KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MELEZ TEK BİR İŞLEM BİRİMLİ YAPAY SİNİR AĞI HÜCRESİNİN OTOMATİK ÖĞRENME VERİTABANLARINA UYGULANMASI

YÜKSEK LİSANS

Teknik Öğretmen Ali ÖZDEMİR

Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışman: Yard. Doç. Dr. Melih İNAL

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MELEZ TEK BİR İŞLEM BİRİMLİ YAPAY SİNİR AĞI HÜCRESİNİN OTOMATİK ÖĞRENME VERİTABANLARINA UYGULANMASI

YÜKSEK LİSANS

Teknik Öğretmen Ali ÖZDEMİR

Tezin Enstitüye Verildiği Tarih: 25 Aralık 2006

Tezin Savunulduğu Tarih: 5 Şubat 2007

Tez Danışması	Üye	Üye
Yrd.Doc.Dr.Melih İNAL	Prof.Dr.Yılmaz ÇAMURCU	Yrd.Doc.Dr.Mehmet YILDIRIM
(.....)	(.....)	(.....)

KOCAELİ, 2006

ÖNSÖZ ve TEŞEKKÜR

Günümüz dünyasında hız çok önem arz etmektedir. Yapılan her işin mümkün olan en kısa zamanda ve doğru olarak sonuçlandırılması beklenmektedir. Bu durum bilişim sektöründe de kendini göstermiştir. Dünyamızda her gün milyonlarca bilgi depolanmakta; bu veriler üzerinde analizler ve istatistikler yapılmaktadır. Bu analiz ve istatistiklere göre kararlar alınmaktadır. Dolayısı ile yapılan bu işlemlerin en kısa zamanda doğru olarak sonuçlanması büyük önem kazanmıştır. Bilim insanları bu işlemleri yapan yeni fikirler ortaya atmakta ve bunun üzerine çalışmalar yapmaktadırlar. Bu çalışmaların sonucunda ise yeni yaklaşımlar ortaya çıkmaktadır. Bu tez çalışmasında verileri çok kısa sürede ve doğru olarak sınıflandıran birbirinden farklı iki işlem birimi modelinin melez bir biçimde kullanımı amaçlanmıştır. Amaçlanan yöntem otomatik öğrenme veritabanlarına uygulanarak, sistemin sınıflama başarımı irdelenmiştir.

Bu tez çalışmasında beni fikirleri ile yönlendiren, teşvik eden ve desteğini her zaman yanımda hissettiğim danışmanım Sn. Yard. Doç. Dr. Melih İNAL'a teşekkür ederim. Ayrıca hayatım boyunca beni destekleyen annem Sultan ÖZDEMİR'e, ağabeyim Yüksel ÖZDEMİR'e ve ablam Seher ÖZDEMİR'e sonsuz minnet duygularımı sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER

x	: giriş vektörü
w	: ağırlık vektörü
v	: işlem biriminin net girişi
y	: çıkış vektörü
f	: fonksiyon
n	: eğim
d	: gecikme
d(t)	: istenilen çıkış değeri
o	: hedef değeri
δ	: hata faktörü
η	: öğrenme katsayısı
α	: momentum katsayısı
Δ	: fark
b	: eşik değeri
t	: ağırlık vektörü
N	: giriş vektöründeki elaman sayısı
H	: gizli katmandaki işlem birimi sayısı
M	: çıkış katmandaki işlem birimi sayısı

Alt indisler

th	: eşik aktivasyon fonksiyonu
t	: çevrilmiş
m	: değiştirilmiş
r	: giriş katmanı düğümleri
i	: gizli katmandaki işlem birimleri
j	: çıkış katmandaki işlem birimleri

Kısaltmalar

ÖVN	: Öğrenme Vektör Nicemlemesi
ENKK	: En Yakın K Komşuluğu
YSA	: Yapay Sinir Ağları
Adaline	: Uyarlamalı Doğrusal İşlem Birimi
Madaline	: Çoklu Uyarlamalı Doğrusal İşlem Birimi
İBYSA	: İleri Beslemeli Yapay Sinir Ağları
ÇKA	: Çok Katmanlı Ağ
QP	: Hızlı Yansıtma (Quikprop)
DBD	: Delta-Bar-Delta
EDBD	: Genişletilmiş Delta-Bar-Delta (Extended Delta-Bar-Delta)
GA	: Genetik Algoritmalar
LEF	: Sözlüksel Değerlendirme Fonksiyonu (Lexicographic Evaluation Function)

AQ : Keskinlik Kalitesi (Acuity Quality)
OC : Düzenli Sınıflandırma (Ordered Classification)
IB3 : Örnek Tabanlı (Instance Based)
CI : Yapı Atma (Constructive Induction
CM–2 : Bağlantı Makine (Connection Machine)

MELEZ TEK BİR İŞLEM BİRİMLİ YAPAY SİNİR AĞI HÜCRESİNİN OTOMATİK ÖĞRENME VERİ TABANLARINA UYGULANMASI

Ali ÖZDEMİR

Anahtar Kelimeler: Otomatik Öğrenme; Çevrilmiş Çarpımsal İşlem Birimi; McCulloch-Pitts İşlem Birimi Modeli

Özet: Teknolojinin gelişmesi ile farklı alanlarda oluşturulan verilerin depolanması yanında bu verilerin analizi de büyük önem kazanmıştır. Verilerin işlenmesinde farklı otomatik öğrenme yöntemleri kullanılmaktadır. Bu tez çalışmasında yeni bir ikili sınıflama yöntemi ile bazı otomatik öğrenme veri tabanlarının sınıflandırılması amaçlanmıştır. Sınıflandırma için N-bit değer eşitliği temelli bir tek yapay sinir işlem birimi kullanılmıştır. N-bit değer eşitliği probleminde amaç; bitleri 1 olanların sayılarının tek ya da çift olması durumuna göre sonucun sırasıyla 1 ya da 0 olduğunu bulmaktır. Bu problemi çözmek için kullanılan çevrilmiş çarpımsal işlem birimi modeli önerilen modellerin arasında en iyisidir. Bu model McCulloch-Pitts işlem birimi modelinden esinlenerek oluşturulmuştur. Bu tez çalışmasında sınıflandırılacak probleme göre, bu iki işlem birimi modeli kullanarak melez bir işlem birimi modelinin oluşturulması amaçlanmıştır. Kullanılan çevrilmiş çarpımsal işlem birimi modelindeki parametre değerleri yine sınıflandırma problemine göre farklı seçilmiştir. Sonuç olarak, çeşitli otomatik öğrenme veri tabanları için ikili sınıflandırma işlemi yapılmıştır.

APPLICATION OF A SINGLE HYBRID ARTIFICIAL NEURON TO MACHINE LEARNING DATABASES

Ali ÖZDEMİR

Keywords: Machine Learning; Translated Multiplicative Neuron; McCulloch-Pitts Neuron Model

Abstract: Analyzing and storing data that are consisted from different domain become important thing with development of technology. Different machine learning methods are used to process these kinds of data. In this thesis, a new binary classification method is proposed for classification of some machine learning algorithms. Only one artificial neuron based on N-bit parity rule is used for the classification. The aim of N-bit parity problem is specified the result as 1 or 0, according to the number of 1's bits that is odd or even. The translated multiplicative neuron used to solve this problem is the best model which is recommended among the other studies. This model has been constructed by means of inspiring from the McCulloch-Pitts neuron model. In this thesis, it is proposed to constitute a hybrid model of these two models according to the problem that is classified. The parameter values of translated multiplicative neuron model are chosen different according to the classification problem. Finally, binary classification is realized for different machine learning databases.

1. GİRİŞ

Öğrenme, bir deneyim ya da denemenin sonucunda davranışlarda meydana gelen sürdürülebilir ve kalıcı değişikliklerdir. Teknolojideki öğrenme ise elde edilen veriler üzerinde anlamlı analizler yapmaktadır. Buradaki analiz aslında verinin istenen sonuca göre doğru olarak sınıflandırılmasıdır.

Günümüzde verinin depolanmasının önemi artmıştır. Çünkü günlük yaşamda milyonlarca bilgi etrafta dolaşmaktadır. Bu verilerden daha sonra yararlanmak için ise bunların depolanması gerekmektedir. Örneğin bir hastanenin hastalarının oluşturduğu veriler, bir okuldaki öğrencilerin oluşturduğu veriler güvenlik kameralarının çektiği görüntüler vb. Bu verilerin depolanması ne kadar önemli ise bu veriler üzerinde analiz yapmak da o kadar önem kazanmıştır. Bilim insanları bu verileri sınıflandırmak için birçok farklı yöntemler üzerinde çalışmaktadır. Bu çalışmaların sonucunda ortaya çıkan birçok yöntem günümüzde kullanılmaktadır. Yapay sinir ağları, genetik algoritmalar, bulanık mantık, uzman sistemler vb. konularda sınıflandırma yapan algoritmalar geliştirilmektedir. Buradaki çalışmanın amacı ise, tek bir yapay sinir işlem birimi kullanarak, verileri mümkün olan en kısa sürede doğru olarak sınıflandırılmasını sağlamaktır. Bu tez çalışmasını oluşturan bölümler aşağıdaki gibi belirlenmiştir:

Bölüm 2'de otomatik öğrenmenin tanımı yapılarak bu alanda kullanılan sınıflandırma yöntemleri tanıtılmıştır.

Bölüm 3'te Yapay Sinir Ağları açıklanarak, tarihsel gelişimi anlatılmıştır. Bu ağların içinde kullanılan işlem birimi modelleri ve bu işlem birimlerinde kullanılan aktivasyon fonksiyonları tanıtılmıştır.

Bölüm 4'te Yapay Sinir Ağlarında kullanılan çarpımsal işlem birimi modeli açıklanmıştır. Bu işlem birimi modeli üzerinde yapılan değişiklik ile oluşturulan

çevrilmiş çarpımsal işlem birimi (π_t) modeli ile bu modelin oluşturulmasında temel alınan McCulloch-Pitts toplamsal işlem birimi modelleri anlatılmıştır. Ayrıca π_t işlem birimi modelinde farklı değer ve işarette seçilen parametreler ile McCulloch-Pitts işlem birimi modelinin melez kullanımı amaçlanmıştır. π_t işlem biriminde, farklı değer ve işaretlerle seçilen parametre değişikliği nedeniyle amaçlanan bu yönteme değiştirilmiş çarpımsal işlem birimi (π_m) modeli adı verilmiştir.

Bölüm 5'te π_m ve McCulloch Pitts işlem birimi modellerinin birlikte ya da π_m işlem birimi modeli tek başına, farklı özelliklerdeki otomatik öğrenme veri tabanlarına uygulanmasıyla, önerilen modelin yüzdelik sınıflama başarımları açıklanmıştır.

Bölüm 6 ise amaçlanan yöntemin sözü edilen veritabanlarına uygulanması sonucu elde edilen sonuçlar yorumlanarak ileriki çalışmalar için önerilerde bulunulmuştur.

2. OTOMATİK ÖĞRENME

Bu bölüm içinde otomatik öğrenme işlemi açıklanarak, kullanılan yöntemler açıklanmaktadır.

2.1. Otomatik Öğrenme Nedir?

Bilişim teknolojisindeki gelişmeler sayesinde artık çok büyük miktarlarda veriler saklanabilmektedir. Örneğin süpermarket kasalarından, kredi kartı cihazlarından her an milyonlarca veri, verilerin saklandığı merkezlere ulaşmaktadır. Bir güvenlik kamerasından, bir iris tanıma sisteminden, borsadaki işlemlerden birçok veri elde edilmekte ve analiz için beklemektedir. Bu işlemlerin her birinin analizden farklı sonuçlara ulaşılabilmektedir. Bir süpermarket işletmecisi hangi tür ürünlerin bir arada satıldığını, kredi kartı tanıma sistemi kullanılan kredi kartının sahibi tarafından kullanılıp kullanılmadığı, bir güvenlik kamerası olağan dışı bir şey olup olmadığını, bir iris tanıma sistemi verinin kime ait olduğunu ya da bir borsa analisti hisse senedinin yarınki değerini öğrenmek isteyebilir. Bu durumda istenilenleri bulmak için; kayıtlı veriyi kullanarak yeni verinin işlenmesi gerekmektedir. Burada çok miktarda bulunan verinin elle işlenmesi ve analizinin yapılması mümkün değildir. Bundan dolayı bu tür problemleri çözmek için otomatik öğrenme yöntemleri geliştirilmiş ve geliştirilmeye devam edilmektedir. Otomatik öğrenme yöntemleri önceki verileri kullanarak veriye en uygun modeli bulmaya çalışır. Yeni gelen verileri de bu modele göre analiz ederler. Büyük miktarda verinin incelenip içinden işe yarayan bilginin elde edilmesi işlemine veri madenciliği de (data mining) denmektedir. Doğal olarak farklı uygulamaların analizinden farklı beklentiler oluşmaktadır. Bu beklentilere göre otomatik öğrenme yöntemleri aşağıda açıklanmıştır [1].

- Sınıflandırma: Mevcut olan sınıflandırılmış verilerden hareket edilerek; yeni bir verinin bu sınıflardan hangisine ait olduğunun bulunmasıdır.

- Eğri Uydurma (Regresyon): Mevcut olan verilerin sınıflandırılması yerine; veriler arasında bir fonksiyon eğrisi bulunmaya çalışır.
- Kümeleme: Buradaki amaç üyelerinin birbirine çok benzediği, ancak özellikleri birbirinden çok farklı olan kümelerin bulunması ve veri tabanındaki kayıtların bu farklı kümlere ayrılmasıdır.
- Özellik seçimi/çıkarımı: Veriye ait birçok özellikten veri kümesinin veya sınıfının değerini belirleyen özelliklerinin hangileri olduğu bilinmeyebilir. Bu durumda tüm özellik kümesinin bir alt kümesi seçilir (özellik seçimi) ya da bu özelliklerin birleşmelerinden yeni özellikler elde edilir (özellik çıkarımı).
- İlişki belirleme: Bir süpermarkette X ürününü alan müşterilerin %80'i Y ürününü de alıyorsa, X ürününü alıp Y ürününü alan müşteriler, Y ürününün potansiyel müşterileridir. Müşterilerin bu tür bilgilerinin bulunduğu veritabanından potansiyel Y müşterilerini bulma işlemine ilişki belirleme denir.

Buradaki otomatik öğrenme yöntemlerinden sınıflandırma kendi içinde çeşitli yöntemlerle yapılabilmektedir.

2.1. Sınıflandırma Yöntemleri

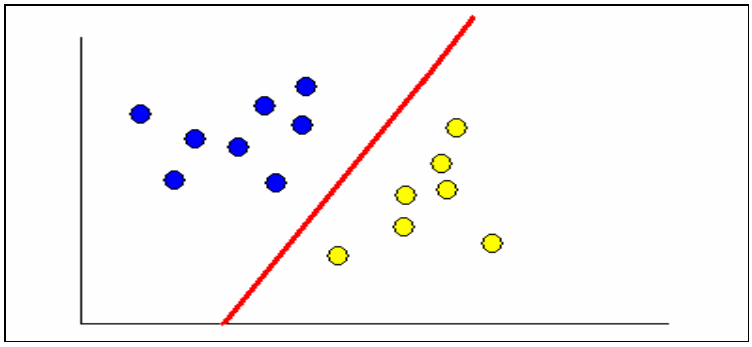
Sınıflandırma için kullanılan başlıca yöntemler aşağıdaki gibidir:

- Destek vektör makineleri (Support vector machines)
- Öğrenme vektör nicemlemesi (Learning vector quantization)
- Karar ağaçları (Decision trees)
- En yakın k komşuluğu (K-nearest neighbor)
- Yapay sinir ağları (Artificial neural networks)

2.1.1. Destek vektör makineleri

Buradaki amaç; sınıflandırılmak için verilmiş verileri ayıran bir doğrunun bulunmasıdır. Bulunan bu doğrunun iki öbek arasında eşit uzaklıkta olması sağlanır [2]. Böylelikle hatayı azaltma olasılığı en yüksek seviyeye ulaşır. Şekil 2.1' de iki sınıfı birbirinden ayırmak için, destek vektör makinenin bulduğu sınır çizgisi

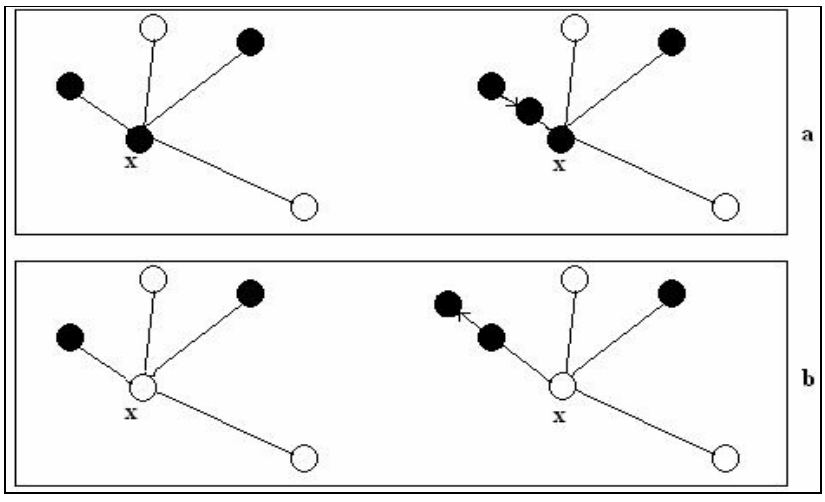
görülmektedir. Eğitim verisiyle sınır çizgisi bulunur. Test verileri ise bu sınır çizgisine göre sınıflandırarak başarım belirlenir.



