kareler toplamı,

$$E = \sum_{j=1}^{m} \left(t_j - y_j\right)^2 \tag{2.18}$$

şeklinde olacaktır. Bu durumda E, ağırlıkların karmaşık fonksiyonu ve bu fonksiyonun gradient vektörü, ağırlıklara göre kısmi türevlerin vektörü olacaktır. Gradient vektörü yönünde fonksiyonun değeri artar, ters yönünde ise değer azalır. Bunun içinde hatanın azaltılması yönünü bulmak için $-\partial E / \partial w_{ij}$ kısmi türevinin bulunması gerekir. w_{ij} ağırlığının sadece y_j birimini etkilediği $\partial E / \partial w_{ij}$ türevi,

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{j=1}^{m} (t_j - y_j)^2 = \frac{\partial}{\partial w_{ij}} (t_j - y_j)^2 \tag{2.19}$$

şeklinde bulunur. Ayrıca, $y\_in_j = \sum_i x_i w_{ij}$ ve $y_j = f(y\_in_j)$ eşitlikleri kullanılırsa sonuç olarak bu türev,

$$\frac{\partial E}{\partial w_{ij}} = -2(t_j - y_j) \frac{\partial y\_in_j}{\partial w_{ij}} = -2(t_j - y_j) x_i f'(y\_in_j) \tag{2.20}$$

eşitliğiyle elde edilir. Böylelikle, belirli bir α öğrenme oranı için delta kuralı ile ağırlıkların değişimi

$$\Delta w_{ij} = \alpha (t_j - y_j) x_i f'(y\_in_j) \tag{2.21}$$

ve ağırlık güncellenmesi

$$w_{ij}(yeni) = w_{ij}(eski) + \Delta w_{ij} \tag{2.22}$$

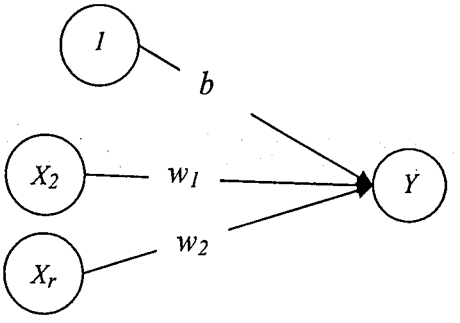
şeklinde olacaktır.

Ağırlık değişimi formülünde aktivasyon fonksiyonunun türevi kullanılmaktadır. İlk bakışta bu türevin bulunması zor görünmektedir. Ancak aktivasyon fonksiyonu differansiyellenebilir olmakla beraber onun türevi bu fonksiyonun kendisiyle ifade edilmelidir. Başka bir ifadeyle fonksiyonun kendisinin yardımıyla hesaplanabilir. Örneğin, $f(x)=\frac{1}{1+e^{-\sigma x}}$ ikili sigmoid fonksiyonu için türev ifadesi $f'(x)=\sigma f(x)[1-f(x)]$ şeklindedir. [5,10,32]

2.9.4. ADALINE

ADALINE (Adaptive Linear Neuron), Widrow ve Hoff (1960) tarafından tanımlanmış bir doğrusal aktivasyon fonksiyonlu ve delta kuralına dayanan bir perseptrondur. Bu ağda giriş sinyalleri ve hedef çıktısı olarak genellikle iki kutuplu değerler kullanmaktadır; fakat bu değerlerle sınırlı değildir. İkili ve ya sürekli değerlerde olabilir.

Başlangıçta ADALINE için tek bir çıktı birimi olarak düşünülse de günümüzde geliştirilmiş formunda çıktı birimlerinin sayısı birden fazlada olabilir. Şekil 2.20.'de gösterildiği gibi ADALINE doğrusal perseptrondur başka bir ifadeyle doğrusal aktivasyon fonksiyonu olan tek katmanlı bir yapay sinir ağıdır. Ağırlıkların güncellenmesi delta kuralı ile yapılmaktadır. [7]



Şekil 2.20. ADALINE mimarisi

ADALINE'nın çok katmanlı mimarileri de incelenmiştir, böyle yapay sinir ağ modelleri MADALINE olarak bilinmektedir.

2.9.4.1. ADALINE için Eğitim Algoritması

Aşağıda, ağ eğitildikten sonra aktivasyon fonksiyonunun kullanımını göstermek için bir uygulama algoritması verilmiştir:

Adım 0 Başlangıç ağırlıkları ata.
(Genellikle küçük rasgele değerler kullanılır.)
Öğrenme oranını α'yı ayarla.
(Algoritmayı izleyen yorumlara bakınız.)

Adım 1 Dur koşulu yanlış olduğu sürece Adım 2-6'ya devam et.

Adım 2 Her bir s:t iki kutuplu eğitim çifti için Adım 3-5'i yap.

Adım 3 Giriş birimleri için aktivasyonları ayarla , $i = 1, ..., n$

$$x_i = s_i.$$

Adım 4 j. çıkış birimi için ağırlıklı girdi sinyalini hesapla:

$$y\_in_j = b_j + \sum_i x_i w_{ij} \ , \ j = 1,...,m$$

Adım 5. Sapma ve ağırlıkları güncelleştir, $i = 1,...,n; j = 1,...,m$

$$b_j(yeni) = b_j(eski) + \alpha(t_j - y\_in_j)$$

$$w_{ij}(yeni) = w_{ij}(eski) + \alpha(t_j - y\_in_j)x_i$$

Adım 6. Durma koşulunu test et :
Eğer Adım 2'deki en büyük ağırlık değişimi belirtilmiş toleransdan daha küçük ise dur, aksi takdirde devam et.

Öğrenme oranını uygun bir değere ayarlarken dikkat edilmelidir. Hecht-Nielsen (1990)'e göre üst sınır değeri *x(p)*, giriş vektörlerinin oluşturduğu *R* korelasyon matrisinin en büyük öz değerinden bulunabilir:

$$R = \frac{1}{P}\sum_{p=1}^{P} x(p)^T x(p) \qquad (2.23)$$

Başka bir ifadeyle α < R 'in en büyük öz değerinin yarısı olarak ayarlanabilir. Fakat, ağırlık güncellemelerin hesaplanabilmesi için, *R*'nin hesaplanması gerekmediğinden başlangıçta genellikle α için küçük bir değer (α = 0.1 gibi) kullanılabilir. Eğer gereğinden büyük bir değer seçilirse, o zaman

öğrenme süreci yakınsamayacaktır, bunun aksine çok küçük bir değer seçilirse o zamanda öğrenme çok yavaş gerçekleşecektir. Bir tek nöron için elverişli bir α öğrenme oranı aralığı, *n* değeri giriş birimlerinin sayısını göstermek üzere, $0{,}1 \leq n\alpha \leq 1{,}0$ olur.

ADALINE'nın eğitim sürecindeki yakınsama delta kuralının açıklanmasındaki türevin hesaplanması işleminde görünmektedir. [5]

2.9.4.2. ADALINE Uygulaması

Bir ADALINE birimi, eğitimden sonra, giriş numunelerin sınıflandırılmasında kullanılabilir. Eğer hedef değerleri iki değerli (ikili ve ya iki kutuplu) iseler eşik fonksiyonu çıktı biriminin aktivasyon fonksiyonu olarak uygulanabilir. Aşağıdaki algoritmada iki kutuplu hedefler için adım fonksiyonu gösterilmiştir:

Adım 0 Ağırlıkları ayarla.
(ADALINE eğitim algoritmasından).

Adım 1 Her bir **x** girdi örnek vektörü için, Adım 2-4'ü yap.

Adım 2 Girdi birimlerinin aktivasyonlarını **x** 'e göre ayarla.

Adım 3 Çıktı birimleri için ağ girdisini hesapla :

$$y\_in_j = b_j + \sum_i x_i w_{ij} \quad , \; j = 1,\ldots,m$$

Adım 4 Aktivasyon fonksiyonunu uygula :

$$y_j = \begin{cases} -1 & , \; y\_in_j < 0 \quad \text{ise} \\ 1 & , \; y\_in_j \geq 0 \quad \text{ise} \end{cases} \quad ; \; j = 1,\ldots,m$$

Örnek 2.6.

ADALINE iki kutuplu girdiler ve hedefler için bildirilmişse de, delta kuralı ikili giriş içinde geçerlidir. Bu örnekte, *VE* fonksiyonu, ikili giriş ve iki kutuplu hedefler için ele alınacaktır. Fonksiyon çizelge 2.12.'de gösterilen dört eğitim numunesi ile tanımlanmıştır:

Çizelge 2.12. *VE* fonksiyonu için ikili girdiler ve iki kutuplu hedefler

GİRİŞ			HEDEF
x_1	x_2	*1*	
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

Bir ADALINE ağırlıkları bulunurken, hata kareleri toplamı, $E = \sum_{p=1}^{4} (x_1(p)w_1 + x_2(p)w_2 + w_0 - t(p))^2$ 'yi minimize edecek şekilde tasarlanır. Burada, $x_1(p)w_1 + x_2(p)w_2 + w_0$, p numunesinin çıktı birimi için net girdisi ve *t(p)* ise p numunesinin ilgili hedefidir. [5]

Bu hatayı minimize eden ağırlıklar $w_1 = 1$ ve $w_2 = 1$ olup, sapma ise $w_0 = -\frac{3}{2}$'dir.Böylece, ayırma çizgisi $x_1 + x_2 - \frac{3}{2} = 0$ olur.

Bu ağırlıklarla dört eğitim numunesi için hata kareler toplamı 1'e eşittir.

2.10. Çok Katmanlı Yapay Sinir Ağ Modelleri

Tek katmanlı ağlar ayrılabilmeyen problemlerin çözümünde başarısız olmaktadırlar. Bunun içinde bilim adamları çok katmanlı YSA modellerini incelemişlerdir. Burada önemli aşamalardan biri bu tip ağlar için akıllı bir eğitim algoritmasını geliştirmekti. 1986 yılında Rumelhart, Hinton ve Williams tarafından bu gerçekleştirildi. Benzer fikirler daha önce Werbos (1974), Parker (1985), Cun (1985) gibi bilim adamlarının yayınlamış olduğu makalelerde gözükmektedir. Standart geriye yayılım (backpropagation) olarak adlandırılan bu eğitim metodu hata kareler toplamının geriye yayılım yöntemiyle küçültülmesi fikrine dayanır ve genelleştirilmiş delta kuralını kullanır. Dolayısıyla bu yöntem her adımda hatanın küçültülmesi için, Widrow-Hoff eğitiminde olduğu gibi, gradient azalış yöntemini kullanır. Bu durumda gizli katmanda doğrusal olmayan aktivasyon fonksiyonlar, örneğin logistik sigmoid fonksiyonu ve ona uygun olarak genelleştirilmiş delta kuralı uygulanmaktadır. Farklı geriye yayılım yöntemlerinin temelinde aşağıdaki aşamalar uygulanır:

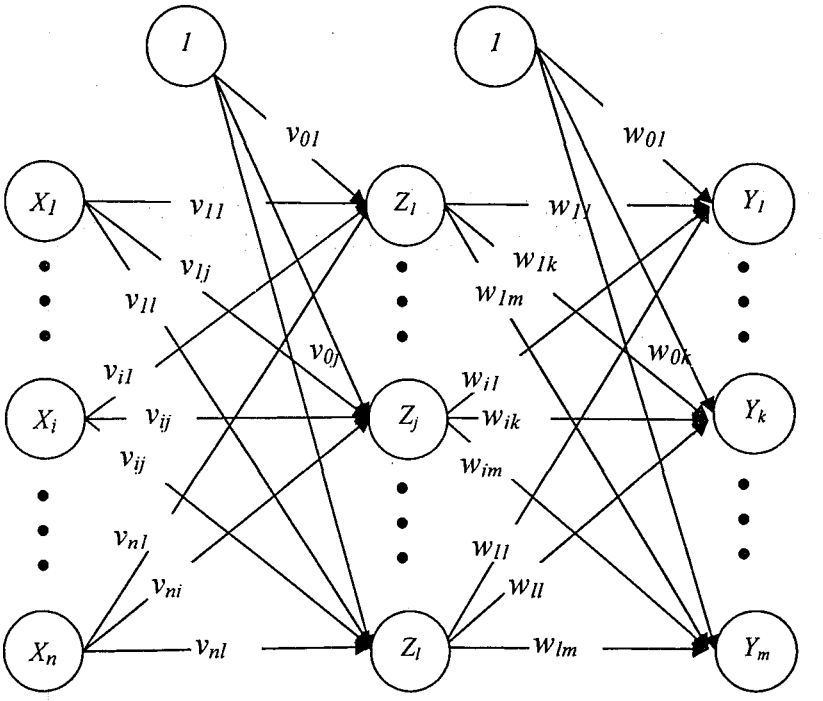
- Girdi eğitim örneklerinin ileri beslemesi,
- Birleşmiş hatanın hesaplanması ve geriye yayılması,
- Ağırlıkların güncellenmesi.

Eğitimden sonra ağ uygulaması sadece ağın ileri besleme aşamasını gerçekleştirir ve hesaplamaları yapar.

Geriye yayılım yöntemi çok katmanlı ağlar için genelleştirilmiş delta kuralını ve doğrusal olmayan aktivasyon fonksiyonunu kullanarak, belirli girdi ve hedeflere uygun herhangi bir doğrusal olmayan fonksiyonu yaklaşımını gerçekleştirme yeteneğine sahip olur. Bu yöntem daha iyi tahmin yapmak, sınıflandırmak ve öngörü problemleri için büyük imkanlar sağlamaktadır. [12]

2.10.1. Standart Geriye Yayılım Ağ Mimarisi

Şekil 2.21.'de bir gizli katmana sahip çok katmanlı ileri beslemeli bir ağ gösterilmiştir. Burada X girdi katmanı, Z gizli katman, Y çıktı katmanına ait birimlerdir. Gizli katmanın j. birimine dahil olan sapma ağırlıkları v_{0j} $(j=1,...,l)$, çıktı katmanının k. birimine dahil olan sapma ağırlıkları w_{0k} $(k=1,...,m)$ ve sapmalara uygun birimlerin girdi sinyalleri "1" olarak gösterilmiştir. Ağda sinyallerin yayılımı girdi birimlerinden gizli birimlere sonra ise, gizli birimlerden çıktı birimlerine doğru yönelmiştir. Bu nedenle ağ ileri beslemeli çok katmanlı ağ gibi göze alınmaktadır. Gizli birimin sayısı "1" olduğundan şekilde gösterilen ağ iki katmanlı bir yapay sinir ağıdır. Gizli birimlerin sayısının birden fazla olduğu yapay sinir ağ mimarileri de vardır. [5,12]



Şekil 2.21. Tek gizli katmanlı ileri beslemeli çok katmanlı bir yapay sinir ağı

2.10.2. Standart Geriye Yayılım Eğitim Algoritması

Standart geriye yayılım yöntemi üç aşamadan oluşmaktadır:

İleri Besleme: Bu aşamada, her bir *Xi* girdi birimi, bir girdi sinyali alarak bu sinyali tüm $Z_1,...,$ *Zp* gizli birimlerine iletirler. Daha sonra her bir gizli birim z_j aktivasyonlarını hesaplayarak onları tüm çıktı birimlerine gönderirler. Çıktı birimlerinde y_k aktivasyonları hesaplanır ve ağın uygun örneği için ağın cevabı belirlenir.

Hatanın Hesaplanması ve Geriye Yayılması: Bu aşamada, her bir çıktı birimi için hata, t_k hedefi ile y_k çıktı sinyalinin farkı şeklinde bulunur. Bu hata yardımıyla δ_k $(k=1,...,m)$ faktörü hesaplanır ve y_k çıktı birimindeki hatayı ondan önceki katmandaki tüm *Z* gizli birimlere dağıtmak için kullanılır. Bu değer aynı zamanda çıktı ve gizli birimlerine uygun ağırlıkları güncellemek için kullanılır. Benzer olarak, her z_j gizli birimi için δ_j^h $(j=1,...,p)$ faktörleri hesaplanır. Hatanın gizli birimden girdi birimine geri yayılması gerekli değildir, fakat δ_j^h faktörleri gizli birim ile girdi birimi arasındaki ağırlıkların güncellenmesinde kullanılır.

Ağırlıkların Güncellenmesi: Bu aşamada ise tüm δ faktörlerin yardımıyla tüm ağırlıklar eş zamanlı olarak uygun formüllerle güncellenir. w_{jk} ağırlıklarını güncellemek için temel olarak δ_k faktörleri ve gizli katmanın z_j aktivasyonları kullanılır. Benzer olarak v_{ij} ağırlıklarını güncellemek için δ_j^h ve x_i girdi aktivasyonları kullanılır.

İleri yayılımlı eğitim algoritmasında herhangi bir aktivasyon fonksiyonu kullanılabilir. Gizli katmanda daha çok doğrusal olmayan, örneğin sigmoid fonksiyonları kullanılır. Çıktı katmanında doğrusal aktivasyon fonksiyonu da kullanılabilir. Ancak, gizli ve çıktı katmanlarında aynı zamanda doğrusal aktivasyon fonksiyonu kullanılırsa ağın doğrusal olmama özelliği kaybolur. Bu yüzden belirli sınıf problemlerin çözümünde uygun aktivasyon fonksiyonların seçimi çok önemli olmaktadır.

Yukarıda aşamaları açıklanan standart geriye yayılım eğitim algoritması aşağıdaki gibidir:

I. AŞAMA : İleri Besleme

Adım 0 Ağırlıkların ilk değerlerini başlat (küçük rasgele değerler olarak ayarla).

Adım 1 Durma koşulu yanlış iken 2-9 adımlarını uygula.

Adım 2 Tüm eğitim çiftleri için 3-8 adımlarını uygula.

Adım 3 Tüm *Xi (i=1,...,n)* girdi birimlerinin x_i girdi sinyallerini al ve bu sinyali üst katmanın tüm elemanlarına (gizli birimlere) dağıt.

Adım 4 Tüm Z_j $(j=1,...,p)$ gizli birimlerin ağırlıklı girdi sinyallerini topla:

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \ ,$$

ve çıktı sinyalini hesaplamak için aktivasyon fonksiyonunu uygula:

$$z_j = f(z\_in_j)$$

ve bu sinyali üst katmanındaki tüm elemanlarına gönder.

Adım 5 Her Y_k (k =*1,...,m*) çıktı birimi ağırlıklı girdi sinyallerini topla:

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

ve çıktı sinyalini hesaplamak için aktivasyon fonksiyonunu uygula:

$$y_k = f(y\_in_k)$$

II. AŞAMA : Hatanın Hesaplanması ve Geriye Yayılması

Adım 6 Her Y_k (k =*1,...,m*) çıktı birimi girdi numunesine uygun bir hedef numunesi alır ve onun hata bilgi terimini hesapla:

$$\delta_k = (t_k - y_k).f'(y\_in_k)$$

ve bu birim için ağırlık değişim terimini hesapla:

$$\Delta w_{jk} = \alpha \delta_k z_j$$

ve sapma değişim terimini hesapla:

$$\Delta w_{0k} = \alpha \delta_k$$

ve δ_k değerini alt tarafındaki katmanın birimlerine gönder.

Adım 7 Her Z_j (j=*1,...,p*) gizli birimlerinin delta girdilerini topla:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

ve bu değer için hata bilgi terimini hesaplamak için aktivasyon fonksiyonunun türevi ile çarp:

$$\delta_j^h = \delta\_in_j f'(z\_in_j)$$

ve ağırlık değişim terimini hesapla:

$$\Delta v_{ij} = \alpha \delta_j^h x_i$$

ve sapma değişim terimini hesapla:

$$\Delta v_{0j} = \alpha \delta_j^h$$

III. AŞAMA : Ağırlıkların Güncellenmesi

Adım 8 Her bir çıktı biriminin sapmasını ve ağırlıklarını güncelle:

$$w_{jk}(yeni) = w_{jk}(eski) + \Delta w_{jk}$$

Her bir gizli birimin sapmasını ve ağırlıklarını güncelle:

$$v_{ij}(yeni) = v_{ij}(eski) + \Delta v_{ij}$$

Adım 9 Durma koşulunu kontrol et.

