İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KARINCA KOLONİSİ OPTİMİZASYONU İLE EĞİTİLMİŞ ÇOK KATMANLI YAPAY SİNİR AĞI İLE SINIFLANDIRMA

YÜKSEK LİSANS TEZİ
Hale Hilal DODURGALI

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Bilgisayar Mühendisliği

OCAK 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KARINCA KOLONİSİ OPTİMİZASYONU İLE EĞİTİLMİŞ ÇOK KATMANLI YAPAY SİNİR AĞI İLE SINIFLANDIRMA

YÜKSEK LİSANS TEZİ
Hale Hilal DODURGALI
(504051513)

Tezin Enstitüye Verildiği Tarih : 25 Aralık 2009
Tezin Savunulduğu Tarih : 29 Ocak 2010

Tez Danışmanı : Prof.Dr. Muhittin GÖKMEN (İ.T.Ü)
Diğer Jüri Üyeleri : Prof.Dr. Coşkun SÖNMEZ (Y.T.Ü.)
Yrd. Doç.Dr. Şima ETANER UYAR (İ.T.Ü.)

OCAK 2010

ÖNSÖZ

Mühendislik öğrenimim ve tez çalışmam boyunca değerli fikir ve yorumlarıyla beni yönlendiren, cesaretlendiren tez danışmanım Prof. Dr. Muhittin GÖKMEN'e, tez hazırlama sürecinde öncelikle dostluğunu, daha sonra da ilgi ve yardımını esirgemeyen arkadaşlarım Yük. Müh. Ayhan YÜKSEL'e ve Yük. Müh. Mihrican ÖZTÜRK'e, her konuda olduğu gibi tez çalışmamda da en büyük desteğim olan, babam Prof. Dr. Abdurrahman DODURGALI başta olmak üzere tüm aileme, sevgisi, ilgisi ve sabrıyla hep yanımda olan eşim Gültekin YİĞİT'e, teşekkürlerimi sunarım.

Aralık 2009

Hale Hilal DODURGALI

Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

ABC : Artificial Bee Colony
ACO_R-ÇKA : Ant Colony Optimization (ACO) ile Eğitilmiş Çok Katmanlı Ağ
ADALINE : Adaptive Lineer Element
AS_{RANK} : Mertebe Temelli Karınca Sistemi
BP-ÇKA : Geri Yayılım ile Eğitilmiş Çok Katmanlı Ağ
ÇKA : Çok Katmanlı Yapay Sinir Ağı
DEA : Doğa Esinli Algoritmalar
GetÇKA : Genetik Çok Katmanlı Ağ
GMM : Gizli Markov Modeli
GSP : Gezgin Satıcı Problemi
HAS-QAP : Hibrid Karınca Sistemi ile KAP probleminin çözümü
KAP : Karesel Atama Problemi
KKO : Karınca Kolonisi Optimizasyonu
KKS : Karınca Koloni Sistemi
KLD : Karhunen- Loeve Dönüşümü
KNN : K-En Yakın Komşu Sınıflandırıcısı
KS : Karınca Sistemi
MM-KS : Min- Max Karınca Sistemi
OKT : Optik Karakter Tanıma
PDF : Olasılık – Yoğunluk Fonksiyonu
PSO : Parçacık Sürü Optimizasyonu
RT-YSA : Radyal Temelli Yapay sinir Ağı
UF : Uyum Fonksiyonu
YSA :Yapay Sinir Ağı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

KARINCA KOLONİSİ OPTİMİZASYONU İLE EĞİTİLMİŞ ÇOK KATMANLI YAPAY SİNİR AĞI İLE SINIFLANDIRMA

ÖZET

Tez çalışmasında ilk olarak yapay sinir ağları üzerine yapılan bilimsel çalışmalar incelenmiş, çok katmanlı yapay sinir ağının, doğrusal olmama, öğrenme, genelleme, hata toleransı gibi üstün özellikleri nedeniyle sınıflandırma ve tanıma problemlerinde sıklıkla kullanıldığı bilgileri elde edilmiştir. Tez çalışması ile, çok katmanlı bir yapay sinir ağı tasarlanmış ve sınıflandırıcı olarak kullanılmıştır.

Çok katmanlı yapay sinir ağının sınıflandırma başarımı, büyük oranda ağın iyi eğitilmiş olmasına bağlıdır. Yapılan araştırmalarda, çok katmanlı ağ eğitiminde geleneksel olarak geri yayılım algoritmasının kullanıldığı, ancak bu algoritmanın zayıf yönleri olduğu görülmüştür. Ağın yerel çözümlerde takılabilmesi ve ağdaki düğüm sayılarının ihtiyaca göre belirlenememesi nedeniyle, çok katmanlı ağın eğitimi için geri yayılım algoritması yerine farklı algoritmalar önerilmiştir.

Çok katmanlı ağın eğitimi, bir en iyileme problemi olarak görülmüş ve bu amaçla en iyileme problemlerinin çözümünde kullanılan doğa esinli algoritmalar incelenmiştir. Bazı doğa esinli algoritmaların, çok katmanlı ağın ve diğer yapay sinir ağlarının eğitiminde kullanıldığı görülmüştür. Tez çalışmasında, en başarılı doğa esinli algoritmalardan biri olan karınca kolonisi optimizayonu algoritması, literatürde ilk defa, çok katmanlı ağın eğitiminde kullanılmıştır.

Eğitim sırasında, karınca kolonisi optimizasyonu algoritması kullanılarak, çok katmanlı ağın birinci katmanındaki ağırlıkların en iyi değerleri elde edilmeye çalışılmıştır. Ayrıca bu algoritmanın, ağın katmanlarındaki düğüm sayılarını, problemin türüne göre belirlemesi sağlanmıştır. Bu yöntemle eğitilen çok katmanlı ağın sınıflandırma yeteneği, öncelikle bilinen bazı veri kümelerinde (IRIS ve tic-tac-toe) test edilmiştir. Aynı veri kümeleri üzerinde, geri yayılım algoritması ve genetik algoritmalar ile eğitilmiş çok katmanlı ağlar ve k-en yakın komşu sınıflandırıcısı da test edilmiştir. Bu sınıflandırıcıların başarımı, karınca kolonisi algoritması kullanılarak oluşturulan yeni ağın başarımıyla karşılaştırılmıştır.

Tez çalışmasında, en zor sınıflandırma problemlerinden biri olarak görülen el yazısı karakter tanıma problemi ve yapay sinir ağlarının bu amaçla kullanımları incelenmiştir. Elde edilen yeni ağ kullanılarak, el yazısı rakamlar sınıflandırılmaya çalışılmıştır.

Pen-digit, Optical-Digit, İTÜ elyazısı türkçe karakter veri kümesi ve NIST rakam veri kümesi olmak üzere dört ayrı el yazısı rakam veri kümesi kullanılmıştır. İTÜ ve NIST veri kümelerinde, örnekler el yazısı rakamların görüntülerinden oluşmaktadır. Çok katmanlı ağın giriş uzayının boyutunu azaltmak amacıyla, bu görüntülerden üç farklı öznitelik vektörü elde edilmiştir. Karhunen-Loeve dönüşümü ile elde edilen öznitelik vektörü, rakamların şekilsel özellikleri kullanılarak elde edilen öznitelik

vektörü ve iki öznitelik vektörü birleştirilerek elde edilen birleşik öznitelik vektörü, belirtilen sınıflandırıcılara ayrı ayrı sunulmuş ve başarımları karşılaştırılmıştır.

Tez çalışması ile literatüre yeni bir çok katmanlı ağ ve eğitim yöntemi kazandırılmıştır. Geliştirilen yöntemle, çok katmanlı ağın birinci katman ağırlıklarının ve bu katmandaki düğüm sayısının en iyi değerlerinin elde edilmesi ve bu yolla sınıflandırma başarımının artması beklenmektedir.

CLASSIFICATION WITH MULTILAYER PERCEPTRON TRAINED WITH ANT COLONY OPTIMIZATION

SUMMARY

In this thesis study, fist of all, all applications of artificial neural networks were investigated. Multilayer perceptron were most preferred neural network for the solution of classification and recognition problems because of its specifications like nonlinearity, learning, generalization and error tolerance. A multilayer perceptron is designed and used as classifier in this thesis study.

Classification performance of the multilayer perceptron is strongly depended on training performance of it. Investigations about multilayer perceptron shows that, back propagation algorithm is most common multilayer perceptron training algorithm. This algorithm has some poorness about saving the neural network from local minimums and determining the neuron numbers of each network layer according to problem definition. Because of this poorness, different training algorithms have been suggested instead of back propagation.

Multilayer perceptron training has been thought as a kind of optimization problem so that nature inspired algorithms which are used to solve optimization problems were investigated. It is observed that some of the nature inspired algorithms were used for training neural networks. In this study, most successful nature inspired algorithm, ant colony optimization, is used for multilayer perceptron training first time in literature.

In the training stage, the first layer weights of the multilayer perceptron are optimized by ant colony optimization algorithm. Additionally, number of the neurons in first layer is determined by the algorithm according to problem. Classification performance of new multilayer perceptron is tested with known classification datasets like IRIS, Tic-Tac-Toe. Multilayer perceptrons trained with back propagation and genetic algorithms and k-nearest neighbor classifier are also tested with the same datasets. Performances of all classifiers are compared.

In this study, one of the hardest classification problems, handwritten character recognition problem and usage of multilayer perceptrons for this problem are researched. Handwritten digits are tried to classify using new multilayer perceptron trained by ant colony optimization.

PenDigit, Optical Digit, ITU Turkish Chracter Dataset and NIST Handwritten Digit Dataset are used for testing. Samples in ITU and NIST datasets consist of handwritten digit images. Feature extraction methods are used to decrease the input space dimensions and three different feature vectors are obtained. Karhaugnen-Loeve transformation feature vector, geometrical feature vector and combination of two feature vectors are introduced to classifiers separately and performance of all classifiers are compared.

In this study, a new multilayer perceptron and a training algorithm are provided to literature. With the new method, extraction the optimum weights and determination neuron numbers, as a result higher classification performance are expected.

1. GİRİŞ

İnsanoğlunun mağara duvarlarına hayvan figürleri çizerek başladığı veri işleme süreci dört bin yıl gibi uzun bir gelişim sürecinin ardından, son 50 yıl içindeki çalışmalarla günümüz teknolojisine ulaşmıştır. Veri işleme, artık gözle görülmeyecek kadar küçük işlemci birimleriyle ve son derece hızlı bilgisayar yazılımları ile yapılabilmektedir. Bu gelişim sürecinin mimarı olan insan zekası, bu noktada atılan en büyük adımlardan biri olarak kendini taklit etme fikrini ortaya atmış ve makinelere de zeka verme uğraşısı içine girmiştir. Zeka, insanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma, soyutlama, öğrenme ve yeni durumlara uyum sağlama yeteneklerini kapsayan bir ifadedir. Zeka, karmaşık problemleri çözmek için gerekli bilgileri toplayıp birleştirme ve kısa yoldan çözüme ulaşma kabiliyetidir. Bu yetenekleri makinelere de sağlama uğraşısı yapay zeka kavramını ortaya çıkarmıştır. Yapay zeka konusu, felsefeden bilgisayar bilimine, elektrik-elektronikten biyoloji ve psikolojiye kadar birçok bilimin ortak üretimleri ile gelişmektedir. Yapay zeka araştırmaları ile öğrenebilen, sezme, tahminleme ve çıkarsama yapabilen, öngörü, sınıflama ve kontrol kabiliyetleri olan, insan davranışlarını örnekleyebilen ve karar verebilen makinelerin üretilebilmesi amaçlanmaktadır.