Şekil 2.1: DVM ile sınıflandırma

2.1.2. Öğrenme vektör nicemlemesi

Öğrenme vektör nicemleme (ÖVK) algoritması nicemlemek istenen verilerle aynı boyutta betimleyici vektörler rasgele seçilir. Daha sonra eğitim setinin her bir örneği için; bu örnek vektöre betimleyici vektörlerden en yakın olan belirlenir. Eğer örnek vektör ile betimleyici vektör aynı sınıfta ise; betimleyici vektör o sınıfı daha iyi temsil etsin diye örnek vektöre yakınlaştırılır. Aksi durumda ise betimleyici vektör örnek vektörden uzaklaştırılır [3].



Şekil 2.2: ÖVN'de ödül (a) ve ceza (b)

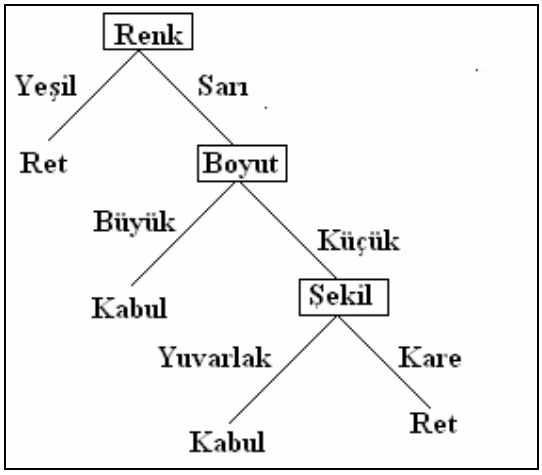
Şekil 2.2'de 4 çeşit betimleyici vektör ve bir tane x örneği gösterilmektedir. Bunlardan ikisi siyah, diğer ikisi ise beyaz sınıfı temsil etmektedir. x örneği ile betimleyici vektörler arsındaki mesafeler hesaplanmıştır. Şekil 2.2 (a)'da x örneğine en yakın betimleyici vektör siyahtır. x örneği de aynı sınıftan olduğu için, betimleyici vektör ödül olarak x örneğine yakınlaştırılır. Şekil 2.2 (b)'de ise en yakın betimleyici vektör ile x örneği aynı sınıfta olmadığı için, betimleyici vektör ceza olarak x örneğinden uzaklaştırılmıştır. Bu işlem örnek vektöre en yakın betimleyici vektör bulunana kadar devam eder.

2.1.3. Karar ağaçları

Karar ağacı karar düğümleri, dallar ve yapraklardan oluşur. Buradaki karar düğümleri gerçekleştirilecek testi belirtir. Bu testin sonucu ağacın veri kaybetmeden dallara ayrılmasına neden olur. Eğer bir dalın ucunda sınıflama işlemi gerçekleşmiyorsa, o dalın ucunda bir karar düğümü oluşur. Eğer bir sınıflama işlemi gerçekleşiyorsa, o dalın sonunda yeni bir yaprak oluşarak sınıflama işlemi devam eder [4, 5]. Tablo 2.1'de verilen eğitim kümesine göre Şekil 2.3'te oluşturulmuş karar ağacı görülmektedir. Buradaki kareler karar düğümlerini, düz çizgilerin üzerindeki değerler dalları, düz çizgilerin ucundaki değerler ise yaprakları temsil etmektedir. Karar ağaçlarının en iyi özelliği kolay anlaşılır olmasıdır.

Tablo 2.1: Karar ağaçları için eğitim kümesi

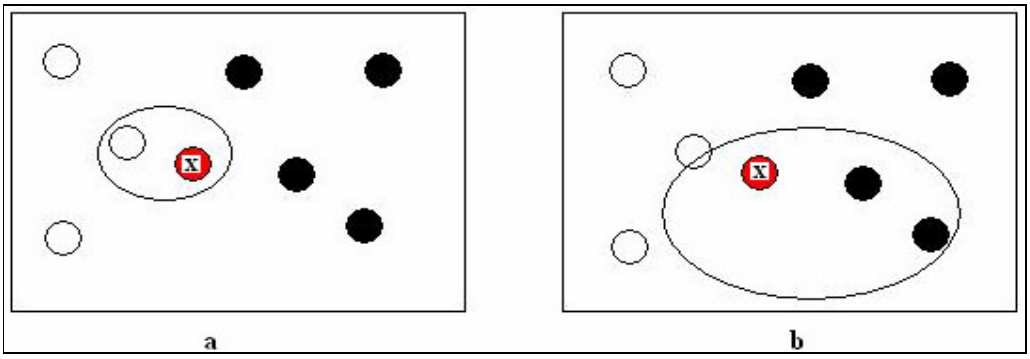
Şekil	**Renk**	**Boyut**	**Karar**
Yuvarlak	Yeşil	Küçük	Ret
Kare	Siyah	Büyük	Kabul
Kare	Sarı	Büyük	Kabul
Yuvarlak	Sarı	Küçük	Ret
Kare	Yeşil	Büyük	Ret
Kare	Sarı	Küçük	Kabul



Şekil 2.3: Karar ağaçları

2.1.4. En yakın k komşuluğu

En yakın k komşuluğu (ENKK) algoritması çok kullanılan, eğiticili ve örnek tabanlı bir örüntü tanıma algoritmasıdır. Örnek tabanlı algoritmalarda, eğitim işlemi gerçekleştirilmez. Test edilecek örnek, eğitim kümesindeki her bir örnekle bire bir karşılaştırılarak öbekleme işlemi gerçekleştirilmektedir [6]. Şekil 2.4'de daireler içi boş ve dolu olarak yerleştirilmişlerdir. Buradaki içi boş olanlar 0'ları içi dolu olanlar ise 1'leri temsil etmektedir. Şekil 2.4 (a)'daki x dairesini hangi sınıfa ait olduğunu bulmak için, en yakın birinci komşu daireye bakılır. Bu daire 0 sıfırı temsil ettiği için, 0 olarak sınıflandırılır. Şekil 2.4 (b)'de ise x dairesine en yakın 3 komşu daireye bakılır. Bu dairelerden ikisi 1'i, diğeri ise 0'ı temsil ettiği için, x dairesi 1 olarak sınıflandırılır.



Şekil 2.4: ENKK ile yapılmış olan iki ayrı sınıflandırma

2.1.5. Yapay sinir ağları sınıflandırıcısı

İlk kez 1943'te ortaya çıkmış olan yapay sinir ağları bilgisayarlarda 1980'lerde kullanıma başlanmıştır. Beynin yapısından esinlenilmiş bir bilgi işleme sistemidir [7]. Bu tez çalışmasında önerilen model yapay sinir işlem birimi model olduğundan, yapay sinir ağları Bölüm 3'te ayrıntılı bir şekilde açıklanmaktadır.

3. YAPAY SİNİR AĞLARI

Yapay Sinir Ağları (YSA), insandaki biyolojik sinir hücresinden esinlenilerek oluşturulmuş bilgi işleme modelleridir. Literatürde 100'den fazla yapay sinir ağı modeli bulunmaktadır. YSA'ların en büyük özelliği öğrenme yetenekleridir. Bu durum araştırmacıların dikkatini çeken en önemli özelliğidir.

1943 yılında bir nörobiyolojist olan Warren McCulloch ve bir istatistikçi olan Walter Pitts, "Sinir Aktivitesindeki Düşüncelere Ait Bir Mantıksal Hesap" başlıklı bir makale ile ilk sayısal bilgisayarlara ışık tutmuştur [8]. John Von Neumann bu makaleyi, "elektronik beyinler" için bir kopya olarak görmüştür. Yapay zekâ alanındaki araştırmacılar içerisinde ayrı bir yeri olan Marvin Minsky, bu makaleden aldığı ilhamla geniş ölçekli zekâ fikrini ortaya atmış ve uzman sistemlerin doğmasına neden olmuştur. Bronx Yüksek Bilim Okulu'ndan Frank Rosenblatt, gözün hesaplamaları ile ilgilenmiştir. Bu bilim adamları, öğrenmenin ve zekânın herhangi bir özelliğinin benzetiminde bilgisayarların aktif olarak nasıl kullanılabileceğini, 1956 yılında düzenlemiş oldukları ilk yapay zekâ konferansında tartışmışlardır.

1959'da Bernard Widrow geliştirmiş olduğu, basit işlem birimine dayanan adaline'ı (Adaptive Linear Neuron) 1960'da bir elektronik devreye uygulamıştır [9]. Adaline ve çok katmanlı biçimi olan "madaline" (Multiple Adaline); ses tanıma, karakter tanıma, hava tahmini ve uyarlamalı kontrol gibi çok çeşitli uygulamalar için kullanılmıştır. Daha sonraları Adaline, ayrık bir çıkış yerine sürekli bir çıkış üretmek için geliştirilmiştir. Widrow, telefon hatları üzerindeki yankıları gidermeye yarayan uyarlamalı filtreleri geliştirmede, uyarlamalı doğrusal işlem birimi algoritmasını kullanmıştır. Böylelikle ilk defa YSA'lar gerçek bir probleme uygulanmıştır.

Helsinki Teknik Üniversitesi'nden Teuvo Kohonen, 1970'lerin ilk yıllarında uyarlamalı öğrenme ve birleşik hafızalar üzerine temel çalışmalar yapmış. Yapmış bu çalışmalar da eğiticisiz öğrenme algoritmalarının gelişmesine ışık tutmuştur [10].

Minsky ve Papert'in almaç (perceptron) isimli kitaplarında, YSA'nın temel olarak ilgi çekici konular olmadığını belirtmeleri birçok araştırmacının bu alanda çalışmaktan vazgeçmelerine neden olmuştur. YSA konusunda çalışmaya devam eden Grossberg, YSA modellerini yapılandırmak için nörolojik verinin kullanılması, algı ve hafıza için YSA tabanlı mekanizmaların önerilmesi, belirgin eşitliklerle bütünleşen bir bağlantı noktası (sinaptik) modeli için bir ilişkilendirici kural üzerinde çalışmıştır.

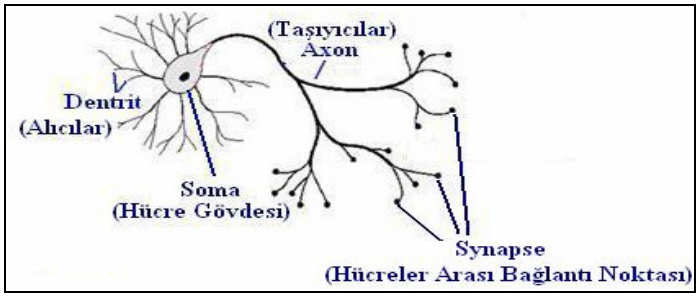
1982 yılında ilgi çeken bir başka gelişme, moleküler biyolojiden beyin kuramcılığına geçiş yapan bir model, Caltech fizikçisi Hopfield tarafından sunulmuştur. Kendi adıyla anılan bir ağ yapısı geliştirmiş olup birçok alana uygulanmıştır.

İlk kez 1987 yılında yapılan yapay sinir ağları kongresinden sonra YSA uygulamaları yaygınlaşmıştır. Günümüzde, YSA'larla ilgili araştırmalar yapan çok sayıda bilim adamı ve araştırma grupları vardır. Farklı bilim alanlarında çalışan araştırmacılar, birçok yeni gelişmeleri sunmaya devam etmektedirler.

3.1. Biyolojik Sinir Hücresi

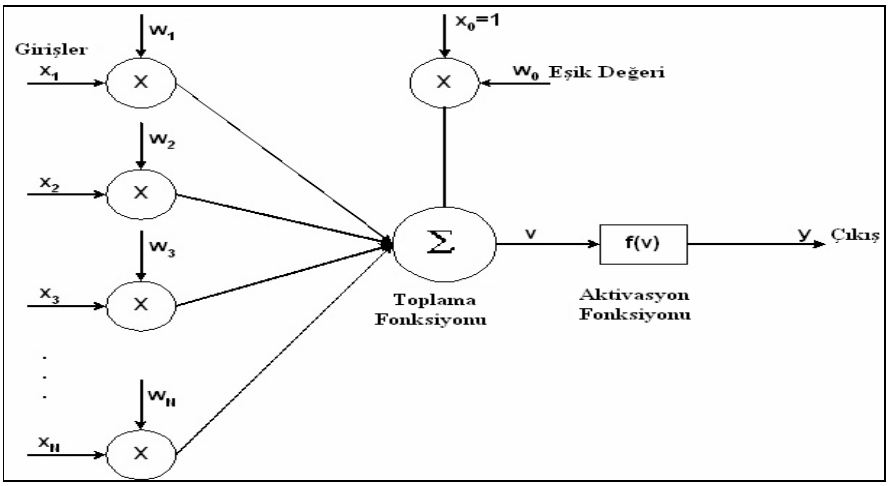
Şekil 3.1 'de görünen biyolojik sinir hücresi yapısı üç bölümden oluşmaktadır [7]. Bunlar,

- Hücre gövdesi (Soma)
- Taşıyıcılar (Axon)
- Alıcılar (Dentrit)



Şekil 3.1: Biyolojik sinir hücresi yapısı

Biyolojik sinir hücresindeki hücre gövdesi hücreyi denetler ve hücre etkinliklerinin tümünü yönetmekle sorumludur. Hücre gövdesinden iki çeşit uzantı çıkmaktadır. Bunlardan alıcılar bilgiyi taşıyıcılardan alır ve gerekli işlemler için hücre gövdesine taşır. Taşıyıcılar ise hücre içerisinde işlenmiş olan bilgiyi diğer sinir hücrelerinin alıcılarına iletir. Fakat iki sinir hücresinin arasında direk bir bağlantı yoktur. Bir sinir hücresinin taşıyıcıları ile diğer bir sinir hücresinin alıcıları arasında synapse denilen hücreler arası bağlantı noktaları vardır. Bilgi taşıyıcıların ucuna ulaştığında iletimi sağlayacak kimyasal madde bu synapse bağlantı noktasına yayılır. Yayılan bu madde sonraki sinir hücresinin alıcılarını uyarır. Böylece bir sinir hücresindeki bilgi diğer bir sinir hücresine ulaşmış olur. Biyolojik sinir hücresinden esinlenilerek oluşturulan yapay sinir hücresi yapısı Şekil 3.2'de verilmiştir [7]. Tablo 3.1 ise iki hücre modeli arasındaki karşılaştırmayı vermektedir.



