TÜRKİYE CUMHURİYETİ
ANKARA ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARININ İNCELENMESİ VE SIRT AĞRISI OLAN BİREYLER ÜZERİNDE BİR UYGULAMASI

Burcu KARAKAYA

BİYOİSTATİSTİK ANABİLİM DALI
YÜKSEK LİSANS TEZİ

DANIŞMAN
Prof. Dr. Ersöz TÜCCAR

2007 - ANKARA

Ankara Üniversitesi Sağlık Bilimleri Enstitüsü

Biyoistatistik Yüksek Lisans Programı

çerçevesinde yürütülmüş olan bu çalışma, aşağıdaki jüri tarafından
Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 17 / 04 / 2007

Prof. Dr. Ersöz TUCCAR
Ankara Üniversitesi Tıp Fakültesi
Biyoistatistik Anabilim Dalı
Jüri Başkanı – Tez Danışmanı

Prof. Dr. Osman SARAÇBAŞI
Hacettepe Üniversitesi Tıp Fakültesi
Biyoistatistik Anabilim Dalı
Üye

Doç. Dr. Atilla Halil ELHAN
Ankara Üniversitesi Tıp Fakültesi
Biyoistatistik Anabilim Dalı
Üye

Yrd. Doç. Dr. Serdal Kenan KÖSE
Ankara Üniversitesi Tıp Fakültesi
Biyoistatistik Anabilim Dalı
Üye

Yrd. Doç. Dr. Yasemin GENÇ
Hacettepe Üniversitesi Tıp Fakültesi
Biyoistatistik Anabilim Dalı
Üye

İÇİNDEKİLER

	Sayfa
Kabul ve Onay	ii
İçindekiler	iii
Önsöz	v
Simgeler ve Kısaltmalar	vi
Şekiller	vii
Çizelgeler	ix
1. GİRİŞ	1
1.1. Biyolojik Sinir Hücresi (Nöron)	2
1.2. Biyolojik Sinir Hücresinin ve Yapay Sinir Hücresinin Bağdaştırılması	3
1.3. Yapay Sinir Ağı ve Yapı Taşları	3
1.4. Etkinleştirme İşlevi	8
1.5. Öğrenme Algoritmaları	9
1.5.1. Algılayıcı Öğrenme Kuralı	10
1.5.2. ADALINE (ADAptive LINear Element)	12
1.6 Öğrenme Katsayısı ve Momentum	16
1.7. Öğrenme Algoritmalarının Uygulanış Biçimleri	18
1.7.1. Toplu İşlem Yolu İle Öğrenme (Batch Learning) ve Anında Öğrenme (Incremental Learning)	18
1.7.2. Aşırı Öğrenme (OverTraining)	22
1.8. Öğrenme Algoritmalarının Sınıflandırılması	25
1.8.1. Danışmalı Öğrenme	25
1.8.2. Danışmansız Öğrenme	26
1.8.3. Takviyeli Öğrenme	26
1.9. Tezin Amacı	26
2. GEREÇ VE YÖNTEM	27
2.1. Joone (Java Object Oriented Neural Engine)	27
2.2. Joone Kurulması	27
2.3. Sırt Ağrısı Yapay Sinir Ağının "Joone Editör" Kullanılarak Hazırlanması	28
2.4. Kullanılan Bileşenler ve Özellikleri	29
2.5. Hazırlanan Yapay Sinir Ağının Eğitilmesi	36
2.6. Öğrenme Yönteminin Seçilmesi	38
2.6.1. Geri Bildirimli Anında Öğrenme (The Basic On-Line Backprop Algorithm)	38
2.6.2. Geri Bildirimli Toplu Öğrenme (The BachProp Algorithm)	39
2.6.3. Esnek Geri Bildirimli Toplu Öğrenme (The Resilient BackProp Algorithm)	40
2.7. Öğrenme Sonrası	40
2.8. Hazırlanan Ağın "Joone Editor" İçinde Test Edilmesi	40
2.9. Joone Editörü İle Hazırlanmış Bir Yapay Sinir Ağına Bir Java Programından Erişilmesi ve Bu Program Yolu İle Test Edilmesi	41

	Sayfa
2.10. Sırt Ağrısı	41
2.10.1. Ağrı Bilimi	41
2.10.2. Ağrı Sınıfları	42
2.10.3. Ağrı Oluşumu	42
2.10.4. Sırt Ağrısında Tanı	43
2.11. Girdi Verileri	44
2.11.1. Girdi Dosyasındaki Girdi Değişkenleri (Bağımsız Değişkenler)	44
2.11.2. Bağımlı Değişken	44
2.11.3. Uygulama Veritabanı	45
3. BULGULAR	51
3.1. YSA Parametrelerindeki Değişimin RMSE'ye Etkisi	51
3.2. En Küçük RMSE ve En Uygun YSA Parametrelerine Karşılık Gelen Bağımlı Değişken Tahmin Değerleri	53
4. TARTIŞMA	55
5. SONUÇ ve ÖNERİLER	57
ÖZET	58
SUMMARY	59
KAYNAKLAR	60
EKLER	61
Ek-1	61
ÖZGEÇMİŞ	65

ÖNSÖZ

Yüksek lisans yapmam için beni yüreklendiren annem ve babam Ziraat Y.Müh. Mediha KUŞMAN ve Dr. Noyan KUŞMAN'a, çalışmalarımda bilgi ve deneyimlerini paylaşarak yönlendiren eşim Bilgisayar Müh. Zülfikar KARAKAYA'ya teşekkürü bir borç bilirim. İstatistik ve biyoistatistik bilimini öğrendiğim değerli Gazi Üniversitesi İstatistik Bölümü ve Ankara Üniversitesi Biyoistatistik A.B.D ve Hacettepe Üniversitesi Biyoistatistik A.B.D'na teşekkür ederim. Tez izleme komitemde yer alarak çalışmaya sağladıkları değerli katkılarından dolayı Prof. Dr. Ersöz TÜCCAR, Doç. Dr. Atilla Halil ELHAN, Y.Doç. Kenan KÖSE, Y.Doç. Yasemin GENÇ, Prof. Dr. Osman SARAÇBAŞI ve uygulama veritabanı için gösterdiği ilgi ve yönlendirmeler için Uzm. Dr. Dilek KESKİN'e ayıca teşekkür ederim.

SİMGELER VE KISALTMALAR

ADALINE	Adaptive Linear Element
$d(x)$	İstenen Sonuçlar Kümesi
E	Hata Maliyeti İşlevi
F	Etkinleştirme İşlevi
FIFO	First In First Out (İlk Giren İlk Çıkar)
Joone	Java Object Oriented Neural Engine
LMS	En Küçük Kareler Ortalaması
m	Momentum
n	Girdi Örüntüleri Sayısı
RMSE	Hata Kareler Ortalaması Kökü
s	Toplama İşlevi
t	Adım
t_a	Öğrenme oranının değiştirilmeye başladığı adımı
t_i	i. Örüntü İçin İstenen Çıktı Değeri
W(eski)	Bir Önceki Adımda Kullanılan Ağırlık Değerini
Δw_i	Ağırlıktaki Değişim Miktarı
W(yeni)	Bir Sonraki Adımda Kullanılacak Ağırlık Değerini
y_i	i. Örüntü İçin Ağın Verdiği Çıktı Değerini
YSA	Yapay Sinir Ağı
ε	Adım Büyüklüğü
$\Delta\theta$	Eşik Değerindeki Değişim
$\frac{\partial E}{\partial W}$	Hata Maliyeti İşlevinin Ağırlık Değerine Göre Kısmi Türevi
w_i	i. Birimi Besleyen Ağırlık Değeri
μ_o	ilk öğrenme oranı değerini
μ	Öğrenme Oranı
$\mu(t)$	t. adımdaki öğrenme oranı değerini
θ	Yönlendirme(Bias)

ŞEKİLLER

		Sayfa
Şekil 1.1.	Biyolojik Sinir Hücresi	2
Şekil 1.2.	Sinir Ağı Genel Görünümü	4
Şekil 1.3.	Algılayıcının Genel İşleyişi	4
Şekil 1.4.	Tek Katmanlı İleri Beslemeli Yapay Sinir Ağı	5
Şekil 1.5.	Denklem 1.4.'ün Geometrik Gösterimi	6
Şekil 1.6.	$< w, x >$ Gösterimi	8
Şekil 1.7.	Bazı Etkinleştirme İşlevleri	9
Şekil 1.8.	Girdi Kümesinin Geometrik Gösterimi	11
Şekil 1.9.	Denklem 1.23'ün Geometrik Gösterimi	14
Şekil 1.10.	$x_0 = 2, \varepsilon = 0{,}1$ İçin Adımların Gösterimi	14
Şekil 1.11.	$x_0 = 0{,}01$ ve $\varepsilon = 0{,}1$ değerleri için adımlar	15
Şekil 1.12.	Yerel Minimuma Takılıp Kalma	16
Şekil 1.13.	Momentumsuz Arama	17
Şekil 1.14.	Momentum Kullanılarak Arama	18
Şekil 1.15.	Mutlak En Küçük Değer	19
Şekil 1.16.	Hata Oranının Deneme Sayısına Bağlı Çizelgesi	20
Şekil 1.17.	Yerel Minimum Noktalar	21
Şekil 1.18.	İki Sigma Tabanlı Gizli Katmanın Mutlak En Küçük Değere Ulaşması	21
Şekil 1.19.	Aşırı Gizli Katmanın Olmadığı Durum	22
Şekil 1.20.	Gereğinden Fazla Gizli Katmanın Olduğu Durum	22
Şekil 1.21.	Aşırı Öğrenme	23
Şekil 1.22.	Durdurma Noktası	24
Şekil 1.23.	"Brain Damage"	24
Şekil 1.24.	Sigma Tabanlı Etkinleştirme İşlevinin Doğrusal İşlevi	25
Şekil 2.1.	Sırt Ağrısı Yapay Sinir Ağı	28
Şekil 2.2.	Excell Girdisinin Tanımlanması	29
Şekil 2.3.	Verilerin Kontrol Edilmesi	30
Şekil 2.4.	Girdi Katmanının Özelliklerinin Belirlenmesi	30
Şekil 2.5.	Gizli Katmanının Özelliklerinin Belirlenmesi	31
Şekil 2.6.	Çıktı Katmanının Özelliklerinin Belirlenmesi	31
Şekil 2.7.	Çıktı Anahtarının Özelliklerinin Belirlenmesi	32
Şekil 2.8.	Öğreticinin Özelliklerinin Belirlenmesi	32
Şekil 2.9.	İstenen Sonucun Özelliklerinin Belirlenmesi	33

		Sayfa
Şekil 2.10.	İstenen Sonuçların İzlenebilmesi	34
Şekil 2.11.	Sonuçların Yazılacağı Dosyanın Belirlenmesi	34
Şekil 2.12.	Hafıza Bileşeninin Özelliklerinin Girilmesi	34
Şekil 2.13.	Tüm Kombinasyonların Gerçeklenmesi	35
Şekil 2.14.	Gizli Çıktı Bağlantısının Özellikleri	35
Şekil 2.15.	Kontrol Sonuç Penceresi	36
Şekil 2.16.	İşleyişin Özet Gösterimi	36
Şekil 2.17.	YSA Eğitimi İçin Gerekli Düzenlemeler	37
Şekil 2.18.	Yüzeyde Mutlak Minimum Arayışının Betimlenmesi	39
Şekil 3.1.	YSA için RMSE Değerlerinin İzlenmesi	48

ÇİZELGELER

		Sayfa
Çizelge 1.1.	Biyolojik Sinir Sistemi İle Yapay Sinir Sistemi Arasındaki Benzerlikler	3
Çizelge 2.1.	Uygulama Veritabanı	45
Çizelge 3.1.	YSA Parametrelerindeki Değişimin RMSE'ye Etkisi	47
Çizelge 3.2.	YSA Nüks ve Gerçek Nüks	48
Çizelge 3.3.	YSA'nın Yanlış Sınıfladığı Hastalar	49
Çizelge 4.1.	Uygulama Veritabanına Lojistik Regresyon Çözümlemesi	52

1. GİRİŞ

Yapay sinir ağları (YSA), ilk olarak 1943 yılında McCulloch & Pitts tarafından ortaya atılan, yapısal olarak biyolojik sinir sistemlerini temel alan mantıksal bir model ile gündeme gelmiştir. Ardından 1969 yılında Minsky ve Papert tarafından yayımlanan "Perceptrons" adlı kitapla olgunlaşan yapay sinir ağları kavramı, Teuvo Kohonen, Stephen Grossberg, James Anderson ve Kunihiko Fukushima gibi bilim adamlarının çalışmalarıyla daha da gelişmiştir. Seksenli yılların başlarında gelişen donanım teknolojilerine bağlı olarak ilk meyvelerini vermeye başlayan yapay sinir ağları, bugün bir çok üniversitenin psikoloji, fizik, bilgisayar bilimleri, biyoloji gibi bölümlerinde yapılan araştırmalarda kullanılan çok önemli bir öğrenme algoritması halini almıştır (Kröse, B. Van Der Smagt, P. 1996).

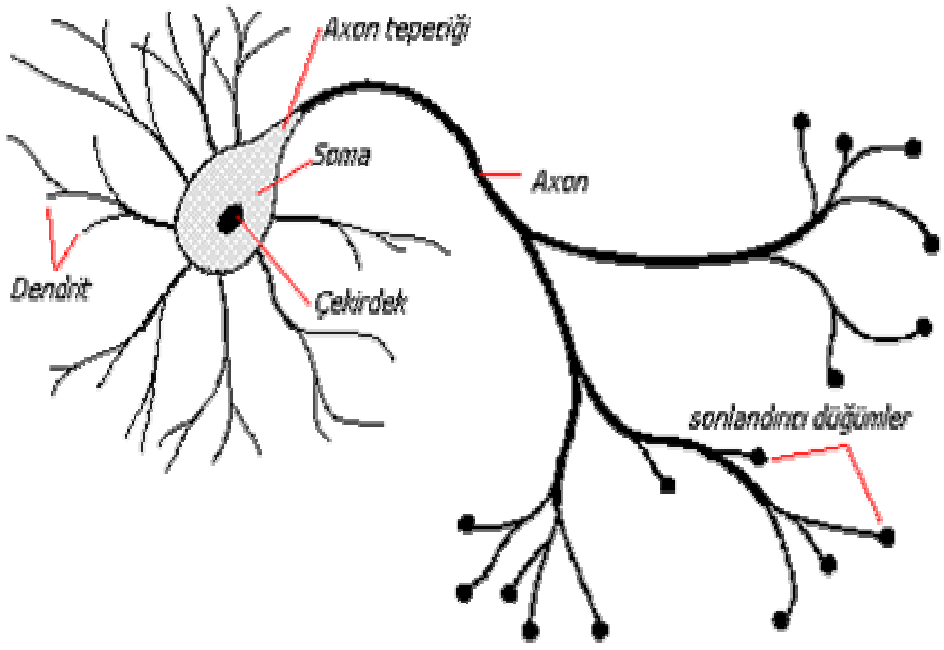
Yapay sinir ağı, öğrenme, veriyi sınıflandırarak bilgiye çevirme ve eşzamanlı birden çok işlem yapabilme kabiliyetine sahip matematiksel bir modeldir. Ancak kendisinden esinlenilen biyolojik sinir ağlarının çalışma yönteminin mükemmelliğine, karmaşıklığına ve verimliliğine erişebilmesi mümkün olmamıştır.

Yapay sinir ağları beynin çalışma ilkelerinin sayısal bilgisayarlar üzerinde taklit edilmesi fikri ile ortaya çıkmış ve ilk çalışmalar beyni oluşturan biyolojik hücrelerin ya da nöronların matemetiksel olarak modellenmesi üzerinde yoğunlaşmıştır. Yapay sinir ağları, karar hızı açısından insan beyni ile yarışabilecek aşamayı henüz katetmemiş olmalarına rağmen gün geçtikçe hızları artmaktadır. Bu anlamda yapay sinir ağları konusu üzerinde çalışırken, bir ağ yapısının çözebileceği problem uzayının, insan beyninin çözebildiği problem uzayının kısıtlanmış bir alt kümesi olmaktadır.

1.1. Biyolojik Sinir Hücresi (Nöron)

Biyolojik sinir sistemi, bilgiyi alan, yorumlayan ve uygun bir karar üreten bir merkez ve bu merkezin kontrolünde bulunan alıcı ve tepki sinirlerinden oluşmaktadır. Sinir hücrelerine tıp terminolojisinde nöron denmektedir. Nöron, sinir sisteminin temel birimidir ve gövde, gövdeye giren sinyal alıcılar (dentrit), gövdeden çıkan sinyal ileticiler (akson) olmak üzere başlıca üç kısımdan oluşmaktadır.

Dendritler üzerinden girişler alınır, soma tarafından girişler işlenir. Nörondaki sinyalleri taşıyan uzun bir sinirsel bağlantı halindeki akson ise, işlenen girişleri çıkışa aktarır. Akson dendrit bağlantısı ise sinaps olarak adlandırılır. Sinaps nöronlar arasında elektrokimyasal bağlantıyı sağlamaktadır.