Turing makinesi (1950) ile başlayan yapay zeka serüveni, Genel Sorun Çözücü (General Problem Solver - 1959), STRIPS (Stanford Research Institute Problem Solver - 1971) gibi yöntemlerin, bir doğal dil işleyici olan ELIZA (1966) gibi programların üretilmesiyle devam etmiştir [1]. 80'li yıllardan itibaren robotik sistemlerin ve uzman sistemlerin geliştirilmesi, yapay zeka alanında atılmış önemli adımlar olmuş ve büyük ticari başarılar kazanmıştır. 21. yy ile birlikte, bilgisayar bilimine ait bir çok zorlu problemin çözümünde yapay zeka uygulamaları kullanılmaya başlanmıştır. Yapay zeka uygulamaları ile çözülmeye çalışılan başlıca bilimsel problemler şunlardır.

- Arama ve eniyileme problemleri
- Otomatik mantık üretme problemleri

- Sınıflandırma ve istatistiksel öğrenme problemleri
- Yapay sinir ağları
- Kontrol problemleri

Günümüzde yapay zeka alanında tüm bu problemlerin çözülmesi, var olan çözümlerin iyileştirilmesi amacıyla çok sayıda farklı çalışma ve araştırmalar yapılmaktadır. Yapay Sinir Ağları (YSA) ve eniyileme problemleri, yapay zekanın en çok araştırma ve geliştirme yapılan alanlarından olmuştur.

1.1 Yapay Sinir Ağları ve Kullanım Alanları

YSA, insan beyninin bir işlevi yerine getirme yöntemini modellemek için tasarlanan bir sistem olarak tanımlanabilir. Bilindiği gibi; öğrenme, hatırlama, düşünme gibi tüm insan davranışlarının temelinde sinir hücreleri bulunmaktadır. İnsan beyninde tahminen 10^{11} adet sinir hücresi olduğu düşünülmektedir ve bu sinir hücreleri arasında sonsuz diyebileceğimiz sayıda sinirler arası bağ vardır. YSA da, beyine benzer şekilde, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar şeklinde düzenlenir. Donanım olarak elektronik devrelerle ya da bilgisayarlarda yazılım olarak gerçeklenebilir. YSA, günümüzde makinelerin öğrenmesini sağlayan en güçlü yöntemlerdendir.

YSA'nın tarihçesi insanların nörobiyoloji konusuna ilgi duyması, araştırmalar yoluyla insan beyni ve çalışma şekli hakkında bilgiler elde etmesi ve elde ettiği bilgileri bilgisayar bilimine uygulamaları ile başlamaktadır. Beyin, öğrenme, öğrenilen bilgiyi hafızada tutma ve genelleme yoluyla benzer bir durum için kullanma ilkeleri ile çalışmaktadır. Beynin bilgi işleme yöntemine uygun olarak YSA, bir öğrenme sürecinden sonra bilgiyi toplama, hücreler arasındaki bağlantı ağırlıkları ile bu bilgiyi saklama ve genelleme yeteneğine sahip, paralel dağılmış bir işlemci olarak çalışır.

YSA'yı oluşturan hücrelerde bilgi toplama işlemi, dış dünyadan verilen ya da diğer hücrelerden gönderilen girdilerin alınmasıyla gerçekleşir. Bu girdiler, ağın öğrenmesi istenen örneklerdir. Hücreye gelen her girdi, taşınan bilginin önemini ve hücre üzerindeki etkisini gösteren bir ağırlığa sahiptir. Hücre, kendisine gelen tüm girdi ve ağırlıkları birleştirerek net girdi hesabı yapar. Elde edilen bu net girdiye karşılık, hücre bir çıkış üretir. Bu çıkış, dış dünyaya yanıt olarak ya da başka bir hücreye girdi

olarak sunulur. Bu sistemin çalışmasında en önemli süreç YSA'nın öğrenmesidir. Öğrenme süreci, arzu edilen amaca ulaşmak için YSA ağırlıklarının yenilenmesini sağlayan öğrenme algoritmalarını içerir.

Günümüzde YSA, doğrusal olmama, öğrenme, genelleme, bilgiyi saklama, görülmemiş örnekler hakkında bilgi üretebilme, eksik bilgi ile çalışabilme, sınıflandırma, uyarlanabilme, hata toleransı gibi üstün özellikleri nedeniyle birçok bilim alanında etkin olmuş ve uygulama yeri bulmuştur.

YSA, arıza analizi ve tespiti alanında, elektrik makinelerinin, uçakların yada bileşenlerinin, bütünleşmiş devrelerin arıza analizinde, tıp alanında EEG ve ECG gibi tıbbi sinyallerin analizi, kanserli hücrelerin analizi, protez tasarımı gibi konularda, savunma sanayinde, silahların otomasyonu, nesne ayırma ve tanıma, algılayıcı tasarımı konularında, haberleşme alanında, görüntü ve veri sıkıştırma, gerçek zamanlı konuşma çevirisi gibi konularda, üretim alanında, ürün analizi ve tasarımı, kalite kontrolü, planlama ve yönetim analizi gibi konularda, otomasyon sistemlerinde, robot sistemi kontrolü, otomatik yol bulma ve gösterme gibi konularda, işaret ve görüntü işleme, örüntü tanıma problemlerinde etkin olarak kullanılmaktadır.

YSA, yapısına ve öğrenme kurallarına göre çeşitlilik gösterir. Çok katmanlı ağlar, Hopfield ağı, Neocognitron ağı, Kohonen ağı, Bellek tabanlı ağlar, İleri beslemeli ve geri beslemeli ağlar, Büyü ve öğren ağı gibi birçok YSA modeli uygulamalarda yer almaktadır.

YSA, birçok konuda, birçok yönden tercih edilir durumda olmasına rağmen, bazı problemleri ve iyileştirmeye açık yönleri bulunmaktadır. Her problem için, problemin türüne göre ağın yapısı değişmektedir. YSA ile problem çözülmeden önce, uygun ağ yapısının belirlenmesi gerekmektedir. Benzer şekilde ağın parametre değerlerinin başlangıçta belirlenmesi gerekmektedir. Bu parametreler, katman sayısı, katmanlardaki hücre sayısı, öğrenme katsayısı gibi ağın çalışmasını ve elde edeceği sonucu doğrudan etkileyen değerlerdir. Ağın eğitim süreci de problemin türüne göre belirlenmelidir. Ağ problem üzerinde ne kadar iyi eğitilebilirse, ağdan elde edilecek sonuç da o kadar iyi olacaktır.

YSA eğitiminde günümüze kadar birçok algoritmadan yararlanılmıştır. Adaline algoritması, geriye yayılım algoritması ve varyasyonları bu amaçla üretilmiştir. Son

zamanlarda ise, çeşitli eniyileme problemlerinde kullanılan doğa esinli algoritmalar, yine bir eniyileme problemi olan YSA eğitiminde kullanılmaya başlanmıştır.

1.2 Doğa Esinli Algoritmalar ve Karınca Kolonisi Optimizasyonu

1980'lerin başında, zeki makineler üretmeyi amaçlayan yapay zeka araştırmacıları biyolojik olayları ve canlıların doğal yaşam süreçlerini gözlemlemeye başlamıştır. Elde edilen bilgilerden esinlenilerek yeni algoritmalar ve yazılımlar ortaya konmuştur. Özellikle biyolojik kalıtım, genetik ve evrimleşme ilkeleri ve sosyal kolonilerin hayatta kalma, yiyecek bulma, haberleşme yöntemleri, birçok algoritma için esin kaynağı olmuştur. Bu yolla oluşturulan algoritmalar doğa esinli algoritmalar olarak adlandırılır.

Doğa esinli algoritmalar aşağıdaki gibi sınıflandırılabilir.

- Evrimsel (genetik) algoritmalar
- Sürü zekası algoritmaları
- Sinirsel hesaplama algoritmaları

Evrimsel hesaplama, arama, eniyileme, öğrenme gibi problemlere biyolojik populasyon genetiği kurallarına dayanarak yaklaşan hesaplama tekniklerini içerir. Evrimsel hesaplamalar ilintilendirilmelerine göre Genetik Algoritmalar, Evrimsel Programlama, Evrim Yönelimleri ve Genetik Programlama gibi değişik isimlerle anılırlar.

Sürü zekası algoritmaları, doğada sürü ve koloniler şeklinde yaşayan canlıların yaşama ilkelerine dayanarak çeşitli problemlere çözüm üretirler. Balıklar, karıncalar, kuşlar, arılar gibi birçok hayvanın yaşam süreci modellenerek sürü zekası algoritmaları ortaya konmuştur. Parça Sürü Optimizasyonu, Karınca Kolonisi Optimizasyonu, Yapay Arı Kolonileri bu algoritmalara örnek olarak verilebilir.

Doğa esinli algoritmalar özellikle arama, iyileştirme, karar verme ve öğrenme gibi temel yapay zeka problemlerinin çözümünde sıkça kullanılmaktadır. Özellikle geniş çaplı çözüm uzayına sahip problemlerde, geleneksel arama tekniklerinin yetersiz kaldığı durumlarda, sezgisel arama teknikleri devreye girmektedir. Bu tekniklerin geleneksel tekniklere karşı olan üstünlüğü, arama sırasında seçim yaparak ve işe yaramayan çözümlerden sakınarak, elde edilen sonuçları her aşamada iyileştirerek,

kötü sonuçlar bulunması durumunda da arama yönünü değiştirerek ilerliyor olmalarıdır.

Yapay zeka kavramında bulunan, insan zekası modelleme odaklı, karmaşık, merkezî, planlı yaklaşımların aksine, sürü zekası, basit yapılı, özerk, önceden planlama yapmayan dağınık aracıların, karmaşık problemlerin çözümünde başarılı olduklarını göstermiştir. Bunların en başarılıları ise Karınca Kolonisi Algoritmaları ile Parçacık Sürü Optimizasyonu (Particle Swarm Optimization) algoritmalarıdır. Bu algoritmaların benzerlerine olan üstünlükleri ve gelişime açık olmaları sürü zekasının, yapay zekanın gittikçe önem kazanan ve gelişen bir konusu olmasını sağlamıştır [2].

Karınca Kolonisi Algoritmaları, gerçekte kör olan karıncaların yiyecek kaynağına giden en kısa yolu bulma yetenekleri [3] modellenerek geliştirilmiş algoritmalardır. 1991 yılında M. Dorigo ilk karınca kolonisi optimisazyonu algoritmasını doktora tezi olarak ortaya koymuştur. Daha sonra bu algoritma üzerinde çeşitlemeler ve iyileştirmeler yapılmış ve karmaşık problemlerin çözümünde kullanılmıştır. Gezgin satıcı problemi, karesel atama problemi karınca kolonisi algoritmaları ile çözümlenmiş problemlerden bazılarıdır. Çeşitli arama ve eniyileme problemlerinin çözümünde, karınca kolonisi algoritmalarının kullanımı giderek yaygınlaşmaktadır.

Birçok doğa esinli algoritma ile birlikte KKO algoritmalarının da denendiği önemli bir eniyileme problemi, YSA'ya ait parametrelerin eniyilenmesi problemidir.

1.3 KKO Algoritması ile YSA Eğitimi

YSA parametrelerinin eniyilenmesi, aslında, YSA'nın eğitilmesi anlamına gelmektedir. Daha önce bahsedildiği gibi, YSA, girdi olarak aldığı bilgilere ağırlık ataması yapar. Bu ağırlıklar sayesinde bilgileri öğrenir ve hafızasında saklar. Ağa verilen eğitim örneklerine göre değerler alan bu ağırlıklar, ağın istenen sonuca yaklaşmasını doğrudan etkiler. Bu nedenle ağın eğitiminde kullanılan algoritma, ağırlıkları aşamalı olarak değiştirmeli ve eniyi değerlerine ulaştırmalıdır.