Şekil 3.2: Yapay sinir hücresi yapısı

Tablo 3.1: Biyolojik sinir hücresi ile yapay sinir hücresinin karşılaştırılması

Biyolojik Sinir Hücresi	Yapay Sinir Hücresi
Sinir Hücresi	İşlem Birimi
Hücre Gövdesi	Transfer Fonksiyonu
Alıcılar	Toplama Fonksiyonu
Taşıyıcılar	İşlem Biriminin Çıkışı
Sinaps	Ağırlıklar

3.2. Yapay Sinir Hücresi

Yapay sinir hücreleri en küçük bilgi işleme birimleridir. Şekil 3.2 de görülen bir yapay sinir hücresi 5 bölümden oluşur [11]. Bunlar girişler, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıkıştır. Girişler, diğer hücrelerden veya dış ortamdan yapay sinir hücresine giren bilgilerdir. Bilgiler ağırlıklar üzerinden hücreye girer. Ağırlıklar ilgili girişin hücre üzerindeki etkisini belirler. Toplama fonksiyonu bir hücreye giren net girdiyi hesaplar. Buradaki toplama fonksiyonu girişler ile ağırlıkların çarpımlarının toplamı olduğu gibi, girişler ile ağırlıkların çarpımının çarpımı da olabilir. Bunun dışında maksimum ya da minimum olan fonksiyonlarda kullanılmaktadır. Aktivasyon fonksiyonu ise hücre çıkışını hesaplayan genellikle doğrusal olmayan bir fonksiyondur. Yapay sinir hücrelerinde net girdiyi +1 arttıran ya da -1 azaltan değerli eşik girişi vardır. Bu giriş değeri sabit değerli olarak bir giriş vektörü (x_0) olarak polarma (bias) girişi şeklinde adlandırılır. Bu giriş aşırlık vektörü w_0 katsayısı ile gösterilir ve eşik değer b katsayısı olarak adlandırılır.

3.2.1. İşlem birimi modeli

Şekil 3.2 de görünen yapay sinir hücresini matematiksel denklemi aşağıdaki gibidir:

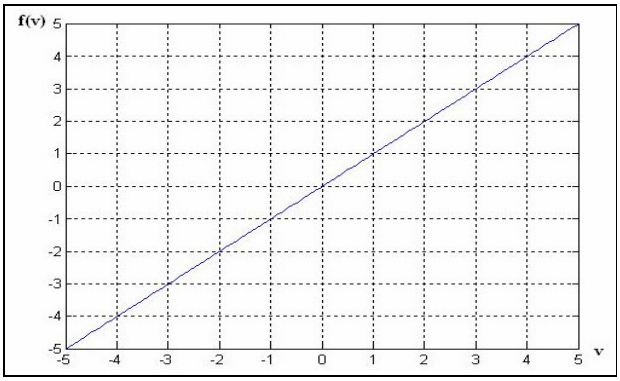
$$v = \sum_{i=1}^{N} x_i w_i + w_0, y = f(v) \tag{3.1}$$

Burada $x_i \varepsilon R$ (i=1,…,N) işlem birimi modelinin giriş vektörleridir, $w_i \varepsilon R$ (i=1,…,N) ağırlık matrisidir ve w_0 eşik değeridir. v net giriş ve f: $R \rightarrow R$ işlem birimi modelinin aktivasyon fonksiyonudur, y ise çıkışıdır.

3.2.2. Aktivasyon fonksiyonları

Aktivasyon fonksiyonları, yapay sinir hücresinin çözmek istediği probleme göre çeşitlilik gösterir. Bu fonksiyonlar sabit parametreli olduğu gibi, ayarlanabilir parametreli de olabilir. Eğer doğrusal bir problem çözülüyorsa doğrusal aktivasyon fonksiyonu kullanılır. Şekil 3.3'te değişim grafiği verilen doğrusal fonksiyonun denklemi aşağıdaki olup, n katsayısı eğimi ifade etmektedir.

$$f(v) = nv \tag{3.2}$$



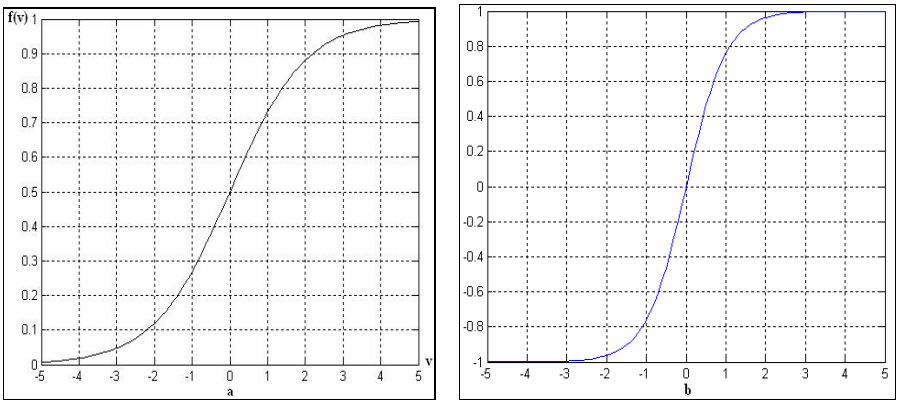
Şekil 3.3: Doğrusal aktivasyon fonksiyon

Şekil 3.4'de grafikleri verilen sigmoid fonksiyonlarının, sürekli, türevi alınabilir ve doğrusal olmadığı için; doğrusal olmayan problemlerin çözümünde sıkça kullanılır. Logaritmik ve hiperbolik tanjant sigmoid fonksiyonlarının denklemi sırasıyla aşağıdaki gibidir:

$$f(v) = \frac{1}{1+e^{-av}} \tag{3.3}$$

$$f(v) = \frac{1-e^{-av}}{1+e^{-av}} \tag{3.4}$$

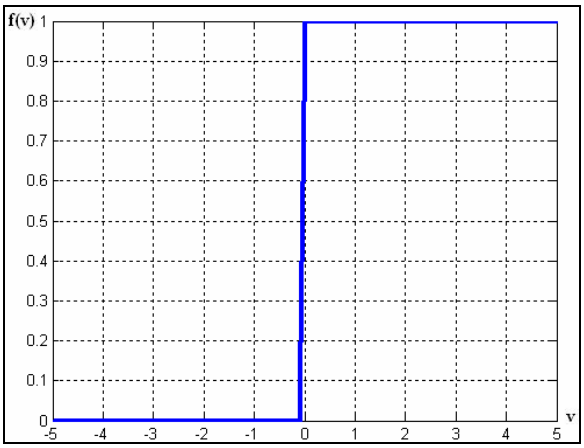
Yukarıdaki denklemde a katsayısı sabit olarak 1 seçilir. Fakat istenirse, problemin özelliğine göre a değişkeni de ayarlanabilir.



Şekil 3.4**:** Logaritmik Sigmoid (a) ve Hiperbolik Tanjant (b) aktivasyon fonksiyonu

McCulloch-Pitts modelinde de kullanılan eşik aktivasyon fonksiyonu, işlem biriminde mantıksal çıkış verdiği için; ikili (binary) sınıflandırma problemlerinde sıkça kullanılır [7]. Bu fonksiyonun grafiği Şekil 3.5'te ve denklemi ise Denklem 3.5'te verilmiştir.

$$f = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \tag{3.5}$$



Şekil 3.5: Eşik aktivasyon fonksiyonu

3.3. Yapay Sinir Ağlarının Sınıflandırılması

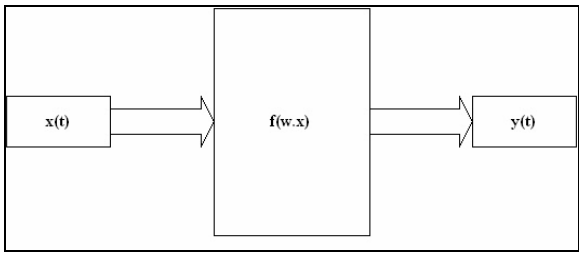
İşlem birimlerinin birbiri ile çeşitli şekilde bağlanması sonucu yapay sinir ağları oluşur. Buradaki işlem birimlerinin birbiri ile bağlanma şekilleri ağın yapısını belirler. Oluşturulan bu çeşitli ağların, istenilen problemi çözmesi için ağırlıkların değiştirilmesi gerekir. Bu ağırlıkların nasıl değiştirileceğini de öğrenme algoritmaları belirler. Böylelikle YSA'lar yapılarına ve öğrenme algoritmalarına göre iki sınıfa ayrılırlar.

3.3.1. Yapay sinir ağlarının yapılarına göre sınıflandırılması

YSA'lar yapılarına göre; ileri beslemeli (feedforward) ve geri beslemeli (recurrent) olarak sınıflandırılır [12].

3.3.1.1. İleri beslemeli yapay sinir ağları

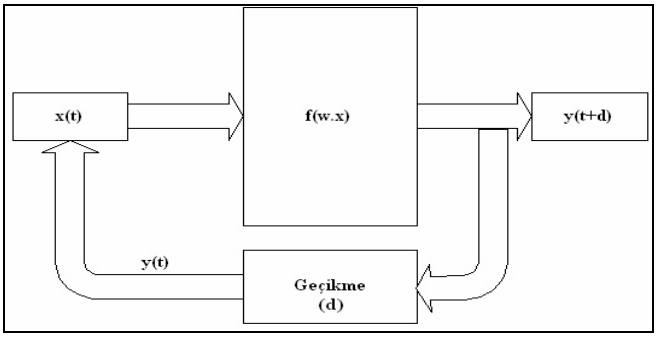
İleri beslemeli yapay sinir ağları (İBYSA); işlem birimlerinin katmanlar şeklinde düzenlenmesiyle oluşur. Bir katmandaki işlem birimlerinin çıkışları diğer katmandaki işlem birimlerinin ağırlıklarına bağlanarak bu katmanın girişlerini oluşturur. Bu tür yapılar genellikle 3 katman şeklinde düzenlenir. Bu katmanlar; giriş katmanı, orta katman (gizli katman) ve çıkış katmanıdır. Fakat istenirse orta katman tek bir katmandan değil birkaç katmandan oluşturulabilir. Giriş katmanı ağa sunulan bilgiler üzerinde hiçbir değişiklik yapmadan orta katmana iletir. Bilgiler orta ve çıkış katmanında işlenerek ağ çıkışı olarak verilir. Bu tür katmanlarda gecikme yoktur. Çıkış değeri, olması geren çıkış değeri ile karşılaştırılarak bir hata işareti elde edilir. Bu hata işaretine göre, en çok kullanılan geriye yayma algoritması (Backpropagation) ile ağırlıkların yenilenmesi sağlanır. Bu ağın blok diyagramı Şekil 3.6 da verilmiştir. Bu tür ağlara örnek olarak, Çok Katmanlı Almaç (Multi Layer Perceptron (MLP)) ve Öğrenme Vektör Nicemleme (Learning Vector Quantization (LVQ)) ağları gösterilebilir.



Şekil 3.6: İleri beslemeli ağın blok diyagramı

3.3.1.2. Geri beslemeli yapay sinir ağları

Geri beslemeli yapay sinir ağları (GBYSA), en az bir işlem biriminin çıkışı kendisine veya başka bir işlem birimine giriş olarak verilmesi ile oluşturulur. Geri besleme, bir katmandaki hücreler arasında olabileceği gibi katmanlar arasındaki hücrelerde de olabilir. Böylelikle girişler hem ileri yönde hem de geri yönde aktarılmış olur. Bu durum ağa dinamiklik kazandırır. Bu tür ağlar özellikle tahmin edici sistemlerde kullanılır. Şekil 3.7'de blok diyagramı gösterilen GBYSA'ya örnek olarak Hopfield, Self Organizing Map, Elman ve Jordan ağları verilebilir.



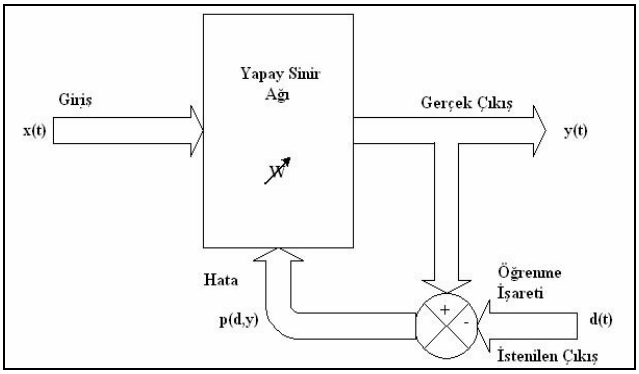
Şekil 3.7: Geri beslemeli yapay sinir ağının blok diyagramı

3.3.2. Yapay sinir ağlarının öğrenme algoritmalarına göre sınıflandırılması

YSA'da öğrenme oluşturulan ağın istenilen sonucun elde edebilmesi için; ağırlıkların değiştirilmesidir. Aşağıdaki alt bölümlerde anlatılan üç temel öğrenme yönteminden söz edilebilir.

3.3.2.1. Eğiticili öğrenme (Supervised learning)

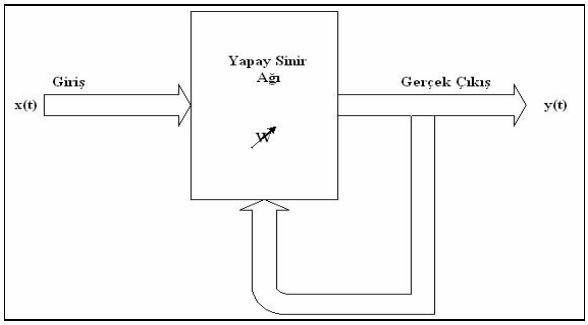
YSA'nın hesapladığı her bir çıkış değeri için istenilen çıkış değeri verilir. Verilen bu çıkış değeri ile ağın gerçek çıkış değeri arasındaki farka göre; işlem birimleri ağırlıklarının yeniden düzenlemesine eğiticili öğrenme denir. Eğiticili öğrenme blok diyagramı Şekil 3.8'de gösterilmektedir. Eğiticili öğrenme algoritmasına örnek olarak delta kuralı ve genişletilmiş delta kuralı veya geri yansıtma algoritmalarını verebiliriz [12].



Şekil 3.8: Eğiticili öğrenme blok diyagramı

3.3.2.2. Eğiticisiz öğrenme (Unsupervised learning)

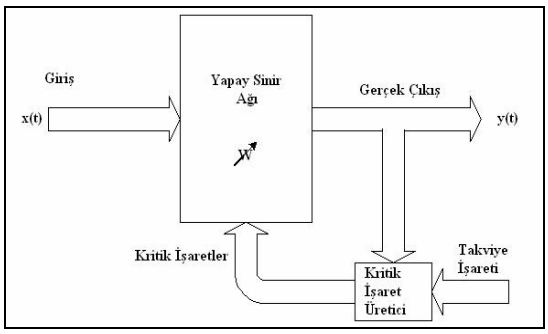
Blok diyagramı Şekil 3.9'da verilen eğiticisiz öğrenme yönteminde; eğiticili öğrenmede olduğu gibi YSA'dan istenilen çıkış değeri önceden verilmemektedir. Ağ elde ettiği çıkış değerlerine göre, kendi yaklaşımlarını kendisi belirlemektedir [12]. ART (Adaptive Resonance Theory) veya SOM (Self Organizing Map) algoritmalarını eğiticisiz öğrenmeye örnek olarak verebiliriz.