Şekil 1.1. Biyolojik Sinir Hücresi (Anderson, D., Mc Neıll, G., 1992)

1.2. Biyolojik Sinir Hücresinin ve Yapay Sinir Hücresinin Bağdaştırılması

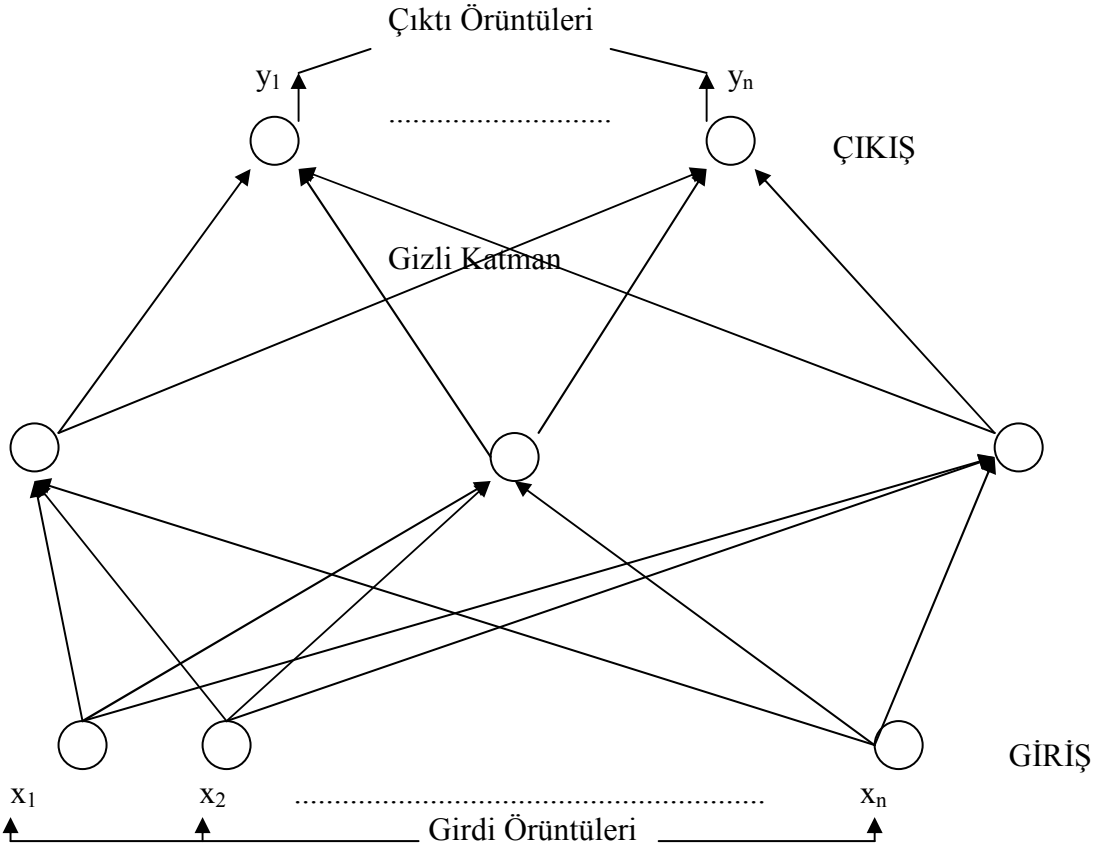
Yapay sinir ağları, birbirine bağlı doğrusal ve/veya doğrusal olmayan birçok elemandan oluşmaktadır. Biyolojik sinir sistemi ile yapay sinir sistemi arasındaki benzerlikler aşağıdadır.

Çizelge 1.1. Biyolojik Sinir Sistemi İle Yapay Sinir Sistemi Arasındaki Benzerlikler (Beşdok, E., Erler, M., Sağıroğlu, Ş. 2003)

Biyolojik Sinir Sistemi	**Yapay Sinir Ağı**
Nöron	Algılayıcı (Perceptron)
Dentrit	Toplama işlevi
Hücre gövdesi	Etkinleştirme işlevi
Aksonlar	Algılayıcı çıkışı
Sinapslar	Ağırlıklar

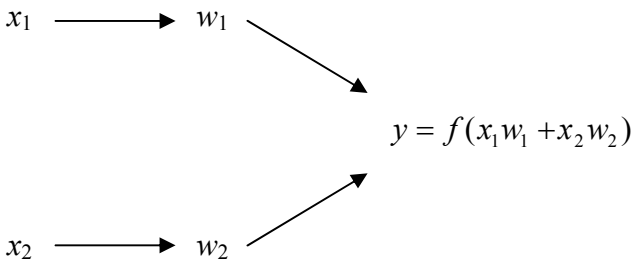
1.3. Yapay Sinir Ağı ve Yapı Taşları

Bir yapay sinir ağı, katmanlar, bağlantılar, bağlantı ağırlıkları, etkinleştirme işlevleri, giriş ve çıkış değerlerinden oluşmaktadır. Genel olarak bir sinir ağında üç ayrı katman bulunmaktadır. Bu katmanlar giriş katmanı, gizli katman ve çıkış katmanıdır. Giriş katmanı yapay sinir ağının beslenme noktasıdır ve veriyi dışarıdan alan kısımdır. Gizli katman, girdi ve çıktı katmanları arasındaki bağları içermektedir. Çıktı katmanı sonucun alındığı katmandır.



Şekil 1.2. Sinir Ağı Genel Görünümü

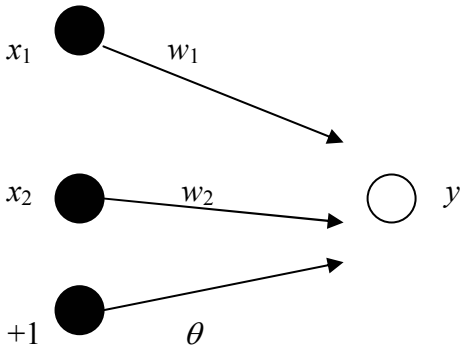
Her bir katman özellikli düğümlerden oluşmaktadır. Bu özellikli düğümlere algılayıcı denir. Algılayıcı, biyolojik bir sinir ağındaki sinir hücrelerine karşılık gelmektedir. Bir algılayıcı yapay sinir ağının en küçük birimidir. Algılayıcının genel işleyişi Şekil 1.3.'de gösterilmektedir.



Şekil 1.3. Algılayıcının Genel İşleyişi

x_1 ve x_2 değerleri giriş değerleri ve sırasıyla bu giriş değerlerinin ağırlıkları olan w_1 ve w_2 değerleri birbirleriyle çarpılmaktadır. Bu değerler toplanarak "etkinleştirme işlevi" (activation function) diye anılan bir işlevden geçirilerek çıkış değeri elde edilmektedir. Bu işlevin "etkinleştirme işlevi" diye anılmasının sebebi, biyolojik sinir hücresinin etkin olabilmesi (iletim yapabilmesi) için gerekli olan eşik değerini aşabilecek değeri üreten bir işlev gibi düşünülmesidir. Bölüm 1.4' te bazı etkinleştirme işlevleri verilmektedir.

Şekil 1.4.'deki gibi bir algılayıcı aslında tek başına bir "tek katmanlı ileri beslemeli" (single layer feed-forward) yapay sinir ağıdır. Tek katmanlı ileri beslemeli sinir ağı bir ya da birden fazla çıkış sinir hücresi ya da düğüm içerebilmektedir (Kröse, B. Van Der Smagt, P. 1996).



Şekil 1.4. Tek Katmanlı İleri Beslemeli Yapay Sinir Ağı

Şekil 1.4.'teki +1 giriş değeri θ ağırlığı ile sınıflandırma işlevine katılmaktadır. θ'ya, "destek" ya da "yönlendirme" (bias) denir. Bu algılayıcının etkinleştirme işlevinin matematiksel olarak gösterimi Denklem 1.1.'de gösterilmektedir.

$$y = F(\sum_{i=1}^{2} w_i x_i + \theta) \qquad [1.1]$$

Bu etkinleştirme işlevi doğrusal ya da doğrusal olmayan türde olabilir. Aşağıda doğrusal bir işlev gösterilmiştir.

$$F(s) = \begin{cases} 1 & ,s > 0 \\ -1 & ,diger\ durumlar \end{cases} \qquad [1.2]$$

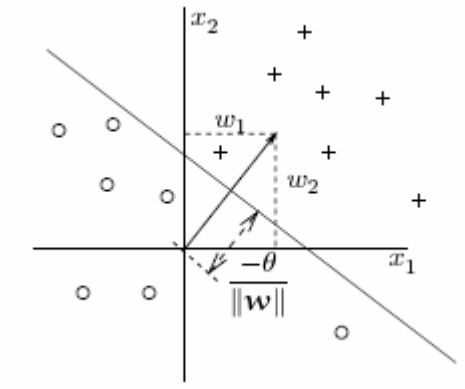
"Toplama İşlevi" sonucu elde edilen sonuç etkinleştirme işlevine aktarılır. Denklem 1.2. ye aktarılan sonucu üreten toplama işlevi Denklem 1.3.' te verilmektedir.

$$s = w_1 x_1 + w_2 x_2 + \theta \qquad [1.3]$$

Böyle bir etkinleştirme işlevi sayesinde sınıflandırma yapmak mümkün olmaktadır. Toplama işlevi sonucunun sıfırdan büyük olduğu durumlar bir sınıfı, diğer durumlar ise bir başka sınıfı temsil edebilir. Bu sayede iki sınıfı ayrıştırmak için kullanılabilecek bir doğru denklemi Denklem 1.4.'deki gibi ifade edilebilir.

$$w_1 x_1 + w_2 x_2 + \theta = 0 \qquad [1.4]$$

Bu denklemin geometrik gösterimi ise aşağıdaki gibidir.



Şekil 1.5. Denklem 1.4.'ün Geometrik Gösterimi

Sınıflandırma çizgisinin başlangıç noktasından ((0,0) noktası) uzaklığı bir noktanın bir doğruya olan uzaklığı formülünden;

$$\frac{-\theta}{\|w\|} \qquad [1.5]$$

olarak bulunur. $\|w\| = 1$ olduğu zaman, $\|w\|$ değeri normalleştirilmiş demektir. Ayrıştırma doğrusu başlangıç noktasından eşik değeri kadar uzaktadır. Bu, yardımcı değer olan θ (bias) değerinin ne kadar anlamlı olduğunu göstermektedir (Kröse, B. Van Der Smagt, P. 1996).

Ağırlıklar değiştirilerek doğrunun eksenleri kestiği noktalar değiştirilmekte ve doğru parçası sağa veya sola döndürülmüş olmaktadır. θ değerinin değiştirilmesi ile doğru parçası başlangıç noktasına göre taşınmaktadır.

Ayrıştırma doğrusunu ayrıca, aşağıdaki gibi sayıl (scalar) çarpım ile de göstermek mümkündür.

$$< w,x >= \|w\| \|x\| \cos(w,x) \qquad [1.6]$$

$$\|w\| = (w_1^2 + w_2^2)^{1/2} \quad , \|x\| = (x_1^2 + x_2^2)^{1/2} \text{ ise} \qquad [1.7]$$

$$< w,x >= (w_1^2 + w_2^2)^{1/2} (x_1^2 + x_2^2)^{1/2} \cos(w,x) = \|w\| \|x\| \cos(w,x) \qquad [1.8]$$

Bu gösterimin çıkarımı aşağıdaki gibidir.

$$\cos(w,x) = \cos(\alpha - \theta) \qquad [1.9]$$

$$\cos(w,x) = \cos(\alpha)\cos(\theta) + \sin(\alpha)\sin(\theta) \qquad [1.10]$$

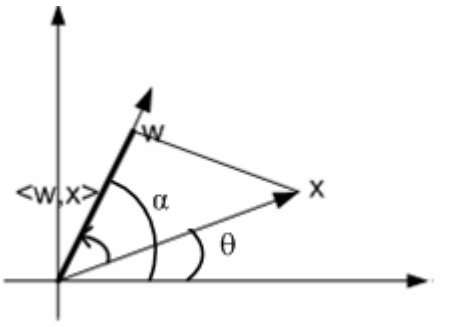
$$\cos(w,x) = ((w_1 w_2) / (w_1^2 + w_2^2)^{1/2} (x_1^2 + x_2^2)^{1/2}) + ((x_1 x_2) / (w_1^2 + w_2^2)^{1/2} (x_1^2 + x_2^2)^{1/2}) \qquad [1.11]$$

bu durumda,

$< w, x >= (w_1^{\,2} + w_2^{\,2})^{1/2}\ (x_1^{\,2} + x_2^{\,2})^{1/2} \cos(w, x)$ olduğundan, cos (*w*,*x*) yerine konursa

$$< w, x > = w_1 x_1 + w_2 x_2 \qquad [1.12]$$

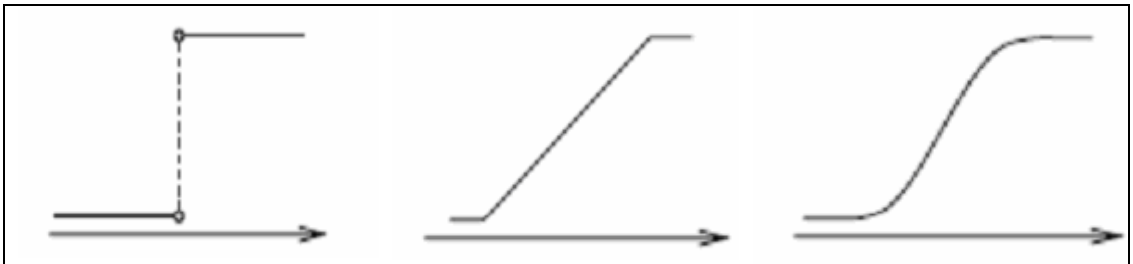
bulunur. $\cos(w, x) = \|w\| \|x\| \cos(w, x)$ gösterimine geri dönersek, burada $\|w\| = 1$ olması durumunda, $\cos(w, x) = \|x\| \cos(w, x)$ halini almakta ve bu x' in w üzerine izdüşümü olarak yorumlanmaktadır (Finland, 2005).



Şekil 1.6. $< w, x >$ Gösterimi

1.4. Etkinleştirme İşlevleri

Algılayıcı davranışını etkileyen önemli etmenlerden biri de etkinleştirme işlevidir. Biyolojik sinir hücresinde bir eşik değerini aşan uyarılara karşı nöronun kısa süreli bir darbe gönderdiği bilinmektedir. Yapay sinir hücresinde ise toplama işlevinin sonucu belli bir değeri aştığında alınacak yanıtın biçimini belirleyen etkinleştirme işlevidir (Finland, 2005). Bu yüzden toplama işlevinin sonucu bir etkinleştirme işlevine aktarılmaktadır. Bazı etkinleştirme işlevleri aşağıda verilmiştir.



Kesikli İşlev (İşaret İşlevi) Yarı Doğrusal İşlev Sigma Tabanlı İşlev

Şekil 1.7. Bazı Etkinleştirme İşlevleri

Eğer fazlaca karmaşık, iç içe geçmiş, ayrıştırılması ve sınıflandırılması zor gibi görünen birden fazla sınıf ayrıştırılıyorsa, etkinleştirme işlevi olarak sigma tabanlı etkinleştirme işlevi gibi sürekli (türevlenebilen) ve esnek sonuçlar üreten işlevler kullanılmaktadır.

1.5. Öğrenme Algoritmaları

Aşağıda iki çeşit öğrenme yöntemine değinilmiştir. Bunlar "algılayıcı öğrenme kuralı" ve "delta kuralı" ya da "en küçük hata kare" kuralıdır. Her iki yöntem de tekrarlı bir şekilde ağırlıkları güncelleme ilkesine dayanır (Kröse, B. Van Der Smagt, P. 1996). Denklem 1.12'de, t+1'inci adımdaki ağırlık güncelleştirmesi gösterilmektedir.

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \qquad [1.13]$$

w_i : i. birimi besleyen ağırlık değeri,

Δw_i : ağırlıktaki değişim miktarı

t: adım

1.5.1. Algılayıcı Öğrenme Kuralı

x elimizdeki girdi verilerini temsil eden bir girdi vektörü ve *d(x)*'in istenen sonuçlar kümesi olduğunda yapılması gereken adımlar aşağıdadır (Kröse, B. Van Der Smagt, P. 1996).

1. Ağırlık değerlerine rasgele küçük sayılar ata.
2. Girdi verilerinden bir örüntüyü x vektörüne ata.
3. Eğer $y \neq d(x)$ ise tüm ağırlıkları aşağıdaki kurala göre güncelle.

$$\Delta w_i = d(x_i) \qquad [1.14]$$

4. 2. adıma geri dön.

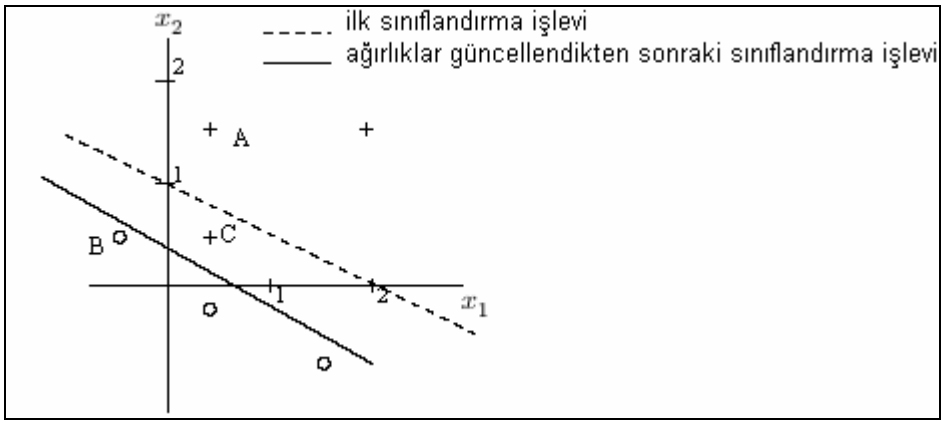
Ağırlıkların yanı sıra eşik değerinin de güncellenmesi gerekir. Eşik değerindeki değişim aşağıdaki gibi hesaplanır.

$$\Delta\theta = \begin{cases} 0 & ,algılayıcı\ dogru\ sonuc\ uretirse \\ d(x) & ,diger\ durumlar \end{cases} \qquad [1.15]$$

Bir algılayıcının ilk ağırlık değerlerinin şöyle olduğunu varsayalım.

$$w_1 = 1, \quad w_2 = 2, \quad \theta = -2 \qquad [1.16]$$

Şekil 1.8'te gösterilen girdi kümesinin sınıflandırılmaya çalışıldığını varsayalım.



Şekil 1.8. Girdi Kümesinin Geometrik Gösterimi

"+" işaretli noktalar için istenen değerin $d(x) = +1$ olduğunu düşünüldüğünde, "o" işaretli noktalar için ise istenen çıktı değeri –1 olur.