Doğa esinli algoritmalar, sezgisel yollarla çözüm havuzları oluşturması, bu çözümlerden iyi olanları saklayıp, kötü olanları elemesi, çözüm uzayının değişik noktalarında arama yapabilmesi, aday çözümleri evirebilmesi gibi özellikleri nedeniyle son yıllarda YSA eğitiminde kullanılmaktadır. Radyal Tabanlı Ağın

genetik algoritmalar, parçacık sürü optimizasyonu algoritması ve karınca kolonisi algoritması ile eğitilmesi, çok katmanlı YSA'nın yine genetik algoritma ile eğitilmesi bu alanda yapılmış çalışmalardır.

Tez çalışmasında, Karınca Kolonisi algoritması, çok katmanlı YSA eğitiminde ilk kez kullanılarak bu konuda seçilebilir bir çözüm sunulmaya çalışılmıştır. ACO_R-ÇKA olarak isimlendirilen bu yeni YSA, çeşitli sınıflandırma problemlerinde test edilerek, seçilen algoritmanın YSA eğitimindeki başarısı ölçülmüştür. Ayrıca, geri yayılım algoritması ile eğitilen çok katmanlı YSA (BP-ÇKA), genetik algoritma kullanılarak eğitilen çok katmanlı YSA (GetÇKA) ve en yakın komşu sınıflandırıcıları (kNN) aynı problemler üzerinde çalıştırılmıştır. Bu sınıflandırıcılar ve ACO_R-ÇKA ile elde edilen sonuçlar karşılaştırılmıştır. Kullanılan sınıflandırıcıların, sınıflandırma becerisinin test edilmesi için önce el yordamıyla oluşturulmuş veri kümeleri, daha sonra IRIS, Tic-Tac-Toe gibi bilinen veri kümeleri kullanılmıştır. Tez çalışmasında, ayrıca, karmaşık ve yaygın bir sınıflandırma problemi olan el yazısı rakamların tanınması amacıyla ACO_R-ÇKA, BP-ÇKA, GetÇKA ve kNN sınıflandırıcıları kullanılmış ve başarımları karşılaştırılmıştır. El yazısı rakam veri kümeleri olarak Pen-digit, Opt-digit, İTU ve NIST veri kümeleri kullanılmıştır.

Bu tezin ikinci bölümünde YSA tarihçesi, YSA yapıları, eğitim algoritmaları, kullanım alanları, üçüncü bölümünde doğa esinli algoritmalar, dördüncü bölümünde YSA eğitim algoritması olarak seçilen karınca kolonisi algoritmalarının ortaya çıkışı, algoritma üzerinde yapılan geliştirmeler, çeşitli problemlere uygulanışı, beşinci bölümünde, yapay zeka ve örüntü tanıma problemi olan el yazısı karakter tanıma problemi ve çözüm aşamaları ile ilgili bilgiler verilmiştir. Altıncı bölümde tez çalışmasında kullanılan sınıflandırıcılar ve öznitelik çıkarma yöntemleri tanıtılmıştır. karınca kolonisi algoritması ile çok katmanlı YSA eğitiminde kullanılan yöntem ve ayrıntıları anlatılmıştır. Yedinci bölümde, kullanılan veri kümeleri ve bu kümeler üzerinde elde edilen deneysel sonuçlar verilmiştir. Sekizinci bölümde ise elde edilen sonuçlar ve ileride yapılabilecek çalışmalar yorumlanmıştır.

2. YAPAY SİNİR AĞLARI

Doğadaki canlıların görsel sınıflandırma ve tanıma konularındaki yetenekleri, en gelişmiş teknolojilerle karşılaştırıldığında çok daha güçlüdür. Canlıların görsel yeteneği, makinelerden çok daha hızlı ve doğru karar üretebilmektedir. Bu üstün yetenek bilim adamları tarafından taklit edilerek, geliştirilen modeller problemlerin çözümünde kullanılmak istenmiştir. İnsanlığın doğayı araştırma ve taklit etme çabalarının en son ürünlerinden bir tanesi, Yapay Sinir Ağları (YSA) teknolojisidir. Beynin üstün özelliklerine ve canlıların görsel yeteneklerine ulaşabilmek için, beynin, nörofiziksel yapısından esinlenerek matematiksel modeli çıkarılmaya çalışılmıştır. Beynin bütün davranışlarını tam olarak modelleyebilmek için fiziksel bileşenlerinin doğru olarak modellenmesi gerektiği düşüncesi ile çeşitli yapay hücre ve ağ modelleri geliştirilmiştir. Böylece bilgisayarların geleneksel algoritmik hesaplama yönteminden farklı olarak Yapay Sinir Ağları ortaya çıkmıştır.

Yapay Sinir Ağı, basit işlem birimlerinden oluşan, deneyimsel bilgileri biriktirmeye yönelik doğal bir eğilimi olan ve bunların yoğun bir şekilde paralelliği sağlanmış bir işlemci olarak tanımlanabilir. Bu yüksek paralellik, YSA'na, yüksek bir hesaplama gücü ve büyük verileri hızlı işleme özelliği kazandırmaktadır [4].

YSA'nın hesaplama ve bilgi işleme gücünü, paralel dağılmış yapısından, öğrenebilme ve genelleme yeteneğinden aldığı söylenebilir. Genelleme, eğitim ya da öğrenme sürecinde karşılaşılmayan girişler için de YSA'nın uygun tepkileri üretmesi olarak tanımlanır. Bu üstün özellikleri, YSA'nın karmaşık problemleri çözebilme yeteneğini gösterir. Günümüzde birçok bilim alanında YSA, aşağıdaki özellikleri nedeniyle etkin olmuş ve uygulama yeri bulmuştur.

Doğrusal Olmama: YSA'nın temel işlem elemanı olan hücre, doğrusal değildir. Dolayısıyla hücrelerin birleşmesinden meydana gelen YSA da doğrusal değildir ve bu özellik bütün ağa yayılmış durumdadır. Bu özelliği ile YSA, doğrusal olmayan karmaşık problemlerin çözümünde en önemli araç olmuştur.

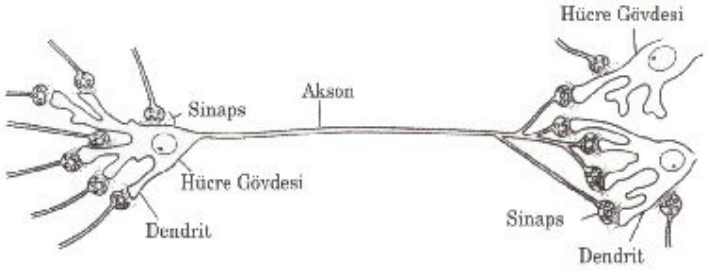
Öğrenme: YSA'nın arzu edilen davranışı gösterebilmesi için amaca uygun olarak ayarlanması gerekir. Bu, hücreler arasında doğru bağlantıların yapılması ve bağlantıların uygun ağırlıklara sahip olması gerektiğini ifade eder. YSA'nın karmaşık yapısı nedeniyle bağlantılar ve ağırlıklar önceden ayarlı olarak verilemez ya da tasarlanamaz. Bu nedenle YSA, istenen davranışı gösterecek şekilde ilgilendiği problemden aldığı eğitim örneklerini kullanarak problemi öğrenmelidir.

Genelleme: YSA, ilgilendiği problemi öğrendikten sonra eğitim sırasında karşılaşmadığı test örnekleri için de arzu edilen tepkiyi üretebilir. Örneğin, karakter tanıma amacıyla eğitilmiş bir YSA, bozuk karakter girişlerinde de doğru karakterleri verebilir ya da bir sistemin eğitilmiş YSA modeli, eğitim sürecinde verilmeyen giriş sinyalleri için de sistemle aynı davranışı gösterebilir.

Uyarlanabilirlik: YSA, ilgilendiği problemdeki değişikliklere göre ağırlıklarını ayarlar. Yani, belirli bir problemi çözmek amacıyla eğitilen YSA, problemdeki değişimlere göre tekrar eğitilebilir ve değişimler devamlı ise gerçek zamanda da eğitime devam edilebilir. Bu özelliği ile YSA, uyarlamalı örnek tanıma, sinyal işleme, sistem tanılama ve denetim gibi alanlarda etkin olarak kullanılır.

Hata Toleransı: YSA, çok sayıda hücrenin çeşitli şekillerde bağlanmasından oluştuğu için paralel dağılmış bir yapıya sahiptir ve ağın sahip olduğu bilgi, ağdaki bütün bağlantılar üzerine dağılmış durumdadır. Bu nedenle, eğitilmiş bir YSA'nın bazı bağlantılarının hatta bazı hücrelerinin etkisiz hale gelmesi, ağın doğru bilgi üretmesini önemli ölçüde etkilemez. Bu nedenle, geleneksel yöntemlere göre hatayı tolere etme yetenekleri son derece yüksektir [5].

YSA'nın oluşturulması için örnek teşkil eden, biyolojik sinir ağlarının ve insan beyninin en temel parçaları, hatırlama, düşünme, her harekette daha önceki deneyimlere başvurma yeteneğini sağlayan kendine özgü sinir hücreleridir. Aşağıdaki şekil basitleştirilmiş biyolojik bir beyin sinir hücresi ve bileşenleri ile ilişkisini göstermektedir [6].



Şekil 2.1: Beyin sinir hücresi

YSA, bir bütün olarak canlı sinir sisteminin modellenmiş halidir. Bu modelde, işlem elemanı nöronları temsil eder. Toplama fonksiyonu dendriti, aktarım fonksiyonu hücre gövdesini temsil eder. İşlem elemanlarının çıkışı aksonların, ağırlıklar ise sinapsların modellenmiş halidir.

YSA'nı oluşturan işlem elemanları, gerçek nöronlar gibi davranır. Diğer işlem elemanlarından (nöronlardan) ve yerel bellekten (sinaps- ağırlıklar) gelen bilgileri (sinyaller) toplama işleviyle (dendrit) birleştirir, aktarım işleviyle (hücre gövdesi) dönüştürür ve sayısal bir çıkış (akson) üretir.

YSA, matematiksel olarak, paralel bilgi işleme özelliğine sahip yönlü bir çizgedir. Bu çizge aşağıdaki tanımlama ve kısıtlamalara sahiptir.

a) Yönlü çizgenin düğümleri işlem elemanı olarak tanımlanır.

b) Yönlü dallar, bağlantılara karşılık düşer ve tek yönlü işaret iletim yolu olarak çalışırlar.

c) Her bir işlem elemanı, belirli sayıda giriş bağlantısına sahiptir.

d) Her bir işlem elemanı, belirli sayıda çıkış bağlantısına sahiptir. Ancak çıkış işaretlerinin değeri aynı olmalıdır.

e) İşlem elemanları yerel belleklere sahip olabilir.

f) Her işlem elemanı, giriş işaretini ve yerel belleği kullanan bir aktarım fonksiyonuna sahiptir. Bu fonksiyon işlem elemanının çıkış değerini oluşturur. Aktarım fonksiyonu sürekli veya ayrık olarak çalıştırılabilir. Ayrık çalıştırma modunda fonksiyon, bir aktif işareti ile kontrol edilir [7].

İşlem elemanları bir ağ içinde birbirleri ile çeşitli şekillerde bağlanırlar. Genelde katmanlar şeklinde de YSA'nı oluştururlar. Tüm katmanlardaki işlem elemanlarında

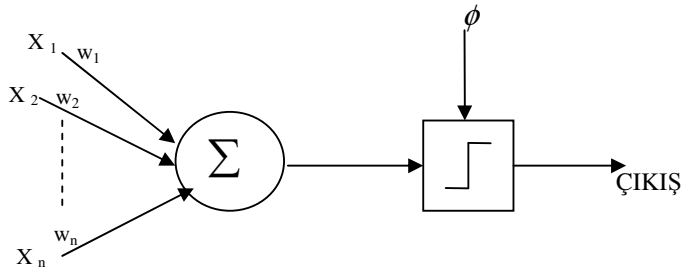
aynı aktarım fonksiyonu bulunur. Aynı ya da farklı katmanlardaki işlem elemanları birbirleri ile bağlantı oluşturabilirler.

