İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE SİSTEM TANIMA

YÜKSEK LİSANS TEZİ
Müh. M. Orkun ÖĞÜCÜ
(504041113)

Tezin Enstitüye Verildiği Tarih : 8 Mayıs 2006
Tezin Savunulduğu Tarih : 13 Haziran 2006

Tez Danışmanı : Prof.Dr. Müjde GÜZELKAYA
Diğer Jüri Üyeleri Prof.Dr. Serhat ŞEKER
Yrd.Doç.Dr. A. Şima ETANER-UYAR

HAZİRAN 2006

ÖNSÖZ

Bu çalışmanın hazırlanmasında yardım ve desteklerini esirgemeyen Sayın Prof. Dr. Müjde GÜZELKAYA'ya, Sayın Prof. Dr. Serhat ŞEKER'e, Sayın Prof. Dr. İbrahim EKSİN'e ve Sayın Engin YEŞİL'e teşekkürlerimi sunarım.

Haziran, 2006 **M.Orkun ÖĞÜCÜ**

İÇİNDEKİLER

	Sayfa No
KISALTMALAR	v
TABLO LİSTESİ	vi
ŞEKİL LİSTESİ	vii
SEMBOL LİSTESİ	ix
ÖZET	x
SUMMARY	xii
1. GENEL KAVRAMLAR	**1**
1.1 Yapay Zekanın Tanımı	1
1.2 Yapay Zeka Teknolojileri	1
1.3 Makinaların Öğrenme Süreci	3
1.4 Yapay Sinir Ağlarında Öğrenme Biçimleri	4
1.5 Yapay Sinir Ağlarında Öğrenme Kuralları	5
2. YAPAY SİNİR AĞLARINA GİRİŞ	**7**
2.1 Yapay Sinir Ağlarının Genel Özellikleri	7
2.2 Yapay Sinir Ağlarının Yetenekleri	8
2.3 Yapay Sinir Ağlarının Kullanıldığı Alanlar	9
2.4 Yapay Sinir Ağlarının Avantajları	10
2.5 Yapay Sinir Ağlarının Dezavantajları	11
2.6 Yapay Sinir Ağlarının Tarihi	12
2.6.1 Yapay Sinir Ağlarının Tarihi – 1970 Öncesi Çalışmalar	12
2.6.2 Yapay Sinir Ağlarının Tarihi – 1970 Sonrası Çalışmalar	15
2.7 Yapay Sinir Ağlarının Sınıflandırılması	16
3. YAPAY SİNİR AĞLARININ YAPISI	**21**
3.1 Teknoloji ve Biyolojik Yapılar	21
3.2 Biyolojik Sinir Ağlarının Yapısı	22
3.3 Yapay Sinir Hücresinin Yapısı	24
3.4 Yapay Sinir Ağının Yapısı	27
3.5 Farklı Modellerin Ortaya Çıkmasını Sağlayan Unsurlar	29
4. TEK KATMANLI ALGILAYICILAR	**30**
4.1 Tek Katmanlı Algılayıcıların Ortak Özellikleri	30
4.2 En Basit Tek Katmanlı Algılayıcı	32
4.3 ADALINE Modeli	33
4.4 Perceptron ve ADALINE Modellerinin Karşılaştırılması	35
5. ÇOK KATMANLI YAPAY SİNİR AĞLARI	**41**
5.1 Çok Katmanlı Yapay Sinir Ağlarına Olan Gereksinim	41
5.2 Çok Katmanlı Yapay Sinir Ağlarının Yapısı	42

5.3 Çok Katmanlı Yapay Sinir Ağlarının Öğrenme Kuralı 43
5.4 Tasarımda Karşılaşılabilecek Problemler 49
6. SİSTEM TANIMA : YAPAY SİNİR AĞLARI İLE DİĞER METODLARIN KARŞILAŞTIRILMASI 51
6.1 Sistem Bilgileri 51
6.2 Lineer Modelin Eldesi ve Klasik PID İle Kontrolü 52
6.3 Bulanık Modelin Eldesi ve Bulanık-PID İle Kontrolü 58
6.4 Yapay Sinir Ağı Modelinin Eldesi ve Öz-Uyarlamalı Nöro-PID İle Kontrolü 63

7. SONUÇLAR 77

KAYNAKLAR 79

EK.A Perceptron ve ADALINE Modellerine İlişkin Algoritma **80**
EK.B Çok Katmanlı Yapay Sinir Ağına İlişkin Algoritma **82**

ÖZGEÇMİŞ 85

KISALTMALAR

ADALINE : Adaptive Linear Neuron
AIC : Akaike's Information Criterion
ANFIS : Adaptive Neuro-Fuzzy Inference System
ART : Adaptive Resonance Theory
CV : Cross Validation
FPE : Final Prediction Error
MADALINE : Multi-Adaptive Linear Neuron
MDL : Minimum Description Length
PID : Proportional-Integral-Derivative
YSA : Yapay Sinir Ağları

TABLO LİSTESİ

		Sayfa No
Tablo 3.3.1 :	Yapay sinir hücrelerinde kullanılan toplama fonksyonları	25
Tablo 3.3.2 :	Yapay sinir hücrelerinde kullanılan aktivasyon fonksyonları	26
Tablo 4.4.1 :	Giriş matrisi ve hedef vektörü	38
Tablo 5.3.1 :	Yapay sinir ağlarında kullanılan toplama fonksyonları	44
Tablo 5.3.2 :	Yapay sinir ağlarında kullanılan aktivasyon fonksyonları	45
Tablo 6.4.1 :	Farklı modelleme yaklaşımları için anlınan sonuçlar ve yorumlar	75
Tablo 6.4.2 :	Farklı kontrolör yapıları için alınan sonuçlar	76

ŞEKİL LİSTESİ

		Sayfa No
Şekil 1.2.1 :	Yapay zeka yaklaşımlarının sınıflandırılmasına ait şema	2
Şekil 1.5.1 :	Makinaların öğrenme süreci	6
Şekil 2.6.1 :	İki boyutlu öklid uzayında (x_1,x_2) ikililerine karşılık gelen semboller	14
Şekil 2.6.2 :	Lojik-VE fonksyonunun geometrik yorumu	14
Şekil 2.6.3 :	Lojik-ÖzelVeya fonksyonunun geometrik yorumu	15
Şekil 2.7.1 :	3-girişli 2-çıkışlı çok katmanlı ileribeslemeli ağ yapısı	17
Şekil 2.7.2 :	1-girişli 1-çıkışlı Elman-tipi geribeslemeli ağ yapısı	18
Şekil 3.2.1 :	Sinir ağlarının temel birimi olan sinir hücresinin şematik yapısı	22
Şekil 3.2.2 :	Vücudumuzdaki sinir ağının küçük bir kısmı	23
Şekil 3.3.1 :	Yapay sinir hücresinin yapısı	24
Şekil 3.4.1 :	Yapay sinir ağının yapısı	27
Şekil 4.1.1 :	Tek katmanlı yapay sinir ağı modeli	30
Şekil 4.1.2 :	İki-girişli bir sistem için sınıf ayracı	31
Şekil 4.2.1 :	En basit tek katmanlı algılayıcı modeli	32
Şekil 4.3.1 :	Aktivasyon fonksyonu sigmoidal olan bir ADALINE modeli	33
Şekil 4.3.2 :	ADALINE modelinin ayrıntılı yapısı	35
Şekil 4.4.1 :	Çıkışların sınıflandırılması için kullanılan yüzey ayracı	36
Şekil 4.4.2 :	İterasyon sayısına göre karesel hatanın değişimi	39
Şekil 5.2.1 :	İleribeslemeli çok katmanlı yapay sinir ağı	42
Şekil 5.3.1 :	İleribeslemeli çok katmanlı yapay sinir ağının ayrıntılı yapısı	45
Şekil 6.1.1 :	Sisteme ait giriş ve çıkış verileri	51
Şekil 6.1.2 :	Sistemin birim basamak cevabı	52
Şekil 6.2.1 :	Farklı yaklaşımlar ile elde edilen modeller için alınan sonuçlar	55
Şekil 6.2.2 :	Lineer ARX modelinden alınan çıkışlar ve ölçülen sistem çıkışları	55
Şekil 6.2.3 :	Lineer modelin PID ile kontrolünde kullanılan blok diagram	56
Şekil 6.2.4 :	Farklı kazanç değerleri için elde edilen birim basamak yanıtları	57
Şekil 6.2.5 :	Farklı kazanç değerleri için elde edilen karesel hata işaretleri	57
Şekil 6.2.6 :	Karesel hatayı en azlayacak şekilde belirlenen kazanç değerleri için sistemin birim basamak yanıtı	58
Şekil 6.3.1 :	Sistemin ölçülen çıkışı ve bulanık modelden alınan çıkış değerleri	60
Şekil 6.3.2 :	Bulanık modelin bulanık-PID ile kontrolünde kullanılan blok diagram	61
Şekil 6.3.3 :	Farklı kazanç değerleri için elde edilen birim basamak yanıtları	62
Şekil 6.3.4 :	Farklı kazanç değerleri için elde edilen karesel hata işaretleri	62
Şekil 6.3.5 :	Karesel hatayı en azlayacak şekilde belirlenen ölçekleme çarpanları için sistemin basamak yanıtı	63
Şekil 6.4.1 :	YSA modelinden alınan değerler ve ölçülen sistem değerleri	65
Şekil 6.4.2 :	Yapay sinir ağı modelinin benzetim sonuçları	66
Şekil 6.4.3 :	Elman-tipi yapay sinir ağı ile elde edilen sistem modeli	68

Şekil 6.4.4 : YSA modelinin öz-uyarlamalı nöro-PID ile kontrolüne ilişkin blok diagram 69
Şekil 6.4.5 : Nöro-PID ile kontrol edilen sistemin birim basamak cevabı ve hatanın değişimi 70
Şekil 6.4.6 : KP, KI ve KD kazançlarının değişimi ve bu işaretlerden elde edilen kontrol işareti 71
Şekil 6.4.7 : Öz-uyarlamalı nöro-PID ile kontrol edilen sistemin kare dalga cevabı 72
Şekil 6.4.8 : Sistemin basamak giriş için geçici hal davranışına ilişkin değerleri ve YSA modeli çıkışı 73
Şekil 6.4.9 : Nöro-PID ile kontrol edilen geçici hal davrnaışı modelinden alınan değerler 74
Şekil 6.4.10 : Klasik PID ile kontrol edilen model ve bulanık-PID ile kontrol edilen model 74

SEMBOL LİSTESİ

$x_1, x_2, ...$	:	Yapay sinir hücresine olan girişler
$w_1, w_2, ...$	:	Yapay sinir hücresinin ağırlık değerleri
$\Delta w_1, \Delta w_2, ...$	:	Ağırlıklardaki değişim
NET	:	Yapay sinir hücresinin ağırlıklandırılmış girişlerinin toplamı
F	:	*NET* 'in, aktivasyon fonksyonundan geçirildikten sonraki değeri
ϕ	:	Yapay sinir hücresinin eşik değeri
η	:	Öğrenme katsayısı
y^*	:	Hedef değer
y	:	Çıkış değeri
ε	:	Hedef değer ile çıkış değeri arasında tanımlanan hata
na, nb	:	ARX modeli polinomlarının dereceleri
nk	:	ARX modeli için sistem gecikmesi
K_P, K_I, K_D	:	PID kontrolör için oransal, integral ve türev kazançları

YAPAY SİNİR AĞLARI İLE SİSTEM TANIMA

ÖZET

Bu çalışmada yapay sinir ağları ile sistem tanıma konusuna değinilmiştir. Çalışma altı bölümden oluşmaktadır. İlk dört bölümde, çok katmanlı yapay sinir ağlarının anlaşılabilmesi için gerekli olan kavramlar tanıtılmıştır. Beşinci bölümde çok katmanlı yapay sinir ağları ile ilgili teorik bilgiler anlatılmıştır. Bu bilgiler ışığı altında, çok katmanlı yapay sinir ağları için bir algoritma yazılmıştır. Altıncı bölümde ise bu algoritma kullanılarak elde edilen yapay sinir ağı modeli, parametrik ve parametrik olmayan sistem tanıma yöntemleri ile karşılaştırılmıştır. Elde edilen modeller, kullanılan modelleme tekniklerine dayanan kontrolör yapıları ile sürülmüş ve bu kontrolörler zaman tanım bölgesi kriterleri dikkate alınarak kıyaslanmıştır.

Yapay sinir ağlarını, yapay zekanın bir alt başlığı olarak değerlendirmek gerekmektedir. Bu sebeple, birinci bölümde yapay sinir ağlarınında olduğu yapay zeka yaklaşımları tanıtılmıştır. Yapay sinir ağlarında kullanılan öğrenme biçimleri ve öğrenme kuralları incelenmiştir.

İkinci bölümde, yapay sinir ağlarının kullanıldığı alanlar, avantajları ve dezavantajlarına değinilmiştir. Kısa bir tarihçe verilmiş ve lojik-ÖzelVeya kapısının, yapay sinir ağlarının gelişim sürecindeki yeri vurgulanmıştır. Ayrıca yapay sinir ağı modellerinin bir sınıflandırması yapılmıştır.

Üçüncü bölümde, yapay sinir ağı çalışmalarına ilham veren, biyolojik sinir ağları incelenmiştir. Yapay sinir ağlarının temel birimi olan sinir hücreleri irdelenmiştir.

Dördüncü bölümde, tek katmanlı yapay sinir ağlarına diğer bir ifade ile algılayıcılara yer verilmiştir. Yüzey ayracının tanımı yapılmış ve özel bir durum için yüzey ayracının geometrik yorumu irdelenmiştir. Perceptron ve ADALINE modelleri incelenmiş, arasındaki farklar ortaya konmuştur. Perceptron modelinde kullanılan Hebb öğrenme kuralı ve ADALINE modelinde kullanılan Delta öğrenme kuralı tanıtılmıştır. Bu iki modelin sınıflandırma ve lineer ilişkileri belirleme performansları karşılaştırılmış ve aktivasyon fonksyonları açısından birbirlerine olan üstünlükleri ortaya konulmuştur.

Beşinci bölümde, hatanın geriye yayılımı yöntemini kullanan çok katmanlı yapay sinir ağları incelenmiştir. Genelleştirilmiş delta kuralı kullanılarak, ağırlıkların güncellenme formülleri türetilmiştir. Ayrıca ağların tasarımında karşılaşılan problemler ve dikkat edilmesi gereken hususlar belirtilmiştir.

Altıncı bölümde, giriş ve çıkış değerleri bilinen ölü zamanlı lineer bir sistem için üç farklı yaklaşım kullanılarak modeller elde edilmiştir. Sistem için parametrik lineer ARX modeli, parametrik olmayan nonlineer bulanık model ve yine parametrik

olmayan yapay sinir ağı modeli elde edilmiştir. Yapılan modelleme hataları ile bu yöntemlerin karşılaştırılması yapılmıştır. Ayrıca bu modeller, kullanılan metodlara dayanan kontrolör yapıları ile kontrol edilmiştir. ARX modeli klasik PID ile, bulanık model bulanık-PID ile, yapay sinir ağı modeli ise öz-uyarlamalı nöro-PID ile sürülmüş ve zaman tanım bölgesi kriterlerine göre karşılaştırılmıştır.

SYSTEM IDENTIFICATION WITH ARTIFICIAL NEURAL NETWORKS

SUMMARY

In this work, system identification with artificial neural networks is examined. From section one to five fundamentals of the neural networks are studied. Multilayer neural network which is used to form the algorithm for system identification is explained at section five. The neural network model which is obtained by this algorithm is compared with parametric and non-parametric identification methods at section six. Also, these models are driven by controllers which are based to these system identification methods and they are compared with time-domain criterions.

The artificial neural networks can be interpreted as a subsection of artificial intelligence, so at section one artificial intelligence approaches are introduced. Learning strategies and learning rules are explained.

At section two the research areas, advantages and disadvantages of neural networks are mentioned. A short history of the neural networks is given and the importance of the logic-XOR function is clarified according to the developing process. Also a classification of the best known neural network type is given.

At section three, biological neural networks is examined which inspires to artificial neural networks concept. Artificial neuron which is the basic unit of the artificial neural networks is explored.

At section four, single-layer neural networks are introduced. Decision surface is defined and geometrical interpretation of the decision surface is examined for a special situation. Perceptron and ADALINE models are studied and differences between these models are revealed. Hebb and Delta rules which are used by perceptron and ADALINE respectively are introduced. Linear learning and classification performances of these two models are compared.

At section five, multilayer neural networks are examined which use backpropogation algorithm. The update formulas are derived by the help of generalized delta rule. Also design problems of the artificial neural networks are explained.

At section six, a linear system which has dead-time is modeled with using the pair of input-output values. Three models are obtained by parametric linear ARX modeling, non-parametric non-linear fuzzy modeling and even so neural networks modeling. The models are compared with the modeling errors. Also, these models are driven by controllers which are based to these modeling approaches. A classical PID is used for ARX model, fuzzy-PID is used for fuzzy model and at last self-tuning neural-PID is used for neural model. The results are compared according to time-domain criterion like maximum overshoot and settling time.

1. GENEL KAVRAMLAR

Bu bölümde, yapay sinir ağlarını incelerken kullanılacak olan genel kavramlara yer verilecektir. Kısım 1.1'de zeki sistemler ve yapay zeka çalışmalarına ilişkin tanımlar yapılacaktır. Kısım 1.2'de yapay zeka sınıflandırılmaya çalışılacak ve yapay sinir ağlarının diğer yapay zeka yaklaşımları ile olan ilişkisine değinilecektir. Kısım 1.3'de makinaların öğrenme sürecine ilişkin bilgiler verilecektir. Kısım 1.4'de yapay sinir ağlarında öğrenme biçimleri tanıtılacaktır. Kısım 1.5'de ise yapay sinir ağlarında yaygın olarak kullanılan öğrenme kuralları genel hatları ile açıklanacaktır.

1.1 Yapay Zekanın Tanımı

Günümüz teknolojisi sayesinde, bilgisayarlar ve bilgisayar sistemleri yaşamın vazgeçilmez bir unsuru haline gelmiştir. Önceleri sadece karmaşık hesaplamaları gerçekleştirmek üzere geliştirilen bilgisayarlar, günümüzde olaylar arasındaki ilişkileri öğrenebilen, mevcut bilgileri kullanarak yorum yapabilen ve karar verebilen akıllı makinalar haline dönüşmüşlerdir. Matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler sezgisel yöntemler yolu ile bu makinalar tarafından çözülebilmektedir. Bu özellikleri taşıyan makinalara zeki sistem, bu özellikleri kazandırmak için yapılan çalışmalara ise yapay zeka çalışmaları denir.

Zeki sistemlerin en önemli özelliği mevcut bilgiler ile olayları öğrenerek, benzer olaylar hakkında karar verebilmeleridir. Bu amaç ile insan beyninin öğrenme ve karar verme mekanizmaları taklit edilmektedir [1].

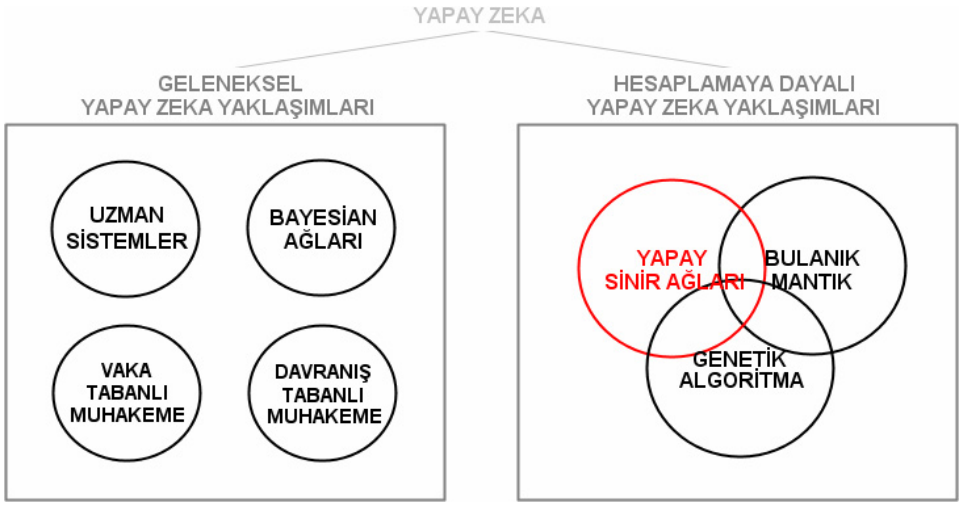
1.2 Yapay Zeka Teknolojileri

İnsan beyninin öğrenme ve karar verme mekanizmalarının taklit edilmesi için yapılan çalışmalar farklı yaklaşımların ve dolayısıyla farklı metodların ortaya çıkmasını sağlamıştır. Yapay zeka teknolojileri iki ana başlık altında incelenebilir.

Bunlar geleneksel yapay zeka yaklaşımları (conventional artificial intelligence) ve hesaplamaya dayalı yapay zeka yaklaşımlarıdır (computational artificial intelligence) [2].

Geleneksel Yapay Zeka Yaklaşımları : Uzman sistemler ve vaka tabanlı muhakeme bu gruba örnek olarak verilebilir. Uzman sistemlerin geniş bir bilgi tabanına ihtiyaçları vardır. Dolayısıyla fazla miktarda bilgi işleyebilmeleri gerekmektedir. Uygulamada elde edilen bilgi, birikim ve gözlem sonuçları ile oluşturulacak bilgi tabanından yararlanarak muhakeme yapabilir ve sonuca ulaşabilirler fakat bilgi tabanında olmayan bir durum için çözüm üretemezler. Vaka tabanlı muhakemede ise daha önce tecrübe edilmiş neden-sonuç ilişkilerinden yararlanılır.

Hesaplamaya Dayalı Yapay Zeka Yaklaşımları : Yapay sinir ağları, bulanık mantık ve evrimsel hesaplama bu gruba dahildir. Yapay sinir ağı yaklaşımında, genellikle örnekler kullanılarak, olayların girdi ve çıktıları arasındaki ilişkiler öğrenilir ve benzer olaylar için yorumlar yapılabilir. Bulanık mantık sistemlerin belirsizlik durumlarında muhakeme yapabilmelerini sağlar. Evrim stratejileri, genetik algoritma ve evrimsel programlama kavramları için genel olarak evrimsel hesaplama terimi kullanılır. Genetik algoritma karmaşık optimizasyon problemlerinin çözülmesinde kullanılan bir yaklaşımdır. Yapılacak ilk işlem rastgele bir çözüm üretmektir. Bu çözümler eşleştirilerek daha iyi çözümler elde edilir. Bu işlem daha iyi bir sonuç bulunamayıncaya kadar devam eder. Yapay zeka yaklaşımlarının sınıflandırılmasına ait şema şekil 1.2.1'de gösterilmektedir.



Şekil 1.2.1 : Yapay zeka yaklaşımlarının sınıflandırılmasına ait şema

Bu farklı yaklaşımlar, uygulamanın niteliğine göre ayrı ayrı yada birlikte kullanılabilirler. Örneğin, yapay sinir ağlarında gizli katmanların ve gizli düğümlerin sayısının belirlenmesi bir optimizasyon problemidir. Bu sebeple yapay sinir ağları ve genetik algoritma birlikte kullanılabilir. Girdi ve çıktılar arasındaki ilişkiler yapay sinir ağları ile öğrenilirken, gizli katman ve düğümlerin sayısı genetik algoritma ile belirlenebilir. Eğer uygulamada kesin olmayan durumlar söz konusu ise bu yaklaşımlara bulanık mantığında eklenmesi mümkündür [2].

1.3 Makinaların Öğrenme Süreci

En genel halde öğrenme, zaman içinde yeni bilgilerin keşfedilmesi yoluyla davranışların iyileştirilmesi süreci olarak tanımlanabilir. Makina öğrenmesi ise, makinaların bir olay ile ilgili bilgileri ve tecrübeleri öğrenerek gelecekte oluşacak benzer olaylar hakkında kararlar verebilmesi ve problemlere çözümler üretebilmesidir.

Bilgilerin makinalara tanıtılması çok değişik biçimlerde olabilir. Uzman sistemlerde bilgiler, makinalara talimatlar yada kurallar ile tanıtılırken, yapay sinir ağları yaklaşımında bilgilerin tanıtılması örnekler yolu ile olur [1].

Bilgilerin tanıtılma şekli hangisi olursa olsun, makinalar sonuca iki farklı öğrenme sürecini kullanarak ulaşırlar. Bunlar kural belirleyerek öğrenme (regularity detection) ve çağrışım yolu ile öğrenme (associative mapping) yöntemleridir. Birinci yöntemde girdi ve çıktılar incelenerek kural tabanları oluşturulmaktadır. Çağrışım yolu ile öğrenme yöntemi ise iki alt başlıkta incelenebilir. Bunlar farklı çağrışım (hetero-association) ve aynı çağrışım (auto-association) yöntemleridir. Farklı çağrışım yönteminde, sistemin girdisi ve çıktısı birbirinden farklıdır. Bu yöntemde olayın genel yönleri ortaya çıkartılmakta ve benzeri durumlarda bu genellemeler kullanılarak problem çözülmektedir. Sınıflandırma ve tahmin problemlerinde yaygın olarak kullanılır. Aynı çağrışım yönteminde ise sistemin girdileri ve çıktıları aynıdır. Bu yöntemde olay öğrenilir ve öğrenilen bu olay sonraki bir durumda, örneğin bazı bilgilerin eksik olması durumunda, çıktı olarak istenir [2].

Farklı ve aynı çağrışım yöntemlerini bir örnek ile irdeleyelim. Fotoğraflar ile oluşturulan bir veritabanından iki fotoğrafın araştırıldığı farzedilsin. Bunlardan birinci fotoğrafdaki kişi sakal ve bıyık bırakmış olsun. İkinci fotoğraf ise yırtık ve eksik olsun. Birinci fotoğrafdaki kişinin veritabanından araştırılması, farklı çağrışım yolu ile öğrenme yöntemine bir örnek teşkil eder. Fotoğrafdaki yüz değişmiştir fakat genel özellikler ön plana çıkartılarak fotoğraf eşleştirilmeye çalışılır. İkinci fotoğrafdaki kişinin veritabanından araştırılması ise, aynı çağrışım yolu ile öğrenme yöntemine bir örnekdir. Girdi olarak verilen fotoğraf, çıktı olarak istenen fotoğrafın eksik ve yırtık halidir.

1.4 Yapay Sinir Ağlarında Öğrenme Biçimleri

Yapay sinir ağları yaklaşımında bilgilerin tanıtılması örnekler yolu ile olur. Örnekler, bir olayın girdi ve çıktılarını temsil etmektedir. Girdi ve çıktılar arasındaki ilişkiyi öğrenen yapay sinir ağının, verilecek farklı girişler için çıkışları üretebilecek bilgi ve tecrübeyi kazandığı söylenebilir. Örneğin bir sistemin birim basamak girişe verdiği cevap biliniyorsa, birim basamak giriş ve çıkış değerleri ile yapay sinir ağının, sistemin karakteristiğini öğrenmesi sağlanabilir. Artık yapay sinir ağı, sistemin diğer girişler için nasıl cevap vereceğini bilmektedir.

Örnekler oluşturulurken mutlaka giriş ve çıkış değerlerinin birlikte kullanılması gerekmez. Bazı durumlarda sadece giriş değerleri ile örnekler oluşturulabilir. Örneklerin nasıl oluşturulduğuna bağlı olarak yapay sinir ağlarının üç temel öğrenme biçimi vardır.

Öğretmenli (Supervised) Öğrenme : Örnekler girdi-çıktı kümesinin birlikte kullanılması ile oluşturulur. Sistem girişleri ve sistemin bu girişlere verdiği cevaplar, yapay sinir ağına tanıtılır ve yapay sinir ağının giriş çıkış arasındaki ilişkileri öğrenmesi istenir.

Destekleyici (Reinforcement) Öğrenme : Örnekler sadece giriş değerleri ile oluşturulur. Çıkış değerlerini yapay sinir ağına tanıtmak yerine, yapay sinir ağının bir çıkış üretmesi beklenir ve üretilen bu çıkışın doğru yada yanlış olmasına göre yapay sinir ağına, onaylayıcı yada reddedici bir sinyal gönderilir.

Öğretmensiz (Unsupervised) Öğrenme : Yapay sinir ağlarının öğrenmesine yardımcı olacak herhangi bir öğretmen yada destekleyici unsur yoktur. Sisteme sadece giriş değerleri tanıtılır. Bu değerler arasındaki ilişkileri, benzerlikleri ve farklılıkları yapay sinir ağının kendi kendisine öğrenmesi istenir. Fakat sistemin öğrenme süreci bittikten sonra çıktıların ne anlama geldiğini gösteren etiketlemenin kullanıcı tarafından yapılması gerekmektedir. Bu öğrenme biçimi daha çok sınıflandırma problemleri için kullanılır.

Karma Öğrenme : Yukarda bahsedilen öğrenme biçimlerinin birlikte kullanılmasıyla karma öğrenme gerçekleştirilebilir [1].

1.5 Yapay Sinir Ağlarında Öğrenme Kuralları

Farklı yaklaşımlar, farklı algoritmaların ve dolayısıyla farklı kuralların doğmasına sebep olmuştur. Yapay sinir ağlarında dört temel öğrenme kuralı vardır.

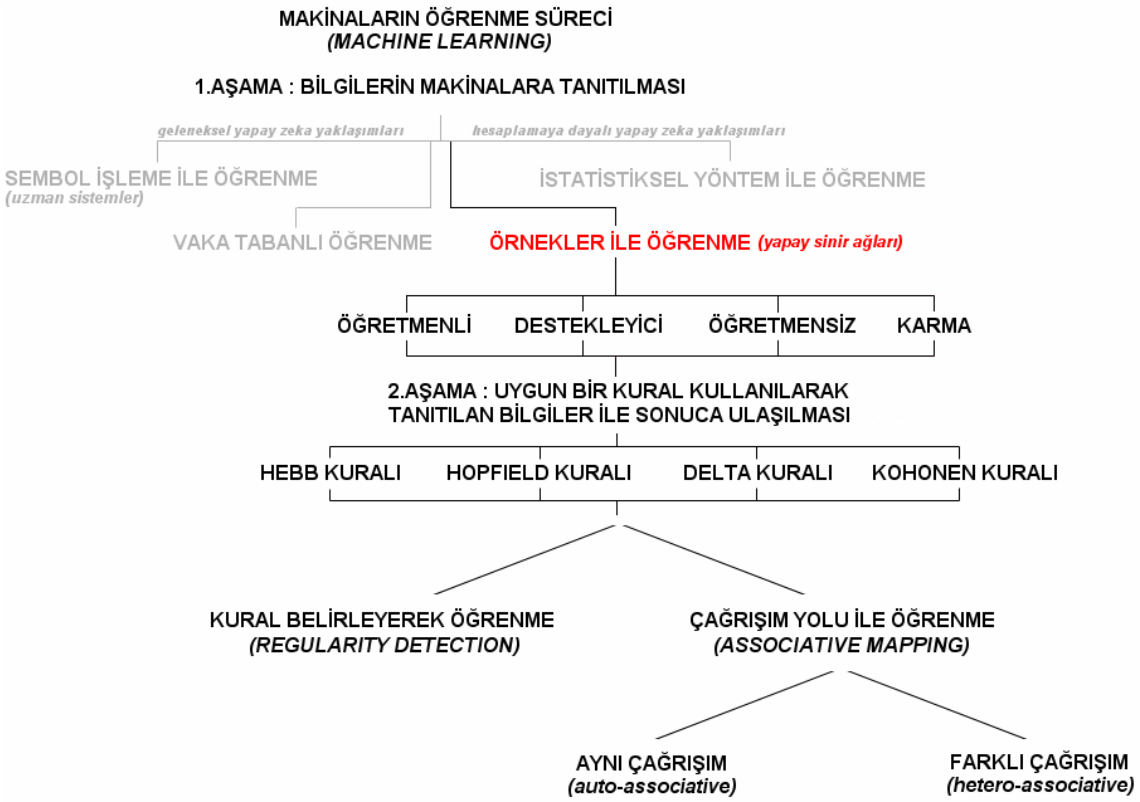
Hebb Kuralı : 1949 yılında geliştirilen Hebb kuralı diğer tüm kuralların temelini oluşturmaktadır. Bu kurala göre, eğer ard arda gelen iki hücrenin matematiksel işareti aynı ise, bu iki hücrenin bağlantısı kuvvetlendirilmelidir. Bağlantının kuvvetlendirilmesi, iki hücre arasındaki ağırlık değerinin artırılması ile yapılır. Eğer hücrelerin matematiksel işaretleri farklı ise bu durumda da hücreler arası bağlantılar zayıflatılır, diğer bir ifade ile ağırlıklar azaltılır.