A için girdi vektörü x=(0,5;1,5) ve Denklem 1.2'ye göre A noktası için hesaplanan çıktı değeri +1'dir. Bu durumda hesaplanan çıktı değeri ile istenen çıktı değeri arasında fark olmadığından herhangi bir ağırlık güncellemesi yapmaya gerek yoktur.

B noktası için hesaplana çıktı değeri -1 olur. Buradaki durum da A noktasındaki gibidir ve ağırlıkların değiştirilmesine gerek yoktur.

C noktası için, yani x=(0,5;0,5) girdi vektörüne karşılık hesaplanan çıktı değeri –1 olur. İstenen değer hesaplanan değerden farklı olduğundan yeni ağırlık değerleri hesaplanmalıdır. Bu durumda ağırlıklardaki ve θ değerindeki değişim aşağıdaki gibi olur.

$$\Delta w_1 = 0{,}5 \quad \Delta w_2 = 0{,}5, \quad \Delta\theta = 1 \qquad [1.17]$$

Böylece algılayıcı öğrenme yöntemine göre hesaplanan yeni ağırlıklar şu şekilde olur.

$$w_1 = 1{,}5 \quad w_2 = 2{,}5, \quad \theta = -1 \qquad [1.18]$$

Oluşan yeni sınıflandırma denklemi şekilde sürekli çizgi ile gösterilmiştir. Bu şekilde C noktası da doğru bir şekilde sınıflandırılmış olur.

1.5.2. ADALINE (ADAptive LINear Element)

ADALINE en önemli algılayıcı öğrenme algoritmalarından biridir. Widrow ve Hoff tarafından "En küçük kareler ortalaması" (LMS – Least Mean Square) olarak sunulan bu yöntem "Delta Kuralı" (Delta Rule) olarak da bilinmektedir. Algılayıcı Öğrenme Kuralı ile Delta Kuralı arasındaki en önemli fark çıktı biçimidir. Algılayıcı öğrenme kuralında kesikli bir etkinleştirme işlevi kullanılmakta iken (–1 ya da +1 sonuçlarından birini vermektedir), Delta Kuralında sürekli (türevlenebilir) bir işlev kullanılmaktadır (–1 ile +1 arasındaki tüm sonuçlar alınabilmektedir) ve sezgisel yaklaşım güç kazanmaktadır.

Elimizde yapay sinir ağımızın öğrenebilme kabiliyetini ve öğrenme oranını ölçen bir hata maliyeti işlevi olduğunu düşünelim. Bunun için kullanılabilen örnek bir hata maliyeti işlevi aşağıda verilmiştir (Kröse, B. Van Der Smagt, P. 1996 ve Cummins, F., Schraudolph, N., 1999).

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - y_i)^2 \qquad [1.19]$$

Yukarıdaki işlev tek çıkışı olan bir ağ içindir.
Burada;
n : girdi örüntülerinin sayısını,
t_i : i. örüntü için istenen çıktı değerini,
y_i : i. örüntü için ağın verdiği çıktı değerini
göstermektedir.

Öğrenme sırasında ağırlık değerleri güncelleme işlemi aşağıdaki gibi yapılmaktadır.

$$W(yeni) = W(eski) - \mu \frac{\partial E}{\partial W} \quad [1.20]$$

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - y_i)^2 \quad [1.21]$$

$$\frac{\partial E}{\partial W} = \sum (t_i - y_i)x_i \quad [1.22]$$

Burada,

$W(yeni)$: Bir sonraki adımda kullanılacak ağırlık değerini,

$W(eski)$: Bir önceki adımda kullanılan ağırlık değerini,

μ : Öğrenme oranını (bu değer aşağıda ayrıntılı açıklanacaktır),

$\frac{\partial E}{\partial W}$: Hata maliyeti işlevinin ağırlık değerine göre kısmi türevini göstermektedir.

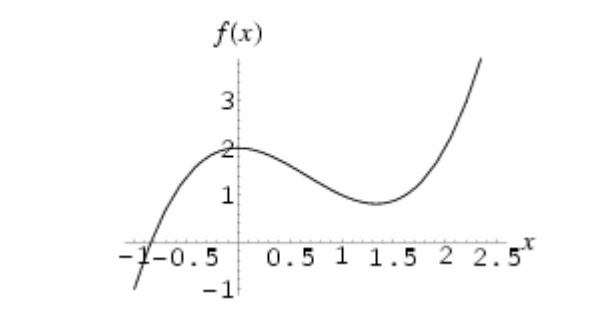
Ayrıca,

$$y = w.x \quad [1.23]$$

varsayımıyla kısmi türev alınmıştır.

Burada hatanın en küçük yapılmaya çalışılması yoluyla en doğru ağırlık değerine erişilmeye çalışılmaktadır. Hatanın en küçük yapılmaya çalışılması genellikle eğim düşümü (*gradient descent* ya da s*teepest descent*) yöntemi ile yapılmaya çalışılır. Bu yöntem, hata maliyeti işlevi üzerinde yerel en küçük değerleri bulmaya çabalar. Bunun için alınan noktadaki eğimin aksi yönüne doğru küçük adımlar atmak gerekmektedir (Cummins, F., Schraudolph, N., 1999 ve Wolfram, S., 2002).

Bu yöntem, aşağıdaki çizgesi verilen işlev için örneklenmiştir.



Şekil 1.9. Denklem 1.23'ün Geometrik Gösterimi

$$f(x) = x^3 - 2x^2 + 2 \qquad [1.24]$$

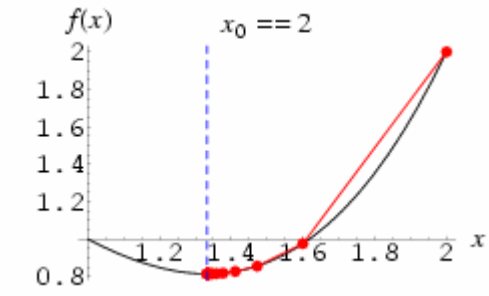
x_0 başlangıç noktası olsun, x_0 noktasından itibaren, $\varepsilon > 0$ yardımıyla istenen en küçük yerel noktaya aşağıdaki kural ile ulaşılır.

$$x_i = w_{i-1} - \varepsilon f'(x_{i-1}) \qquad [1.25]$$

Bu kural daha yukarıda verilen

$$W(yeni) = W(eski) - \mu \frac{\partial E}{\partial W} \qquad [1.26]$$

ağırlık güncelleme kuralı ile bağdaştırılırsa büyük resim rahatlıkla görülmektedir. $x_0 = 2, \varepsilon = 0{,}1$ alınırsa adımlar aşağıdaki gibi olacaktır.



Şekil 1.10. $x_0 = 2, \varepsilon = 0{,}1$ İçin Adımların Gösterimi

İlk adımda adım büyüklüğü -0.4, ikinci adımın büyüklüğü -0.128 olarak bulunur.

$x_1 = x_0 - 0.1\,(3x_0^2 - 4x_0)$ ise, $x_0 = 2$ yerine konulduğunda,

$x_1 = 2 - 0.1\,(4) = 1.6$ elde edilir.

$x_2 = x1 - 0.1\,(3x_1^2 - 4x_1)$, $x_1 = 1.6$ yerine konulduğunda,

$x_2 = 1.6 - 0.1\,(1.28) = 1.472$

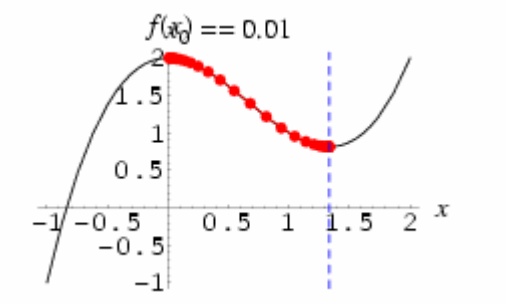
Adım büyüklüklerinin gittikçe azalması dikkate değerdir. Adım büyüklüğünü ayarlayan en önemli parametre, burada ε ile gösterilen ve aslında eğim düşümü yönteminde μ ile gösterilen öğrenme oranına karşılık gelen 0.1 değeridir. Bu parametre sayesinde adım büyüklüğü uyumlu bir şekilde en küçük yerel değere yaklaşmaktadır.

Yine burada kullanılan

$$f(x) = x^3 - 2x^2 + 2 \qquad [1.27]$$

işlevi ise eğim düşümü yöntemindeki hata maliyeti işlevine karşılık gelmektedir.

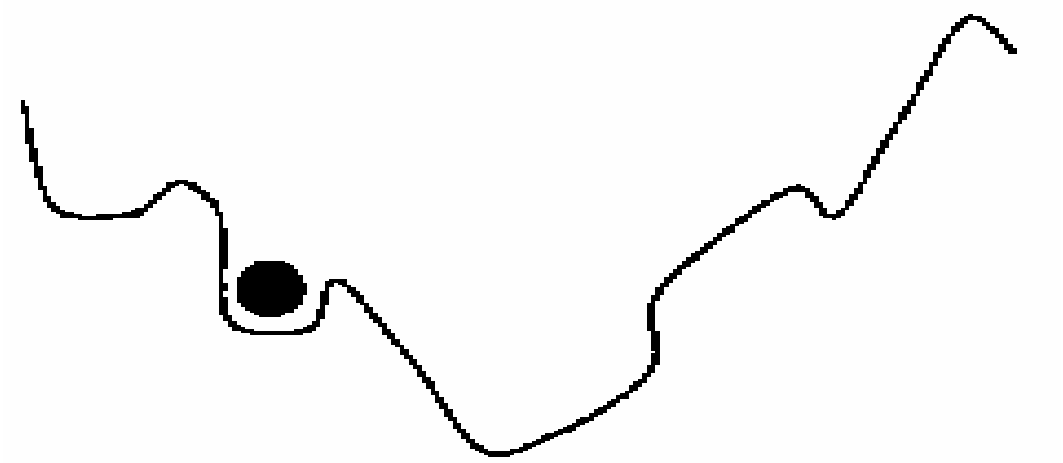
Aşağıda ikinci bir örnek çizge üzerinde $x_0 = 0.01$ ve $\varepsilon = 0.1$ değerleri için adımlar gösterilmiştir.



Şekil 1.11. $x_0 = 0{,}01$ ve $\varepsilon = 0{,}1$ değerleri için adımlar

1.6. Öğrenme Katsayısı ve Momentum

Eğim düşümü yönteminde mutlak en küçük değeri ararken yerel en küçük değerlere takılıp kalmak tehlikesi vardır. Gerçek bir uygulamada hata işlevleri çok karmaşık olabilir ve birçok yerel en küçük değere sahip olabilir (Cummins, F., Schraudolph, N., 1999).



Şekil 1.12. Yerel Minimuma Takılıp Kalma

Şekil 1.12.'de örneklenen bu durumdan kaçınmak için kullanılan yöntemlerden birisi, ağırlık değerini değiştirirken bir önceki ağırlık değişim miktarını momentum parametresi ile çarpıp yeni ağırlık değişimine katmaktır.

t. adımdaki ağırlık değişim miktarı daha önce aşağıdaki şekilde ifade ediliyordu.

$$\Delta w(t) = \mu(\frac{\partial E}{\partial w}) \qquad [1.28]$$

Bu sayede yeni ağırlık değeri Denklem 1.28'deki gibi ifade edilmekteydi.

$$W(yeni) = W(eski) - \mu(\frac{\partial E}{\partial w}) \qquad [1.29]$$

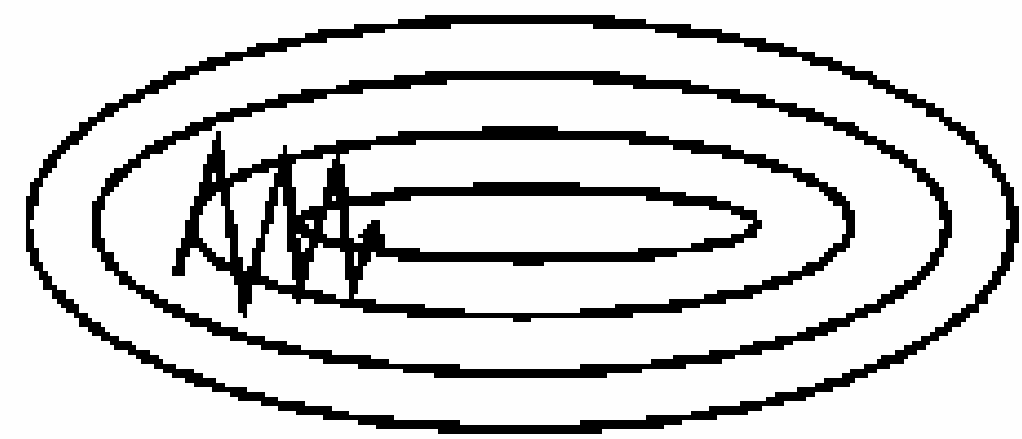
Ağırlık değişimine momentum parametresi eklendiğinde, değişim miktarı aşağıdaki gibi olmaktadır.

$$\Delta w(t) = \mu(\frac{\partial E}{\partial w}) + m\Delta w(t-1) \qquad 0 < m < 1 \qquad [1.30]$$

Bir önceki adımın değişim miktarı, momentum parametresi m ile çarpılıp yeni değişim miktarına eklenmektedir.

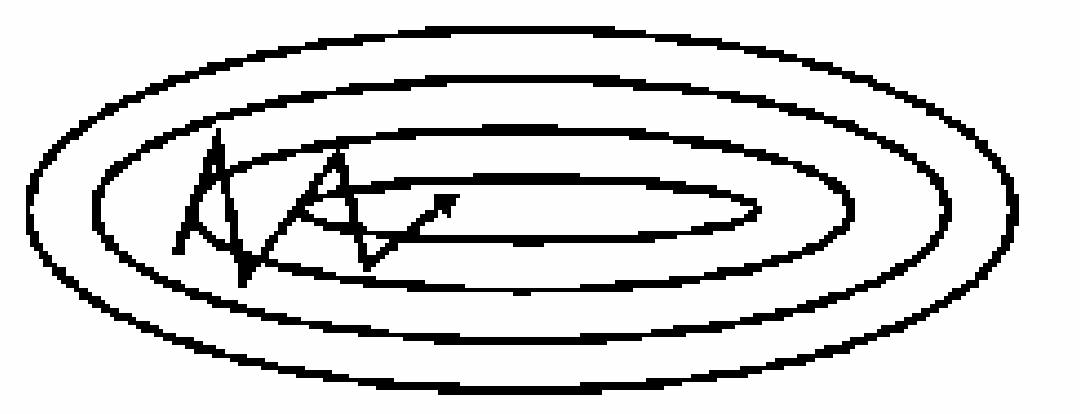
Momentum parametresinin en doğru değeri denemeler yoluyla bulanabilmektedir. Momentum parametresi yoluyla en küçük değere giden adımların büyüklüğü küçük bir miktar arttırılmaktadır. Bu artış miktarının daha uyumlu olabilmesi için öğrenme katsayısının düşürülmesi daha uygun olmaktadır. Eğer büyük bir öğrenme katsayısı (1' e yakın) ile yine büyük bir momentum katsayısı (1' e yakın) kullanılırsa adımlar çok büyüyecektir. Bu şekilde belki de mutlak en küçük değeri bulabilmek mümkün olamayacaktır.

Şekil 1.13.'te, hata yüzeyinde en küçük değeri ararken, momentum parametresi olmaksızın, ağırlık uzayındaki ilerlemeleri göstermektedir. Bu şekilde en küçük değeri bulmak çok uzun sürmektedir.



Şekil 1.13. Momentumsuz Arama (Cummins, F., Schraudolph, N., 1999)

Aşağıdaki şekil ise momentum parametresi yardımıyla en küçük değere nasıl ulaşıldığını göstermektedir. Bu sayede atılan adımlar ve salınım sayıları azalmakta ve en küçük değere daha çabuk ulaşılmaktadır.



Şekil 1.14. Momentum Kullanılarak Arama (Cummins, F., Schraudolph, N., 1999)

1.7. Öğrenme Algoritmalarının Uygulanış Biçimleri

1.7.1 Toplu İşlem Yolu İle Öğrenme (Batch Learning) ve Anında Öğrenme (Incremental Learning)

Öğrenme algoritmaları genel olarak iki biçimde uygulanmaktadır. Bunlar toplu işlem yapma yoluyla öğrenme ve anında öğrenme yöntemleridir.

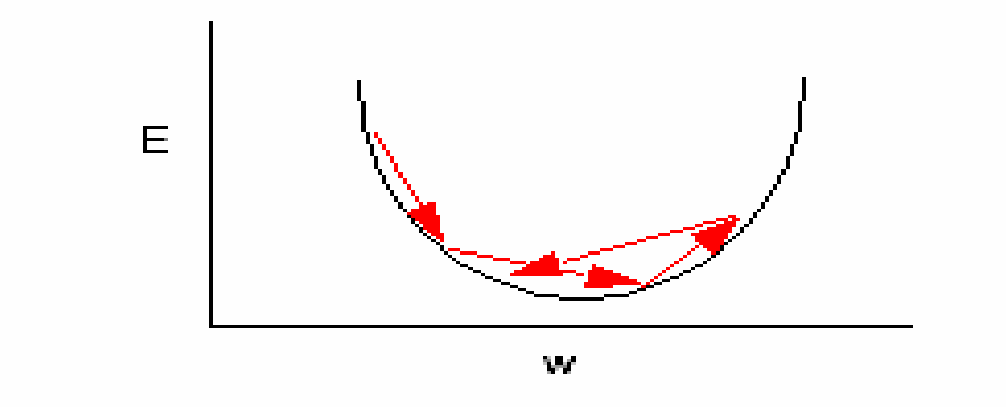
Anında öğrenmede, ilk değerleri atanan ağırlıklar, çalışma girdilerinin her bir tanesinin işlenmesinden hemen sonra güncellenmektedir. Böylece defalarca denenen girdi kümesindeki her bir girdi örüntüsü için bu güncelleme yapılmış olmaktadır.

Toplu öğrenmede ise, ağırlık değerleri tüm çalışma girdileri üzerinden her geçişte güncellenmektedir. Elbette tüm çalışma girdileri, defalarca yapay sinir ağında tetkik edilecek ve tüm girdiler her bittiğinde güncelleme yapılacaktır.