Şekil 3.9: Eğiticisiz öğrenme blok diyagramı

3.3.2.3. Takviyeli öğrenme (Reinforcement learning)

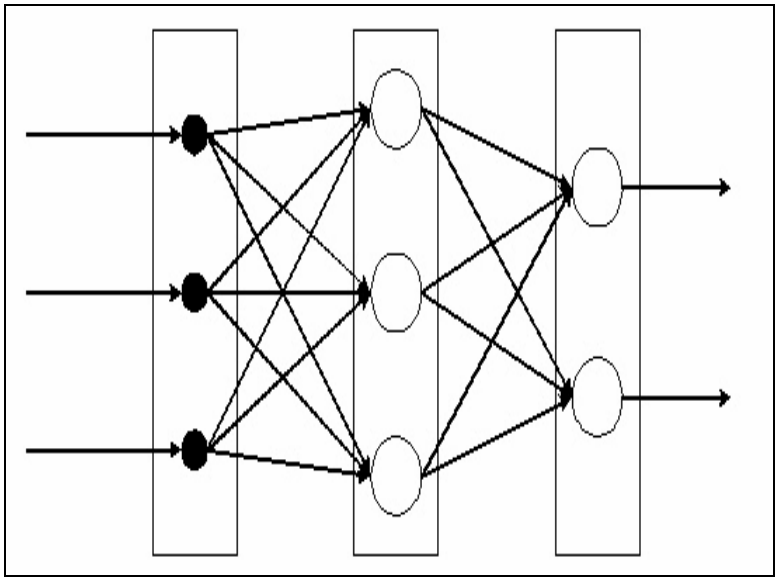
Blok diyagramı Şekil 3.10 da görülen takviyeli öğrenme eğiticili öğrenme yöntemine benzemektedir. Bu yöntemde istenilen çıkış değeri ağa verilmemektedir. Bunun yerine elde edilen gerçek çıkış değerini, ağa verilen giriş değerine göre doğruluğunu kontrol eden bir ölçüt kullanılmaktadır. Bu tip öğrenmeye örnek olarak ise Boltzman kuralını veya genetik algoritmaları verebiliriz.



Şekil 3.10: Takviyeli öğrenme blok diyagramı

3.4. İleri beslemeli Çok Katmanlı Ağlar Ve Öğrenme Algoritmaları

İleri beslemeli çok katmanlı ağlar (İBÇKA)'ın eğitiminde çok farklı öğrenme algoritmaları kullanılmaktadır. Bu tip ağlarda eğiticili öğrenme algoritması olarak; geriye yayma, Levenberg-Marquardt, hızlı yayılım (QP-Quikprop, delta-bar-delta (DBD), genişletilmiş dalta-bar-dalta (EDBD-extended delta-bar-delta) algoritmaları örnek olarak verilebilir. Takviyeli öğrenme yönteminde ise genetik algoritma (GA) verilebilir. İBÇKA'nın blok diyagramı Şekil 3.11 de verilmiştir. Bu tip ağaların yaygın olarak kullanılmasının sebebi; yukarıda saydığımız birçok algoritmanın çok rahat bir biçimde kullanılmasıdır. Yukarıda belirtilen algoritmaların olarak kullanılan geriye yayma algoritması anlatılacaktır.



Şekil 3.11: İleri beslemeli çok katmanlı yapay sinir ağı

3.4.1. Geriye yayma algoritması

Bu algoritmadaki amaç; hata işaretini bulmak ve bu hata işaretine göre ağırlıkları yenilemektir. Eğitim setindeki her bir örnek için aşağıdaki 3 adımın tekrarlanması gerekmektedir. Aşağıda adımları verilen algoritmanın akış şeması Şekil 3.12'de gösterilmiştir [12].

1. Adım: İleri yayılım

Saklı katmanda bulunan işlem birimleri için v_i ve y_i değerleri hesaplanır:

$$v_i = \sum_{r=1}^{N} x_r w_{ri} + w_0 \quad , y_i = f_i(v_i) \tag{3.6}$$

Denklem 3.6'te i=1,…H gizli katmandaki işlem birimlerinin sayısını, r = 1,…,N ise yapay sinir ağına verilen giriş vektörlerinin boyutunu ifade etmektedir. x_r ağın giriş vektörü, w_{ri} gizli katmandaki her bir işlem biriminin giriş katmanında bulunan giriş vektörlerinin sayısına göre ağırlıkları, w_0 ise eşik değeri göstermektedir. v_i gizli katmandaki her bir işlem biriminin net girişini, f_i gizli katmanda kullanılan aktivasyon fonksiyonu ve y_i ise gizli katmandaki işlem birimlerinin çıkışını ifade etmektedir.

Çıkış katmanında bulunan her işlem birimi için v_j ve y_j değerleri hesaplanır:

$$v_j = \sum_{i=1}^{H} y_i w_{ij} + w_0 \quad , y_j = f_j(v_j) \tag{3.7}$$

Denklem 3.7'daki j=1…M çıkış katmanında kullanılan işlem birimlerinin sayısını göstermektedir. v_j çıkış katmanındaki her işlem biriminin net girişi, w_{ij} ise bu işlem birimlerine bir önceki katmandan gelen giriş değerlerinin sayısına göre ağırlık değerleri ve w_0 ise eşik değeri ifade etmektedir. f_j yine çıkış katmanında j. İşlem birimi için kullanılan aktivasyon fonksiyonunu, y_j ise ağın çıkış değerini göstermektedir.

2. Adım: Geri yayılım

Çıkış katmanında kullanılan her bir işlem birimi için hata aşağıdaki gibi hesaplanır:

$$\delta_{2j} = (o_j - y_j) f_j'(v_j) \tag{3.8}$$

Burada o_j istenen çıkış değerini, y_j ağın o anki hesaplanan çıkış değeri ise yine j. İşlem biriminin aktivasyonun fonksiyonun türevini ifade etmektedir. Gizli katmanda kullanılan her bir işlem birim için hata Denklem 3.9'a göre aşağıdaki gibi hesaplanır:

$$\delta_{1i} = f'(v_i)\sum_{i=1}^{H}\delta_{2j}w_{ij} \tag{3.9}$$

3. Adım: Ağırlıkların yenilenmesi

Çıkış ile gizli katman arasında kullanılan ağırlıkların yenilenmesi sırasında kullanılan ağırlık değişimleri aşağıdaki denkleme göre hesaplanır:

$$\Delta w_{ij}(t) = \eta\delta_{2j}y_i + \alpha\Delta w_{ij}(t-1) \tag{3.10}$$

Gizli katman ile giriş katmanı arasında kullanılan ağırlıkların yenilenmesi sırasında kullanılan ağırlık değişimleri aşağıdaki denkleme göre hesaplanır:

$$\Delta w_{ri}(t) = \eta\delta_{1i}x_r + \alpha\Delta w_{ri}(t-1) \tag{3.11}$$

Denklem 3.10 ve 3.11'deki t değeri ağın iterasyon sayısını, η öğrenme ve α ise momentum katsayılarını göstermektedir. Daha sonra ise aşağıdaki denklemler kullanılarak ağın yeni ağırlıkları hesaplanır.

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t) \quad , \quad w_{ri}(t) = w_{ri}(t-1) + \Delta w_{ri}(t) \tag{3.12}$$

3.5. Yapay Sinir Ağlarının Özellikleri Ve Uygulama Alanları

Yapay sinir ağlarında işlem birimlerinin paralel bağlanarak oluşturulmasından dolayı; YSA'lar karmaşık problemleri çözebilme yeteneğine sahiptirler. Aşağıdaki özelliklerden dolayı, YSA birçok bilim alanında kullanılmaktadır:

- Doğrusallık: YSA'ların yapıları doğrusaldır. Fakat ağı oluşturan işlem birimlerinde kullanılan aktivasyon fonksiyonuna göre doğrusallığı belirlenmektedir. İşlem birimlerinin bu özelliği doğrusal olmayan karmaşık problemlerin çözümünde kolaylık sağlar.
- Öğrenme: YSA'larda kullanılan ağırlıkların değerleri; belirli bir hata ölçütü ve öğrenme algoritmaları yardımıyla yenilenir. Bu özellik ile ağın öğrenmesi gerçekleştirilir.
- Genelleme: YSA, istenilen problemi öğrendikten sonra eğitim sırasında karşılaşmadığı test örnekleri içinde istenen tepkiyi üretir.

- Uyarlanabilirlik: YSA'lar bir problemi çözmek için eğitildikten sonra, problemdeki değişikliğe göre ağ tekrar eğitilebilir.
- Hata Toleransı: YSA'ların yapılarından dolayı, herhangi bir problemin çözümü sırasında hatayı indirgeme yetenekleri son derece iyidir.
- Donanım ve Hız: Ağın içinde kullanılan işlem birimlerinin paralel bağlantılarından dolayı, YSA'ların bilgi işleme hızları yüksektir.
- Analiz ve Tasarım Kolaylığı: YSA'lar çözülmek istenen probleme göre kolaylıkla tasarlanabilir.

YSA'ların uygulama alanları aşağıdaki gibi 6 gruba ayrılabilir:

Arıza Analiz ve Tespiti: Elektrik makinelerinin, uçakların ya da bileşenlerinin, tümleşik devrelerin ve benzeri arıza analizlerinde kullanılır.

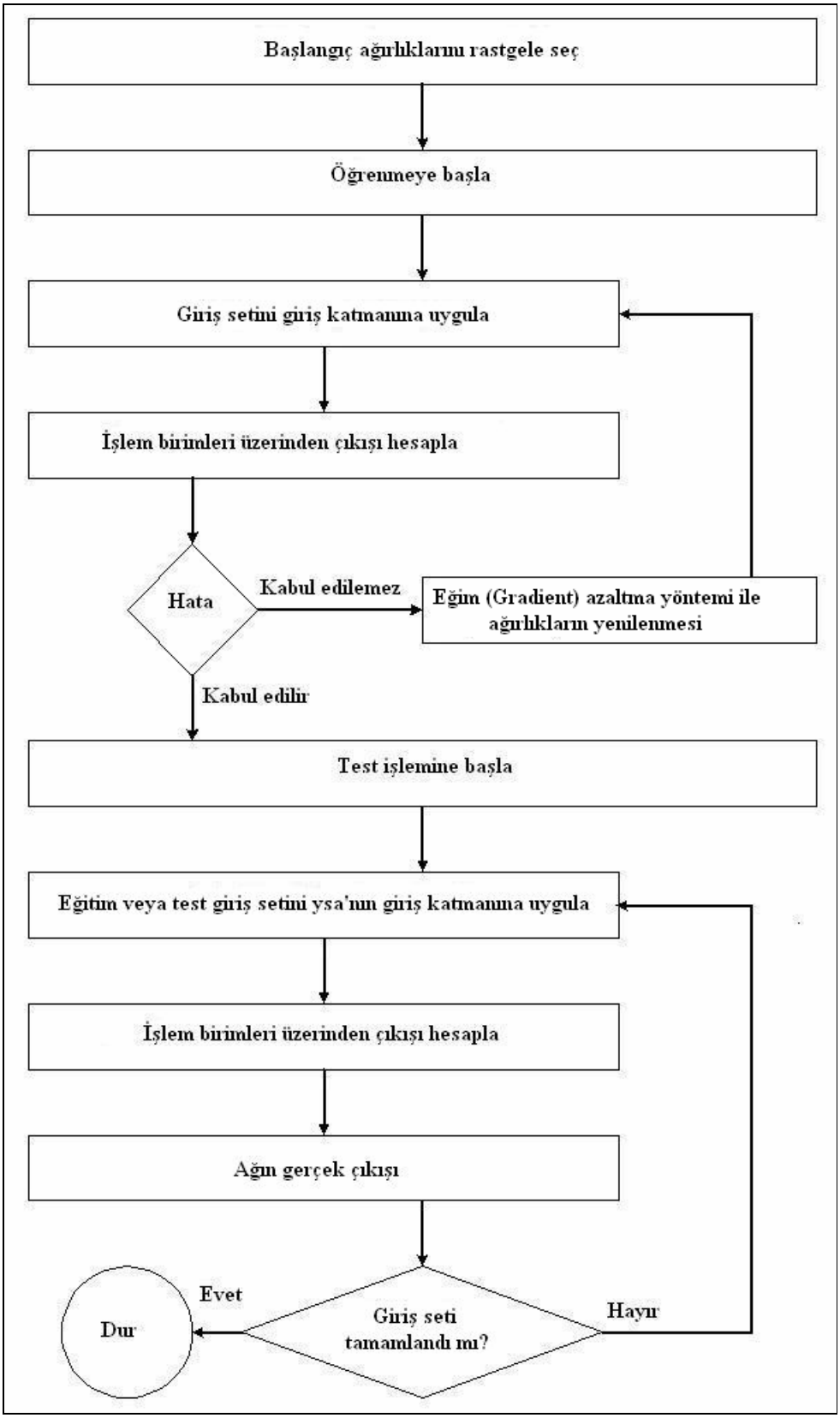
Tıp Alanında: İşitme engelliler için ses analizi, belirli hastalıkların teşhis ve tedavisi, ameliyat görüntüleme, ilaçların yan etkilerinin analizi, X-ışınların okunması, Epileptik felcin nedenlerinin araştırılması ve şeker hastalığının riskinin belirlenmesi başlıca uygulama alanlarıdır.

Savunma Sanayi: Radar işaretlerinin analizi, yeni ve gelişmiş silahların yapımında, hedef tanıma ve izleme v.b.

Haberleşme: Görüntü ve veri sıkıştırma, otomatik bilgi sunma servisleri, konuşma verisinin gerçek zamanda analizi v.b.

Üretim: Robot ve kontrol sistemlerini otomatikleştirme, üretim ve kalite kontrolü, montaj hattında parça seçimi.

Otomasyon ve Kontrol: Uçaklarda otomatik pilot sistemi otomasyonu, ulaşım araçlarında otomatik yol bulma ve gösterme, robot sistemlerin kontrolü, doğrusal olmayan sistem modelleme ve kontrolü v.b.



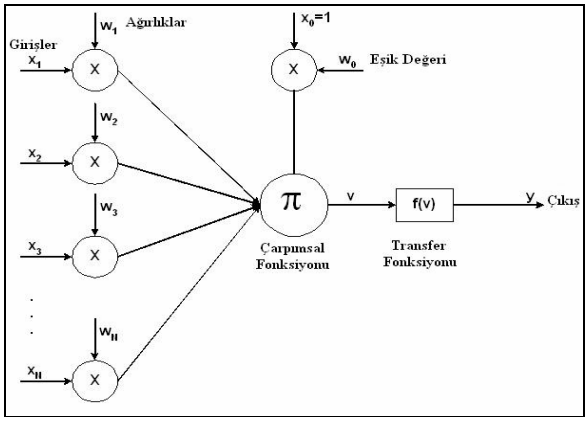
Şekil 3.12: İleri beslemeli çok katmanlı ağda geri yayılım akış şeması

4. ÇARPIMSAL İŞLEM BİRİMİ MODELİ

Çarpımsal işlem birimi genellikle yüksek dereceli ve melez olan yapay sinir ağlarında kullanılmıştır [13–15]. Şekil 4.1'de yapısı ve Denklem 4.1'de de matematiksel ifadesi görünen çarpımsal işlem birimi; π işlem birimi olarak adlandırılır [16].

$$v = \prod_{i=1}^{N} w_i . x_i , \quad y = f(v) \tag{4.1}$$

Denklem 4.1'de, $x_i \in R$ (i=1,…,N) çarpımsal işlem biriminin girişleridir; $w_i \in R$ (i=1,…,N) ağırlıklarıdır. f: $R \rightarrow R$ çarpımsal işlem biriminin aktivasyon fonksiyonu ve y ise işlem birimi çıkışını ifade etmektedir.



Şekil 4.1: Çarpımsal işlem birimi

Çarpımsal işlem birimi bazı melez yapay sinir ağlarında başarılı bir şekilde kullanılmış olsa da dezavantajları bulunmaktadır [15, 17]. Bu dezavantajlar aşağıdaki gibidir.

- Çarpımsal işlem birimi, giriş değerlerinin oluşturduğu yüzeydeki herhangi bir noktayı merkez kabul ederek kendi karar yüzeyini oluşturamamasıdır.
- Seçilebilir değişken parametrelere sahip olmamasıdır.

Çarpımsal işlem birimi N-bit değer eşitliği (parite) probleminin çözümünde de kullanılmıştır. Fakat burada kullanılan çarpımsal işlem birimini yapısında değişiklik yapılmıştır. Yapılan bu değişiklik ile oluşan çarpımsal işlem birimi π_t işlem birimi olarak adlandırılmıştır [18]. π_t işlem biriminin denklemi aşağıdaki gibidir:

$$v = b\prod_{i=1}(x_i - t_i)\ ,\ y = f_{th}(v) \tag{4.2}$$

Denklem 4.2'de b ε R olup Şekil 4.1'deki w_0 eşik değeri parametresi ile eş değerdir. x_i ε R (i=1,…,N) π_t işlem biriminin giriş değerleri ve t_i ε R (i=1,…,N) olup yine Şekil 4.1'de ağırlık (w_1,….w_N) parametreleri ile ifade edilmektedir. Aktivasyon fonksiyonu olarak da Şekil 3.5'de verilen eşik aktivasyon fonksiyonunu kullanılmaktadır. π_t işlem birimi yapısı için bir diğer önemli husus, girişlerle ağırlıkların çarpımı yerine farkı alınmaktadır.