2.1 Yapay Sinir Ağı Modelleri

2.1.1 Perceptron modeli

1943 yılında Mc Culloch ve Pitts tarafından nöronların ilk matematiksel modeli geliştirilmiştir. Bu model girişlerin ağırlıklandırılmış toplamını bulan basit bir işlemcidir. Bu toplam, belirli bir değerle karşılaştırılır ve doğrusal olmayan bir sonuç üretilir [8].

Bu nöron modelinden sonra 1958 yılında Frank Rossenblatt tarafından perceptron modeli geliştirilmiştir. Perceptron, YSA'da en temel yapıdır. Model üç katmandan oluşur. Birinci katman, retina, ikinci katmana giden dağıtılmış girişlerden oluşur. İkinci katman, birleşme üniteleri (association units), girişleri ağırlıklar ile birleştirir ve bir basamak fonksiyonundan geçirerek son katmana iletir. Üçüncü katman, çıkış katmanı (output layer), değerleri birleştirir [9]. Perceptron, birden fazla giriş, bir çıkış, bir aktarım fonksiyonu ve giriş sayısı kadar yerel bellek elemanından oluşur.



Şekil 2.2: Perceptron modeli

Aşağıdaki eşitlikte, x_i giriş vektörünün i-nci elemanını, w_i yerel belleğin i-nci elemanını ve n boyutu göstermek üzere, perceptronun matematiksel ifadesi verilmiştir.

$$y = \sum_{i=0}^{n} w_i x_i \quad , \quad x_0 = 1 \tag{2.1a}$$

$$y' = F(y) = \begin{cases} 1 & y \geq 0 \\ 0 & y < 0 \end{cases} \tag{2.1b}$$

y' İfadesinde *F(y)* doğrusal olmayan bir aktarım fonksiyonudur. İfadede bu fonksiyon hard-limiter olarak seçilmiştir.

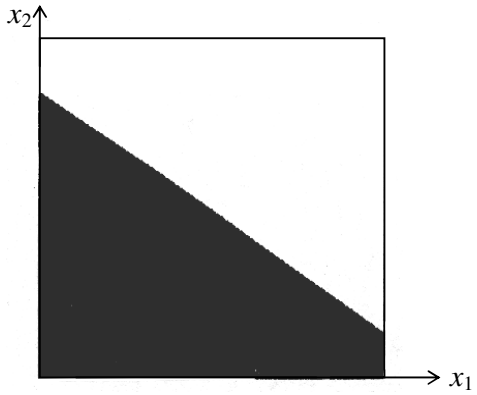
Perceptronda aktarım fonksiyonu olarak, doğrusal olmayan üç fonksiyondan biri kullanılır. Bunlar, hard-limiter, sigmoid ve kernel adı verilen fonksiyonlardır.

Hard-limiter fonksiyonunun matematiksel ifadesi aşağıdaki gibidir.

$$Y' = F(Y) = \begin{cases} 1 & Y \geq 0 \\ 0 & Y < 0 \end{cases} \quad \textbf{(2.2)}$$

Aktarım fonksiyonu olarak, hard-limiter fonksiyonu kullanan bir perceptron, iki boyutlu giriş uzayını iki parçaya ayırır. Bu iki parça 0 / +1 ya da -1/+1 değerleri ile temsil edilir.

Aşağıdaki şekil hard-limiter fonksiyonun perceptronun kararına yaptığı etkiyi göstermektedir.

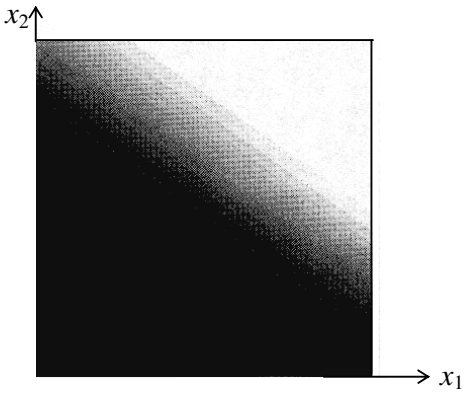


Şekil 2.3: Hard-limiter fonksiyonunun perceptron kararına etkisi

Perceptronda, aktarım fonksiyonu olarak kullanılabilen sigmoid fonksiyonunun tanımı aşağıda verilmiştir.

$$Y' = F(Y) = \frac{1}{1+e^{-Y}} \quad \textbf{(2.3)}$$

Sigmoid fonksiyonu, giriş uzayını üç parçaya ayırır. Aşağıdaki şekilde bu parçalar, doğruya paralel olarak 0 (veya -1) değeri alan bölge, 0 (veya -1) ile +1 arasında değer alan bölge, +1 değerini alan bölge olarak gösterilmektedir.

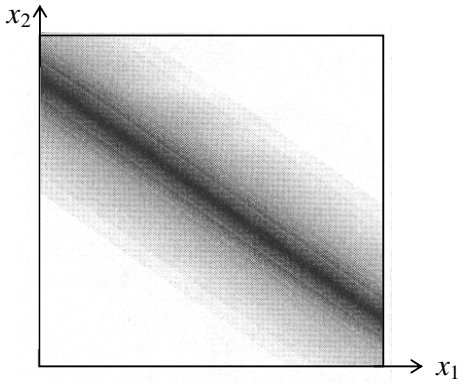


Şekil 2.4: Sigmoid fonksiyonunun perceptron kararına etkisi

Kernel adı verilen aktarım fonksiyonun tanımı ise aşağıdaki gibidir.

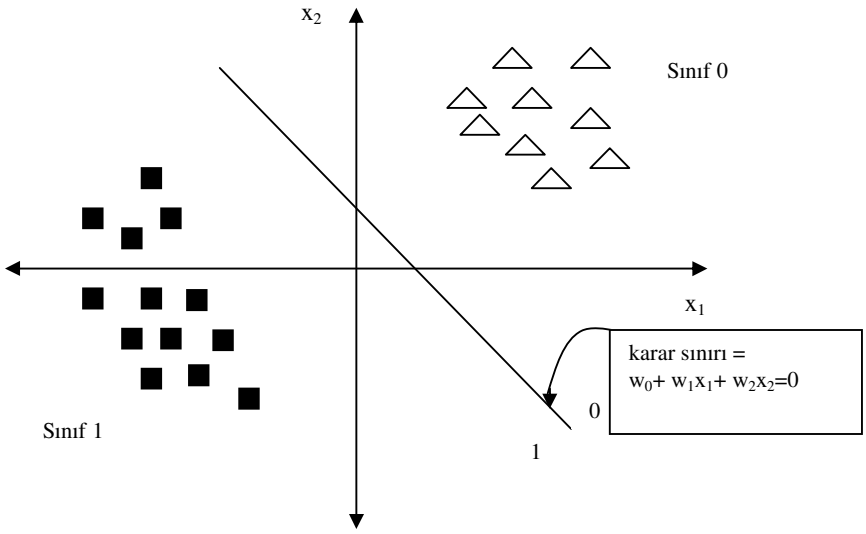
$$Y' = F(Y) = \exp\left(\frac{\| Y \|}{h}\right) \quad \textbf{(2.4)}$$

h bu tanımda zayıflatma katsayısı olarak isimlendirilir. Kernel fonksiyonu giriş uzayını +1 değerini alan bölge ve 0 ile+1 arasında değer alan bölge olmak üzere iki parçaya ayırır. Aşağıdaki şekil kernel fonksiyonunun düzlemi bölmelemesini göstermektedir.



Şekil 2.5: Kernel fonksiyonunun perceptron kararına etkisi

Perceptronlar, denklem 2.1'de verilen matematiksel ifadeyi kullanarak, YSA için belirlenen amaç ölçütünü sağlamaya çalışırlar. Aşağıdaki şekilde iki boyutlu bir örnek uzay gösterilmiştir. Örnek uzayda üçgen sınıfı ve kare sınıfı olmak üzere iki sınıf bulunmaktadır.



Şekil 2.6: Perceptronun 2-boyutlu örnek uzayı sınıflaması

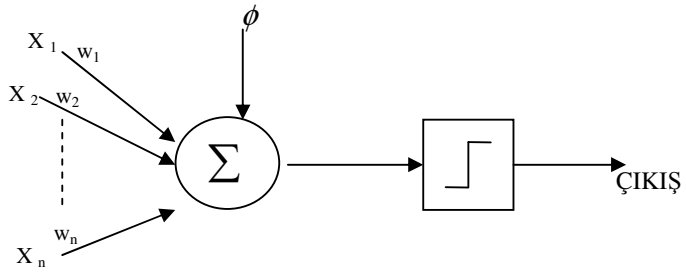
Perceptronun amaç ölçütünün, bu iki sınıfın birbirinden ayrılması olarak seçildiği varsayılsın. Percpetronun amaç ölçütünü sağlaması için önce eğitilmesi gerekir. Eğitim ya da öğrenme işlemi, amaç ölçütünü en iyi sağlayacak w_i yerel bellek elemanlarının değerlerinin bulunması işlemidir. Bir eğitim algoritması seçilerek w_i değerleri her adımda değiştirilir ve en iyi değerlerine ulaşılmaya çalışılır. Bulunan w_i değerleri işlem elemanının ağırlık vektörü olarak tanımlanır.

Eğitim işleminde perceptrona bir eğitim kümesi hazırlanır. Eğitim kümesi olarak her iki sınıftan da eşit sayıda vektörler alınır. Bu vektörler perceptrona giriş olarak uygulanır. Başlangıçta ağırlık değerleri (w_i) rasgele alınır. Her bir örnek vektör verildiğinde ağırlık değerleri yenilenir. Yenilenen ağırlık değerleri ile perceptron sonuç üretir. Üretilen sonuç, eğitim kümesiyle birlikte verilen sınıf bilgisiyle uyuşuyorsa, bulunan ağırlık değerleri saklanır. Böylece, belirlenen eğitim algoritması, bu örnek vektörlerden ve sınıf dağılım bilgisinden yararlanarak, ölçüt amacını sağlayacak en iyi w_i değerlerini bulmaya çalışır. Şekil 2.6'da, iki sınıfı ayıran doğru (karar sınırı), hesaplanan w_i değerleri ile oluşmuştur.

2.1.2 ADALINE modeli

ADALINE (ADAptive LINear Element) modeli, perceptrona çok benzeyen ancak öğrenme algoritması daha gelişmiş olan bir YSA modelidir. 1959 yılında Bernand Widrow ve Marcian Hoff tarafından geliştirilmiştir [10]. Öğrenme algoritması en küçük ortalamaların karesi (least mean square) yöntemine dayanmaktadır. Öğrenme

kuralı, ağın çıktısının beklenen çıktı değerine göre hatasını en azlayacak şekilde ağın ağırlıklarının değiştirilmesi prensibine dayanmaktadır. x_i giriş değerleri, w_i ağırlık değerleri, φ eşik değeri olmak üzere, aşağıdaki şekil ADALINE ünitesini göstermektedir.