Bu iki yöntem arasında, toplu öğrenme yöntemi çok daha hızlı işlem yapmaktadır. Ancak bu yöntem gerçek zamanlı sistemlerde yerini anında öğrenmeye bırakmak zorunda kalmaktadır. Gerçek zamanlı bir sistemde, ağırlıklar sürekli değiştirilir ve sürekli öğrenme, girdilerin birer birer gelmesi ile sağlanır.

Bazı sistemlerde ise toplu öğrenme, tüm girdiler bitmeden, birkaç girdide bir ağırlık güncellemesi yapılarak uygulanır. Bu yöntem her iki yöntem bir nevi birleşimi gibidir ve "*mini-batch*" olarak anılmaktadır (Morse, P. M., Feshbach, H. 1953).

Toplu işlem yolu ile öğrenmenin bir üstün yönü de, hata yüzeyinde mutlak en küçük değeri ararken, ağırlık güncellemeleri sırasında oluşabilecek kayma ve yanlışlıkları (*gürültü - noise*) en aza indirmesidir. Bu kayma ve yanlışlıklar sebebiyle oluşabilecek yanlış salınımlar Şekil 1.15.'te örneklenmiştir (Cummins, F., Schraudolph, N., 1999).



Şekil 1.15. Mutlak En Küçük Değer (Cummins, F., Schraudolph, N., 1999)

Yapay sinir ağlarında, bu tip kaymaların ve yanlışlıkların sebepleri, gizli katmandaki düğüm sayısının doğru olmaması, düğümler arasındaki ağırlık bağlantıları sayısının doğru olmaması, gereğinden fazla deneme yapılması, kullanılan gizli katman sayısının yanlış seçilmesi olabilmektedir.

Anında öğrenme, bu sorunu, uygun bir adımdan sonra öğrenme katsayısı üzerinde değişiklik yapma yoluyla çözmektedir. Öğrenme katsayısındaki yapılan değişiklik aşağıdaki şekilde ifade edilmektedir.

$$\mu(t) = \frac{\mu_0}{(t - t_a)} \qquad [1.31]$$

Burada

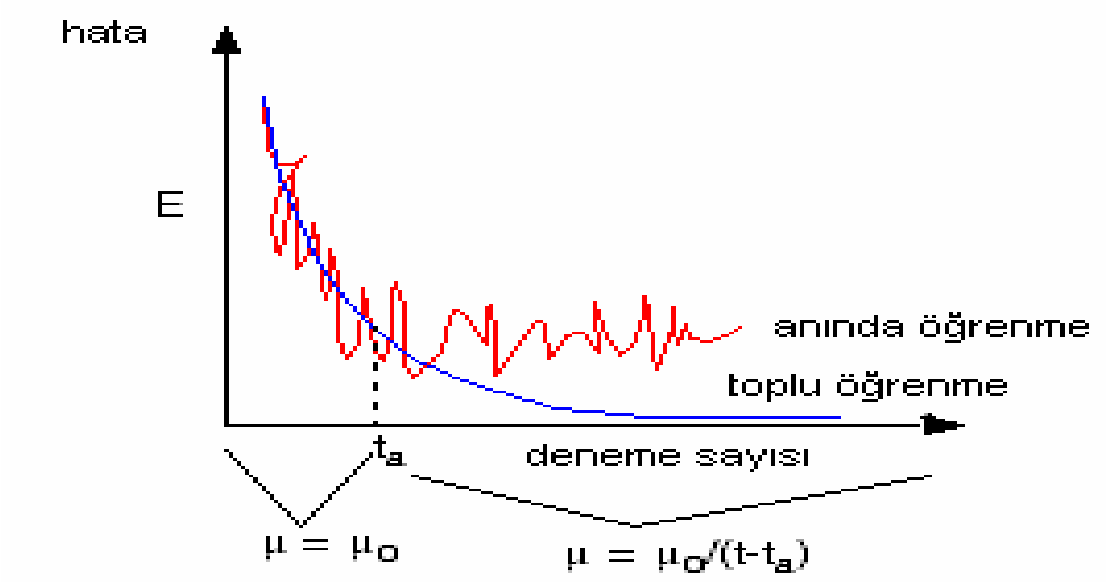
$\mu(t)$: t. adımdaki öğrenme oranı değerini,

μ_o : ilk öğrenme oranı değerini,

t_a : öğrenme oranının değiştirilmeye başladığı adımı,

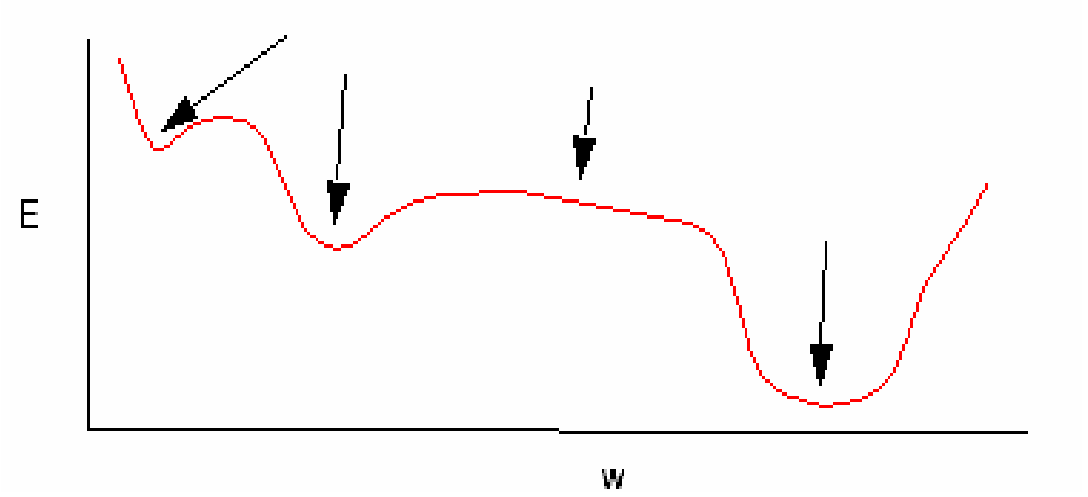
t : şu anki adımı göstermektedir.

Bu yöntemdeki öğrenme adımları sırasında ortaya çıkan hata oranının deneme sayısına bağlı çizgesi Şekil 1.16'da gösterilmiştir.

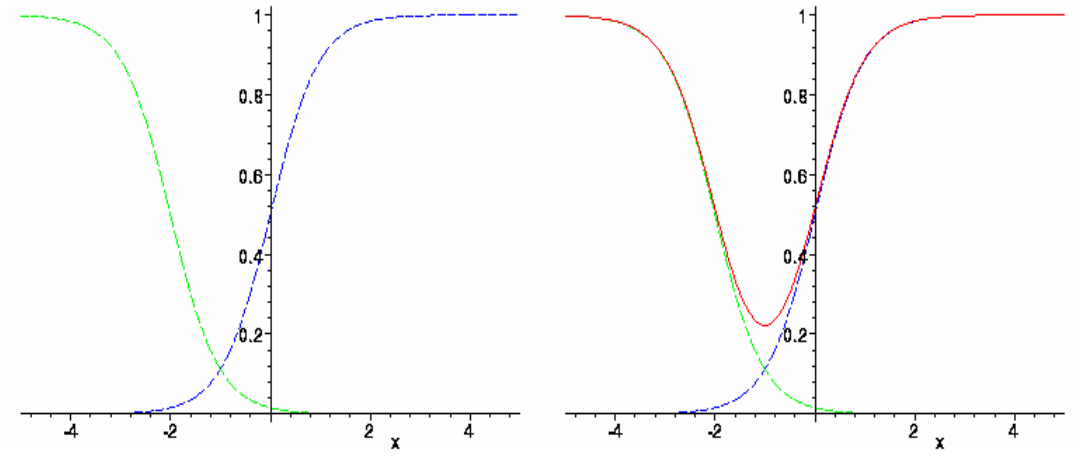


Şekil 1.16. Hata Oranının Deneme Sayısına Bağlı Çizelgesi (Cummins, F., Schraudolph, N., 1999)

Yapay sinir ağında oluşturulacak yapay bir gürültünün ise bazen faydası olabilmekte, sigma tabanlı işleve sahip iki gizli katmanın mutlak en küçük değeri bulabilmesi daha kolay olabilmektedir. Doğrusal olmayan bir ayrıştırma çizgisine sahip ağın hata eğrisi aşağıdaki gibi birden çok yerel en küçük değere sahip olabilmektedir. Böyle bir durumda iki sigma tabanlı gizli katmanın mutlak en küçük değere nasıl ulaştığı Şekil 1.18.'te gösterilmektedir.



Şekil 1.17. Yerel Minimum Noktalar

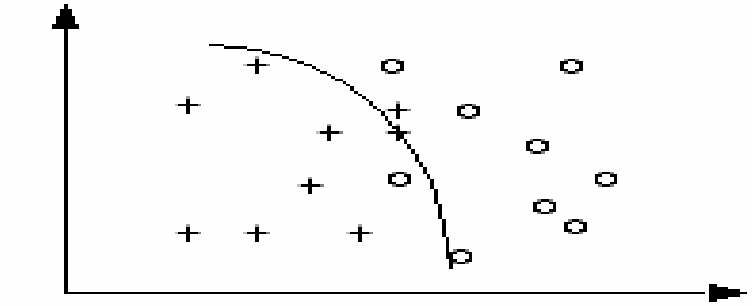


Şekil 1.18. İki Sigma Tabanlı Gizli Katmanın Mutlak En Küçük Değere Ulaşması

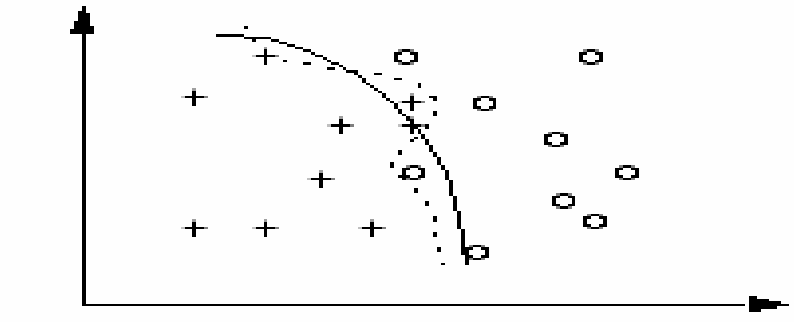
Burada yeşil sigma tabanlı bir katman ve mavi sigma tabanlı bir katman birlikte kırmızı çizgiyle gösterilen etkinleştirme işlevini oluşturmaktadır. Bu, anında öğrenmede yerel en küçük değere düşmeyi engelleyici bir durumdur.

1.7.2. Aşırı Öğrenme (OverTraining)

Çok katmanlı bir yapay sinir ağı, yeterli miktarda gizli katmana sahip ise yumuşak hatlara sahip, esnek bir işlev olabilir. Ancak bu esneklik sırasında, ağa karışan gürültüleri de öğrenmekte ve aşağıdaki Şekil 1.19.'daki gibi bir durum ortaya çıkmaktadır.



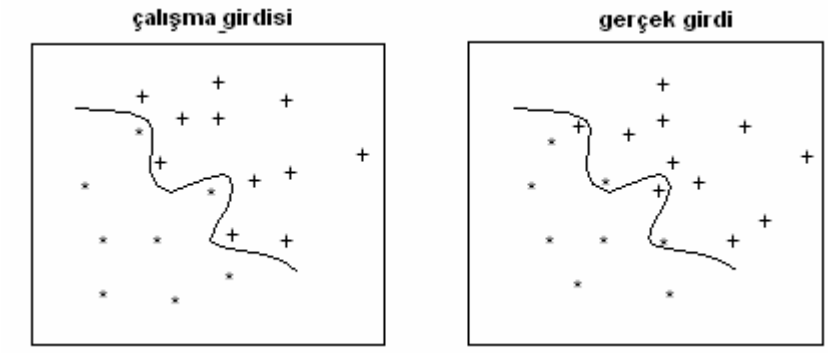
Şekil 1.19. Aşırı Gizli Katmanın Olmadığı Durum



Şekil 1.20. Gereğinden Fazla Gizli Katmanın Olduğu Durum

Şekil 1.20.'deki gibi bir durum, fazla sayıda gizli katman olduğunda ortaya çıkabilmektedir. Böyle bir durumda, bir yapay sinir ağının gürültüyü dahi öğrenmesi, aşırı öğrenmeyi göstermektedir.

Aşırı öğrenmenin yol açtığı bir başka durum, ağın daha önce karşılaşmadığı girdi kümesi için yanlış sonuçlar üretmesidir. Ağ, çalışma girdileri yardımıyla belirlenen ağırlıklar ile dış dünyadan gelen gerçek girdileri karşıladığında bazı yanlış sınıflandırma durumları ortaya çıkabilmektedir. Bu durum Şekil 1.21 ile gösterilmektedir.



Şekil 1.21. Aşırı Öğrenme(Cummins, F., Schraudolph, N., 1999 ve Efe, M.Ö., Kaynak, O., 2000)

Çalışma girdi kümesinde olmayan bir girdinin yanlış sınıflandırıldığı görülmektedir. Bu gibi durumlar aşırı öğrenmenin aslında her zaman iyi sonuçlar vermediğini örneklemektedir. Böyle durumları engellemek için kullanılan birkaç çözüm yolu vardır. Bunların en basiti öğrenmeyi erken durdurmadır (early stopping).

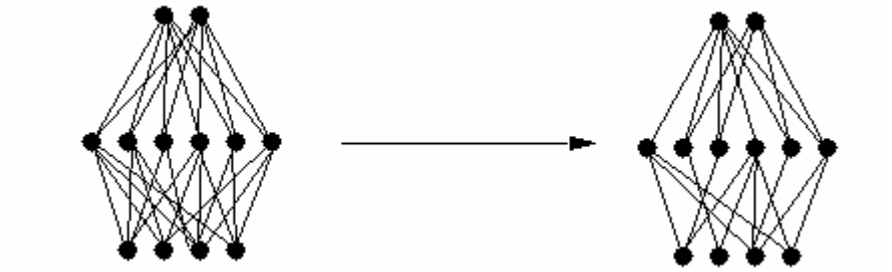
Birbirlerinden olabildiğince farklı oluşturulması gereken test ve çalışma girdileri üzerinde çalışırken test girdisi için hatanın artmaya başladığı noktada eğitim durdurulmalıdır. Bu durum Şekil 1.22'de gösterilmiştir.



Şekil 1.22. Durdurma Noktası (Cummins, F., Schraudolph, N., 1999)

Bir başka yöntem ise doğru gizli katman sayısını bulmak ve gizli katmandaki düğüm veya bu düğümlerden çıkan ağırlık bağlantıları sayısını azaltmaktır. Ancak gerçek girdilerin düzensizliği, bir ağın doğru düğüm sayısını ve ağırlık bağlantıları sayısını önceden kestirebilmeyi güçleştirmektedir.

Yapılacak şey, büyük bir ağ ile başlamak ve bu ağı yavaş yavaş, denemeler yardımıyla kırpmak ve doğru yerde durmaya çalışmaktan ibarettir. Bu yöntemlerden birisi "brain damage" olarak bilinmektedir ve Şekil 1.23'te gösterilmiştir.

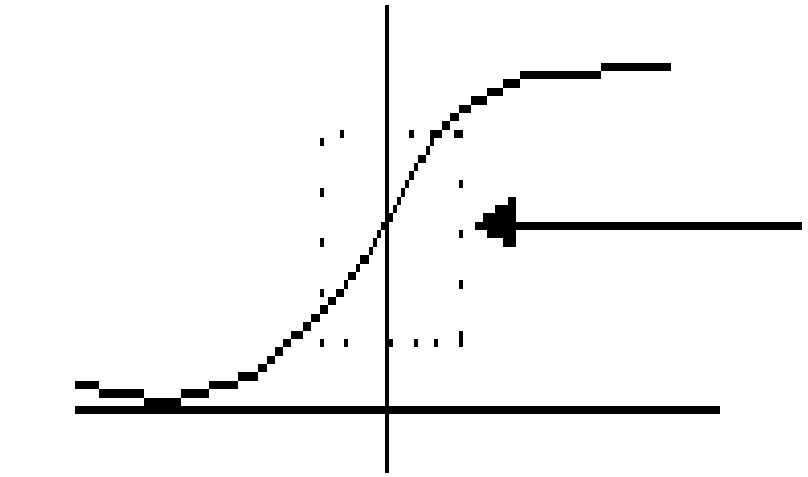


Şekil 1.23. "Brain Damage" (Cummins, F., Schraudolph, N., 1999)

Bir başka yöntem ise, ağdaki ağırlık güncelleştirmesi yöntemiyle oynama yoluyla çözüme ulaşmayı hedeflemektedir. Ağdaki ağırlık bağlantılarına momentum ekleme yönteminin bir anlamda tersi olan bu yöntem ise "weight decay" olarak anılmaktadır.

$$\Delta w(t) = \mu(\frac{\partial E}{\partial w}) - \gamma w \qquad [1.32]$$

Bu yöntem ağırlık değerinin sıfıra yakınsamasını sağlamaktadır. Böylece sigma tabanlı bir etkinleştirme işlevi doğrusal bir işlev gibi işlem görmektedir.



Şekil 1.24. Sigma Tabanlı Etkinleştirme İşlevinin Doğrusal İşlevi

1.8. Öğrenme Algoritmalarının Sınıflandırılması

Gözlem, eğitim ve hareketin var olanda meydana getirdiği değişiklik öğrenme olarak tanımlanabilir. Yapay sinir ağlarında ağırlıkların güncellenmesi ile sağlanan öğrenme metodlarını öğrenme algoritmalarına göre üç genel sınıfa ayrılmaktadır (Beşdok, E., Erler, M., Sağıroğlu, Ş. 2003 ve Efe, M.Ö., Kaynak, O., 2000).