Hopfield Kuralı : Hebb kuralından farklı olarak iki hücrenin bağlantılarının, ne kadar kuvvetlendirileceği yada zayıflatılacağı, öğrenme katsayısına göre belirlenir. Öğrenme katsayısı genel olarak 0 – 1 arasında kullanıcı tarafından seçilir.

Delta Kuralı : Ağırlık değerleri, yapay sinir ağının ürettiği çıkış değeri ile üretilmesi istenen hedef değer arasındaki hatanın karesel ortalamasını en az yapacak şekilde değiştirilir.

Kohonen Kuralı : Bu kurala göre yapay sinir ağındaki hücreler, ağırlıklarını değiştirmek için birbirleri ile yarışırlar. Çıkış değeri en büyük olan hücre yarışı kazanır ve ağırlık değerlerinin değiştirilmesine izin verilir [1].

Yukarda anlatılan kurallar, öğrenme sürecini çevrimiçi yada çevrimdışı yapabilmelerine göre sınıflandırılabilir. Örneğin, delta öğrenme kuralı çevrimdışı çalışırken, kohonen öğrenme kuralı ise çevrimiçi çalışabilmektedir. Öğrenme sürecinin tüm unsurları şekil 1.5.1'de özetlenmektedir.



Şekil 1.5.1 : Makinaların öğrenme süreci

2. YAPAY SİNİR AĞLARINA GİRİŞ

Bu bölümde, ağ yapılarını ayrıntılı bir şekilde incelemeye başlamadan önce yapay sinir ağları hakkında genel bir fikir vermek amaçlanmaktadır. Kısım 2.1'de yapay sinir ağlarının tanımı, eğitilmesi ve bilgilerin saklanması gibi genel özelliklere değinilecektir. Kısım 2.2'de yapay sinir ağlarının uygulamalarda hangi amaçlarla kullanıldığı anlatılacaktır. Kısım 2.3'de ağların hangi alanlarda kullanıldığı açıklanacak, teorik ve pratik uygulamalara örnekler verilecektir. Sırasıyla kısım 2.4 ve kısım 2.5'de yapay sinir ağlarının avantajlarına ve dezavantajlarına yer verilecektir. Kısım 2.6'da yapay sinir ağlarının kısa bir tarihi anlatılacaktır. Kısım 2.7'de ise en çok bilinen ağ yapılarına ilişkin bir sınıflandırma verilecektir.

2.1 Yapay Sinir Ağlarının Genel Özellikleri

Yapay sinir ağları, örnekler aracılığı ile olayları öğrenebilen, öğrendiği bilgileri kullanarak yeni bilgiler türetebilen ve keşfedebilen, böylelikle kazandığı bilgi, birikim ve tecrübeler ile çevreden gelen etkilere insanlarınkine benzer tepkiler verebilen sistemlerdir. Örneklerden elde ettikleri bilgiler ile kendi deneyimlerini oluşturur ve benzer konularda kararlar verebilirler. Bu özellikleri sayesinde ilişkilendirme, sınıflandırma ve optimizasyon konularında başarılı bir şekilde uygulanmaktadırlar [1].

Yapay sinir ağlarında yapılması gerekli ilk işlem ağın eğitilmesidir. Ağın eğitilmesi, eğitim verilerinin yapay sinir ağına tanıtılması ve bu verilerin temsil ettiği olaylar arasındaki ilişkilerin yapay sinir ağı tarafından belirlenmesidir. Eğitilen ağın, doğruluğunu ve güvenilirliğini ölçebilmek için test verileri kullanılır. Test verileri, eğitim verilerinin bir bölümü olabileceği gibi yapay sinir ağının eğitiminde kullanılmayan verilerde olabilir. Test amacı ile farklı verilerin kullanılması yapay sinir ağının doğruluğunun daha iyi ölçülmesini sağlayacaktır.

Yapay sinir ağı eğitildikçe hücreler arasındaki bağlantı değerleri, ki bunlara ağırlık değeri denilmektedir, değişir. Yapay sinir ağlarında bilgi, bu bağlantılardaki ağırlık değerleridir. Yapay sinir ağları dağıtık belleğe sahiptir, diğer bir ifade ile bilgi ağa yayılmış durumdadır. Ağ üzerindeki ağırlık değerleri, yani bilgiler tek başlarına bir anlam taşımazlar, çünkü öğrenilen olay tüm ağ tarafından karakterize edilir [1].

2.2 Yapay Sinir Ağlarının Yetenekleri

Yapay sinir ağları uygulamalarda genellikle şu fonksyonları yerine getirmek üzere kullanılırlar ;

Tahmin : Yapay sinir ağına tanıtılan örneklerden yararlanılarak, verilen giriş değerleri için çıkış değerleri tahmin edilir. Hava tahmini, döviz kurlarının tahmini gibi örnekler vermek mümkündür.

Veri Filitreleme : Bu amaçla eğitilen ağlar, birçok veri arasından uygun olanları belirleme görevini yerine getirirler. Telefon konuşmalarındaki gürültüleri asıl konuşmalardan ayıran ağlar bu konudaki uygulamalara örnek olarak verilebilir.

Arıza Tespiti ve Teşhisi : Bu amaçla geliştirilen ağlar, makinalarda, sistemlerde yada en genel halde süreçlerde meydana gelen olumsuzlukların ve problemlerin ortaya çıkartılmasında, arızaların tespit edilmesinde kullanılmaktadır.

Sınıflandırma : Yapay sinir ağları, sınıflandırma uygulamalarında başarılı bir şekilde kullanılmaktadır. Bu uygulamalarda örnekler, belirli sınıflara ayrıştırılarak kümelendirilmekte ve daha sonra gelen bir örneğin hangi sınıfa ait olduğu belirlenmektedir. Birçok problem, sınıflandırma problemi haline getirilmekte ve yapay sinir ağları kullanılarak çözülebilmektedir. Bir makina üzerinde görülen hataların sınıflandırılması bu konudaki uygulamalara örnek olarak verilebilir.

Eksik Verilerin Tamamlanması : Bu amaçla eğitilen ağlar, verilerin hatalı ve eksik olup olmadığını belirler. Eksik olan bilgileri tamamlar. Yırtık bir fotoğrafın tamamlanması bu konudaki uygulamalara örnek olarak verilebilir.

Tanıma ve Eşleştirme : Bu amaçla eğitilen ağlar, değişik şekil ve örüntülerin tanınmasında, eksik, karmaşık, belirsiz bilgilerin işlenerek eşleştirilmesinde kullanılmaktadır. Yapay sinir ağları günümüzde geliştirilmiş en mükemmel şekil tanıyıcı ve sınıflandırıcı olarak kabul edilmektedir. Yapay sinir ağlarının, rakiplerine kıyasla bu kadar üstün olmasını sağlayan eksik bilgiler ile çalışabilme ve normal olmayan verileri işleyebilme yetenekleridir [1].

2.3 Yapay Sinir Ağlarının Kullanıldığı Alanlar

Yapay sinir ağlarının kullanılabileceği birçok alan vardır. Dolayısıyla yapay sinir ağlarının kullanıldığı ve başarılı sonuçlar elde edilen yüzlerce uygulama sayılabilir. Örneğin, 1997 yılında Caere firması tarafından üretilen optik karakter okuma sistemi yılda 3 milyon$'dan fazla gelir getirmiştir. Aynı yıl HNC firması tarafından pazarlanan ve kredi kartlarının haksız yere kullanılmasını ortaya çıkartan Falcon isimli yapay sinir ağı sistemi yılda 23 milyon$ kar sağlamıştır. 1998 yılında Sensory firması tarafından geliştirilen ses tanıma sistemindeki yonganın 5$'a mal olduğu ve bir milyondan fazla sattığı bilinmektedir.

Bu örneklerin çoğaltılması mümkündür. Fakat herhangi bir problemin çözümü için yeterli etkinlikte ve verimlilikte bir yöntem varsa, bu problemi yapay sinir ağı ile çözmek mantıklı olmayabilir. Başarılı uygulamalar incelendiğinde, yapay sinir ağlarının, doğrusal olmayan, çok boyutlu, gürültülü ve hata olasılığı yüksek sensör verilerinin bulunduğu, kesin olmayan, karmaşık, verilerin eksik olduğu, ortaya çıkan çözümde matematiksel modele ve algoritmaya ihtiyaç duyulmayan hallerde yaygın olarak kullanıldıkları görülmektedir [1].

a) Yapay Sinir Ağlarının Kullanıldığı Teorik Uygulamalar

- Doğrusal olmayan sistem modelleme
- Akıllı kontrol
- Sinyal filtreleme ve doğrusal olmayan sinyal işleme
- Optimizasyon
- Probabilistik fonksyon kestirimleri
- Sınıflandırma
- Örüntü tanıma, ilişkilendirme ve eşleştirme

b) Yapay Sinir Ağlarının Kullanıldığı Pratik Uygulamalar

- Kimyasal proseslerin modellenmesi
- Akıllı araçlar ve robotlar için optimum rota belirleme
- Robotlarda görme sistemlerinin ve hareket mekanizmalarının kontrol edilmesi
- Makina, sistem ve süreçler için arıza tespiti
- İletişim kanallarındaki ekoların filtrelenmesi, anahtarlama ve yoğunluğun kontrolü
- Hedef tanıma ve takip sistemleri
- Radar ve sonar sinyallerinin sınıflandırılması
- Radar ve görüntü sinyalleri işleme
- Güvenlik sistemlerinde konuşma ve parmak izi tanıma
- Mekanik parçaların ömürlerinin ve kırılmalarının tahmin edilmesi
- Endüstriyel ürünlerin görsel kalite kontrolü ve imalatta meydana gelen hataların tespiti
- Kredi kartı hilelerinin tespiti
- Döviz kuru tahminleri, risk analizleri

2.4 Yapay Sinir Ağlarının Avantajları

Gerçek dünyada olaylar birçok parametreye bağlı olabilir. Ayrıca bu parametrelerin birbirleri üzerinde, açıkça görülemeyen ilişkileri ve etkileri olabilir. Geleneksel yöntemler kullanıldığında bu ilişkileri belirlemek, eğer belirlenemiyorsa yok saymak gerekmektedir. Oysaki yapay sinir ağları kullanıldığında, kullanıcının bu ilişkileri bilmesi ve ağa söylemesi beklenmemektedir. Örneklerin dışında herhangi bir ön bilgiye ihtiyaç yoktur. Yapay sinir ağı, bu ilişkileri, örnekler yardımıyla kendisi öğrenir.

Olayların bağlı olduğu parametrelerin tümü bilinse bile, parametreler arasındaki ilişkiler doğrusal olmayabilir. Geleneksel yöntemler kullanıldığında, varsayımlarla çalışma noktaları civarında lineerleştirmeler yapılır. Bu durum, oluşturulan modelin, gerçek sisteme olan uygunluğunu azaltmakta ve gerçek sistemin kontrolünü zorlaştırmaktadır. Yapay sinir ağlarında ise ilişkilerin doğrusal olup olmaması problem teşkil etmez.

Yapay sinir ağları, yeni bilgilerin ortaya çıkması ve sistemde bazı değişikliklerin olması durumunda yeniden eğitilebilirler, hatta bazı ağların eğitilmesine bile gerek yoktur.

Bilgilerin eksik olduğu durumlarda, yapay sinir ağları etkin çözümler üretebilmektedir. Ayrıca yapay sinir ağlarının belirsiz bilgileri işleyebilme yetenekleri vardır.

Yapay sinir ağları hata toleransına sahiptir. Ağın bazı hücrelerinin bozulması ve çalışamaz duruma gelmesi halinde bile ağ çalışmaya devam eder. Fakat ağın bozuk olan hücrelerinin önemine göre performansda düşmeler görülebilir.

Yapay sinir ağlarının paralel çalışabilmeleri, gerçek zamanlı kullanımlarını kolaylaştırmaktadır [1, 2].

2.5 Yapay Sinir Ağlarının Dezavantajları

Biyolojik sinir ağlarının taklit edilme sebeplerinden en önemlisi paralel çalışabilme yeteneğidir. Dolayısıyla paralel çalışma, yapay sinir ağınında en önemli özelliğidir. Yapay sinir ağlarının bu özelliğinin etkin bir biçimde kullanılabilmesi paralel çalışabilen işlemcilere bağlıdır. Oysaki günümüzde yaygın olarak kullanılan bilgisayar sistemleri seri şekilde çalışmakta ve aynı anda sadece tek bir bilgiyi işleyebilmektedir. Bu durum yapay sinir ağının tam performansında çalışamamasına sebep olmakta ve zaman kaybına yol açmaktadır. Paralel şekilde çalışabilen işlemciler kullanılsa bile, bu işlemcilerin birbiri ile paralel çalıştırılabilmesi mümkün olmayabilir.

Probleme uygun ağ yapısının belirlenmesi genellikle deneme yanılma yolu ile yapılmaktadır. Uygun ağın oluşturulamaması, düşük performanslı çözümlere neden olabilir. Uygun ağ oluşturulduğunda ise iyi bir çözüm bulunabilir fakat yapay sinir ağı bu çözümün en iyi (optimum) çözüm olduğunu garanti etmez. Oysaki geleneksel yöntemler ile optimum çözümler üretilebilir.
Bazı ağlardaki öğrenme katsayısını, gizli katman sayısını ve gizli katmanlardaki nöron sayılarını belirlemek için genel geçer bir kural yoktur. Bu durum iyi çözümler

bulmayı güçleştirir. Yapay sinir ağı parametreleri, her problem için ayrı faktörler dikkate alınarak tasarlayıcının tecrebesine bağlı olarak belirlenir.

Yapay sinir ağları sadece sayısal veriler ile çalışırlar. Sembolik ifadelerin nümerik gösterime çevrilmesi gerekmektedir.

Yapay sinir ağının ne kadar eğitileceğine keyfi olarak karar verilir. Ağın yaptığı hatanın belirli bir değerin altına düşmesi, eğitimin tamamlanması için yeterli görülmektedir. Fakat en iyi (optimum) öğrenmenin gerçekleştirildiği söylenememektedir. Optimum sonuçlar veren bir mekanizma henüz geliştirilememiştir. "Yapay sinir ağlarında en iyi çözüm" ucu açık ve araştırılan bir konudur.

Bütün bu dezavantajlara rağmen, yapay sinir ağları ile değişik şekilde çözümler üretilmekte ve uygulamalarda başarılı şekilde kullanılabilmektedir [1, 2].

2.6 Yapay Sinir Ağlarının Tarihi

Yapay sinir ağlarının ilk olarak ortaya çıkışı, bilim adamlarının nörobiyoloji konusuna ilgi duyması ve elde ettikleri bilgileri bilgisayar sistemleri ile birleştirmeleri sayesinde olmuştur. Yapay sinir ağlarının yadsınamaz katkıları ile aritmetik işlemleri yapan bilgisayarlar, bugün olayları öğrenebilen ve çevre şartlarına göre karar verebilen sistemler haline dönüşmüşlerdir. Bu konuda yapılan çalışmalar, 1970 yılında yaşanan dönüm noktasından ötürü iki kısımda incelenebilir [1].

2.6.1 Yapay Sinir Ağlarının Tarihi - 1970 Öncesi Çalışmalar

1890 yılında beyin fonksyonları hakkında bilgi veren ilk eser yayınlanmıştır. 1940'dan önceki yıllarda Helmholtz, Pavlov ve Poincare'in yapay sinir ağı kavramı üzerinde çalıştıkları bilinmektedir. Fakat bu çalışmalar mühendislik disiplinlerine dayanmamaktadır. 1940'lı yıllardan sonra Hebb, McCulloch ve Pitts yapılan araştırmaları mühendislik alanlarına kaydırmış ve günümüzdeki yapay sinir ağlarının temellerini atmışlardır. İlk yapay sinir hücresini oluşturmuşlardır. Yapay sinir hücreleri ile her türlü mantıksal ifadeyi formülüze etmenin mümkün olduğunu göstermişlerdir. 1949 yılında Donald Hebb, yapay sinir ağının değerlerini değiştiren

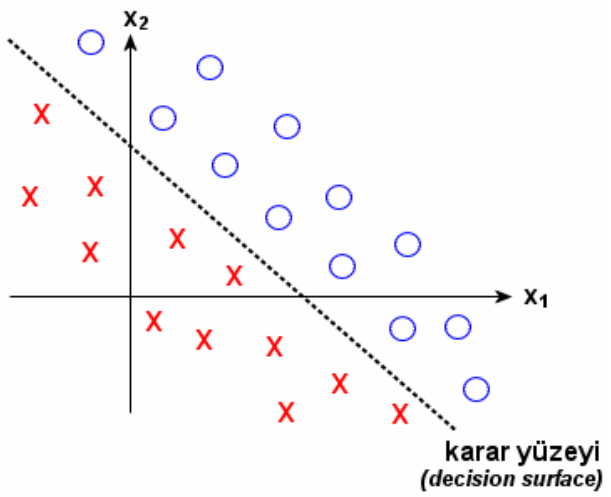
bir öğrenme kuralı geliştirmiştir. Hebbian öğrenme diye bilinen bu kural, günümüzde de birçok öğrenme kuralının temelini oluşturmaktadır. 1951 yılında ilk nöro-bilgisayar üretilmiştir. 1954 yılında Farley ve Clark tarafından adaptif tepki üretme kavramı ortaya atılmıştır. Bu kavram 1958 yılında Rosenblatt ve 1961 yılında Caianiello tarafından geliştirilmiştir. Özellikle Rosenblatt tarafından geliştirilen algılayıcı model (perceptron), yapay sinir ağlarının gelişim sürecinde bir dönüm noktası sayılan çok katmanlı algılayıcıların temelini oluşturmuştur. 1962 yılında Widrof ve Hoff, ADALINE (ADAptive LInear NEuron) modelini ortaya atmışlardır. Bu model, Rosenblatt'ın perceptron modeline benzemekle birlikte, daha gelişmiş bir öğrenme algoritmasına sahiptir. Adaptif öğrenmeninde temelini oluşturan MADALINE modeli, bu çalışmanın neticesinde ortaya çıkmıştır.

Bu arada bilim dünyasında başka gelişmelerde olmuştur. 1956 yılında Yapay Zeka kavramı ortaya atılmış ve bilim çevreleri tarafından kabul görmüştür. İlk yapay zeka çalışmalarında, sinir ağlarına değinilmemiştir. Nöro-bilgisayarlar ve yapay sinir ağları güncelliğini kaybetmiş, tüm ilgi yapay zeka çalışmalarına odaklanmıştır. Yapay sinir ağı savunucularından Grosberg, Kohonen, Rosenblatt, Widrow, Nilssons ve Fukushima'nın yaptığı çalışmalar, özellikle 1965 yılında Nilssons tarafından kaleme alınan ve o tarihe kadar yapılan teorik çalışmaların bir araya getirildiği "Öğrenen Makinalar" isimli kaynak, yapay zekanın yükselmesine ve yapay sinir ağlarının önüne geçmesine engel olamamıştır.

1969 yılında, yapay sinir ağı çalışmaları, yapay zeka biliminin önde gelen isimlerinden Misnky ve Pappert tarafından yazılan Algılayıcılar (perceptrons) isimli kitap yüzünden duraklama devrine girmiştir. Bu kitapta araştırmacılar, yapay sinir ağlarına dayalı algılayıcıların bilimsel bir değer taşımadığını çünkü doğrusal olmayan problemlere çözüm üretemediğini iddia etmişlerdir. Bu tezlerini desteklemek amacı ile XOR probleminin yapay sinir ağı algılayıcıları tarafından çözülememesini örnek göstermişlerdir. Bu örnek ve ortaya atılan iddia, bilim çevreleri tarafından kabul görmüş ve yapay sinir ağı çalışmaları durma noktasına gelmiştir. Araştırma geliştirme çalışmaları yürüten ve eşgüdümü sağlayan bir organizasyon olan DARPA, yapay sinir ağları ile ilgili çalışmaları desteklemeyi durdurmuştur [1, 7].

XOR problemini bu kadar özel yapan nedir ?

İki-boyutlu bir öklid uzayı düşünelim. Bu öklid uzayındaki (x_1, x_2) noktalarının her birisine "χ" yada "o" sembollerinin karşılık geldiğini farzedelim. Bu semboller iki boyutlu öklid uzayında şekil 2.6.1'de gösterildiği gibi dağılmış olsun.

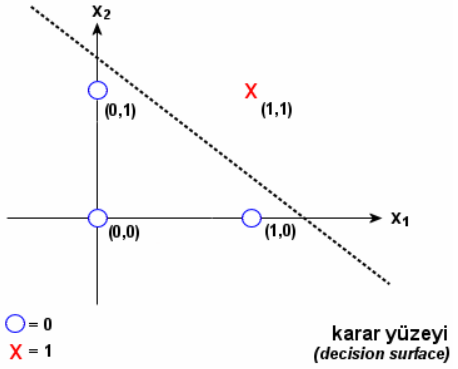


Şekil 2.6.1 : İki boyutlu öklid uzayında (x_1,x_2) ikililerine karşılık gelen sembollerin dağılımı

Şekil 2.6.1'deki "χ" ve "o" sembolleri, tek bir doğru ile izole edilebilmiştir. Bu sebeple, "χ" ve "o" sembolleri lineer ayrıştırılabilirdir (linearly separable) [3].

Örneğin lojik-VE fonksyonu lineer ayrıştırılabilirdir. Çünkü tek bir doğru ile 0 ve 1 çıkışları izole edilebilmektedir. Detaylı açıklamaları ileriki bölümlere bırakarak, bu geometrik yorumu, yapay sinir ağları ile ilişkilendirmek istersek, şekil 2.6.2'de gösterilen doğrunun, tek katmanlı bir algılayıcıya tekabül ettiğini söyleyebiliriz.

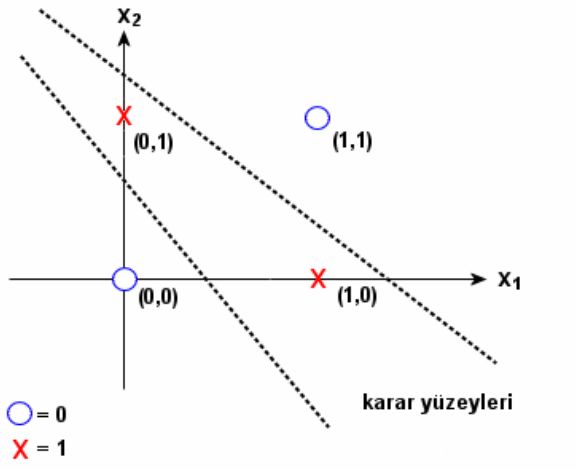
X1	X2	VE
0	0	0
0	1	0
1	0	0
1	1	1



Şekil 2.6.2 : Lojik VE fonksyonuna ait doğruluk tablosu ve geometrik yorumu

Fakat lojik-ÖZELVEYA (XOR) fonksyonu lineer ayrıştırılamamaktadır. 0 ve 1 çıkışlarını izole edebilmek için tek bir doğru yeterli değildir.

X1	X2	Özel VEYA
0	0	0
0	1	1
1	0	1
1	1	0



Şekil 2.6.3 : Lojik ÖZELVEYA fonksyonuna ait doğruluk tablosu ve geometrik yorumu

XOR problemini bu kadar özel kılan, **en basit non-lineer problem** oluşudur. XOR probleminin, lineer yaklaşımlar ile çözülmesi mümkün değildir [3].

2.6.2 Yapay Sinir Ağlarının Tarihi - 1970 Sonrası Çalışmalar

1972 yılında elektrik mühendisi Kohonen ve nöropsikolojist Anderson, çağrışımlı bellek (associative memory) konusunda benzer çalışmalar yayınladılar. Bu çalışmalar öğretmensiz öğrenme kurallarının temelini oluşturdu. Bu yıllarda Grosberg, yapay sinir ağlarının psikolojik mantıksallığını ve mühendislik konularına kolaylıkla uygulanabileceğini gösterdi ve 1978 yılında Carpenter'la birlikte yürüttüğü ortak çalışma ile Adaptif Rezonans Teorisini (ART) geliştirdi. ART, öğretmensiz öğrenme konusunda zamanının en karmaşık yapay sinir ağı olma özelliğini taşıyordu.

1982 ve 1984 yıllarında Hopfield tarafından yayınlanan çalışmalar ile yapay sinir ağlarının genelleştirilebileceği ve özellikle geleneksel programlama teknikleri ile çözülmesi zor olan problemlere çözüm üretilebileceği gösterildi. Gezgin satıcı probleminin çözülmesi bunun en güzel örneğiydi. Hopfield'in çalışmaları, Hinton ve arkadaşlarının geliştirdikleri Boltzman makinasının temellerini oluşturdu.

1982 yılında, Rumelhart ve arkadaşları paralel programlama ve çok katmanlı algılayıcılar (multilayer perceptron) konularındaki çalışmalarını iki ciltlik bir eserde topladılar. Çok katmanlı algılayıcıların bulunması yapay sinir ağlarının tarihsel gelişimi bakımından çok önemli bir adım oldu. Çünkü tek katmanlı algılayıcıların

çözemediği XOR problemi çok katmanlı algılayıcıların geliştirilmesi ile çözüme ulaştı. Yeni bir yapay sinir ağı dalgası, tüm disiplinlerde kendini göstermeye başladı. Çok katmalı algılayıcılar sadece XOR problemini çözmekle kalmadı, aynı zamanda Hopfield ağlarının ve Boltzman makinasının kısıtlarınında aşılmasını sağladı.

1988 yılında Broomhead ve Lowe, radyal tabanlı fonksyonlar modelini (radial basis functions) geliştirdiler. Çok katmanlı algılayıcılar modeline alternatif olarak geliştirilen bu model, özellikle filtreleme problemlerinde çok başarılı oldu.

1990 ve 1991 yıllarında Specht, temelini radyal tabanlı fonksyonlar modelinin oluşturduğu, probabilistik ağları (probabilistic neural networks) ve genel regrasyon ağlarını (general regression neural networks) geliştirdi [1, 7].

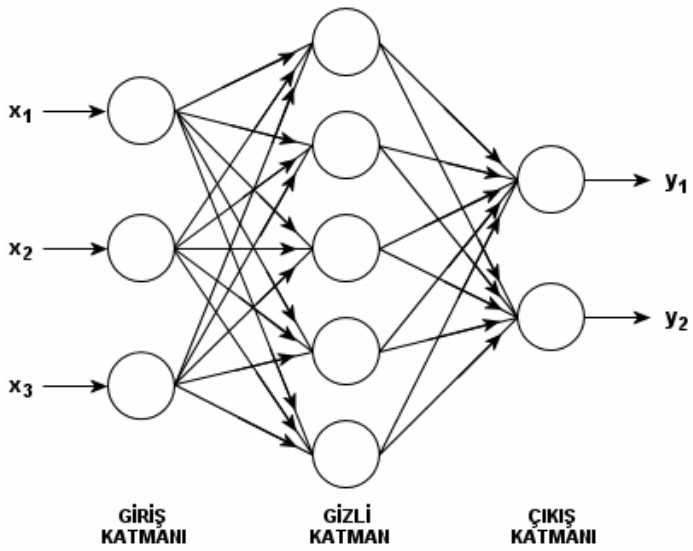
Yapay sinir ağlarının gelişim süreci halen devam etmektedir. Yapılan konferanslarda ve sempozyumlarda sinir ağları tartışılmakta ve yeni modeller, yeni öğrenme teknikleri ortaya atılmaktadır. Bununla birlikte, donanım teknolojisindeki gelişmeler sayesinde, sadece kağıt üzerindeki çalışmalar olmaktan çıkmış, günlük hayatta kullanılan pratik uygulamalar haline gelmiştir.

2.7 Yapay Sinir Ağlarının Sınıflandırılması

Yapay sinir ağlarına olan ilgi, her geçen gün yeni bir model ve yeni bir öğrenme kuralının geliştirilmesini sağlamaktadır. Bu kısımda sadece en çok bilinen yapay sinir ağı modellerine değinilicektir.

Öğrenme algoritmaları en genel halde, öğretmenli (supervised) ve öğretmensiz (unsupervised) olarak iki grupta incelenebilir. Öğretmenli öğrenme algoritmalarında, örnekler girdi-çıktı kümesinin birlikte kullanılması ile oluşturulur. Sistem girişleri ve sistemin bu girişlere verdiği cevaplar, yapay sinir ağına tanıtılır ve yapay sinir ağının giriş-çıkış arasındaki ilişkileri öğrenmesi istenir. Öğretmensiz öğrenme algoritmalarında, sisteme sadece girdi değerleri tanıtılır. Giriş değerleri arasındaki ilişkileri, benzerlikleri ve farklılıkları yapay sinir ağının kendi kendisine öğrenmesi istenir. Bu öğrenme biçimi daha çok sınıflandırma problemleri için kullanılır [4].

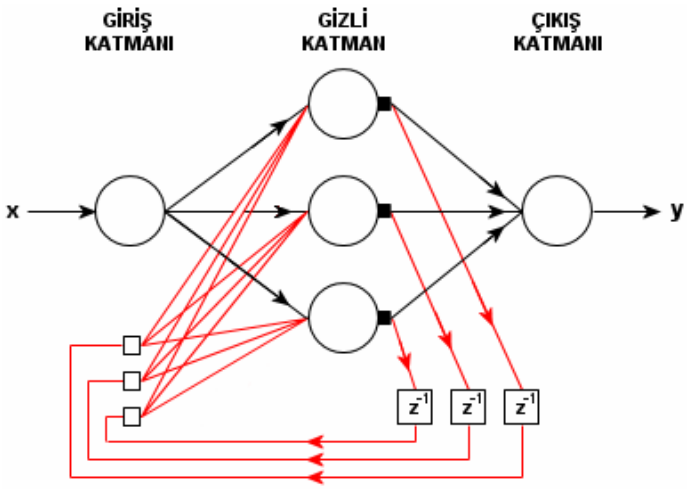
Yapay sinir ağları, ağ yapıları bakımından ileribeslemeli (feedforward) ve geribeslemeli (feedback yada recurrent) olarak gruplandırılabilir. İleribeslemeli ağlarda, bağlantılar kapalı çevrimler oluşturmaz, diğer bir ifade ile herhangi bir nöronun çıkışı, sadece ve sadece bir sonraki katmanda bulunan nöronlar için giriş olarak kullanılır, bu çıkış önceki katmandaki veya aynı katmandaki nöronlara giriş olarak verilemez. İleribeslemeli ağlar, geleneksel nümerik yöntemler ile eğitilebilir. Şekil 2.7.1'de 3-girişli 2-çıkışlı çok katmanlı ileribeslemeli bir ağ yapısı gösterilmektedir.



Şekil 2.7.1 : 3-girişli 2-çıkışlı çok katmanlı ileribeslemeli ağ yapısı

Geribeslemeli ağlarda ise bağlantılar kapalı çevrimler oluşturur. Herhangi bir nörondan alınan çıkış, aynı katmandaki veya bir önceki katmandaki nöronlara giriş olarak verilebilir. Geribeslemeli ağların eğitim süreleri daha uzun, eğitim yöntemleri ise daha karmaşıktır.

Şekil 2.7.2'de 1-girişli 1-çıkışlı Elman-tipi geribeslemeli bir ağ yapısı gösterilmektedir.