N-bit değer eşitliği çözülürken kullanılan π_t işlem biriminde; çözülmek istenen verinin giriş değerlerinin sayı çift ise b<0, tek ise b>0 seçilir (Eğer N=çift ise b<0, N=tek ise b>0). t_i ise $0<t_i<1$ arasında seçilir [18].

Ayrıca oluşturulan π_t işlem birimi ile Denklem 4.1 ile tanımlanmış olan çarpımsal işlem birimi modelinin dezavantajları giderilmiştir [19]. Çarpımsal işlem birimi N-bit değer eşitliği problemini çözmek için başka çalışmalarda da kullanılmıştır [20]. N-bit değer eşitliği problemini çözmek için kullanılan π_t işlem birimi modelinin avantajları aşağıdaki gibi sıralanabilir [18]:

- Tek bir işlem biriminin kullanılması
- Gizli katman kullanılmasına gerek olmaması
- Öğrenme algoritmalarına ihtiyaç duymaması

Denklem 4.2'de verilen π_t işlem birimi modeli McCulloc-Pitts işlem birimi modelinden esinlenerek oluşturulmuştur. McCulloc-Pitts işlem birimi modelinin denklemi ise aşağıdaki gibidir.

$$v = w_0 + \sum_{i=1}^{N} x_i w_i\ ,\ y = f_{th}(v) \tag{4.3}$$

Denklem 4.3'te w_0 Şekil 4.1'deki eşik değeri parametresi ile aynıdır. $x_i \varepsilon R$ (i=1,…,N) işlem birimi elamanın giriş değerleri ve $w_i \varepsilon R$ (i=1,…,N) ağırlık değerleridir. Aktivasyon fonksiyonu olarak yine eşik aktivasyon fonksiyonu kullanılmaktadır.

McCulloc-Pitts işlem birimi modeli, π_t çarpımsal işlem birimi modelinin parametreleri ile karşılaştırıldığında w_0, b parametresine, w_i ise t_i parametresine karşılık gelmektedir.

5. OTOMATİK ÖĞRENME VERİ TABANLARI VE UYGULAMALARI

Bu bölümde tez çalışmasında amaçlanan değiştirilmiş π_t (π_m) işlem birimi modeli ile McCullcoh-Pitts işlem birimi modellerinin melez kullanımı hedeflenmiştir. π_m işlem birimi modelinde hedeflenen, π_t işlem birimi modeline bağlı kalarak b ve t_i parametrelerinin probleme uygun şekilde önceden belirlenerek, hiçbir eğitim algoritmasına gerek duyulmadan sınıflama işleminin yapılması amaçlanmıştır. Sınıflandırma problemleri için kullanılan π_m işlem birimi ile McCullcoh-Pitts işlem birimi modelleri üç farklı otomatik öğrenme veritabanına uygulanarak, amaçlanan melez yapının başarımı test edilmiştir. Bu tez çalışmasında kullanılan Rahip (Monk's), Balon ve Tic-Tac-Toe veritabanlarına Internet üzerinden ulaşılabilir [21, 22]. Tez çalışmasındaki farklı uygulamaların kodları Dev-C++ Versiyon 4.0.8.0 programı kullanılarak yazılmıştır. Bu veritabanları ile yapılan uygulamalar ve değerlendirmeler altbölümlerde açıklanmıştır.

5.1. Rahip (Monk's) Problemi Uygulanması

Rahip problemi ile aslında yapay bir robottan söz edilmektedir. Bu yapay robotun altı tane farklı özelliği bulunmaktadır. Bu özellikler sırası ile aşağıdaki gibi tanımlanmaktadır [23]:

x_1 : kafa şekli (head_shape) ε yuvarlak, kare, sekizgen (round, square, octagon)
x_2 : vücut şekli (body_shape) ε yuvarlak, kare, sekizgen (round, square, octagon)
x_3 : gülümsemesi (is_smiling) ε evet, hayır (yes, no)
x_4 : elinde tuttuğu (holding) ε kılıç, balon, bayrak (sword, balloon, flag)
x_5 : ceket rengi (jacket_color) ε kırmızı, sarı, yeşil, mavi(red, yellow, green, blue)
x_6 : kuyruğu (has_tie) ε evet, hayır (yes, no)

Yukarıdaki özelliklere dayanarak Rahip problemi M_1, M_2, M_3 şeklinde isimlendirilerek üç sınıfa ayrılmıştır [23]. Bu üç sınıf aşağıdaki özelliklerle belirlenmektedir:

Problem M_1: Eğer verilen özelliklerden kafa şekli ve vücut şekli aynı özelliğe sahipse ya da ceket rengi kırmızı ise bu problem M_1'e aittir (kafa şekli = vücut şekli ya da ceket rengi = kırmızı).

Problem M_2: Eğer robotun altı özelliğinden her hangi iki tanesi içerdikleri özelliklerin ilk değerini alıyorsa; bu problem M_2 'ye aittir.

Problem M_3: Eğer verilen özelliklerden robotun ceket rengi yeşil ve elinde tuttuğu kılıç ise ya da ceket rengi mavi değil ve vücut şekli sekizgen değilse; bu problem M_3'e aittir ((ceket rengi = yeşil ve elinde tuttuğu = kılıç) ya da (ceket rengi ≠ mavi ve vücut şekli ≠ sekizgen)).

Üç sınıfa ayrılmış olan Rahip probleminden yalnızca M_2 problemi değer eşitliği problemine benzerlik göstermektedir. Bundan dolayı π_m işlem birimi ve McCulloch-Pitts işlem birimi modelleri yalnızca M_2 problemine uygulanmıştır. M_2 probleminde toplam 432 tane veri bulunmaktadır. Diğer çalışmalarda [23] bu verilerin 169 tanesi algoritmaların testi için kullanılmıştır. Bu tez çalışmasında da aynı veri seti kullanılarak amaçlanan yöntemin sınıflandırma başarımı araştırılmıştır.

Yapay robotların 6 farklı yerinden alınan verilerle işlem yapabilmek için; bu verilerin yeniden düzenlenmesi ve ikili sayıya çevrilmesi gerekmektedir. Bu düzenleme aşağıdaki gibi yapılmıştır:

x_1 : kafa şekli (head_shape) ε 1, 2, 3 (0000, 0001, 0011)
x_2 : vücut şekli (body_shape) ε 1, 2, 3 (0000, 0001, 0011)
x_3 : gülümsemesi (is_smiling) ε 1, 2 (0000, 0001)
x_4 : elinde tuttuğu (holding) ε 1, 2, 3 (0000, 0001, 0011)
x_5 : ceket rengi (jacket_color) ε 1, 2, 3, 4 (0000, 0001, 0011, 0100)
x_6 : kuyruğu (has_tie) ε 1, 2 (0000, 0001)

Yukarıda görüldüğü gibi yapay robotun her bir noktasındaki özelliklere onluk tabanda sayısal giriş değerleri verilerek, 4 bitlik ikili sayı sistemine çevrilmiştir. Çevirme sırasında onluk sayı değerlerinin doğrudan karşılığı olan ikili sayı sitemine

bire bir değil bir eksiği olarak çevrilmiştir. Bunun nedeni ise veriyi değer eşitliği kuralına uydurabilmektir.

5.1.1. Rahip problemi ile ilgili yapılan diğer çalışmalar

Thurn ve diğerleri Belçika'da 1991 yazında otomatik öğrenme üzerinde gerçekleşen 2. Avrupa Yaz Okulunda farklı öğrenme tekniklerinin karşılaştırılmasını bir rapor şeklinde özetlemişlerdir [23]. Sembolik ve sembolik olmayan öğrenme teknikleri Rahip problemine uygulanmıştır. Bunun sonucunda elde edilen başarı oranları Tablo 5.1'de yalnızca M_2 problemi için verilmiştir. Bu karşılaştırmanın önemli bir özelliği de bir araştırmacı topluluğu tarafından yapılmasıdır. Thurn ve diğerlerinin, kendi algoritmalarında kullanmış oldukları kurallar, algoritmalarının özellikleri ya da yapay sinir ağları ile ilgili diğer özellikler aşağıdaki altbölümlerde vurgulanmıştır.

5.1.1.1. AQ17 algoritmaları

AQ17-DCI (Acuity Quality 17- Data-driver Constructive Induction) algoritması AQ öğrenme programına dayanmaktadır. AQ (Acuity Quality) algoritması bir örnek setinde minimum ya da minimuma yakın sayıda karakterize edilmiş kurallar üretir. Bu algoritma aşağıdaki gibidir [24, 25]:

1. Verilen bir karar sınıfı için; tohum denilen tek bir pozitif örnek seçilir. Bu örnekten bir set oluşturulur. Oluşturulan bu set çok genel birleşik tanımlamalardan elde edilir. Oluşan bu set yıldız olarak adlandırılır. Buradaki tanımlamaların her biri tüm negatif örnekleri dışlamak zorundadır.
2. Yıldız olarak adlandırılan örnek içinde tek bir tanımlama seçilir. Seçilen tanımlama en iyi tanımlama olarak adlandırılır. Eğer bu en iyi tanımlama tüm pozitif örnekleri kapsarsa algoritma durur.
3. Diğer durumda ise; yeni tanımlanmış örnekler arasında yeni bir tohum seçilir ve 1. ve 2. adımlar tüm örnekler kapsanana kadar sırasıyla tekrarlanır.

AQ17-DCI sınıf 0 için 2 kural ve sınıf 1 için 1 kural kullanır.

5.1.1.2 Geri yayılım ve ağırlık düşüşlü geri yayılım

Bu YSA yapısında 17 tane giriş birimi vardır. Bunarın hepsi 0 ya da 1 olarak seçilmiştir. Bütün giriş birimleri bir gizli katmandaki 2 işlem birimine bağlantısı vardır. Bu 2 gizli işlem birim ise çıkış işlem birimine bağlanmıştır. 0 ve 1 olarak kısıtlanmış olan çıkış değeri 0.5 'den büyük ve eşit ise bir giriş sınıf üyesi olarak sınıflandırılmıştır. Eğitim bir Sun Sparch iş istasyonunda 10 ile 30 saniye arasında gerçekleşmektedir. Connection Machine CM-2 olarak isimlendirilmiş paralel bir bilgisayarda eğitim zamanı bir rahip problemi için 5 saniyeden az bir süreye indirgenmiştir. 90 eğitim döngüsünde (epoch) sonra bu sistemin verimi %100 doğruluğa ulaşmaktadır.

Ağırlık düşüşlü tekniği geri yayılım algoritmasını kullanır. Burada ağın eğitimi sırasında ağırlık düşüşü denilen bir katsayı kullanılır. Kullanılan bu katsayı Thrun M_3 problemi için 0.1 alınmıştır. Böylelikle doğru sınıflandırma yapmıştır. Bu katsayıyı ağın gizli katmanında 2, 3 veya 4 işlem birimi olarak da farklı ağ yapılarında denemiştir.

5.1.1.3 Cascade correlation algoritması

Cascade Correlation eğiticili bir yapay sinir ağı yapısı kullanılmaktadır. Bu ağın yapısı iki türlüdür [26]. Birincisinde ağın yalnızca girişler, çıkış birimleri ve bunların arasındaki gizli katmandan meydana gelir. Bu tek gizli katman hatayı en aza indirgemek için eğitilir. Hatanın seviyesinde başka bir gelişme görünmez ise, bu ağın performansı değerlendirilir. Eğer hata yeteri kadar küçüklükte ise eğitim durur. İkincisinde ise, gizli katmana yeni işlem birimleri arta kalan hatayı azaltmak için ağa eklenir. M_2 problemi için Cascade Correlation algoritmasının sonucu; 82 iterasyondan sonra, 1 gizli işlem birimi, eğitim ve test setinde 0 hata, harcanan zaman ise 7.75 sn. olarak gerçekleşmiştir.

5.1.2 π_m işlem birimi ile McCulloch-Pitts işlem birimi modellerinin rahip M_2 problemine uygulanması

California üniversitesinin ftp sunucusundan M_2 problemi için alınan veriler 169x7 büyüklüğünde bir matris oluşturmaktadır [21]. Bu matristeki 7. sütun yapay robotun özelliklerine göre üretilmiş olan 1 ve 0'lardan oluşmaktadır. Buradaki verinin 64 tanesi 1 kalanı 0 değerine sahiptir.

Çalışmamızda ilk olarak 169 veri matrisi π_t işlem birimi modeline uygulanmıştır. Iyoda vd. N-bit değer eşitliği problemi için $b<0$ ve $0<t_i<1$ olarak belirlemelerinden dolayı, bu uygulamada π_t işlem biriminin parametrelerinden b ve t_i sırasıyla -1 ve 0.5 seçilmiştir ($b=-1$, $t_1,\ldots,t_N = 0.5$) [18]. Sonuç olarak da 169 tane verinin 105 tanesi doğru olarak sınıflandırılmıştır. Başarım ise %62.130 olarak gerçekleşmiştir.

Bu sonuç yeterince tatmin edici bulunmamıştır. Çünkü giriş değerine bakılmaksızın çıkış değerini 0 yapıldığında zaten verinin yapısından dolayı 115 veri doğru olarak sınıflandırılabilmektedir. Bu durumda da başarım zaten %68.047 olacaktır. Sınıflama başarımını arttırmak için π_t işlem birimindeki b değerinin seçimi N-bit değer eşitliği probleminden farklı seçilerek ve π_t işlem birimi ile birlikte McCulloch-Pitts işlem birimi modelinin birlikte melez kullanımı amaçlanmıştır. Bu model ise π_m işlem birimi modeli olarak adlandırılmıştır. Her iki işlem birimi modeli için ise Denklem 3.4'deki eşik aktivasyon fonksiyonu ve t_i ise 0 ile 1 arasında seçilmiştir. Daha sonra ise aşağıdaki kurallar probleme göre belirlenmiştir. Aşağıdaki kurallar sırası ile eklendiğinde sistemin sınıflama başarımı gittikçe artmaktadır. Yazılan kurallar, Rahip M_2 probleminin sahip olduğu onluk sayılar göz önünde bulundurularak yazılmıştır. Fakat veriyi N-bit değer eşitliği yöntemine benzetmek için, işlem birimine bu onluk sayıların bir eksiğinin ikili sayı sistemindeki karşılıkları verilmiştir. M_2 probleminde 6 giriş olduğuna göre ve her giriş için 4 bitlik bir ikili sayı belirlendiğinden işlem birimi modeli 24 girişe sahiptir. Aşağıdaki kurallar için $t_i=0.5$ seçilirken $b=\pm 2$ seçilerek ve verinin durumuna göre işareti değiştirilerek sistem sınıflandırma başarımları değerlendirilmiştir.

Kural 1: EĞER ($x_1=3$ ya da $x_2=3$ ya da $x_4=3$ ya da $x_5=3$)

İSE $b>0$ ve Denklem 4.2'yi DEĞİLSE $b<0$ ve Denklem 4.3'ü kullan.

Eğer yalnızca Kural 1 kullanılırsa, 169 verinin 125 tanesi doğru olarak sınıflandırılır. Sistem başarımı ise % 73.964 olur.

Kural 2: EĞER x_5=4 İSE b<0 ve Denklem 4.2'yi kullan.

Eğer Kural 1 ve Kural 2 beraber kullanılırsa, 169 tane verinin 143 tanesi doğru olarak sınıflandırılır. Sistem başarımı % 84.615 olur.

Kural 3: EĞER x_5=4 ve (($x_1 = x_2 = x_3 = 1$ ve $x_4 \neq 1$) ya da (x_1=3 ve $x_2 \neq 1$ ve $x_3 \neq 1$ ve $x_4 \neq 1$ ve $x_6 \neq 1$) ya da (x_1=3 ve $x_2 = x_3 = x_4 = x_6 = 1$) ya da ($x_1 = x_2 = x_3$=2 ve $x_4 \neq 1$) ya da (x_1=2 ya da 3 ve x_2 =2 ya da 3 ve x_3 =2 ya da3 ve x_4 =2 ya da3 ve x_6=2))

İSE b>0 ve Denklem 4.2'yi DEĞİLSE b<0 ve Denklem 4.2'yi kullan.