1.8.1. Danışmanlı Öğrenme

Arzu edilen çıktı değerleri ile gerçek çıkış arasındaki farka göre ağırlıkların güncellenmesi sağlanır. Bu yaklaşımda mutlaka bir danışmana veya yapay sinir ağına ne öğrenmesi gerektiğini aktaracak bir yaklaşıma ihtiyaç vardır.

1.8.2. Danışmansız Öğrenme

Girişte verilen örnekten elde edilen çıkış bilgisine göre, ağ sınıflandırma kurallarını kendi kendine geliştirmektedir. Bu öğrenme algoritmalarında istenilen çıkış değerinin bilinmesine gerek yoktur. Giriş verileri arasındaki matematiksel ilişkilere göre bağlantı ağırlıkları ayarlanır. Aynı özellikleri gösteren örüntülerde aynı çıkışlar, farklı çıkışlarda ise yeni çıkışlar oluşturulur.

1.8.3. Takviyeli Öğrenme

Giriş değerlerine karşı istenilen çıkış değerlerinin bilinmesine gerek yoktur. Yapay sinir ağına bir hedef verilmemekte fakat elde edilen çıkışın verilen girişe karşılık uygunluğunu değerlendiren bir kriter kullanılmaktadır.

1.9. Tezin Amacı

Sağlık bilimlerinin yeni yeni uygulamalarına dahil ettiği problem çözme tekniklerinden birisi de yapay sinir ağlarıdır. Diğer tüm yöntemler gibi bu yöntemin de gelişen teknolojiye, bilgisayar teknolojilerine, ayak uydurması uygulama alanın genişlemesi bakımından gerekli idi. Tezin tamamlanması ile uygulama veritabanı üzerinden yapay sinir ağlarının kullanılacağı uygulamaların ve tanıtılan Joone paket programının kullanımının yaygınlaşması amaçlanmaktadır. Joone paket programının seçilmesinin en önemli sebepleri, paket programının ücretsiz olması, görsel gösterimler bakımından zengin ve kendi kütüphanelerinin kullanılması sonucunda uygulama alanının genişletilebilir olmasıdır. Bu amaca yönelik olarak sırt ağrısı olan ve ameliyat teşhisi konan hastaların izlenerek şikayetlerinin nüks edip etmediğinin araştırıldığı gerçek uygulama veritabanı kullanılarak, ameliyat öncesinde hastanın şikayetleri ve bazı özelliklerinden faydalanılarak oluşturulacak yapay sinir ağı ile ameliyat sonrası nüks durumları doğrulanmaya çalışılacaktır.

2. GEREÇ VE YÖNTEM

Bu uygulamada YSA teknikleri kullanılacaktır. Uygulama veritabanı için, sekiz bağımsız değişken fizik tedavi ve rehabilitasyon uzmanlarının tavsiyeleri göz önünde tutularak belirlenmiş ve 01.01.2005 – 06.30.2006 tarihleri arasında Ankara Numune Hastanesi, Fizik Tedavi ve Rehabilitasyon Kliniğine sırt ağrısı şikayetiyle başvuran, fıtık teşhisinin tedavisi için ameliyat kararı alınan ve ameliyat sonrası durumları kayıt altına alınan 67 kişinin verileri kullanılacaktır.

2.1. Joone (Java Object Oriented Neural Engine)

Açık kaynak kodlu bir yazılım olan Joone, yapay sinir ağları ile yapay zeka uygulamaları geliştirmek amacıyla tasarlanmış bir nesne tabanlı uygulama geliştirme çatısı ve kütüphanesidir. Hali hazırda yazılmış bir görsel kullanıcı arayüzüne (GUI – Graphical User Interface) de sahiptir. Bu arayüz yardımıyla istenen amaca göre son derece hızlı ve kolay bir şekilde hazırlanan yapay sinir ağı modeli, çalıştırılıp test edilebilmektedir. Ardından ayrıştırılmış (serialized) olarak kaydedilip gömülü (embedded) java uygulamalarında rahatlıkla kullanılabilmektedir. Bu şekildeki bir java uygulaması örneklenecektir. Ayrıca joone kütüphanelerini kullanarak java kaynak kodlu yapay sinir ağı uygulamaları da geliştirmek mümkün olabilmektedir.

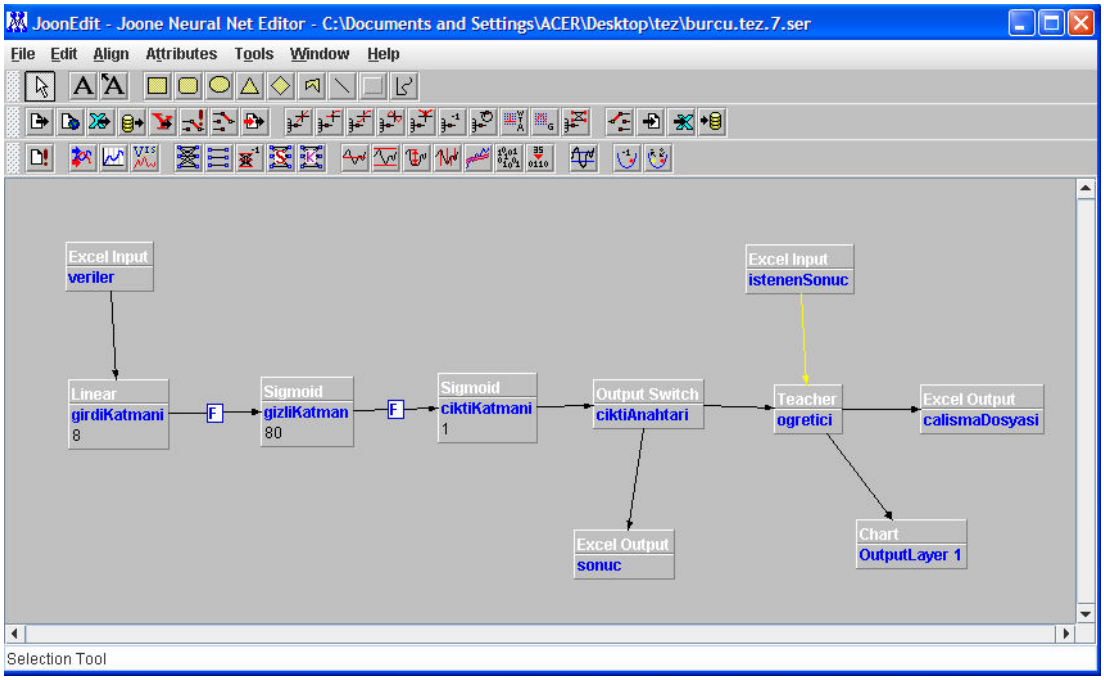
2.2. Joone Kurulması:

http://sourceforge.net/projects/joone ya da www.jooneworld.com adreslerinden indirilebilen joone hakkında her türlü detaylı bilgiye de aynı adreslerden erişebilmek mümkündür. Yapılan çalışmada, indirilen ".exe" uzantılı dosya çalıştırılarak üzerinde "Windows XP Home Edition" işletim sistemi ve "Java Sanal Makinesi (JVM - Java Virtual Machine)" kurulu olan bir bilgisayara "joone editor 1.2.1" kurulmuştur. Editör indirilirken, eğer kurulum yapılacak bilgisayarda JVM kurulu değil ise

editörün JVM ile birlikte olan sürümü indirilmelidir. Uygulama 632 MB ana belleğe sahip, P4 2.00 GHz işlemcili bir bilgisayarda gerçekleştirilmiştir. Editörün rahatlıkla kullanabilmesi için en az 128 MB erişilebilir hafıza olması gerekmektedir.

2.3. Sırt Ağrısı Yapay Sinir Ağının "Joone Editor" Kullanılarak Hazırlanması

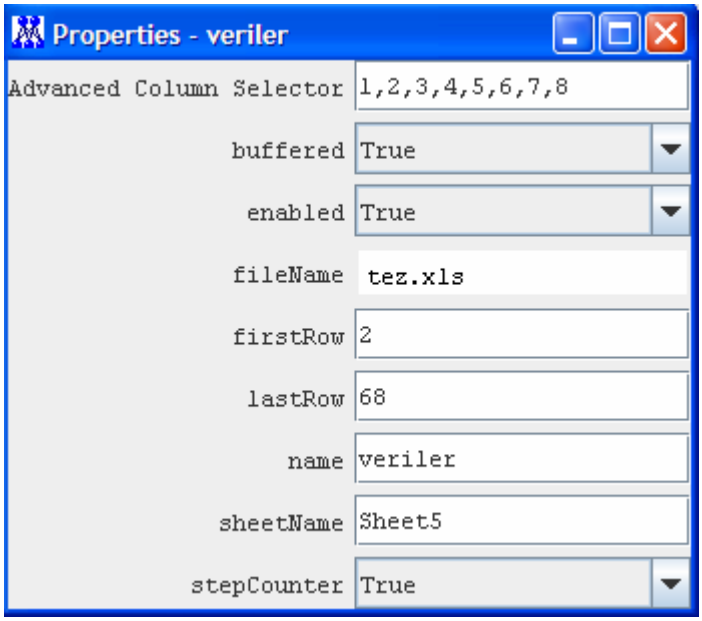
Sırt ağrısı yapay sinir ağı, girdi katmanı, gizli katman, çıktı katmanı, çıktı anahtarı, Excel girdisi, öğretici, Excel çıktısı bileşenlerinden oluşmaktadır. Bu bileşenler editör tarafından ".ser" uzantılı bir dosyaya kaydedilmektedir. Oluşturulan yapay sinir ağı geri beslemeli bir ağdır.



Şekil 2.1. Sırt Ağrısı Yapay Sinir Ağı

2.4. Kullanılan Bileşenler ve Özellikleri

veriler (Excel Input Bileşeni) : Girdi verisinin (bağımsız değişkenler) bir excel dosyasından alınabilmesi için kullanılmaktadır. “Input Layer” bileşenine bağlanan “excel input” bileşeninin özellikleri aşağıdadır.



Şekil 2.2. Excel Girdisinin Tanımlanması

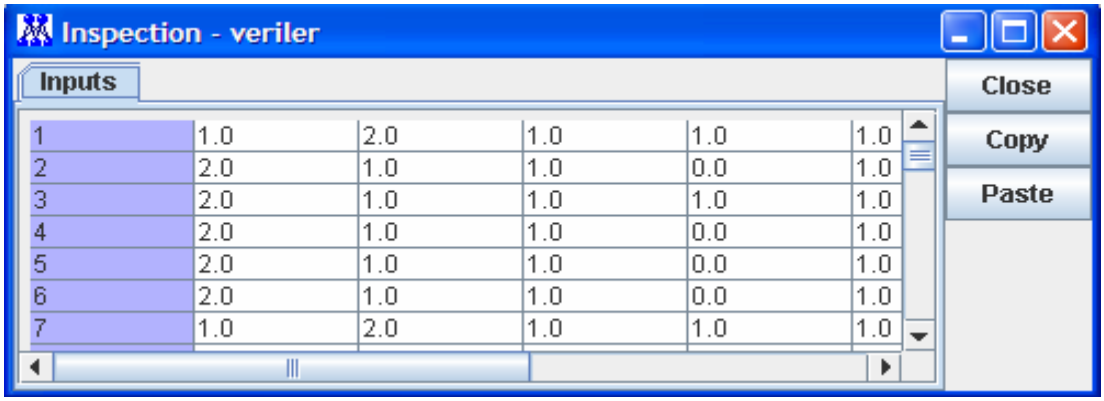
“Advanced Column Selector” özelliğinin değeri excel dosyasındaki ilk 8 sütunu seçebilecek şekilde “1,2,3,4,5,6,7,8” olarak atanmıştır. Bu ilk 8 sütun girdi dosyasındaki girdi örüntülerinin bulunduğu sütunlardır. Bu özellik “1-8” şeklinde de düzenlenerek, yine ilk sekiz sütünun seçilmesi sağlanabilir.

“fileName” özelliğine verilen dosya adı tam adres olarak da verilebilir. Burada sadece dosya adı verilmiştir. Bu durumda tez.xls dosyasi “.ser” uzantılı dosyanın bulunduğu dizinde aranmaktadır.

“firstRow” özelliği girdi örüntülerinin girdi dosyasında hangi satırdan başladığını gösterir. “lastRow” özelliğinin “firstRow” dan küçük olması, dosyanın sonuna kadar okunacağını belirtir.“sheetName” özelliği ile verilerin hangi excel çalışma sayfasında

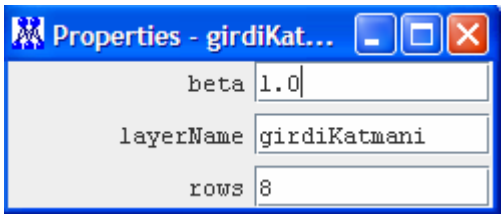
olduğu belirtilebilir. “buffered” özelliğinin “True” değerine sahip olması verinin girdi dosyasından tek seferde okunup ana belleğe alınmasını sağlar.

Bu noktadan sonra “girdiKatmani” üzerinde sağ tuşla elde edilen menüden “inspect” seçeneği seçilirse girdilerin aşağıdaki gibi bir önizlemesi yapılabilir.



Şekil 2.3. Verilerin Kontrol Edilmesi

girdiKatmani (Linear Layer) : Girdi katmanı bir “Linear Layer” bileşeni kullanılarak oluşturulmuştur.



Şekil 2.4. Girdi Katmanının Özelliklerinin Belirlenmesi

“rows” özelliği ile girdi katmanındaki örüntü çeşidi sayısı verilmiş, bu yolla yapay sinir ağının girdi katmanındaki düğüm sayısı da belirlenmiştir. “Linear Layer” bileşeni birden fazla giriş ve çıkış düğümüne sahip olduğundan bir anlamda çok değişkenli çoklu doğrusal regresyon yapmaktadır (www.jooneworld.com).

Beta parametresi örneğimizde 1.0 olduğundan girdi değerleri sinir ağına herhangi bir değişikliğe uğramadan girmektedir. Bu girdi katmanı için kullanılan aktivasyon fonksiyonu aşağıdaki gibidir.

$$y = \beta \cdot x \qquad [2.1]$$

gizliKatman (Sigmoid Layer) : Gizli katman bir "Sigmoid Layer" bileşeni kullanılarak oluşturulmuştur.



Şekil 2.5. Gizli Katmanının Özelliklerinin Belirlenmesi

"rows" özelliği ile gizli katmandaki düğüm sayısı belirlenmiştir.

ciktiKatmani (Sigmoid Layer) : Çıktı katmanı bir "Sigmoid Layer " bileşeni kullanılarak oluşturulmuştur.



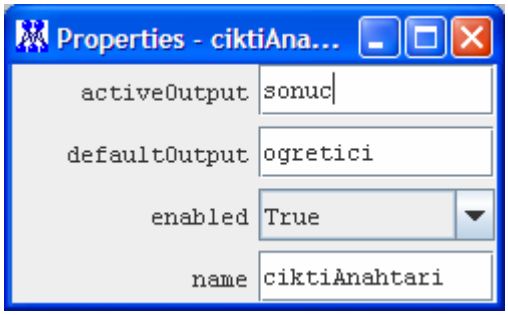
Şekil 2.6. Çıktı Katmanının Özelliklerinin Belirlenmesi

"rows" özelliği ile çıktı katmanındaki düğüm sayısı belirlenmiştir. Bu sayede 3 farklı sonuc elde edilebilmektedir.

"Sigmoid Layer" bileşeni içindeki tüm düğümler aşağıdaki şekilde bir lojistik aktivasyon fonksiyonu kullanmaktadır. Uygulamamızda çıktı birimlerinin sayısı birden fazla olduğu için, çıktı katmanı bir anlamda çok değişkenli çoklu lojistik regresyon yapan bir algılayıcıdur.

$$y = \frac{1}{1 + e^{-x}} \qquad [2.2]$$

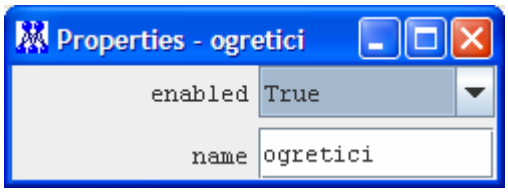
ciktiAnahtari (Output Switch) : Bu bileşen yardımıyla yapay sinir ağındaki öğrenme ve test etme anahtarlaması yapılmaktadır.



Şekil 2.7. Çıktı Anahtarının Özelliklerinin Belirlenmesi

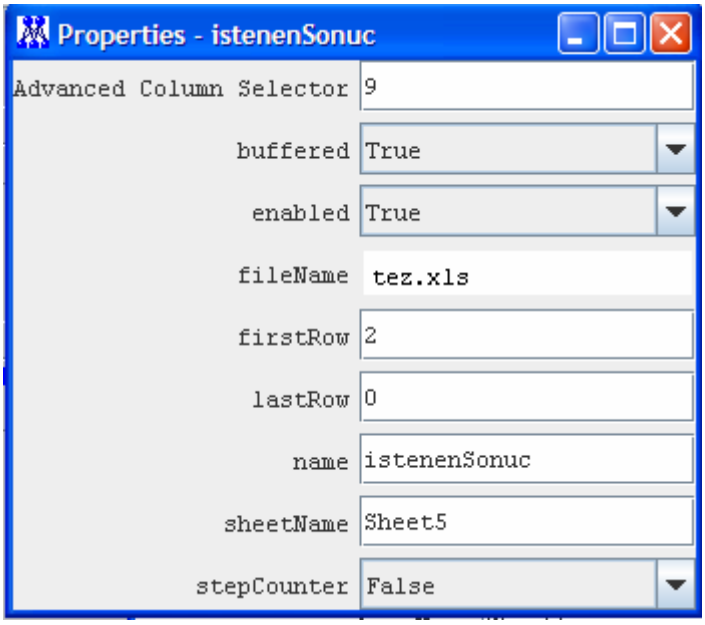
"activeOutput" özelliği "ogretici" yi gosterdiginde yapay sinir ağı öğrenme durumundadır ve çıktı değerleri "ogretici" yardımıyla "calismaDosyasi" na aktarılır. Eğer "sonuc" bileşeni işaret edilirse, yapay sinir ağı ögrenme durumundan test edilme durumuna geçer ve çıktı değerleri "sonuc" bileşenine aktarılır.

ogretici (Teacher) : Bu bileşen istenen değerler (bağımlı değişkenler) ile çıkan değerleri (tahmin değerleri) karşılaştırıp aralarındaki farkları hesaplar. RMSE'yi her adımda bularak "calismaDosyasi" na yazdırır.