Şekil 2.7.2 : 1-girişli 1-çıkışlı Elman-tipi geribeslemeli ağ yapısı

Yapay sinir ağları, çalıştıkları veriler bakımından kategorik (categorical) ve nicel (quantitative) olmak üzere iki grupta incelenebilir. Kategorik veriler yeni, eski, kırmızı, lacivert vb. gibi nitel ifadeler olsa bile, bu sembolik ifadeler, nümerik olarak kodlanmalı ve nümerik değerler ile çalışılmalıdır. Sınıflandırma problemlerinde, öğretmenli öğrenme algoritmalarındaki hedef değerler ve öğretmensiz öğrenme algoritmalarındaki çıkış değerler kategorikdir. Nicel ifadeler ise herhangi bir parametrenin nümerik değeri olabilir. Bazı veriler ise hem kategorik hemde nicel olarak yorumlanabilir. İkilik tabandaki "0" ve "1" değerleri buna örnek olarak verilebilir.

Bu bilgiler ışığı altında en çok bilinen yapay sinir ağı modelleri aşağıdaki gibi sınıflandırılabilir [4].

1. Öğretmenli Algoritmalar

A. İleribeslemeli Modeller

A1.Lineer Algılayıcılar (Linear)

- Hebbian (1949 - Hebb)
- Perceptron (1958 - Rosenblatt)
- Adaline (1960 – Widrow, Hoff)
- Functional Link (1989 - Pao)

A2.Çok Katmanlı Algılayıcılar (Multilayer Perceptron)

- Backpropogation (1989 - Rumelhart, Hinton, Williams)
- Cascade Correlation (1990 - Fahlman, Lebiere)
- Quickpropogation (1989 - Fahlman)
- RPROP (1993 - Riedmiller, Braun)

A3.Radyal Tabanlı Ağlar (Radial Basis Functional Networks)

- Orthogonal Least Squares (1991 - Chen, Cowan, Grant)

A4.Cerebellar Model Articulation Controller (1995 - Albus)

A5.Classification

- LVQ : Learning Vector Quantization (1988 - Kohonen)
- PNN : Probabilistic Neural Network (1990 - Specht)

A6.Regression

- General Regression Neural Network (1991 - Specht)

B. Geribeslemeli Modeller

B1. Bidirectional Associative Memory (1992 - Kosko)

B2. Boltzman Machine (1985 - Ackley)

B3. Recurrent Time Series

- Backpropagation Through Time (1990 - Werbos)
- Elman (1990 - Elman)
- Finite Impulse Response (1990 - Wan)
- Jordan (1986 - Jordan)
- Real Time Recurrent Network (1989 - Williams, Zipser)
- Recurrent Backpropagation (1989 - Pineda)
- Time Delay Neural Networks (1990 - Lang, Waibel)

C. Yarışa Dayalı Modeller

C1. ARTMAP : Adaptive Resonance Theory with Mapping

C2. Fuzzy ARTMAP (1992 - Carpenter, Grossberg)

C3. Gaussian ARTMAP (1995 - Williamson)

C4. Counterpropagation (1987 - Hecht Nielsen)

2. Öğretmensiz Algoritmalar

A. Yarışa Dayalı Modeller

A1. Vector Quantization

- Grossberg (1976 - Grossberg)
- Kohonen (1984 - Kohonen)
- Conscience (1988 - Desieno)

A2. SOM : Self Organizing Map

- Kohonen (1995 - Kohonen)
- GTM : Generative Topographic Map (1997 - Bishop)
- Local Linear (1995 - Mulier, Cherkassky)

A3. ART : Adaptive Resonance Theory

- ART1 (1987 - Carpenter, Grossberg)
- ART2 (1987 - Carpenter, Grossberg)
- ART3 (1990 - Carpenter, Grossberg)
- Fuzzy-ART (1991 - Carpenter, Grossberg)

A4. DCL : Differential Competitive Learning (1992 - Kosko)

B. Boyut Azaltmaya Dayalı Modeller

B1. Hebbian (1949 - Hebb)

B2. Sanger (1989 - Sanger)

B3. Differential Hebbian (1992 - Kosko)

C. Çağrışımlı Öğrenmeye Dayalı Modeller

C1. Linear Autoassociator (1977 - Anderson)

C2. BSB : Brain State in a Box (1977 - Anderson)

C3. Hopfield (1982 - Hopfield)

3. YAPAY SİNİR AĞLARININ YAPISI

Bu bölümde yapay sinir ağlarının yapısı incelenecektir. Kısım 3.1'de doğadaki canlılardan esinlenerek yapılmış ve başarılı olmuş uygulamalara örnekler verilecektir. Kısım 3.2'de modellemeye çalıştığımız mekanizmanın biyolojik özelliklerine değinilecektir. Kısım 3.3'de yapay sinir ağlarının temel birimi olan sinir hücreleri incelenecektir. Kısım 3.4'de temel birim olan hücrelerin bir araya gelmesiyle oluşturulan topoloji tetkik edilecektir. Kısım 3.5'de ise farklı modellerin ortaya çıkmasını sağlayan unsurlar anlatılacaktır.

3.1 Teknoloji ve Biyolojik Yapılar

Günümüz teknolojisi, biyolojik yapılardan esinlenerek bulunmuş yada geliştirilmiş örneklerle doludur. Uçağın mucidi olarak kabul edilen Wright kardeşler, Kittyhawk adındaki uçaklarının kanatlarını yaparken akbaba kanatlarının yapısını örnek almışlardır. Planörlerin kanat yapısı albatroslardan örnek alınarak tasarlanmıştır.

Yunusların burun çıkıntısı, modern gemilerin pruvasına örnek olmuştur. Yunus burnu şeklindeki bu tip pruvalar sayesinde %25'e ulaşan oranda yakıt tasarrufu sağlanmaktadır. Yunusların derisi ise denizaltı tasarlayan mühendislere ilham vermiştir. Yunus derisi üç katmanlı bir yapıdadır. Dıştaki katman ince ve çok esnektir, içteki katman kalındır, orta katman ise süngerimsi bir maddeden oluşur. Alman denizaltı mühendisleri, bu özelliğe sahip sentetik bir kaplama yapmayı başarmışlardır. Bu kaplamaların kullanıldığı denizaltıların hızlarında %45 oranında bir artış görülmüştür.

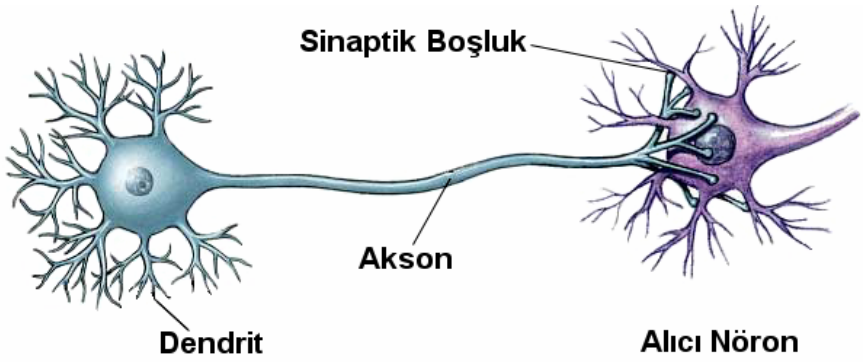
Yarasalar ultrason denilen çok yüksek titreşimli ses dalgaları yayarlar. Bu ses dalgaları havadaki ve yerdeki hayvanlara veya yarasanın önündeki diğer engellere çarparak yansır. Yarasa, yansıyan ve kendine gelen bu titreşimlere göre yönünü ve hareket hızını belirler. Radarın çalışma prensibi, yarasanın yön ve hız tayini yaptığı mekanizma ile aynıdır.

Arıların gözleri yanyana dizilmiş altıgenlerden oluşur. Gök cisimlerinden gelen x-ışınlarını toplamak için kullanılan uzay teleskoplarının merceği arıların göz yapısından ilham alınarak, yanyana dizilmiş altıgen şeklindeki aynalardan üretilmektedir. Altıgenlerden oluşan dizilim teleskopa geniş bir görüş alanı ve yüksek kalite sağlamaktadır.

Bilimadamlarının ve mühendislerin taklit etmeye çalıştığı bir diğer biyolojik yapı ise insan beyni ve sinir ağlarıdır. Beyin ve sinir ağları algılama, öğrenme, düşünme, problem çözme, hafızada tutma ve hafızada tutulan bilgileri tekrar çağırma gibi yetenekleri olan karmaşık bir yapıya sahiptir. Ağ yapılarını incelemeye başlamadan önce modellemeye çalıştığımız mekanizmanın biyolojik özellikleri incelenecektir [5, 6].

3.2 Biyolojik Sinir Ağlarının Yapısı

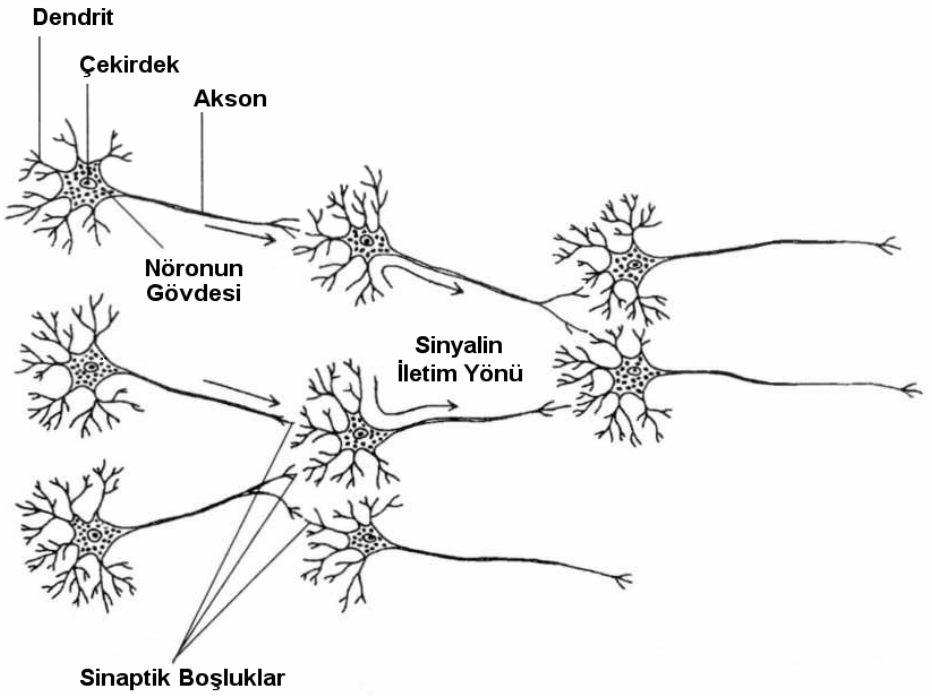
Sinir sistemi, genel olarak merkezi sinir sistemi ve çevresel sinir sistemi olmak üzere iki kısma ayrılır. Beyin ve omurilik merkezi sinir sistemini oluşturur. Çevresel sinir sistemi ise, vücudun sensörlerinden alınan bilgileri (görme, işitme, ağrı, ısı, titreşim, vb.) merkezi sinir sistemine taşıyan ve merkezden alınan emirleri ilgili birimlere (kas, salgı bezleri, vb.) ulaştıran sinir ağlarından meydana gelmiştir. Sinir ağının temel birimi sinir hücreleridir (nöron). Nöronların birbirlerini uyarması ve böylelikle bilginin bir nörondan diğer nörona geçmesiyle elektriksel sinyaller sinir ağı boyunca taşınır. Şekil 3.2.1'de biyolojik bir nöronun şematik yapısı gösterilmektedir.



Şekil 3.2.1 : Sinir ağlarının temel birimi olan sinir hücresinin (nöronun) şematik yapısı

Nöronların bilgiyi aldıkları noktalara dendrit adı verilir. Eğer nörona gelen işaret yeteri kadar güçlüyse, nöron bir çıkış işareti üretir. Üretilen bu çıkış işareti bir sonraki nöronun yada nöronların dendritlerine, aksonlar vasıtasıyla iletilir. İşaretin, verici nöronun aksonundan, alıcı nöronun dendritlerine iletilmesi sinapslar (sinaptik boşluk) sayesinde gerçekleşir. Sinaptik boşluklar, nöro-transmitter sıvısı ile doludur. Bu sıvı iletilen işaretin güçlendirilmesini veya zayıflatılmasını sağlar. Verici nörondan gelen işaret, nöro-transmitter sıvısı içindeki kimyasalların yer değiştirmesine sebep olur. Kimyasalların yer değiştirmesi ile meydana gelen potansyel sayesinde oluşan elektrik sinyali alıcı nöronun dendritine ulaşır. Öğrenme ve hafızaya alma olayları, sinaptik boşluğun empedansının yada kondüktansının ayarlanması ile mümkün olmaktadır [7].

Beynin kıvrımlı bölümü olan beyin kabuğunda (korteks), yaklaşık 100 milyar nöron bulunur. Herbir nöronun yaklaşık 1000 nöron ile bağlantısı vardır. İki nöronun bağlantısını sinaptik boşluklar sağladığından, toplam 100 trilyon sinaptik boşluğun olduğu söylenebilir. Şekil 3.2.2'de 100 milyar nöronun oluşturduğu sinir ağının küçük bir kısmı gösterilmektedir.



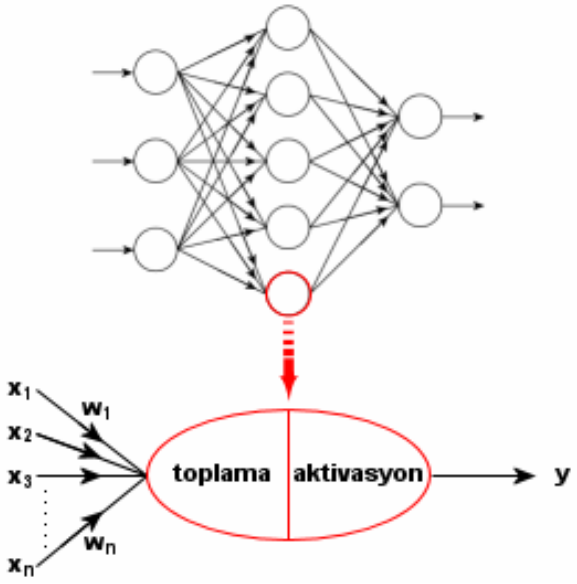
Şekil 3.2.2 : Vücudumuzdaki sinir ağının küçük bir kısmı

Bu rakamlar en karmaşık yapay sinir ağı ile karşılaştırıldığında bile çok büyüktür. Biyolojik sinir ağlarının, yapay sinir ağlarına olan sayıca üstünlüğü açıkça görülmektedir. Fakat insan beyninin 400 - 500 [Hz] ile, işlemcilerin ise 3 - 4 [GHz] ile çalıştığı düşünülürse, bilgisayar tabanlı sistemlerin hız bakımından daha avantajlı olduğu söylenebilir. Buna rağmen, seri olarak çalışan işlemciler, paralel olarak çalışan beyin yanında bu avantajını kaybetmektedir. Bu durumu aşmak için son yıllarda paralel tarzda çalışan nöroişlemciler üretilmekte ve yapay sinir ağları için özel donanımlar geliştirilmektedir [7].

Bundan sonraki kısımda yapay sinir ağlarının temel birimi olan yapay sinir hücresi incelenecek ve biyolojik sinir hücreleri ile olan benzerlikleri irdelenecektir.

3.3 Yapay Sinir Hücresinin Yapısı

Yapay sinir ağlarının temel birimi sinir hücreleridir. Mühendislik disiplinlerinde, yapay sinir hücrelerine, proses elemanlarıda denilmektedir. Her bir proses elemanı 5 temel unsura sahiptir. Bunlar, girişler, ağırlıklar, toplama fonksyonu, aktivasyon fonksyonu ve çıkıştır. Şekil 3.3.1'de görüldüğü gibi her bir hücre çok-girişli tek-çıkışlı yapıdadır [1, 7].



Şekil 3.3.1 : Yapay sinir hücresinin yapısı

Girişler (x_1, x_2, ..., x_n) : Giriş katmanındaki hücreler için, kullanıcı tarafından örnekler ile oluşturulmuş veri kümesidir. Diğer katmandaki hücreler için, herhangi bir katmandaki hücrenin çıkışı olabilir.

Ağırlıklar (w_1, w_2, ..., w_n) : Girişlerin, çıkışa ne oranda aktarılacağını gösterir. Örneğin w_1 ağırlığı, x_1 girişinin, çıkışa olan etkisini göstermektedir. Ağırlıkların büyük, küçük, pozitif yada negatif olması, ilgili girişin önemli yada önemsiz olduğunu göstermez. Ağırlıklar sabit yada değişken değerler olabilir.

Toplama Fonksyonu : Bir hücrenin net girdisini hesaplamak için kullanılır. Bu amaç ile değişik fonksyonlar kullanılmaktadır. En fazla tercih edilen, ağırlıklı toplam fonksyonudur. Bu fonksyonda her giriş kendi ağırlığı ile çarpılır ve bu değerler toplanır. x_i girişleri, w_i ağırlıkları ve n hücrenin giriş sayısını göstermek üzere ağırlıklı toplam fonksyonu ;

$$NET = \sum_{i=1}^{n} x_i \cdot w_i \quad \text{'dir.} \qquad \textbf{(3.3.1)}$$

Yapay sinir ağındaki bütün hücrelerin toplama fonksyonlarının aynı olması gerekmez. Her hücre bağımsız olarak farklı bir toplama fonksyonuna sahip olabilir. Kullanılan diğer toplama fonksyonları tablo 3.3.1'de gösterilmektedir.

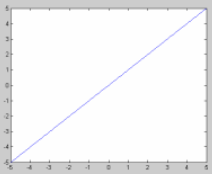
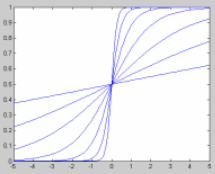
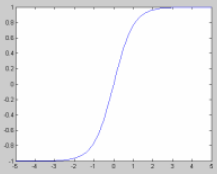
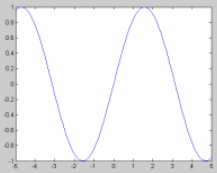
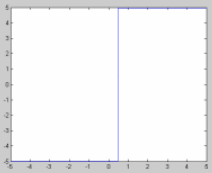
Tablo 3.3.1 : Yapay sinir hücrelerinde kullanılan toplama fonksyonları

Fonksyonun Adı	Fonksyon *(i = 1, 2, ..., n)*	Açıklama
Ağırlıklı Toplam	$NET = \sum x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile toplanır.
Çarpım	$NET = \prod x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile çarpılır.
Maksimum	$NET = Max(x_i \cdot w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en büyüğü alınır.
Minimum	$NET = Min(x_i \cdot w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en küçüğü alınır.
Artımlı Toplam	$NET_k = NET_{k-1} + \sum x_i \cdot w_i$	Ağırlıklı toplam bulunur. Bir önceki ağırlıklı toplam ile toplanır

Aktivasyon Fonksyonu : Hücrenin net girdi değerine karşılık üretilecek çıktı değerinin hesaplanmasında kullanılır. Çok katmanlı algılayıcılarda olduğu gibi, bazı sinir ağı modellerinde aktivasyon fonksyonunun türevlenebilir olması şartı vardır.

Ayrıca aktivasyon fonksyonunun türevinin hesaplanması, ağın eğitim süresi bakımından önem taşır. Sigmoid fonksyonunun türevi, fonksyonun kendisi cinsinden yazılabilmektedir. Bu durum, işlem kolaylığı sağladığından sigmoid fonksyonu, yaygın şekilde kullanılmaktadır. Toplama fonksyonunda olduğu gibi, bütün hücrelerin aynı aktivasyon fonksyonunu kullanması gerekmez. Her hücre bağımsız olarak farklı bir aktivasyon fonksyonuna sahip olabilir. Tablo 3.3.2'de çeşitli aktivasyon fonksyonları gösterilmektedir.

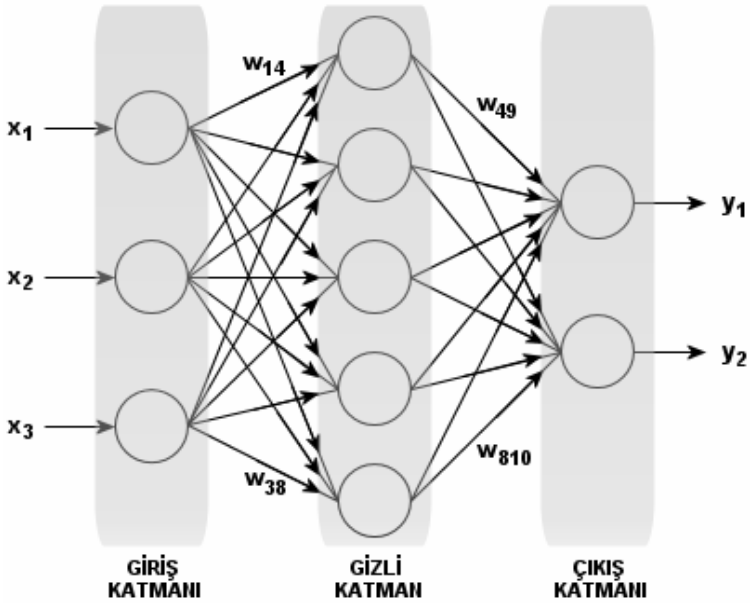
Tablo 3.3.2 : Yapay sinir hücrelerinde kullanılan aktivasyon fonksyonları

Fonksyonun Adı	Fonksyonun Şekli	Matematiksel İfadesi	Açıklama
Lineer Fonksyon		$F(NET) = NET$	Hesaplanan net giriş değeri, hücrenin çıkışı olarak kabul edilir
Sigmoid Fonksyonu		$F(NET) = \frac{1}{1+e^{-\beta.NET}}$	β değeri keyfi şekilde değiştirilerek farklı eğimlere sahip eğriler elde edilebilir
Hiperbolik Tanjant Fonksyonu		$F(NET) = \frac{e^{NET} + e^{-NET}}{e^{NET} - e^{-NET}}$	Hesaplanan net giriş değeri, tanjant fonksyonuna uygulanır
Sinüs Fonksyonu		$F(NET) = Sin(NET)$	Öğrenilmesi istenen olayın, sinüs fonksyonuna uygun dağılım gösterdiği durumlarda kullanılır
Basamak Fonksyon		$F(NET) = \begin{cases} 1 & NET > k \\ 0 & NET \le k \end{cases}$	k, eşik değeri göstermektedir. Hesaplanan değerin, eşik değerden büyük yada küçük olmasına göre 1 yada 0 çıkışları üretilir

Çıkış (y) : Aktivasyon fonksyonu tarafından belirlenen değerdir. Üretilen çıktı, başka bir hücreye yada dış dünyaya gönderilebilir. Geribesleme olması durumunda, hücre kendi üzerinde geribesleme yaparak, kendi çıktı değerini, giriş olarak kullanabilir. Bununla birlikte, geribesleme başka bir hücreyede yapılabilir. Ağ şeklinde gösterildiğinde, bir hücrenin, birden fazla çıkışı varmış gibi görünmektedir. Fakat bu durum sadece gösterim amaçlıdır. Bu çıkışların hepsi aynı değere sahiptir.

3.4 Yapay Sinir Ağının Yapısı

Yapay sinir hücreleri bir araya gelerek, yapay sinir ağını oluşturur. Yapay sinir ağı üç kısımdan meydana gelir. Bunlar giriş katmanı, gizli katman(lar) ve çıkış katmanıdır. Giriş ve çıkış katmanlarındaki hücre sayıları, uygulama tarafından belirlenir. Örneğin 3-girişli 2-çıkışlı bir sistem için kurulacak yapay sinir ağında, giriş katmanında üç hücre, çıkış katmanında ise iki hücre bulunacaktır. Gizli katmanların sayısı ve bu katmanlardaki gizli hücrelerin sayısı ise keyfi şekilde tasarlayıcı tarafından belirlenir. Gizli katman ve gizli hücre sayıları arttıkça, yapay sinir ağının arıza toleransı artacaktır, bununla birlikte işlem karmaşıklığı ve eğitim süreside artacaktır. İyi bir çözüm için gizli katman ve gizli hücrelerin sayısı önem teşkil etmektedir. Şekil 3.4.1'de 3-girişli 2-çıkışlı bir sistem için kurulan yapay sinir ağı gösterilmektedir.



Şekil 3.4.1 : Yapay sinir ağının yapısı

Bu sistem için bir tane gizli katman ve bu gizli katmanda beş tane gizli hücre kullanılmıştır. Ağırlık değerleri w_{ij} notasyonu ile belirtilir. Bu notasyon i.hücreden j.hücreye olan bağlantıyı gösterir. Yapay sinir ağlarında bilgi, bağlantıların sahip olduğu ağırlık değerlerinde saklanır. Ağırlık değerleri bütün ağa yayılmış olduğundan, ağ belleği dağıtık hafızalıdır. Olay tüm ağ tarafından karakterize edilir, bu yüzden tek bir ağırlık değeri bir anlam ifade etmez [1, 7].

Yukarda gösterilen ağ için giriş vektörü x_1, x_2 ve x_3'den oluşmaktadır. Bu girişler motora verilen gerilim değerleri, bir kontrol sistemi için birim basamak giriş, bir resmin gri tonlarını temsil eden sayısal değerler yada bir parmak izini gösteren nümerik değerler olabilir. Çıkış vektörü ise y_1 ve y_2'den oluşmaktadır. Öğretmenli öğrenme algoritmalarında y_1 ve y_2, hedef değer olarak tanımlanmaktadır.

Verilen giriş vektörüne karşılık, hedef vektörün üretilebilmesi için ağırlıkların değiştirilmesine yapay sinir ağının eğitilmesi denir. Diğer bir ifade ile eğitim, ağırlıkların, hedef vektörü üretecek şekilde güncellenmesidir.

Başlangıçta ağırlık değerleri keyfi şekilde belirlenir, çünkü ilk iterasyonda büyük bir hataya sebep olsa bile, bir değerinin olması gerekmektedir. Örnekler yapay sinir ağına tanıtıldıkça, ağırlıklar güncellenir ve hedef değeri üretecek şekilde değişir. Güncelleme işlemi öğrenme kurallarına göre gerçekleştirilir. Öğretmenli yada öğretmensiz öğrenme stratejileri için geliştirilmiş farklı öğrenme kuralları vardır.

Hedef değerler ile yapay sinir ağının ürettiği çıkışlar arasındaki hata, belirli bir değerin altına düştüğünde, ağın eğitilmesi durdurulur ve performansını ölçmek için test aşamasına geçilir. Genellikle test verileri, ağın eğitimi sırasında kullanılmayan örneklerden seçilir. Test aşamasında ağırlıklar değiştirilmez. Eğitim sırasında bulunan ağırlıklar kullanılarak, yapay sinir ağının çıkış üretmesi sağlanır. Üretilen bu çıkışların doğruluğu, ağın performansı hakkında bilgi verir. Eğer performans yeterli görülürse, yapay sinir ağının öğrendiği kabul edilir. Artık yapay sinir ağı, gösterilen örneklerin temsil ettiği olay hakkında genelleme yapabilecek yeteneği kazanmıştır [1, 7].

3.5 Farklı Modellerin Ortaya Çıkmasını Sağlayan Unsurlar

Öğrenme stratejileri (öğretmenli, destekleyici, öğretmensiz), kullanılan öğrenme kuralı (Hebb, Hopfield, Delta, Kohonen), yapay sinir hücrelerinin bağlanması sonucu oluşan topoloji (ileribeslemeli veya statik, geribeslemeli veya dinamik), hücrelerin sahip oldukları toplama ve aktivasyon fonksyonları, yapay sinir ağının modelini belirlemektedir.

Bu modellere ilişkin ayrıntılı bir sınıflandırma kısım2.7'de verilmişti. Bu modellerden en çok kullanılanları ; Tek Katmanlı Algılayıcılar, Çok Katmanlı Algılayıcılar, Vektör Kuantizasyon Modelleri, Olasılığa Dayalı Ağlar, Radyal Tabanlı Ağlar, Boltzman Makinası, Elman Ağı, Adaptif Rezonans Teorisi Modelleri'dir [1].

Çalışmanın bundan sonraki kısmında sırasıyla tek katmanlı algılayıcılara ve çok katmanlı algılayıcılara değinilicektir. 4.bölümde yer verilecek olan tek katmanlı algılayıcılar, tüm modellerin temelini oluşturmaktadır. Fakat tek katmanlı algılayıcılar ile sadece lineer problemler çözülebilmektedir. Lineer olmayan sistemleride modelleyebilmek için, 5.bölümde çok katmanlı algılayıcılara değinilicektir.

4. TEK KATMANLI ALGILAYICILAR

Bu bölümde, çok katmanlı algılayıcıların temeli olan tek katmanlı algılayıcılar incelenecektir. Kısım 4.1'de tek katmanlı algılayıcıların ortak özelliklerine değinilecektir. Yüzey ayracının tanımı yapılacak ve özel bir durum için yüzey ayracının geometrik yorumu irdelenecektir. Kısım 4.2'de Rosenblatt tarafından geliştirilen en basit tek katmanlı algılayıcı olan "perceptron", kısım 4.3'de ise diğer bir algılayıcı modeli olan "ADALINE" incelenecktir. Perceptron ve ADALINE arasındaki farklar ortaya konulacaktır. Perceptron modelinde kullanılan Hebb öğrenme kuralı ve ADALINE modelinde kullanılan Delta öğrenme kuralı tanıtılacaktır. Kısım 4.4'de bu iki tek katmanlı algılayıcı modelinin, sınıflandırma ve lineer ilişkileri belirleme performansları karşılaştırılacak ve aktivasyon fonksyonları açısından birbirlerine olan üstünlükleri ortaya konulacaktır.

4.1 Tek Katmanlı Algılayıcıların Ortak Özellikleri

Tek katmanlı algılayıcılar girişler, toplama-aktivasyon görevlerini yerine getiren bir gövde ve çıkışdan oluşur. Bu bakımdan çok-girişli tek çıkışlı yapıdadır. Şekil 4.1.1'de tek katmanlı bir yapay sinir ağı modeli gösterilmektedir.



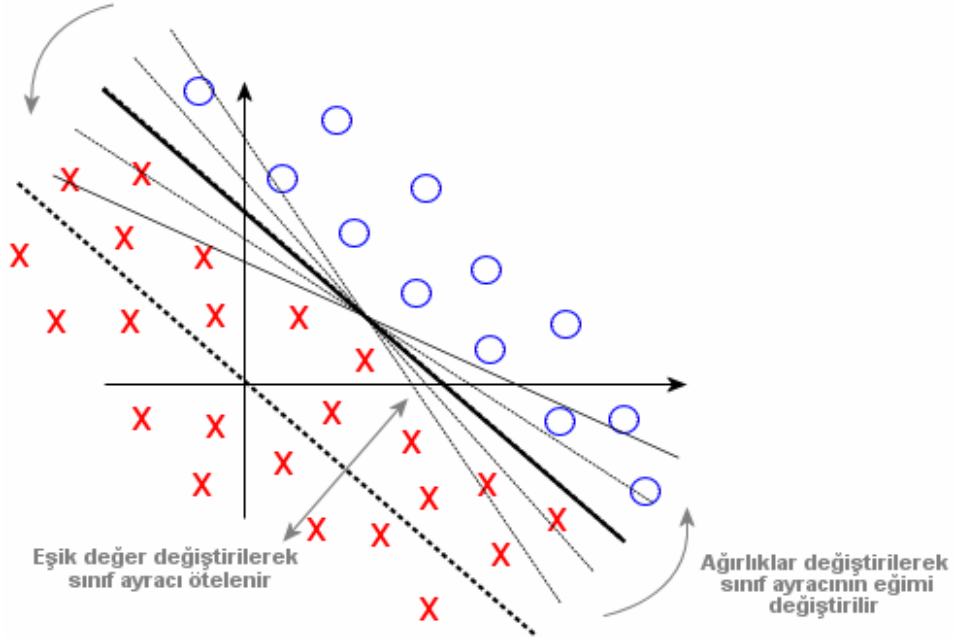
Şekil 4.1.1 : Tek katmanlı yapay sinir ağı modeli

Tek katmanlı ağlarda, girişlerin toplamının ve dolayısıyla çıkışın sıfır olmasını önleyecek bir eşik değer (ϕ) kullanılabilir. Kullanılmasını zorunlu kılan durumların

dışında, eşik değerin olması yada olmaması keyfidir. Eşik değer iterasyonlar sırasında, ağırlık değerleri gibi değiştirilebilir. Eşik değerin girişi her zaman 1'dir [1].