Her bir döngüde tüm eğitim vektörlerinin tüm kümesi bir kez uygulanır. Bir geri yayılım sinir ağının eğitimi için bir çok döngü gerekir. Buradaki algoritmada her eğitim numunesi kullanıldıktan sonra ağırlıklar güncellenir. Diğer yaygın olarak kullanılan başka bir uygulamada yığın güncelleme (batch updating)'dir. Bu güncelleme türünde ise ağırlık güncellemeleri bir döngüde biriktirilir ve döngü bitince uygulanır.

Geriye yayılım algoritmasının matematiksel temeli eğim azaltma optimizasyon tekniğidir. Bir fonksiyonun eğimi, bu fonksiyonun hangi yönde daha hızlı arttığını gösterir. Eğimin ters yönü ise bu fonksiyonun hangi yönde en hızlı şekilde azaldığını gösterir. [5,32]

2.10.3. Geriye Yayılım Algoritma Çeşitleri

2.10.3.1. Momentum

Momentum geriye yayılımda , ağırlık değişiminin yönü o anki eğimle bir önceki eğimin kombinasyonu şeklindedir. Bu eğim azaltma yönteminin değiştirilmiş bir şeklidir ve bazı eğitim verileri, eğitim verilerinin büyük bir çoğunluğundan farklılık gösteriyorsa bu değişim iyi bir avantaj sağlar. Eğer hiç alışılmamış bazı veriler kullanılacaksa bu değişikliği küçük bir öğrenme oranı ile kullanmak iyi olacaktır. Bununla birlikte eğitim verileri benzer olsa da bu değişiklik kullanılarak yaklaşmanın hızı arttırılabilir. Momentumu kullanmak için bir veya daha önceki eğitim numunelerinin ağırlıkları saklanmalıdır.

Örnek olarak, geri yayılımın momentumlu basit bir biçimi, *t+1* eğitim adımının yeni ağırlıkları *t* ve *t–1* eğitim adımlarındaki ağırlıkları temel alır. Momentumlu geri yayılımın ağırlık güncelleme formülü,

$$w_{jk}(t+1) = w_{jk}(t) + \alpha\delta_k z_j + \mu[w_{jk}(t) - w_{jk}(t-1)] \quad (2.24)$$

$$\Delta w_{jk}(t+1) = \alpha\delta_k z_j + \mu\Delta w_{jk}(t) \quad (2.25)$$

$$v_{ij}(t+1) = v_{ij}(t) + \alpha\delta_j x_i + \mu[v_{ij}(t) - v_{ij}(t-1)] \quad (2.26)$$

$$\Delta v_{ij}(t+1) = \alpha\delta_j x_i + \mu\Delta v_{ij}(t) \quad (2.27)$$

şeklindedir. Momentum katsayısı değeri olan μ, 0 -1 aralığında sınırlandırılmıştır.

Momentum, ağın ağırlık değişimini kabul edilebilir ölçülerde yapmasını sağlar ve küçük bir öğrenme oranı ile birlikte kullanıldığında herhangi bir örnek için büyük hatalara yol açacak olan cevapları engeller. Momentum kullanıldığında ağ eğimin yönünde işlem yapmaz. O anki ve ondan önceki ağırlık düzeltmesinin yönlerinin kombinasyonunun yönünde işlem yapar. [5,10,13]

2.10.3.2. Eşlenik Eğim Algoritması (The Conjugate Gradient Algorithm)

Standart geriye yayılım algoritması, ağırlıkları eğimin ters yönünde adımsal olarak ayarlamaktadır. Bu yön, ağ performans fonksiyonunun en hızlı azaldığı yöndür. Fonksiyon eğimin ters yönü boyunca en hızlı şekilde azalsa da, bu önemli sayılacak hızlı bir yakınsama üretmemektedir. Eşlenik eğim algoritmalarında, genellikle adımsal azalma yönlerinden daha hızlı yakınsamayı sağlayan eşlenik yönler boyunca bir arama yapılmaktadır.

Bir çok eğitim algoritmasında ağırlık güncellenmesinin belirlenmesinde bir öğrenme oranı kullanılmaktadır. Çoğu eşlenik eğim algoritmalarında ağırlık güncellemesi her bir tekrarda yapılmaktadır. Ağ performans fonksiyonunu minimize eden bu işlem için eşlenik eğim yönleri boyunca bir arama yapılmaktadır. Bir çok arama fonksiyonu vardır. Bazı arama fonksiyonları, belirli eğitim fonksiyonları için çok uygundur.

Bütün eşlenik eğim algoritmaları ilk tekrarda eğimin ters yönünde arama işlemine başlamaktadır:

$$\mathbf{p}_0 = -\mathbf{g}_0 \tag{2.28}$$

Geçerli arama yönü boyunca taşımak için en iyi uzunluğun belirlenmesi amacıyla bir doğru araması yapılır:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{2.29}$$

Daha sonra önceki arama yönlerine eşlenik olan bir sonraki arama yönü belirlenir. Yeni arama yönü belirlenmesinde kullanılan genel yöntem bir önceki arama yönü ile yeni adımsal azalma yönünün birleştirilmesidir:

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} \tag{2.30}$$

Eşlenik eğimin hesaplanan β_k sabitinin davranış şekline göre birkaç çeşidi vardır. Bunlardan biri olan Fletcher-Reeves algoritmasında güncelleme için yapılan işlem,

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{2.31}$$

şeklindedir. Bu eşitlik geçerli eğimin karesel normunun bir önceki eğimin karesel normuna oranıdır.

Eşlenik eğim algoritmasının diğer bir çeşidi de Polak ve Ribiére tarafından önerilmiştir. Fletcher-Reeves algoritmasında olduğu gibi, her bir tekrardaki yön araması (2.30) eşitliği ile belirlenir.

Polak- Ribiére güncellemesinde β_k sabiti,

$$\beta_k = \frac{\Delta\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{2.32}$$

eşitliği ile hesaplanır.

Bütün eşlenik eğim algoritmalarında, arama yönü eğimin negatif yönüne göre periyodik olarak tekrarlanır. Standart tekrarlama noktası, tekrar sayısı ağ parametre sayısı (ağırlıklar ve sapmalar)'na eşit olduğunda meydana çıkmaktadır. Ancak eğitimin etkinliğini geliştirebilen diğer tekrarlama yöntemleri de vardır. Bu yöntemlerden biri Beale'nin önceki yöntemleri üzerine dayanan Powell'in önerdiği yöntemdir. Bu yöntemde eğer geçerli eğim ve bir önceki eğim arasında ufak bir dikeylik varsa işlemler yeniden başlatılır. Bu ise (2.33)'deki eşitsizlikle test edilir.

$$\left|\mathbf{g}_{k-1}^{T}\mathbf{g}_{k}\right| \geq 0,2\left\|\mathbf{g}_{k}\right\|^{2} \tag{2.33}$$

Eğer bu koşul sağlanırsa, arama yönü eğimin ters yönüne göre tekrarlanır.

2.10.3.3. Adapte Olabilen Öğrenme Oranları

Standart geriye yayılım algoritması, o anki ağırlıklar için olan hata yüzeyinde hatanın en hızlı azaldığı yönde ağırlıkları değiştirir. Ağırlık ayarlamasının yönünün değiştirilmesi hakkında çeşitli yöntemler önerilmiştir ve bu konuda çalışılmıştır. Bunlardan biri de, geriye yayılım ağının öğrenme oranının arttırılması için öğrenme oranının eğitim sırasında değiştirilmesidir. Başlangıç ağ çıktısı ve hatası bulunur . Her döngü de ağırlıklar ve sapma o anki öğrenme oranı kullanılarak hesaplanır ve yeni ağırlıklar ve hata bulunur. Yeni hata eski hatayı geçerse yeni ağırlıklar ve sapma uygulanmaz ve buna ek olarak öğrenme oranı azaltılır (genelde 0,7 ile çarpılır), aksi halde yeni ağırlıklar ve sapma uygulanır ve öğrenme oranı arttırılır (genelde 1,05 ile çarpılır).

2.10.3.4. Delta-Bar-Delta

Bu yaklaşımın temelinde, her ağırlığın kendi öğrenme oranının olması vardır. Öğrenme oranları eğitim sırasında değiştirilir. Eğer ağırlık değişimi (artma veya azalma) bir kaç adım için aynı yönde ise ağırlıklar için öğrenme oranı arttırılmalıdır. Bir ağırlık için oluşan hatanın kısmi türevinin işareti birkaç adım

için aynı olursa ağırlık değişimi aynı yönde değiştirilmelidir, tam tersi olursa azaltılmalıdır.

$w_{jk}(t)$ t rasgele seçilmiş bir ağırlığı, $\alpha_{jk}(t)$ bu ağırlık için öğrenme oranı ve E ise örnek için hata kareleri belirtmek üzere,

$$w_{jk}(t+1) = w_{jk}(t) - \alpha_{jk}(t+1)\frac{\partial E}{\partial w_{jk}} \tag{2.34}$$

şeklinde delta-bar-delta kuralı ağırlıkları güncellemektedir. Aslında bu işlem, standart geriye yayılım algoritmasının ağırlık güncellemesi aşamasının, her bir ağırlığın, kendisine uygun gelen hatanın kısmi türevinin farklı bir oranı ile güncellenmesi ile değiştirilmiş bir durumudur.

"Delta" değerleri,

$$\Delta_{jk} = \frac{\partial E}{\partial w_{jk}} = \delta_k z_j \tag{2.35}$$

ve

$$\Delta_{ij} = \frac{\partial E}{\partial v_{ij}} = \delta_j x_i \tag{2.36}$$

şeklinde belirlendikten sonra delta-bar-delta yöntemi her bir çıktı birimi için

$$\overline{\Delta}_{jk}(t) = (1-\beta)\Delta_{jk}(t) + \beta\overline{\Delta}_{jk}(t-1) \tag{2.37}$$

ve her bir gizli birimi için

$$\overline{\Delta}_{ij}(t) = (1-\beta)\Delta_{ij}(t) + \beta\overline{\Delta}_{ij}(t-1) \tag{2.38}$$

birleşimlerini kullanarak ağırlıkları güncelleme işlemini gerçekleştirir. Buradaki β $(0 < \beta < 1)$ parametre değeri kullanıcı tarafından belirlenmelidir.

Delta-bar-delta yönteminde her bir ağırlık kendi öğrenme oranı sahiptir. Bu oran eğitim sırasında

$$\alpha_{jk}(t+1)=\begin{cases}(1-\gamma)\alpha_{jk}(t) & , \quad \bar{\Delta}(t-1)\Delta(t)<0 \ ise \\ \alpha_{jk}(t)+\kappa & , \quad \bar{\Delta}(t-1)\Delta(t)>0 \ ise \\ \alpha_{jk}(t) & , \quad d.d.\end{cases} \tag{2.39}$$

yardımıyla değiştirilmektedir.

Bu yöntem, öğrenme oranını doğrusal bir biçimde arttırırken, onları üstel bir biçimde azaltır. Bu yolla, öğrenme oranlarının çok büyük olması ve çok hızlı bir şekilde artması da engellenmiş olur. Yine bu yolla, ağırlıkların pozitif olması ve hızlı bir biçimde azalması da sağlanabilir. [5,13]

2.10.4. Başlangıç Ağırlıklarının Seçimi

Başlangıç ağırlıklarının seçimi minimum hataya ne kadar hızlı yaklaşılacağını etkiler. Uygulamalarda başlangıç ağırlıklarının seçiminde sıkça kullanılan yöntemler rasgele ilk değer atama ve Nguyen–Widrow ilk değer atama yöntemleridir. [11]

2.10.4.1. Rasgele İlk Değer Atama

İki birim arasındaki ağırlık güncellemesi bu birimlerin aktivasyonlarınn türevine bağlıdır bu nedenle fonksiyonları ve türevlerinin aktivasyonlarını sıfır yapan başlangıç ağırlıklarından sakınılmalıdır. Ağırlıklar çok büyük olursa girdi ve çıktı birimlerine giden ilk girdi sinyalleri sigmoid fonksiyonunun türevinin değerinin küçük olduğu bölgeye düşecektir. Eğer ağırlıklar çok küçük olursa bu kez de 0 a yaklaşacaklardır. En yaygın yöntem (-0.5 , +0.5) veya (-1 , +1) arasında rasgele değerler seçmektir. [10]

2.10.4.1. Nguyen – Widrow İlk Değer Atama

Bu yöntem ile daha hızlı yaklaşma sağlanmaktadır. Yaklaşım gizli nöronların bir tek çıktı birimine olan cevaplarının geometrik analizine dayanmaktadır. İlk önce gizli birimlerle çıktı birimleri arasındaki ağırlıklar (-0.5 , +0.5) arasında ilk değer ataması yapılır.

Nguyen-Widrow ilk değer atama yöntemi, n giriş birimlerinin sayısı, p gizli birimlerin sayısı ve β orantı faktörü ($\beta = 0.7(p)^{1/n} = 0.7\sqrt[n]{p}$) olmak üzere aşağıda verilen üç aşamada gerçekleştirilir.

I. Giriş birimlerinden gelen ağırlık vektörünün başlangıç durumuna getirilmesi:

$v_{ij}(eski) = -0,5$ ile $0,5$ arasında rasgele bir sayı

$$\left\|v_j(eski)\right\| = \sqrt{V_{1j}(eski)^2 + V_{2j}(eski)^2 + ... + V_{nj}(eski)^2}$$

II. Ağırlıklara tekrar değer atanması:

$$v_{ij} = \frac{\beta v_{ij}(eski)}{\left\|v_j(eski)\right\|}$$

III. Sapmanın ayarlanması:

$v_{oj} = -\beta$ ile β arasında rasgele bir sayı

Nguyen-Widrow ilk değer atama yönteminde aktivasyon fonksiyonu olarak aşağıda verilen hiperbolik tanjant fonksiyonuna dayandırılır.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (2.40)$$

2.11. Verilerin Hazırlanması

Birçok problemde girdi vektörleri ve çıktı vektörleri aynı değer aralığındaki bileşenlere sahiptir. Girdi değerleri ve çıktılar ikili olarak hazırlanabilir fakat eğer girdiler iki kutuplu biçiminde hazırlanırsa ve iki kutuplu sigmoid fonksiyonu aktivasyon için kullanılırsa öğrenme arttırılabilir.

Birçok yapay sinir ağı uygulamasında veriler, sürekli değerli değişken, bir küme ve ya bir aralık şeklinde verilmiş olabilir. Örnek olarak, besin sıcaklığı gerçek sıcaklık şeklinde hazırlanmalıdır veya donmuş, soğumuş, oda sıcaklığı veya sıcak gibi sıcaklık aralıkları şeklinde verilmelidir. Bunun için dört adet iki kutuplu nöron kullanılabilir, normal durumda ise tek nöron kullanılmalıdır. Genelde nöron ağının ayrı cevapların kümesini öğrenmesi, sürekli değerleri öğrenmesinden daha kolaydır.

2.12. Gizli Katman Sayısının Belirlenmesi

Birden fazla gizli katmana sahip bir sinir ağını, algoritmada küçük değişiklikler yaparak gerçekleştirmek mümkündür. Girdi birimlerinden çıktı birimlerinin haritalanması işleminin belirli bir doğruluk yüzdesi ile yapılmasında tek bir gizli katman yeterlidir. Bununla birlikte, iki gizli katman bazen ağın eğitilmesini kolaylaştırabilir.

2.13. Gerekli Eğitim Çifti Sayısı

Uygun olan eğitim numunelerinin sayısı (P), eğitilecek olan ağırlık sayısı (W) ve beklenen sınıflandırmanın doğruluğu ise (e) olarak düşünülürse, ağ, yeterli sayıdaki eğitim çiftiyle $(1-e/2)$'lik eğitim numuneleri parçasını doğru şekilde sınıflandırmak için eğitilirse $0 < e \leq 1/8$ durumunda ağ (1-e) eğitim numunesini doğru sınıflandırabilir. Yeterli sayı,

$$\frac{W}{P} = e \tag{2.41}$$

ve ya

$$P = \frac{W}{e} \tag{2.42}$$

şeklinde hesaplanabilir.

Örnek olarak, e = 0,1 ile ağda 80 ağırlık varsa 800 eğitim çifti test numunelerinin %90'ının doğru sınıflandırılmasını sağladığı söylenebilir. [5,7,32]

2.14. Yapay Sinir Ağlarının Uygulama Alanları

Yapay sinir ağlarının uygulamaları gözden geçirildiğinde binlerce uygulamanın yapıldığı ve başarılı sonuçların elde edildiği görülebilir. Endüstriyel, finansal, askeri ve savunma, sağlık ve diğer bir çok alanlarda kullanılan yapay sinir ağları, bu uygulamalar incelendiğinde şu fonksiyonları gerçekleştirmek için uygulandıkları görülmektedir:

Tahmin: Bu amaçla kullanılan yapay sinir ağları, ağa sunulan bilgilerden yararlanılarak karşılık gelen çıktı değerlerini tahmin ederler. Hava tahmini, borsada hisse değerlerinin tahmini, döviz kurlarının tahmini gibi örnekler vermek mümkündür.

Sınıflandırma: Bu amaçla kullanılan yapay sinir ağları kendilerine sunulan bilgileri kategorize etmek görevini üstlenirler. Bir makine üzerinde görülen hataların sınıflandırılması buna örnek olarak verilebilir.

Veri İlişkilendirme: Bu amaçla eğitilen ağlar, ağa sunulan verilerin hatalı ve eksik olup olmadığını belirlerler. Öğrendikleri bilgiler ile eksik olan bilgileri tamamlarlar. Eksik bir resmin tamamlanması bu konuda örnek olarak verilebilir.

Veri Filtreleme: Bu amaçla eğitilen ağlar, bir çok veri arasından uygun verileri belirleme görevini yerine getirirler. Telefon konuşmalarındaki gürültüleri asıl konuşmalardan ayıran ağlar bu konudaki uygulamalara örnek olarak verilebilir.

Tanıma ve Eşleştirme: Değişik şekil ve örüntülerin tanınması, eksik, karmaşık, belirsiz bilgilerin işlenerek eşleştirme ve tanıma fonksiyonları

gerçekleştirilebilir. Kalite kontrol şemaları üzerindeki şekilleri tanıyan ağ, bu konuda örnek olarak verilebilir.

Teşhis: Bu amaçla geliştirilen ağlar, sistemlerin olumsuzluklarının ortaya konulması ve problemlerin teşhis edilmesi işlemini yerine getirirler. Makinelerin, süreçlerin arazi durumlarının ve hatalarının teşhis edilmesi buna örnek olarak verilebilir. Tıp alanında da bu tür sistemler yaygın olarak geliştirilmiştir.