Şekil 2.8. Öğreticinin Özelliklerinin Belirlenmesi

istenenSonuc (Excel Input) : Bu bileşen yardımıyla istenen sonuçlar "ogretici" ye aktarılır. Uygulamamızda girdi verileri (bağımsız değişkenler) ve çıktı verileri (bağımlı değişkenler) aynı dosyada farklı sütunlarda yer almaktadır.



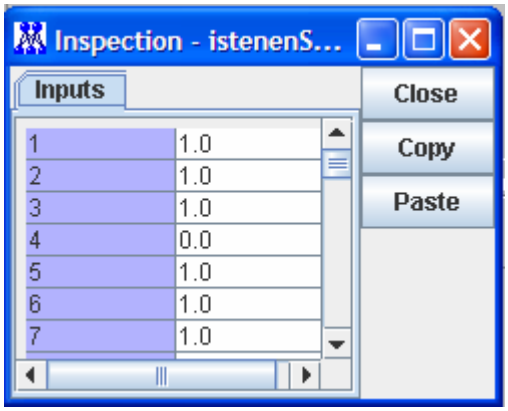
Şekil 2.9. İstenen Sonucun Özelliklerinin Belirlenmesi

"Advanced Column Selector" özelliği "input.xls" dosyasının 9 nolu sütununun, istenen değerlere sahip olduğunu göstermektedir.

Daha önce girdi verileri için "veriler" isimli bileşenin özellikleri açıklanırken "firstRow", "lastRow", "sheetName" gibi özelliklere değinildiğinden burada tekrar değinmeye gerek görülmemiştir.

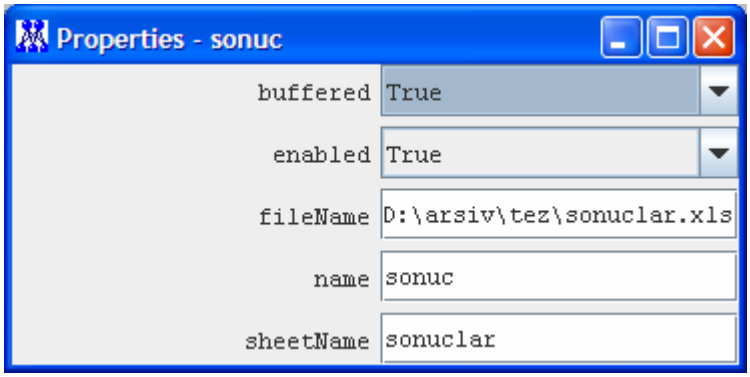
"istenenSonuc" ile "ogretici" arasındaki bağlantı diğer bağlantıların aksine gösterilen ok yönünde değil aksi yönde ve "ogretici" üzerindeki küçük pembe karenin üzerine basılı tutulup "istenenSonuc" üzerine sürüklenip bırakılması yoluyla elde edilir. Bu bağlantı ayrıca renk olarakta farklı bir renk ile belirtilmiştir.

Yine girdi verileri gibi istenen sonuçların da önizlemesine aynı yolla ("inspect" seçeneğiyle) ulaşmak mümkündür.



Şekil 2.10. İstenen Sonuçların İzlenebilmesi

sonuc (Excel Output) : Test aşamasında elde edilen sonuçların bir excel dosyasına aktarılmasını sağlar.



Şekil 2.11. Sonuçların Yazılacağı Dosyanın Belirlenmesi

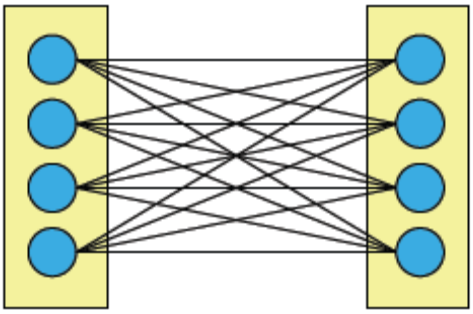
girdiGizli bağlantısı (Full Synapse) : Bu bağlantı bileşeni aslında yapay sinir ağının hafıza bileşenidir. İki katman arasında paylaşımlı bir hafıza bölgesi oluşturarak veri aktarımı sağlar. Birinci katmanın veriyi bu hafıza bölgesine yazması ile ikinci katmanın okuması, katmanlar arasında eşgüdümlü olarak gerçekleşir.



Şekil 2.12. Hafıza Bileşeninin Özelliklerinin Girilmesi

Burada belirtilmesi gereken önemli bir nokta “loopBack” özelliğinin ismi dolayısıyla “back propagation” yöntemini çağrıştırmasıdır. Aslında geri beslemeli bir yapay sinir ağı kurabilmek için bağlantının “loopBack” özelliğinin “false” olması ile “true” olması arasında hiç bir ilişki yoktur. Bu değer eğer bağlantılar arasında bir döngü oluşmuş ise editör tarafından otomatik olarak kırmızı renkle gösterilip değeri “true” yapılmaktadır. Joone ile kurulan bir yapay sinir ağı varsayılan olarak geri beslemelidir.

Bir “Full Synapse” bileşeni, iki katmanı aşağıdaki gibi düğümler arasındaki tüm kombinasyonlar ile bağlar.



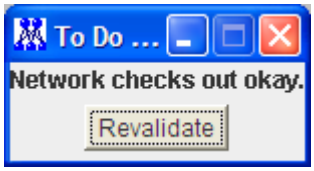
Şekil 2.13. Tüm Kombinasyonların Gerçeklenmesi (www.jooneworld.com)

gizliCikti bağlantısı (Full Synapse) : Bu bağlantı gizli katman ve çıktı katmanı arasındaki bağlantıdır.



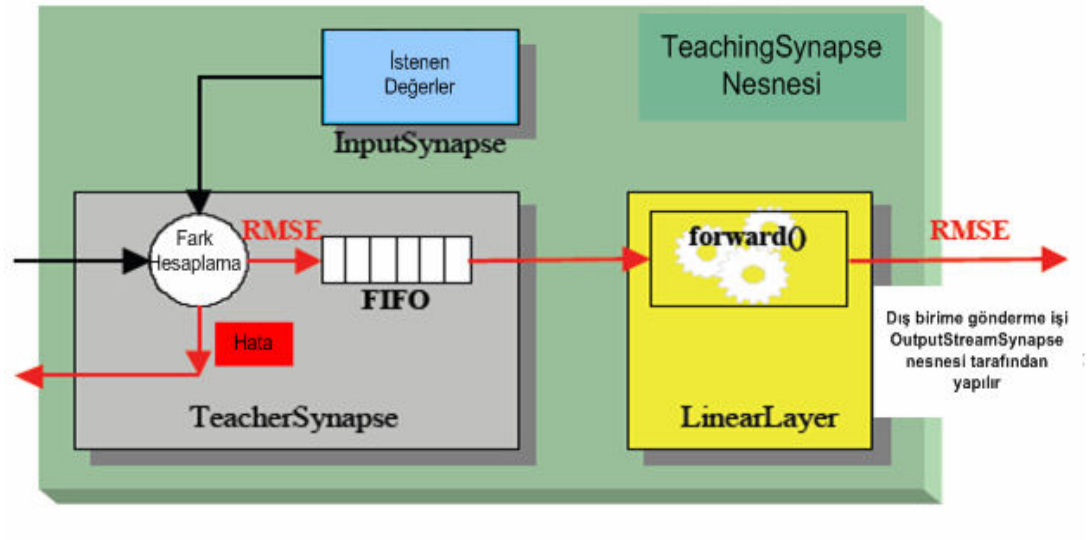
Şekil 2.14. Gizli Çıktı Bağlantısının Özellikleri

Yapılması gerekenler listesi (to do list) : Yapay sinir ağı kurulduktan sonra “tools” menüsünden “To Do List” seçeneği seçilir. Eğer her şey yolunda ise aşağıdaki gibi bir mesaj belirir. Bu aşamadan sonra yapay sinir ağını eğitmeye ve ardından test etmeye başlanılabilir.



Şekil 2.15. Kontrol Sonuç Penceresi

Aşağıda buraya kadar olan işleyişin bir özeti niteliğinde olan gösterim Şekil 2.16'da verilmiştir.

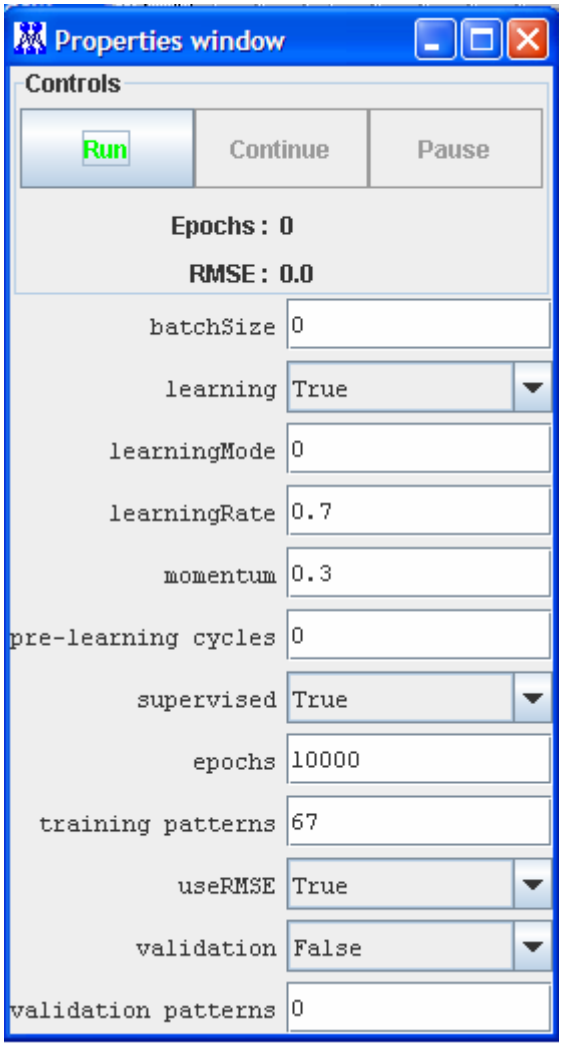


Şekil 2.16. İşleyişin Özet Gösterimi (www.jooneworld.com)

Çizgeden de anlaşılabileceği gibi hata hesaplama işini "teacher" bileşeni yapar ve bu bileşen kendisinden önceki bileşen olan "output layer" bileşeni ile çift yönlü iletişimdedir. Hata, her örüntüden sonra geri besleme mantığına uygun olarak önceki katmanlara aktarılır. Ayrıca dış bir uygulamanın da hata hesaplamalarına erişebilmesi için bu bileşenden sonra konulacak bir "Linear Layer" bileşenine de kendi içerisindeki bir FIFO (First In First Out) mekanizması ile aktarma yapabilmektedir.

2.5. Hazırlanan Yapay Sinir Ağının Eğitilmesi

"Tools" menüsünden "Control Panel" açılır.



Şekil 2.17. YSA Eğitimi İçin Gerekli Düzenlemeler

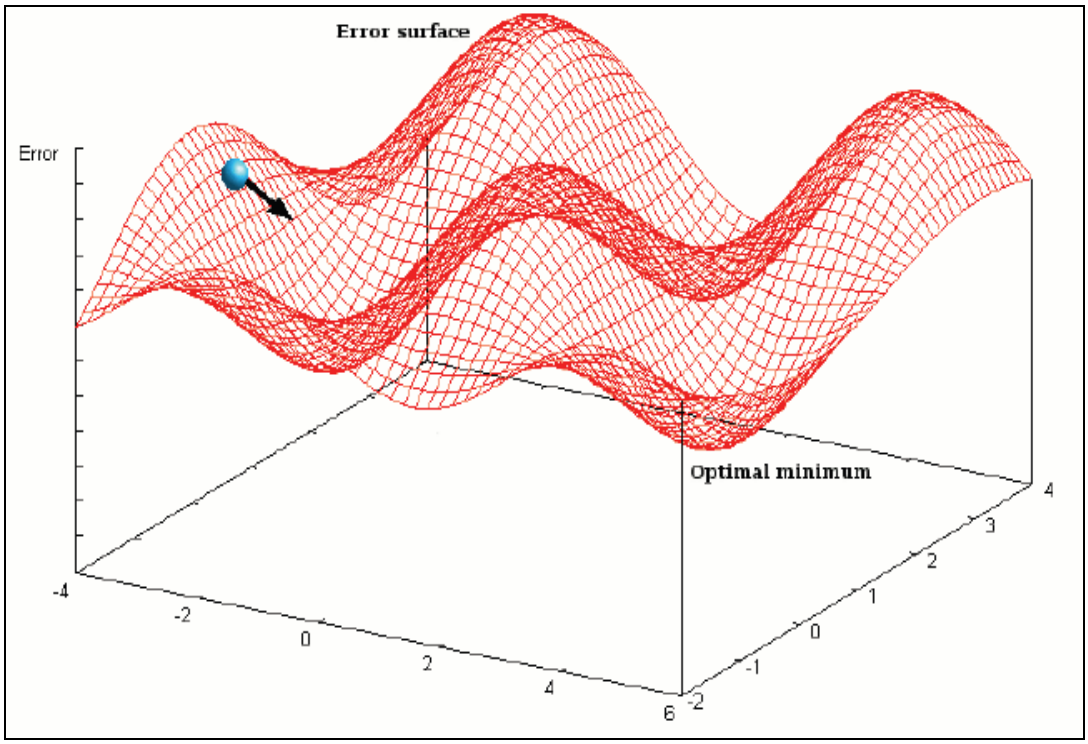
Parametreler şekildeki gibi hazırlanıp run seçeneğine tıklanmalıdır. Öğrenme işlemine başlamadan önce "ciktiAnahtari" nin "Active Output" özelliği "ogretici" olarak atanmalıdır. Kontrol paneldeki ayarlar tüm katmanlar ve sinaps bağlantıları için geçerlidir. "learning" değeri "true" olduğunda öğrenme modunun açık olduğu anlaşılmaktadır. "epochs" özelliği yapılacak deneme sayısısını belirlemektedir. "training patterns" ise girdi dosyasındaki örüntü sayısıdır.

2.6 Öğrenme Yönteminin Seçilmesi

"learning mode" özelliği ile joone kütüphanesinde tanımlı olan 3 değişik öğrenme metodundan biri veya yazılımsal olarak yazılacak yeni bir öğrenme yöntemi seçilebilir. Bu değere karşılık gelen öğrenme yollarının ayrıntıları aşağıda verilmiştir. Bu algoritmalar kılavuzlu (supervised) öğrenme için kullanılmaktadır. Burada açıklanan algoritmaların ve öğrenme oranı, momentum gibi parametrelerin teknik detayı Bölüm 1.7 de verilmiştir.

2.6.1. Geri Bildirimli Anında Öğrenme ("The Basic On-line Backprop Algorithm")

Bu yöntem en çok kullanılan yöntemdir. "learning Mode" özelliğinin 0 yapılması gerekir. Katmanlardaki bias değerleri ve sinaps bağlantılarındaki ağırlık değerleri geri beslemeli olarak, her girdi verisinden sonra tekrar tekrar hesaplanır ve güncellenir ve bu değerlerin optimum kombinasyonu bulunmaya çalışılır. Bir başka deyişle eğri ya da yüzey üzerindeki en küçük noktalar ve mutlak en küçük nokta bulunmaya çalışılır. Bu arayış aşağıdaki şekilde betimlenmiştir.



Şekil 2.18. Yüzeyde Mutlak Minimum Arayışının Betimlenmesi (www.jooneworld.com)

Bu algoritmanın çalışabilmesi için "learning rate" ve "momentum" parametrelerinin girilmesi gereklidir. "learning rate", yani öğrenme katsayısı, eğri üzerindeki minumum nokta arayışı sırasındaki adım büyüklüğünü tarifler, "momentum" ise bu adım büyüklüğünün salınım büyüklüğünü belirler. Daha duyarlı hale gelmesini ve sağa sola sapmaların ayarlanmasını sağlar. Geri besleme sırasında, ağırlıklardaki değişimin büyüklüğü bu iki parameterenin değeri ile belirlenir ve en uygun değer kombinasyonu ancak birkaç deneme yapıldıktan sonra belirlenebilir.

2.6.2. Geri Bildirimli Toplu Öğrenme ("The Batch BackProp Algorithm")

Bu yöntem yukarıda anlatılan yöntemin bir fark dışında aynısıdır. Geri besleme işlemi her örüntüden sonra değil tüm örüntüler bittikten sonra yapılır.

Bu öğrenme yolunun seçilebilmesi için "learningMode" özelliği 1 yapılmalıdır.

2.6.3. Esnek Geri Bildirimli Toplu Öğrenme (“The Resilient BackProp Algorithm”)

Bu yöntem de “Geri Bildirimli Toplu Öğrenme” gibi toplu işlem yapmaktadır. Ancak geri besleme sırasında eğimin büyüklüğü değil işaretini dikkate almaktadır. Bu algoritma “Geri Bildirimli Toplu Öğrenme” algoritmasından daha duyarlı hesaplamakta ve “learning rate” ile “momentum” parametlerine ihtiyaç duymamaktadır.

Bu öğrenme yolunun seçilebilmesi için “learningMode” özelliği 2 yapılmalıdır. Ayrıca toplu işlem yapan algoritmalar ilk anlatılan “Anında Geri Bildirimli Öğrenme” algoritmasından daha hızlı işlem yapar ve dağıtık bir bilgisayar sisteminde ağ üzerindeki trafiği fazla meşgul etmemektedir.

2.7. Öğrenme Sonrası

Öğrenme işlemi sırasında “çalışma.xls” dosyasına, hesaplanan RMSE değerleri bilgi amaçlı olarak yazdırılır. Artık sistem öğrenmeyi tamamlamıştır ve test edilmeye hazırdır. Güncellenen ağırlıklar ile yeni gelecek olan örüntüler test edilebilir.