Tek katmanlı algılayıcılarda, giriş değerleri iki farklı sınıfa ayrılarak, kümelendirilmeye çalışılır. Dolayısıyla problem, iki sınıfı birbirinden ayıran bir yüzeyin bulunmasıdır. Bu yüzey, uzayı iki farklı bölgeye ayırır ve farklı çıkış değerlerini alan giriş parametreleri, bu yüzey tarafından kümelendirilmiş olur.

İki sınıfı birbirinden ayırmak için kullanılan yüzey, sınıf ayracı olarak tanımlanır. Bu kavramı daha kolay anlayabilmek için, tek katmanlı yapay sinir ağının 2-girişli olduğunu farzedelim. Bu durumda, sınıf ayracı iki-boyutlu öklid uzayında bir doğru belirtir. İki girişli sistem için sınıf ayracı şekil 4.1.2'de gösterilmektedir.

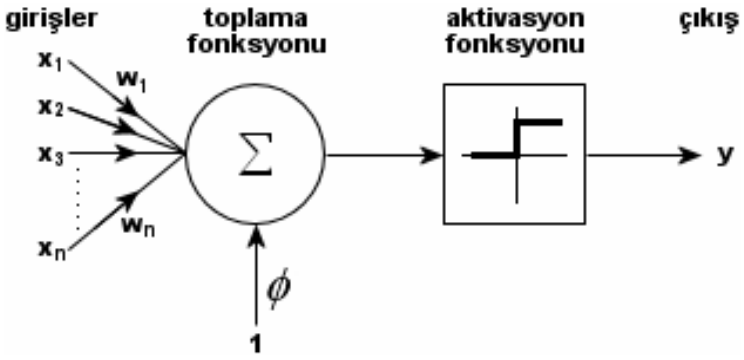


Şekil 4.1.2 : İki-girişli bir sistem için sınıf ayracı

Eşik değerin değiştirilmesi, sınıf ayracını ötelerken, ağırlıkların değiştirilmesi eğimini etkiler. Eğitim sırasında eşik değer ve ağırlıklar değiştirilerek, doğrunun, sınıfları ayıracak şekilde konumlandırılması sağlanır [3].

4.2 En Basit Tek Katmanlı Algılayıcı

Perceptron, 1958 yılında Rosenblatt tarafından şekil sınıflandırma amacı ile geliştirilmiştir. Şekil 4.2.1'de en basit tek katmalı algılayıcı modeli gösterilmektedir [1, 8].



Şekil 4.2.1 : En basit tek katmanlı algılayıcı modeli (perceptron)

Giriş ve hedef vektörleri : Giriş değerlerinin sayısı için bir kısıtlama yoktur. Giriş vektörü, negatif yada pozitif herhangi bir değeri içerebilir. Giriş değerlerinin 1'den büyük yada küçük olması performansı etkilemez. Perceptronun üretmesini istediğimiz çıkışlara hedef değer denir. Giriş vektörü için söylenenler, hedef vektör için geçerli değildir. Kullanılan aktivasyon fonksyonu sebebi ile hedef vektör, sadece 0 yada 1 değerlerini içerebilir.

Toplama fonksyonu : Ağırlıklı toplam fonksyonu kullanılır. Yüzey ayracının ötelenmesi gereken durumlarda eşik değer ilave edilebilir.

$$NET = \left(\sum_{i=1}^{n} w_i \cdot x_i \right) + \phi \qquad \textbf{(4.2.1)}$$

Aktivasyon fonksyonu : Perceptron, sınıflandırma amacı ile geliştirildiğinden, farklı sınıfları temsil edecek değerler üretilmelidir. Sınıflandırma işleminde, klasik küme kuramındaki ait olma yada ait olmama durumları incelenir. Bulanık mantıkda olduğu gibi bir aitlik derecesi vermek mümkün olmamaktadır. Bu amaçla aktivasyon fonksyonu olarak basamak fonksyon kullanılır.

NET girdinin eşik değerden büyük yada küçük olmasına göre çıkış değeri 0 yada 1 olarak üretilir.

$$y = f(NET) = \begin{cases} 1 & Eger \quad NET > \phi \\ 0 & aksi \quad takdirde \end{cases} \qquad \textbf{(4.2.2)}$$

Öğrenme kuralı (Hebb kuralı) : Eğer perceptronun ürettiği çıkış ile hedef değer aynı olursa, ağırlıklarda herhangi bir değişme olmaz. Bu durum sınıf ayracının doğru pozisyonda olduğunu gösterir. Fakat ağ, hedef değerden farklı bir çıkış üretmiş ise, ki bu durumda ağırlıklar değiştirilerek sınıf ayracının pozisyonu değiştirilmelidir, iki durum söz konusu olabilir.

i) Hedef değer 0 ve gerçek çıkış 1 ise $\Rightarrow$ $\quad w_i(k+1) = w_i(k) - \eta \cdot x_i$

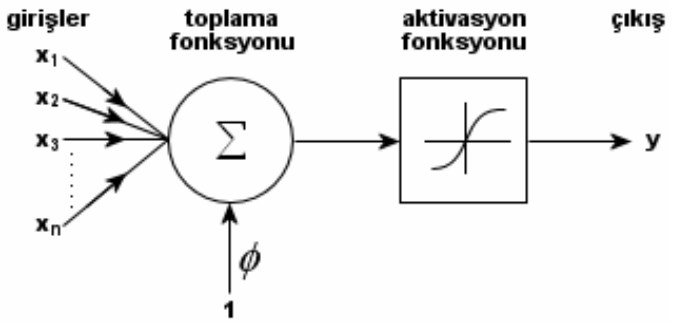
ii) Hedef değer 1 ve gerçek çıkış 0 ise $\Rightarrow$ $\quad w_i(k+1) = w_i(k) + \eta \cdot x_i$

η, öğrenme katsayısı olup $[0 \quad 1]$ aralığında herhangi bir değer alabilir. Eşik değer benzer şekilde güncellenir.

$$\phi(k+1) = \phi(k) \pm \eta \qquad \textbf{(4.2.4)}$$

4.3 ADALINE Modeli

Widrow ve Hoff tarafından 1959 yılında geliştirilmiştir. "ADAptive LINear Element" kelimelerinin kısaltılmış şeklidir. Şekil 4.3.1'de ADALINE modeli gösterilmektedir [1, 8].



Şekil 4.3.1 : Aktivasyon fonksyonu sigmoidal olan bir ADALINE modeli

ADALINE modelinde, perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılabilir. Fakat genellikle hiperbolik tanjant fonksyonu yada sigmoidal fonksyon

kullanılır. Bunun dışında öğrenme kuralı, perceptronda kullanılan Hebb kuralından farklıdır.

Giriş ve hedef vektörleri : Perceptronlarda olduğu gibi, giriş değerlerinin sayısı için bir kısıtlama yoktur. Giriş vektörü, negatif yada pozitif herhangi bir değeri içerebilir. Giriş değerlerinin 1'den büyük yada küçük olması performansı etkilemez. Hedef vektörüne getirilecek kısıtlamalar ise, aktivasyon fonksyonları ile ilişkilidir. Eğer perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılırsa, hedef değerler sadece 0 veya 1 olabilir. Fakat hiperbolik tanjant yada sigmoidal aktivasyon fonksyonları kullanılırsa, bu durumda hedef vektör, 0 ve 1 aralığındaki değerleride içerebilir. Kısım3.3 tablo 3.3.2'de aktivasyon fonksyonları ayrıntılı bir şekilde gösterilmişti.

Toplama fonksyonu : Ağırlıklı toplam fonksyonu kullanılır. Yüzey ayracının ötelenmesi gereken durumlarda eşik değer ilave edilebilir.

$$NET = \left(\sum_{i=1}^{n} w_i \cdot x_i \right) + \phi \qquad \textbf{(4.3.1)}$$

Aktivasyon fonksyonu : ADALINE modelinde, genellikle hiperbolik tanjant fonksyonu yada sigmoidal fonksyon kullanılır. Sigmoidal fonksyonda β değerini değiştirerek, farklı eğimlerde fonksyonlar elde etmek mümkündür.

$$y = f(NET) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}} \quad \text{yada} \quad y = f(NET) = \frac{1}{1 + e^{-\beta . NET}} \qquad \textbf{(4.3.2)}$$

Öğrenme kuralı (Delta kuralı) : y^* hedef değeri ve y gerçek çıkışı göstermek üzere, delta öğrenme kuralını elde etmek amacı ile hatayı tanımlayalım.

$$e = y^* - y \qquad \textbf{(4.3.3)}$$

Hatanın negatif değerlerinden kurtulmak için karesel hatayı tanımlayalım.

$$kareselhata = \left(y^* - y \right)^2 \qquad \textbf{(4.3.4)}$$

Hata yüzeyinin minimum noktasını bulabilmek için gradyanın tersi yönde ilerleyelim.

$$\frac{\partial (kareselhata)}{\partial w_i} = -2 \left(y^* - \sum_i w_i x_i \right) \cdot x_i \Rightarrow \eta \cdot e \cdot x_i \qquad \textbf{(4.3.5)}$$

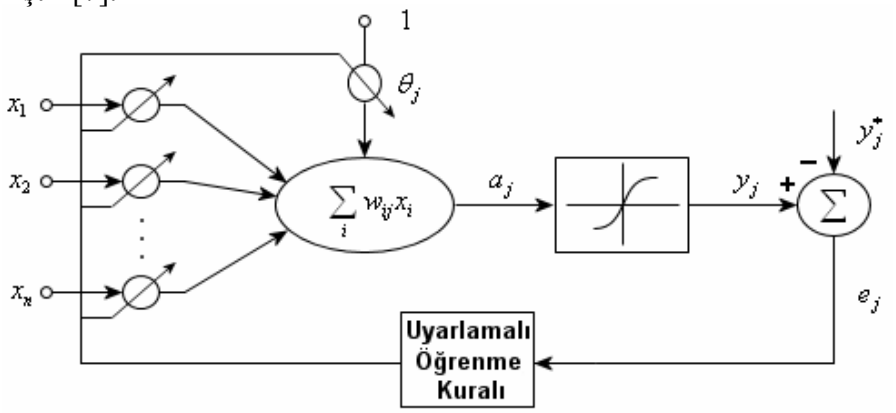
Yukardaki ifadede katsayılar, öğrenme katsayısına, η, dahil edilmiştir. Böylelikle ağırlıkları güncellemek için kullanılacak delta öğrenme kuralı bulunmuş olur.

$$\Delta w_i = \eta \cdot e \cdot x_i \quad \Rightarrow \quad w_i(k+1) = w_i(k) + \eta \cdot e \cdot x_i \qquad \textbf{(4.3.6)}$$

Benzer şekilde eşik değer, aşağıdaki ifade yardımıyla değiştirilir.

$$\phi(k+1) = \phi(k) + \eta \cdot e \qquad \textbf{(4.3.7)}$$

ADALINE modelinin yapısı şekil 4.3.1'de verilmektedir. Fakat ağırlıkların, çıkış ve hedef vektörler arasında tanımlanan hatadan yararlanarak güncellenmesi durumunu görebilmek için, ADALINE modeli şekil 4.3.2'de ayrıntılı bir biçimde tekrar gösterilmiştir [7].



Şekil 4.3.2 : ADALINE modelinin ayrıntılı yapısı [7]

4.4 Perceptron ve ADALINE Modellerinin Karşılaştırılması

Bu kısımda perceptron ve ADALINE modellerinin karşılaştırılması yapılacak ve birbirlerine olan üstünlükleri incelenecektir. Bu amaç ile basamak aktivasyon fonksyonu kullanan perceptron ve sigmoidal aktivasyon fonksyonu kullanan ADALINE modelleri için Matlab'de m-dosyaları hazırlanmıştır. Bu dosyalar Ek.A'da verilmektedir. Öncelikle bu iki model sınıflandırma konusunda karşılaştırılacaktır. Lojik fonksyonların gerçeklenmesi, sınıflandırma problemi olarak düşünülebilir. Örneğin lojik-VE fonksyonunda $(0,0)$, $(0,1)$ ve $(1,0)$ girişleri 0-kümesine dahil edilirken, $(1,1)$ girişi 1-kümesine dahil edilmektedir. Perceptron ve ADALINE modellerinin performansını, lojik-VE fonksyonunu gerçekleyerek inceleyelim.

Perceptronun Sınıflandırma Performansı : Yazılan m-dosyasını çağıran komut, perceptronu geliştiren kişinin ismine ithafen rosenblatt olarak kullanılmıştır. Giriş

vektörü, “giris” ve yapay sinir hücresinin üretmesini istediğimiz çıkış vektörü ise “hedef” olarak tanımlanacaktır. “defa” ve “hata” parametreleri sırasıyla iterasyonun kaç kere tekrarlanacağını ve kabul edilebilecek en büyük hatayı göstermektedir. Bu kriterlerden birisinin sağlanması iterasyonu durduracaktır. Bu parametreler belirtilmezse, sırasıyla 1000 ve 10^{-12} olarak kabul edilecektir.

```
y = rosenblatt(giris,hedef,defa,hata)
    giris = [0 0; 0 1; 1 0; 1 1];
    hedef = [0 ; 0; 0; 1];

    >> rosenblatt(giris,hedef)

    Yapilan iterasyon sayisi 12 dir

    Nöronun egitilmesi sonucunda bulunan çikis vektörü

    0
    0
    0
    1

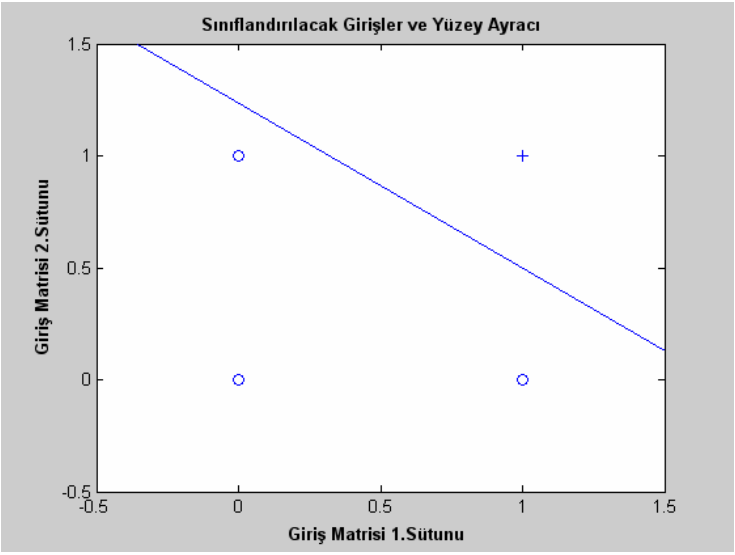
    Nöronun egitilmesi sonucunda bulunan agirlik vektörü

    2.535670381
    3.434425738

    Böylelikle yapilan karesel hata 0.0000000000e+000 dir

    Bu islem toplam 0.521 saniye sürmüstür
```

Hazırlanan kod tarafından çizdirilen yüzey ayracı şekil 4.4.1’de gösterilmektedir.



Şekil 4.4.1 : Çıkışların sınıflandırılması için kullanılan yüzey ayracı (Lojik-VE fonksyonu)

ADALINE'in Sınıflandırma Performansı : Benzer işlemler ADALINE için de yapılabilir. Sadece adaline komutunda, perceptrondan farklı olarak öğrenme katsayısını gösteren "eta" parametresi vardır. Aynı giriş ve hedef değerleri için ADALINE'den alınan sonuçlar aşağıdaki gibidir.

```
y = adaline(giris,hedef,eta,defa,hata)
    giris = [0 0; 0 1; 1 0; 1 1];
    hedef = [0; 0; 0; 1];

    >> adaline(giris,hedef,0.1,100000)

    Yapilan iterasyon sayisi 100000 dir

    Nöronun egitilmesi sonucunda bulunan çikis vektörü

    0.0001000419811
    0.0001000614995
    0.0001000614996
    0.9998999385

    Nöronun egitilmesi sonucunda bulunan agirlik vektörü

    -0.5479858383
    -0.5398764858

    Böylelikle yapilan karesel hata 4.0045309050e-008 dir

    Bu islem toplam 21.4 saniye sürmüstür
```

Görüldüğü gibi, perceptron istenilen hedef vektörü 12 iterasyon yaparak tam olarak bulmuş ve hatayı sıfırlayabilmiştir. Fakat ADALINE modeli, daha fazla iterasyona rağmen, sonuca ancak 10^{-8} mertebesinde bir hata ile yaklaşabilmişdir. Bunun sebebi istenilen hedef değerlerinin 0 ve 1 (binary) olmasıdır. Hatırlanacağı gibi perceptron basamak aktivasyon fonksyonu kullanırken (sadece 0 ve 1 çıkışını üretebilmektedir), ADALINE sigmoidal aktivasyon fonksyonu kullanmaktadır. Bunun bir sonucu olarak perceptron sınıflandırma problemlerinde daha üstün bir performans göstermektedir.

Perceptronun Lineer İlişkileri Belirleme Performansı : Giriş ve hedef vektörlerini uygun şekilde seçerek, aynı m-dosyalarını, lineer ilişkileri belirlemek için kullanabiliriz. Giriş matrisi tablo 4.4.1'de gösterildiği gibi seçilsin. Hedef vektör ise, birinci, üçüncü, beşinci elemanların toplanması, ikinci ve dördüncü elemanların çıkarılması ile oluşturulsun.

Tablo 4.4.1 Giriş matrisi ve hedef vektörü

Giriş Matrisi					Hedef Vektörü
x_1	x_2	x_3	x_4	x_5	$x_1 - x_2 + x_3 - x_4 + x_5$
0.2	0.7	0.9	0.1	0.5	0.8
0.3	0.5	1.0	0.9	0.3	0.2
0.4	0.6	0.8	0.3	0.7	1.0
1.0	0.9	0.4	0.5	0.1	0.1

Görüldüğü gibi, perceptron $hedef = x_1 - x_2 + x_3 - x_4 + x_5$ şeklinde tanımlanan fonksyonu ortaya çıkaramamıştır. Yapılan hata çok büyüktür ve iterasyon sayısının artırılması durumu fazla değiştirmemektedir.

```
>> rosenblatt(giris,hedef,100000)

Yapilan iterasyon sayisi 100000 dir

Nöronun egitilmesi sonucunda bulunan çikis vektörü

0
0
1
0

Nöronun egitilmesi sonucunda bulunan agirlik vektörü

2.729088764
-0.8487043736
-0.0467753825
1.964989341
9.207235741

Böylelikle yapilan karesel hata 6.9000000000e-001 dir

Bu islem toplam 26.6 saniye sürmüstür
```

ADALINE'in Lineer İlişkileri Belirleme Performansı : Benzer işlemler ADALINE için yapılabilir. Aynı giriş ve hedef değerleri için ADALINE modelinden alınan sonuçlar aşağıdaki gibidir. Şekil 4.4.2'de karesel hatanın değişimi gösterilmektedir.

```
>> adaline(giris,hedef)

Yapilan iterasyon sayisi 10000 dir

Nöronun egitilmesi sonucunda bulunan çikis vektörü

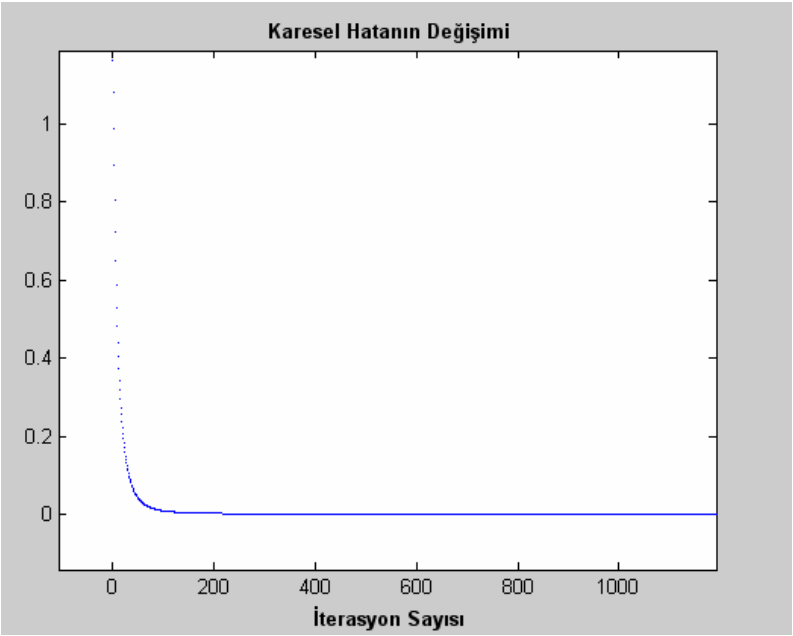
0.8002624194
0.2001787478
0.9992995502
0.1001263783

Nöronun egitilmesi sonucunda bulunan agirlik vektörü

0.9000673777
-0.2487610977
-0.984165417
-0.8868143373
-2.790121933

Böylelikle yapilan karesel hata 6.0741615217e-007 dir

Bu islem toplam 2.05 saniye sürmüstür
```



Şekil 4.4.2 : İterasyon sayısına göre karesel hatanın değişimi

ADALINE modeli tarafından yapılan karesel hata 10^{-7} mertebesindedir. Sınıflandırma probleminin tam tersine, $hedef = x_1 - x_2 + x_3 - x_4 + x_5$ şeklinde tanımlanan fonksyonun saptanmasında ve bu lineer ilişkinin ortaya çıkartılmasında, ADALINE modelinin daha üstün bir performans gösterdiği açıktır. Bunun sebebi $[0 \quad 1]$ aralığında değer alabilen aktivasyon fonksyonudur.

Sonuç olarak sınıflandırma probleminde, basamak aktivasyon fonksyonu kullanan perceptron daha iyi bir performans gösterirken, lineer ilişkileri belirleme probleminde, sigmoidal aktivasyon fonksyonu kullanan ADALINE ile daha üstün bir performans elde edilmiştir. Yapılan bu basit karşılaştırma sayesinde, uygulamanın niteliğine göre, doğru aktivasyon fonksyonu ile çalışmanın önemi anlaşılmaktadır.

5. ÇOK KATMANLI YAPAY SİNİR AĞLARI

Bu bölümde hatanın geriye yayılımı "backpropogation" yöntemini kullanan ileribeslemeli çok katmanlı yapay sinir ağı modeli incelenecektir. Kısım 5.1'de "neden tek katmanlı algılayıcılar yeterli değildir?" sorusuna cevap aranacaktır. Kısım 5.2'de yapay sinir ağının yapısı incelenecektir. Kısım 5.3'de hatanın geriye yayılımı yönteminde kullanılan genelleştirilmiş delta öğrenme kuralı açıklanacaktır. Son olarak kısım 5.4'de ise ağların tasarımında karşılaşılan problemler ve dikkat edilmesi gereken noktalar belirtilecektir.

5.1 Çok Katmanlı Yapay Sinir Ağlarına Olan Gereksinim

Tek katmanlı algılayıcılar basit ve hızlı olmakla birlikte, sadece karmaşık olmayan lineer problemleri çözebildiklerinden, mühendislik uygulamalarında kullanılabilecek yeterlilikte değillerdir. Bu durum eğitilebilecek tek bir ağırlık matrisinin olmasından kaynaklanmaktadır. Bu yüzden karmaşık ve lineer olmayan problemlerin çözülmesinde çok katmanlı yapay sinir ağlarına ihtiyaç duyulmaktadır. Bu gereksinim, tek katmanlı algılayıcılar bölümünde lojik-Ve probleminin çözülmesinde daha iyi bir performans gösteren "rosenblatt" m-dosyası üzerinde gösterilecektir. Hatırlanacağı gibi, "rosenblatt", 12 iterasyonda, sıfır hata ile istenilen lojik-Ve çıkışını üretebilmişti. Lojik-ÖzelVeya problemi için alınan sonuçlar aşağıda gösterilmektedir.

```
>> giris = [0 0; 0 1; 1 0; 1 1];
>> hedef = [0; 1; 1; 0];
>> rosenblatt(giris,hedef)

Yapilan iterasyon sayisi 1000 dir

Nöronun egitilmesi sonucunda bulunan çikis vektörü

1
1
1
1

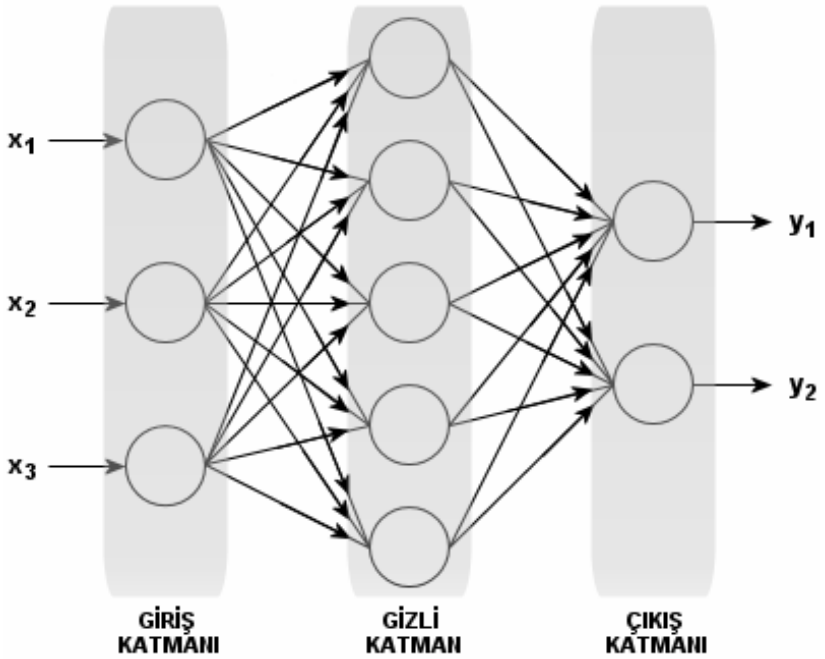
Böylelikle yapilan karesel hata 2.0000000000e+000 dir
```

Görüldüğü gibi m-dosyasinin ürettiği çıkış, istenilen lojik-ÖzelVeya çıkışı ile uyuşmamaktadır. Lineer olmayan bu problemin çözülebilmesi için daha gelişmiş modellere ihtiyaç duyulmaktadır.

5.2 Çok Katmanlı Yapay Sinir Ağlarının Yapısı

Bu modelde, giriş ve çıkış katmanlarının dışında, sayısı tasarımcı tarafından belirlenen ara katman veya katmanlar bulunur. Bazı kaynaklarda ara katman yerine gizli katmanda denilmektedir.

Giriş ve çıkış katmanlarındaki hücre sayıları, uygulamanın niteliğine bağlıdır. Örneğin 3-girişli 2-çıkışlı bir sistem için tasarlanacak yapay sinir ağında, giriş katmanında 3 ve çıkış katmanında 2 tane hücre bulunacaktır. Gizli katman sayısı ve gizli katmanlarda bulunacak gizli hücre sayılarının belirlenmesinde ise bir kısıtlama yoktur. Fakat gizli katman ve gizli hücre sayılarının uygun şekilde seçilmemesi, yapay sinir ağının performansını olumsuz şekilde etkileyecektir. Bu sayıların belirlenmesi bir optimizasyon problemidir. Şekil 5.2.1'de 3-girişli 2-çıkışlı bir sistem için tasarlanan çok katmanlı yapay sinir ağı gösterilmektedir. Bu tasarımda, tek bir gizli katman ve bu gizli katmanda 5 tane gizli hücre kullanılmıştır.



Şekil 5.2.1 : İleribeslemeli çok katmanlı yapay sinir ağı

Şekil 5.2.1'deki gibi ileribeslemeli ağ yapılarında, her bir hücre, sonraki katmanda bulunan tüm hücrelere bağlıdır. Bu yapısından ötürü tam-olarak-bağlanmış "fully-connected" terimi kullanılır. Fakat bu durum hücrelerin birden fazla çıkışı olduğu anlamına gelmez. Her hücrenin yalnız ve yalnız bir çıkışı vardır. Diğer bir ifade ile hücrelerin ürettikleri çıkış değeri tekdir. Fakat bu çıkış değeri, hücrelere aktarılırken ilgili ağırlıklar ile çarpılır ve böylelikle sonraki hücrelere kuvvetlendirilerek yada zayıflatılarak iletilir.

5.3 Çok Katmanlı Yapay Sinir Ağlarının Öğrenme Kuralı

Hatanın geriye yayılımı "backpropogation" yöntemini kullanan çok katmanlı yapay sinir ağlarında öğretmenli öğrenme stratejisi kullanılır. Diğer bir ifade ile ağa tanıtılan örnek kümesi hem giriş işaretlerini hemde hedef değerleri içerir.

Geriye yayılım yöntemini kullanan yapay sinir ağlarının öğrenme kuralı, en küçük kareler yöntemine dayanan delta öğrenme kuralının genelleştirilmiş halidir. Genelleştirilmiş delta kuralı iki aşamadan oluşur. İleriye doğru hesaplama aşamasında başlangıç değerleri ile yapay sinir ağının çıkışı bulunur. Hedef değerler kullanılarak yapılan hata hesaplanır. Geriye doğru hesaplama aşamasında ise ilk aşamada hesaplanan hata geriye doğru yayılarak ağırlıkların nasıl güncelleneceği bulunur.

Yapılacak işlemler, kolaylık açısından tek ara katmanı bulunan bir yapay sinir ağı üzerinde anlatılacaktır. Daha fazla katmanı bulunan ağlar için formül sayısı artmakla birlikte yapılacak işlemler aynı olacaktır.

İleriye Doğru Hesaplama : Bu aşamada amaç, verilen girişler için yapay sinir ağı çıkışının bulunması ve hedef değerler kullanılarak yapılan hatanın hesaplanmasıdır. Bu amaç ile örnekler giriş katmanındaki giriş hücrelerine tanıtılır. Bu bilgiler giriş hücrelerinden ara katman hücrelerine aktarılırken ilgili ağırlıklar ile çarpılır ve ağırlıklı toplam fonksyonuna göre her ara katman hücresinin NET girdisi hesaplanır.

$$NET_{ap} = \sum_{i} w_{gi,ap} x_i \qquad \textbf{(5.3.1)}$$

Bu formülde kullanılan notasyon aşağıda açıklanmaktadır.

NET_{ap} : ara katmanın p.hücresine olan girdi

$w_{gi,ap}$: giriş katmanının i.hücresini, ara katmanın p.hücresine bağlayan ağırlık

x_i : giriş katmanındaki i.hücresinin çıkışı

Ağırlıklı toplam dışında kullanılan diğer toplama fonksyonları tablo 5.3.1'de gösterilmektedir.

Tablo 5.3.1 : Yapay sinir ağlarında kullanılan toplama fonksyonları

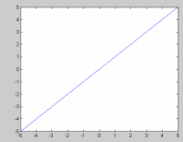
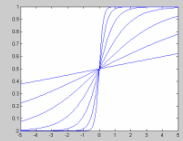
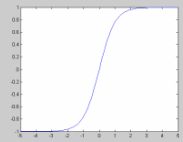
Fonksyonun Adı	Fonksyon *(i = 1, 2, ..., n)*	Açıklama
Ağırlıklı Toplam	$NET = \sum x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile toplanır.
Çarpım	$NET = \prod x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile çarpılır.
Maksimum	$NET = Max(x_i \cdot w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en büyüğü alınır.
Minimum	$NET = Min(x_i \cdot w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en küçüğü alınır.
Artımlı Toplam	$NET_k = NET_{k-1} + \sum x_i \cdot w_i$	Ağırlıklı toplam bulunur. Bir önceki ağırlıklı toplam ile toplanır

Bulunan NET girdi, bir aktivasyon fonksyonundan geçirilerek, ara katman hücrelerinin çıkışı bulunur. NET_{ap}, ara katmanın p.hücresine olan girdi, F_{ap} ise ara katmanın p.hücresinin aktivasyon fonksyonundan geçirildikten sonraki değerini göstermektedir.