Yorumlama: Bir olay hakkında toplanan örneklerden elde edilen ve eğitim sonucu oluşturulan bilgileri kullanarak yeni olayların yorumlanması işlemleri bu kapsamda düşünülmektedir. Bir olay hakkında toplanan verilerin yorumlanarak istatistiksel dağılımlarının belirlenmesi bu konu da örnek olarak verilebilir. [9]

3. YAPAY SİNİR AĞLARI VE BAZI İSTATİSTİKSEL TEKNİKLER

Bazı yapay sinir ağı modelleri, bazı istatistiksel tekniklere benzer veya aynısıdır. [14,15] Bu bölümde perseptronlar ile bazı istatistiksel teknikler arasındaki benzerlikler göz önüne alınacaktır. Bilindiği gibi regresyon analizinde, elde edilen gözlemlerin yardımıyla bağımlı değişkenlerle bağımsız değişkenler arasında bir fonksiyonel ilişkinin belirlenmesi, başka bir ifadeyle regresyon parametrelerinin bulunması ve bunlar yardımıyla ileri dönük tahmin yapılması amaçlanmaktadır. Bu problem matematiksel olarak belirli "girdi" ve "çıktı" değerleri verilen bir (çok değişkenli çoklu) fonksiyonun yaklaştırılması problemidir. Kolmogorov (1957), çok değişkenli sürekli bir fonksiyonun, sonlu sayıda bir değişkenli sürekli fonksiyonlar yardımıyla ifade edilebileceğini bir teoremle kanıtladı. Bu teoreme "Kolmogorov nöron ağ haritalama varlık teoremi" de denir. Bu teorem, herhangi sürekli bir fonksiyonun üç katmanlı (iki gizli ve çıktı katmanları) ileri beslemeli bir yapay sinir ağı ile kesin olarak tasvir edilebilirliğini anlatmaktadır. Bu bakımdan çok katmanlı perseptronun iyi bir yaklaşımcı olduğu gözükmektedir ve bu regresyon analizinde yapay sinir ağı modellerinin kullanımına yol göstermektedir.

Bundan dolayı, örnek sınıflandırma ve tanıma problemlerinde istatistiksel teknikler ve yapay sinir ağı modellerinin kullanımı için uygun teknik ve modellerin karşılaştırılmasını gerektirmektedir. Bu bakımdan diskriminant analizi, lojistik regresyon vb. istatistiksel tekniklere uygun yapay sinir ağı modellerinin belirlenmesi, onların avantajı ve dezavantajlarının açıklanması büyük önem taşımaktadır. [1,15]

3.1. Uygun Yapay Sinir Ağı ile İstatistik Terim ve Sembolleri

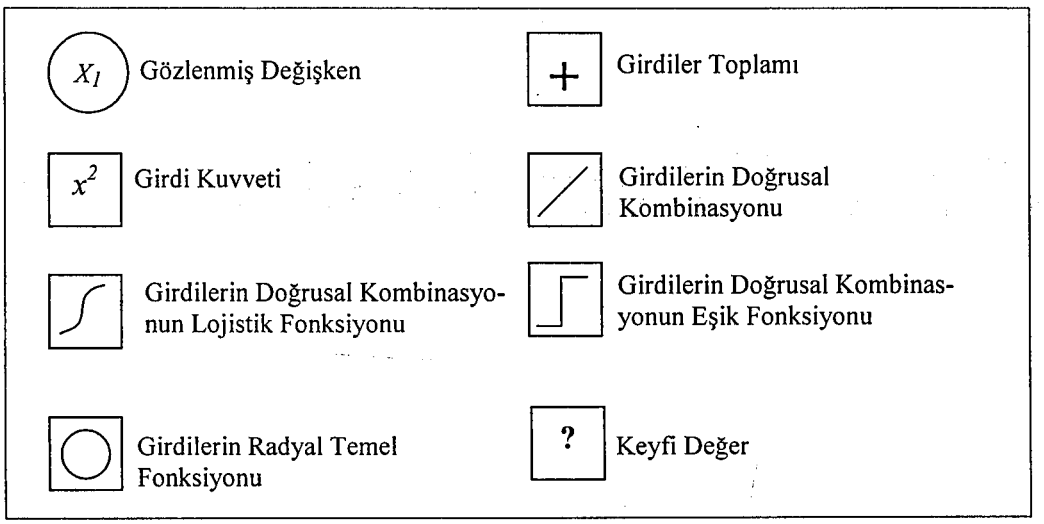
Bazı yapay sinir ağı modelleri, bazı istatistiksel tekniklere benzer ve ya aynısıdır. [14] Bu amaçla, önce adları farklı olan uygun yapay sinir ağı ve istatistik terimlerini karşılaştırmak gerekir. Aşağıdaki tabloda her iki alana uygun terimler verilmiştir:

Çizelge 3.1. Uygun yapay sinir ağı ve istatistik terimleri [15]

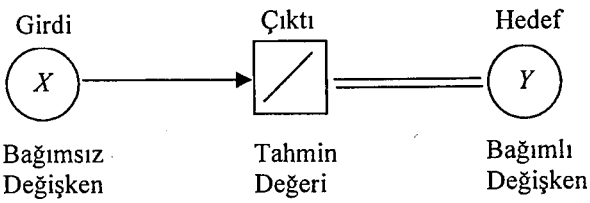
İstatistik terimleri	Yapay sinir ağı terimleri
bağımsız değişken	giriş
bağımlı değişken	hedef
artık (kalan)	hata
tahmin	öğrenme (eğitim)
tahmin ölçütü	hata fonksiyonu
gözlem	numune (örnek, eğitim çifti)
regresyon katsayıları	ağırlıklar
dönüşüm	fonksiyonel link
regresyon	kontrollü (supervised) öğrenme
veri indirme	kontrolsüz (unsupervised) öğrenme

Yapay sinir ağı modellerinin diyagram (çizgi) tasvirinde kullanılan bazı semboller çizelge 3.2.'de verilmiştir:

Çizelge 3.2. Yapay sinir ağı modellerinin diyagram tasvirinde kullanılan bazı semboller [15]



Çeşitli yapay sinir ağı ile istatistiksel modeller ağ diyagramı (çizgi) formunda gösterilebilir. Örneğin, şekil 3.1'deki diyagram basit bir doğrusal regresyonu ve ya basit bir yapay sinir ağı modelini tasvir etmektedir.



Şekil 3.1. Tek girdili ve tek çıktılı basit perseptron ve basit doğrusal regresyon modeli

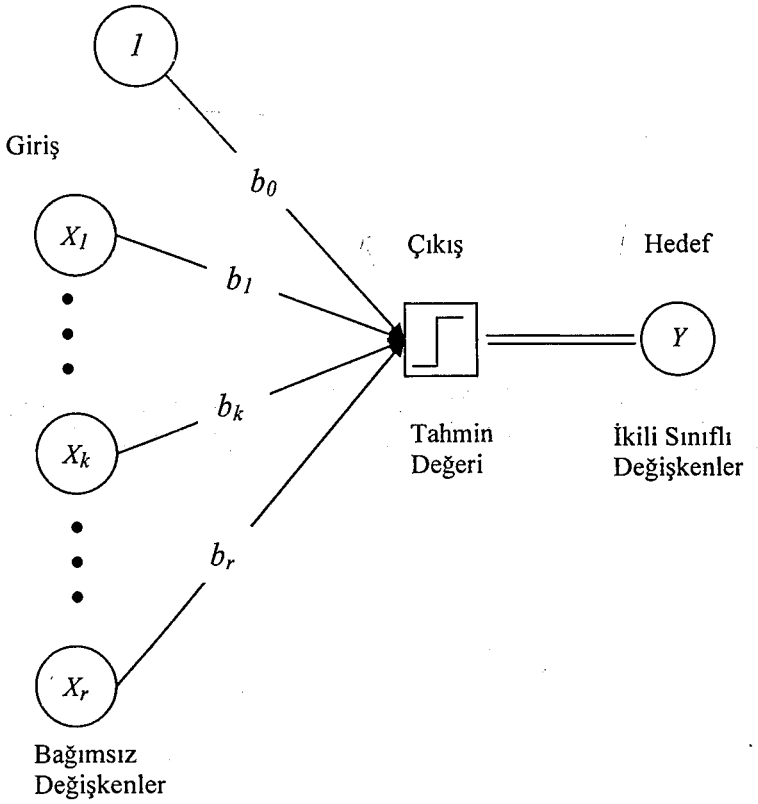
Bu diyagramlardaki sembollerin üstünde yapay sinir ağı terimleri, altında ise uygun istatistik terimleri yazılmıştır. Çemberler adı onun içinde yazılan gözlem değişkenlerini, kutular ise tek veya çok değişkenli bir (aktivasyon) fonksiyonu ifade etmektedir. Kutunun içindeki işaret ise fonksiyonun tipini göstermektedir.

3.2. Tek Katmanlı Perseptronlar ve Bazı İstatistiksel Teknikler

Bu kısımda klasik doğrusal regresyon modeller ile ilişkili olan doğrusal perseptronlar ele alınacaktır. Böyle modeller belirli girdi numunelerine uygun hedef çıktıları olan bir sürekli fonksiyonun yakınlaştırılmasına benzemektedir. Burada girdi ve çıktı değerleri sürekli değerler olmaktadır.

3.2.1. Çıktı Birimine Eşik Fonksiyonu Uygulanan ve *r* Sayıda Girdi Birimine Sahip Tek Katmanlı Perseptron ve Diskriminant Fonksiyonu

Çıktı birimine eşik (adım) aktivasyon fonksiyonu uygulanan ve *r* sayıda girdi birimine sahip tek katmanlı bir perseptron şekil 3.2.'de gösterilmiştir:



Şekil 3.2. Eşik fonksiyonlu tek katmanlı perseptron ve doğrusal diskriminant fonksiyonu

Burada çıkış birimindeki $\lrcorner\!\lrcorner$ işareti aktivasyonun eşik fonksiyonu olduğunu belirtir. Eşik fonksiyonu,

$$f(y\_in)=\begin{cases}-1\ , & y\_in<-\theta \ \text{ise}\\ 0\ , & -\theta\le y\_in\le\theta \ \text{ise}\\ 1\ , & y\_in>\theta \ \text{ise}\end{cases} \tag{3.1}$$

şeklindedir. θ eşik değeri, sıfıra eşit olursa (3.1) fonksiyonu

$$f(y\_in)=\begin{cases}-1\ , & y\_in<0 \ \text{ise}\\ 1\ , & y\_in\ge 0 \ \text{ise}\end{cases} \tag{3.2}$$

şeklini alır.

Burada y_in, Y çıktı birimine dahil olan toplam ağırlıklı sinyaldir ve

$$y\_in=b_0+\sum_{i=1}^{r} b_i\cdot x_i \tag{3.3}$$

şeklinde hesaplanır. Bu durumda y değeri ise,

$$y=f(y\_in) \tag{3.4}$$

formülü ile elde edilir.

Ağ belirli bir giriş numuneler için eğitildiği zaman $\hat{b}_o,\hat{b}_1,...,\hat{b}_r$ ağırlıkları belirlenir ve $\hat{b}_0+\sum_{i=1}^{r}\hat{b}_i\cdot x_i<\theta$ ile $\hat{b}_0+\sum_{i=1}^{r}\hat{b}_i\cdot x_i\ge\theta$ hiper yarıdüzlemleri örnekleri sınıflandırır. Başka bir ifadeyle, eğer örnek için ağ çıktı aktivasyonu "1" ise o birinci hiper yarıdüzleme (1. gruba), "-1" ise ikinci hiper yarıdüzleme (2. gruba) ait edilir. $\theta=0$ ise ayırıcı hiper düzlem $\hat{b}_0+\sum_{i=1}^{r}\hat{b}_i\cdot x_i=0$ olur. Bu işlem, istatistikteki doğrusal diskriminant rolünü gerçekleştirir. [1,15]

Bilindiği gibi gözlemlerin belirli sınıflara ayrılması, istatistikte önemli konulardan biridir. Yapay sinir ağlarının da sınıflandırma, tanıma ve ilişkilendirme problemlerinde çok etkili olabileceği gözükmektedir.

Doğrusal diskriminant yönteminde örnekler alanı hiper düzlemlerle gruplara ayrılır. Uygun hiper düzlemlerin denklemi,

$$y_i = b'_{(i)}x + b_{i0} \quad , i = 1,....,c \tag{3.5}$$

şeklindedir. Bu durumda r boyutlu karakteristik x vektörü, sayıları sınıfların sayısına eşit olan tek boyuta indirilir. Burada $c = 2$ sınıfa ayırma durumunu göz önüne alınacaktır.

Fisher'in doğrusal diskriminant yaklaşımı, r boyutlu veriyi doğru üzerinde tahminlemeyi önerir. Tahminlerin doğru üzerine sıçraması sınıflara ayırımı kolaylaştırır. Bu doğrunun kat sayıları sınıflar ayırımının "maksimum" olması hedefi ile belirlenir. $c = 2$ sınıf durumu ele alındığında, verilen eğitim örnekleri kümesi şöyledir:

$$C = \left\{x_{(1)}, x_{(2)}, ..., x_{(n)}\right\} = \left\{C_1, C_2\right\} \tag{3.6}$$

Burada C_1 alt kümesi s_1 sınıfına uygun $n_1 \leq n$ sayıda eğitim vektörünü, C_2 alt kümesi s_2 sınıfına uygun $n_2 \leq n$ sayıda eğitim vektörünü dahiline alan kümelerdir ($n_1 + n_2 = n$). Karakteristik vektör tasarımları şu şekilde yapılır:

$$y_i = b'x_{(i)} \quad , \; i = 1,2,...,n \tag{3.7}$$

Eğer $\|b\| = 1$ olarak kabul edilirse, her bir y_i yönü b vektörü olan doğru üzerinde $x_{(i)}$'in tahmini olacaktır. Bu doğru her zaman R^r'in başlangıcından geçer. İdeal durumda b vektörü öyle seçilmelidir ki, C_1 ve C_2 kümelerine uygun doğru üzerinde Y_1 ve Y_2 kümeleri için, eğer $y_i \in Y_1$ ise $x_{(i)} \in C_1$, eğer $y_i \in Y_2$ ise $x_{(i)} \in C_2$ olsun. Dolayısıyla, esas problem diskriminant fonksiyonunun parametrelerinin uygun seçilmesi problemidir.

Tahminlerin ayırımı ölçütlerinden biri onların ortalama değerlerden sapmalarıdır. C_1 ve C_2 için örneklem ortalaması şu şekilde hesaplanır:

$$m_i = \frac{1}{n_i} \sum_{x_{(i)} \in C_i} x_{(i)} \tag{3.8}$$

Her bir sınıf için tahminlerin ortalaması sabittir ve

$$a_i = \frac{1}{n_i} \sum_{x_{(i)} \in C_i} b' x_{(i)} = \frac{1}{n_i} \sum_{x_{(i)} \in C_i} y_i = b' \frac{1}{n_i} \sum_{x_{(i)} \in C_i} x_{(i)} = b' m_i \tag{3.9}$$

şeklinde hesaplanır. Tahminlerin ortalamaları farkı ise aşağıdaki gibidir:

$$|a_1 - a_2| = |b'(m_1 - m_2)| \tag{3.10}$$

İyi bir sınıflandırma için tahmin verilerinin ortalamaları farkı tek başına yeterli değildir.

İyi ayırım durumunda karışıma yol verilmemelidir. Bu amaçla Y_i'den olan y_i'lerin ortalamaya ilişkin varyansı kullanılmalıdır. Bu nedenle, en iyi sınıf ayırımı ölçütü ortalamalar farkının grup içi verilerin varyansına oranıdır. Örneğin, $c = 2$ için bu ölçüt,

$$J(b) = \frac{(a_1 - a_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{3.11}$$

şeklindedir. Burada σ_i^2 grup içi tahmin verilerinin dağıtım ölçütüdür. (3.11) formülünde kullanılan varyansların yerine tahmin verilerinin formül (3.12)'deki gibi hesaplanan grup içi varyansı kullanılmalıdır.

$$s_i^2 = \sum_{y \in Y_i} (y - a_i)^2 \tag{3.12}$$

Fisher, (3.11) formülünün

$$J(b) = \frac{(a_1 - a_2)^2}{s_1^2 + s_2^2} \tag{3.13}$$

şeklinde yazılabileceğini açıklamıştır. (3.13) formülü ile tanımlanan fonksiyona maksimum değer veren b vektörü, Fisher'in doğrusal diskriminant fonksiyonunun katsayılar vektörü olur. Sonraki süreçte (3.13) fonksiyonunu b değişkenine göre açık olarak yazmak gerekir. Bunun için bir S_i dağıtma matrisi (3.14) formülü ile tanımlanmalıdır.

$$S_i = \sum_{x \in C_i} (x - m_i)(x - m_i)' \ , \ i = 1,2 \tag{3.14}$$

$S_w = S_1 + S_2$ olarak kabul edilirse (3.13) formülünün paydası

$$s_1^2 + s_2^2 = b' S_w b \tag{3.15}$$

şeklinde olacaktır. Benzer olarak (3.13) kesrinin payı, (3.10) formülü kullanılarak

$$(a_1 - a_2)^2 = b'(m_1 - m_2)(m_1 - m_2)'b = b' S_B b \tag{3.16}$$

şeklinde yazılabilir.Burada S_B gruplar arası serpme matrisidir. Bu nedenle, (3.13) formülü

$$J(b) = \frac{b' S_B b}{b' S_B b} \tag{3.17}$$

şeklini alır. Burada S_W ve S_B'in belirlenmesi için C_1 ve C_2'de olan örnek verileri kullanılır. $\frac{\partial J}{\partial b} = 0$ olması (3.18) formülünü verir:

$$S_W \hat{b} (\hat{b}' S_B \hat{b})(\hat{b}' S_W \hat{b})^{-1} = S_B \hat{b} \tag{3.18}$$

(3.18) formülünde, $(\hat{b}' S_B \hat{b})(\hat{b}' S_W \hat{b})^{-1} = \lambda$ olarak alınırsa,

$$\lambda S_W \hat{b} = S_B \hat{b} \tag{3.19}$$

şekline dönüşür. Eğer S_W matrisinin tersi var ise,

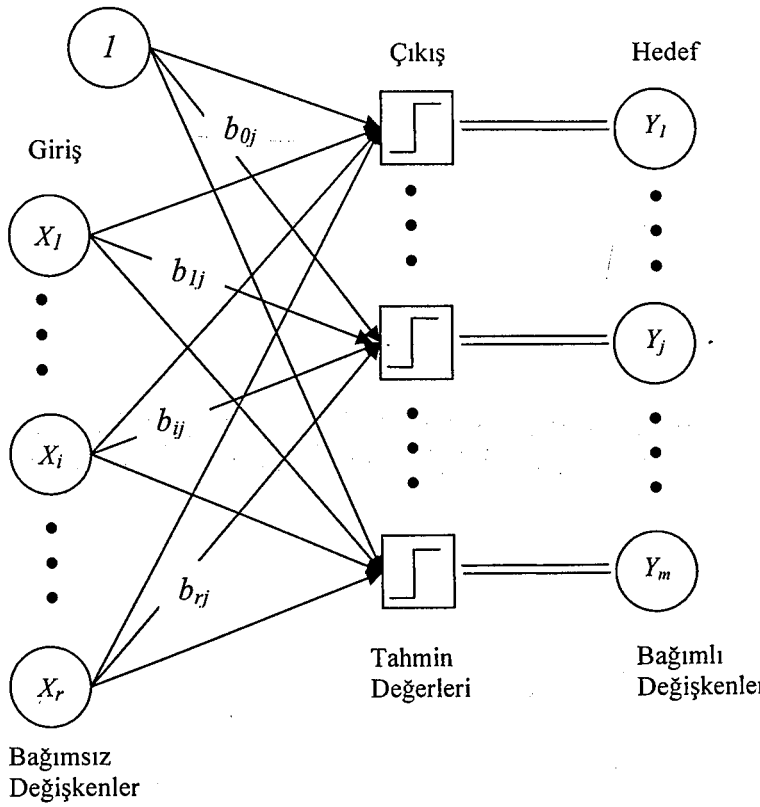
$$\lambda\hat{b} = (S_W^{-1}S_B)\hat{b} \tag{3.20}$$

olur ve $\hat{b}$ çözümü $(S_W^{-1}S_B)$ matrisinin bir öz vektörü şeklinde bulunabilir. [16]

Böylelikle, eşik fonksiyonlu perseptron ve doğrusal diskriminant fonksiyonunun her ikisi de örnekleri doğrusal ayrılabilir gibi iki sınıfa ayırır. Böyle bir problem çözülebilirse, perseptron yakınsama teoremine göre, perseptron algoritması uygun doğrusal diskriminant fonksiyonunun kat sayılarını yaklaşık olarak bulabilir. [1]