2.8. Hazırlanan Ağın “Joone Editor” İçerisinde Test Edilmesi

Test etmek için ilk yapılması gereken, “Output Switch” bileşeninin “activeOutput” özelliğinin değerinin “sonuc” yapılmasıdır. Bu sayede sistemeden alınan test sonucları “sonuclar.xls“ dosyasına yazdırılacaktır.

Daha sonra “tools - control panel” seçeneği ile açılan pencerede “learning” değeri “false” yapılmalıdır. Test işleminin bir defa yapılması yeterli olacağından “epochs” değeri 1 yapılmalıdır. Daha sonra “run” komutu yardımıyla test işlemi başlatılır.

2.9. Joone Editörü İle Hazırlanmış Bir Yapay Sinir Ağına Bir Java Programından Erişilmesi ve Bu Program Yoluyla Test Edilmesi

Joone kütüphanesi bir java programı yoluyla yapay sinir ağı oluşturma, eğitme ve test etmeye imkan verecek şekilde hazırlanmıştır. Aslında "joone editor" yazılımı da bu kütüphane kullanılarak yazılmış bir java programıdır ve yapay sinir ağı hazırlamayı görsel bir hale getirmiştir. Yani "joone editor" ile yapılan her işlemin bir java kaynak kodu karşılığı çok kolaylıkla yazılabilir. Bu sayede yapay sinir ağı üzerinde daha fazla hakimiyet elde etme şansı olur. Gerçekleşen her olay esnasında ve olay sonrasında çalışan ağa müdahale etmek, ağdan bilgi almak ve bu bilgileri değerlendirmek gibi pek çok işlem yazılacak bir java kaynak koduyla rahatlıkla yapılabilir. Bu konuda ayrıntılı bilgi joone kurulumu anlatılırken verilen web adreslerinden elde edilebilir.

Ek-1'de Joone ile hazırlanmış ve hazırlanış aşaması yukarıda anlatılan yapay sinir ağına, bir java programı ile nasıl erişilebileceğine ve bu ağın tek bir örüntü yada birden fazla örüntü ile nasıl test edilebileceğine değinilmiştir. Yapay sinir ağı üzerinde hiç bir değişiklik yapmayan bu program, bir girdi dizisi ile ağı besleyen düzeneğin kurulumunu örneklemektedir.

2.10. Sırt Ağrısı

2.10.1. Ağrı Bilimi

İkinci dünya savaşından sonra önemli gelişmeler olmuştur. Bugün ağrı biliminin yani algolojinin kurucususu sayılan John Bonica ilk ağrı kliniğini 1946'da Amerika Birleşik Devletleri'nde kurmuştur (Kanbir, O., 2004). Daha sonra hızla gelişen bu bilim dalı ülkemizde de ilk olarak İstanbul Üniversitesi Tıp Fakültesi bünyesinde kurulmuştur.

Uluslararası Ağrı Araştırmaları Teşkilatına göre "Ağrı, vücudun herhangi bir yerinden kaynaklanan organik bir nedene bağlı olan veya olmayan, insanın geçmişteki deneyimleriyle de ilişkili olan, hoş olmayan öznel bir duygudur."

2.10.2. Ağrı Sınıfları

Ağrı, kaynaklandıkları yere göre; baş, diş, sırt veya bel, süreye göre; akut ve kronik olarak sınıflandırılmaktadır. Akut ağrılar, yeni oluşmuş ağrılardır. Kronik ağrılar ise üç, altı aydır süregelen ağrılardır. Birçok uzman kronik ağrıları başlı başına bir hastalık olarak kabul eder (Kanbir, O., 2004). Tıpta ağrı, her zaman temel bir sorun olarak öne çıkmıştır.

Ağrı vücut için bir savunma mekanizmasıdır. Ağrı bölgeyi hareketsiz kılarak daha fazla hasar oluşmasını engeller. Hastalıkların habercisi olarak uyarı görevi yapar. Hastanın bir sağlık kuruluşuna başvurmasında çoğu zaman ağrıların rolü büyüktür.

2.10.3. Ağrı Oluşumu

Vücudun ağrı algılayıcıları (nosiseptörler) diğer duyu cisimleri gibi deride ve deri altında bulunurlar. Ağrılı uyaranlar fiziksel ya da kimyasal karekterli olabilirler.

Melzack ve Wall'in **kapı kontrol teorisine** göre omurilik, ağrı mekanizmasında bir kapı gibidir. Buraya ulaşan ağrılı uyaranlar, bir grup sinir hücresi tarafından beyine iletilmeye çalışılırken diğer bir grup ağrıyı omurilik seviyesinde kontrol etmeye çalışır. Ağrı "kapıyı" geçecek kadar güçlü ise yani omurilik seviyesinde kontrol edilemez ise, ağrı uyarıları beyne ulaşabilir. Ancak uyarıların beyin tarafından algılanmış olması, ağrının mutlaka başlayacağı anlamına gelmez. Bu aşamada bile organizma ağrıyı kontrol altında tutmaya çalışır. Beyinden omuriliğe doğru inen kontrol sistemi harekete geçerek çeşitli kimyasal maddelerin salgılanmasını sağlar.

Endomorfin denilen morfine benzer hormonal salgılar ile ağrı kontrol edilemye çalışılır. Ağrının sübjektif bir duygu olması nedeniyle kesin bir ölçme ve değerlendirme yapmak her zaman mümkün olmamaktadır (Kanbir, O., 2004).

2.10.4. Sırt Ağrısında Tanı

Sırt ağrılarının kaynağını belirlemek her zaman mümkün değildir. Farklı tıbbi disiplinlerin çabasını gerektirir. Ağrının kendisi apaçık ortada iken nedenini aynı açıklıkta ortaya koyabilmek o kadar kolay değildir (Kanbir, O., 2004).

Sırt-bel ağrılarının %85'inden diskler sorumlu tutulmaktadır. Fıtıklaşma, baskının yönüne ve şiddetine bağlı olarak sırta (bele) veya her iki bacağa vuran ağrılar hissettirir (Özer, A F., Sasani, M., 2001). Disk fıtığı hiçbir ağrı üretmeden gelişebileceği gibi, yakındaki başka dokulara, bağlara, sinirlere baskı yaparak şiddetli ağrıların kaynağı olabilir (Zileli, M., 2001).

Sırt ağrıları çoğunlukla konservatif yöntemlerle tedavi edilir. Ancak ağır nörolojik kayıpları olanlar ve çok sık olarak bel ağrısı atağı geçirenlerin ameliyatı düşünülmektedir (Kanbir, O., 2004).

Cerrahi tedavi ile konservatif tedavinin etkilerini karşılaştıran en kapsamlı çalışmayı 1983'te Weber yapmıştır. 10 yıl süren bu çalışmanın sonuçlarına göre birinci yılın sonunda cerrahi tedavi grubundan %65 olumlu sonuç alınırken, cerrahi tedavi görmeyen diğer gruptan %24 olumlu sonuç alınmıştır. Dört yıl sonra ise her iki grup arasında bir fark kalmadığı gözlenmiştir.

Çeşitli araştırmaların sonuçlarına bakıldığında, sırt ağrılarında cerrahi tedavinin kısa dönemde belli bir rahatlama sağlasa bile; aradan geçen 4-7 yıl sonunda sırt ağrısı şikayetleri tekrarlayabilmektedir (Kanbir, O., 2004).

2.11. Girdi Verileri

Cerrahi tedaviye başvurmak için öncelikle acil durumların mevcut olması gerekir. İlerleyen güç kaybı varsa, yani bacaklarda kas kuvvetinde azalma ve/veya hareket kaybı varsa, cerrahi tedavi yapılabilir (Yıldızhan A., 2001).

2.11.1. Girdi Dosyasındaki Girdi Değişkenleri (Bağımsız Değişkenler)

cinsiyet: 2 kadın, 1 erkek
meslek durumu: 1 çalışmayan, 2 çalışan
medeni durumu: 1 evli, 2 bekar
sigara öyküsü: 1 var, 0 yok
bacak ağrısı öyküsü: 1 var, 0 yok
kuvvet kaybı öyküsü: 1 var, 0 yok
refleks kaybı öyküsü: 1 var, 0 yok
duyu kaybı öyküsü: 1 var, 0 yok

2.11.2. Bağımlı Değişken

Sırt ağrısı şikayeti ile başvuran ve fıtık teşhislerinin ardından cerrahi yöntemle tedavi uygulanan hastalarda aynı şikayetin tekrar görülüp görülmemesi bağımlı değişken olarak kullanılmıştır.
nüks: aynı seviden fıtık öyküsü 1 var 0 yok

2.11.3. Uygulama Veritabanı

Ankara Numune Araştırma ve Eğitim Hastanesi Fizik Tedavi ve Rehabilitasyon Kliniği, Uzm. Dr Dilek Keskin Tarafından sağlanan uygulama veritabanı Çizelge 2.1.'de verilmiştir.

Çizelge 2.1. Uygulama Veritabanı*

Değişken	**N**	**%**
Cinsiyet	26	38,8
	41	61,2
Meslek Durumu	51	76,1
	16	23,9
Medeni Durumu	57	85,1
	10	14,9
Sigara Öyküsü	51	76,1
	16	23,9
Bacak Ağrısı Öyküsü	8	11,9
	59	88,1
Kuvvet Kaybı Öyküsü	27	40,3
	40	59,7
Refleks Kaybı Öyküsü	40	59,7
	27	40,3
Duyu Kaybı Öyküsü	21	31,3
	46	68,7
Nüks	25	37,3
	42	62,7
Toplam	67	100

***Ankara Numune Eğitim ve Tedavi Hastanesi, FTR Kliniği, Dr Dilek KESKİN tarafından sağlanmıştır.**

3. BULGULAR

Gereç ve Yöntem'de açık olarak verilen uygulama veritabanı, Joone ile geliştirilen yapay sinir ağına tanıtıldığında, yapay sinir ağının ürettiği bağımlı değişken değerleri ve gerçekteki nüks değerleri elde edilmiştir.

Joone ile oluşturulan ağ yapısında momentum, öğrenme katsayısı, ara katman sayısı gibi YSA parametreleri değiştirilerek, YSA'nın daha iyi öğrenmesi sağlanmıştır. Arayışta momentum, öğrenme katsayısı, ara katman sayısı, aktivasyon fonksiyonu parametrelerinin çeşitli değerleri için denemeler yapılmıştır ancak mümkün tüm kombinasyonların denenmesi matematiksel ve teknolojik olarak mümkün olmadığından sezgisel bir yakaşımla arayışa son verilmiştir.

3.1. YSA Parametrelerindeki Değişimin RMSE'ye Etkisi

YSA parametrelerinin çeşitli değerlerine karşılık geliştirilen uygulama sonucunda Çizelge 3.1. elde edilmiştir.

Uygulamada hangi parametre değerlerinin daha iyi sonuç vereceği bilinmediğinden rasgele başlangıç değerleri seçilmiştir. Tüm kombinasyonları denemek mümkün değildir ancak YSA'nın sezgisel yaklaşımıyla parametre değerleri elenmiştir.

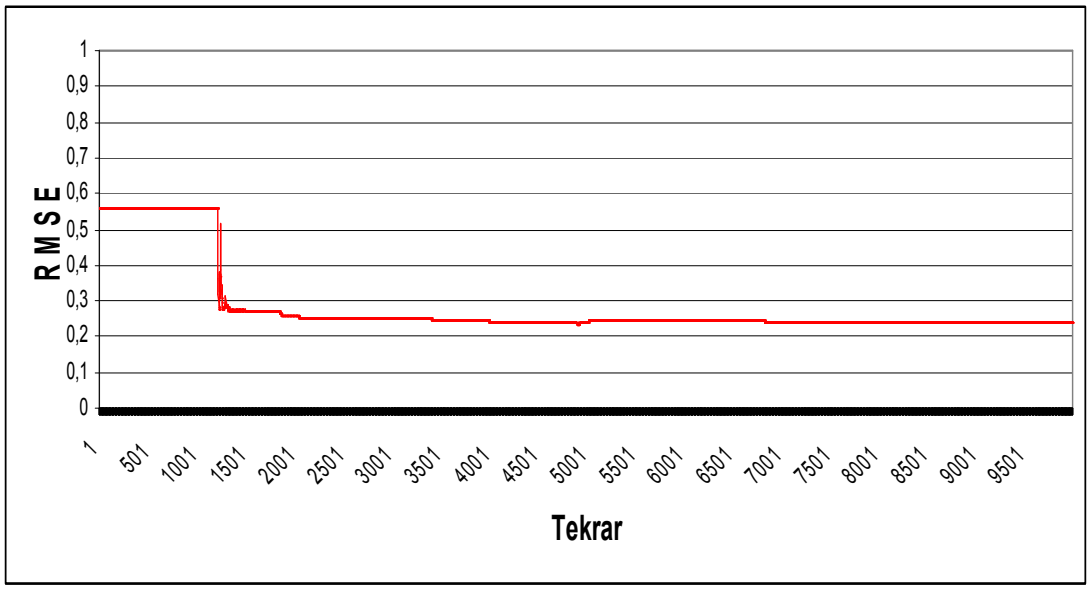
Çizelge 3.1'den görülebileceği gibi öğrenme katsayısının 0,9 değerine karşılık RMSE'de düşüş gözlemlenmiştir. Momentumun 0,3 değeri 0,7 değerinden daha iyi sonuç vermektedir. Benzer bir yaklaşımla, ara katman sayısının 80 olduğu durumda RMSE en küçük değerine ulaşmış ayrıca aktivasyon fonksiyonun lineer kullanılması sigmoid aktivasyon fonksiyonundan daha iyi sonuç elde etmemizi sağlayamamıştır. Bu sebeple arayışın devam eden adımlarında bu tespitler üzerinden gidilerek, RMSE'yi en küçük yapan YSA parametreleri;

öğrenme katsayısı: 0,9
momentum: 0,3
ara katman sayısı:80
aktivasyon fonksiyonu: sigmoid
olmak üzere 0,237805247173697000 elde edilmiştir.

Bir başka deyişle, öğrenme katsayısındaki büyüme RMSE'yi hızla küçültürken, momentum değerindeki artışın RMSE üzerinde arttırıcı etkisi gözlemlenmiş, ara katman sayısındaki artış RMSE'yi etkili olarak küçültürken bilgisayar hafıza birimlerine aşırı yük bindirmiş bu sebeple daha büyük ara katman sayılarıyla çalışılamamıştır. Ayrıca denemeler içerisinde, sigmoid aktivasyon fonksiyonu kullanmanın RMSE değerini küçültmek bakımından lineer aktivasyon fonksiyonuna göre daha etkili olduğu gözlemlenmiştir. İlgili parametre değerlerinin kullanılması sonucunda 10000 deneme sonunda elde edilen RMSE değerleri Çizelge 3.1.'de verilmiştir.

Çizelge 3.1. YSA Parametrelerindeki Değişimin RMSE'ye Etkisi

Öğrenme Katsayısı	Momentum	Ara Katman Sayısı	Aktivasyon Fonksiyonu	RMSE
0,3	0,3	20	Sigmoid	0,559850665274135000
0,7	0,3	20	Sigmoid	0,559850655240586000
0,9	0,3	20	Sigmoid	0,559850548505746000
0,3	0,7	20	Sigmoid	0,559850658471899000
0,7	0,7	20	Sigmoid	0,559850677490775000
0,9	0,7	20	Sigmoid	0,559850573709727000
0,7	0,3	40	Sigmoid	0,559850611162318000
0,9	0,3	40	Sigmoid	0,245403899072439000
0,9	**0,3**	**80**	**Sigmoid**	**0,237805247173697000**
0,9	0,3	40	Linear	0,311476706961812000
0,9	0,3	80	Linear	**0,311472676322960000**



Şekil 3.1. YSA için RMSE Değerlerinin İzlenmesi

1400 tekrarın ardından RMSE değerleri anlamlı bir düşüş göstermiş ve 2000. tekrardan sonra hemen hemen sabit değere ulaşmış olduğu Çizelge 3.1.'den görülmektedir. Elde edilecek Joone_Nüks değişkenine ait RMSE değeri 0,237805247173697000'dir.

3.2. En Küçük RMSE ve En Uygun YSA Parametrelerine Karşılık Gelen Bağımlı Değişken Tahmin Değerleri

Çizelge 3.2. YSA Nüks ve Gerçek Nüks

		YSA		Toplam
		Nüks yok	Nüks var	
GERÇEK	Nüks Yok	19	6	25
	Nüks Var	3	39	42
Toplam		22	45	67

Kesim noktası 0,5 alındığında YSA Nüks ve Gerçek Nüks değerleri Çizelge 3.2.'de gösterilmektedir. Gerçekte nüks olmayan 25 hastanın 6 sı (%24) YSA tarafından 'nüks var' olarak sınıflandırılırken; gerçeke nüks olan 42 hastanın 3'ü (%0,71) YSA tarafından 'nüks yok' olarak sınıflandırılmıştır.

Çizelge 3.3. YSA'nın Yanlış Sınıfladığı Hastalar

No	Joone_Nüks	Gerçek Nüks
1	0,94967376	0
2	0,57492547	0
3	0,21184167	1
4	0,58390649	0
5	0,21184167	1
6	0,21184167	1
7	0,95397035	0
8	0,94228636	0
9	0,57492547	0

Çizelge 3.3'te, YSA, 67 hasta içerisinden 9 hastayı (%13,43) yanlış sınıfladığı görülmektedir.

Çizelge 3.4 ile ilgili olarak FTR uzmanlarıyla görüşüldüğünde 1 ve 9 numaralı hastalarda farklı bir lokasyonda fıtık gözlemlendiği belirtilmiştir. Ancak diğer yanlış sınıflamalar için aynı durum söz konusu değildir.

Sonuç olarak YSA'nın ameliyat sonrası nüks şikayetinin oluşmasını tahmin etmedeki başarısı %86,56 olarak elde edilmiştir.