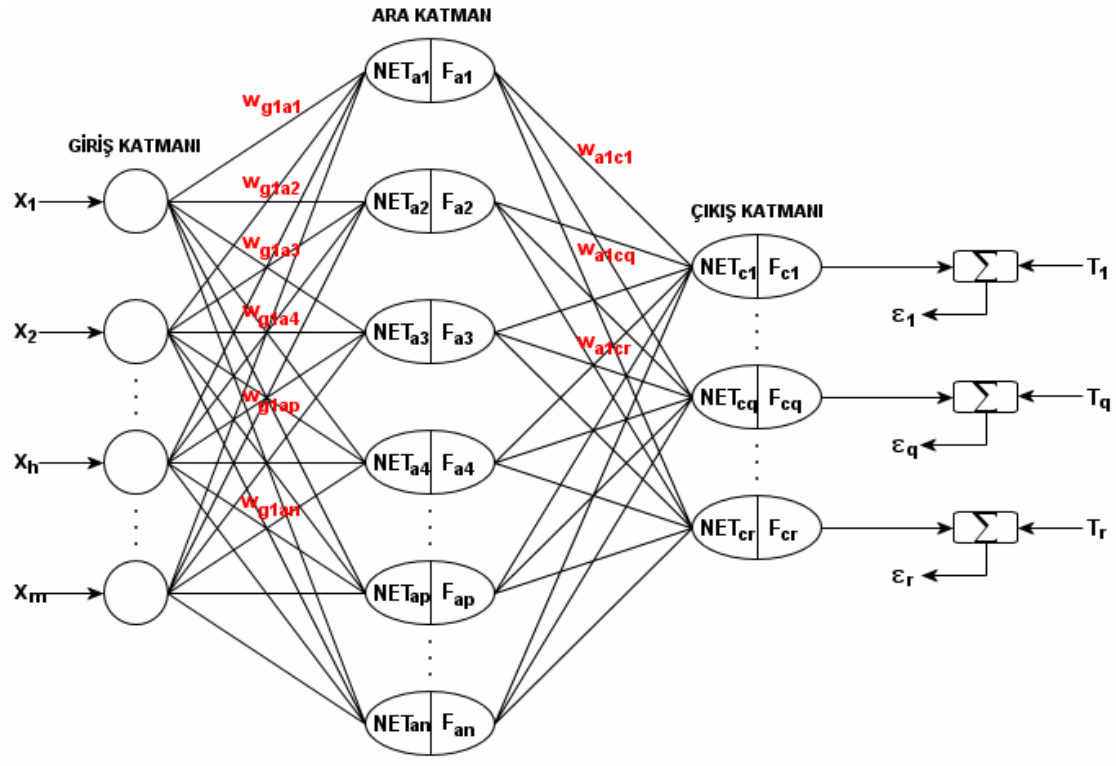
$$F_{ap} = f(NET_{ap}) \qquad \textbf{(5.3.2)}$$

Kullanılacak aktivasyon fonksyonunun türevlenebilir olma şartı vardır. Çok katmanlı yapay sinir ağlarında yaygın bir şekilde kullanılan aktivasyon fonksyonları tablo 5.3.2'de gösterilmektedir.

Tablo 5.3.2 : Yapay sinir ağlarında kullanılan aktivasyon fonksyonları

Fonksyonun Adı	Fonksyonun Şekli	Matematiksel İfadesi	Türevi	Açıklama
Lineer Fonksyon		$f(NET) = NET$	$F' = 1$	Hesaplanan net giriş değeri, hücrenin çıkışı olarak kabul edilir
Sigmoid Fonksyonu		$f(NET) = \frac{1}{1+e^{-\beta.NET}}$	$F' = \beta.F.(1-F)$	β değeri keyfi şekilde değiştirilerek farklı eğimlere sahip eğriler elde edilebilir
Hiperbolik Tanjant Fonksyonu		$f(NET) = \frac{e^{NET} + e^{-NET}}{e^{NET} - e^{-NET}}$	$F' = (1-F^2)$	Hesaplanan net giriş değeri, tanjant fonksyonuna uygulanır

Şekil 5.3.1'de hesaplanan ifadeler ve ilgili notasyon, ayrıntılı bir şekilde gösterilmektedir.



Şekil 5.3.1 : İleribeslemeli çok katmanlı yapay sinir ağının ayrıntılı yapısı

Ara katman ve çıkış hücreleri arasında da benzer işlemler yapılabilir. Çıkış katmanındaki hücrelerin NET girdisi aşağıdaki şekilde hesaplanabilir.

$$NET_{cq} = \sum_{p} w_{ap,cq} F_{ap} \qquad \textbf{(5.3.3)}$$

Bu formülde kullanılan notasyon aşağıda açıklanmaktadır.

NET_{cq} : çıkış katmanının q.hücresine olan girdi

$w_{ap,cq}$: ara katmanın p.hücresini,çıkış katmanının q.hücresine bağlayan ağırlık

F_{ap} : ara katmandaki p.hücrenin çıkış değeri

Bulunan NET girdi, bir aktivasyon fonksyonundan geçirilerek, çıkış katmanındaki hücrelerin çıkışı, dolayısıyla yapay sinir ağının çıkışı bulunur. NET_{cq}, çıkış katmanındaki q.hücreye olan girdi, F_{cq} ise çıkış katmanının q.hücresinin aktivasyon fonksyonundan geçirildikten sonraki değerini göstermektedir.

$$F_{cq} = f\left(NET_{cq}\right) \qquad \textbf{(5.3.4)}$$

Bundan sonra yapılacak işlem, çıkış hücresinin bulunan değerine karşı gelen hedef değeri kullanarak, yapılan hatanın hesaplanmasıdır. Eğer çıkış katmanındaki q.hücrenin hesaplanan değeri F_{cq} ve olması istenen çıkış değeri T_q ise, bu durumda yapılan hata ε_q aşağıdaki gibi hesaplanabilir.

$$\varepsilon_q = T_q - F_{cq} \qquad \textbf{(5.3.5)}$$

Görüldüğü gibi hata ilk kez hesaplanırken, ağırlık değerlerine ihtiyaç duyulmaktadır. Bu yüzden ağırlıkların başlangıç değerlerinin saptanması gerekmektedir. Ağırlıkların başlangıç değerlerinin belirlenmesi, özen gösterilmesi gereken bir işlemdir. İlgili literatürde bu değerlerin, gaussian dağılım gösteren, büyük değerler almayan rastgele sayılar olması gerektiği söylenmektedir. Genellikle $[-0.1 \;\; 0.1]$ aralığı kullanılmaktadır.

Geriye Doğru Hesaplama : Bu aşamada ağırlıkların, yapılan hatayı azaltacak şekilde değiştirilmesi sağlanır. Bu amaç ile ilk aşamada hesaplanan hata değerleri geriye doğru yayılır ve ağırlıklara ait güncelleme kuralları bulunur. Yönteme ismini veren "backpropogation" işlemi bu şekilde gerçekleştirilmiş olur. Bu aşamada hata, çıkış katmanından giriş katmanına doğru yayılacağından, öncelikle çıkış katmanı ve ara katman arasındaki ağırlık değerleri daha sonra ara katman ve giriş katmanı arasındaki ağırlık değerlerine ait kurallar bulunur.

Ara katman ve çıkış katmanı arasındaki ağırlıkların değiştirilmesi : Çıkış katmanındaki q.hücre için yapılan hata ;

$$\varepsilon_q = T_q - F_{cq} \tag{5.3.7}$$

Hatanın negatif değerlerinden kurtulmak için karesel hata tanımlansın ;

$$\varepsilon_q^2 = \left(T_q - F_{cq}\right)^2 \tag{5.3.8}$$

Delta kuralına göre, ağırlıklardaki değişim, karesel hatanın bu ağırlığa göre değişimiyle orantılıdır. Diğer bir ifade ile hatayı minimum yapabilmek için, ağırlıklar hatanın negatif gradyanı yönünde değiştirilmelidir. η terimi öğrenme katsayısıdır. Ara katmandaki p.hücreyi, çıkış katmanındaki q.hücreye bağlayan ağırlık değeri $w_{ap,cq}$ ile gösterilsin. Giriş katmanında m adet, ara katmanda n adet ve çıkış katmanında r adet hücre olduğu hatırlanarak, ara katman çıkış katmanı arasındaki ağırlıkların değişimini yazalım ;

$$\Delta w_{ap,cq} = -\eta_{ap,cq} \frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} \tag{5.3.9}$$

Gradyanı hesaplayabilmek için zincir kuralı işletilmelidir ;

$$\frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} = \frac{\partial \varepsilon_q^2}{\partial F_{cq}} \cdot \frac{\partial F_{cq}}{\partial NET_{cq}} \cdot \frac{\partial NET_{cq}}{\partial w_{ap,cq}} \tag{5.3.10}$$

$$\frac{\partial \varepsilon_q^2}{\partial F_{cq}} = -2\left(T_q - F_{cq}\right) \tag{5.3.10.a}$$

Eşitlik 5.3.10'daki ikinci terim kullanılan aktivasyon fonksyonuna göre değişir. Aktivasyon fonksyonunun tanjant hiperbolik veya sigmoidal olması durumlarında ;

$$\tanh \Rightarrow \frac{\partial F_{cq}}{\partial NET_{cq}} = \left(1 - F_{cq}^2\right) \quad \text{sigmoidal} \Rightarrow \frac{\partial F_{cq}}{\partial NET_{cq}} = F_{cq}\left(1 - F_{cq}\right)$$

Eşitlik 5.3.10'daki son terim ise ;

$$\frac{\partial NET_{cq}}{\partial w_{ap,cq}} = F_{ap} \tag{5.3.10.c}$$

Sigmoidal aktivasyon fonksyonu kullanıldığında, eşitlik 5.3.10 yeniden yazılırsa ;

$$\frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} = -2\left(T_q - F_{cq}\right) \cdot F_{cq}\left(1 - F_{cq}\right) \cdot F_{ap} \tag{5.3.11}$$

Böylelikle ara katman ve çıkış katmanı arasındaki ağırlıkların nasıl değiştirilmesi gerektiği bulunmuş olur ;

$$\Delta w_{ap,cq} = -\eta_{ap,cq} \frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} = -\eta_{ap,cq} \cdot \left(T_q - F_{cq}\right) \cdot F_{cq}\left(1 - F_{cq}\right) \cdot F_{ap} \qquad \textbf{(5.3.12.a)}$$

$$w_{ap,cq}\left(N+1\right) = w_{ap,cq}\left(N\right) + \Delta w_{ap,cq} \qquad \textbf{(5.3.12.b)}$$

Giriş katmanı ve ara katman arasındaki ağırlıkların değiştirilmesi : Ara katman-çıkış katmanı arasındaki ağırlıklardan farklı olarak, giriş katmanı-ara katman arasındaki ağırlıkları hesaplarken, herbir hücrenin hatası değil, toplam hata gözönünde bulundurulmalıdır.

$$\varepsilon^2 = \sum_{q=1}^{r} \varepsilon_q^2 \qquad \textbf{(5.3.13)}$$

Giriş katmanındaki h.hücreyi, ara katmandaki p.hücreye bağlayan ağırlık değeri $\Delta w_{gh,ap}$ ile gösterilsin. Çıkış katmanında r adet hücre olduğu hatırlanarak, giriş katmanı ve ara katman arasındaki ağırlıkların değişimini yazalım ;

$$\Delta w_{gh,ap} = -\eta_{gh,ap} \frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = -\eta_{gh,ap} \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial w_{gh,ap}} \qquad \textbf{(5.3.14)}$$

Gradyanı hesaplayabilmek için zincir kuralı işletilmelidir ;

$$\frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial F_{cq}} \cdot \frac{\partial F_{cq}}{\partial NET_{cq}} \cdot \frac{\partial NET_{cq}}{\partial F_{ap}} \cdot \frac{\partial F_{ap}}{\partial NET_{ap}} \cdot \frac{\partial NET_{ap}}{\partial w_{gh,ap}} \qquad \textbf{(5.3.15)}$$

$$\frac{\partial \varepsilon^2}{\partial F_{cq}} = -2\left(T_q - F_{cq}\right) \qquad \textbf{(5.3.15.a)}$$

$$\frac{\partial F_{cq}}{\partial NET_{cq}} = F_{cq}\left(1 - F_{cq}\right) \qquad \textbf{(5.3.15.b)}$$

$$\frac{\partial NET_{cq}}{\partial F_{ap}} = w_{ap,cq} \qquad \textbf{(5.3.15.c)}$$

$$\frac{\partial F_{ap}}{\partial NET_{ap}} = F_{ap}\left(1 - F_{ap}\right) \qquad \textbf{(5.3.15.d)}$$

$$\frac{\partial NET_{ap}}{\partial w_{gh,ap}} = x_h \qquad \textbf{(5.3.15.e)}$$

Hesaplanan terimler ile eşitlik 5.3.15 yeniden yazılırsa ;

$$\frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = \sum_{q=1}^{r} -2\left(T_q - F_{cq}\right)\cdot F_{cq}\left(1-F_{cq}\right)\cdot w_{ap,cq}\cdot F_{ap}\left(1-F_{ap}\right)\cdot x_h \qquad \textbf{(5.3.16)}$$

Böylelikle giriş katmanı-ara katman arasındaki ağırlıkların güncellenme kuralı ;

$$\Delta w_{gh,ap} = -\eta_{gh,ap}\sum_{q=1}^{r} 2\left(T_q - F_{cq}\right)\cdot F_{cq}\left(1-F_{cq}\right)\cdot w_{ap,cq}\cdot F_{ap}\left(1-F_{ap}\right)\cdot x_h \qquad \textbf{(5.3.17.a)}$$

$$w_{gh,ap}\left(N+1\right) = w_{gh,ap}\left(N\right) + \Delta w_{gh,ap} \qquad \textbf{(5.3.17.b)}$$

olarak bulunur.

5.4 Tasarımda Karşılaşılabilecek Problemler

Yapay sinir ağının ileriye doğru hesaplama kısmında, ilk iterasyonun yapılabilmesi için ağırlıklara değer atamak gerekmektedir. Ağırlıklara atanacak başlangıç değerleri ağın performansını yakından ilgilendirmektedir. İlgili literatürde bu değerlerin, gaussian dağılım gösteren, büyük değerler almayan rastgele sayılar olması gerektiği söylenmektedir. Genellikle $\left[-0.1 \quad 0.1\right]$ aralığı kullanılmaktadır [1].

Öğrenme katsayısının belirlenmesi, istenilen hata değerine ulaşılması için yapılması gerekli iterasyon sayısını etkilemektedir. Öğrenme katsayısı $\left[0 \quad 1\right]$ aralığında seçilebilir. Bu değer sıfıra yakın seçildiğinde istenilen hata değerine ulaşmak için daha fazla iterasyon yapılması gerekir. Diğer bir ifade ile yakınsama yavaş olur. Öğrenme katsayısının değeri arttıkça yapılması gerekli iterasyon sayısı azalır. Fakat büyük değerler yapay sinir ağının öğrenmesi yerine ezberlemesine neden olmaktadır. "Yapay sinir ağı öğreniyor mu yoksa ezberliyor mu?" sorusunun cevabını bir örnek ile açıklamak yararlı olacaktır. Eğitim verileri için yapılacak 10^{-4} mertebesindeki bir hatanın uygulama için yeterli olacağını düşünelim. Bu durumda test verileri için yapılacak $\left[10^{-3} \quad 10^{-5}\right]$ aralığındaki hatalar seçilen öğrenme katsayısının uygun olduğunu gösterir. Fakat test verileri için 10^{-1} mertebesinde bir hata yapılıyorsa bu durumda yapay sinir ağının öğrenmediğini, bunun yerine ezberlediğini söyleyebiliriz. Bu durumda öğrenme katsayısının küçültülmesi gerekir. Görüldüğü gibi öğrenme katsayısı küçük seçildiğinde yakınsama hızı düşmekte, büyük seçildiğinde ise öğrenme problemi doğmaktadır. Bu sorunu aşmak için öğrenme katsayısı adaptif olarak değiştirilebilir. Çözüm noktasından uzakta öğrenme katsayısı büyük seçilerek

yakınsama hızlandırılabilir. Çözüm noktası civarında ise katsayı azaltılarak öğrenme problemi aşılabilir [7].

Hatanın değişiminde büyük salınımlar görülüyorsa, öğrenme katsayısının değeri küçültülmelidir [8].

Momentum katsayısı, yerel minimum noktalarında takılmayı önlemektedir. Ağırlıklardaki değişimin bir önceki değerleri momentum katsayısı ile çarpılarak yerel minimumlarda sıçrama sağlanır. Momentum katsayısı $[0 \quad 1]$ aralığında seçilebilir. Bu katsayı küçük seçildiğinde daha iyi sonuçlar alındığı gözlenmiştir [7].

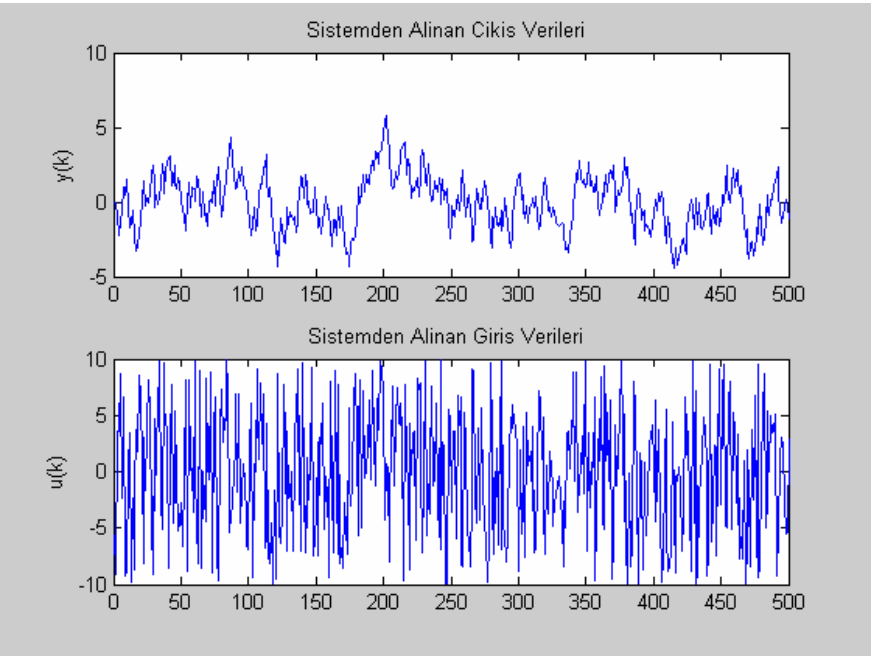
Yapay sinir ağlarında kullanılan aktivasyon fonksyonları sebebi ile giriş ve çıkış değerlerinin ölçeklendirilmesi gerekebilir. Örneğin giriş kümesinde bulunan 10, 100 ve 1000 değerleri için tanjant hiperbolik aktivasyon fonksyonu 1 değerlerini üretecektir. Aktivasyon fonksyonlarının doymasını "saturation" önlemek için bu değerlerin ölçeklendirilmesi gerekmektedir. Çıkışta ise ölçeklendirme işlemi tersine çevrilmelidir. Lineer aktivasyon fonksyonunu kullanan ağlar için ölçeklendirme işlemine gerek yoktur [7].

6. YAPAY SİNİR AĞLARI İLE DİĞER METODLARIN KARŞILAŞTIRILMASI

Bu bölümde giriş ve çıkış verileri bilinen bir sistem için üç farklı yöntem kullanılarak modeller elde edilecek ve bu modeller, kullanılan modelleme tekniklerine dayanan kontrolör yapıları ile kontrol edilecektir. Kısım 6.1'de modeli oluşturulacak ve kontrol edilecek sistem hakkında bilgiler verilecektir. Kısım 6.2'de sistem için lineer bir model elde edilecek ve bu model klasik bir PID ile kontrol edilecektir. Kısım 6.3'de Matlab/Anfis kullanılarak bulanık bir model elde edilecek ve bulanık-PID ile kontrolü sağlanacaktır. Son olarak kısım 6.4'de sistem için yapay sinir ağı modeli oluşturulacak ve bu model öz-uyarlamalı nöro-PID ile kontrol edilecektir.

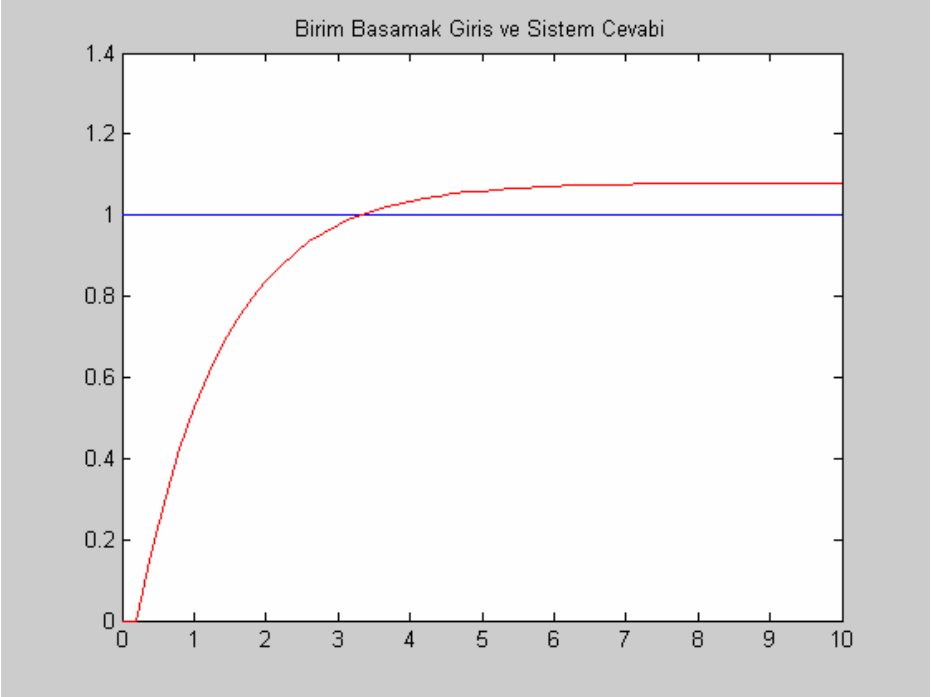
6.1 Sistem Bilgileri

Bu kısımda, modellenecek ve kontrol edilecek sisteme ait bilgiler verilecektir. Sistemin, ölü zamanı bulunan lineer bir süreç olduğu bilinmektedir. Toplam 500 adet giriş ve bu girişlere ait çıkış verileri, 0.2 [s] örnekleme zamanı kullanılarak toplanmıştır. Bu veriler şekil 6.1.1'de gösterilmektedir.



Şekil 6.1.1 : Sisteme ait giriş ve çıkış verileri

Sistemin birim basamak cevabı ise şekil 6.1.2'de verilmektedir. Görüldüğü gibi birim basamak giriş için sistemin sürekli hal hatası bulunmaktadır.



Şekil 6.1.2 : Sistemin birim basamak cevabı

6.2 Lineer Modelin Eldesi ve Klasik PID İle Kontrolü

Toplanan giriş-çıkış verileri için çok sayıda lineer model elde etmek mümkündür. Bunun sebebi keyfi şekilde seçilecek pay derecesi, payda derecesi ve sistem gecikmesidir. Fakat keyfi şekilde belirlenecek bu parametreler ile elde edilen çok sayıda modelden bazıları sistemi iyi bir şekilde temsil edemeyecek ve büyük modelleme hatalarına sebep olacaktır. Bu sebeple, sistem parametreleri, amaca uygun performans kriterleri kullanılarak belirlenir. Bu kısımda üç farklı performans kriteri kullanılarak, lineer modeller elde edilecektir. Bunlardan en az hataya sebep olan model seçilecek ve klasik bir PID ile kontrol edilecektir.

MDL Kriteri (Minimum Description Length) : Bu kriter kullanılarak bulunan ARX modeli aşağıda verilmektedir. Elde edilen modeli değerlendirmek için " V : loss function" ve "FPE : final prediction error" değerleri kullanılabilir. (na : A(q)'nun derecesi, nb : B(q)'nun derecesi, nk : sistem gecikmesi)

[na nb nk] = [2 2 2], V = 1.067 x 10^{-9} ve FPE = 1.087 x 10^{-9}

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.3534 q^-1 - 0.4174 q^-2

B(q) = 0.1658 q^-2 + 0.08176 q^-3
```

A(q) ve B(q) polinomları ile ayrık transfer fonksyonu elde edilebilir.

```
    0.1658 z + 0.08176
--------------------------
z^3 - 0.3534 z^2 - 0.4174 z
```

AIC (Akaike's Information Criterion) : Elde edilen model aşağıda gösterilmektedir. Elde edilen modeli değerlendirmek için " V : loss function" ve "FPE : final prediction error" değerleri kullanılabilir.

[na nb nk] = [8 8 2], V = 9.621 x 10^{-10} ve FPE = 9.059 x 10^{-10}

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.0648 q^-1 - 0.07287 q^-2 - 0.06529 q^-3 - 0.04209 q^-4
         - 0.02262 q^-5 + 0.005239 q^-6 + 0.003642 q^-7 - 0.1597 q^-8

B(q) = 0.1658 q^-2 + 0.1296 q^-3 + 0.09762 q^-4 + 0.07181 q^
        -5 + 0.05381 q^-6 + 0.0418 q^-7 + 0.03625 q^-8 + 0.03129 q^-9
```

A(q) ve B(q) polinomları ile ayrık transfer fonksyonu elde edilebilir.

```
0.1658 z^7 + 0.1296 z^6 + 0.09762 z^5 + 0.07181 z^4 + 0.05381 z^3 + 0.0418 z^2

                                                          + 0.03625 z + 0.03129

-------------------------------------------------------------------------------
z^9 - 0.0648 z^8 - 0.07287 z^7 - 0.06529 z^6 - 0.04209 z^5 - 0.02262 z^4 + 0.005239 z^3

                                                  + 0.003642 z^2 - 0.1597 z
```

CV Yaklaşımı (Cross Validation) : Elde edilen model aşağıda gösterilmektedir. Elde edilen modeli değerlendirmek için " V : loss function" ve "FPE : final prediction error" değerleri kullanılabilir.

[na nb nk] = [10 9 2], V = 9.612 x 10^{-10} ve FPE = 9.053 x 10^{-10}

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.07571 q^-1 - 0.07272 q^-2 - 0.06468 q^-3 - 0.0435 q^-4
       - 0.03297 q^-5 - 0.0004284 q^-6 - 0.009445 q^-7 - 0.1744 q^-8
                                                       + 0.03371 q^-9


B(q) = 0.1658 q^-2 + 0.1278 q^-3 + 0.09612 q^-4 + 0.07064 q^
       -5 + 0.05258 q^-6 + 0.03904 q^-7 + 0.03298 q^-8 + 0.02635 q^-9
                                    - 0.006602 q^-10 + 1.981e-007 q^-11
```

A(q) ve B(q) polinomları ile ayrık transfer fonksyonu elde edilebilir.