Eğer Kural 1, Kural 2 ve Kural 3 birlikte kullanılır ise 169 tane verinin 150 tanesi doğru olarak sınıflandırılır. Bu durumda sistem başarım % 88.767 olarak gerçekleşir.

Kural 4: EĞER (x_1=3 ya da x_2 =3 ya da x_4=3 ya da x_5 =3) ve (($x_1 = x_2 = x_3 = x_6 = 1$) ya da ($x_1$=3 ve x_2 = 2 ya da 3 ve x_3 = 2 ya da 3 ve x_4 = 2 ya da 3 ve x_5 = 2 ya da 3 ve x_6=2) ya da (x_1 = 2 ya da 3 ve $x_2 \neq 2$ ve $x_3 \neq 2$ ve $x_4 = x_5 = x_6 = 1$) ya da (x_1 = 2 ve x_2 = 3 ve x_3 = 2 ve $x_4 \neq 1$ ve $x_6 \neq 1$) ya da ($x_1 \neq 3$ ve $x_2 \neq 2$ ve $x_3 = x_4 = 1$ ve $x_6 \neq 2$) ya da (x_1 = 3 ve $x_2 = x_5 = x_6 = 1$) ya da (x_1=1 ya da 2 ve x_2=2 ya da 3 ve x_4=1 ya da 3 ve x_5=1 ya da 2 ve $x_3 = x_5 = x_6$) ya da (x_1=1 ve x_2=3 ve x_3=1 ve x_4=1 ve x_5=1 ve x_6=2))

İSE b<0 ve Denklem 4.2'yi DEĞİLSE b>0 ve Denklem 4.2'yi kullan.

Eğer Kural 1, Kural 2, Kural 3 ve Kural 4 birlikte kullanılır ise 169 tane verinin 167 tanesi doğru olarak sınıflandırılır. Sitem başarımı % 98.817 olarak gerçekleşir.

Kural 5: EĞER ($x_1 \neq 3$ ya da $x_2 \neq 3$ ya da $x_4 \neq 3$ ya da $x_5 \neq 3$) ve ($x_1 = x_2 = x_3 = x_4 = x_5 = x_6$)

İSE b<0 ve Denklem 4.2'yi kullan.

EĞER ($x_1 \neq 3$ ya da $x_2 \neq 3$ ya d $x_4 \neq 3$ ya da $x_5 \neq 3$) ve ($x_1 = x_2 = x_3 = x_5 = x_6$ ve $x_4 \neq x_6$) İSE b>0 ve Denklem 4.2'yi DEĞİLSE b<0 ve Denklem 4.3'ü kullan.

Eğer Kural 1, Kural 2, Kural 3, Kural 4 ve Kural 5 birlikte kullanılır ise 169 verinin 169 tanesi doğru olarak sınıflandırılır. Böylece sistem başarımı ise % 100.000 olur. Bu algoritmaya ilişkin programlama kodları, tez çalışmasındaki diğer otomatik öğrenme veri tabanlarına örnek oluşturması için, Bölüm EK A 'da verilmiştir.

Diğer çalışmalarda M_2 probleminin çözümü için eğiticili öğrenme algoritmaları seçildiğinden, kendi algoritmalarının eğitimini yaptıktan sonra bu tez çalışmasında kullandıkları 169 adet veri ile kendi algoritmalarını test etmişlerdir. Buna göre yüzdelik sistem başarımlarını hesaplanmıştır. Bu çalışmada önerilen melez model ise eğitim olmaksızın veri setindeki tüm veriler için yazılan kurallar ile sistem başarımını belirlemek ve diğer çalışmalarla karşılaştırmak doğru olacaktır. Bu durumda 432 veri için amaçlanan melez modelin yüzdelik sistem başarımı yaklaşık % 90 olarak gerçekleşmektedir.

Yukarıdaki çalışmaya ek olarak b ve t_i parametreleri farklı seçilerek tüm kuralların yer aldığı durumda sistem başarımının nasıl değiştiği belirlenmiştir. Bu işlemin sonucunda elde edilen veriler Tablo 5.1'de görülmektedir. Tablo 5.1'e göre M_2 problemi için; sistem sınıflandırma başarımını en yüksek düzeyde sağlayacak b ve t_i parametrelerinin değerleri aşağıda verilmiştir:

- b = ± 1 ve t_i = 0.3
- b = ± 2 ve t_i = [0.5-0.6]
- b = ± 3 ve t_i = [0.8-0.9]

Tablo 5.1: M_2 problemi için farklı b ve t parametrelerine göre sistem başarımı

b	**t**	**Sistem Başarımı (%)**	**b**	**t**	**Sistem Başarımı (%)**
± 1	0.1	87,500	**± 3**	0.1	87,500
	0.2	87,500		0.2	87,500
	0.3	**89,814**		0.3	87,500
	0.4	85,185		0.4	87,500
	0.5	81,712		0.5	87,500
	0.6	81,712		0.6	86,342
	0.7	81,712		0.7	86,342
	0.8	81,712		**0.8**	**89,814**
	0.9	81,712		**0.9**	**89,814**
± 2	0.1	87,500	± 4	0.1	87,500
	0.2	87,500		0.2	87,500
	0.3	87,500		0.3	87,500
	0.4	87,500		0.4	87,500
	0.5	**89,814**		0.5	87,500
	0.6	**89,814**		0.6	87,500
	0.7	85,185		0.7	87,500
	0.8	85,185		0.8	87,500
	0.9	85,185		0.9	86,342

Tablo 5.2'de M_2 problemi için yapılan çalışmaların başarım sırası görülmektedir. Tablo 5.2'de görüldüğü gibi bu tez çalışmasında önerilen π_m işlem birimi ile McCulloch-Pitts toplamsal işlem birimi modellerinin melez kullanımı % 89,81 sınıflandırma başarımı göstererek, sıralamada 7. sırada yer almaktadır.

Tablo 5.2: M_2 problemi için yapılan çalışmaların başarım sıralaması

Başarım Sıralaması	**Yöntem / Kaynak**	**Sistem Başarımı(%)**
1	AQ17-DCI / Bala ve diğerleri.	100,00
2	Backpropagation / Thrun	100,00
3	Backpropagation with weight decay / Thrun	100,00
4	Cascade Correlation / Fahlman	100,00
5	AQ17-HCI / Bala ve diğerleri.	93,10
6	AQ17-FCLS / Bala ve diğerleri.	92,60
7	**Melez işlem birimi modeli / bu tez çalışması**	**89,81**
8	AQ15-GA / Bala ve diğerleri.	86,80
9	Assistant Professional /Cestnik vd.	81,30
10	AQR / Kreuziger ve diğerleri	79,70
11	Prism / Keller	72,70
12	Ecobweb l.p. & information utility / Van de Welde	71,30
13	Mfoil / Dzeroski	69,20
14	ID5R / Kreuziger ve diğerleri.	69,20
15	ID3, no windowing / Kreuziger vd.	69,10
16	CN2 / Kreuziger ve diğerleri	69,00
17	ID3 / Kreuziger ve diğerleri	67,90
18	Ecobweb leaf prediction / Reich vd.	67,40
19	TDIDT / Van de Welde	66,70
20	IDL / Van de Welde	66,20
21	ID5R-hat / Van de Welde	65,70
22	Classweb 0.10 / Kreuziger ve diğerleri.	64,80
23	ID5R / Van de Welde	61,80
24	Classweb 0.15 / Kreuziger ve diğerleri.	61,60
25	Classweb 0.20 / Kreuziger ve diğerleri.	57,20

5.2 π_m İşlem Birimi Modelinin Balon Veri Tabanına Uygulanması

Bu veri tabanındaki balon dört farklı özelliği sahiptir. Bu özelliklerden her biri ise kendi içinde farklı özellikler içermektedir. Balonun sahip olduğu bu özellikler aşağıdaki gibi tanımlanmıştır [21, 22]:

x_1: renk (color) ε sarı, mor (yellow, purple)

x_2: boyut (size) ε geniş, küçük (large, small)

x_3: hareket (act) ε gergin, dalma (stretch, dip)

x_4: yaş (age) ε yetişkin, çocuk (adult, child)

Balon sahip olduğu özelliklere göre Tablo 5.3'te gösterildiği gibi 16X5 boyutunda bir matris verisi oluşturmaktadır. Oluşan bu veriler ise; 5. sütunda sahip olduğu özelliklere göre şişirilmiş ya da şişirilmemiş şekilde evet (E) veya hayır (H) olarak sınıflandırılmaktadır.

Tablo 5.3: Balon veri tabanının anlamsal özellikleri

Sıra No.	Renk	Boyut	Hareket	Yaş	Şişmiş
1	Sarı	Küçük	Gergin	Yetişkin	E
2	Sarı	Küçük	Gergin	Çocuk	E
3	Sarı	Küçük	Dalma	Yetişkin	E
4	Sarı	Küçük	Dalma	Çocuk	H
5	Sarı	Geniş	Gergin	Yetişkin	E
6	Sarı	Geniş	Gergin	Çocuk	E
7	Sarı	Geniş	Dalma	Yetişkin	E
8	Sarı	Geniş	Dalma	Çocuk	H
9	Mor	Küçük	Gergin	Yetişkin	E
10	Mor	Küçük	Gergin	Çocuk	E
11	Mor	Küçük	Dalma	Yetişkin	E
12	Mor	Küçük	Dalma	Çocuk	H
13	Mor	Geniş	Gergin	Yetişkin	E
14	Mor	Geniş	Gergin	Çocuk	E
15	Mor	Geniş	Dalma	Yetişkin	E
16	Mor	Geniş	Dalma	Çocuk	H

Tablo 5.3'teki anlamsal veriler, N-bit değer eşitliği problemine benzetilerek aşağıdaki gibi sayısallaştırılmıştır:

x_1: renk ε sarı, mor (1, 0)

x_2: boyut ε geniş, küçük (0, 1)

x_3: hareket ε gergin, dalma (1, 0)

x_4: yaş ε yetişkin, çocuk (1, 0)

x_5: şişmiş ε E, H (1, 0)

Balon probleminde Tablo 5.4'te görüldüğü gibi 4 farklı sınıflandırma veri seti bulunmaktadır. Tablo 5.3'deki E ve H olarak yapılan sınıflandırma aşağıdaki kuralara göre yapılmaktadır:

Veri Seti A: Eğer yaş=yetişkin ya da hareket=gergin İse şişmiş=E Değilse şişmiş=H

Veri Seti B: Eğer yaş=yetişkin ve hareket=gergin İse şişmiş=E Değilse şişmiş=H.

Veri Seti C: Eğer renk=sarı ve boyut=küçük İse şişmiş=E Değilse şişmiş=H.

Veri Seti D: Eğer (renk=sarı ve boyut=küçük) ya da (yaş=yetişkin ve hareket=gergin) İse şişmiş=E Değilse şişmiş=H.

Tablo 5.4: İkili sayı sistemine çevrilmiş olan balon veri setleri

Sıra No.	Veri Seti A	Veri Seti B	Veri Seti C	Veri Seti D
	x_1 x_2 x_3 x_4 x_5	X_1 x_2 x_3 x_4 x_5	x_1 x_2 x_3 x_4 x_5	x_1 x_2 x_3 x_4 x_5
1	1 1 1 1 1	1 1 1 1 1	1 1 1 1 1	1 1 1 1 1
2	1 1 1 0 1	1 1 1 0 0	1 1 1 0 1	1 1 1 0 1
3	1 1 0 1 1	1 1 0 1 0	1 1 0 1 1	1 1 0 1 1
4	1 1 0 0 0	1 1 0 0 0	1 1 0 0 1	1 1 0 0 1
5	1 0 1 1 1	1 0 1 1 1	1 0 1 1 0	1 0 1 1 1
6	1 0 1 0 1	1 0 1 0 0	1 0 1 0 0	1 0 1 0 0
7	1 0 0 1 1	1 0 0 1 0	1 0 0 1 0	1 0 0 1 0
8	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0
9	0 1 1 1 1	0 1 1 1 1	0 1 1 1 0	0 1 1 1 1
10	0 1 1 0 1	0 1 1 0 0	0 1 1 0 0	0 1 1 0 0
11	0 1 0 1 1	0 1 0 1 0	0 1 0 1 0	0 1 0 1 0
12	0 1 0 0 0	0 1 0 0 0	0 1 0 0 0	0 1 0 0 0
13	0 0 1 1 1	0 0 1 1 1	0 0 1 1 0	0 0 1 1 1
14	0 0 1 0 1	0 0 1 0 0	0 0 1 0 0	0 0 1 0 0
15	0 0 0 1 1	0 0 0 1 0	0 0 0 1 0	0 0 0 1 0
16	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0

Bu veri setlerinde kullanılan b parametresi N-bit değer eşitliği problemindeki b değerinin seçilme kurallarına göre belirlenmemiştir. Her bir veri seti için belirlenen kurallar ve seçilen parametreler alt bölümlerde verilmiştir.

5.2.1 Balon veri seti A

Veri seti A için eğer aşağıdaki gibi bir kural yazılmaz ve b=-2, t_i =0.5 ile Denklem 4.2 kullanılırsa 16 tane verinin 12 tanesi doğru olarak sınıflandırılmaktadır.

Böylelikle sistem başarımı % 75.000 olmaktadır. Fakat aşağıdaki gibi bir kural yazılması ile 16 verinin tamamı doğru olarak sınıflandırılmaktadır. Böylece sistem başarımı ise % 100.000 olmaktadır.

Kural: EĞER $x_3 = 0$ ya da $x_4 = 0$ İSE $b<0$ ve Denklem 4.2'yi kullan DEĞİLSE $b>0$ ve Denklem 4.2'yi kullan ($b=\pm 0.5$, $t_i=0.5$).

5.2.2 Balon veri seti B

Bu veri seti için kural yazılmadan önce $b=-2$, $t_i=0.5$ ile Denklem 4.2 kullanılırsa 16 verinin 4 tanesi doğru olarak sınıflandırılmaktadır. Sistem başarımı % 25.000 olmaktadır. Fakat aşağıdaki tek bir kural ile 16 verinin tümü doğru olarak sınıflandırılmakta ve sistem başarımı % 100 olmaktadır.

Kural: EĞER $x_3 = 0$ ve $x_4 = 0$ İSE $b<0$ ve Denklem 4.2'yi kullan DEĞİLSE $b>0$ ve Denklem 4.2'yi kullan ($b=\pm 0.5$, $t_i=0.5$).

5.2.3 Balon veri seti C

Bu veri setinde de kural yazılmadan önce $b=-2$, $t_i=0.5$ ve Denklem 4.2 kullanılırsa 16 verinin 4 tanesi doğru olarak sınıflandırılır. Sistem başarımım 25.000% olur. Kural yazılması halinde 16 verinin tamamı doğru olarak sınıflandırılmaktadır. Böylece sistem başarımı 100.000% olur.

Kural: EĞER $x_1 = 0$ ve $x_2 = 0$ İSE $b<0$ ve Denklem 4.2'yi kullan DEĞİLSE $b>0$ ve Denklem 4.2'yi kullan ($b=\pm 0.5$, $t_i=0.5$).

5.2.4 Balon veri seti D

Burada sistemin 16 verisinin tamamının doğru sınıflandırılması için 2 tane kural gerekmektedir. İlk kuralda sistem, 16 verinin 9 tanesini doğru olarak sınıflandırarak sistem başarımını % 56.250 yapmaktadır. Fakat Kural 1 ve Kural 2'yi birlikte kullanmamız halinde 16 verinin tamamı doğru sınıflandırılmakta ve sistem başarımı % 100.000 olarak gerçekleşmektedir. Ayrıca farklı b ve t_i parametrelerine göre sistem başarımlarının etkisi Tablo 5.5'te görülmektedir.