4. TARTIŞMA

Eldeki mevcut veriler kullanılarak matemetiksel olarak modellenmesi güç olan ve ilerleyen zaman içerisinde toplanan ve toplanmaya devam edilecek olan veriler sözkonusu olduğunda, bir örnekten diğer örneklerin nasıl olacağı tahmin edilemediğinde, YSA uygulamaları fonksiyonel ve zamana ayak uyduran çözümler sunmuştur.

67 kişiden oluşan uygulama veritabanına YSA teknikleri Joone aracılığı ile uygulanmış ve %86,56 oranında doğru sonuç üretebilen bir YSA tasarımına denemeler sonunda ulaşılmıştır. Sezgisel yöntem YSA ile problem çözmede uzman için vazgeçilmez bir araçtır. Bu çalışmada da sürekli parametre değerlerinin tüm kombinasyonları değil özellikle RMSE'yi düşüreceği düşünülen parametreler sezgisel olarak elde edilmiştir.

Çalışmaya devam edilmesi halinde ise, YSA yapısal özelliklerinin ve önemli parametre değerlerinin daha iyi tahminler elde edebilmek için farklılaşma olasılığı oldukça yüksek olacaktır. Bu durum YSA'nın öğrenme kabiliyetinin öğretici veritabanına karşı hassasiyetinin göstergesidir.

Elde edilen uygulama veri tabanı için hangi YSA mimarisi ile çözüm geliştirileceği bilinmediğinen temel YSA mimarisi üzerinde çeşitli temel YSA parametre değerleri denenerek sezgisel yaklaşımla çözüm geliştirilmiştir. Sezgisel yaklaşım YSA çözüm algoritma geliştirmede en güçlü araçlardan birisidir (Beşdok, E., Erler, M., Sağıroğlu, Ş. 2003). YSA temel parametrelerinden gizli katman sayısı , gizli katmanlardaki nöron sayısı, öğrenme katsayısı ve momentum parametrelerinin ne olacağı ile ilgili başlangıç değerlerinin rasgele atanmasına karşın bu değerlerin değişimine yönelik bir kuralın literatürlerde bulunmaması da sezgisel yaklaşımın önemini vurgulamaktadır.
Gizli katmandaki nöron miktarı ağın hatırlatma ve performansı üzerinde çok etkili olmaktadır ancak gizli katmandaki nöron sayısı fazlalığı eğitim işleminin fazla

zaman almasına neden olurken ağın hatırlama kabiliyetine de aynı oranda arttırmaktadır.

Yanlış sınıflanan veriler bakımından FTR uzmanları, farklı bölgelerinde fıtık nüks değerlerinin pozitif olma ihtimallerinin çok yüksek olduğu ve bu bağlamda %86,56 değerinin başarı oranının minimum değeri olarak nitelendirilebileceği belirtilmiştir. Bu bağlamda YSA'nın bilinmeyen bazı kabiliyetleri olduğunu gözlemek için daha fazla veri ile test edilmesi gerekmektedir.

Bel fıtığı teşhisinde ameliyat kararı sonunda başarı olasılığı %95'dir (Yıldızhan, A., 2001). YSA ile sırt ağrısı şikayetlerinin sonunda ameliyat teşhisi konan hastaların şikayetlerinin fiziki anlamda tekrar edip etmeme durumları üzerine geliştirilen uygulama 67 birimlik bir veri tabanında denenmiş ve literatürde %95 nüks etmeme koşuluna uymayan bir veritabanı üzerinde çalışılmıştır. Ancak tez için verilen sürenin tamamı kullanılmış ve araştırma sonlandırılmıştır.

1983'te Weber'in çalışmasında cerrahi tedavi grubunun %65'inden olumlu sonuç alındığı, bu çalışmada doğrulanmaya çalışılan hipotezlerden biri değildi ancak kullanılan uygulama veri tabanında da bu uygulamadakine yakın, %63'lük olumlu sonuç alma oranı sağlanmaktadır. Dolayısıyla Yıldızhan ve Siple'nin bel fıtığı teşhisinde ameliyat kararı sonunda başarı olasılığı %95 denmekteyse de, uygulama veri tabanı ve Weber'in çalışma sonuçlarına göre bu oran %65 civarındadır. Uygulama ve literatür olumlu sonuç oranlarının çakışmaması, uzman hekimlerin ameliyat kararı alırken çok çeşitli hasta öykülerine ihtiyaç duyduklarına işaret etmektedir.

YSA'nın diğer istatistiksel tekniklerle karşılaştırıldığı bir çalışmada, YSA, %36 oranında regresyon yönteminden daha iyi sonuç, %14 oranında regresyon yöntemi YSA'dan daha iyi sonuç ve %50 oranında iki yöntem de benzer sonuçlar elde etmeyi sağlamıştır (Sargent, J. D., 2001).

Uygulama veritabanına, YSA yöntemi yerine kullanılabilecek lojistik regresyon yöntemi ile çözümleme yapıldığında, tüm bağımsız değişkenlerin modelde olduğu durum için Çizelge 4.1. elde edilmiştir.

Çizelge 4.1. Uygulama Veritabanına Lojistik Regresyon Çözümlemesi

		Lojistik Regresyon		**Toplam**
		Nüks yok	**Nüks var**	
GERÇEK	**Nüks Yok**	**14**	**7**	**21**
	Nüks Var	**11**	**35**	**46**
Toplam		**25**	**42**	**67**

Lojistik Regresyon kullanıldığında başarı oranı %73 olmaktadır. Çizelge 4.1.'den kullanılan uygulama veritabanı için YSA yöntemi Lojistik Regresyon yöndeminden daha başarılı sonuç elde edildiği görülmektedir ancak bu durumun uygulama veritabanından kaynaklanabileceği unutulmamalıdır. Bu bakımdan bu iki alanın karşılıklı incelenmesi ileri araştırmalar gerektirir.

5. SONUÇ ve ÖNERİLER

Kullanılan doğru araç, yöntem ve tasarım ile sırt ağrıları şikayetleri ve konulan tanılar arasındaki ilişki doğrulanmış ve YSA'nın bu ilişkiyi %86,56 oranında öğrenebilmesi sağlanmıştır.

En uygun YSA modelini bulurken, Joone editörün kullanım kolaylıklarından fazlasıyla yararlanılmıştır. Joone yordam kütüphanesi YSA tasarımı yapabilen diğer öğrenen bir sistem haline getirebilme olanağı vermektedir. Yazılım araçlarından farklı olarak, tasarlanan YSA ya dışarıdan bir Java programı yardımıyla erişebilmek, ve hatta onu var olan hazır veritabanlarıyla sürekli güncel tutmak mümkündür.

YSA standart istatistiksel tekniklere kıyasla daha karmaşık problemlerin modellenmesi ve çözülmesinde kullanılabilecek matematiksel bir modeldir. Standart istatistiksel yöntemler YSA'nın yeteneklerine erişemeyebilmektedir.

Bu çalışma göstermiştir ki, yapay sinir ağları diğer alanlarda olduğu tıp alanında da rahatlıkla kullanılabilecek, sezgisel yaklaşımı güçlü olan bir öğrenme algoritmasıdır. Tıpta daha çok örüntü tanıma ve değerlendirmede kullanılan ve çok doğru sonuçlar üretebilen bir yöntemdir. Aynı şekilde herhangi bir hastalığın teşhisinde ve tedavisinde en uygun parametreleri bulma ve sonuca giden yolda iyi bir yönlendirme aracı olabilir. Kullanılan bağımsız değişkenleri doğru seçerek, kullanılacak YSA modelini en doğru şekilde tasarlayarak, var olan veritabanları üzerinde araştırmalar yapmak son derece kolay hale gelmektedir.

İlerleyen zaman içerisinde bilgisayar teknolojilerininde gelişmesiyle bu teknolojilere paralel gelişen YSA tekniklerinin de ilerlemesi ve özellikle tıp alanında yaygın olarak kullanılması beklenmektedir.

ÖZET

Yapay Sinir Ağlarının İncelenmesi ve Sırt Ağrısı Olan Bireyler Üzerinde Bir Uygulaması

Yapay sinir ağları insan beynini taklit etme yöntemlerini, günümüz problemlerini modellemek ve çözmek için kullanılan bir öğrenme algoritmasıdır. Uygulama için, Java teknolojisi kullanılarak geliştirilmiş bir bilgisayar yazılımı olan Joone yordam kütüphanesi ve bu kütüphane ile geliştirilmiş bir kullanıcı arayüzü kullanıldı. Bu kullanıcı arayüzü kullanılarak hazırlanan YSA modeli ilişkiyi öğrenme aşamasından geçirilmiş ve ardından aynı arayüz ve aynı veriler ile test edilmiştir. Çalışma sonunda, Joone editörü kullanılarak yapay sinir ağlarının modellenmesi ve uygulaması tıp alanında da diğer alanlarda olduğu gibi uygulanabilir olduğu gösterilmiştir.

Anahtar Kelimeler: Aktivasyon Fonksiyonu, Joone, Matematiksel Nöron Modeli, Momentum, Yapay Sinir Ağları

SUMMARY

Study of Artifcial Neural Networks and an Aplication on Patients with Back Pain.

Artificial neural networks use human brain learning methods for modeling and solving daylife problems. In the application, Joone software, the user interface and its libraries are used. The artificial neural network is teached about the relations between the independent and dependent variables. These results show that working with joone made it very easy to model and apply artificial neural networks in medicine as in other fields.

Anahtar Kelimeler: Activation Function, Artificial Neural Networks, Joone, Mathmatical Neuron Model, Momentum.

KAYNAKLAR

ANDERSON, D., MC NEILL, G., (1992). Artificial Neural Networks. Kaman Sciences Corp., New York, s.:1-40.

BEŞDOK, E., ERLER, M., SAĞIROĞLU, Ş. (2003). Mühendislikte Yapay Zeka Uygulamaları1: Yapay Sinir Ağları. Kayseri: Ufuk Kitap Kırtasiye-Yayıncılık Tic. Ltd. Şti. s.:1-165

BORMAN, P., KESKİN, D., BODUR, H., (2003). The Efficacy of Lumbar Traction in the Management of Patients with Low Back Pain. Rheumatol Int. 23 s.:82-86

CUMMİNS, F., SCHRAUDOLPH, N., (1999). Lecture Notes prepared by Genevieve Orr, Willamette University.
Erişim: [http://www.willamette.edu/~gorr/classes/cs449/intro.html]. Erişim Tarihi: 20.12.2005

EFE, M.Ö., KAYNAK, O., (2000). Yapay Sinir Ağları ve Uygulamaları. İstanbul: Boğaziçi Üniversitesi Yayınevi, s.:1-25

FINLAND, Åbo Akademi University. Erişim: [http://www.abo.fi/~rfuller/nfs8.pdf]. Erişim Tarihi: 24.12.2005

KANBİR, O., (2004) Bel Ağrısı. Bursa: Ekin Kitapevi, s.:141-180

KRÖSE, B. VAN DER SMAGT, P. (1996). An Introduction to Neural Networks. 8th. Edition, The Netherlands: University of Amsterdam, s.:1-64

MINSKY, M., PAPERT, S., (1969). Perceptron, MIT Press ,Cambridge.

MORSE, P. M., FESHBACH, H. (1953). Methods of Theoretical Physics. Part I. New York: McGraw-Hill, s.:434-443

ÖZER, A. F., SASANİ, M., (2001). Basit Lomber Diskektomi Cerrahi Endikasyonları ve Teknik, Lomber Disk Hastalığı. Logos Yayıncılık, s.. 89-115

SARGENT, J. D., (2001). Comparision of Artificial Neural Networks with Other Statistical Approaches. Cancer 91 s.:1636-1642

SARLE, W., (2002). Erişim. [ftp://ftp.sas.com/pub/neural/FAQ2.html]. Erişim Tarihi: 02.01.2006

SİPLE, B. J., (2000). Bel Ağrısının Tedavisi. Logos Yayıncılık, s.:24-29

WEISSTEIN, E.W. "Global Maximum." From MathWorld--A Wolfram Web Resource. Erişim: [http://mathworld.wolfram.com]. Erişim Tarihi: 18.09.2005

WOLFRAM, S., (2002). A New Kind of Science. Champaign, IL: Wolfram Media, s.:985. Erişim: [www.jooneworld.com]. Erişim Tarihi: 01.04.2006

YILDIZHAN, A., (2001). Bel fıtığı ve Korunma Yolları. Fatih Yayıncılık, İstanbul, 7. Baskı

ZILELLI, M., (2001). Lomber Disk Hastalığında Klinik Tanıya Yaklaşım ve Cerrahi Endikasyonları, Lomber Disk Hastalığı. Logos Yayıncılık, s.:116-141

EKLER

Ek – 1

```
import org.joone.engine.DirectSynapse;
import org.joone.engine.Layer;
import org.joone.engine.Pattern;
import org.joone.net.NeuralNet;
import org.joone.net.NeuralNetLoader;


public class tez
{
        private double[][] ornekVeri = new double[][] {
                        {0.0, 0.0, 0.0, 0.0, 0.0, 0.0}};
        /**
         * @param args
         */
        public static void main(String[] args) {
                // tez nesnesinin yeni bir örneğini oluştur
                tez tez1 = new tez();
                // yapay sinir ağını test et
                tez1.test("D:\\arsiv\\joone\\burcu_tez\\tez.snet");
                //System.out.println("Output2#");
        }

        /**
         * @param fileName
         * Bu işlev fileName parametresi ile belirtilen dosyadan, joone editör ile
         * hazırlanan ayrıştırılmış yapay sinir ağını alarak, bu ağa
         * “veriler” dizi nesnesi yardımıyla veri aktarır.
```

```
        * Ağı çalıştırır ve çalışma sonucunu alıp yazdırır.
        */

private void test(String fileName) {
                // ayrıştırılmış yapay sinir dosyasını yükle
                NeuralNetLoader loader = new NeuralNetLoader(fileName);

                // dosya içindeki ağ nesnesine eriş ve kopyala
                NeuralNet myNet = loader.getNeuralNet();

                if (myNet != null) { // ağ nesnesine erişim başarılı ise
                        // ağdaki giriş katmanını al
                        Layer input = myNet.getInputLayer();

// giriş katmanının girdilerini temizle
                        input.removeAllInputs();

// yeni bir DirectSynapse nesne örneği al
                        DirectSynapse memInp = new DirectSynapse();

// girdi katmanı, FullSynapse yerine
// DirectSynapse ile besleniyor
                        input.addInputSynapse(memInp);

                        // ağdaki çıkış katmanını al
                        Layer output = myNet.getOutputLayer();

                        // çıkış katmanının çıktı yolunu kaldır
                        output.removeAllOutputs();

                        // yeni bir DirectSynapse nesne örneği al
                        DirectSynapse memOut = new DirectSynapse();
```

```
                    // çıktı katmanı, FullSynapse yerine
// DirectSynapse'e çıktı veriyor
                    output.addOutputSynapse(memOut);

                    // ağı işler hale getir
                    myNet.start();

                    // yeni girdi örüntülerini test et
                    // ornekVeri içerisindeki tüm örüntüler test ediliyor
                    for (int i=0; i < 1; ++i) {
                        // örüntü katarından bir örüntü al
                        Pattern iPattern = new Pattern(ornekVeri[i]);
                        iPattern.setCount(1);

// alınan örüntüyü girdi akımına gönder
// bu sayede, giriş katmanı
// yeni bir örüntü ile beslenir
                        memInp.fwdPut(iPattern);

// giriş katmanı yeni bir örüntü
// ile beslendikten sonra
// hali hazırda çalışan ağda, gizli katman
// ve çıkış katmanında da
// işletime alınır ve çıkış katmanı
// bir sonuç üretir.
                        // çıkış katmanının ürettiği bu sonucu al
                        Pattern pattern = memOut.fwdGet();

// sonucu görüntüle
System.out.println("Output#"+(i+1)+ " = "+pattern.getArray()[0] + " " +
pattern.getArray()[1] + " " + pattern.getArray()[2]);
```

```
                    }
                    // ağı durdur
                    myNet.stop();
                }
            }
        }
```

ÖZGEÇMİŞ

1. BİREYSEL BİLGİLER

Burcu KARAKAYA (1978 - ANKARA)
Uyruğu : T.C.
Medeni Durumu : Evli
Ev : 312 213 6241
GSM : 533 436 8071
E-posta : burcubr@gmail.com
Adres : Ergin Sokak. 33/7 06580 Mebusevleri / ANKARA

2. EĞİTİM

- Gazi Üniversitesi / İstatistik, Ankara
- Ankara Üniversitesi / Biyoistatistik Ana Bilim Dalı
- TSE / İç Kalite Tetkiki Eğitimi (ISO 9001:2000)
- TSE / Tetkik/Baştetkik Görevlisi

3. ÜNVANLAR

- İSTATİSTİKÇİ
- EOQ QUALITY AUDITOR

4. MESLEKİ DENEYİMİ

- Lisans stajı: Halk Bankası Genel Müdürlüğü-Ankara, AR-GE
- Doktaş Doğa Koruma Teknikleri Sanayi ve Ticaret A.Ş., ISO 1994 Kalite Güvence Sistemi
- Osiyek Enstitüsü-Hırvatistan, Dönemsel Danışman
- Motali Teknolojik Sistemler A.Ş. (Kalite Yönetim Temsilcisi) - ANKARA

5. ÜYE OLDUĞU BİLİMSEL KURULUŞLAR

- European Organisation for Quality

6. BİLİMSEL İLGİ ALANLARI

İstatistik, Biyoistatistik, İstatistiksel Süreç Yönetimi, Yapay Sinir Ağları

7. BİLİMSEL ÇALIŞMALAR

1- JOSİPOVIC,Makro., .., Burcu KUŞMAN ve Arkadaşları (uluslararası)
Adaptation Studies on Some Hungarian and German Potato Cultivars,
Agronomi Semposium, Opatija - Croatia 2004
2- JOSİPOVIC,Makro., ..,Burcu KARAKAYA ve Arkadaşları (uluslararası)
Correlations Among Tuber Morphological Characteristics and Some Yield Components with General Preference of the Cultivar, Agronomi Semposium, Opatija - Croatia 2005