3.2.2. Birden Fazla Çıktı Birimi Olan, Çıktı Birimlerine Eşik Fonksiyonu Uygulanan ve *r* Sayıda Girdi Birimine Sahip Tek Katmanlı Perseptron ve Çoklu Diskriminant Fonksiyonu

Birden fazla çıktı birimi olan, çıktı birimlerine eşik fonksiyonu uygulanan ve *r* sayıda girdi birimine sahip tek katmanlı bir perseptron şekil 3.3.'de gösterilmiştir:



Şekil 3.3. Birden fazla çıktı birimi olan eşik fonksiyonlu tek katmanlı perseptron ve çoklu diskriminant fonksiyonu

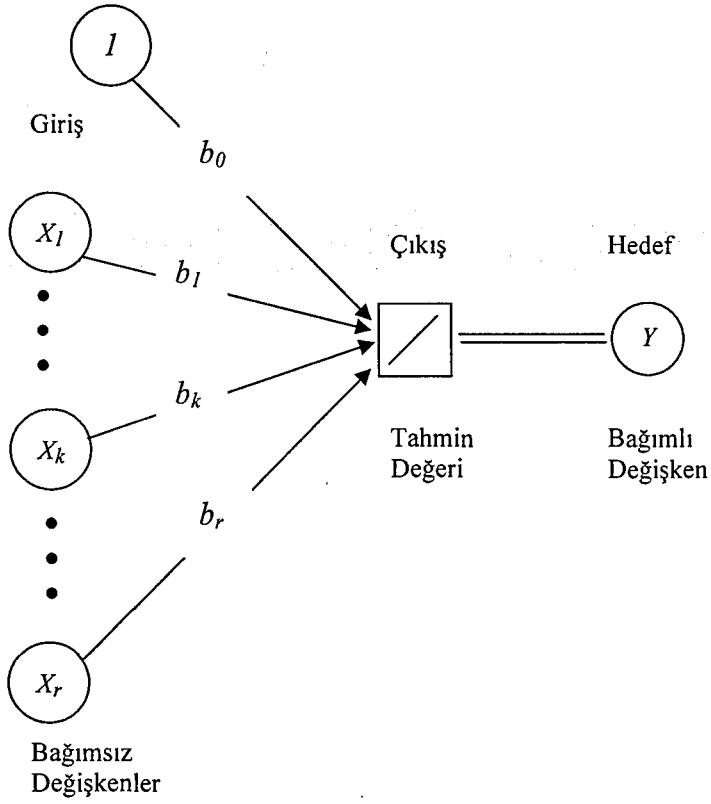
Birden fazla çıktı birimi olan eşik aktivasyon fonksiyonlu perseptron, çoklu diskriminant fonksiyonudur. [1]

Bu durumda karakteristik veriler R^r uzayından bir R^q uzayına aktarılır ($q << r$) ve sınıflandırma R^q'in yardımıyla yapılır. Burada R^q bir q boyutlu hiper düzlem gibi anlaşılabilir. Yapay sinir ağ alanında işlem zamanı, örnek ilişkilendirme ve çıktı örneklerin yardımıyla sınıflandırma yapılabilir. [16]

3.2.3. Çıktı Birimine Özdeşlik Fonksiyonu Uygulanan ve *r* Sayıda Girdi Birimine Sahip Tek Katmanlı Perseptron ve Çoklu Doğrusal Regresyon Modeli

Bu kısımda klasik doğrusal çoklu regresyon modeller ile ilişkili olan doğrusal perseptronlar ele alınacaktır. Böyle modeller belirli girdi numunelerine uygun hedef çıktıları olan bir sürekli fonksiyonun yakınlaştırılmasına benzemektedir. Burada girdi ve çıktı değerleri sürekli değerler olmaktadır.

r girdi birimi ve bir çıktı birimi olan, sapmaya sahip ve çıktı biriminin aktivasyon fonksiyonu özdeşlik fonksiyonu olan basit bir perseptron şekil 3.4.'te gösterilmiştir:



Şekil 3.4. Özdeşlik fonksiyonlu tek katmanlı perseptron ve çoklu doğrusal regresyon modeli

Bu durumda çıktı birimine dahil olan toplam ağırlıklı sinyal,

$$y\_in = b_0 + \sum_{i=1}^{r} b_i \cdot x_i \qquad (3.21)$$

formülü ile hesaplanır. Burada b_0 sapma ağırlığı, b_i , $i = 1,...,r$ *i.* girdi birimi ile çıktı birimine uygun bağlantı ağırlığıdır. Aktivasyon fonksiyonu doğrusal olduğu için, y çıktı değeri y_in'e eşit olur ve çıktı biriminin aktivasyonu

$$y = b_0 + \sum_{i=1}^{r} b_i x_i \qquad (3.22)$$

olur. Eğer ağ P sayıda $x^p = (x_{p1}, x_{p2},..., x_{pr})$; $p = 1,2,...,P$ örnek için delta kuralı ile eğitilirse, $\hat{b}_0, \hat{b}_1,..., \hat{b}_r$ ağırlıkları bulunmuş olur. Bu durumda p. örneğe uygun çıktı sinyali,

$$\hat{y}_p = \hat{b}_0 + \sum_{i=1}^{r} \hat{b}_i \cdot x_{pi} \;\; ; \;\; p = 1,2,...,P \qquad (3.22)$$

formülü ile belirlenir ve (3.22) formülü herhangi bir başka örnek için tahmin yapabilir.

Diğer taraftan (3.22) formülü, çoklu doğrusal regresyon modeli ile aynıdır. [1] Burada P gözlemler sayısı, $x^p = (x_{p1}, x_{p2}, ..., x_{pr})$ bağımsız değişkenlerin, y_p ise bağımlı değişkenin değeridir. (3.22) formülünde $\hat{y}_p$, bağımlı değişkenin $p.$ gözlem için tahmin edilen değeridir.

Bilindiği gibi $\hat{b}_0, \hat{b}_1, ..., \hat{b}_r$ katsayıları, çoklu doğrusal regresyonda en küçük kareler yöntemi ile belirlenir. Başka bir ifadeyle hata kareler toplamını minimize eden katsayıların bulunması yoluyla belirlenir. Diğer taraftan perseptron için kullanılan delta eğitim kuralında her adım için hata kareler toplamının eğiminin ters yönünde indirilmesi metodu ile bu katsayılar güncellenerek son haline getirilir. Sonuç olarak hem en küçük kareler yöntemi, hem de delta kuralı yöntemleri ağırlıkların (katsayıların) ayarlanması için hata kareler toplamının minimize edilmesi hedeflenmiştir. Böylelikle bu iki yöntem aynı anlamı taşıdığı için, tek çıktı birimine sahip doğrusal perseptronun çoklu doğrusal regresyon modeli olduğu kanıtlanabilir.

Çoklu doğrusal regresyon modellerinde katsayıların hesaplanması aşağıdaki gibi yapılmaktadır.

P sayıda bağımsız değişkene sahip çoklu doğrusal regresyon modeli,

$$Y_i = b_0 + b_1 X_{i1} + b_2 X_{i2} + ... + b_p X_{ip} + \varepsilon_i \qquad (3.23)$$

şeklindedir. *i.* (i = *1, ... ,n*) gözlem için yazılan bu çoklu regresyon modeli p sayıda bağımsız değişkenin oluşturduğu bir toplamsal modeldir. Bu modelde b_j ($j = 0,1,...p$)'ler tahmin edilecek regresyon katsayılarıdır ve dolayısıyla $p+1$ tane tahmin edilecek regresyon katsayısı vardır. Çoklu regresyon modeli matris formunda,

Y=Xb+ε

veya

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}_{nx1} = \begin{bmatrix} 1 & X_{11} & \cdots & X_{1p} \\ 1 & X_{21} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \cdots & X_{np} \end{bmatrix}_{nx(p+1)} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix}_{(p+1)x1} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}_{nx1}$$

şeklinde gösterilebilir.

Çoklu regresyon modelinin parametreleri b_j'lerin tahmini en küçük kareler yöntemi yardımıyla elde edilmektedir. En küçük kareler yönteminin esası örnek hata terimlerinin kareler toplamını minimum yapan b_j'lerin elde edilmesidir. Buna göre b vektörü aşağıdaki gibi elde edilir:

$$\begin{aligned} \sum e_i^2 &= \left(\mathbf{Y} - \hat{\mathbf{Y}}\right)' \left(\mathbf{Y} - \hat{\mathbf{Y}}\right) \\ &= \left(\mathbf{Y} - \mathbf{X}\hat{\mathbf{b}}\right)' \left(\mathbf{Y} - \mathbf{X}\hat{\mathbf{b}}\right) \\ &= \mathbf{Y}'\mathbf{Y} - 2\hat{\mathbf{b}}'\mathbf{X}'\mathbf{Y} + \hat{\mathbf{b}}'\mathbf{X}'\mathbf{X}\hat{\mathbf{b}} \end{aligned} \tag{3.24}$$

Örnek hata terimlerinin kareler toplamının b'ye göre kısmi türevi alınıp, sıfıra eşitlenirse,

$$\frac{\partial \sum e_i^2}{\partial \hat{\mathbf{b}}} = -2\mathbf{X}'\mathbf{Y} + 2\mathbf{X}'\mathbf{X}\hat{\mathbf{b}} = 0$$

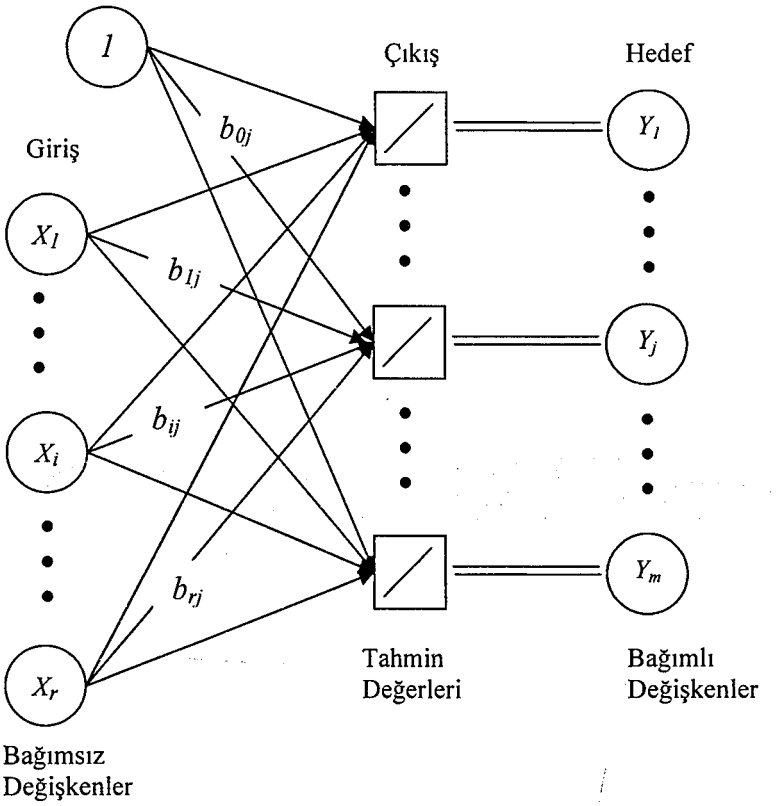
$$\mathbf{X}'\mathbf{X}\hat{\mathbf{b}} = \mathbf{X}'\mathbf{Y}$$

$$\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.25}$$

olarak elde edilir. [17-24]

3.2.4. Birden Fazla Çıktı Birimi Olan, Çıktı Birimlerine Özdeşlik Fonksiyonu Uygulanan ve *r* Sayıda Girdi Birimine Sahip Tek Katmanlı Perseptron ve Çok Değişkenli Çoklu Doğrusal Regresyon Modeli

r girdi birimine ve *m* çıktı birimine sahip, çıktı birimlerinin aktivasyon fonksiyonları özdeşlik fonksiyonu olan basit bir perseptron şekil 3.5.'te gösterilmiştir:



Şekil 3.5. Birden fazla çıktı birimi olan özdeşlik fonksiyonlu tek katmanlı perseptron ve çok değişkenli çoklu doğrusal regresyon modeli

Burada b_{0j} ($j=1,2,...,m$) sapma ağırlıkları, b_{ij} ise *i.* girdi birimi ile *j.* çıktı birimi arasındaki bağlantının ağırlığıdır.

Bu durumda (3.22) formülü

$$y_j = y\_in_j = b_{0j} + \sum_{i=1}^{r} b_{ij} \cdot x_i \;\; ; \;\; j=1,2,...,m \qquad (3.26)$$

şekline dönüşür. Burada y_j, $j.$ çıktı biriminin çıkış değerini göstermektedir. Eğer ağ, P sayıda $x^p = (x_{p1}, x_{p2}, ..., x_{pr})$, $p = 1,2,...,P$ örnekleri için eğitilirse, her bir $p.$ örnek için $y^p = (y_{p1}, y_{p2}, ..., y_{pm})$ çıktı vektörü,

$$y_{pj} = b_{0j} + \sum_{i=1}^{r} b_{ij} \cdot x_{pi} \;\; ; \;\; j = 1,2,...,m \tag{3.27}$$

formülü ile belirlenir. Bu durumda genelleştirilmiş delta kuralı ile (en küçük kareler yöntemine benzer olarak) $\hat{b}_{0j}, \hat{b}_{1j}, ..., \hat{b}_{rj}$, $j = 1,2,...,m$ ağırlıkları (kat sayıları) belirlenmiş olur. Başka bir ifadeyle (3.27) formülü,

$$Y_j = b_{0j} + \sum_{i=1}^{r} b_{ij} X_i + \varepsilon_j \;\; ; j = 1,...m \tag{3.28}$$

çok değişkenli çoklu regresyon formülüne dönüşmüş olur. [1] Burada $\left[\varepsilon_{(1)}\, \varepsilon_{(2)} \cdots \varepsilon_{(m)}\right]'$ hata terimlerinin ortalaması $E(\varepsilon) = \mathbf{0}$ ve varyansı $V(\varepsilon) = \sum$'dır.

$\left[X_{j0} \quad X_{j1} \quad \cdots \quad X_{jr}\right]$ bağımsız değişkenlerin $j.$ gözlem değerleri vektörü $\mathbf{Y}_j = \left[Y_{j1} \quad Y_{j2} \quad \cdots \quad Y_{jm}\right]'$ bağımlı değişkenlerin $j.$ gözlem değerleri vektörü ve $\varepsilon_j = \left[\varepsilon_{j1} \quad \varepsilon_{j2} \quad \cdots \quad \varepsilon_{jm}\right]'$ hata terimleri vektörü olmak üzere çok değişkenli çoklu regresyon modelini matris formunda aşağıdaki şekilde yazılabilir:

$$\mathbf{Y=XB+\varepsilon} \tag{3.29}$$

(3.29) formülünde,

$$\mathbf{X}_{[nx(r+1)]} = \begin{bmatrix} X_{10} & X_{11} & \cdots & X_{1r} \\ X_{20} & X_{21} & \cdots & X_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n0} & X_{n2} & \cdots & X_{nr} \end{bmatrix}$$

$$\mathbf{Y}_{[n\times m]}=\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1m} \\ Y_{21} & Y_{22} & \cdots & Y_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nm} \end{bmatrix}=\left[\mathbf{Y}_{(1)}\vdots\mathbf{Y}_{(2)}\vdots\cdots\vdots\mathbf{Y}_{(m)}\right]$$

$$\mathbf{B}_{[(r+1)\times m]}=\begin{bmatrix} b_{01} & b_{02} & \cdots & b_{0m} \\ b_{11} & b_{12} & \cdots & b_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \cdots & b_{rm} \end{bmatrix}=\left[\mathbf{B}_{(1)}\vdots\mathbf{B}_{(2)}\vdots\cdots\vdots\mathbf{B}_{(m)}\right]$$

$$\varepsilon_{[n\times m]}=\begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \cdots & \varepsilon_{1m} \\ \varepsilon_{21} & \varepsilon_{22} & \cdots & \varepsilon_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{n1} & \varepsilon_{n2} & \cdots & \varepsilon_{nm} \end{bmatrix}=\left[\varepsilon_{(1)}\vdots\varepsilon_{(2)}\vdots\cdots\vdots\varepsilon_{(m)}\right]=\begin{bmatrix} \varepsilon'_{(1)} \\ \cdots \\ \varepsilon'_{(2)} \\ \cdots \\ \vdots \\ \cdots \\ \varepsilon'_{(m)} \end{bmatrix}$$

olarak tanımlanmıştır.

Çok değişkenli doğrusal regresyon modelinde $E(\boldsymbol{\varepsilon}_{(i)})=\mathbf{0}$ ve $Cov(\boldsymbol{\varepsilon}_{(i)},\boldsymbol{\varepsilon}_{(j)})=\sigma_{ik}\mathbf{I}$; $i,k=1,2,...,m$'dir. j. denemede m gözlemleri $\sum=\{\sigma_{ik}\}$ kovaryans matrisini sahiptir, fakat farklı denemelerdeki gözlemlerle ilişkili değildir. Burada $\mathbf{B}$ ve σ_{ik} bilinmeyen parametrelerdir.

j. gözlemin bağımlı değişken değerleri $\mathbf{Y}_{(j)}$, $Cov(\boldsymbol{\varepsilon}_{(i)})=\sigma_{jj}\mathbf{I}$ ile aşağıdaki doğrusal regresyon modeli ile ifade edilmiş olsun.

$$\mathbf{Y}_{(j)}=\mathbf{X}\mathbf{B}_{(j)}+\varepsilon_{(j)}\ ;\ j=1,2,...,m \qquad (3.30)$$

Bununla birlikte, aynı deneme üzerindeki farklı bağımlı değişkenlerin hataları ilişkilendirilebilir. Verilmiş olan $\mathbf{Y}$ sonuçları ve tam sütün ranklı $\mathbf{X}$ bağımsız değişken değerleri ile $\hat{\mathbf{B}}_{(j)}$ en küçük kareler tahmincilerini sadece j. yanıttaki $\mathbf{Y}_{(j)}$ gözlemlerinden yararlanılarak belirlenebilir. Tek cevaplı çözüm için uygun olan katsayılar aşağıdaki gibi hesaplanır:

$$\hat{\mathbf{B}}_{(i)} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}_{(i)} \tag{3.31}$$

Bu çok değişkenli en küçük kareler tahmincilerinin toplanmasıyla,

$$\hat{\mathbf{B}} = \left[\hat{\mathbf{B}}_{(1)} \vdots \hat{\mathbf{B}}_{(2)} \vdots \cdots \vdots \hat{\mathbf{B}}_{(m)}\right] = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\left[\mathbf{Y}_{(1)} \vdots \mathbf{Y}_{(2)} \vdots \cdots \vdots \mathbf{Y}_{(m)}\right] \tag{3.32}$$

veya

$$\mathbf{B} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.33}$$

olarak elde edilir.

Herhangi bir $\mathbf{B} = \left[\mathbf{b}_{(1)} \vdots \mathbf{b}_{(2)} \vdots \cdots \vdots \mathbf{b}_{(m)}\right]$ parametresinin seçimi için hatalar matrisi $(\mathbf{Y} - \mathbf{Xb})$'dir. Hata kareler matrisi,

$$(\mathbf{Y} - \mathbf{Xb})'(\mathbf{Y} - \mathbf{Xb}) = \begin{bmatrix} (\mathbf{Y}_{(1)} - \mathbf{Xb}_{(1)})'(\mathbf{Y}_{(1)} - \mathbf{Xb}_{(1)}) & \cdots & (\mathbf{Y}_{(1)} - \mathbf{Xb}_{(1)})'(\mathbf{Y}_{(m)} - \mathbf{Xb}_{(m)}) \\ \vdots & \ddots & \vdots \\ (\mathbf{Y}_{(m)} - \mathbf{Xb}_{(m)})'(\mathbf{Y}_{(1)} - \mathbf{Xb}_{(1)}) & \cdots & (\mathbf{Y}_{(m)} - \mathbf{Xb}_{(m)})'(\mathbf{Y}_{(m)} - \mathbf{Xb}_{(m)}) \end{bmatrix} \tag{3.34}$$

eşitliği ile hesaplanır.