Kural 1: EĞER (($x_1 = 1$ ve $x_2 = 1$) ya da ($x_3 = 1$ ve $x_4 = 1$)) ve (($x_1 = 0$ ve $x_2 = 0$) ya

da ($x_1 = 0$ ve $x_2 = 1$) ya da ($x_3 = 1$ ve $x_4 = 0$)) İSE b<0 ve Denklem 4.2'yi kullan DEĞİLSE b>0 ve Denklem 4.2'yi kullan (b=±2, t_i=0.5).

Kural 2: EĞER ($x_1 = 0$ ve $x_2 = 0$) ya da ($x_3 = 0$ ve $x_4 = 0$) İSE b<0 Denklem 4.3' ü kullan (b=-2, t_i=0.5).

Tablo 5.5: Balon problemi için farklı b ve t_i parametrelerine göre sistem başarımı

b	t_i	Sistem Başarımı (%)	b	t_i	Sistem Başarımı (%)
	0.1	**100.00**		**0.1**	**100.00**
	0.2	**100.00**		**0.2**	**100.00**
	0.3	**100.00**		**0.3**	**100.00**
	0.4	**100.00**		**0.4**	**100.00**
± 1	0.5	75.00	**± 2**	**0.5**	**100.00**
	0.6	75.00		**0.6**	**100.00**
	0.7	75.00		**0.7**	**100.00**
	0.8	75.00		**0.8**	**100.00**
	0.9	75.00		**0.9**	**100.00**

5.3 π_m İşlem Birimi Modelinin Tic-Tac-Toe Problemine Uygulanması

Tic-Tac-Toe veri tabanı bir oyundan alınmış olan verilerden oluşmaktadır. Oyun aşağıdaki gibi 9 kareden oluşmaktadır [27, 28].

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

Şekil 5.1: Tic-Tac-Toe oyunu

Bu karelerin her biri 'x', 'o', 'b' değerini almaktadır. 'x' birinci oyuncuyu, 'o' ikinci oyuncuyu 'b' ise boşluğu ifade etmektedir. Oyuncular sırasıyla Şekil 5.1'deki karelerin her hangi birisine kendi harflerini yazarlar. Hangi oyuncu kendi işaretlediği

harfinden 3 sıra oluşturmuş ise oyunu kazanır. Bu 3 sıra bir sütunda, satırda veya çapraz da olabilir. Bu veritabanında 958 tane veri bulunmaktadır [21, 22]. Fakat bu verilere göre kazanan oyuncu 'x' harfine yani birinci oyuncuya göre belirlenmiştir, kazanma pozitif kaybetme de negatif olarak belirlenmiştir. 958 oyunda 626 tanesi pozitif yani 'x' kazanmıştır, 332 tanesi negatif yani 'x' kaybetmiştir. Bu verileri ikili sayı sisteminde ifade etmek için 'x' sembolü 1, 'o' ve 'b' sembolleri ise 0 seçilerek N-bit değer eşitliği problemine benzetimi yapılmıştır. Birinci oyuncunun ('x') kazanması durumunda ilgili veri kümesinin sonucu 1, aksi durumda is 0 seçilmiştir (x_{10}: pozitif, negatif (1,0)).

958X10 büyüklüğünde oluşan veri matrisi; sınıflandırılmak üzere π_m işlem birimi modeline uygulanmıştır. Burada da b parametresi N-bit değer eşitliği probleminde kullanılan b değerini seçme kuralına göre belirlenmemiştir. Eşik aktivasyon fonksiyonları $t_i = 0.5$ ile ilk önce b=-2 seçilerek Denklem 4.2 kullanılmıştır (b=-2, t_i=0.5). Bunun sonucunda 958 verinin 608 tanesi doğru olarak sınıflandırılmış ve başarım % 63.466 olmuştur. Daha sonra ise b<0 ve Denklem 4.3 kullanılmış (b=-2, t_i=0.5). Bu durumda ise 958 verinin 654 tanesi doğru olarak sınıflandırılmıştır. Sistem başarımı % 68.267 olmuştur. Veri tekrar incelendikten sonra her satırdaki 'b' (boşluk) sayısı belirlenmiş ve aşağıdaki kural yazılmıştır.

Kural: EĞER boşluk sayısı=2,3 İSE b<0 ve Denklem 4.2'yi kullan DEĞİLSE b>0 ve Denklem 4.2'yi kullan (b=±2, t_i=0.5).

Yukarıdaki kural kullanıldığı zaman 958 verinin 942 tanesi doğru olarak sınıflandırılmış ve başarım % 98.330 olarak gerçekleşmiştir. Bu problemi çözen diğer algoritmaların ve bu tez çalışmasında kullanılan algoritmanın başarım sırası Tablo 5.6'da görülmektedir.

Tablo 5.6'daki sonuçlara göre bu tez çalışmasında önerilen işlem birimi modeli Newboole ve IB3-CI (Instance Based 3- Constructive Induction) algoritmasından sonra en iyi sınıflandırma verimini sağlamaktadır. Newboole'yı Pierre Boneli ve Alexander Parodi 1991 yılında Stewart W. Wilson'ın Boole sınıflandırma sisteminden esinlenilerek geliştirilmiştir. Bu yeni sınıflandırma sistemi genetik tabanlı olup öğrenme algoritması olarak eğiticili öğrenmeyi kullanılmaktadır [29, 30].

IB3-CI örnek tabanlı bir algoritma olup; 1991 yılında Aha'nın Citre algoritmasından esinlenilerek oluşturmuş olduğu başka bir algoritmadır [31]. IN3-CI ENKK algoritmasını kullanır ve var olan özelliklerin bileşenlerini üretir. Bu bileşenlerde örnekleri pozitif ve negatif olarak eşleştirir. Böylece verilen örneklerin pozitif ve negatif olarak sınıflandırılmasını yapar.

Tablo 5.6: Tic-tac-toe problemi için farklı algoritmaların başarım sıralaması

Sıralama	**Algoritmalar**	**Başarım(%)**
1	NEWBOOLE	100.00
2	IB3-CI	99.10
3	**π_m İşlem Birimi Modeli**	**98.33**
4	IB1	98.10
5	CN2	98.10
6	IB3	82.00
7	MBRtalk	88.40
8	NewID	88.00

6. SONUÇLAR VE ÖNERİLER

N-bit değer eşitliği problemi her zaman YSA yapılarını ve öğrenme algoritmalarını test etmek için kullanılmıştır. Bunun sebebi bu tip bir problemin doğrusal olmamasıdır. Bu tür bir problem için; şimdiye kadar önerilen YSA yapılarında en az bir gizli katman ve öğrenme algoritması kullanılmıştır. π_t işlem birimi modeli ile bu yaklaşım ortadan kalkmıştır. Bu model ile parametreleri bir kez doğru seçildikten sonra tek bir çarpımsal işlem birimi kullanılarak her hangi bir öğrenme algoritmasına ihtiyaç olmadan N-bit değer eşitliği problemi çözülmüştür [20]. Böylelikle işlem hızı da yükselmiştir. Bu tez çalışmasında ise π_t işlem birimi modelindeki b parametresi, problemin özelliğine göre farklı işaret ve değerde seçilerek değiştirilmiş π_t (π_m) işlem birimi modeli kullanılmıştır. π_m işlem birimi modeli ile McCulloch-Pitts işlem birimi modeli göz önüne alınarak; bu iki modelin melez kullanımı amaçlanmıştır. Fakat problemin özelliğine göre bazen bu iki model birlikte bazen de yalnızca π_m işlem birimi modeli kullanılmıştır. Çalışmamızda uygulama olarak üç tip problem ve bunlara ait altı farklı veritabanı bulunmaktadır. Kullanılan veritabanlarının hepsi N-bit değer eşitliği problemine benzetmek için ikilik sayı sistemine çevrilerek sisteme uygulanmıştır.

İlk uygulama olarak Rahip problemi seçilmiştir. Bu problem için; π_m işlem birimi modeli ile birlikte McCulloch-Pitts işlem birimi modelinin melez kullanımı sağlanmıştır. Kullanılan bu melez yapı 5 adet kurallar ile birlikte veritabanındaki tüm örüntüleri doğru şekilde sınıflandırarak, sistem başarımı yaklaşık % 90 olarak gerçekleşmiştir. Bu tez çalışmasında M_2 probleminin çözümünde önerilen yöntem diğer farklı yöntemlerle karşılaştırıldığında sistem başarımı açısından 7. sırada olduğu görülmüştür. İkinci uygulama olarak Balon problemi seçilmiştir. Bu problem ise kendi içerisinde dört çeşit veri tabanı barındırmaktadır. Bu veri tabanlarından birinci, ikinci ve üçüncüsünde π_m işlem birimi modeli, dördüncüsünde ise π_m işlem birimi modeli ile birlikte McCulloch-Pitts işlem birimi modeli birlikte kullanılmıştır.

Veri tabanlarının hepsinde probleme uygun sırasıyla 1., 2., ve 3. veri tabanlarına tek bir kural, 4. veri tabanına 2 adet kural yazılarak sistem sınıflandırma başarımları 100% olarak gerçekleşmiştir. Bu tez çalışmasında balon problemi için sınıflandırma hatası 0.0 iken Soloria T. ve Feuntes O. Tarafından geliştirilen OC (Ordered Classification) algoritması balon probleminde 0.27930 hata ile sınıflandırma işlemini yapmaktadır [32].

Üçüncü uygulama olarak Tic-Tac-Toe problemi seçilmiştir. Bu problemde de π_m işlem birimi modeli ve tek bir kural kullanılarak, sistem başarımı %98.330 olmuştur. Diğer çalışmalarla karşılaştırıldığında yöntemin önerilen sistem başarımı 3. sırada yer almaktadır. Bu tez çalışmasında kullandığımız otomatik öğrenme veri tabanları ile ilgili daha önceden yapılan çalışmaların hepsinde bilinen yöntemler kullanılmıştır. Önceki çalışmalarda kullanılan YSA yapılarında en az bir gizli katmanda iki ya da üç işlem birimi ve öğrenme algoritmaları kullanılırken, bu tez çalışmasında ise başlangıçta parametreleri en uygun şeklide seçilmiş tek bir işlem birimi kullanılarak her hangi bir öğrenme algoritmasına gereksinim duymadan uygun çözümler üretmektedir. Böylelikle işlem hızı yüksek olarak gerçekleşmektedir. Bu tez çalışmasının dezavantajı ise, probleme özgü kuralların otomatik olarak oluşturulmamış olmasıdır.

Gelecek çalışmalar için, burada yazılan kuralları otomatik oluşturan bir çalışma yapılması önerilmektedir. Örneğin bulanık mantık kural çıkartım sistemi kullanılarak kuralların otomatik olarak oluşturulması sağlanabilir.

Bu tez çalışmasında önerilen işlem birimi modelinde öğrenme bulunmamaktadır. Fakat sınıflandırmaya başlamadan önce b ve t_i parametrelerinin uygun biçimde seçim işlemi söz konusudur. Öneri olarak yazılan algoritma probleme göre bir ön işlemden geçirilerek kural eklenmesi gerekip gerekmediğine ve b ve t_i parametrelerinin hangi değerde seçilmesi gerektiğine karar verecek şekilde düzenlenmesi ileriki çalışmalar için önerilmektedir.

KAYNAKLAR

[1] Alpaydın E. "Introductions to Machine Learning", ***Mit Pres***, 3-6, (2004).

[2] Andrew W. Moore, 2001, Support Vector Machines, School of Computer Science Carnegie Mellon University, http://www.autonlab.org/tutorials/svm15.pdf (**Erişim tarihi: 09 Ekim 2006).**

[3] Hollemen, J., Tresp, V., Simula, O. "A Learning Vector Quantization Algorithm For Probabilistic Models", ***In Proceedings of EUSIPCO 2000 – X European Signal Processing Conference***, ***Vol. II***, 721-724, (2000).

[4] Nilsson, N. J., "Introduction to Machine Learning", 81-83, December 4, (1996).

[5] Han, j., Kamber, M., "Data Mining Concepts and Techniques", ***Morgan Kaufman Publishers***, 1 st Ed, San Francisco, USA, (2000).

[6] Kanungo, T., Member S., Mount, D., M., Netanyahu N., S., Piatko, C., D., Silverman, R., Wu, A., Y., "An Efficient k-Means Clustering Algorithm: Analysis and Implementation", ***IEEE Transaction on Pattern and Machine Intelligence***, Vol.24, No.7, 881-892, July (2002).

[7] Nabiyev, V. V., "Yapay Zeka", İkinci Baskı, ***Seçkin Yayıncılık***, (2005).

[8] McCulloch, W. S., Pitts, W., "A logical Calculos of the Ideas Immanent in Nervous Activity", ***Bulletin of Mathematical Biophysics***, ***Vol. 5***, 115-133, (1943)

[9] Widrow, B., Hoff, M., E., "Adaptive switching circuits", ***IRE Western Electric Show and Convention Record***, ***Part 4***, 96-104, (1960)

[10] Kohonen, T., "The Self Organizing Map", ***Proc. of the IEEE, Vol. 78, No. 9*** (1990)

[11] Haykin, S., "Neural Networks", Second edition, ***Simon & Schuster / A Viacom Company"***, 10-12, (1999).

[12] Elmas, Ç., "Yapay Sinir Ağları", Birinci baskı, ***Seçkin Yayıncılık***, (2003)

[13] Brown, D. A., "N-bit Parity Networks", ***Neural Networks***, 607-607, (1987).

[14] Giles, C. L. ve Maxwell, T., "Learning, invariance, and generalization in high-order neural networks", ***Applied Optics*** 26(23), 4972-4978, (1987).

[15] Zhang, B.-T., "A Bayesian evolution approach to the design and learning of heterogeneous neural trees", ***Integrated Computer-Aided Engineering*** 9(1), 73-86, (2002).

[16] Yadow, R., N., Kalra, P., K., Jhon J., "On the Use of Multiplicative Neuron in Feedforward Neural Networks", ***International Journal of Modelling and Simulation***, ***Vol. 26***, ***No. 4***, (2006)

[17] Iyoda, E. M., Hirota, K ve Von Zuben, F. J., "Sigma-Pi cascade hybrid neural networks, ***Journal of Advanced Computational Intelligence*** 6, 126-134, (2002).

[18] Iyoda, E., M., Nobuhara, H., ve Hrota, K., "A solution for N-bit parity problem using a single multiplicative neuron", ***Neural Processing Letters***, 18, 213-218, (2003).

[19] Iyoda, E., M., Nobuhara, H., ve Hirota, K., "An extended multiplicative neuron that can translate decision surfaces", ***Journal of Advanced Computational Intelligence and Intelligent Informatics***, ***Vol. 8***, ***No. 5***, pp. 460-468, (2004).

[20] Dazi, L., Hirasawa, K., Jinglu, H., ve Murata, J., "Studying the Effects of Multiplication Neurons for Parity Problem", ***Sice 2002 5-7***, 2678-2681, Osaka, (2002).

[21] University of Californaia, Irvine, ftp://ftp.ics.edu/pub/machine-learning-databases/, **(Ziyaret tarihi: 5 Kasım 2006).**

[22] Yıldız, O., T., Machine Learning Datasets, Boğaziçi Üniversitesi, http://haydut.cmpe.boun.edu.tr/datasets.htm, **(Ziyaret tarihi: 5 Kasım 2006).**

[23] Thrun, S., B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., Jong, K., D., Dzeroski, S., Fahlman, S., E., Fisher, D., Hamann, R., Koufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R., S., Mitchell, T., Pachowicz, P., Reaich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., ve Zhang, J., "The Monk's Problems: A performance Comparison of Different Learning Algorithms", ***A Report, Carnegie Mellon University CMU-CS-91-197***, (1991).

[24] Michalski, R., S. "On the Quasi-Minimal Solution of the Covering Problem", ***The V. International Symposium on Information Processing (FCIP)"***, Vol. A3 (Switching Circuits), 125-123, Bled, Yugoslavia, (1969).

[25] Mozetic, I., "Newgem: Program for learning from Examples, Program Documentation and User's Guide", ***Report No. UIUCDCS-F-85-949, Department of Computer Science, University of Illinois, Urbana,*** (1971).