```
  0.1658 z^9 + 0.1278 z^8 + 0.09612 z^7 + 0.07064 z^6 + 0.05258 z^5 + 0.03904 z^4

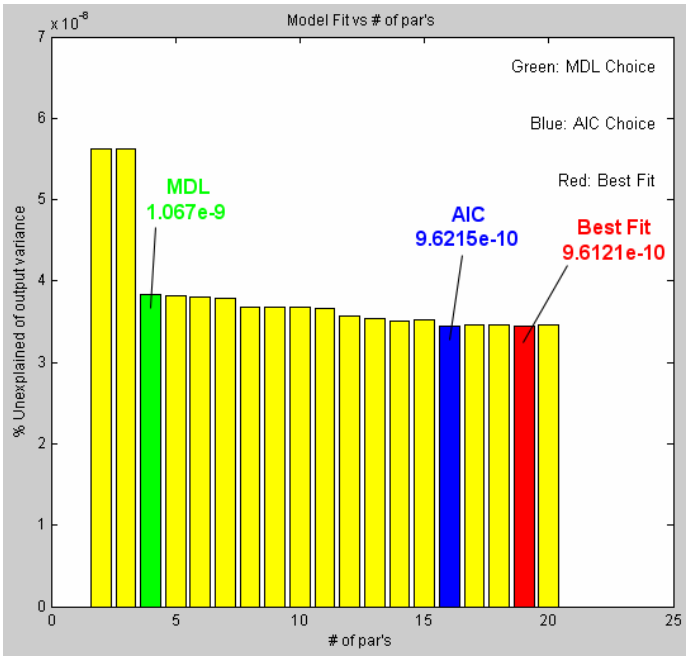
                                + 0.03298 z^3 + 0.02635 z^2 - 0.006602 z + 1.981e-007

-------------------------------------------------------------------------------------
z^11 - 0.07571 z^10 - 0.07272 z^9 - 0.06468 z^8 - 0.0435 z^7 - 0.03297 z^6 - 0.0004284 z^5

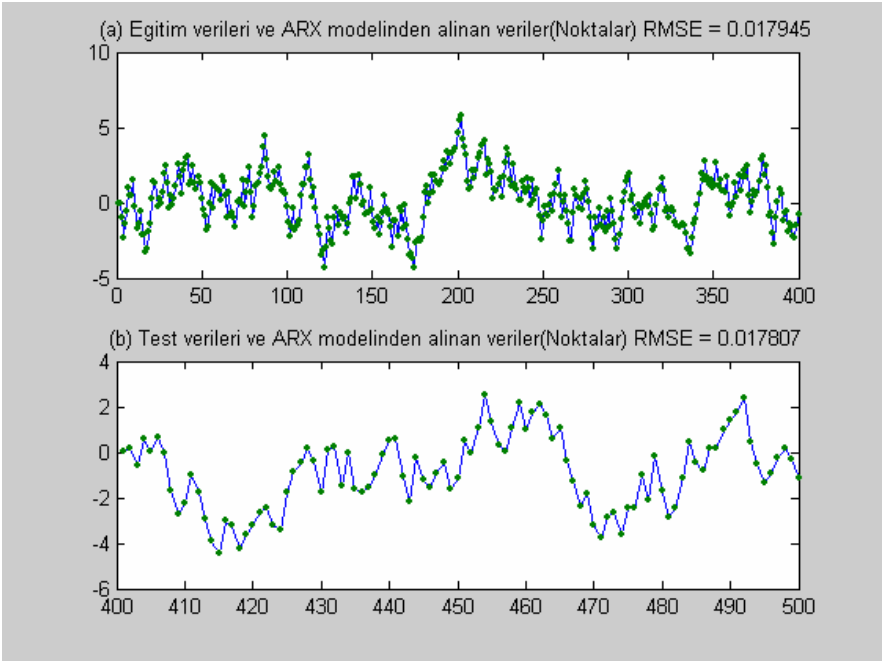
                                    - 0.009445 z^4 - 0.1744 z^3 + 0.03371 z^2
```

Uygulamanın niteliğine ve model elde etmekteki amacımıza bağlı olarak, bu üç farklı yaklaşımın birbirlerine göre avantajları veya dezavantajları olabilir. Örneğin uygulama hassasiyet gerektirmiyorsa veya hesaplamalar ardı arkasına hızlı bir şekilde yapılacaksa bu durumda MDL kriteri ile elde edilen model kullanılabilir. Buna karşın model mertebesinin büyük olması problem teşkil etmiyorsa, CV yaklaşımı daha doğru sonuçlar elde etmemizi sağlayacaktır.

Bu üç farklı kriter için elde edilen sonuçlar şekil 6.2.1'de özetlenmektedir. Yapay sinir ağı ile elde edilecek modeli, en iyi lineer modele karşı test etmek için CV yaklaşımı kullanılacaktır. Sistemin ölçülen çıkışları ve CV yaklaşımı ile elde edilen ARX modelinden alınan çıkışlar şekil 6.2.2'de gösterilmektedir. Sistem için oluşturulan lineer model ile yapılan karesel ortalama hata 10^{-2} mertebesindedir.

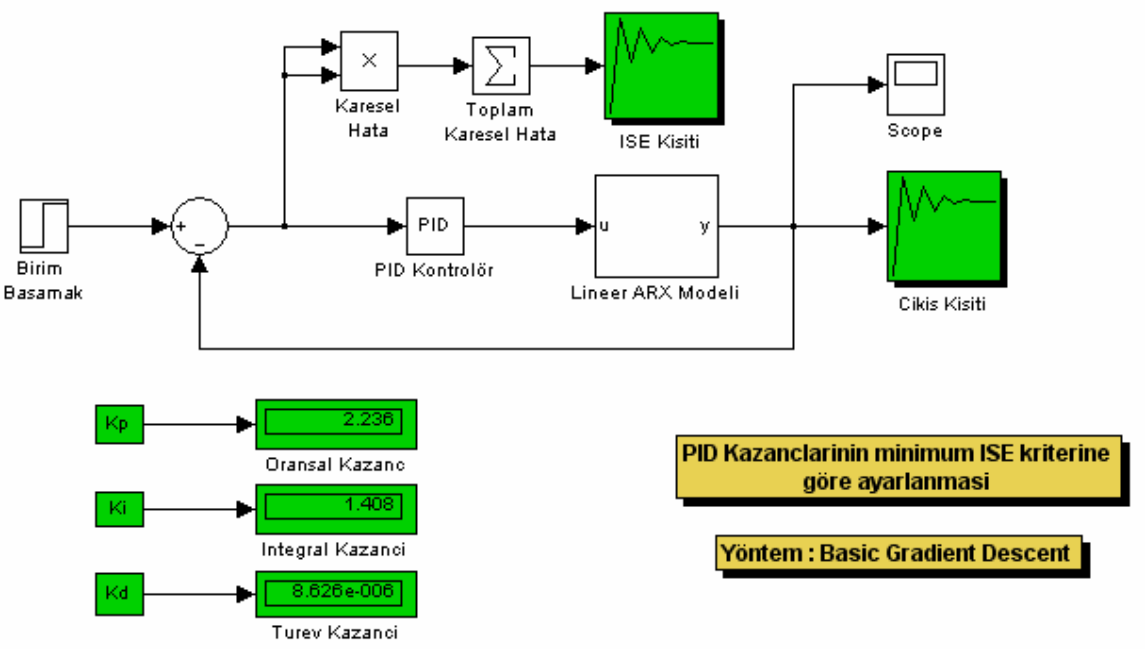


Şekil 6.2.1 : Farklı yaklaşımlar ile elde edilen modeller için alınan sonuçlar



Şekil 6.2.2 : Lineer ARX modelinden alınan çıkışlar ve ölçülen sistem çıkışları

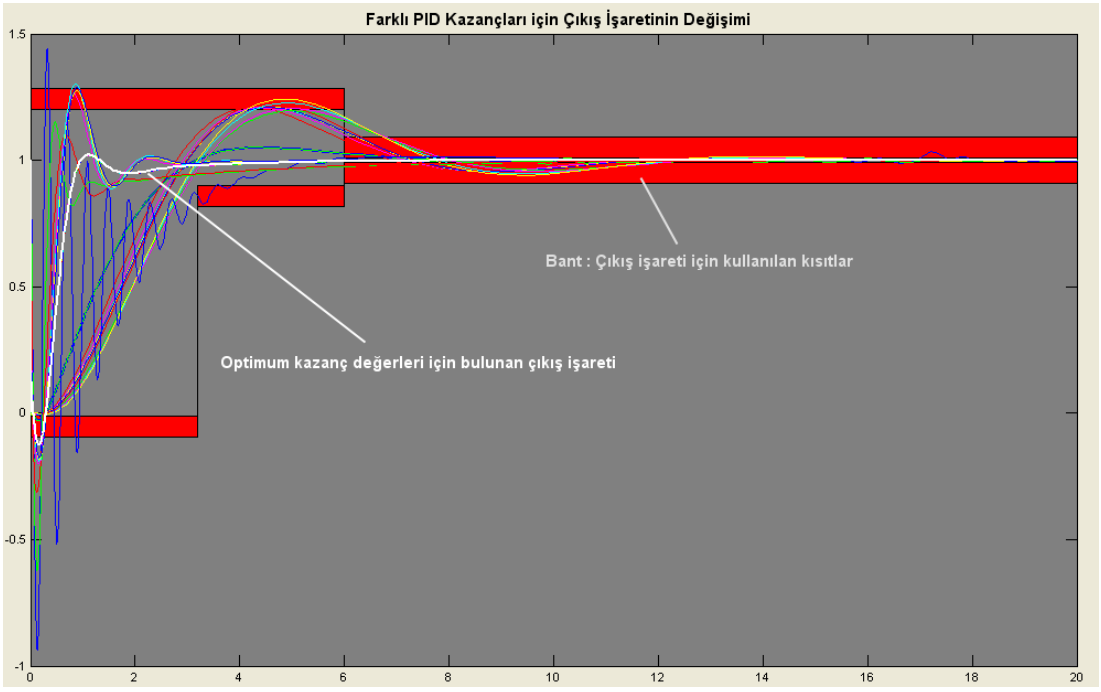
Elde edilen lineer model, klasik bir PID ile kontrol edilecektir. Klasik PID ve ileriki bölümlerde elde edilecek olan bulanık-PID, nöro-PID kontrolörlerini doğru ve anlamlı bir şekilde karşılaştırmak için klasik PID'nin kazanç değerleri ve bulanık-PID'nin ölçeklendirme çarpanları karesel hatayı minimum yapacak şekilde belirlenecektir. Bunun sebebi nöro-PID kontrolörünün geriye yayılım algoritmasını kullanmasıdır. Hatırlanacağı gibi genelleştirilmiş delta kuralına dayanan geriye yayılım algoritması, ağırlık değerlerini, çıkış ile hedef değer arasında tanımlanan karesel hatayı en azlayacak şekilde güncellemektedir. Bu amaçla şekil 6.2.3'de gösterilen blok diagram kullanılmıştır.



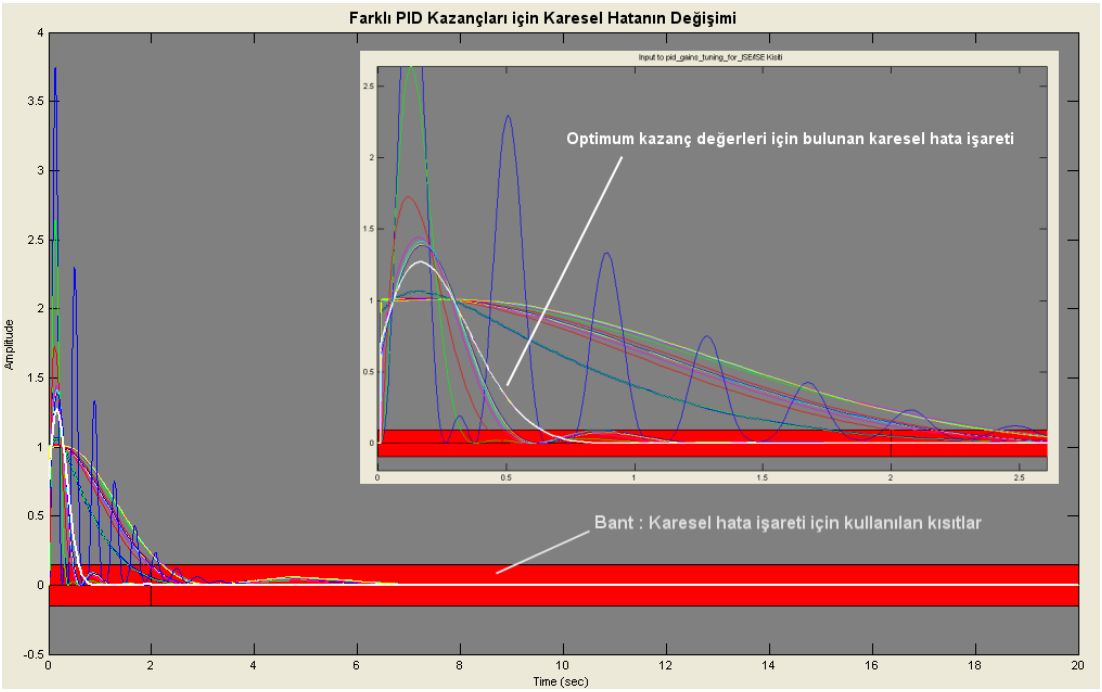
Şekil 6.2.3 : Lineer modelin PID ile kontrolünde kullanılan blok diagram

Çıkış ve karesel hata işaretleri üzerine konulan kısıtlar sayesinde karesel hatayı mimimum yapacak optimum kazanç değerleri bulunmuştur. Bu değerler sırası ile $K_P = 2.2360$, $K_I = 1.4080$ ve $K_D = 8.6260x10^{-6}$'dir. Kazançların optimizasyonu için "basic gradient descent" metodu kullanılmıştır.

Şekil 6.2.4'de kazanç değerlerinin optimizasyonu sırasında, farklı kazanç değerleri için elde edilen birim basamak yanıtları gösterilmektedir. Şekil 6.2.5'de ise bu kazanç değerleri için elde edilen karesel hata işaretleri verilmektedir.



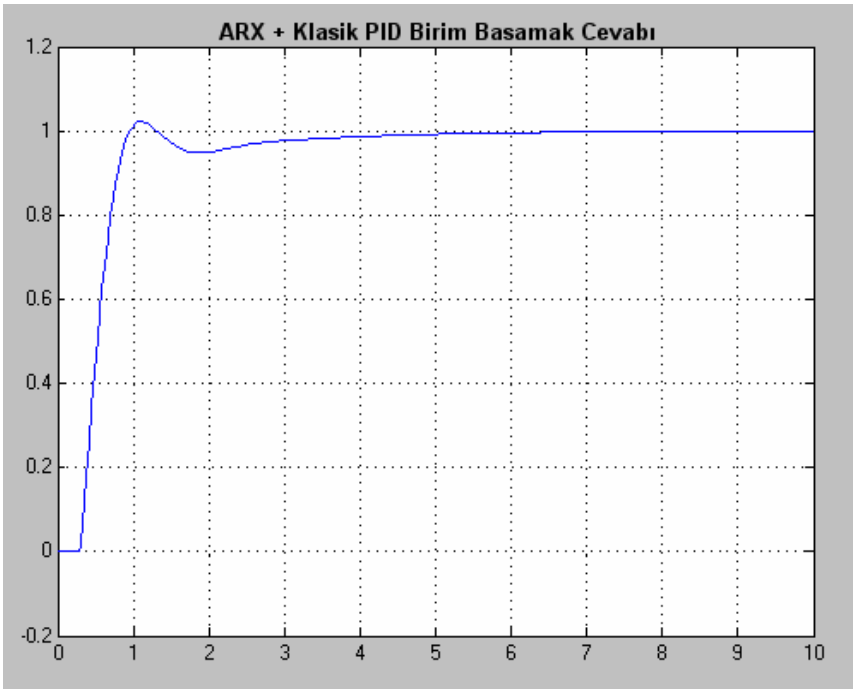
Şekil 6.2.4 : Farklı kazanç değerleri için elde edilen birim basamak yanıtları



Şekil 6.2.5 : Farklı kazanç değerleri için elde edilen karesel hata işaretleri

Şekil 6.2.6'de kazanç değerleri karesel hatayı en azlayacak şekilde belirlenen klasik PID ile kontrol edilen modelin birim basamak yanıtı gösterilmektedir.

$K_P = 2.2360$, $K_I = 1.4080$ ve $K_D = 8.6260x10^{-6}$



Şekil 6.2.6 : Karesel hatayı en azlayacak şekilde belirlenen kazanç değerleri için sistemin birim basamak yanıtı

Sistem için oluşturulan modelin, klasik PID ile kontrol edilmesinde karşılaşılan en büyük problem, karesel hatayı en azlayacak K_P, K_I ve K_D parametrelerinin belirlenmesi olmuştur. Bu parametrelerin optimizasyonu için "basic gradient descent" methodu kullanılmıştır. Yöntemin yakınsama hızının düşük olmasından dolayı işlem saatler mertebesi ile belirtilecek kadar uzun sürmüştür.

6.3 Bulanık Modelin Eldesi ve Bulanık-PID İle Kontrolü

Eğer modellemek istediğimiz sistem, dinamik olmasaydı, yani verilen girişler için çıkışları elde edebilseydik, bu durumda giriş ve çıkış değerlerinin ANFIS'e tanıtılması ile modeli kolaylıkla elde edebilirdik. Fakat ilgilendiğimiz sistem, dinamikdir ve çıkışlar sadece o anki girişler ile değil, ancak daha önceki değerlerinde kullanılmasıyla elde edilebilir. Bu sebeple bulanık modelin oluşturulması için

yapılacak ilk işlem, önceki değerlerden hangilerinin kullanılması gerektiğinin saptanmasıdır.

Giriş ve çıkışa ait veriler kullanılarak, önceki değerler elde edilebilir ve ANFIS'e tanıtılarak deneme-yanılma yöntemi ile bu terimlerden hangilerinin hatayı minimum yaptığı belirlenebilir. Fakat bu çok çaba ve zaman gerektiren bir işlemdir. Özelliklede mertebeleri büyük olan sistemler için bu yöntem hiç pratik olmayacaktır.

Bu sebeple önceki değerlerden hangilerinin seçileceği, deneme-yanılma ile değil, ANFIS'in çok sayıda kombinasyonu denemesi ile bulunacaktır. Bu işlem *seqsrch* veya *exhsrch* komutları ile yapılabilir. *seqsrch* komutu ile elde edilen sonuç aşağıda gösterilmektedir.

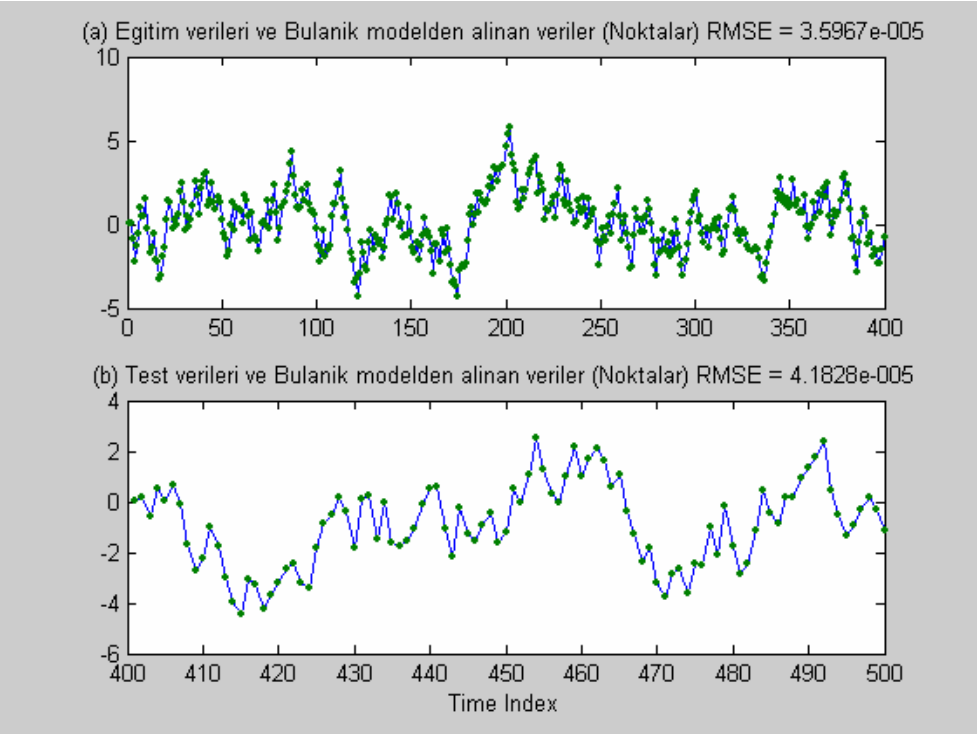
```
Selecting input 1 ...
ANFIS model 1: y(k-1) --> trn=0.9292, chk=0.9531
ANFIS model 2: y(k-2) --> trn=1.2232, chk=1.2700
ANFIS model 3: y(k-3) --> trn=1.3777, chk=1.4311
ANFIS model 4: y(k-4) --> trn=1.4949, chk=1.5642
ANFIS model 5: u(k-1) --> trn=1.7207, chk=1.9369
ANFIS model 6: u(k-2) --> trn=1.4560, chk=1.6384
ANFIS model 7: u(k-3) --> trn=1.5257, chk=1.7359
ANFIS model 8: u(k-4) --> trn=1.5928, chk=1.8467
ANFIS model 9: u(k-5) --> trn=1.6145, chk=1.8700
ANFIS model 10: u(k-6) --> trn=1.6484, chk=1.8851
Currently selected inputs: y(k-1)

Selecting input 2 ...
ANFIS model 11: y(k-1) y(k-2) --> trn=0.9190, chk=0.9691
ANFIS model 12: y(k-1) y(k-3) --> trn=0.9171, chk=0.9797
ANFIS model 13: y(k-1) y(k-4) --> trn=0.9215, chk=0.9639
ANFIS model 14: y(k-1) u(k-1) --> trn=0.9262, chk=0.9605
ANFIS model 15: y(k-1) u(k-2) --> trn=0.0000, chk=0.0000
ANFIS model 16: y(k-1) u(k-3) --> trn=0.9179, chk=0.9671
ANFIS model 17: y(k-1) u(k-4) --> trn=0.9253, chk=0.9560
ANFIS model 18: y(k-1) u(k-5) --> trn=0.9223, chk=0.9677
ANFIS model 19: y(k-1) u(k-6) --> trn=0.9227, chk=0.9668
Currently selected inputs: y(k-1) u(k-2)
```

Belirtilmelidir ki, bu komutlar en fazla 4 tane önceki değer belirleyebilmektedir. Eğer sistem yüksek mertebeli ise deneme-yanılma metodu kullanılacaktır. Bu durum, bulanık modelleme için büyük bir dezavantaj oluşturmaktadır. Dikkat edilirse, lineer modelin eldesinde benzer bir işlem yapılmıştır. Fakat pay-payda derecelerinin ve sistem gecikmesinin belirlenmesini sağlayan CV, AIC ve MDL kriterleri için

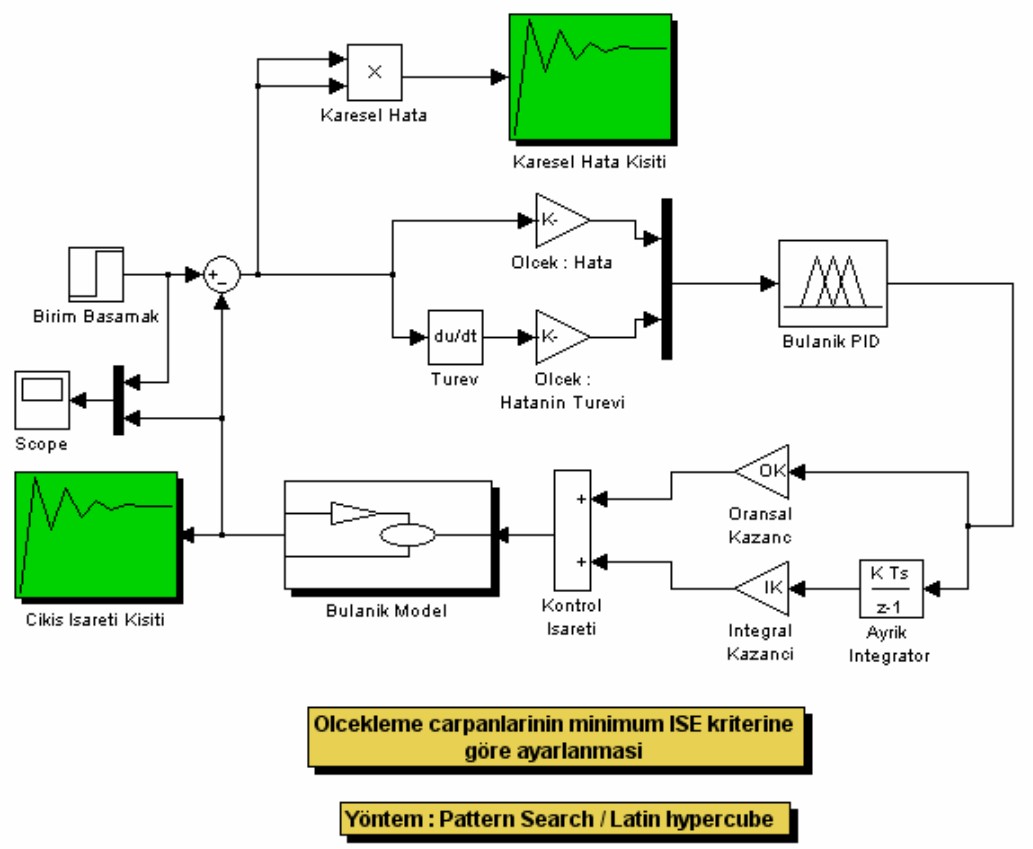
herhangi bir kısıt olmadığından, aynı dezavantaj lineer modelin eldesinde yaşanmamıştır. Modellemek istediğimiz sistem çok yüksek mertebeli olsa bile, lineer bir model elde edilebilir. Aynı şeyi bulanık model için söylemek zordur.

Yukarda görüldüğü gibi *seqsrch* komutu kullanılması gereken önceki değerleri $y(k-1)$ ve $u(k-2)$ olarak belirlemiştir. Bu değerler kullanılarak elde edilen bulanık model çıkışı ve sistemin ölçülen çıkışları şekil 6.3.1'de gösterilmektedir. Sistem için elde edilen bulanık model ile yapılan hata 10^{-5} mertebesindedir.



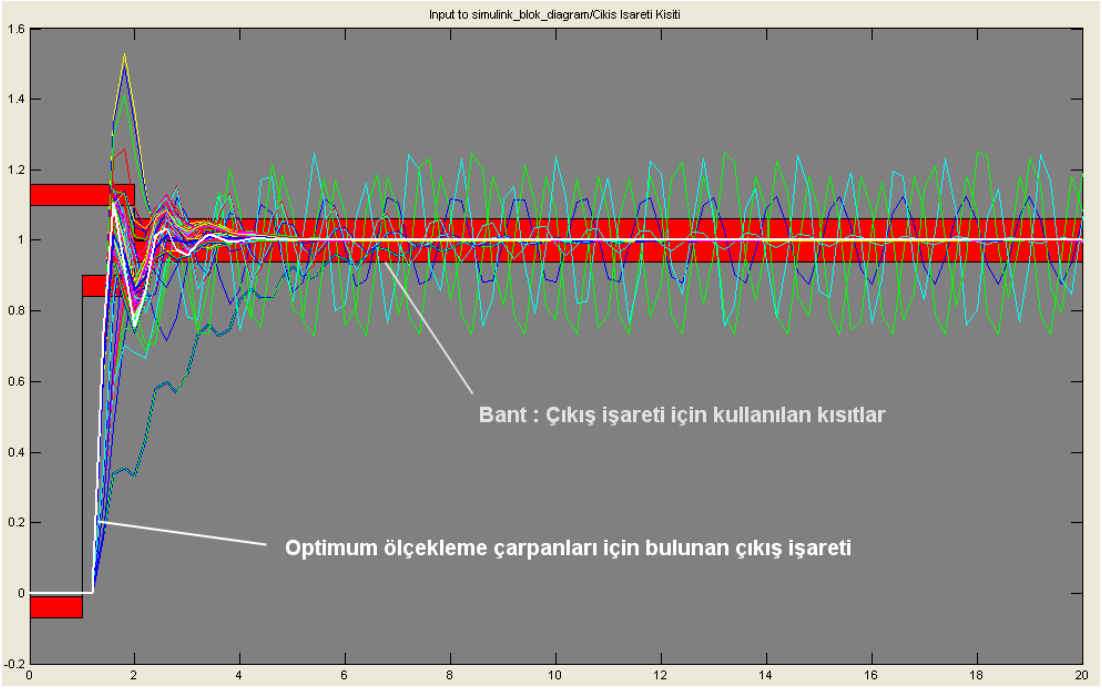
Şekil 6.3.1 : Sistemin ölçülen çıkışı ve bulanık modelden alınan çıkış değerleri

Elde edilen bulanık modelin kontrolü, bulanık-PID ile sağlanacaktır. Önceki bölümde olduğu gibi, bulanık-PID kontrolörünün ölçeklendirme çarpanları karesel hatayı en azlayacak şekilde belirlenecektir. Bu amaç ile şekil 6.3.2'de gösterilen blok diagram kullanılmıştır. Ölçeklendirme çarpanlarının optimizasyonu için "basic gradient descent" metodu ile başarı sağlanamadığından "pattern search : latin hypercube" metodu kullanılmıştır. Bu metod kullanılabilecek genetik algoritma ve Nelder-Mead yöntemlerine göre daha hızlı olduğu için tercih edilmiştir.

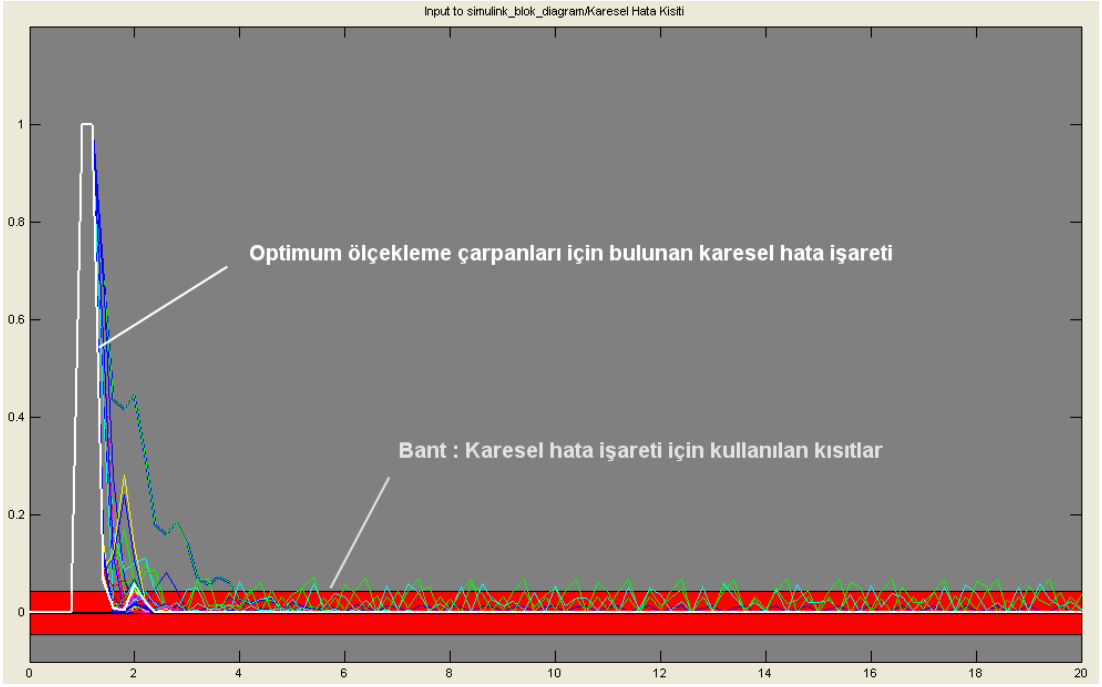


Şekil 6.3.2 : Bulanık modelin bulanık-PID ile kontrolünde kullanılan blok diagram

Şekil 6.3.3'de kazanç değerlerinin optimizasyonu sırasında, farklı kazanç değerleri için elde edilen birim basamak yanıtları, şekil 6.3.4'de ise karesel hata işaretleri gösterilmektedir.

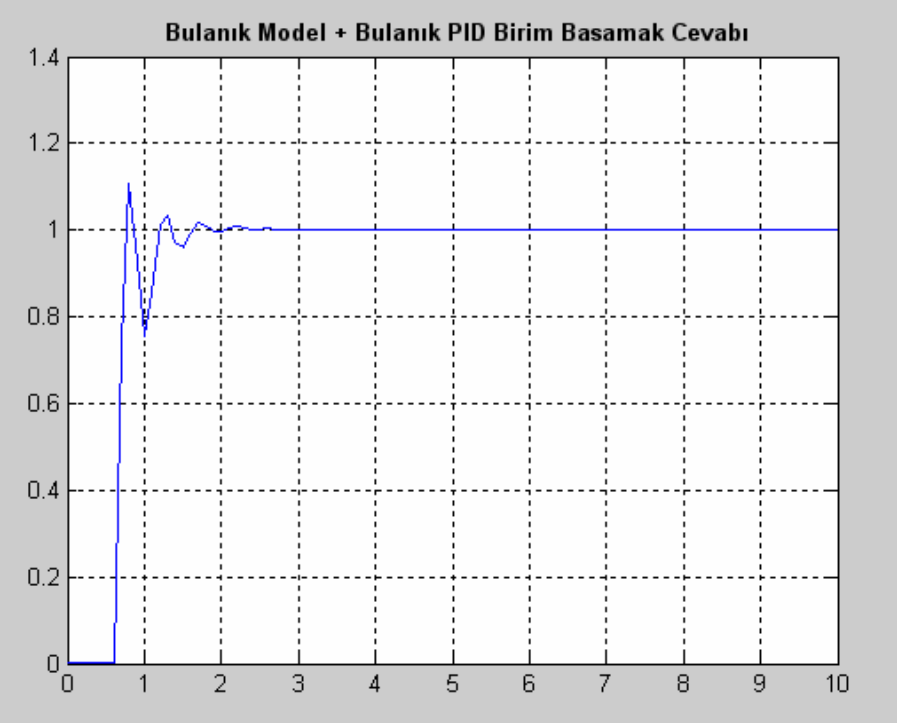


Şekil 6.3.3 : Farklı kazanç değerleri için elde edilen birim basamak yanıtları



Şekil 6.3.4 : Farklı kazanç değerleri için elde edilen karesel hata işaretleri

Şekil 6.3.5'de kazanç değerleri karesel hatayı en azlayacak şekilde belirlenen bulanık-PID ile kontrol edilen modelin birim basamak yanıtı gösterilmektedir. "Latin hypercube" metodu ile bulunan değerler $OC1 = 0.23$ (hata işareti için ölçekleme çarpanı), $OC2 = 0.15$ (hatanın türevi için ölçekleme çarpanı), $OK = 8.3925$ (oransal ölçekleme çarpanı) ve $IK = 18.8850$ (integral ölçekleme çarpanı)'dir.



Şekil 6.3.5 : Ölçekleme çarpanları karesel hatayı en azlayacak şekilde belirlenen bulanık-PID ile kontrol edilen modelin birim basamak yanıtı

6.4 Yapay Sinir Ağı Modelinin Eldesi ve Öz-Uyarlamalı Nöro-PID İle Kontrolü

Bu kısımda, giriş-çıkış değerleri bilinen sistem için çok katmanlı ileribeslemeli yapay sinir ağı modeli elde edilecektir. Bu amaçla Matlab ile bir m-dosyası hazırlanmıştır. Bu dosya Ek.B'de verilmektedir. Ayrıca hazırlanan bu dosyanın doğruluğunu ölçmek amacı ile Matlab/Neural Network Toolbox kullanılarak bulunmuş geribeslemeli Elman-tipi yapay sinir ağı modeli ile bir karşılaştırma yapılacaktır. Son olarak, sistem için elde edilen yapay sinir ağı modeli, öz-uyarlamalı nöro-PID ile kontrol edilecek ve daha önce incelenen yaklaşımlar ile karşılaştırmalar yapılacaktır.

Hazırlanan m-dosyası "ileribeslemeliegit" komutu ile çağrılmaktadır. Bu komut giriş, hedef, defa ve hata olmak üzere dört parametre kabul etmektedir. Defa ve hata

parametreleri belirtilmediğinde sırayla 5000 ve 10^{-8} olarak kabul edilmektedir. Komut, eğitim verilerinin yüzdesini, öğrenme ve momentum katsayısını kullanıcıdan isteyen bir interaktif bölüm ile başlamaktadır.