$\mathbf{b}_{(i)} = \hat{\mathbf{B}}_{(i)}$ olarak seçilmesi, *i.* diyagonal kareler $[(\mathbf{Y} - \mathbf{Xb})'(\mathbf{Y} - \mathbf{Xb})]$ toplamını minimum yapar. Sonuç olarak $\mathbf{B}=\hat{\mathbf{B}}$ seçilmesiyle $tr\left[(\mathbf{Y} - \mathbf{Xb})'(\mathbf{Y} - \mathbf{Xb})\right]$ minimum olur. Aynı zamanda genelleştirilmiş varyans $\left|(\mathbf{Y} - \mathbf{Xb})'(\mathbf{Y} - \mathbf{Xb})\right|$, $\hat{\mathbf{B}}$ en küçük kareler tahmincileri tarafından minimize edilir.

$\hat{\mathbf{B}}$ en küçük kareler tahmincileri matrisi kullanılarak tahmin değerleri matris formunda,

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.35}$$

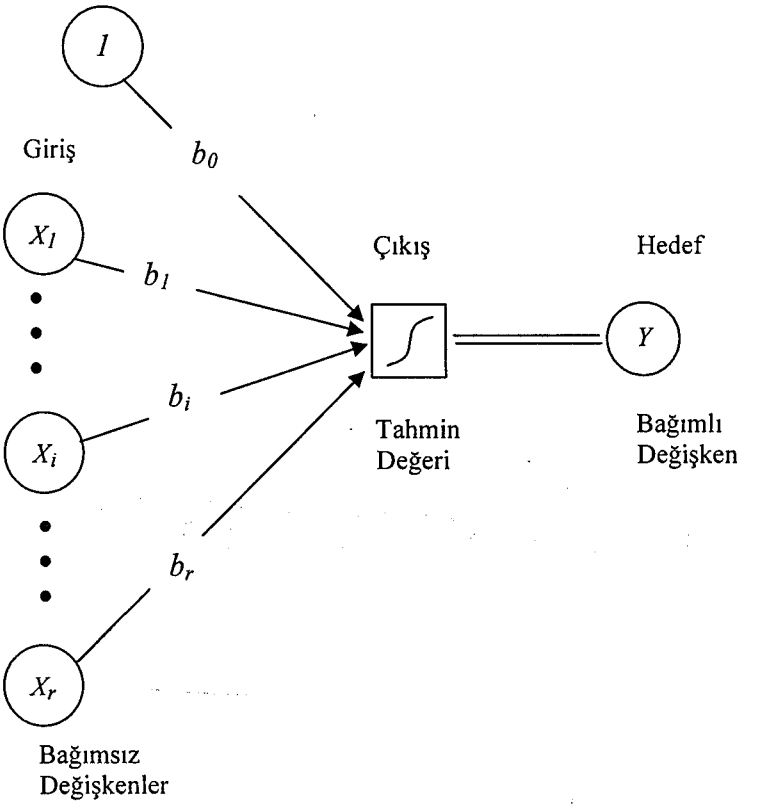
eşitliği ile, hatalar ise

$$\hat{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}} = \left[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\right]\mathbf{Y} \tag{3.36}$$

eşitliği ile elde edilir. [19,25,26]

3.2.5. Çıktı Birimine Lojistik Fonksiyon Uygulanan ve *r* Sayıda Girdi Birimine Sahip Tek Katmanlı Perseptron ve Lojistik Regresyon Modeli

r girdi birimi ve bir çıktı birimine sahip, çıktı biriminin aktivasyon fonksiyonu lojistik fonksiyonu olan basit bir perseptron şekil 3.6.'da gösterilmiştir:



Şekil 3.7. Lojistik fonksiyonlu tek katmanlı perseptron ve lojistik regresyon modeli

Bu durumda çıktı birimine dahil olan ağırlıklı sinyal,

$$y\_in = b_0 + \sum_{i=1}^{r} b_i \cdot x_i \tag{3.37}$$

şeklinde olacaktır. Bu sinyale $y = \frac{1}{1+e^{-y\_in}}$ lojistik fonksiyonu uygulanır. Başka bir ifadeyle çıkış sinyali,

$$y = \frac{1}{1+e^{-\left[b_0+\sum b_i \cdot x_i\right]}} \tag{3.38}$$

formülü ile hesaplanır. Girdi örneklerinin sayısı n ise, $x^j = (x_{j1}, x_{j2}, ..., x_{jr})$, $j = 1,2,...,n$ j. girdi örneğine uygun y_j çıkış sinyali,

$$y_j = \frac{1}{1+e^{-\left[b_0+\sum b_i \cdot x_{ji}\right]}} \; ; \; j = 1,2,...,n \tag{3.39}$$

şeklinde hesaplanır. Burada x_{ji}, $j = 1,2,...,n$; $i = 1,2,...,r$ girdi değişkenleri bağımsız, y_j, $j = 1,2,...,n$ çıktı değişkenleri ise bağımlı değişkenlerdir. (3.39) formülü ile ifade edilen model hata terimi eklenmekle lojistik regresyon modeline benzemektedir. [1] Ele alınan yapay sinir ağında verilen numunelere göre eğitim genelleştirilmiş delta kuralı ile yapılır ve uygun ağırlıklar bulunmuş olur.

Lojistik regresyon modelinde y bağımlı değişkenin ortalama değeri $E(y)$ ile ifade edilir ve

$$E(y) = \frac{1}{1+e^{-\left[b_0+\sum_{i=1}^{r} b_i x_i\right]}}$$

veya

$$E(y) = \frac{1}{1+e^{-x'b}} \tag{3.40}$$

formülleri ile hesaplanır.

y, (0,1) değerlerini alan rassal değişken olduğu için, onun beklenen ortalama değeri $E(y)$, $p(y=1) = \pi$ ihtimaline eşittir. Buna göre (3.40) formülü,

$$\pi = p(y=1) = \frac{1}{1+e^{-x'b}} \tag{3.41}$$

şeklinde yazılabilir. Burada $x' = (1, x_1, ..., x_r)$ ve $b' = (b_0, b_1, ..., b_r)$ 'dir.

Lojistik model kolaylıkla doğrusallaştırıla bilir. Bu amaçla,

$$\eta = x'b \tag{3.42}$$

olarak tanımlansın. (3.42) formülü ile tanımlanan η doğrusal tahmindir ve (3.41) formülünden

$$\eta = \ln \frac{\pi}{1-\pi} \tag{3.43}$$

elde edilir. (3.43) formülü ile verilen dönüşüm π ihtimali için logit dönüşüm olarak adlanır ve bu dönüşümdeki $\frac{\pi}{1-\pi}$ oranına "bahis oranı" denir.

Lojistik regresyon modelinin genel şekli,

$$y_j = E(y_j) + \varepsilon_j \ , \ j = 1,2,...,n \tag{3.44}$$

şeklindedir. Burada birbirlerinden bağımsız olan y_j gözlemlerinin beklenen ortalama değerleri,

$$E(y_j) = \pi_j = \frac{1}{1+e^{-x_j'b}} \tag{3.45}$$

olan Bernoulli rassal değişkenlerdir. $\eta = x'b$ doğrusal ifadesinde parametreleri tahmin etmek için maksimum olabilirlik metodu uygulanır. Her bir gözlem Bernoulli dağılımına ait olduğuna göre, onların her birinin olasılık dağılımı

$$f_j(y_j) = \pi_j^{y_j}(1-\pi_i)^{1-y_j} \ , \ j = 1,2,...,n \tag{3.46}$$

olacaktır ve her bir y_j gözlemi "0" veya "1" değerini alacaktır. Gözlemler bağımsız olduklarından, olabilirlik fonksiyonu

$$L(y_1, y_2, ..., y_n, b) = \prod_{j=1}^{n} f_j(y_j) = \prod_{j=1}^{n} \pi_j^{y_j}(1-\pi_j)^{1-y_j} \tag{3.47}$$

şeklinde yazılabilir.Logaritmalı olabilirlikle çalışmak daha uygun olduğundan, bu eşitliliğin her iki kısmının logaritması alınırsa

$$\ln L(y_1, y_2, ..., y_n, b) = \ln \prod_{j=1}^{n} f_j(y_j) = \sum_{j=1}^{n} [y_j \ln(\frac{\pi_j}{1-\pi_j})] + \sum_{j=1}^{n} \ln(1-\pi_j) \quad (3.48)$$

eşitliği elde edilir. Diğer taraftan, $1-\pi_j = [1+e^{x_j'b}]^{-1}$ ve $\eta_j = \ln[\pi_j/(1-\pi_j)] = x_j'b$ olduğu için, log-olabilirlik,

$$\ln L(y,b) = \sum_{j=1}^{n} y_j x_i' b - \sum_{j=1}^{n} \ln\left[1+e^{x'b}\right] \quad (3.49)$$

şeklinde yazılabilir.

Lojistik regresyon modelinde genellikle x değişkeninin aynı düzeyi için gözlemler veya denemeler tekrarlanır. y_j $j.$ gözlemin "1" olmasını gösterirse ve n_j böyle olayların sağlandığı denemelerin sayısı ise log-olabilirlik,

$$\ln L(y,b) = \sum_{j=1}^{n} y_j \pi_j + \sum_{j=1}^{n} n_j \ln(1-\pi_j) - \sum_{j=1}^{n} y_j \ln(1-\pi_j) \quad (3.50)$$

şeklini alır.

$\hat{b}$ maksimum olabilirlik tahmincilerinin bulunması için (3.50) formülü kullanılarak sayısal arama yöntemi kullanılır. Lojistik regresyon model için bu yöntemi işleten SAS PROC GENOD ve R For Windows gibi çok iyi istatistik paket programları mevcuttur.

$\hat{b}$ vektörünün yukarıda adı geçen algoritmanın verdiği son tahmin vektörü olduğunu varsayılır ve model için geçerli varsayımlar sağlanırsa, $E(\hat{b}) = b$ ve $Var(\hat{b}) = (X'V^{-1}X)^{-1}$ iken doğrusal tahmin edici $\hat{\eta} = x_j'\hat{b}$ olur ve lojistik regresyon modelin seçilmiş değeri genellikle

$$\hat{y}_j = \hat{\pi}_j = \frac{1}{1+e^{\hat{\eta}_j}} = \frac{1}{1+e^{-x_j'\hat{b}}} \quad (3.51)$$

şeklinde hesaplanır. [22,24,27,28]

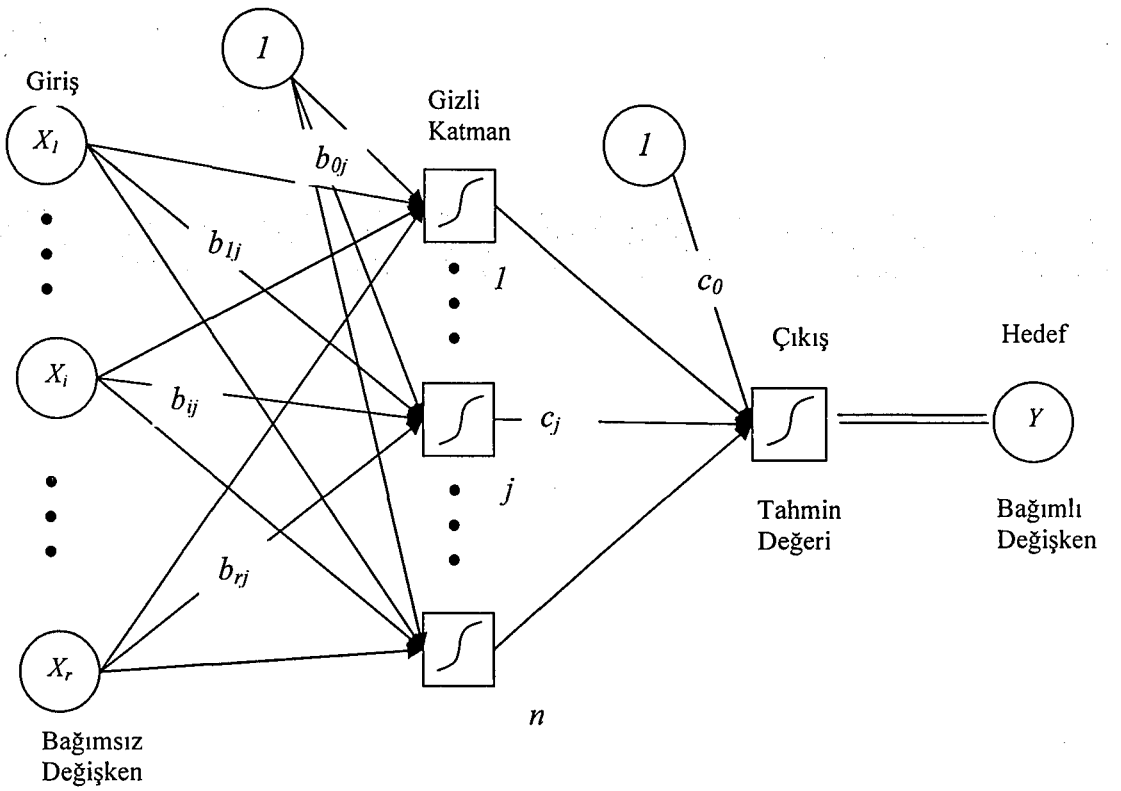
3.3. Çok Katmanlı Perseptronlar ve Bazı İstatistiksek Teknikler

Tek katmanlı yapay sinir ağlarının kısıtlı uygulama imkanları bilim adamlarını birden fazla katmana sahip ağ modellerini inceleme zorunda bırakmıştır. Önceki bölümde de değinildiği gibi, çok katmanlı yapay sinir ağları için Rumelhart, Hinton ve Williams (1986), McChelland ve Rumelhart (1988)'ın yakınsama sağlanan bir eğitim algoritmasını bulması yapay sinir ağlarına olan ilgiyi arttırmıştır. Hata kareler toplamının küçültülmesine dayanan bu eğitim metodu geriye yalılım veya genelleştirilmiş delta kuralı olarak isimlendirilmiş ve optimizasyon problemlerinde uygulanan eğim yöntemlerine dayandırılmıştır.

Çok katmanlı ileri beslemeli yapay sinir ağı modellerinde gizli katmanın birimlerine uygulanan aktivasyon fonksiyonu doğrusal olmayan bir fonksiyon olduğu için bu modeller gerçek doğrusal olmayan modellerdir. Buna göre çok katmanlı yapay sinir ağı modelleri ile yapılan yakınsamalar, öngörüler ve sınıflandırmalar daha iyi sonuçlar vermektedir. Bu bakımdan yapay sinir ağı yöntemleri uygun istatistiksel teknikler için doğrusal olmayan hesaplamalar yapma avantajını sağlamıştırlar. [1,15]

3.3.1. Çıktı Birimine Lojistik Fonksiyon Uygulanan ve *r* Sayıda Girdi Birimine Sahip Çok Katmanlı Perseptron ve Lojistik Regresyon Modeli

r girdi birimi ve bir çıktı birimine sahip, çıktı biriminin aktivasyon fonksiyonu lojistik fonksiyonu olan ve bir gizli katmana sahip iki katmanlı bir perseptron şekil 3.7.'de gösterilmiştir:



Şekil 3.7. Lojistik fonksiyonlu çok katmanlı perseptron ve lojistik regresyon modeli

Şekil 3.7.'deki iki katmanlı perseptronda, r adet girdi katmanda, bir adet çıktı katmanında ve n adet gizli katmanda nöron bulunmaktadır. i. girdi birimini j. gizli birimle bağlayan ağırlık b_{ij} $(i=1,...,r; j=1,...,n)$ ile; gizli katmanın j. birimini çıktı katmanına bağlayan ağırlık ise c_j $(j=1,...,n)$ ile gösterilmiştir. b_{0j} $(j=1,...,n)$ gizli katmana uygun, c_0 ise çıktı katmanına uygun sapma ağırlıklarıdır.

Çıktı ve gizli katmanın aktivasyon fonksiyonlarını sırasıyla $f_o(\cdot)$ ve $f_h(\cdot)$ ile gösterilsin. Bu fonksiyonlar lojistik fonksiyonlardır. Bu durumda şekil 3.6'da gösterilen ağ için, p. $x^p=(x_{p1},...,x_{pr})$; $(p=1,...,P)$ girdi örneğine uygun ağ çıktı vektörü ,

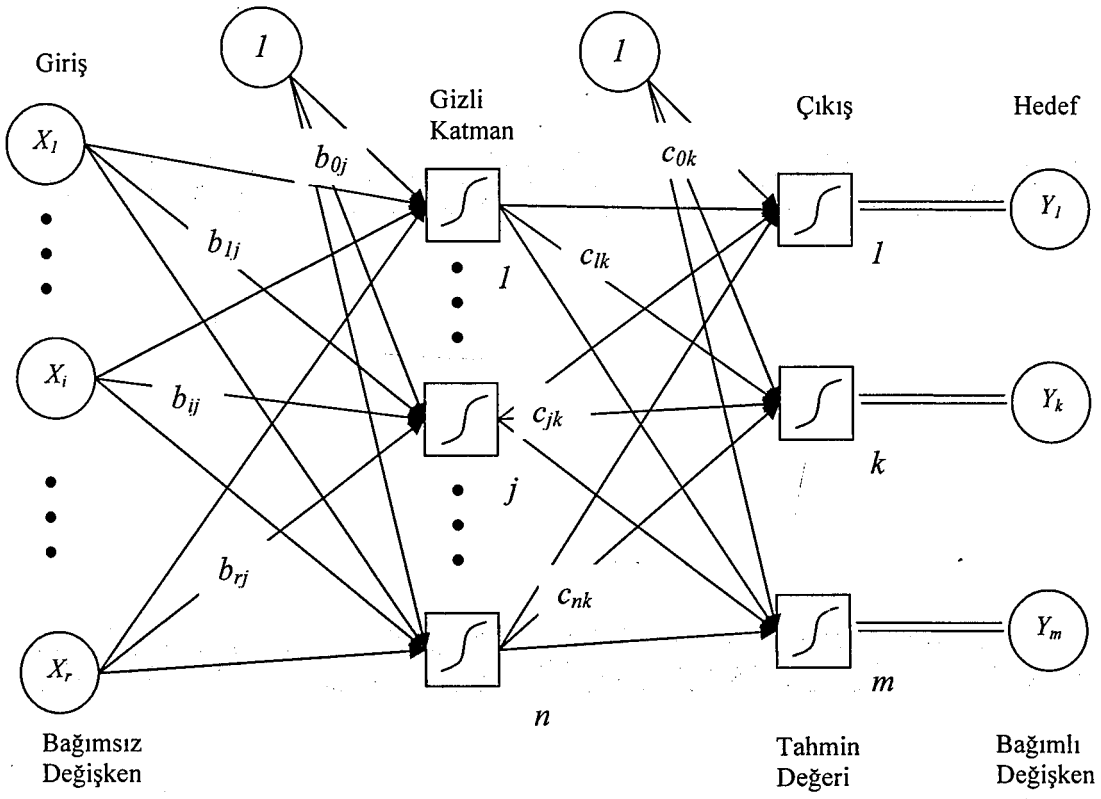
$$y_p = f_0\left[c_0 + \sum_{j=1}^{n} c_j \cdot f_h(b_{0j} + b_j x_p)\right],\ p=1,...,P \qquad (3.52)$$

formülü ile hesaplanır.

Çıktı birimlerine aktivasyon fonksiyonu olarak lojistik fonksiyon uygulanan çok katmanlı perseptronlar, istatistikteki logistik regresyon problemlerinde, çıktı birimlerine aktivasyon fonksiyonu olarak lojistik fonksiyon uygulanan tek katmanlı perseptronlardan daha iyi sonuçlar verme eğilimindedirler.