Şekil 2.7: ADALINE ünitesi

ADALINE ünitesinde, önce ağın net girdisi hesaplanır.

$$Y = \phi + x_1 w_1 + x_2 w_2 + ... + x_n w_n \quad \textbf{(2.5)}$$

Ağın çıkışı hesaplanır.

$$Y' = \begin{cases} +1 & Y \geq 0 \\ -1 & Y < 0 \end{cases} \quad \textbf{(2.6)}$$

Ağın çıktısı için beklenen değerin V olduğunu varsayalım. ADALINE ağının sonuç ürettikten sonraki hatası E şu şekilde bulunur.

$$E = V - Y' \quad \textbf{(2.7)}$$

Öğrenme algoritmasında amaç, E değerini en küçük yapacak w_i değerini bulmaktır. Bu amaçla her yeni eğitim vektöründe çıkan sonuçla E hesaplanır, aşağıdaki formülle w_i değerleri E'yi azaltacak şekilde değiştirilir.

$$w_{i+t} = w_i + \alpha E x_i \quad \textbf{(2.8)}$$

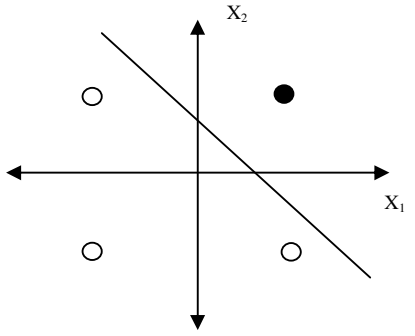
Burada α öğrenme katsayısı olarak isimlendirilir. Ağırlık değerlerine benzer şekilde eşik değeri de (φ) yenilenir.

$$\phi_{i+t} = \phi_i + \alpha E \quad \textbf{(2.9)}$$

2.2 Çok Katmanlı Yapay Sinir Ağları

Doğrusal sınıflandırma problemlerinde perceptron başarılı sonuçlar üretebilmektedir. Eğer (x_1,x_2) düzleminde bir hat varsa, iki sınıfa ait iki boyutlu vektörler (x_1,x_2) doğrusal olarak ayrılabilir. Mantıksal *VE* (AND) ve *VEYA* (OR) problemleri doğrusal problemlerdir.

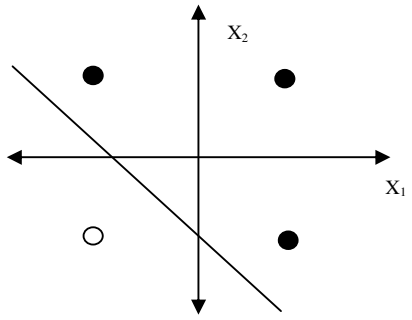
X_1	X_2	Y
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1



Şekil 2.8: Mantıksal VE (AND) problemi

Amaç ölçütü olarak VE problemini çözümleyen bir perceptron, eğitildiğinde x_0=1,x_1,x_2 girişleri ile ağırlık değerleri w_0=-1, w_1=1, w_2=1 olarak bulunur.

X_1	X_2	Y
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



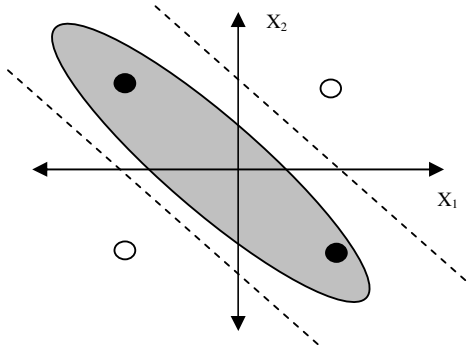
Şekil 2.9: Mantıksal VEYA (OR) problemi

Amaç ölçütü olarak VEYA problemini çözümleyen bir perceptron, eğitildiğinde x_0=1,x_1,x_2 girişleri ile ağırlık değerleri w_0=1, w_1=1, w_2=1 olarak bulunur.

1969 yılında Minsky ve Papert, tek katmanlı bir nöronun (perceptron), gerçek hayatta karşılaşılabilecek birçok problemi çözemeyeceğini ortaya koymuştur. Bunun en klasik örneği YADA (XOR -*Exclusive OR*) problemidir [11].

Daha önce de belirtildiği gibi, doğrusal problemlerde (VE, VEYA) başarılı olan perceptron, *YADA* (XOR) problemi gibi doğrusal olmayan problemlerde başarılı değildir.

X_1	X_2	Y
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1



Şekil 2.10: Mantıksal YADA (XOR) problemi

YADA probleminde, iki boyutlu örnek uzaydaki sınıflar, bir doğru ile birbirlerinden ayrılamazlar. Dolayısıyla ilk eğitim vektörüyle ulaşılan w_i değerleri, ikinci eğitim girdisi için hatalı olacaktır. Perceptron doğru sınıflandırma yapamayacaktır.

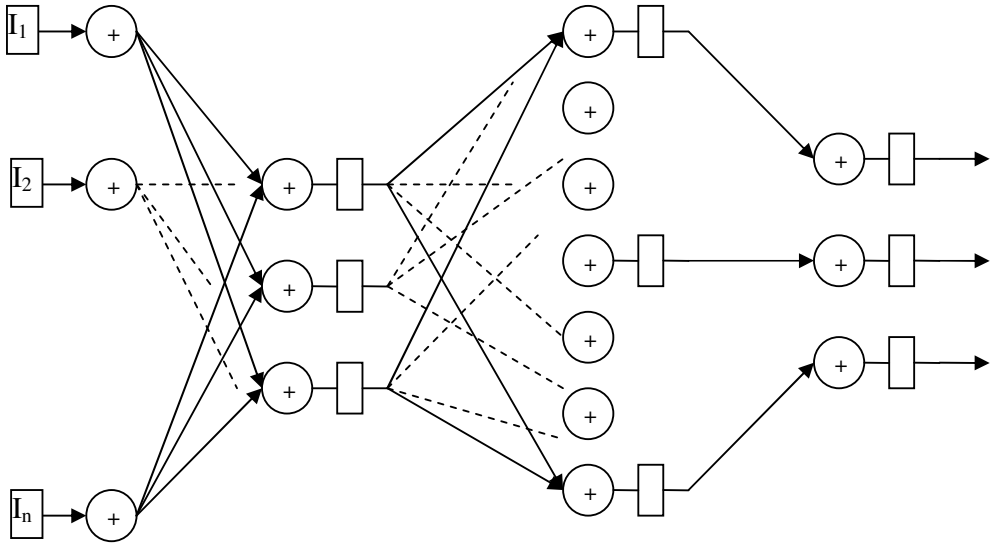
YADA problemindeki sınıfları gösteren şekil (şekil 2.10) incelendiğinde, -1 ve +1 sınıflarını ayıran bir doğru bulmak mümkün değildir. Oval alanın içinde kalan örnekler +1 sınıfına, dışındakiler ise -1 sınıfına aittir. [12]

Perceptronun doğrusal olmayan problemleri çözemeyişinin ispatından sonra, yapay sinir ağı konusundaki çalışmalarda duraksama olmuştur. Çünkü gerçek hayattaki problemlerin çoğu doğrusal değildir.

80'li yıllarda YADA probleminin çözümü için çalışmalar yapılmış ve Rumelhart ve arkadaşları tarafından çok katmanlı algılayıcı modeli geliştirilmiştir (1986).

Çok Katmanlı Ağ (ÇKA- Multilayer Perceptron - MLP) giriş ve çıkış düğümleri arasında birden çok katmana sahiptir. Giriş katmanı, çıkış katmanı ara katmanlardan oluşur.

Giriş katmanı, ÇKA'ya gelen girişleri alır ve ara katmana iletir. Gelen veri üzerinde hiçbir işlem yapmaz, olduğu gibi ara katmana iletir. Bu katmandaki işlem elemanları, sonraki ara katmandaki tüm işlem elemanlarına bağlanır.



Şekil 2.11: Çok Katmanlı Ağ modeli

Ara (saklı) katman, giriş katmanından gelen verileri işler ve çıkış katmanına iletir. ÇKA'da birden çok ara katman ve her katmanda birden çok işlem elemanı olabilir.

Çıkış katmanı, ara katmandan gelen bilgileri işler ve ağın çıktısını belirler. Çıktı katmanında birden çok işlem elemanı bulunabilir. Çıkış katmanındaki işlem elemanları, bir önceki katmandaki tüm işlem elemanlarına bağlıdır. Her işlem elemanının bir çıkışı vardır.

2.2.1 Çok katmanlı ağın öznitelik uzayını bölmelemesi

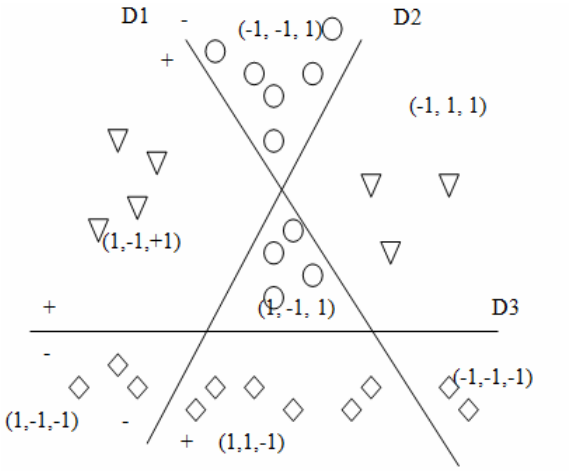
Çok Katmanlı Ağın birinci katmanı, uzayı bölmeleyen hiper düzlemleri bulundurur. Bu katmandaki düğümler ağırlıkları ile uzayı farklı bölgelere ayırır. Uzay, ne kadar çok bölgeye ayrılırsa, sınıf dağılımı o kadar iyi temsil edilir.

Birinci katmanın çıkış düğümleri, ikinci katman için bir bölge adresi olarak giriş olmaktadır. Şekil 2.12'de bu bölgeler gösterilmiştir. Görüldüğü gibi başarılı bir bölmelemede her bölgede yalnızca bir sınıfa ait eleman bulunmalıdır.

İkinci katman, birinci katman çıkışındaki bölge adresini alıp, bunun hangi bölgeye karşı düştüğünü belirtir. Bu katmanda, düğümler eğitim sırasında belirlenmiş adreslere sahiptir. Birinci katman çıkışının kendisine hamming mesafesi olarak en yakın olduğu düğüm bu katmanda kazanan düğüm olur ve ikinci katman çıkışında bu düğüm "1" değerini alırken, diğer düğümler "0" değerini alır. Şekil 2.12'de görüldüğü gibi bir sınıfa ait örnekler birden fazla bölgede bulunabilirler. Bu yüzden, ikinci katman sınıf kararını üretemez, ancak incelenen örneğin hangi bölgede

olduğunu belirler. İkinci katmandaki düğüm adedi en fazla birinci katmandaki düğümlerin sayısına bağlı olarak oluşan ayrık bölge sayısı kadardır. Sınıf kararını üretme işi üçüncü katmana bırakılır.

Üçüncü katman, ikinci katmandan aldığı bölge bilgisini ve eğitim sırasında oluşturulmuş olan ikinci katmana olan bağlantıları kullanarak sınıf kararını üretir. Üçüncü katmandaki düğüm sayısı, sınıf sayısı kadardır. Bu katman, basit bir "veya" işlemi yapar. Bir sınıf birden çok bölgede bulunabileceğinden, sınıfın bulunabileceği bölgelerin çıkışları "veya" lanarak sınıf kararı üretilmiş olur [13].



Şekil 2.12: ÇKA'nın öznitelik uzayını bölmelemesi

2.2.2 Çok katmanlı ağın eğitimi

Çok Katmanlı Ağda, öğrenme yöntemi olarak, önceki bölümlerde anlatılan ADALINE modelindeki öğrenme kurallarının biraz daha geliştirilmiş şekli olan genelleştirilmiş delta (δ) öğrenme kuralı kullanılmaktadır. Çok Katmanlı Ağda öğrenmenin gerçekleşmesi için eğitim kümesi hazırlanır. Bu eğitim kümesinde her örnek için, hem girdiler, hem de bu girdilere karşılık ağın üretmesi gereken çıktılar belirlenmiştir. Bu öğrenme kuralında temel amaç, ağın ürettiği çıktı ile eğitim kümesinde verilen çıktı arasındaki hatayı en aza indirmektir. Bunu yaparken hatayı ağa yaydığı için, çok katmanlı ağın diğer adı hata yayma ağıdır.

Genelleştirilmiş Delta Kuralının iki aşaması vardır.