```
>> ileribeslemeliegit(giris,hedef,10000)

Giris ve hedef vektörleri 500 tane veri kumesi icermektedir
Bunlarin % kaci egitim amaci ile kullanilsin?  Onerilen deger %80 ve yukarisi
0 - 100 arasinda bir rakam girmeniz beklenmektedir > 100

Sistem 2-girisli 1-cikisli oldugundan
giris katmaninda 2 tane ve cikis katmaninda 1 tane hucre bulunacaktir
Gizli katmanda kac tane gizli hucre bulunsun? > 20

Ogrenme katsayisini giriniz
Ogrenme katsayisi [0 1] araliginda olabilir
[0.1 0.3] araligi önerilmektedir > 0.3

Momentum katsayisini giriniz
Momentum katsayisi [0 1] araliginda olabilir
[0.1 0.2] araligi önerilmektedir > 0.1
```

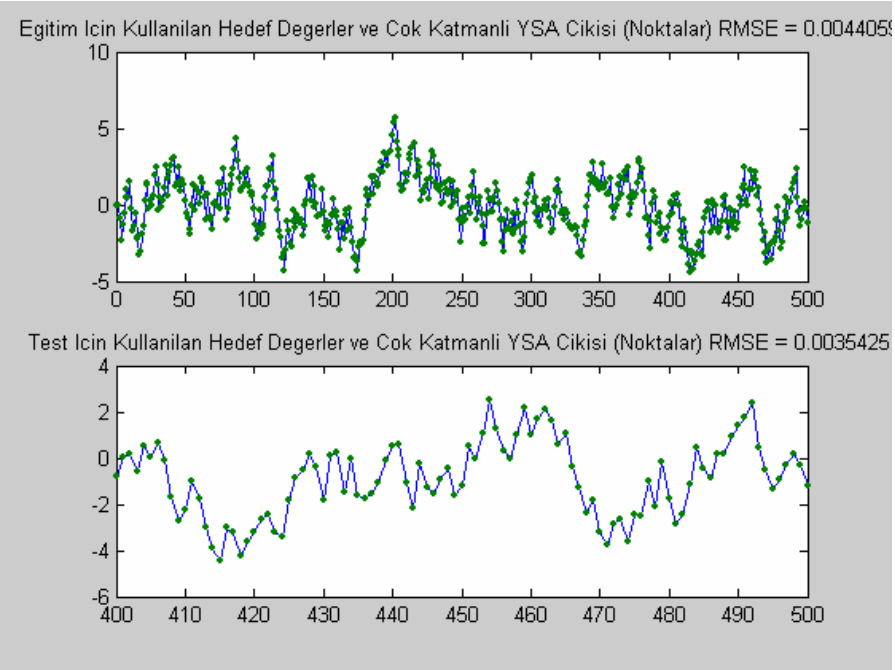
Hatırlanacağı gibi, bulanık modelin eldesinde, önceki değerlerden hangisinin kullanılması gerektiği bizim tarafımızdan bulunmuştu. Yapay sinir ağlarında ise bu işlemin yapılmasına gerek kalmamaktadır. Ağ bu ilişkileri belirleyebilmektedir. Bu durum yapay sinir ağlarına büyük bir üstünlük sağlamaktadır. Bununla birlikte, ağa $u(k-2)$ ve $y(k-1)$ önceki değerlerinin tanıtılması durumunda istenilen hata değerine daha az iterasyon ile ulaşılacağı unutulmamalıdır.

Yapay sinir ağlarının bu konudaki üstünlüğünü vurgulayabilmek için, önceki değerlerden hangilerinin kullanılması gerektiğini bilmediğimizi farzedelim ve ağa $u(k-2)$ ve $y(k-1)$ önceki değerleri değil, $u(k)$ ve $y(k)$ şimdiki değerleri tanıtalım.

Belirlenen iterasyon sayısı sonunda, yazılan algoritma karesel hatayı 10^{-5} mertebesine kadar azaltmıştır. Sonuçlar aşağıda gösterilmektedir.

```
Toplam 10000 iterasyon yapilmasi yada 1e-008 karesel hataya ulasilmasi planlanmaktadir
Ag egitiliyor : 100.iterasyonda yapilan karesel hata 0.00091431 dir
Ag egitiliyor : 200.iterasyonda yapilan karesel hata 0.00081443 dir
Ag egitiliyor : 300.iterasyonda yapilan karesel hata 0.00073988 dir
Ag egitiliyor : 400.iterasyonda yapilan karesel hata 0.00068116 dir
Ag egitiliyor : 500.iterasyonda yapilan karesel hata 0.00063322 dir
Ag egitiliyor : 600.iterasyonda yapilan karesel hata 0.00059307 dir
Ag egitiliyor : 700.iterasyonda yapilan karesel hata 0.00055881 dir
                    :                   :
Ag egitiliyor : 9300.iterasyonda yapilan karesel hata 0.00010099 dir
Ag egitiliyor : 9400.iterasyonda yapilan karesel hata 0.00010043 dir
Ag egitiliyor : 9500.iterasyonda yapilan karesel hata 9.9885e-005 dir
Ag egitiliyor : 9600.iterasyonda yapilan karesel hata 9.9347e-005 dir
Ag egitiliyor : 9700.iterasyonda yapilan karesel hata 9.8817e-005 dir
Ag egitiliyor : 9800.iterasyonda yapilan karesel hata 9.8295e-005 dir
Ag egitiliyor : 9900.iterasyonda yapilan karesel hata 9.778e-005 dir
Ag egitiliyor : 10000.iterasyonda yapilan karesel hata 9.7274e-005 dir
```

Eğitim sonunda elde edilen yapay sinir ağı modelinden alınan çıkış verileri ve ölçülen gerçek çıkış değerleri şekil 6.4.1'de gösterilmektedir.

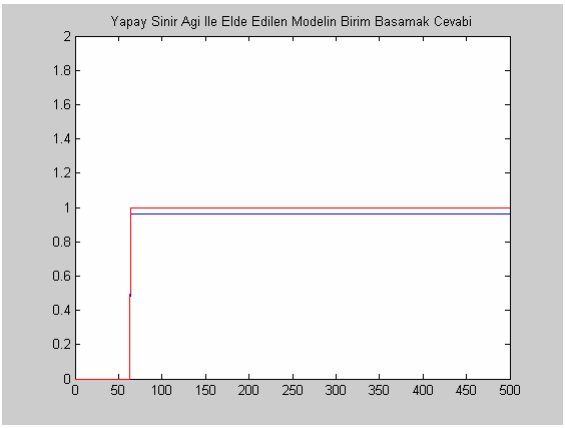


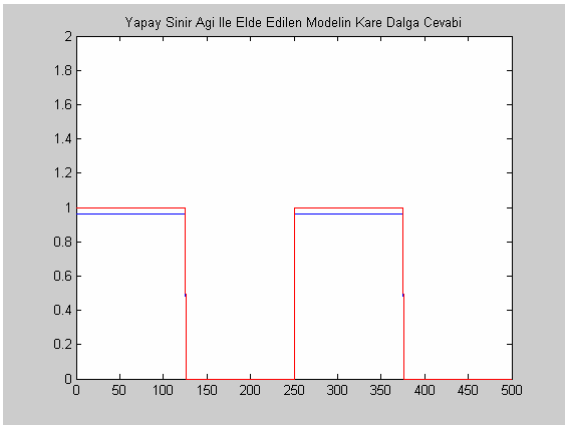
Şekil 6.4.1 : Yapay sinir ağı modelinden alınan değerler ve ölçülen sistem değerleri

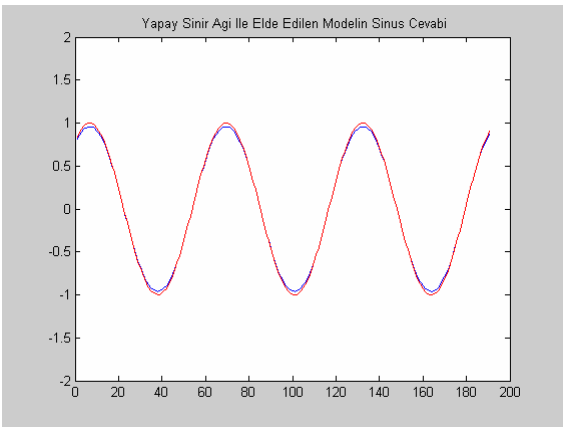
Eğitim işlemi bittikten sonra yapılacak benzetimler kullanıcı tarafından belirlenmektedir. Sonuçlar şekil 6.4.2'de gösterilmektedir.

```
Yapay sinir agi ile modellenen sistemin benzetimini yapmak istediginiz girisi secin
1 : basamak giris
2 : kare dalga
3 : sinus giris
4 : hepsi
(1,2,3,4)? > 4

Agin egitimi bitmistir. Program sonlandirildi
Yeni elde edilen wga ve wac agirliklari, agirliklar isimli dosyaya kaydedildi
```







Şekil 6.4.2 : Yapay sinir ağı modelinin basamak, kare ve sinüs girişleri için benzetimi

Beklendiği gibi sistemin birim basamak giriş için sürekli hal hatası bulunmaktadır. Modeli öz-uyarlamalı nöro-PID ile kontrol etmeye başlamadan önce, elde edilen sonuçların göreli doğruluğunu, geribeslemeli Elman-tipi yapay sinir ağı ile test edelim.

Elman-tipi yapay sinir ağı modeli için Matlab/Neural Network Toolbox hazır komutları kullanılmıştır. Hazırlanan m-dosyası "geribeslemeliegit" komutu ile çağrılmaktadır. Bilgilerin kullanıcı tarafından oluşturulduğu interaktif kısım aşağıda gösterilmektedir.

```
>> geribeslemeliegit(giris',cikis',10000)

Lutfen gizli katman sayisini belirleyiniz. Keyfi sayida gizli katman olabilir
Az sayida gizli katman ile baslayip,
basari saglanamaz ise artirmaniz önerilmektedir > 1

Sistem 2-girisli 1-cikisli oldugundan
giris katmaninda 2 tane ve cikis katmaninda 1 tane hucre bulunacaktir
Gizli katman(lar)da kac tane gizli hucre bulunsun?

Giris katmanindaki hucre sayisi > 2
1.gizli katmandaki gizli hucre sayisi > 12
Cikis katmanindaki hucre sayisi > 1

Giris ve gizli katmanlarin aktivasyon fonksyonlari tanjant hiperbolik
yada logistic fonksyon olabilir
Cikis katmaninin aktivasyon fonksyonu ise lineer olmalidir
Lutfen tanjant icin tansig, logistic icin logsig ve lineer icin purelin yaziniz

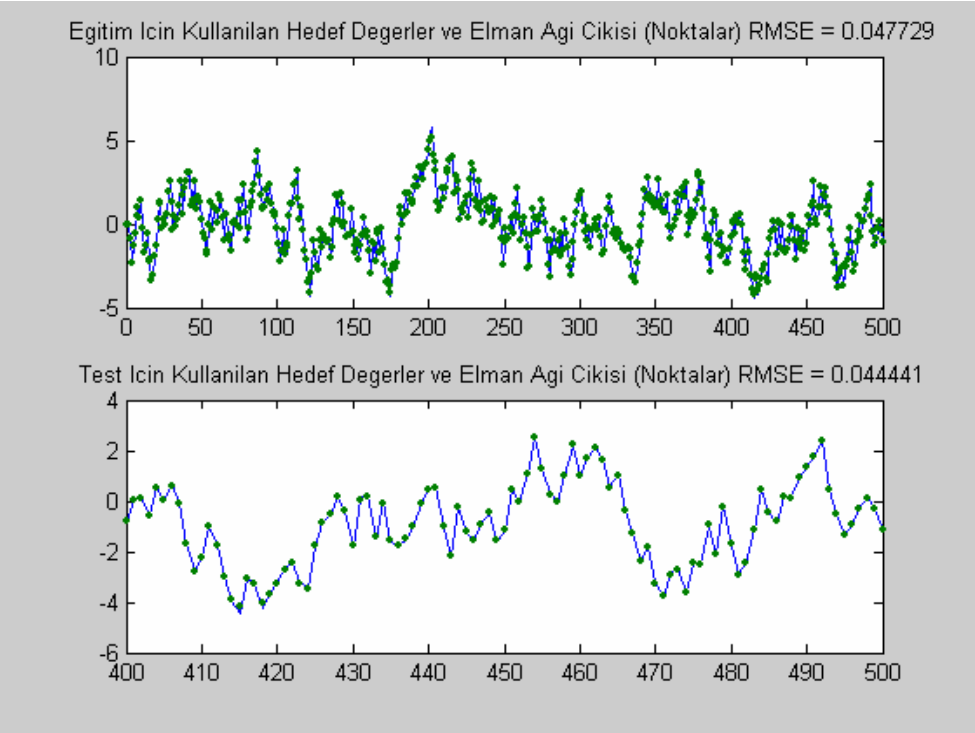
Giris katmaninin aktivasyon fonksyonu > logsig
1.gizli katmanin aktivasyon fonksyonu > logsig
Cikis katmaninin aktivasyon fonksyonu > purelin

Elman aglarinin egitimi icin kullanilmasi tavsiye edilen egitim yontemleri
traingd : gradient descent
traingdm : gradient descent with momentum
traingda : gradient descent with adaptive learning rate
traingdx : gradient descent with momentum and adaptive learning rate

Lutfen egitim yontemini belirleyiniz. Bellek problemi olmayan bilgisayarlar icin
traingdx önerilmektedir. Bu yontem icin -out of memory- problemi dogabilir > traingdx

Giris ve hedef vektörleri 500 tane veri kumesi icermektedir
Bunlarin % kaci egitim amaci ile kullanilsin?  Onerilen deger %80 ve yukarisi
0 - 100 arasinda bir rakam girmeniz beklenmektedir > 100
```

Eğitim sonunda elde edilen Elman-tipi yapay sinir ağı modelinden alınan çıkış verileri ve ölçülen gerçek çıkış değerleri şekil 6.4.3'de gösterilmektedir.



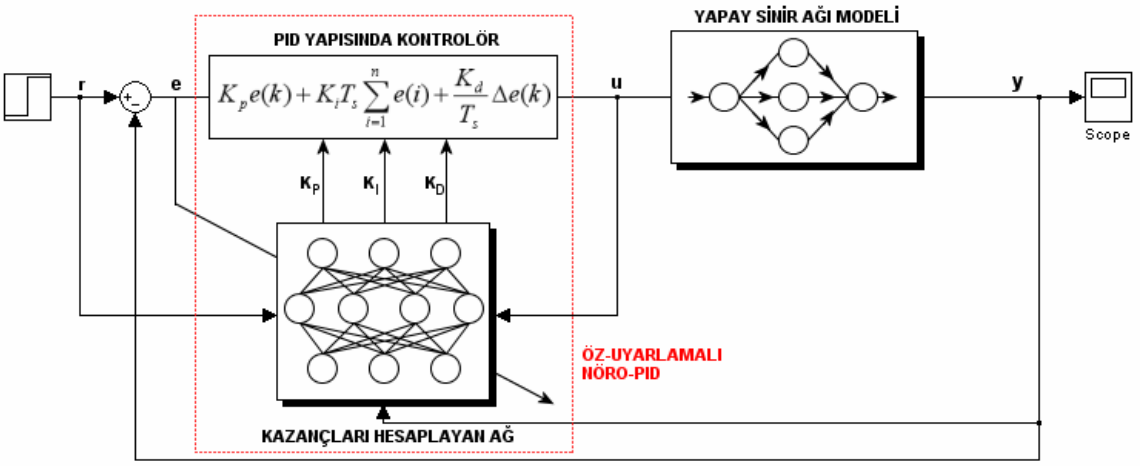
Şekil 6.4.3 : Elman-tipi yapay sinir ağı ile elde edilen sistem modeli

Elman-tipi geribeslemeli yapay sinir ağı ile elde edilen karesel ortalama hata 10^{-2} mertebesindedir. Bu değer ileribeslemeli ağ için 10^{-3} mertebesindeydi. Elman ağı için tek gizli katman ve bu gizli katmanda 12 gizli hücre kullanılmıştır. Burdaki amaç tek gizli katmanı ve 10 gizli hücresi bulunan ileribeslemeli ağın performansını değerlendirmektir. İleribeslemeli ağın performansının iyi olduğu söylenebilir.

Bununla birlikte, giriş olarak $u(k)$ ve $y(k)$ yerine bulanık modelde kullanıldığı gibi $u(k-2)$ ve $y(k-1)$ önceki değerleri kullanılsaydı, yapılan karesel ortalama hatayı, bulanık model ile elde edilen 10^{-5} mertebesine yaklaştırmak mümkün olabilirdi. Gerçektende giriş için $u(k-2)$ ve $y(k-1)$ önceki değerleri kullanılarak oluşturulan ileribeslemeli ağ ile elde edilen model ile yapılan karesel ortalama hata 10^{-4} mertebesinde bulunmuştur.

Bu incelemeden anlaşılacağı gibi eğer önceki değerlerden hangilerinin kullanılması gerektiği biliniyorsa, bulanık model daha küçük bir hataya sebep olacaktır. Fakat, önceki değerlerden hangilerinin kullanılması gerektiği belirlenemiyorsa, ki uygulamalarda sıkça karşılaşıldığı gibi yüksek mertebeli sistemler için belirlenememektedir, bu durumda yapay sinir ağları ile kabul edilebilir düzeylerde hatalara sebep olan modeller oluşturulabilmektedir. Aynı şeyi bulanık model için söylemek mümkün değildir.

Bundan sonraki kısımda ileribeslemeli ağ ile elde edilen model, öz-uyarlamalı nöro-PID ile kontrol edilecektir. Bu amaç ile şekil 6.4.4'de gösterildiği gibi, kazanç değerleri yapay sinir ağı ile hesaplanacak ve bu değerler, PID yapısında bir kontrolörden geçirilerek, kontrol işareti üretilecektir.



Şekil 6.4.4 : YSA modelinin öz-uyarlamalı nöro-PID ile kontrolüne ilişkin blok diagram [9]

```
>> ileribeslemelikontrolet(birimbasamak,wga,wac,1000)

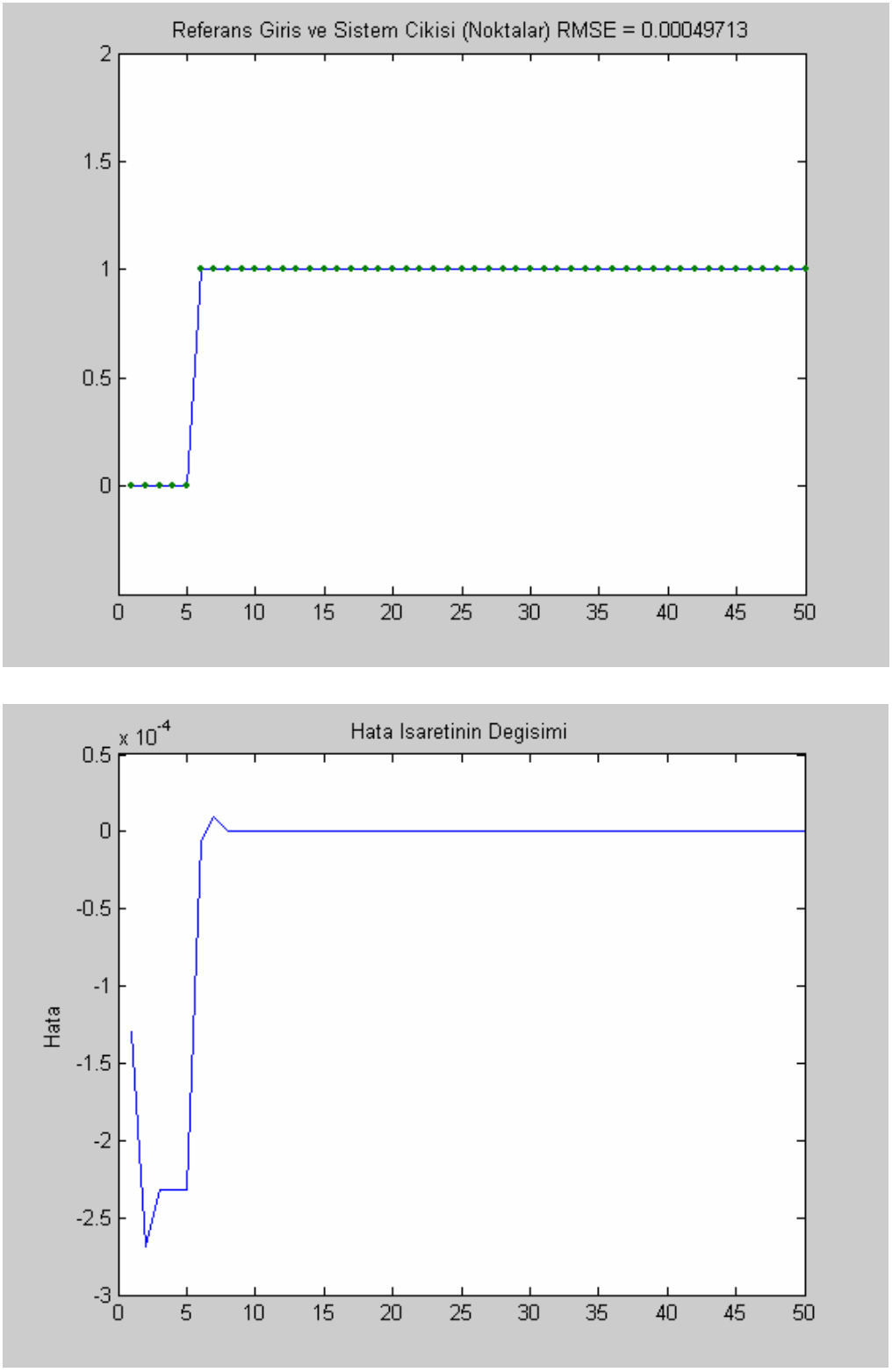
Kontrolörün egitiminde kullanilacak gizli hücre sayisini giriniz > 5

Ogrenme katsayisini giriniz
Ogrenme katsayisi [0 1] araliginda olabilir
[0.1 0.3] araligi önerilmektedir > 0.3

Momentum katsayisini giriniz
Momentum katsayisi [0 1] araliginda olabilir
[0.1 0.2] araligi önerilmektedir > 0.1

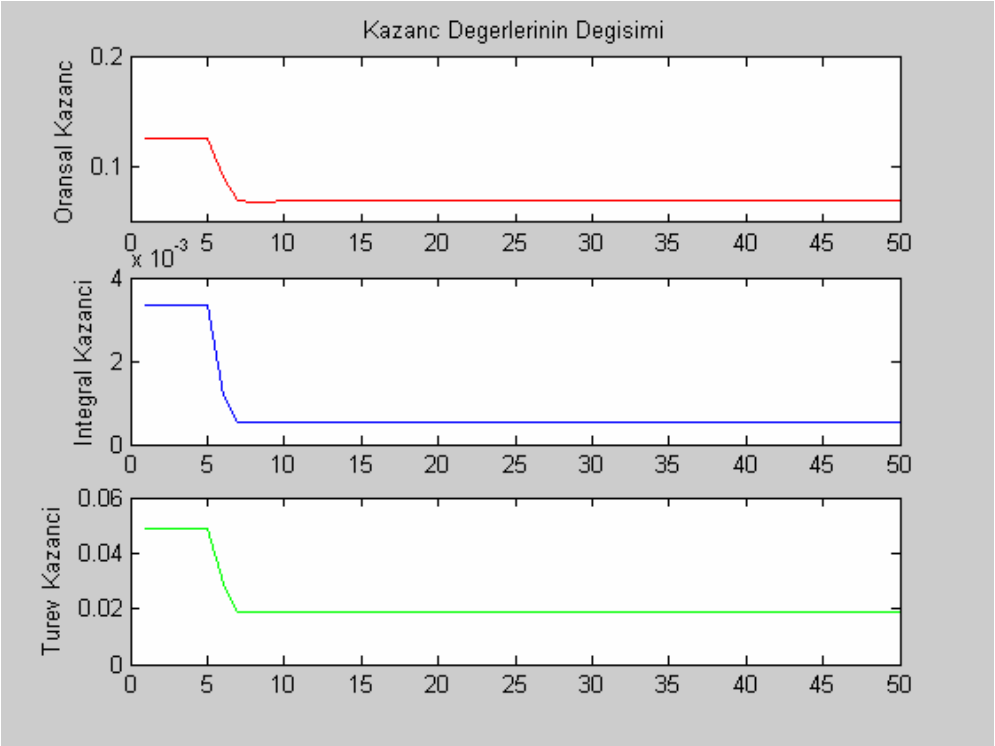
Toplam 1000 iterasyon yapilmasi yada le-008 karesel hataya ulasilmasi planlanmaktadir
Nöro-PID parametreleri arastiriliyor : 100.iterasyonda yapilan karesel hata 6.0705e-006 dir
Nöro-PID parametreleri arastiriliyor : 200.iterasyonda yapilan karesel hata 4.1605e-006 dir
Nöro-PID parametreleri arastiriliyor : 300.iterasyonda yapilan karesel hata 2.4727e-006 dir
Nöro-PID parametreleri arastiriliyor : 400.iterasyonda yapilan karesel hata 1.3077e-006 dir
Nöro-PID parametreleri arastiriliyor : 500.iterasyonda yapilan karesel hata 6.7419e-007 dir
Nöro-PID parametreleri arastiriliyor : 600.iterasyonda yapilan karesel hata 3.9412e-007 dir
Nöro-PID parametreleri arastiriliyor : 700.iterasyonda yapilan karesel hata 2.8098e-007 dir
Nöro-PID parametreleri arastiriliyor : 800.iterasyonda yapilan karesel hata 2.4562e-007 dir
Nöro-PID parametreleri arastiriliyor : 900.iterasyonda yapilan karesel hata 2.4143e-007 dir
Nöro-PID parametreleri arastiriliyor : 1000.iterasyonda yapilan karesel hata 2.4714e-007 dir
```

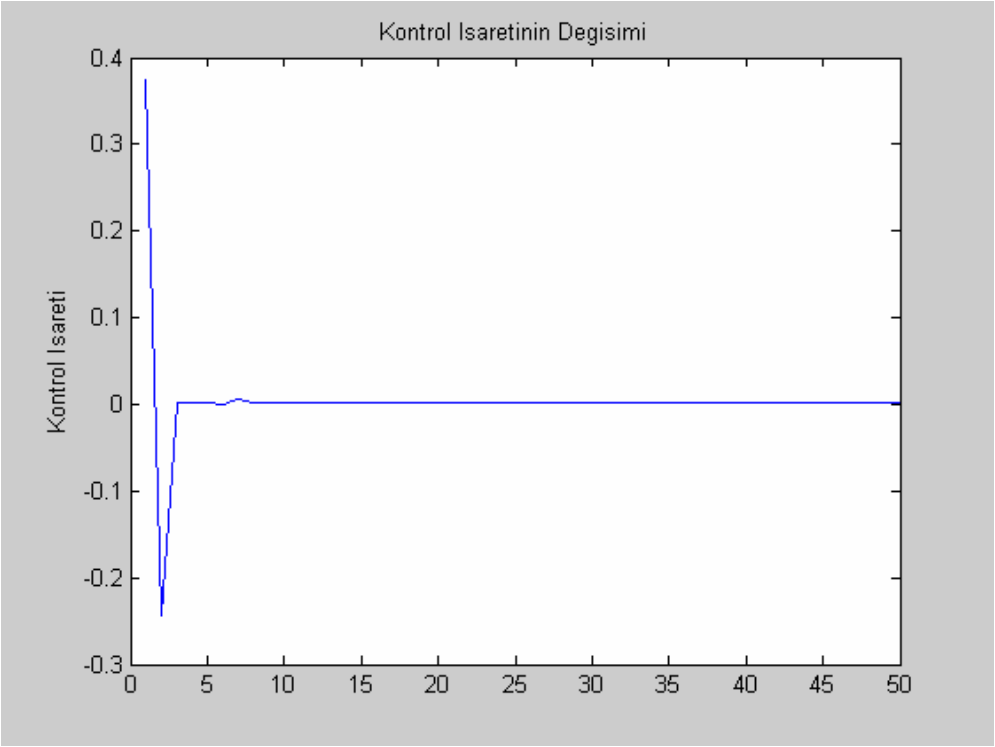
Öz-uyarlamalı nöro-PID ile kontrol edilen sistemin birim basamak cevabı ve hatanın değişimi şekil 6.4.5'de gösterilmektedir.



Şekil 6.4.5 : Nöro-PID ile kontrol edilen sistemin birim basamak cevabı ve hatanın değişimi

K_P, K_I ve K_D kazançlarının değişimi ve bu işaretlerden elde edilen kontrol işareti şekil 6.4.6'da gösterilmektedir.





Şekil 6.4.6 : K_P, K_I ve K_D kazançlarının değişimi ve bu işaretlerden elde edilen kontrol işareti

Kare dalga giriş için nöro-PID ile kontrol edilen sistem cevabı aşağıda gösterilmektedir.

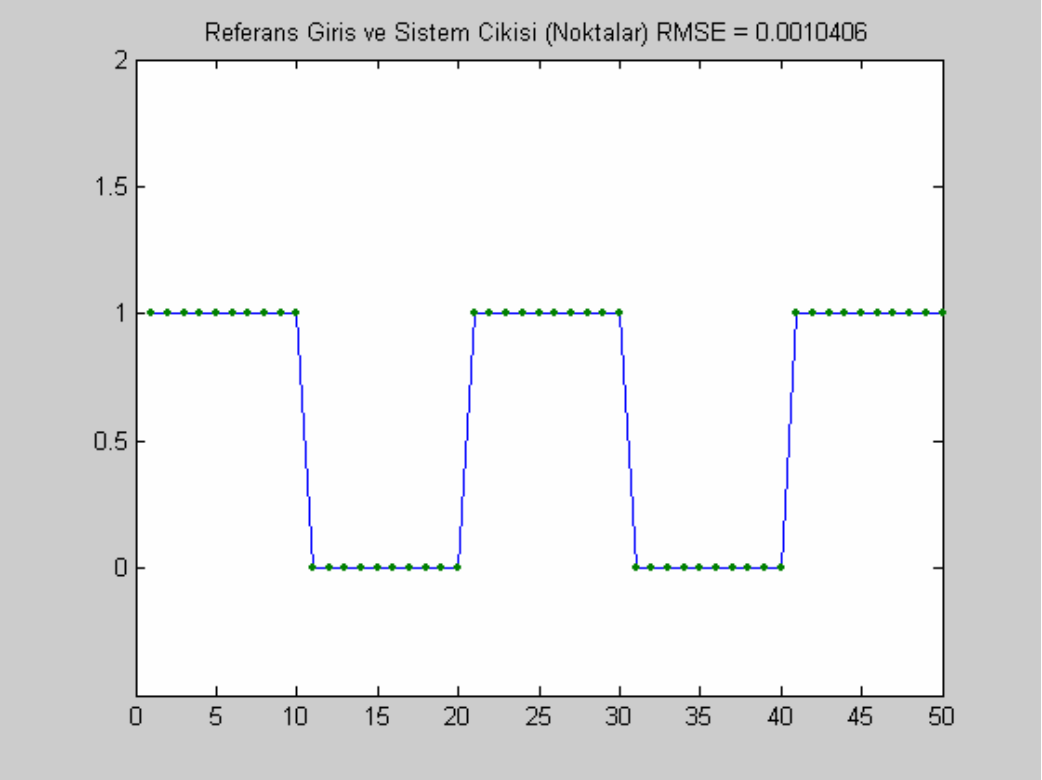
```
>> ileribeslemelikontrolet(karedalga,wga,wac,1000)