3.3.2. Birden Fazla Çıktı Birimi Olan, Çıktı Birimlerine Lojistik Fonksiyon Uygulanan, *r* Sayıda Girdi Birimi ve Bir Gizli Katmana Sahip İki Katmanlı Perseptron ve Çok Değişkenli Lojistik Regresyon Modeli

r girdi birimi ve *m* çıktı birimine sahip, çıktı birimlerinin aktivasyon fonksiyonları lojistik fonksiyonu olan ve bir gizli katmana sahip iki katmanlı bir perseptron şekil 3.8.'de gösterilmiştir:



Şekil 3.8. Birden fazla çıktı birimi olan lojistik fonksiyonlu çok katmanlı perseptron ve çok değişkenli lojistik regresyon modeli

Şekil 3.8.'deki iki katmanlı perseptronda, *r* adet girdi katmanda, *m* adet çıktı katmanında ve *n* adet gizli katmanda nöron bulunmaktadır. *i.* girdi birimini *j.* gizli birimle bağlayan ağırlık b_{ij} $(i=1,...,r; j=1,...,n)$ ile; gizli katmanın *j.*

birimini çıktı katmanının $k.$ birimiyle bağlayan ağırlık ise c_{jk} $(j=1,...,n;$ $k=1,...,m)$ ile gösterilmiştir. b_{0j} $(j=1,...,n)$ gizli katmana uygun, c_{0k} $(k=1,...,m)$ ise çıktı katmanına uygun sapma ağırlıklarıdır.

Çıktı ve gizli kamanın aktivasyon fonksiyonlarını sırasıyla $f_o(\cdot)$ ve $f_h(\cdot)$ ile gösterilsin. Bu fonksiyonlar lojistik fonksiyonlardır. Bu durumda şekil 3.7'de gösterilen ağ için, $p.$ $x^p=(x_{p1},...,x_{pr})$; $(p=1,...,P)$ girdi örneğine uygun ağ çıktı vektörü ,

$$y_p = f_0\left[c_{0k}+\sum_{j=1}^{n} c_{jk}\cdot f_h(b_{0j}+\sum_{i=1}^{r} b_{ij}x_{pi})\right],\ p=1,...,P\ ;\ k=1,...,m \qquad (3.50)$$

formülü ile hesaplanır.

(3.50) formülü, çok değişkenli çoklu lojistik regresyon modelidir. Böyle bir perseptron için eğitim algoritması olarak genelleştirilmiş delta kuralı kullanılabilir. Çok değişkenli lojistik fonksiyon her bir sınıfın koşullu olasılığını tahmin etmek için kullanılan iyi araçlardan biridir. [1]

4. UYGULAMA

Hipertansiyon basit olarak yüksek kan basıncı anlamına gelmektedir. Kan basıncı hastaya ait özellikler (yaş, cinsiyet, ırk gibi) ve fiziksel durumdan (istirahat, efor gibi) etkilenen bir değerdir. Bu nedenle de normal kan basıncı değerlerini belirlemek gerçekte oldukça güçtür. Günümüzde kabul edilen kan basıncı değeri, istirahat halindeki normal bir yetişkinde 120/80 (SKB/DKB) mmHg (milimetre civa)'dır. Herhangi bir kişide kan basıncı uyku sırasında düşük, sinirli ya da heyecanlıyken yüksektir. Genellikle de normalin üst sınırı olarak kabul edilen değer 140/90 (SKB/DKB) mmHg'dır. Kanı kalpten dokulara taşıyan damardaki kan basıncı devamlı olarak 140/90 (SKB/DKB) mmHg üzerinde seyrediyorsa hipertansiyondan bahsedilir.

Hipertansiyon kalp hastalıkları için ana bir risk faktörüdür. Eğer tedavi edilmezse beyin dolaşımı, kalp, damar ve böbrek hastalıkları için ciddi hastalık ve ölüm oranlarında artışa sebep olur. Bir kez teşhis yapılıp tedavi başlanırsa artan kan basıncı düşürülebilir, kalp ve kalp dolaşım sistemindeki hastalık riski azaltılabilir.

Sanayileşmiş ülkelerdeki yetişkin nüfusun %10-20 kadarında hipertansiyon bulunduğu belirtilmektedir. Sınırda hipertansiyon vakaları da katılırsa bu oran kuşkusuz daha yüksektir. Kişinin yaşı, cinsiyeti gibi özellikleri hipertansiyon konusunda belirleyici faktörlerdir.

Kişi yaşının hipertansiyona olan katkısı öncelikle damarlarda yaşlanmaya eşlik eden anormalliklerdir. Bu durum özellikle de kanı kalpten damarlara taşıyan damarlardaki esneklik kaybı ile açıklanabilir. Ancak yaşla hipertansiyon arasındaki bu bağlantıya bazı ilkel toplumlarda hiç rastlanmamaktadır. Bu durumda etkili faktörün "uygarlaşma" ve bununla bağlantılı yaşam biçimi olduğu söylenebilir. Örnek olarak tuz kullanımı, aşırı beslenme, sedanter yaşam (fazla hareket göstermeksizin devamlı oturuşa bağlı), stres, vs. verilebilir.

Hipertansiyon ciddi bir durumdur. Hipertansiyon, kendi başına öldürücü değildir; fakat tedavi edilmediğinde hipertansiyonun sonuçları öldürücü olabilir. Hipertansiyon kalbi zorlayarak kalp yetmezliğine neden olabilir. Üstelik ateroskleroz ve bunun yol açabileceği iskemik kalp hastalığı (belli bir bölgede kan

akımının kesilmesi nedeniyle oluşan geçici kansızlık; bölgesel anemi) riskini önemli ölçüde arttırır. Buna ilaveten, hipertansiyonlu hastalar kanama ve beyindeki kan damarlarının trombozuna (pıhtılaşma, inme) diğerlerinden daha kolay yakalanırlar. Hipertansiyon ayrıca koroner arter hastalığına da büyük katkıda bulunur ki, bu hastalık sanayileşmiş toplumlarda ölümlerin başlıca nedenlerinden biridir. Belirtilenlerin hepsi tedavi edilmeyen hipertansiyonun sonuçları olup hipertansiyona bağlı morbidite (hastalık), mortalitenin (ölüm) büyük bir bölümünü oluşturur. [29]

Bu uygulamada hipertansiyonu etkileyen risk faktörleri araştırılmıştır. Bu amaçla, hipertansiyon üzerine etkili olduğu düşünülen bazı değişkenler araştırma kapsamına alınmış, bu değişkenlerin hipertansiyon üzerine etkili olup olmadıkları ve birer risk faktörü olarak insan sağlığı üzerindeki etkileri saptanmaya çalışılmıştır.

Uygulama aşamasında Osmangazi Üniversitesi Eğitim ve Uygulama Hastanesinin İç Hastalıkları polikliniğine başvuran 225 hastaya ait veri seti kullanılmıştır. 225 hastadan elde edilen veri setinde bulunan değişkenler çizelge 4.1.'de gösterilmiştir.

Çizelge 4.1. OGÜ Eğitim ve Uygulama Hastanesi İç Hastalıkları polikliniğine başvuran 225 hastadan elde edilen veri setinde bulunan değişkenler

Y	:	Hipertansiyon	(yok ise 0, var ise 1)	
X_1	:	Yaş	$61,81 \pm 12,035$	(30 – 87)
X_2	:	Boy	$166,2933 \pm 9,63407$	(140 – 190)
X_3	:	Kilo	$72,58 \pm 13,696$	(5 – 110)
X_4	:	SKB (sistolik kan basıncı)	$147,12 \pm 31,683$	(80 – 290)
X_5	:	DKB (diastolik kan basıncı)	$86,90 \pm 15,071$	(50 – 140)
X_6	:	HDL	$45,14 \pm 13,897$	(15 – 88)
X_7	:	Total Kollesterol	$187,04 \pm 54,351$	(89 – 414)
X_8	:	Trigliserid	$108,18 \pm 70,845$	(18 – 371)
X_9	:	Albümin	$3,809 \pm 0,5847$	(2,1 – 5,0)
X_{10}	:	Glükoz	$166,55 \pm 80,660$	(70 – 440)
X_{11}	:	Cinsiyet	(erkek ise1, kadın ise 2)	
X_{12}	:	Sigara	(kullanmıyor ise 0, kullanıyor ise 1)	
X_{13}	:	Alkol	(kullanmıyor ise 0, kullanıyor ise 1)	
X_{14}	:	Egzersiz	(yapmıyor ise 0, yapıyor ise 1)	
X_{15}	:	Kalp Hastalığı	(yok ise 0, var ise 1)	

Çizelge 4.1.'de belirtilen değişkenlerden hipertansiyon değişkeni bağımlı değişken ve diğer değişkenler bağımsız değişkenler olarak ele alınmıştır. Hipertansiyon değişkeni iki kategoriye (var, yok) sahip olduğundan verilerin analizinde ikili lojistik regresyon analizi (binary logistic regression analysis) kullanılmıştır. Yapılan lojistik regresyon analizi sonuçları ile yapay sinir ağı modeli sonuçları karşılaştırılmıştır.

Lojistik regresyon analizi için SPSS 11.5, yapay sinir ağı hesaplamaları için NeuroSoulitions 4.21 paket programları kullanılmıştır.

4.1. Lojistik Regresyon Analizi ile Sınıflandırma

Hastaların hipertansiyon olan Y bağımlı değişkeni, hipertansiyonu olmayan hastalar için 0, hipertansiyonu olan hastalar için ise 1 olarak alınmıştır.

Hipotezler aşağıdaki şekilde kurulmuştur:

H_0: $b_i = 0$ $(i = 0,...,15)$

Yaş, boy, kilo SKB, DKB, HDL, total kollesterol, trigliserid, albümin, glükoz, cinsiyet, sigara, alkol, egzersiz ve kalp hastalığı değişkenlerinin hipertansiyon değişkeni üzerinde etkisi yoktur.

H_1: En az bir $b_i \neq 0$ 'dir.

Yaş, boy, kilo SKB, DKB, HDL, total kollesterol, trigliserid, albümin, glükoz, cinsiyet, sigara, alkol, egzersiz ve kalp hastalığı değişkenlerinden en az bir tanesinin hipertansiyon değişkeni üzerinde etkisi vardır.

Lojistik regresyon analizinde bağımlı değişken olarak alınan hipertansiyon değişkenini etkileyen bağımsız değişkenlerin belirlenmesi aşamasında Wald testi kullanılmıştır. $\hat{b}$ parametreleri ile bu parametrelere ilişkin Wald istatistikleri, serbestlik dereceleri, önem seviyeleri, Exp($\hat{b}$) değerleri ve güven aralıkları çizelge 4.2.'de verilmiştir.

Çizelge 4.2. Lojistik regresyon analizi sonucunda elde edilen değerler

Değişken	$\hat{b}$	Standart Hata	Wald	s.d.	p	Exp($\hat{b}$)
sabit değer	-10,577	5,119	4,269	1	0,039	0,000
X_1	0,045	0,017	7,419	1	0,006	1,046
X_2	0,029	0,030	0,958	1	0,328	1,030
X_3	0,032	0,017	3,864	1	0,049	1,033
X_4	0,000	0,009	0,003	1	0,959	1,000
X_5	0,035	0,019	3,253	1	0,071	1,035
X_6	0,000	0,015	0,001	1	0,982	1,000
X_7	0,003	0,004	0,572	1	0,449	1,003
X_8	0,000	0,003	0,000	1	0,984	1,000
X_9	-0,508	0,348	2,140	1	0,144	0,601
X_{10}	0,002	0,002	0,676	1	0,411	1,002
X_{11}	-1,744	0,538	10,509	1	0,001	0,175
X_{12}	-0,409	0,428	0,916	1	0,338	0,664
X_{13}	0,485	0,660	0,540	1	0,462	1,624
X_{14}	0,362	0,631	0,329	1	0,566	1,436
X_{15}	-0,473	0,380	1,545	1	0,214	0,623

Hosmer ve Lemeshow lojistik regresyonda Wald testi için önem seviyesi olarak 0,15 veya 0,25 kullanmayı önermişlerdir. [30,31] Çalışmada önemlilik seviyesi olarak p = 0,15 alınmıştır. Buna göre bağımlı değişkeni etkileyen değişkenler, önem seviyeleri 0,15'ten küçük olan X_1 (yaş) [p = 0,006], X_3 (kilo) [p = 0,049], X_5 (DKB) [p = 0,071], X_9 (albümin) [p = 0,144] ve X_{11} (cinsiyet) [p = 0,001] değişkenleridir.

Çizelge 4.2.'deki Exp($\hat{b}$) değerleri OR (Odds Ratio) değerleridir. Bu değer bağımlı değişkenin, bağımsız değişkenlerin etkisi ile kaç kat daha fazla ya da yüzde kaç oranda fazla gözlenme olasılığını sahip olduğunu belirtir. $\hat{b}$ katsayılarının önemliliği aynı zamanda OR = Exp($\hat{b}$)'nında önemliliği olarak değerlendirilir. [18,30]

OR değeri 1'e yakın değişkenler bağımlı değişkenin değişimine önemli bir katkıda bulunan etkenler değildir. Bu değişkenlerin katsayıları önemli değil ise "değişken önemli risk faktörü değildir" biçiminde yorumlanır. 1'den büyük OR değerleri (katsayı önemli olmak koşuluyla) değişkenin önemli bir risk faktörü olduğu yorumu yapılır. Sıfıra yakın değerler ise katsayı önemli olmak koşuluyla değişkenin önemli bir risk faktörü olduğunu fakat bağımlı değişkenin düşük değerler almasına neden olduğu negatif etkili bir faktör olduğunu belirtir.

Elde edilen OR değerlerine göre, kadınların erkeklere göre 0,175 kat daha fazla hipertansiyon hastası olma riski taşımaktadır. Ayrıca yaştaki 1 birimlik artış 1,046 kat hipertansiyon riskini arttırmaktadır. Benzer şekilde kilodaki 1 birimlik artış hipertansiyon riskini 1,033 kat, diastolik kan basıncındaki 1 birimlik artış 1,035 kat hipertansiyon riskini arttırmaktadır.

Lojistik regresyon analizi sonucunda lojistik regresyon modeli

$$\hat{Y} = \frac{1}{1+e^{-\left[\begin{array}{l}-10,577+0,045X_1+0,029X_2+0,032X_3+0,035X_5+0,003X_7-0,508X_9 \\ +0,002X_{10}-1,744X_{11}-0,409X_{12}+0,485X_{13}+0,362X_{14}-0,473X_{15}\end{array}\right]}} \qquad (4.1)$$

olarak elde edilmiştir.

Elde edilen modelin geçerliliği Hosmer Lemeshow testi ile sınanmıştır.

H_0: Tahmin denklemi anlamlıdır.

H_1: Tahmin denklemi anlamlı değildir.

Hosmer Lemeshow testi sonucunda elde edilen değerler çizelge 4.3.'de verilmiştir:

Çizelge 4.3. Hosmer Lemeshow test sonucu

Ki-kare	**s.d.**	***p***
5,599	8	0,692

Elde edilen tahmin modeli için Hosmer Lemeshow ki-kare değeri 5,599 olarak hesaplanmıştır. p = 0,692 > α = 0,05 olarak elde edilmiş ve modelin uygun olduğuna dair H_0 hipotezi kabul edilmiştir. Buna göre lojistik regresyon

sonucunda elde edilen bağımsız değişkenlerden en az birinin katsayısı sıfırdan farklı çıkmıştır. Bulunan tahmin denklemi anlamlıdır.

(4.1) nolu denklem ile sınıflandırma yapılmaktadır. Bu işlem için bağımlı değişken tahmin değerleri hesaplanarak, 0,5'ten küçük olanlara "0" değeri ve 0,5'ten büyük olanlara "1" değeri atanmıştır. Lojistik regresyon analizi sonucu elde edilen bağımlı değişken tahmin değerleri ile yapılan atamalar Ek-1'de verilmiştir. Yapılan bu atama değerleri gözlem değerleri ile karşılaştırılarak doğru sınıflandırma oranları hesaplanmıştır. Elde edilen sonuçlar çizelge 4.4.'de verilmiştir:

Çizelge 4.4. Lojistik regresyon analizi sınıflandırma tablosu

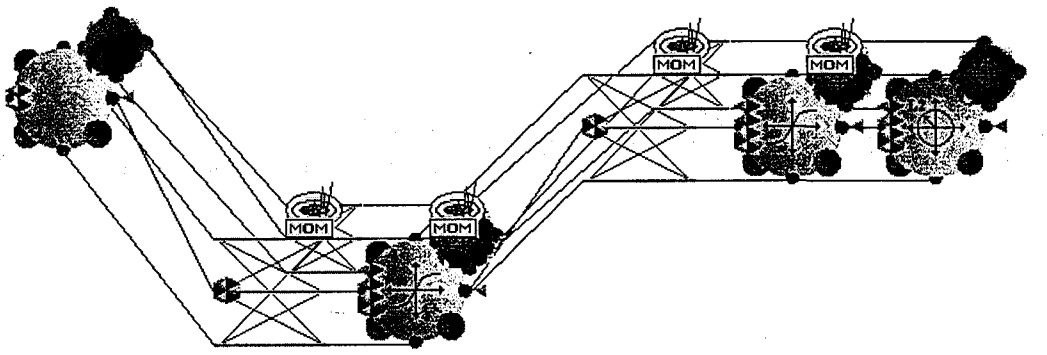
			Tahmin Değerleri			
			Hipertansiyon			
			Olmama	Olma	Toplam	**Doğruluk**
Gerçek Değerler	Hipertansiyon	Olmama	43	39	82	%52,4
		Olma	21	122	143	%85,3
Toplam			64	161	225	**%73,3**

Çizelge 4.4.'te de görüldüğü gibi, uygulamadaki 225 hastadan gerçekte 82 tanesi hipertansiyon hastası değildir. Geriye kalan 143 hasta ise hipertansiyon hastasıdır. Hipertansiyon hastası olmayan 82 hastanın 43 tanesi lojistik regresyon analizi ile yapılan sınıflandırma işleminde doğru, 39 tanesi ise hatalı olmak üzere % 52,4'lük bir doğruluk yüzdesi ile sınıflandırılmıştır. Hipertansiyon hastası olan 143 hastanın ise 122 tanesi lojistik regresyon analizi ile yapılan sınıflandırma işleminde doğru, 21 tanesi ise hatalı olmak üzere % 85,3'lik bir doğruluk yüzdesi ile sınıflandırılmıştır.

Lojistik regresyon analizi ile yapılan sınıflandırma işleminde genel doğruluk değeri ise 225 hastanın 165 tanesi doğru sınıflandırılarak % 73,3 olarak hesaplanmıştır.

4.2. Yapay Sinir Ağı Modeli ile Sınıflandırma

Uygulamada kullanılan veri setinin sınıflandırılması için girdi katmanında 15 birim, çıktı katmanında 1 birim olan ve bir gizli katmana sahip iki katmanlı bir yapay sinir ağı modeli kurulmuştur. Yapılan denemeler sonucunda en iyi doğruluk değerinin gizli katmandaki birim sayısının 7 olduğu durumda gözlendiği, gizli katmandaki birim sayısının 7'den büyük olduğu durumlarda ise doğruluk yüzdesinin azaldığı gözlenmiştir. Bu yüzden çalışmada gizli katmandaki birim sayısı 7 olarak alınmıştır. Hipertansiyon değişkeni iki kategoriye (0, 1) sahip olduğundan gizli katman ve çıktı katmanında aktivasyon fonksiyonu olarak sigmoid lojistik fonksiyonu kullanılmıştır. Kurulan model şekil 4.1.'de gösterilmiştir:



Şekil 4.1. Uygulama için NeuroSoulitions 4.21 paket programında kurulan model

Şekil 4.1.'de gösterilen yapay sinir ağında girdi katmanında 15 adet, gizli katmanda 7 adet ve çıktı katmanında 1 adet nöron bulunmaktadır. Gizli ve çıktı katmanların aktivasyon fonksiyonları olarak lojistik fonksiyon kullanılmaktadır.