- İleri doğru hesaplama (feed forward)
- Geri doğru hesaplama (back propogation)

2.2.2.1 İleri doğru hesaplama (feed forward)

İleri doğru hesaplama aşamasında, ağa uygulanan girdilere karşılık, ağın ürettiği çıkış bulunur. Giriş katmanında, *n*, katmandaki işlem elemanı sayısını, $\overline{X^0}$, giriş vektörünü, $\overline{Y^0}$, giriş katmanındaki çıkış vektörünü göstermek, giriş – çıkış arasındaki bağıntı aşağıdaki gibidir.

$$\overline{X^0} = (X_0^0, X_1^0, ..., X_n^0)\ \overline{Y^0} = (Y_0^0, Y_1^0, ..., Y_n^0) \quad \textbf{(2.10a)}$$

$$X_0^0 = 1, \quad \overline{X^0} = \overline{Y^0} \quad \textbf{(2.10b)}$$

Ara katmanın ilk katmanındaki giriş ve çıkış arasındaki ilişki aşağıdaki denklemde verilmiştir (2.11). Denklemde w_{ji}^1, giriş katmanındaki i-inci işlem elemanı ile ilk ara katmandaki j-inci işlem elemanı arasındaki bağıntıyı, *m* ise ilk ara katmandaki işlem elemanı sayısını göstermektedir.

$$\overline{X^1} = (X_0^1, X_1^1, ..., X_m^1) \quad \textbf{(2.11a)}$$

$$X_j^1 = \sum_{i=0}^{n} w_{ji}^1 . Y_i^0 \qquad \text{j=1,2,……..,m} \quad \textbf{(2.11b)}$$

Ara katmanın çıkışı ise şu şekilde verilmiştir.

$$\overline{O}^1 = \left[F\left(I_1^1\right), F\left(I_2^1\right), ..., F\left(I_m^1\right)\right] \qquad O_0^1 = 1 \quad \textbf{(2.12a)}$$

$$F(x) = \frac{1}{1+e^{-x}} \quad \textbf{(2.12b)}$$

Son olarak, çıkış katmanındaki giriş-çıkış bağıntısı aşağıdaki şekildedir. *k*, katmandaki işlem elemanı sayısını gösterir.

$$\overline{X^Ç} = (X_0^Ç, X_1^Ç, ..., X_n^Ç) \quad \textbf{(2.13a)}$$

$$X_j^Ç = \sum_{i=0}^{k} w_{ji}^Ç . Y_i^{Ç-1} \qquad \text{j=1,2,…….k} \quad \textbf{(2.13b)}$$

$$\overline{O}^{c} = \left[F\left(I_1^{c}\right), F\left(I_2^{c}\right), \ldots\ldots\ldots, F\left(I_k^{c}\right)\right] \qquad \textbf{(2.13c)}$$

Ağın çıkışı hesaplandığında ileri doğru hesaplama aşaması tamamlanmış olur.

2.2.2.2 Geri doğru hesaplama (back propogation)

Eğitim kümesiyle ağa verilen girdiler için ağın ürettiği çıktı ile beklenen çıktı karşılaştırıldığında aradaki fark hata olarak kabul edilir. Bu hata ağdaki ağırlık değerlerine dağıtılarak bir sonraki iterasyonda hatanın azalması sağlanır. Eğitim kümesinde verilen çıktı katmanında j. işlem elemanı için hata şu şekilde hesaplanır.

$$E_j = (t_j - O_j^{c}) \qquad \textbf{(2.14)}$$

Tek işlem elemanı için hesaplanan hata, çıkış katmanındaki tüm işlem elemanları için hesaplandığında şu ifade elde edilir. *k* çıkış katmanındaki işlem elemanı sayısı, $\overline{T}$ beklenen çıkış değerleri vektörü olmak üzere;

$$E = \frac{1}{2} \cdot \sum_{j=1}^{k} (t_j - O_j^{c})^2 \qquad \overline{T} = \left[t_1, t_2, \ldots, t_k\right] \qquad \textbf{(2.15)}$$

Bu hata miktarını azaltmak için, toplam hatanın ağın ağırlık elemanlarına (w_i) dağıtılması gerekir. Ağırlık elemanlarının değerleri değiştirilirken iki durum söz konusudur.

- Çıkış katmanı ile ara katman arasındaki ağırlıkların değiştirilmesi
- Ara katmanlar arası ya da ara katman ile giriş katmanı arasındaki ağırlıkların değiştirilmesi

Ara katmandaki i. işlem elemanı ile çıkış katmanındaki j. işlem elemanı arasındaki ağırlık değerinin değişim miktarı (Δw_{ij}^{c}) hesaplanırken hata ifadesinin (2.15), ağırlık değerine göre türevinden yararlanılır.

$$\Delta w_{ij}^{\varphi} = \frac{\partial E}{\partial w_{ji}^{\varphi}} = \frac{\partial E}{\partial O_j^{\varphi}} \cdot \frac{\partial O_j^{\varphi}}{\partial w_{ji}^{\varphi}}$$

$$= -(t_j - O_j^{\varphi}) \cdot \frac{\partial O_j^{\varphi}}{\partial w_{ji}^{\varphi}}$$

$$= -(t_j - O_j^{\varphi}) \cdot F'(I_j^{\varphi}) \cdot O_i^{\varphi-1} \qquad \textbf{(2.16)}$$

Ağda aktarım fonksiyonu *(F(x))* olarak sigmoid fonksiyonunun kullanıldığı varsayılırsa,

$$F(x) = \frac{1}{1+e^{-x}} \qquad F'(x) = F(x) \cdot (1 - F(x)) \qquad \textbf{(2.17)}$$

İfadesi elde edilir. Bu ifade denklem 2.16 da kullanılırsa, ifade yeni halini alır.

$$\Delta w_{ij}^{\varphi} = \frac{\partial E}{\partial w_{ji}^{\varphi}} = -(t_j - O_j^{\varphi}) \cdot O_j^{\varphi} \cdot (1 - O_j^{\varphi}) \cdot O_i^{\varphi-1} \qquad \textbf{(2.18)}$$

Bu yeni ifadede δ_j^{φ}, çıkış katmanındaki hata ifadesi, şu şekilde tanımlanabilir:

$$\delta_j^{\varphi} = (t_j - O_j^{\varphi}) \cdot O_j^{\varphi} \cdot (1 - O_j^{\varphi}) \qquad \textbf{(2.19)}$$

Bu durumda, çıkış katmanı ile ara katman arasındaki ağırlık değerlerinin değişim miktarı (Δw_{ij}^{φ}) aşağıdaki ifade ile hesaplanır.

$$\Delta w_{ij}^{\varphi} = \eta \delta_i^{\varphi} O_j^{\varphi-1} \qquad \textbf{(2.20)}$$

Yeni ağırlık değeri ise;

$$w_{ji}^{\varphi}(k+1) = w_{ji}^{\varphi}(k) + \eta \cdot \delta_j^{\varphi} \cdot O_i^{\varphi\text{-}1} \qquad \textbf{(2.21)}$$

Denklemiyle bulunur. Burada *"η"* öğrenme katsayısını belirtir.

Ara katmanlar arası ya da ara katman ile giriş katmanı arasındaki ağırlıkların değişim miktarı ($\Delta w_{ij}^{\varphi-1}$) hesaplanırken yine hatanın ağırlık değerine türevi kullanılır.

$$\Delta w_{ij}^{\varsigma-1} = \frac{\partial E}{\partial w_{ji}^{\varsigma-1}} = \sum_{m=1}^{k}\left((t_m - O_m^{\varsigma}) \cdot \frac{\partial O_m^{\varsigma}}{\partial w_{ji}^{\varsigma-1}} \right)$$

$$\Delta w_{ij}^{\varsigma-1} = \frac{\partial E}{\partial w_{ji}^{\varsigma-1}} = -\sum_{m=1}^{k} \delta_m^{\varsigma} \cdot w_{mj}^{\varsigma} \cdot O_j^{\varsigma-1} \cdot (1 - O_j^{\varsigma-1}) \cdot O_i^{\varsigma-2} \quad \textbf{(2.22)}$$

Ara katmanda hata ifadesi, aktarım fonksiyonunun sigmoid fonksiyonu seçilmesi durumunda şu şekilde bulunur.

$$\delta_j^{\varsigma-1} = O_j^{\varsigma-1} \cdot (1 - O_j^{\varsigma-1}) \cdot \sum_{m=1}^{k} \delta_m^{\varsigma} \cdot w_{mj}^{\varsigma} \quad \textbf{(2.23)}$$

$\delta_j^{\varsigma-1}$ ifadesi denklem 2.22'de yerine konulduğunda denklem 2.24 elde edilir..

$$\Delta w_{ij}^{\varsigma-1} = \frac{\partial E}{\partial w_{ji}^{\varsigma-1}} = \delta_j^{\varsigma-1} \cdot O_i^{\varsigma-2} \quad \textbf{(2.24)}$$

Ara katmanlar arasındaki ya da giriş katmanı ile ara katman arasındaki ağırlıkların yeni değeri aşağıdaki ifade ile bulunur.

$$w_{ji}^{\varsigma-1}(k+1) = w_{ji}^{\varsigma-1}(k) + \eta \cdot \delta_j^{\varsigma-1} \cdot O_i^{\varsigma-2} \quad \textbf{(2.25)}$$

Böylelikle tüm katmanlardaki ağırlık değerleri, hesaplanan hata değerini azaltacak şekilde güncellenmiş olacaktır. Çok katmanlı ağın eğitiminde, ileri ve geri hesaplamanın tamamlanmasıyla bir iterasyon tamamlanmış olur.

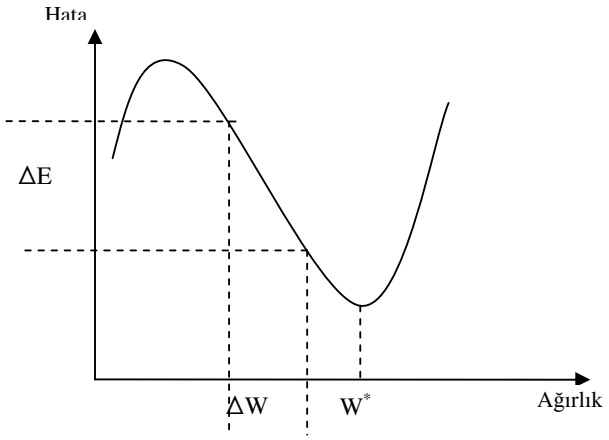
2.2.3 Çok katmanlı ağın başarımı

Bir yapay sinir ağının başarımı öğrenme yeteneğine bağlıdır. Ağın başarımı ölçülürken, eğitim kümesinde bulunmayan, yani ağın daha önce hiç görmediği test örnekleri ağa verildiğinde, bu örnekler karşısında ağın ürettiği doğru yanıtların oranı hesaplanır.

$$Basarim = \frac{Test\ setindeki\ orneklere\ verilen\ dogru\ yanit\ sayisi}{Test\ setindeki\ ornek\ sayisi} x100 \quad \textbf{(2.26)}$$

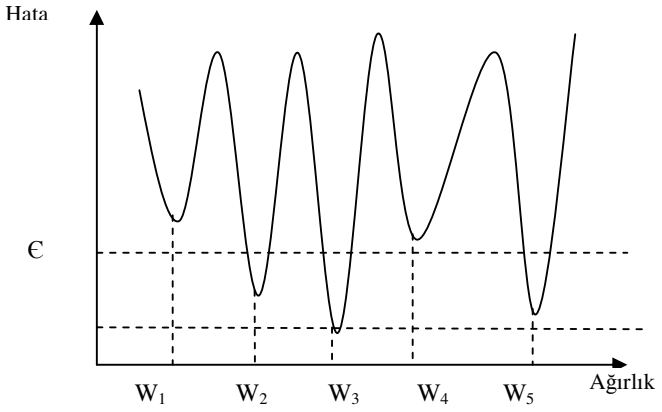
Ağın başarımını etkileyen bazı özellikleri bilmek ve ağla ilgili seçimleri bu özelliklere göre yapmak gerekmektedir. Bunlar ağırlık değerleriyle, ağın parametreleriyle, seçilen örnek kümeleriyle, sonlandırma koşullarıyla ilgili olabilir.