[26] Fahlman, S., E., Lebiere, C., "The Cascade-Correlation Learning Architecture", ***Advances inNeural Information Processing System 2***, 524-532, (1990).

[27] Pilgrim, R., A., "Tic-Tac-Toe: Introduction Expert Systems to Middle School Students", ***Acm Sigcse Bulletin***, ***Vol. 27***, 340–344, (1995).

[28] Gordon, A., "A General Algorithm for Tic-Tac-Toe Board Evaluation", ***Journal of Computing Sciences in Colleges***, ***Vol. 21***, 42-46, (2006)

[29] Sandip, S., "Representational effects in a simple classifier system", ***The 1994 ACM Symposium on Applied Computing***, 206-211, (1994).

[30] Sandip S., "A tale of two representations", ***7th International Conference and Engineering Applications of Artificial Intelligence and Expert System***, 245-254, (1994)

[31] Markovitch, S., Rosentein, D., "Feature Generation Using General Constructor Functions", ***Machine Learning***, ***Vol. 49***, 59-98, (2002).

[32] Solorio, T., Fuentes, O., "Taking Advantage of Unlabeled Data with the Ordered Classification Algorithm", ***ACTA, Proc. of AI and Soft Computing ASC 2002***, (2002).

EK A: PROGRAM KODLARI

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <alloc.h>
#include <time.h>
#include <dos.h>
#include <conio.h>

#define katibsay 10 //Her Katmandaki İşlem Birimi Sayısı belirlenir
#define katsay 10  // Katman Sayısı.
#define sat 169
#define sut 7
#define sut1 25

struct katmantip{                    //*****Katmanlar Arası*******
  int say;                           //Katmanlardaki Nöron Sayısını verir.
  float cikis[katibsay];             //Katmanlardaki Çıkışı verir.
  float t[katibsay][katibsay];       //Katmanlardaki ağırlıkları verir.
  int trans[katibsay];               //Katmanda Kullanılan Transfer Fonk. Belirler.
   };

struct nettip{
   int  say;
   struct katmantip katman[katsay];
   };

int i,j,k;
int sec;
struct nettip net;
float ver[sat][sut];
float veri[sat][sut1];
float hedef;
float x[sut1];                //Ysa'ya Giriş verilerini alırken kullanılır.
float b;
char tc[2];


//****Transfer Fonk. Kullanarak Net  Çıkışı Hesaplar*******
float fx(float y, int tr)
{
 float sonuc;
  switch (tr)
```

```
   {
   case 1:
     sonuc=y;
     break;
    case 2:
     if (y>=0)
           sonuc=1-1/(1+y);
     else
           sonuc=-1+1/(1-y);
     break;
    case 3:  //Eşik aktivasyon fonksiyonu
     if (y>=0)
           sonuc=1;
     else
           sonuc=0;
     break;
   }
 return sonuc;
}

//*****İleri Doğru Net Çıkışı Hesaplar**********
void hesapla_sec1()
{
 int i,j,k;
 float carpim;
 for (k=0 ;k<(net.say-1);k++)
   {
    for(j=0;j<net.katman[k+1].say;j++)
     {
        carpim=b*(-1);
         for (i=0;i<net.katman[k].say;i++)
           {
             carpim *=(net.katman[k].cikis[i]-net.katman[k+1].t[j][i]);
           }
           tc[0]='C';tc[1]='-';
          net.katman[k+1].cikis[j]=fx(carpim,net.katman[k+1].trans[j]);
          }
   }
}

void hesapla_sec2()
{
 int i,j,k;
 float carpim;
 for (k=0 ;k<(net.say-1);k++)
   {
    for(j=0;j<net.katman[k+1].say;j++)
     {
        carpim=b;
```

```
            for (i=0;i<net.katman[k].say;i++)
              {
                carpim *=(net.katman[k].cikis[i]-net.katman[k+1].t[j][i]);
              }
              tc[0]='C';tc[1]='+';
             net.katman[k+1].cikis[j]=fx(carpim,net.katman[k+1].trans[j]);
             }
    }
}

void hesapla_sec3()
{
  int i,j,k;
  float toplam;
  for (k=0 ;k<(net.say-1);k++)
    {
     for(j=0;j<net.katman[k+1].say;j++)
       {
          toplam=b*(-1);
           for (i=0;i<net.katman[k].say;i++)
             {
                toplam +=(net.katman[k].cikis[i]*net.katman[k+1].t[j][i]);
              }
              tc[0]='T';tc[1]='-';
             net.katman[k+1].cikis[j]=fx(toplam,net.katman[k+1].trans[j]);
             }
    }
}

void hesapla()
{
 switch (sec)
 {
  case 1:
    hesapla_sec1();
    break;
  case 2:
    hesapla_sec2();
    break;
  case 3:
    hesapla_sec3();
    break;
  }
}
//***Giriş verilerini Giriş Katmanına aktarır*****
void set_giris()
{
int i;
  for (i=0; i<(net.katman[0].say); i++)
```

```
   {
     net.katman[0].cikis[i]=x[i];
   }
}
void agirlik()
{
  int i,j,k;
  float a,c;
   for(k=1;k<net.say;k++)
    {
      for (j=0;j<net.katman[k].say;j++)
       {
            for(i=0;i<net.katman[k-1].say;i++)
              {
              // net.katman[k].t[j][i]=fabs(((float)(rand()%10000))/10000-0.5);
                 net.katman[k].t[j][i]=0.5;
              }
      }
    }
}
// Verileri Bir Metin Dosyasından Okur
void oku_veri()
{
  int i,j,k,a,sec;
     FILE *dosya;
     dosya=fopen("c:\\tc\\work\\monks\\veri\\monks2.txt","r");
   if(!dosya)
     {
      printf("Dosya(monks2.txt) açılamadı.....\n");
     }
    else
     {
      printf("Dosya(monks2.txt) açıldı....\n");
     }
       for (i=0;i<sat;i++)
            {
              a=0;
              for (j=0;j<sut;j++)
              {
                fscanf(dosya,"%f ",&ver[i][j]);
                sec=ver[i][j];
                 if(j==6)
                  {
                      veri[i][24]=ver[i][j];
                  }
                 else
                  {
                      switch (sec)
                       {
```

```
                    case 1:
                       veri[i][j+a+0]=0;
                       veri[i][j+a+1]=0;
                       veri[i][j+a+2]=0;
                       veri[i][j+a+3]=0;
                       break;
                    case 2:
                       veri[i][j+a+0]=0;
                       veri[i][j+a+1]=0;
                       veri[i][j+a+2]=0;
                       veri[i][j+a+3]=1;
                       break;
                    case 3:
                       veri[i][j+a+0]=0;
                       veri[i][j+a+1]=0;
                       veri[i][j+a+2]=1;
                       veri[i][j+a+3]=0;
                       break;
                    case 4:
                       veri[i][j+a+0]=0;
                       veri[i][j+a+1]=0;
                       veri[i][j+a+2]=1;
                       veri[i][j+a+3]=1;
                       break;
                   }
                }
              a+=3;
             }
            }
 fclose(dosya);
}

void transfertipleri()
{
 int i;
   for (k=1;k<net.say;k++)
    {
      for (i=0;i<net.katman[k].say;i++)
       {
            net.katman[k].trans[i]=3;
       }
    }
}

void tablo()
{
int i,tp=0;
FILE *cikis;
```

```
cikis=fopen("c:\\tc\\work\\monks\\veri\\mnk2ext.txt","w");

 if(!cikis)
   {
    printf("Dosya(mnk2ext.txt) açılamadı.....\n");
   }
  else
   {
    printf("Dosya(mnk2ext.txt) açıldı....\n");
   }

 for (i=0;i<sat;i++)
   {
    if (ver[i][4]==4)
        {
         if ((ver[i][0]==1&ver[i][1]==1&ver[i][2]==1&ver[i][3]!=1)
           | (ver[i][0]==3&ver[i][1]!=1&ver[i][2]!=1& ver[i][3]!=1&ver[i][5]!=1)
           | (ver[i][0]==3&ver[i][1]==1&ver[i][2]==1&ver[i][3]==1&ver[i][5]==1)
           | (ver[i][0]==2&ver[i][1]==2&ver[i][2]==2& ver[i][3]!=1)
           | ((ver[i][0]==2|ver[i][0]==3)&( ver[i][1]==2|ver[i][1]==3)
            &(ver[i][2]==2|ver[i][2]==3)&(ver[i][3]==2|ver[i][3]==3)&ver[i][5]==2))
           sec=2;
         else
           sec=1;
        }
    else
        {
         if (ver[i][0]==3|ver[i][1]==3|ver[i][3]==3|ver[i][4]==3)
           {
            if((ver[i][0]==1&ver[i][1] ==1&ver[i][2]==1&ver[i][5]==1)
            | ((ver[i][0]==3&ver[i][5]==2)&(ver[i][1]==2|ver[i][1]==3)
              &(ver[i][2]==2|ver[i][2]==3)&(ver[i][3]==2|ver[i][3]==3)
              &(ver[i][4]==2|ver[i][4]==3))
            | ((ver[i][0]==3|ver[i][0]==2)&(ver[i][1]!=2&ver[i][2]!=2)&ver[i][3]==1
              &ver[i][4]==1&ver[i][5]==1)
            | (ver[i][0]==2&ver[i][1]==3&ver[i][2]==2&ver[i][3]!=1&ver[i][5]!=1)
            | (ver[i][0]!=3&ver[i][1]!=2&ver[i][2]==1&ver[i][3]==1& ver[i][5]!=2)
            | (ver[i][0]==3&ver[i][1]==1&ver[i][4]==1&ver[i][5]==1)
            | ((ver[i][0]<3&ver[i][1]>1&ver[i][3]!=2&(ver[i][4]==1|ver[i][4]==2))
              &((ver[i][2]==ver[i][4])&(ver[i][4]==ver[i][5])))
            |(ver[i][0]==1&ver[i][1]==3&ver[i][2]==1&ver[i][3]==1&ver[i][4]==1
              &ver[i][5]==2))
                 sec=1;
             else
                 sec=2;
            }
           else
            {
             if ((ver[i][0]==ver[i][1])&(ver[i][1]==ver[i][2])
```

```
                &(ver[i][2]==ver[i][3])&(ver[i][3]==ver[i][4])&(ver[i][4]= =ver[i][5]))
                    sec=1;
              else if ((ver[i][0]==ver[i][1])&(ver[i][1]==ver[i][2])
                        &(ver[i][2]==ver[i][4])&(ver[i][4]==ver[i][5])
                        &(ver[i][3]!=ver[i][5]))
                    sec=2;
               else
                     sec=3;
              }
           }
             fprintf(cikis,"%d ",i+1);

           for(k=0;k<sut-1;k++)
             fprintf(cikis,"%0.0f ",ver[i][k]);

       for(j=0;j<sut1-1;j++)
            {
             x[j]=veri[i][j];
             //fprintf(cikis,"%0.0f ",x[j]);
             }
             hedef=veri[i][24];
       set_giris();
       hesapla();

            if (net.katman[net.say-1].cikis[0]==hedef )
              tp+=1;
           fprintf(cikis,"Parity..:%0.0f Hedef.:%0.0f %s \n",net.katman[net.say-
1].cikis[0],hedef,tc);
     }
    fprintf(cikis,"**************************** \n");
    fprintf(cikis,"Veri say.:%d,Doğru say.:%d,Yüzde:%0.3f\n",sat,tp,(tp*100.00)/sat);
fclose(cikis);
printf("Veriler Dosyaya Kayıt Edildi.\n");
}

void main()
{
 int tp=0;
 b=2;
 net.say=2;
 net.katman[0].say=sut1-1;
 net.katman[1].say=1;
 clrscr();
 agirlik();
 transfertipleri();
 oku_veri();
    for(i=0;i<sat;i++)
     {
      if (ver[i][4]==4)
```

```
    {
     if ((ver[i][0]==1&ver[i][1]==1&ver[i][2]==1&ver[i][3]!=1)
       | (ver[i][0]==3&ver[i][1]!=1&ver[i][2]!=1&ver[i][3]!=1&ver[i][5]!=1)
       | (ver[i][0]==3&ver[i][1]==1&ver[i][2]==1&ver[i][3]=1&ver[i][5]==1)
       | (ver[i][0]==2&ver[i][1]==2&ver[i][2]==2&ver[i][3]!=1)
       | ((ver[i][0]==2|ver[i][0]==3)&( ver[i][1]==2|ver[i][1]==3)
        &(ver[i][2]==2|ver[i][2]==3)&(ver[i][3]= =2|ver[i][3]==3)&ver[i][5]==2))
        sec=2;
     else
        sec=1;
    }
else
    {
      if (ver[i][0]==3|ver[i][1]==3|ver[i][3]==3|ver[i][4]==3)
       {
        if ((ver[i][0]==1&ver[i][1]==1&ver[i][2]==1&ver[i][5]==1)
         | ((ver[i][0]==3&ver[i][5]==2)&(ver[i][1]==2|ver[i][1]==3)
           &(ver[i][2]==2|ver[i][2]==3)&(ver[i][3]==2|ver[i][3]==3)
           &(ver[i][4]==2|ver[i][4]==3))
         | ((ver[i][0]==3|ver[i][0]==2)&(ver[i][1]!=2&ver[i][2]!=2)&ver[i][3]==1
          & ver[i][4]= =1 & ver[i][5]= =1)
         | (ver[i][0]==2&ver[i][1]==3&ver[i][2]==2&ver[i][3]!=1&ver[i][5]!=1)
         | (ver[i][0]!=3&ver[i][1]!=2&ver[i][2]==1&ver[i][3]==1& ver[i][5]!=2)
         | (ver[i][0]==3&ver[i][1]==1&ver[i][4]==1&ver[i][5]==1)
         | ((ver[i][0]<3&ver[i][1]>1&ver[i][3]!=2&(ver[i][4]==1|ver[i][4]==2))
          &((ver[i][2]==ver[i][4])&(ver[i][4]==ver[i][5])))
         | (ver[i][0]==1&ver[i][1]==3&ver[i][2]==1&ver[i][3]==1&ver[i][4]==1
          &ver[i][5]= =2))
              sec=1;
        else
              sec=2;
        }
       else
        {
         if ((ver[i][0]==ver[i][1])&(ver[i][1]==ver[i][2])
             & (ver[i][2]==ver[i][3])&(ver[i][3]==ver[i][4])
             & (ver[i][4]==ver[i][5]))
               sec=1;
         else if ((ver[i][0]==ver[i][1])&(ver[i][1]==ver[i][2])
                & (ver[i][2]==ver[i][4])&(ver[i][4]==ver[i][5])
                & (ver[i][3]!=ver[i][5]))
                sec=2;
         else
                sec=3;
        }
    }
    printf("%d ",i+1);

    for(k=0;k<sut-1;k++)
```

```
            printf("%0.0f ",ver[i][k]);

          for(j=0;j<sut1-1;j++)
            {
            x[j]=veri[i][j];
            }
            hedef=veri[i][24];
          set_giris();
          hesapla();
          printf("P:%0.0f T:%0.0f %s \n",net.katman[net.say-1].cikis[0],hedef,tc);
             if (net.katman[net.say-1].cikis[0]==hedef)
                   tp+=1;
    }
printf("Veri Say.:%d, Doğru Say.:%d Yüzde.:%0.3f\n",sat,tp,(tp*100.00)/sat);
tablo();
getch();
}
```

ÖZGEÇMİŞ

1975 yılında Mersinin Tarsus İlçesinde doğdu. İlk ve orta öğrenimini Tarsus'ta tamamladı. Lise eğitimini Adana Teknik Lisesi Elektronik bölümünde tamamladı. 1993 yılında Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümünde yüksek öğrenimine başladı. 1998 yılında Bilgisayar Öğretmeni olarak bu bölümden mezun oldu. 2000 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'nda Yüksek Lisans eğitimine başladı. 2001 yılında yurt dışına dil eğitimine gittiğinden dolayı öğrenimine ara verdi. 2005 yılında Kocaeli Üniversitesindeki Yüksek Lisans eğitimine geri döndü. 2005 yılından beri Milli Eğitim Bakanlığına bağlı bir okulda Bilgisayar Öğretmeni olarak çalışmaktadır.