Şekil 4.1.'de gösterilen yapay sinir ağı, danışmanlı öğrenme yöntemlerinden olan geriye yalılım algoritmasıyla eğitilmiştir. Eğitim esnasında momentumlu geriye yayılım eğitim algoritması ve eşlenik eğimli geriye yayılım eğitim algoritması olmak üzere faklı iki yöntem kullanılmıştır.

4.2.1. Momentumlu Geriye Yayılım Eğitim Algoritması ile Eğitilen Yapay Sinir Ağı Modeli ile Sınıflandırma

Momentumlu geriye yayılım eğitim algoritması ile ağ eğitildikten sonra elde edilen ağırlıklar çizelge 4.5.'te verilmiştir:

Çizelge 4.5. Momentumlu geriye yayılım eğitim algoritması ile ağ eğitildikten sonra elde edilen ağırlıklar

b_{ij}	Girdi Katmanı ile Gizli Katman Arasındaki Ağırlıklar						
i \ j	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0,038183	-0,377079	-0,092539	-0,844849	1,088823	0,000232	0,340129
2	-0,047652	-0,142448	-0,076438	0,206520	-1,179121	-0,472014	-0,325958
3	-0,018952	-0,426096	-0,585312	-0,207449	0,397902	0,145364	-0,154750
4	0,014908	-0,077396	-0,932752	0,427744	0,030066	-0,362468	0,054019
5	0,033981	-0,489262	0,625766	-0,324133	0,689294	0,617243	-0,005147
6	0,002287	0,120196	0,225298	0,262354	-0,404114	0,168880	0,007833
7	-0,028441	0,263126	-0,171264	0,148204	-0,341717	0,024915	-0,033381
8	-0,017017	0,508269	0,384997	-0,008662	-0,759987	-0,511327	-0,215143
9	-0,003526	-0,018455	-0,599053	-0,484052	-0,683164	-0,286301	-0,071966
10	0,004560	0,071364	-0,690455	-0,308165	1,074355	-0,147219	-0,066622
11	0,016332	-0,303747	-0,752405	0,016994	0,582696	0,530000	0,105504
12	0,021496	0,336081	0,283152	0,030311	0,555092	0,522999	0,293574
13	0,023933	0,421452	0,730063	-0,042374	-0,011385	-0,303856	0,449919
14	0,005816	0,514841	-0,918895	0,267335	-0,460008	0,719241	-0,336804
15	-0,053653	-0,375008	-0,190207	0,348217	-1,206198	-0,020581	-0,566735
c_{jk}	Gizli Katman ile Çıktı Katmanı Arasındaki Ağırlıklar						
k \ j	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0,127995	-0,836024	-1,120090	-1,601550	-1,57311	0,837367	1,375225

Ağ eğitildikten sonra elde edilen ağırlıklar kullanılarak,

$$\hat{y} = f_0\left[c_{0k} + \sum_{j=1}^{n} c_{jk} \cdot f_h(b_{0j} + \sum_{i=1}^{r} b_{ij} x_i)\right] \qquad (4.2)$$

formülü ile ağ çıktısı hesaplanır. Burada önceden belirtildiği gibi f_0 ve f_h fonksiyonları lojistik fonksiyonudur.

Ağ eğitildikten sonra kullanılan veri setine ilişkin sınıflandırma işlemi için hesaplanan $\hat{y}$ değerleri, 0,5'ten küçük olanlara "0" değeri ve 0,5'ten büyük olanlara "1" değeri verilerek atanmaktadır. Elde edilen $\hat{y}$ değerleri ile yapılan atamalar Ek-2'de verilmiştir. Yapılan bu atama değerleri gözlem değerleri ile karşılaştırılarak doğru sınıflandırma oranları hesaplanmıştır. Ağ eğitildikten sonra kullanılan veri setine ilişkin sınıflandırma işleminde elde edilen doğruluk yüzdeleri çizelge 4.6.'da verilmiştir:

Çizelge 4.6. Momentumlu geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli sınıflandırma tablosu

			Tahmin Değerleri			
			Hipertansiyon			
			Olmama	Olma	Toplam	**Doğruluk**
Gerçek Değerler	Hipertansiyon	Olmama	63	19	82	%76,8
		Olma	12	131	143	%91,6
Toplam			75	150	225	**%86,2**

Çizelge 4.6.'da de görüldüğü gibi, uygulamadaki 225 hastadan gerçekte 82 tanesi hipertansiyon hastası değildir. Geriye kalan 143 hasta ise hipertansiyon hastasıdır. Hipertansiyon hastası olmayan 82 hastanın 63 tanesi yapay sinir ağı modeli ile yapılan sınıflandırma işleminde doğru, 19 tanesi ise hatalı olmak üzere % 76,8'lik bir doğruluk yüzdesi ile sınıflandırılmıştır. Hipertansiyon hastası olan 143 hastanın ise 131 tanesi momentumlu geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli ile yapılan sınıflandırma işleminde doğru, 12 tanesi ise hatalı olmak üzere % 91,6'lık bir doğruluk yüzdesi ile sınıflandırılmıştır.

Momentumlu geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli ile yapılan sınıflandırma işleminde genel doğruluk değeri 225 hastanın 194 tanesi doğru sınıflandırılarak % 86,2 olarak hesaplanmıştır.

4.2.2. Eşlenik Eğimli Geriye Yayılım Eğitim Algoritması ile Eğitilen Yapay Sinir Ağı Modeli ile Sınıflandırma

Eşlenik eğimli geriye yayılım eğitim algoritması ile ağ eğitildikten sonra elde edilen ağırlıklar çizelge 4.7.'de verilmiştir:

Çizelge 4.7. Eşlenik eğimli geriye yayılım eğitim algoritması ile ağ eğitildikten sonra elde edilen ağırlıklar

b_{ij}	Girdi Katmanı ile Gizli Katman Arasındaki Ağırlıklar						
i \ j	1	2	3	4	5	6	7
1	0,50044	-1,730519	-0,432385	-2,212496	-0,310118	-0,536704	-0,797588
2	-0,089624	1,59597	-1,718343	0,194532	-3,885706	-1,279213	-1,481379
3	0,073586	0,176147	-2,741652	0,638694	-1,827383	-0,964285	-2,103983
4	-0,650837	-0,115309	-3,471644	0,639703	0,244935	-0,653154	-1,481175
5	-0,192319	-0,979086	1,14439	1,29368	-0,984991	0,506407	-1,323641
6	0,119219	1,328376	-0,461388	0,72299	-1,144382	0,045564	-0,932604
7	-0,597068	0,609235	-2,262588	-0,§68023	0,341118	O,071685	-0,917533
8	-0,831742	-0,473473	0,386028	0,55305	-0,576316	-1,043114	-1,102681
9	0,520319	-1,427755	-2,942906	-0,360444	-1,911586	-0,921742	-1,922553
10	-0,264529	0,101774	-2,476718	-1,508757	0,731404	-1,119844	-0,99558
11	0,088826	0,331771	-4,073246	-0,216949	0,621034	-0,121464	-2,126909
12	-0,897644	0,109378	-0,875451	-0,716155	1,556789	-0,189914	-1,367429
13	-0,307702	-1,136922	3,832603	-1,293699	0,138086	-1,617161	-0,455787
14	-1,509156	0,012926	-2,93491	0,230133	-2,5357	-0,735346	-2,920688
15	0,454645	1,265555	0,10662	0,764483	-1,667776	-0,019351	-1,419592

c_{jk}	Gizli Katman ile Çıktı Katmanı Arasındaki Ağırlıklar						
k \ j	1	2	3	4	5	6	7
1	1,180347	-2,461950	-5,509069	-2,862042	-4,694672	0,832300	-1,448453

Ağ eğitildikten sonra elde edilen ağırlıklar kullanılarak (4.2) formülü ile ağ çıktısı hesaplanır. Momentumlu geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modelinde olduğu gibi, ağ eğitildikten sonra kullanılan veri setine ilişkin sınıflandırma işlemi için hesaplanan $\hat{y}$ değerleri, 0,5'ten küçük olanlara "0" değeri ve 0,5'ten büyük olanlara "1" değeri verilerek atanmaktadır. Eşlenik eğimli geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli ile elde edilen $\hat{y}$ değerleri ile yapılan atamalar Ek-3'de verilmiştir. Yapılan bu atama değerleri gözlem değerleri ile karşılaştırılarak doğru sınıflandırma oranları

hesaplanmıştır. Ağ eğitildikten sonra kullanılan veri setine ilişkin sınıflandırma işleminde elde edilen doğruluk yüzdeleri çizelge 4.8.'de verilmiştir:

Çizelge 4.8. Eşlenik eğimli geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli sınıflandırma tablosu

			Tahmin Değerleri			
			Hipertansiyon			
			Olmama	Olma	Toplam	**Doğruluk**
Gerçek Değerler	Hipertansiyon	Olmama	60	22	82	%73,1
		Olma	4	139	143	%97,2
Toplam			75	150	225	**%88,4**

Çizelge 4.8.'de de görüldüğü gibi, uygulamadaki 225 hastadan gerçekte 82 tanesi hipertansiyon hastası değildir. Geriye kalan 143 hasta ise hipertansiyon hastasıdır. Hipertansiyon hastası olmayan 82 hastanın 60 tanesi yapay sinir ağı modeli ile yapılan sınıflandırma işleminde doğru, 22 tanesi ise hatalı olmak üzere % 73,1'lik bir doğruluk yüzdesi ile sınıflandırılmıştır. Hipertansiyon hastası olan 143 hastanın ise 139 tanesi eşlenik eğimli geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli ile yapılan sınıflandırma işleminde doğru, 4 tanesi ise hatalı olmak üzere % 97,2'lik bir doğruluk yüzdesi ile sınıflandırılmıştır.

Eşlenik eğimli geriye yayılım eğitim algoritması ile eğitilen yapay sinir ağı modeli ile yapılan sınıflandırma işleminde genel doğruluk değeri 225 hastanın 199 tanesi doğru sınıflandırılarak % 88,4 olarak hesaplanmıştır.

5. SONUÇ VE ÖNERİLER

Son yıllarda yapay sinir ağı modelleri araştırmalarda sıklıkla kullanılmaktadır. Bu çalışmada yapay sinir ağı modellerine değinilmiş, bazı istatistiksel tekniklerle bazı yapay sinir ağı modelleri arasındaki ilişki incelenmiş ve sağlık alanından elde edilmiş bir veri seti üzerinde uygulama yapılmıştır. Bu veri seti, önce lojistik regresyon analizi, daha sonra ise yapay sinir ağı modelleriyle çözümlenmiştir. Çözümleme sonrasında yapay sinir ağı modellerinin sınıflandırma problemlerindeki etkinliği araştırılmıştır.

Osmangazi Üniversitesi Eğitim ve Uygulama Hastanesinin İç Hastalıkları polikliniğine başvuran 225 hastaya ait veri seti önce lojistik regresyon analizi, daha sonra da momentumlu ve eşlenik eğimli geriye yayılım eğitim algoritmaları ile eğitilen yapay sinir ağı modelleri ile sınıflandırılmıştır. Uygulama sonucunda elde edilen doğruluk yüzdeleri çizelge 5.1.'de verilmiştir.

Çizelge 5.1. Uygulama sonucunda elde edilen doğruluk yüzdeleri

	Hipertansiyon		
	Olmama	Olma	
	Doğruluk	Doğruluk	**Genel**
Lojistik Regresyon Analizi	% 52,4	% 85,3	**% 73,3**
Momentumlu Geriye Yayılım Eğitim Algoritmasıyla Eğitilen Yapay Sinir Ağı Modeli	% 76,8	% 91,6	**% 86,2**
Eşlenik Eğimli Geriye Yayılım Eğitim Algoritmasıyla Eğitilen Yapay Sinir Ağı Modeli	% 73,1	% 97,2	**% 88,4**

Çizelge 5.1. incelendiğinde verilerin doğru sınıflandırılmasındaki genel doğruluk yüzdeleri sırasıyla lojistik regresyon analizi için % 73,3, momentumlu geriye yayılım eğitim algoritmasıyla eğitilen yapay sinir ağı modeli için % 86,2 ve eşlenik eğimli geriye yayılım eğitim algoritmasıyla eğitilen yapay sinir ağı modeli için ise % 88,4 olarak hesaplandığı görülmektedir.

Bu sonuçlara göre yapay sinir ağı modelleri ile sınıflandırmanın lojistik regresyon analizinde gerçekleştirilen sınıflandırmadan daha iyi sonuçlar verme eğiliminde olduğu görülmektedir.

Yapay sinir ağları farklı istatistiksel işlemlerin yerine getirilmesinde önemli bir rol oynamaktadır. Yapay sinir ağı yöntemlerinin genellikle doğrusal olmaması, istatistiksel modellerde, tahmin analizlerinde, sınıflandırma problemlerinde önemini yükseltmektedir.

KAYNAKLAR

1. MEMMEDOV, M., ERYILMAZ H., *Yapay Sinir Ağları ile Bazı İstatistiksel Modeller Arasındaki İlişki*, International XII. Turkish Symposium on Artificial Intelligence and Neural Network, (2003).

2. http://www.ad.com.tr/mitoloji/insan.html.

3. HALAÇ, A., ERBİL T. ve FALAY, T., *İnsan Zekasına Sayısal Rakip Yapay Zeka*, PC Net, Sayı:59, Ağustos, (2002).

4. ELMAS, Ç., *Yapay Sinir Ağları*, Seçkin Yayıncılık, Ankara, (2003).

5. FAUSETT, L.V., *Fundamentals of Neural Networks*, Printice-Hall, Inc., New Jersey, (1994).

6. AKPINAR, H., *Yapay Sinir Ağları ve Kredi Taleplerinin Değerlendirilmesinde Bir Uygulama Önerisi*, İstanbul Üniversitesi İşletme Fakültesi Sayısal Yöntemler Anabilim Dalı, İstanbul, (1993).

7. FREEMANN, J.A. ve SKAPURA, D.M., *Neural Networks: Algorithms, Applications and Programming Techniques,* Addison-Wesley Publishing Company, Inc., New York, (1991).

8. YILDIZ, B., *Finansal Başarısızlığın Öngörülmesinde Yapay Sinir Ağı Kullanımı ve Ampirik Bir Çalışma*, Doktora Tezi, Dumlupınar Üniversitesi Sosyal Bilimler Enstitüsü, Kütahya, (1999).

9. ÖZTEMEL, E., *Yapay Sinir Ağları*, Papatya Yayıncılık, İstanbul, (2003).

10. HERTZ, J., KROGH A. ve PALMER R.G., *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, Inc., New York, (1993).

11. CARLING, A., *Introducing Neural Networks*, Sigma Press, United Kingdom, (1992).

12. WANG, S., *An Adaptive Approach to Market Development Forecasting*, Neural Computing & Applications **8**, 3-8, (1999).

13. NeuroDimension, Inc., *Neurosolutions Getting Started Manuel Version 4*, (2003).

14. ANDERS, U., *Statistical Model Building for Neural Networks*, 6th AFIR Colloquium,963-979(1996).
http://www.actuaries.org/members/en/AFIR/colloquia/Nuernberg/Anders.pdf

15. WARREN, S.S., *Neural Networks and Statistical Models*, Proceeding of the Nineteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute, 1538-1550, (1994).

16. SCHALKOFF, R.J., *Pattern Recognition*, J. Wiley, New York, (1992).

17. AĞAOĞLU, E., *Uygulamalı Örnekleme*, Anadolu Üniversitesi Fen Fakültesi Yayınları No.11, Eskişehir, (1998).

18. ÖZDAMAR, K., *Paket Programlar ile İstatistiksel Veri Analizi-1*, 4. Baskı, Kaan Kitabevi, Eskişehir, (2002).

19. ÖZDAMAR, K., *Paket Programlar ile İstatistiksel Veri Analizi-2*, 2. Baskı, Kaan Kitabevi, Eskişehir, (1999).

20. ŞIKLAR, E., *Regresyon Analizine Giriş*, Anadolu Üniversitesi Fen Fakültesi Yayınları No.16, Eskişehir, (2000).

21. ERAR, A., *Bağlanım (Regresyon) Çözümlemesi Lisansüstü Ders Notları*, Hacettepe Üniversitesi İstatistik Bölümü, Ankara, (1985).

22. RAWLINGS, J.O., *Applied Regression Analysis: A research Tool*, Wadsworth, Inc., California, (1988).

23. AKKAYA, Ş. ve PAZARLIOĞLU, M.V., *Ekonometri I*, 4. Baskı, Anadolu Matbaacılık, İzmir, (2002).

24. MONTGOMERY, D.C., *Introduction to Linear Regression Analysis*, 3rd Edition, J.Wiley, New York, (2001).

25. TATLIDİL, H., *Uygulamalı Çok Değişkenli İstatistiksel Analiz*, Akademi Matbaası, Ankara, (1996).

26. ALPAR, R., *Uygulamalı Çok Değişkenli İstatistiksel Yöntemlere Giriş-I*, Bağırgan Yayımevi, Ankara, (1997).

27. AKKAYA, Ş. ve PAZARLIOĞLU, M.V., *Ekonometri II*, 2. Baskı, Erkam Matbaacılık, İstanbul, (1998).

28. YÜZER, A.F., *Olasılık ve İstatistik*, Anadolu Üniversitesi Fen Fakültesi Yayınları No.5, Eskişehir, (1996).

29. http://www.tansiyon.gen.tr/tanitim.asp.

30. GÜNERİ, N., *Öğrenci Başarısızlıklarının Analizinde Sinir Ağları Yaklaşımının Lojistik Regresyon Analizi ile Karşılaştırılması*, Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara, (2001).

31. ADIGÜZEL, F., *Neural Networks As a Statistical Tool*, Master Thesis, Middle East Technical University, Ankara, (2001).