ÇKA'nın eğitiminde, hata *(E)* ile ağırlık değerleri *(W)* arasındaki ilişki aşağıdaki şekilde (2.13) verilmiş olsun.



Şekil 2.13: Çok Katmanlı Ağda ağırlık vektörü-hata ilişkisi

Şekle göre, en az hatayı verecek olan ağırlık vektörü W^* dır. İterasyonlarda ağırlık değerleri (W), ΔW kadar değiştirililerek, hatanın ΔE kadar azalması sağlanır. Bu yolla ağın sahip olabileceği en iyi çözüme, W^* değerine ulaşılmaya çalışılır. Ancak bu her zaman mümkün olmayabilir.



Şekil 2.14: Çok Katmanlı Ağda ağırlık vektörü-hata ilişkisi - 2

Burada W_1 ve W_4 vektörleri yerel çözümlerdir ve tolerans değerinin (Є) üstünde kaldıkları için kabul edilmezler. W_2 ve W_5 vektörleri yine yerel çözümlerdir ancak tolerans değerinin altında oldukları için kabul edilebilir çözümlerdir. En iyi çözüm

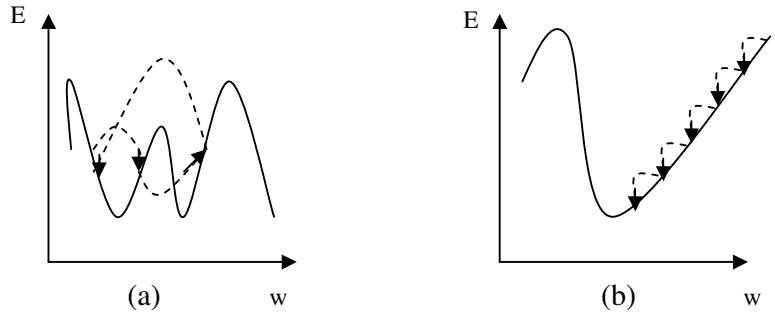
ise W_3 çözümüdür. Ancak aşağıdaki gibi bazı sebeplerle ağ en iyi çözüme ulaşamayabilir.

- Ağ eğitilirken seçilen örnekler problem uzayını %100 temsil etmeyebilir.
- Kullanılan ağın yapısı uygun seçilmemiş olabilir.
- Ağın ağırlıklarına başlangıçta verilen değerler doğru belirlenmemiş olabilir.
- Ağ için parametreler doğru seçilmemiş olabilir.

Bu durumda ağın problemi öğrenebilmesi için bazı değişiklikler yapılarak, ağın yeniden eğitilmesi gerekir. Yapılabilecek değişiklikler aşağıdaki gibidir.

- Başlangıç değerleri değiştirilebilir
- Ara katman sayısı ya da katmanlardaki işlem elemanlarının sayıları değiştirilerek ağın yapısı değiştirilebilir.
- Aktarım fonksiyonu değiştirilebilir
- Öğrenme katsayısı, *"η"*, değiştirilebilir.
- Eğitim kümesi üzerinde değişiklikler yapılabilir.

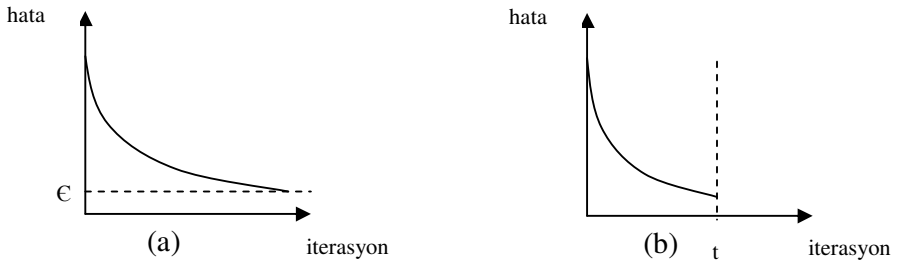
Burada bahsedilen öğrenme katsayısı, *"η"* , ağırlıkların değişim miktarını belirler. Öğrenme katsayısı küçük seçilirse, çözüme ulaşma süresi artar. Eğer büyük seçilirse, problem uzayında rasgele gezinmeler başlar ve ağırlıklar her iterasyonda rasgele seçilmiş olur. Şekil 2.15 öğrenme katsayısının öğrenme sürecine etkisini göstermektedir.



Şekil 2.15: (a) η büyük seçilirse (b) η küçük seçilirse

ÇKA eğitimi sırasında, belirli bir iterasyondan sonra ağın hatasının azalmadığı görülür. Bu, artık o ağda daha iyi bir sonuç alınamayacağı anlamına gelir. Eğer bu noktada bulunan çözüm tolerans değerinin üstünde kalmışsa, ağ yerel bir çözüme

takılmış anlamına gelir. Buradan hareketle ağın eğitimi iki şekilde durdurulabilir. İlk yöntem iterasyon sayısının başlangıçta belirlenmesi ve iterasyonlar bu sayıya ulaştığı zaman eğitimin sonlandırılmasıdır. Diğer yöntem ise, başlangıçta hata için bir tolerans değeri belirlenip, ağın hatası bu değerin altına düştüğü zaman eğitimin sonlandırılmasıdır.



Şekil 2.16: (a) Hata toleransı ile sonlandırma (b) İterasyon sayısı ile sonlandırma

ÇKA'da başarımı etkileyen en önemli özelliklerden biri ara katman sayısının ve katmanlardaki eleman sayısının belirlenmesidir. Bu sayıların en iyi şekilde belirlenmesini sağlayan bir yöntem yoktur. Deneme yanılma yolu ile belirlenmektedir. Birçok çalışmada başlangıç için bir ağ oluşturulup, zaman içinde ağ büyütülerek ya da küçültülerek istenilen sonuca ulaşılmaya çalışılır.

2.3 Öğrenme Yöntemlerine Göre Yapay Sinir Ağları

YSA'nı öğrenme yöntemlerine göre şu şekilde sınıflandırabiliriz.

Yapay Sinir Ağları
- Salt Eylemci (eğitimsiz)
  - Retina
  - Hopfield
- Öğrenen
  - Denetimli
    - Perceptron
    - ADALINE
    - Çok Katmanlı Ağ
    - RCE Ağı
    - İçerik Adreslemeli Bellek
    - Büyü ve Öğren (Grow And Learn - GAL) Ağı
  - Denetimsiz
    - Neocognitron
    - Kohonen Ağı
    - SOM
    - Adaptif Rezonans Teorisi

2.4 ÇKA Eğitiminde Kullanılan Sezgisel Yöntemler

Sınıflandırma (classification) ve tanıma (recognition) problemlerinin çözümünde sıklıkla kullanılan ÇKA'nın eğitiminde bazı problemlerle karşılaşılmıştır.

- ÇKA'nın katmanlarındaki düğüm sayısı, eğitime başlamadan önce belirlenmelidir. Seçilen düğüm sayısı sınıflama performansını doğrudan etkileyecektir. Katmanlardaki düğüm sayısını belirleyecek bir yöntem de mevcut değildir.
- Geriye yayılma algoritması, ÇKA'nın eğitiminde sıkça kullanılmaktadır. Ancak bu eğitim algoritması ağı, yerel çözümlere götürebilmektedir.
- Geriye yayılma algoritması, aynı amaçla kullanılan birçok öğrenme algoritmasından yavaş kalmaktadır.
- ÇKA eğitiminde düğüm sayısını ihtiyaca göre büyütecek bir mekanizma bulunmamaktadır.

Geri yayılım algoritmasının zayıf yönleri düşünülerek, ÇKA eğitiminde başka yöntemlerin kullanılması ile ilgili çeşitli araştırmalar yapılmıştır.

ÇKA eğitiminde, geri yayılma algoritmasının yerine kullanılan diğer bir yöntem Parçacık Sürü Optimizasyonu –PSO (Particle Swarm Optmization) algoritmasıdır. Bu algoritmanın ağın eğitiminde geri yayılım algoritmasından daha başarılı olduğunu gösteren çeşitli çalışmalar yapılmıştır [14,15]. V. G. Gudise, doğrusal olmayan fonksiyonların çözümü için eğitilen ileri beslemeli ağlarda, PSO algoritmasının Geri Yayılım Algoritmasından daha hızlı sonuca ulaştığını kanıtlamıştır. Bu algoritma Eberhart ve Kennedy tarafından, kuş sürülerinden, balık sürülerinden, arı sürülerinden esinlenerek geliştirilmiştir. Her parçacık, eniyileme probleminin muhtemel bir çözümünü temsil eder. Bu parçacıklar bulunduğu en iyi noktayı hafızasında tutarak, çözüm uzayında uyarlanabilir bir hızla hareket eder. Her iterasyonda parçacığın hızı ve bulunduğu nokta güncellenir. Eğer parçacığın hızı fazla artırılırsa çözüm uzayının dışına çıkabilir. Hız küçük artışlarda değiştirilirse, parçacık durma noktasına gelebilir yada yerel eniyi çözümlerden çıkamayabilir. Bu nedenle algoritma bu noktada geliştirilmiş, en küçük ve en büyük hız sınırları eklenmiştir. Ancak, karmaşık sinir ağlarında olduğu gibi, çözüm uzayının boyutları

arttıkça PSO algoritmasının performansı bozulmaktadır. Çok boyutlu problemlerde parçacık sürüleri yerel en küçük noktalarında takılabilmektedir [16].

Geri Yayılım algoritmasının bir varyasyonu olarak geliştirilen Hızlı Yayılım (Quick Propogation) algoritması (Fahlman, 1988; Patterson, 1996) da yapay sinir ağlarının eğitimi için kullanılmıştır. Bu algoritmada Geri Yayılma Algoritmasında hesaplanan hata, hata yüzeyinin eğiminden faydalanılarak daha hızlı küçültülmeye çalışılır [17].

Yapay sinir ağının eğitim sürecini hızlandırmak ve iyileştirmek amacıyla geliştirilmiş geri yayılım algoritmasına dayanan diğer bir algoritma ise Delta-Bar-Delta algoritmasıdır. Bu algoritmanın dayandığı temel fikir, geri yayılım algoritmasında kullanılan öğrenme katsayısının, her ağırlık değeri için farklı olabileceğidir. Bir ağırlık değeri için uygun olan öğrenme katsayısı, diğeri için uygun olmayabilir. Öğrenme katsayısının her ağırlık için ayrı belirlenmesi ve iterasyonlarda güncellenmesi hata fonksiyonunun en küçük değerine yaklaşmayı kolaylaştıracaktır [17].

Eğitim sırasında karşılaşılan problemleri gidermek ve daha az düğümle daha iyi başarım elde etmek için çok katmanlı ağın eğitiminde genetik algoritmalar da kullanılmıştır. Bu yolla genetik ÇKA (GetÇKA) oluşturulmuştur.

ÇKA'nın özellikle birinci katmanındaki düğüm sayısının ve bu düğümlerin ağırlıklarının, sınıflama performansını son derece etkilediği gözlenmektedir. Birinci saklı katmandaki düğüm sayısının önceden doğru belirlenememesi ağın sınıflama performansının aşırı derecede düşmesine yol açacaktır. ÇKA'nın eğitiminde kullanılan klasik yöntemlerde saklı katmandaki düğüm sayısı ihtiyaca göre otomatik olarak belirlenemez, kullanıcı tarafından belirlenmesi gerekmektedir.