Kontrolörün egitiminde kullanilacak gizli hücre sayisini giriniz > 5

Ogrenme katsayisini giriniz
Ogrenme katsayisi [0 1] araliginda olabilir
[0.1 0.3] araligi önerilmektedir > 0.3

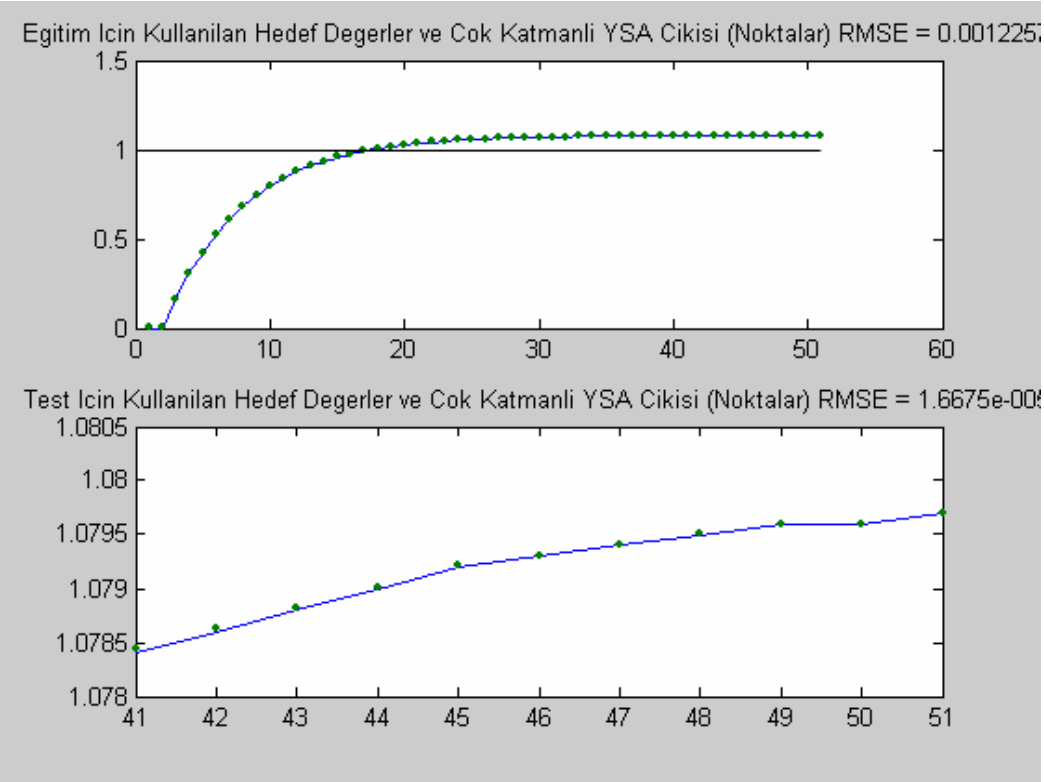
Momentum katsayisini giriniz
Momentum katsayisi [0 1] araliginda olabilir
[0.1 0.2] araligi önerilmektedir > 0.1

Toplam 1000 iterasyon yapilmasi yada 1e-008 karesel hataya ulasilmasi planlanmaktadir
Nöro-PID parametreleri arastiriliyor : 100.iterasyonda yapilan karesel hata 0.00024955 dir
Nöro-PID parametreleri arastiriliyor : 200.iterasyonda yapilan karesel hata 4.4539e-005 dir
Nöro-PID parametreleri arastiriliyor : 300.iterasyonda yapilan karesel hata 4.3107e-006 dir
Nöro-PID parametreleri arastiriliyor : 400.iterasyonda yapilan karesel hata 1.2073e-006 dir
Nöro-PID parametreleri arastiriliyor : 500.iterasyonda yapilan karesel hata 1.0861e-006 dir
Nöro-PID parametreleri arastiriliyor : 600.iterasyonda yapilan karesel hata 1.0829e-006 dir
Nöro-PID parametreleri arastiriliyor : 700.iterasyonda yapilan karesel hata 1.0828e-006 dir
Nöro-PID parametreleri arastiriliyor : 800.iterasyonda yapilan karesel hata 1.0828e-006 dir
Nöro-PID parametreleri arastiriliyor : 900.iterasyonda yapilan karesel hata 1.0828e-006 dir
Nöro-PID parametreleri arastiriliyor : 1000.iterasyonda yapilan karesel hata 1.0828e-006 dir
```



Şekil 6.4.7 : Öz-uyarlamalı nöro-PID ile kontrol edilen sistemin kare dalga cevabı

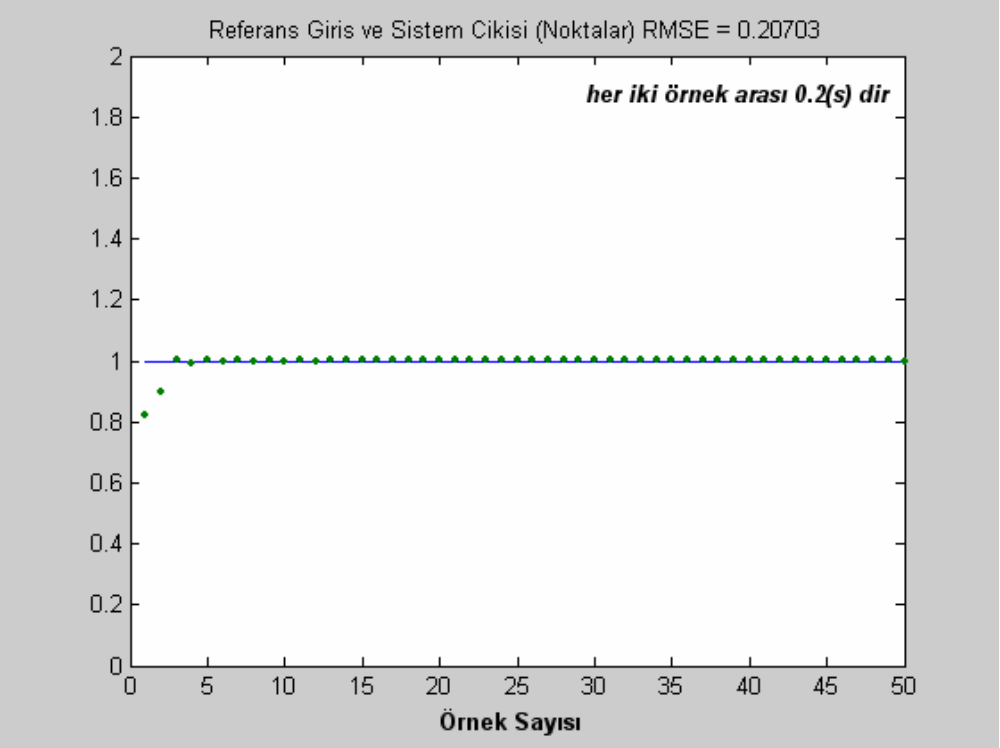
Şekil 6.4.5 ve şekil 6.4.7'de görüldüğü gibi öz-uyarlamalı nöro-PID ile kontrol edilen sistemin sürekli hal hatası sıfırlanabilmiştir. Fakat sistemin geçici hal davranışının nasıl olduğu yukardaki şekillerden anlaşılamamaktadır. Çünkü yapay sinir ağına sistemin kararlı hal davranışına ilişkin değerleri tanıtılmıştır. Nöro-PID kontrolörünün performansını değerlendirebilmek için kontrol edilen sistemin geçici hal davranışınında incelenmesi gerekmektedir. Bu amaçla şekil 6.4.8'de görüldüğü gibi sistemin birim basamak giriş için gösterdiği geçici hal davranışı modellenmiştir.



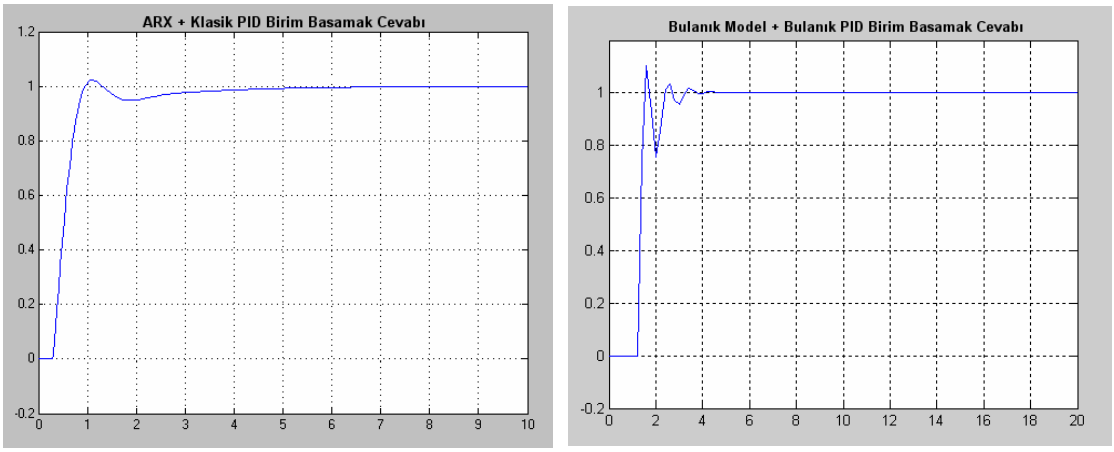
Şekil 6.4.8 : Sistemin basamak giriş için geçici hal davranışına ilişkin değerleri ve YSA modeli çıkışı

Bundan sonra geçici hal davranışı modellenen sistem nöro-PID ile kontrol edilecek ve böylece nöro-PID ile daha önce kullanılan klasik PID ve bulanık-PID arasında zaman tanım bölgesi kriterleri açısından bir karşılaştırma yapılabilecektir.

Nöro-PID ile kontrol edilen sistemin birim basamak cevabı şekil 6.4.9'da gösterilmektedir. Belirtilmelidir ki, yatay eksen zaman değil, örnek sayısıdır. Sistemin örnekleme zamanı 0.2[s] olduğuna göre, nöro-PID ile kontrol edilen sistem 13 örnek sonunda, diğer bir ifade ile 2.6[s]'de referans değere ulaşmaktadır. Şekil 6.4.10'da hatırlatma amacı ile klasik PID ve bulanık-PID ile kontrol edilen sistemin birim basamak cevapları tekrar gösterilmiştir.



Şekil 6.4.9 : Nöro-PID ile kontrol edilen geçici hal davrnaışı modelinden alınan değerler



Şekil 6.4.10 : Klasik PID ile kontrol edilen model(solda) bulanık-PID ile kontrol edilen model (sağda)

Üç farklı modelleme yaklaşımı için elde edilen sonuçlar tablo 6.4.1'de özetlenmektedir.

Tablo 6.4.1 : Farklı modelleme yaklaşımları için anlınan sonuçlar ve yorumlar

Üç Farklı Modelleme Yöntemi İçin Elde Edilen Sonuçlar		
Modelleme Yöntemi	**Karesel Ortalama Hata**	**Yorum**
Lineer ARX Modeli	Eğitim verileri için : 1.7945×10^{-2} Test verileri için : 1.7807×10^{-2}	Modelleme hatasını 10^{-2} mertebesine düşürebilmek için 11.derece bir model kurulmuştur. Non-lineer sistemlerin modellenmesinde yetersiz kalacaktır.
Bulanık Model *u(k-2) ve y(k-1) için*	Eğitim verileri için : 3.5967×10^{-5} Test verileri için : 4.1828×10^{-5}	Kullanılması gereken önceki değerler biliniyorsa, en iyi modelleme yaklaşımı gibi gözükmektedir. Fakat önceki değerlerden hangilerinin kullanılması gerektiği bilinmiyorsa, model kurmak büyük problem yaratmaktadır.
Çok Katmanlı İleribeslemeli Yapay Sinir Ağı Modeli *u(k) ve y(k) için*	Eğitim verileri için : 4.4059×10^{-3} Test verileri için : 3.5425×10^{-3}	Yapay sinir ağına önceki değerler değil, şimdiki değerler tanıtılmıştır. Yapılan hata kabul edilebilir düzeydedir.
Çok Katmanlı İleribeslemeli Yapay Sinir Ağı Modeli *u(k-2) ve y(k-1) için*	Eğitim verileri için : 6.1057×10^{-4} Test verileri için : 4.0247×10^{-4}	Önceki değerlerden hangilerini kullanılması gerktiği biliniyorsa hata küçültülebilmektedir. Fakat sonuç bulanık model kadar iyi değildir.
Geribeslemeli Elman-tipi Yapay Sinir Ağı Modeli *u(k) ve y(k) için*	Eğitim verileri için : 4.7729×10^{-2} Test verileri için : 4.4441×10^{-2}	Elman-tipi yapay sinir ağı yaygın bir şekilde kullanılan iyi performanslı bir ağ olmakla birlikte, geribeslemeler gizli katmandan yapıldığından bu problem için uygun değildir.

Hatırlanacağı gibi genelleştirilmiş delta kuralına dayanan nöro-PID, algoritması gereği, ağırlık değerlerini karesel hatayı en azlayacak şekilde güncellemektedir. Bu sebeple klasik PID kontrolörünün kazanç değerleri ve bulanık-PID kontrolörünün ölçeklendirme çarpanları karesel hatayı en azlayacak şekilde belirlenmiştir. Böylece elde edilen zaman tanım bölgesi kriterleri arasında doğru ve anlamlı bir karşılaştırma yapılabilecektir. Üç farklı kontrolör yapısı için elde edilen sonuçlar tablo 6.4.2'de özetlenmektedir.

Tablo 6.4.2 : Farklı kontrolör yapıları için alınan sonuçlar

Üç Farklı Kontrolör Yapısı İçin Alınan Sonuçlar			
Kontrol Yöntemi	**Klasik PID**	**Bulanık-PID**	**Nöro-PID**
Yerleşme Zamanı (±5 band)	2.1[s]	3.1[s]	0.5[s]
En Büyük Değer	1.04	1.1	1.01
Yüzde Aşım	%4	%10	%1

7. SONUÇLAR

Son kısımda giriş-çıkış değerleri bilinen bir sistem için üç farklı yöntem kullanılarak modeller oluşturulmuştur. Elde edilen modellerin, bu yöntemlere dayanan kontrolör yapıları ile kontrolü sağlanmıştır.

Sistem için lineer bir ARX modeli elde edilmesinde ve transfer fonksyonunun oluşturulmasında herhangi bir zorluk yaşanmamıştır. Sistem çok yüksek mertebeden olsa dahi, model elde etmemizi önleyecek bir kısıt yoktur. Fakat modelleme hatasını kabul edilebilir seviyelere düşürmek için çok yüksek dereceli transfer fonksyonlarına ihtiyaç duyulmaktadır. Bu durum çevrimiçi uygulamalarda problem yaratabilir. Ayrıca modellenen sistemin lineer olmaması durumunda, bu yöntem ile elde edilen modeller, süreçleri temsil etmekte yetersiz kalacaktır.

Üç farklı yaklaşım incelendiğinde, en az modelleme hatasına sebep olan yöntemin bulanık modelleme olduğu görülmektedir. Fakat tüm sistemler için bulanık bir modelin elde edilebileceğini söylemek zordur. Düşük mertebeli sistemler için, kullanılması gereken önceki değerlerin bulunmasını sağlayacak araçlar olmasına rağmen, yüksek mertebeli sistemlerde bu değerlerin deneme-yanılma metodu ile bulunması gerekmektedir. Bu durum bulanık modelleme yönteminin en önemli dezavantajıdır.

Son olarak sistem için yapay sinir ağı modeli elde edilmiştir. Yapılan modelleme hatası, bulanık model ile aynı girişler kullanıldığında 10^{-4} mertebesindedir. Bu değer bulanık modelde 10^{-5} mertebesindeydi. Fakat önceki değerlerden hangilerinin kullanılması gerektiği bilinmiyorsa, yapay sinir ağı modeli ile yapılan hata 10^{-3} mertebesindedir. Oysaki bu durumda sistem için bulanık bir model elde edilememektedir.

Uygulamanın niteliğine bağlı olarak bu üç yönteminde avantajlı yada dezavantajlı olduğu durumlar mevcuttur. Eğer uygulama parametrik bir model gerektiriyorsa, bu durumda bulanık yöntem ve yapay sinir ağı yaklaşımları kullanılamayacaktır. Eğer lineer olmayan bir süreç modellenmek isteniyorsa, bu durumda lineer ARX modeli, çalışma noktası dışında, sistemi yeteri kadar temsil edemeyecektir. Eğer modelin

parametrik olma zorunluluğu yoksa, ve sistem mertebesi biliniyorsa, en iyi sonuç bulanık model ile elde edilebilir. Fakat sistem hakkında hiçbir önbilgi yoksa, bu durumda yapay sinir ağları iyi bir çözüm olacaktır. Giriş-çıkış değerlerinden başka hiçbir bilgiye ihtiyaç duyulmadan, yapay sinir ağı modeli oluşturulabilmektedir. Üstelik 2.mertebeden bir transfer fonksyonu ile temsil edilebilecek bir kimyasal süreci yada 250.mertebeden bir transfer fonksyonu ile temsil edilebilecek gürültüyü modelleyebilmek için kullanılacak algoritma aynı olacaktır. Bu sayede algoritma bir kez hazırlandıktan sonra, ilgilenilen sistemin değişmesi halinde, yapılması gereken tek işlem, ağın yeni sistem verileri ile eğitilmesi olacaktır.

KAYNAKLAR

[1] **Öztemel, E.**, 2003. *Yapay Sinir Ağları*, Papatya Yayıncılık, İstanbul.

[2] http://ai-depot.com/main.html

[3] http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html

[4] http://www.faqs.org/faqs/ai-faq/neural-nets/part1/section-10.html

[5] **Kansu, R.**, 1979. Kusursuz Uçuş Makineleri, *Bilim ve Teknik,* **136,** 21-23.

[6] **Gür, H.,** 1985. Biyonik, Doğayı Kopya Etmektir, *Bilim ve Teknik,* **208,** 19-20.

[7] **Tsoukalas, L.H. and Uhring, R.E.**, 1997. *Fuzzy and Neural Approaches in Engineering*, John Wiley & Sons, Inc.

[8] **Gupta, M.M. and Jin L. and Homma, N.**, 2003. *Static and Dynamic Neural Networks - From Fundamentals to Advanced Theory*, John Wiley & Sons, Inc.

[9] **Norgaard, M. and Ravn, O. and Poulsen, N.K. and Hansen L.K**., 2000. *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag, London.

EK.A Perceptron ve ADALINE Modellerine İlişkin Algoritma

Perceptron

```
function y = rosenblatt(giris,hedef,defa,hata)

tic
if nargin == 3, hata = 10^(-12);
elseif nargin == 2, hata = 10^(-12); defa = 1000; end

[n m] = size(giris);
[p r] = size(hedef);
if n ~= p
    disp('giris ve hedef vektorlerinin boyutlari tutarsizdir')
    return
elseif r ~= 1
    disp('hedef vektorunun boyutu yanlis girilmistir')
    return
end

w = randn(1,m);
esikvektor = ones(n,1);
fi = 1;
e = inf;
sayac = 0;
while e'*e >= hata
cikis = (thresh(fi.*esikvektor+(w*giris')'));
e = cikis - hedef;
kareselhata = e'*e;
plot(sayac,kareselhata)
hold on
for k=1:p
        if e(k) == 0, w = w;, fi = fi;
            elseif e(k) > 0, w = w - (giris(k,:));, fi = fi - 1.75;
            elseif e(k) < 0, w = w + (giris(k,:));, fi = fi + 1.75;
        end

end
sayac = sayac + 1;
if sayac >= defa
    break
end
end

t = toc;

t1 = sprintf('\nYapilan iterasyon sayisi %.10g dir\n',sayac);
disp(t1)
t2 = sprintf('Nöronun egitilmesi sonucunda bulunan çikis vektörü\n');
disp(t2)
t3 = sprintf('%.10g\n',cikis);
disp(t3)
t4 = sprintf('Nöronun egitilmesi sonucunda bulunan agirlik vektörü\n');
disp(t4)
t5 = sprintf('%.10g\n',w);
disp(t5)
t6 = sprintf('Böylelikle yapilan karesel hata %.10e dir\n',e'*e);
disp(t6)
t7 = sprintf('Bu islem toplam %.3g saniye sürmüstür\n',t);
disp(t7)
```

ADALINE

```
function y = adaline(giris,hedef,eta,defa,hata)
tic
if nargin == 4, hata = 10^(-12);
elseif nargin == 3, hata = 10^(-12); defa = 10^4;
elseif nargin == 2, hata = 10^(-12); defa = 10^4; eta = 0.1; end

[n m] = size(giris);
[p r] = size(hedef);
if n ~= p
    disp('giris ve hedef vektorlerinin boyutlari tutarsizdir')
    break
elseif r ~= 1
    disp('hedef vektorunun boyutu yanlis girilmistir')
    return
end

w = randn(1,m);
fi = ones(n,1);
e = inf;
sayac = 0;

while e'*e >= hata
cikis = ( 1./ (1 + exp(fi+(w*giris')')));
e = cikis - hedef;
deltaw = eta*e'*giris;
w = w + deltaw;
fi = fi + eta.*e;
sayac = sayac + 1;
if sayac >= defa
    break
end
end
t = toc;

t1 = sprintf('\nYapilan iterasyon sayisi %.10g dir\n',sayac);
disp(t1)
t2 = sprintf('Nöronun egitilmesi sonucunda bulunan çikis vektörü\n');
disp(t2)
t3 = sprintf('%.10g\n',cikis);
disp(t3)
t4 = sprintf('Nöronun egitilmesi sonucunda bulunan agirlik vektörü\n');
disp(t4)
t5 = sprintf('%.10g\n',w);
disp(t5)
t6 = sprintf('Böylelikle yapilan karesel hata %.10e dir\n',e'*e);
disp(t6)
t7 = sprintf('Bu islem toplam %.3g saniye sürmüstür\n',t);
disp(t7)
```

EK.B Çok Katmanlı Yapay Sinir Ağına İlişkin Algoritma

```
function y = ileribeslemeliegit(giris,hedef,defa,hata)

tic
if nargin == 3, hata = 10^(-8);
elseif nargin == 2, hata = 10^(-8); defa = 5000; end

[n m] = size(giris);
[p r] = size(hedef);

%----Giris ve Hedef vektorleri yanlis girilirse program dursun-------------
if n ~= p
    disp('Giris ve hedef vektorlerinin boyutlari tutarsizdir')
    return
end
%----Kod tek cikisli sistemler icin kullanilabilir-------------------------
if r ~= 1
    disp('Bu program sadece tek cikisli sistemler icin kullanilabilir')
    return
end
%----Giris ve hedef vektorleri standartlastirilsin-------------------------
girisortalama = mean(giris);
girisstandartsapma = std(giris);
giris = (giris(:,:) - girisortalama(:,1)) / girisstandartsapma(:,1);
hedef = hedef';
hedefortalama = mean(hedef);
hedefstandartsapma = std(hedef);
hedef = (hedef(:,:) - hedefortalama(:,1)) / hedefstandartsapma(:,1);
hedef = hedef';
%-----Atamalar yapilsin-----------------------------------------------------
e = inf;
sayac = 0;

%----Esik vektor ekleyelim-------------------------------------------------
bias = ones(n,1);
giris = [giris bias];
[n m] = size(giris);
[p r] = size(hedef);
%----Egitim ve test verileri belirlensin-----------------------------------
bl = ['\nGiris ve hedef vektörleri ' num2str(n) ' tane veri kumesi icermektedir\nBunlarin % kaci egitim
        ' Onerilen deger %80 ve yukarisi\n0 - 100 arasinda bir rakam girmeniz beklenmektedir > '];
yuzde = input(bl);
if yuzde<0 | yuzde>100
    disp('Girilen deger yanlistir')
    return
end
egitimverisisayisi = round(n*(yuzde/100));   %yuzde*toplam veri sayisini yuvarlatalim
egitimgiris = giris(1:egitimverisisayisi,:); %egitimde kullanilacak giris vektorunu bulalim
egitimhedef = hedef(1:egitimverisisayisi,:); %egitimde kullanilacak hedef vektorunu bulalim
if egitimverisisayisi == 0      %yuzde deger cok kucuk girilirse en yakin tam sayiya
    egitimgiris = giris(1:n,:); %yuvarlatildiginda sifir olma ihtimali var
    egitimhedef = hedef(1:n,:);
    egitimverisisayisi = n;
    disp('Girdiginiz egitim verisi yuzdesi cok kucuk oldugundan')
    disp('tum veri kumesi, egitimde kullanilacaktir')
end
testgiris = giris(egitimverisisayisi+1:n,:); %test vektorunu bulalim
testhedef = hedef(egitimverisisayisi+1:n,:); %test vektorunu bulalim
if egitimverisisayisi == n             %yuzde deger cok buyuk girilirse, test verisinin
    testgiris = giris(n,:);            %sifir olma ihtimali var. bu durumda son satir
    testhedef = hedef(n,:);            %test icin kullanilacak
end
```

```
%----Gizli katmanda kullanilacak hucre sayisi belirlensin-----------------
b2 = ['\nSistem ' num2str(m-1) '-girisli ' num2str(r) '-cikisli oldugundan\n'...
      'giris katmaninda ' num2str(m-1) ' tane ve cikis katmaninda ' num2str(r) ' tane hucre bulunacaktir'...
      '\nGizli katmanda kac tane gizli hucre bulunsun? > '];
gizlihucresayisi = input(b2);
if gizlihucresayisi < 1
    disp('Gizli hucre sayisi icin uygun olmayan bir deger girdiniz')
    return
end
%----Kullanici, ogrenme katsayisini girsin--------------------------------
b3 = '\nOgrenme katsayisini giriniz\nOgrenme katsayisi [0 1] araliginda olabilir\n[0.2 0.4] araligi önerilmektedir > ';
eta = input(b3);
if eta<0 | eta>1
    disp('Girilen deger yanlistir')
    return
end
%----Kullanici momentum katsayisini girsin---------------------------------
b4 = '\nMomentum katsayisini giriniz\nMomentum katsayisi [0 1] araliginda olabilir\n[0.6 0.8] araligi önerilmektedir > ';
mom = input(b4);
if mom<0 | mom>1
    disp('Girilen deger yanlistir')
    return
end
%----Baslangic icin agirliklar rastgele belirlensin------------------------
wga = (randn(m,gizlihucresayisi) - 0.5)/10; %giris(g) ve arakatman(a) arasindaki agirliklar wga
wac = (randn(gizlihucresayisi,r) - 0.5)/10; %arakatman(a) ve cikis katmani(c) arasindaki agirliklar wac
%----Diger baslangic degerleri atansin-------------------------------------
deltawga = zeros(m,gizlihucresayisi);  %[mxgizli]
deltawac = zeros(gizlihucresayisi,r);  %[gizlixr]
hepsideltawga = zeros(m,2*gizlihucresayisi);
hepsideltawac = zeros(gizlihucresayisi,2*r);
%----BILGILER OLUSTURULDU--------------------------------------------------



%----ITERASYON BASLIYOR----------------------------------------------------

for sayac = 1:defa

% momentum katsayisi ile carpmak icin agirliklardaki degisimin bir onceki
% degerlerini mommatris de saklayalim : wac
hepsideltawac(:,(r+1):2*r) = deltawac;                    %degisim degerlerini sakliyoruz
mommatrisiwac = hepsideltawac(:,1:r);                     %momentum katsayisi ile carpicagimiz kismi
hepsideltawac = [hepsideltawac(:,r+1:2*r) zeros(gizlihucresayisi,r)];  %matrisi buyutmemek icin öteli

% momentum katsayisi ile carpmak icin agirliklardaki degisimin bir onceki
% degerlerini mommatris de saklayalim : wga
hepsideltawga(:,(gizlihucresayisi+1):2*gizlihucresayisi) = deltawga;    %degisim degerlerini sakliyor
mommatrisiwga = hepsideltawga(:,1:gizlihucresayisi);                    %momentum katsayisi ile carpi
hepsideltawga = [hepsideltawga(:,gizlihucresayisi+1:2*gizlihucresayisi) zeros(m,gizlihucresayisi)]; %
%-------------------------------------------------------------------------


    for j = 1:egitimverisisayisi

        % sirayla inceleyelim
        incelenengiris = egitimgiris(j,:);
        incelenenhedef = egitimhedef(j,:);

        % ara katman ve cikis katmani cikislarini hesaplayalim
        ilericikisga = tanh(incelenengiris*wga);
        ilericikisac = ilericikisga*wac;
        e = ilericikisac - incelenenhedef;  %hatayi tanimlayalim
        % gizli katman - cikis katmani arasindaki agirliklari guncelleyelim
        deltawac = ((eta/10).*e*ilericikisga)' + mom.*mommatrisiwac;
        wac = wac - deltawac;
```

```
        % giris katmani - ara katman arasindaki agirliklari guncelleyelim
        deltawga = (eta.*e'.*wac.*(1-(ilericikisga.^2))'*incelenengiris)' + mom.*mommatrisiwga;
        wga = wga - deltawga;
    end

    % egitim verileri icin yapilan hatayi bulalim
    ilericikisac = (wac'*(tanh(egitimgiris*wga))')';
    e = ilericikisac - egitimhedef;
    % egitim verileri icin yapilan hatayi cizdirelim
    err(sayac) = (sum(e.^2)^0.5);
    figure(1);
    plot(err)
    title(['Karesel Hatanin Iterasyon Sayisina Bagli Degisimi']);
    xlabel('Iterasyon Sayisi'); ylabel('Karesel Hata');

sayac = sayac + 1;
if e'*e <= hata
    break
end
end


% egitim verileri icin standartlastirma islemini tersine cevirelim
eh = (egitimhedef* hedefstandartsapma(:,1)) + hedefortalama(:,1);
ec = (ilericikisac* hedefstandartsapma(:,1)) + hedefortalama(:,1);
egitimhata = ec - eh;
egitimkareselhata = ((egitimhata'*egitimhata)^0.5);



% test verileri icin yapilan hatayi hesaplayalim
testcikisac = (wac'*(tanh(testgiris*wga))')';
% test verileri icin yapilan hatayi hesaplayalim
testcikisac = (wac'*(tanh(testgiris*wga))')';
% test verileri icin standartlastirma islemini tersine cevirelim
th = (testhedef* hedefstandartsapma(:,1)) + hedefortalama(:,1);
tc = (testcikisac* hedefstandartsapma(:,1)) + hedefortalama(:,1);
testhata = tc - th;
testkareselhata = ((testhata'*testhata)^0.5);

%cikisbul fonksyonunda kullanilacak bilgileri sakla
ortalama = hedefortalama;
standartsapma = hedefstandartsapma;

save agirliklar wga wac ortalama standartsapma

t = toc;



t1 = sprintf('\nYapilan iterasyon sayisi %.10g dir\n',sayac-1);
disp(t1)
t2 = sprintf('Egitim amaci ile kullanilan hedef, cikis ve hata matrisleri :\n');
disp(t2)
disp([eh ec egitimhata])
t3 = sprintf('Egitim asamasinda yapilan karesel hata : \n');
disp(t3)
disp([egitimkareselhata])
t4 = sprintf('Test amaci ile kullanilan hedef, cikis ve hata matrisleri :\n');
disp(t4)
disp([th tc testhata])
t5 = sprintf('Test asamasinda yapilan karesel hata : \n');
disp(t5)
disp([testkareselhata])
t6 = sprintf('Bu islem toplam %.3g saniye sürmüstür\n',t);
disp(t6)
```

ÖZGEÇMİŞ

1980 yılında Gaziantep'te doğdum. 1998 yılında Gaziantep Anadolu Lisesinden mezun oldum. 1999 – 2000 yılları arasında İstanbul Teknik Üniversitesi Yabancı Diller Yüksekokulu İngilizce Bölümünü, 2000 – 2004 yılları arasında İstanbul Teknik Üniversitesi Elektrik Mühendisliği Bölümünü bitirdim. Yüksek lisans eğitimimi 2004 – 2006 yılları arasında İstanbul Teknik Üniversitesi Kontrol ve Otomasyon Mühendisliği Bölümünde tamamladım.