32. BISHOP, C.M., *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford, (1995).

EK –1 Lojistik Regresyon Analizi Sonucu Elde Edilen Bağımlı Değişken Tahmin Değerleri ile Atamalar

Sıra	Y	$\hat{Y}$		Sıra	Y	$\hat{Y}$		Sıra	Y	$\hat{Y}$	
1	1	0,78226	1	**41**	1	0,41163	0	**81**	0	0,75472	1
2	1	0,75039	1	**42**	1	0,65597	1	**82**	1	0,70999	1
3	1	0,85439	1	**43**	1	0,86187	1	**83**	0	0,24783	0
4	1	0,44463	0	**44**	1	0,9046	1	**84**	0	0,53928	1
5	1	0,64565	1	**45**	1	0,85317	1	**85**	1	0,91062	1
6	1	0,88828	1	**46**	1	0,89477	1	**86**	1	0,90777	1
7	1	0,89379	1	**47**	1	0,17029	0	**87**	0	0,50997	1
8	1	0,79706	1	**48**	1	0,80771	1	**88**	0	0,88311	1
9	1	0,79013	1	**49**	1	0,93604	1	**89**	0	0,30516	0
10	0	0,87065	1	**50**	1	0,68904	1	**90**	0	0,89867	1
11	1	0,66148	1	**51**	1	0,84898	1	**91**	0	0,30055	0
12	1	0,76689	1	**52**	0	0,49742	0	**92**	0	0,74864	1
13	1	0,92581	1	**53**	1	0,32942	0	**93**	0	0,5705	1
14	1	0,62821	1	**54**	0	0,70958	1	**94**	1	0,85881	1
15	1	0,67387	1	**55**	1	0,71827	1	**95**	1	0,53063	1
16	1	0,23971	0	**56**	1	0,75074	1	**96**	1	0,64264	1
17	1	0,61692	1	**57**	1	0,94175	1	**97**	1	0,8187	1
18	0	0,81657	1	**58**	0	0,26009	0	**98**	1	0,54846	1
19	0	0,75253	1	**59**	1	0,88938	1	**99**	0	0,46773	0
20	1	0,86523	1	**60**	0	0,5639	1	**100**	1	0,4218	0
21	0	0,52543	1	**61**	1	0,96095	1	**101**	0	0,62076	1
22	0	0,50161	1	**62**	1	0,8271	1	**102**	1	0,73832	1
23	1	0,84698	1	**63**	1	0,74062	1	**103**	0	0,51882	1
24	0	0,84892	1	**64**	0	0,51316	1	**104**	1	0,65851	1
25	0	0,95924	1	**65**	0	0,37738	0	**105**	1	0,59354	1
26	1	0,91942	1	**66**	1	0,79484	1	**106**	0	0,13298	0
27	1	0,5612	1	**67**	1	0,87268	1	**107**	0	0,44301	0
28	1	0,72868	1	**68**	1	0,71068	1	**108**	1	0,4748	0
29	0	0,27847	0	**69**	1	0,84701	1	**109**	1	0,81633	1
30	0	0,20491	0	**70**	1	0,70056	1	**110**	1	0,84301	1
31	1	0,73022	1	**71**	1	0,35894	0	**111**	1	0,31609	0
32	0	0,43631	0	**72**	0	0,80048	1	**112**	1	0,7857	1
33	1	0,76539	1	**73**	1	0,85006	1	**113**	0	0,85804	1
34	1	0,53206	1	**74**	1	0,31636	0	**114**	0	0,36166	0
35	0	0,76562	1	**75**	1	0,74131	1	**115**	1	0,36823	0
36	0	0,57938	1	**76**	0	0,43533	0	**116**	1	0,41582	0
37	0	0,38532	0	**77**	1	0,7015	1	**117**	0	0,54933	1
38	1	0,55003	1	**78**	1	0,39067	0	**118**	0	0,64545	1
39	1	0,97481	1	**79**	1	0,9435	1	**119**	0	0,14902	0
40	1	0,71451	1	**80**	1	0,90817	1	**120**	1	0,68122	1

EK –1 Devam Lojistik Regresyon Analizi Sonucu Elde Edilen Bağımlı Değişken Tahmin Değerleri ile Atamalar

Sıra	Y	$\hat{Y}$		Sıra	Y	$\hat{Y}$		Sıra	Y	$\hat{Y}$	
121	1	0,86972	1	**161**	1	0,97168	1	**201**	0	0,3295	0
122	1	0,82511	1	**162**	0	0,62992	1	**202**	0	0,25405	0
123	0	0,22058	0	**163**	1	0,89018	1	**203**	0	0,47917	0
124	0	0,28913	0	**164**	1	0,78489	1	**204**	0	0,28045	0
125	1	0,82419	1	**165**	1	0,88328	1	**205**	0	0,41036	0
126	0	0,25567	0	**166**	1	0,2699	0	**206**	0	0,2405	0
127	1	0,72526	1	**167**	1	0,59137	1	**207**	1	0,94473	1
128	1	0,72688	1	**168**	1	0,85938	1	**208**	1	0,74341	1
129	1	0,72147	1	**169**	0	0,33989	0	**209**	1	0,78167	1
130	1	0,48934	0	**170**	1	0,83314	1	**210**	0	0,47813	0
131	1	0,98558	1	**171**	1	0,50473	1	**211**	0	0,34836	0
132	0	0,59556	1	**172**	1	0,84089	1	**212**	0	0,77928	1
133	0	0,08056	0	**173**	1	0,79131	1	**213**	1	0,51796	1
134	0	0,59799	1	**174**	1	0,75819	1	**214**	0	0,05502	0
135	1	0,64456	1	**175**	1	0,94718	1	**215**	1	0,66835	1
136	1	0,93846	1	**176**	1	0,67186	1	**216**	0	0,33178	0
137	1	0,85631	1	**177**	1	0,90043	1	**217**	0	0,55793	1
138	1	0,93344	1	**178**	1	0,96813	1	**218**	0	0,17121	0
139	1	0,85896	1	**179**	1	0,37301	0	**219**	0	0,63025	1
140	1	0,86547	1	**180**	1	0,74048	1	**220**	0	0,31543	0
141	1	0,94381	1	**181**	1	0,54939	1	**221**	0	0,58117	1
142	0	0,86574	1	**182**	0	0,56748	1	**222**	1	0,511	1
143	1	0,85754	1	**183**	0	0,79693	1	**223**	1	0,78332	1
144	0	0,17218	0	**184**	1	0,84658	1	**224**	1	0,79006	1
145	1	0,83053	1	**185**	0	0,41528	0	**225**	0	0,32863	0
146	1	0,6488	1	**186**	1	0,55353	1				
147	1	0,94037	1	**187**	0	0,48529	0				
148	1	0,9561	1	**188**	0	0,41682	0				
149	1	0,3922	0	**189**	1	0,48357	0				
150	1	0,81001	1	**190**	1	0,91322	1				
151	0	0,4497	0	**191**	0	0,69305	1				
152	1	0,91743	1	**192**	0	0,41275	0				
153	1	0,6793	1	**193**	1	0,54408	1				
154	1	0,51885	1	**194**	0	0,63679	1				
155	0	0,63225	1	**195**	0	0,24192	0				
156	1	0,73558	1	**196**	0	0,09388	0				
157	1	0,94336	1	**197**	1	0,19835	0				
158	1	0,88919	1	**198**	1	0,30839	0				
159	1	0,39834	0	**199**	0	0,13619	0				
160	1	0,87258	1	**200**	0	0,58173	1				

EK –2 Momentumlu Geriye Yayılım Eğitim Algoritması ile Eğitilen Ağın Çıktıları ve Atamalar

Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$	
1	1	0,75657	1	**41**	1	0,48678	0	**81**	0	0,9268	1
2	1	0,88074	1	**42**	1	0,7472	1	**82**	1	0,80365	1
3	1	0,86441	1	**43**	1	0,83849	1	**83**	0	-0,0059	0
4	1	0,80481	1	**44**	1	1,02835	1	**84**	0	0,31543	0
5	1	0,88585	1	**45**	1	1,02271	1	**85**	1	0,97177	1
6	1	0,98522	1	**46**	1	1,01741	1	**86**	1	0,97321	1
7	1	0,99579	1	**47**	1	0,51256	1	**87**	0	0,06009	0
8	1	0,85117	1	**48**	1	0,93882	1	**88**	0	0,98611	1
9	1	0,93798	1	**49**	1	1,02163	1	**89**	0	0,17024	0
10	0	0,97859	1	**50**	1	0,86781	1	**90**	0	0,49488	0
11	1	0,79842	1	**51**	1	0,95324	1	**91**	0	0,41253	0
12	1	0,70512	1	**52**	0	0,33447	0	**92**	0	0,33855	0
13	1	0,89558	1	**53**	1	0,72896	1	**93**	0	0,27372	0
14	1	0,68009	1	**54**	0	0,25512	0	**94**	1	0,99763	1
15	1	0,72716	1	**55**	1	0,87398	1	**95**	1	0,8808	1
16	1	0,34008	0	**56**	1	0,88375	1	**96**	1	0,83174	1
17	1	0,76863	1	**57**	1	1,04224	1	**97**	1	0,88501	1
18	0	0,43335	0	**58**	0	0,40587	0	**98**	1	0,86427	1
19	0	0,61756	1	**59**	1	0,97907	1	**99**	0	0,34691	0
20	1	0,62142	1	**60**	0	0,51718	1	**100**	1	0,97873	1
21	0	0,15823	0	**61**	1	1,03965	1	**101**	0	0,13822	0
22	0	0,37199	0	**62**	1	0,99157	1	**102**	1	0,95971	1
23	1	0,97454	1	**63**	1	0,85089	1	**103**	0	0,21561	0
24	0	0,51592	1	**64**	0	0,05266	0	**104**	1	0,89149	1
25	0	1,0423	1	**65**	0	0,50885	1	**105**	1	0,66268	1
26	1	1,04888	1	**66**	1	0,80903	1	**106**	0	0,37832	0
27	1	0,79166	1	**67**	1	0,82245	1	**107**	0	0,51745	1
28	1	0,63934	1	**68**	1	0,79747	1	**108**	1	0,60209	1
29	0	0,21641	0	**69**	1	0,94321	1	**109**	1	0,71231	1
30	0	0,07489	0	**70**	1	0,72618	1	**110**	1	0,97844	1
31	1	0,81946	1	**71**	1	0,4971	0	**111**	1	0,51568	1
32	0	0,23182	0	**72**	0	0,82086	1	**112**	1	0,82598	1
33	1	0,84196	1	**73**	1	1,01175	1	**113**	0	0,70083	1
34	1	0,78087	1	**74**	1	0,56101	1	**114**	0	0,11173	0
35	0	0,43768	0	**75**	1	0,95306	1	**115**	1	0,61461	1
36	0	0,55433	1	**76**	0	0,47694	0	**116**	1	0,66997	1
37	0	0,23975	0	**77**	1	0,68858	1	**117**	0	0,14618	0
38	1	0,64541	1	**78**	1	0,4527	0	**118**	0	0,43403	0
39	1	1,04154	1	**79**	1	1,02519	1	**119**	0	0,16202	0
40	1	0,9415	1	**80**	1	0,85527	1	**120**	1	0,63394	1

EK –2 Devam Momentumlu Geriye Yayılım Eğitim Algoritması ile Eğitilen Ağın Çıktıları ve Atamalar

Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$	
121	1	0,93002	1	**161**	1	1,04485	1	**201**	0	0,25072	0
122	1	0,72864	1	**162**	0	0,67497	1	**202**	0	0,08499	0
123	0	0,05309	0	**163**	1	1,01723	1	**203**	0	0,44056	0
124	0	0,19311	0	**164**	1	0,74261	1	**204**	0	0,30458	0
125	1	0,96373	1	**165**	1	0,94914	1	**205**	0	0,49703	0
126	0	0,40647	0	**166**	1	0,32715	0	**206**	0	0,34296	0
127	1	0,20467	0	**167**	1	0,60469	1	**207**	1	1,00966	1
128	1	0,91391	1	**168**	1	1,02368	1	**208**	1	0,75069	1
129	1	0,88747	1	**169**	0	0,19558	0	**209**	1	0,58231	1
130	1	0,62228	1	**170**	1	0,62333	1	**210**	0	0,43249	0
131	1	1,03674	1	**171**	1	0,6624	1	**211**	0	0,18097	0
132	0	0,50069	1	**172**	1	0,94388	1	**212**	0	0,86251	1
133	0	0,13089	0	**173**	1	0,86665	1	**213**	1	0,45046	0
134	0	0,11433	0	**174**	1	0,8255	1	**214**	0	0,03598	0
135	1	0,87751	1	**175**	1	1,01652	1	**215**	1	0,67417	1
136	1	0,98266	1	**176**	1	0,77842	1	**216**	0	0,11747	0
137	1	0,93106	1	**177**	1	1,01231	1	**217**	0	0,20673	0
138	1	1,01593	1	**178**	1	0,99558	1	**218**	0	0,06476	0
139	1	0,73753	1	**179**	1	0,24648	0	**219**	0	0,30301	0
140	1	0,80539	1	**180**	1	0,68828	1	**220**	0	0,18974	0
141	1	1,00974	1	**181**	1	0,69258	1	**221**	0	0,22042	0
142	0	0,91078	1	**182**	0	0,25218	0	**222**	1	0,76652	1
143	1	0,92096	1	**183**	0	0,87356	1	**223**	1	0,8719	1
144	0	0,3462	0	**184**	1	1,03482	1	**224**	1	0,652	1
145	1	0,97653	1	**185**	0	0,29457	0	**225**	0	0,09427	0
146	1	0,88207	1	**186**	1	0,77879	1				
147	1	0,99781	1	**187**	0	0,58642	1				
148	1	0,92285	1	**188**	0	0,50639	1				
149	1	0,53283	1	**189**	1	0,18535	0				
150	1	1,01197	1	**190**	1	0,96448	1				
151	0	0,29524	0	**191**	0	0,39104	0				
152	1	0,95771	1	**192**	0	0,16155	0				
153	1	0,79294	1	**193**	1	0,664	1				
154	1	0,84513	1	**194**	0	0,48395	0				
155	0	0,20006	0	**195**	0	0,20337	0				
156	1	0,98191	1	**196**	0	0,15544	0				
157	1	0,9724	1	**197**	1	0,44551	0				
158	1	0,99623	1	**198**	1	0,40256	0				
159	1	0,2827	0	**199**	0	0,1016	0				
160	1	1,04022	1	**200**	0	0,34443	0				

EK –3 Eşlenik Eğimli Geriye Yayılım Eğitim Algoritması ile Eğitilen Ağın Çıktıları ve Atamalar

Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$	
1	1	0,89024	1	**41**	1	0,84816	1	**81**	0	0,84509	1
2	1	1,05556	1	**42**	1	0,75784	1	**82**	1	0,87481	1
3	1	1,02813	1	**43**	1	0,77964	1	**83**	0	-0,0541	0
4	1	0,72467	1	**44**	1	1,05556	1	**84**	0	0,06841	0
5	1	0,91765	1	**45**	1	0,88772	1	**85**	1	1,05556	1
6	1	0,81609	1	**46**	1	0,87049	1	**86**	1	1,05166	1
7	1	1,05555	1	**47**	1	0,78048	1	**87**	0	-0,0223	0
8	1	0,53542	1	**48**	1	1,05556	1	**88**	0	0,90508	1
9	1	1,05556	1	**49**	1	1,05556	1	**89**	0	-0,0551	0
10	0	0,66747	1	**50**	1	0,97301	1	**90**	0	0,40264	0
11	1	0,79094	1	**51**	1	0,90536	1	**91**	0	-0,0556	0
12	1	0,88872	1	**52**	0	0,16309	0	**92**	0	0,88377	1
13	1	1,05554	1	**53**	1	0,9022	1	**93**	0	-0,0519	0
14	1	0,77044	1	**54**	0	0,14021	0	**94**	1	1,05556	1
15	1	0,84211	1	**55**	1	0,73739	1	**95**	1	0,86252	1
16	1	0,51702	1	**56**	1	0,793	1	**96**	1	0,94472	1
17	1	1,05534	1	**57**	1	1,05556	1	**97**	1	1,05556	1
18	0	0,87936	1	**58**	0	0,38994	0	**98**	1	0,81095	1
19	0	0,6057	1	**59**	1	0,99746	1	**99**	0	0,093	0
20	1	0,97436	1	**60**	0	0,86408	1	**100**	1	0,91679	1
21	0	0,1718	0	**61**	1	1,05556	1	**101**	0	0,01885	0
22	0	0,20842	0	**62**	1	1,05556	1	**102**	1	1,05354	1
23	1	0,91427	1	**63**	1	0,8652	1	**103**	0	-0,054	0
24	0	0,12893	0	**64**	0	-0,0547	0	**104**	1	0,94083	1
25	0	1,05556	1	**65**	0	0,89112	1	**105**	1	0,51481	1
26	1	0,99646	1	**66**	1	1,05556	1	**106**	0	0,08312	0
27	1	0,80085	1	**67**	1	0,90335	1	**107**	0	0,11724	0
28	1	0,69824	1	**68**	1	0,83206	1	**108**	1	0,88907	1
29	0	-0,0183	0	**69**	1	0,87614	1	**109**	1	0,77989	1
30	0	-0,055	0	**70**	1	0,61672	1	**110**	1	0,99271	1
31	1	1,05556	1	**71**	1	0,57313	1	**111**	1	0,88033	1
32	0	0,20956	0	**72**	0	0,37926	0	**112**	1	0,73262	1
33	1	0,92768	1	**73**	1	0,90289	1	**113**	0	0,8743	1
34	1	0,69401	1	**74**	1	0,73921	1	**114**	0	0,08716	0
35	0	0,89561	1	**75**	1	0,91198	1	**115**	1	1,05536	1
36	0	0,41113	0	**76**	0	0,30568	0	**116**	1	0,99728	1
37	0	0,04357	0	**77**	1	1,05556	1	**117**	0	0,03623	0
38	1	0,70362	1	**78**	1	0,87102	1	**118**	0	0,71278	1
39	1	1,05556	1	**79**	1	1,05556	1	**119**	0	-0,0337	0
40	1	0,73698	1	**80**	1	1,05556	1	**120**	1	0,85689	1

EK –3 Devam Eşlenik Eğimli Geriye Yayılım Eğitim Algoritması ile Eğitilen Ağın Çıktıları ve Atamalar

Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$		Sıra	y	$\hat{y}$	
121	1	0,94798	1	**161**	1	1,05556	1	**201**	0	0,05673	0
122	1	0,72679	1	**162**	0	1,05556	1	**202**	0	-0,0556	0
123	0	-0,0555	0	**163**	1	1,05556	1	**203**	0	0,59699	1
124	0	0,25731	0	**164**	1	0,39464	0	**204**	0	0,32373	0
125	1	1,05313	1	**165**	1	0,80576	1	**205**	0	0,65919	1
126	0	-0,006	0	**166**	1	0,71277	1	**206**	0	0,08518	0
127	1	0,9575	1	**167**	1	0,59211	1	**207**	1	1,05556	1
128	1	0,75868	1	**168**	1	1,05556	1	**208**	1	0,89274	1
129	1	1,05556	1	**169**	0	0,10023	0	**209**	1	0,68633	1
130	1	0,90522	1	**170**	1	0,8689	1	**210**	0	0,11991	0
131	1	1,05556	1	**171**	1	0,76326	1	**211**	0	0,05516	0
132	0	0,49477	0	**172**	1	0,88155	1	**212**	0	0,75048	1
133	0	-0,0526	0	**173**	1	1,05556	1	**213**	1	0,8371	1
134	0	0,14624	0	**174**	1	0,78359	1	**214**	0	0,07796	0
135	1	0,86621	1	**175**	1	1,05063	1	**215**	1	0,5718	1
136	1	0,88251	1	**176**	1	1,05556	1	**216**	0	-0,0533	0
137	1	1,05556	1	**177**	1	0,82365	1	**217**	0	1,05556	1
138	1	1,05554	1	**178**	1	1,05276	1	**218**	0	-0,0556	0
139	1	1,05556	1	**179**	1	0,65768	1	**219**	0	0,30238	0
140	1	0,88154	1	**180**	1	0,57323	1	**220**	0	-0,0286	0
141	1	1,05556	1	**181**	1	1,02418	1	**221**	0	0,48134	0
142	0	0,84159	1	**182**	0	0,82868	1	**222**	1	0,53959	1
143	1	0,96046	1	**183**	0	0,87545	1	**223**	1	0,59284	1
144	0	-0,0353	0	**184**	1	1,05556	1	**224**	1	0,92536	1
145	1	0,87698	1	**185**	0	0,11214	0	**225**	0	-0,0511	0
146	1	0,87241	1	**186**	1	1,05556	1				
147	1	1,05556	1	**187**	0	-0,0497	0				
148	1	1,05554	1	**188**	0	0,33169	0				
149	1	0,97435	1	**189**	1	0,35802	0				
150	1	0,74506	1	**190**	1	1,02339	1				
151	0	0,33052	0	**191**	0	1,05556	1				
152	1	0,90438	1	**192**	0	0,4413	0				
153	1	0,87157	1	**193**	1	0,62298	1				
154	1	0,85154	1	**194**	0	0,29677	0				
155	0	-0,0544	0	**195**	0	-0,0089	0				
156	1	0,91331	1	**196**	0	0,05315	0				
157	1	1,05556	1	**197**	1	0,63256	1				
158	1	1,05556	1	**198**	1	0,3103	0				
159	1	0,31056	0	**199**	0	-0,0548	0				
160	1	1,05548	1	**200**	0	0,78801	1				