Get-ÇKA yapısında ise, birinci katmandaki düğüm sayısının kullanıcı tarafından belirlenmesine gerek yoktur. Get-ÇKA'nın düğüm sayısı eğitim sırasında otomatik olarak belirlenmektedir. Get-ÇKA ile, ağa ait ve klasik yöntemlerde kullanıcı tarafından girilmesi gereken değerler genetik algoritmaların yardımı ile en iyi biçimde bulunurken, genetik algoritmaların sunduğu araştırma yöntemi sayesinde yerel çözümlerde kalma riski olmaz. Get-ÇKA'nın eğitilmesi sırasında hem ağın yapısı, hem de ağa ait ağırlıklar belirlenebilir [18].

ÇKA, genetik algoritma ile eğitilirken, ağın ağırlıkları bireyler (kromozomlar) ile temsil edilir. Başlangıçta bireyler rasgele belirlenir. Ağın mimarisi başlangıçta

belirlendiği için, bireyler ağın yapısal bilgilerini değil, yalnızca ağırlıklarıyla ilgili bilgiyi taşırlar. Böylece tüm bireyler başlangıç havuzunu oluştururlar.

Havuzdaki her eleman, yani ağırlık kümesi için, uyumluluk fonksiyonu hesaplanır. Ağ, eğitim kümesindeki elemanları öğrendiği zaman, uygulanan uyumluluk fonksiyonu en büyük değerine ulaşacaktır. Hesaplanan uyumluluk fonksiyonu ile elde edilen sonuç için ağın hatası hesaplanır ve bu hata kabul edilebilir orandaysa eğitim sonlandırılır. Ağın hatası kabul edilebilir değilse, eğitim işlemi yeni havuzun oluşturulmasıyla devam eder. Yeni havuz yeniden üretim ve seçim genetik operatörleri kullanılarak oluşturulur. Bu havuza, çaprazlama ve mutasyon işlemleri uygulanır ve yeni havuz üretilir. Oluşan havuz için uyumluluk fonksiyonu hesaplanır ve iterasyonlar bu şekilde devam eder [19].

3. DOĞA ESİNLİ ALGORİTMALAR

Yapay zeka alanında yapılan araştırmalar, 80'li yıllarda, biyolojik olayların ve doğal süreçlerin incelenmeye başlanmasıyla yeni bir boyut kazanmıştır. Zekanın ne olduğunu araştıran ve onu bilgisayar ortamında taklit etmeye çalışan bilim adamları, 'yaşam'ı doğal şekline en yakın hali ile bilgisayar ortamında meydana getirmeye uğraşmışlardır. Bu doğrultuda biyolojik olaylara dayanan yazılımlar üretilmiştir. Bu çalışmalar, doğal yaşamın ve biyolojik fenomenlerin, bilgisayar aracılığı ile taklit edildiği 'yapay yaşam'ı meydana getirmiştir.

Yapay zeka çalışmalarının başlamasından 1980'lere kadar, birimlerin zekası incelenip somut sonuçlar elde edilmeye çalışılmıştır. İlk ortaya çıkan yapay yaşam yazılımlarından biri, 'karınca çiftliği benzetimi'dir [20]. Yapay yaşam konusu üzerinde yapılan çalışmaların bir kısmı, biyolojik ilkelerin, doğal yaşam süreçlerinin matematiksel olarak modellenmesi ve algoritmalar oluşturulması yönünde yapılmıştır. Bu çalışmalar 'doğa esinli algoritmalar'ı, DEA, ortaya çıkarmıştır.

DEA'nın bir bölümü biyolojik ilkelere dayanmaktadır. Örneğin, doğada, çevreye uyum sağlayan canlı türlerinin hayatta kalırken, diğerlerinin yok olması olarak bilinen doğal seçilim süreci modellenerek evrimsel algoritmalar oluşturulmuştur. Evrimsel algoritmalar, genetik algoritmalar, evrimsel programlama ve evrimsel izlemler olarak üç şekildedir.

Doğal süreçler üzerinde çalışan araştırmacılar, zeki bilincin bir birimde değil, birden fazla birime yayılmış şekilde mevcut olabileceği fikrini ortaya atmış ve aynı amaç doğrultusunda çalışan birden fazla zeki birim yaratmanın araştırmaya faydalı olacağı kanısına varmışlardır. Bu fikir yapay yaşam ve yapay zekanın ortak bir dalı olan 'sürü zekası'nın (swarm intelligence) doğumu olmuştur. DEA'dan bazıları sürü zekası ilkelerine dayandırılarak ortaya çıkarılmıştır.

3.1 Evrimsel Algoritmalar

Doğal seçilim süreci sonucunda ortamda çevre koşullarına uyum sağlayabilen canlılar sonraki nesillere sağlıklı genler aktarabilirler. Canlıların sahip oldukları özellikler ortamın değişen şartlarına uyum sağlayacak biçimde baskın çıkarlar ya da yok olurlar. Bir doğal ortamda, bir canlı türünün çeşitli özelliklere sahip birçok üyesi bulunsun. Genetik havuz olarak adlandırılan doğal ortamda, nesiller sonra o ortama en iyi uyum sağlayacak genler baskın çıkarken, bu canlı türü bu ortama en iyi ayak uydurabilecek özelliklere sahip olmuş olur. Doğada bulunan bu süreci taklit eden evrimsel algoritmaların bir türü olan genetik algoritmalar, doğal seçilim süreci ve kalıtım ilkelerine dayanarak doğal bir eniyileme yöntemi ortaya koyar.

Birçok parametreye sahip olan bir sistemin bazı durumlarda en iyi parametrelerini bulmak çok zor olabilir. Böyle durumlarda, çözüm, olası parametre değerlerinin araştırılması şeklindedir. Rasgele bir araştırma yöntemi ise çoğu zaman oldukça zaman alıcıdır. Bu durumda genetik algoritma istenilen çözümü kısa bir sürede üretebilir. Geleneksel arama yöntemleri, probleme bir çözüm adayı önerir ve onu değiştirerek daha iyi çözümler elde etmeye çalışır. Genetik algoritmalar ise, bir çözüm adayları topluluğu oluşturur ve bu topluluk zamanla evrimleşir. Bir adayın çözüme ne kadar yakın olduğu, uygulamaya bağlı bir fonksiyondur. Bir çözüm adayı bir parametreler topluluğunu, bir kuralı, bir kurallar grubunu temsil edebilir. Hepsinde de, algoritma her adayın ne kadar güçlü olduğunu hesaplar ve buna göre bir sonraki neslin ebeveynleri olacak ya da yok olacak bireyleri belirler. Daha sonra, makul bir yeni nesil oluşturmak için ebeveynlere genetik arama işlemcilerini (yeniden yapılanma ve mutasyon) uygular.

Genetik algoritma, çözüm uzayını temsil eden genetik bir havuz ve ortam şartlarını temsil eden bir uyumluluk fonksiyonundan oluşur. Genetik havuzda, sistemin aranılan parametreleri için birçok aday çözüm bulunmaktadır. Çevre şartlarını temsil eden uyumluluk fonksiyonunu enbüyükleyecek genetik havuz elemanı, çevreye en iyi uyum sağlayan eleman olarak düşünülür ve sonraki nesillere aktarılma şansı daha yüksek olur. En uygun çözüm için en büyük değeri alması gereken uyumluluk fonksiyonu, probleme özgü olarak tasarımcı tarafından belirlenir.

Genetik algoritmaların güçlü bir özelliği, aday çözümleri içeren genetik havuz içindeki çözümlerden birkaç nesil boyunca yeni genetik çözümler elde edebilmesidir.

Bu arada, uyumluluk fonksiyonuyla belirlenen daha iyi çözümlere, daha fazla şans tanınarak en iyi çözüme yaklaşılacaktır. Genetik bir algoritmada yapılan temel genetik işlemler kopyalama, çaprazlama ve mutasyondur.

Kopyalama işlemi, havuz içinde bulunan aday çözümlerin uyumluluk fonksiyonuna göre oranlarının belirtilmesi işlemidir. Aday çözümler uyumluluk değerleriyle orantılı olarak seçilme olasılığına sahip olurlar. Bu sayede yüksek uyumluluk değerine sahip dizilerin sonraki nesillerde de havuzda bulunma ihtimali yükselirken, düşük uyumluluk değerine sahip çözümlerin seçilme ihtimali düşer.

Çaprazlama, rasgele seçilmiş genetik iki çözümün, rasgele seçilmiş bir bölgesinin içerdiği bilginin değiş tokuşudur. Çaprazlama işlemi, yeni genetik çözümlerin oluşmasını sağladığı için genetik algoritmalara büyük güç sağlar.

Mutasyon, herhangi bir dizinin rasgele bir pozisyonu üzerinde bir bilgi değişimi olarak ifade edilir. Kopyalama ve çaprazlama işlemlerinin yanında yardımcı rol oynar. Özellikle havuzda yararlı bilginin kaybolma ve en iyi çözümden uzak adayların düşük bir uyumluluk fonksiyonu civarında takılıp kalma ihtimaline karşı kullanılır.

Genetik algoritmalar uygulanırken, topluluk büyüklüğü, çaprazlama oranı, mutasyon oranı, kuşak aralığı başlangıçta belirlenmesi gereken parametrelerdir.

3.2 Sürü Zekası Algoritmaları

Daha önce açıklanan sürü zekası ilkelerine dayanan bu algoritmalar, gerçek değerli bir fonksiyonun genel en iyi değerine ulaşmayı amaçlar. Bir sürünün yada bir kolonin her bireyi ortak bir arama uzayı üzerinde bir fikre sahiptir. Bireyler fikirlerini şu 3 etmene göre değiştirirler.

- Ortam bilgisi (uyumluluk fonksiyonu)
- Bireyin önceki durumları (hafiza)
- Bireyin komşularının önceki durumları

Zamanla bireyler ortamdaki en iyi çözümü sağlayacak fikirlere uyum sağlarlar [21].

Bu algoritmalardan, Parçacık Sürü Optimizayonu, Yapay Arı Kolonisi ve Karınca Kolonisi Optmizasyonu algoritmaları hakkında bilgi verilecektir.

3.2.1 Paçacık sürü optimizasyonu

1995 yılında ortaya atılan PSO algoritması [22], çok boyutlu bir arama uzayında sürü parçacıklarının arama davranışına dayalı bir yöntemdir. Her parçacık problem için bir çözüm adayı olarak değerlendirilebilir. Her parçacık arama uzayında iki önemli etkene bağlı olarak uçuşur. İlk etken, o iterasyona kadar kendisinin en iyi pozisyonudur (yerel en iyi). İkinci etken ise, tüm parçacık sürüsü içinde en iyi (küresel en iyi) pozisyona sahip parçacığın pozisyonudur. i. parçacığın pozisyonundaki değişim hızı parçacık hızı (v_i) olarak adlandırılır. Parçacıkların hızı bir sonraki adımda kullanılacak parçacıkları belirlemek için her adım sonunda güncellenir. ayrıca, her adım sonunda parçacığı konumu da güncellenir. Algoritma problemin çözümü için aday parçacık sürüsünün rasgele oluşturulması ile başlar. Sürüdeki her bir parçacığın belirlediği çözüm için aday parametrelerle sistem çözümü yapılır. Elde edilen çözümden problemin yapına göre önceden belirlenen bir ölçüt ile (uyumluluk fonksiyonu) parçacığın uygunluğu sayısal olarak belirlenir. Bu işlem sürüdeki tüm parçacıklar için tekrarlanır ve içlerindeki en iyi uygunluk değerine sahip olan parçacık küresel en iyi olarak etiketlenir. İkinci adım için yeni parçacık değerleri belirlenir. İkinci ve sonraki adımlarda her parçacık için yerel en iyi parçacık, önceki adımlardaki en iyi uygunluk değerini alan aynı indisli parçacık olarak belirlenirken, küresel en iyi parçacık ise o iterasyona kadarki tüm parçacıklar içinden en iyi uygunluk değerini veren parçacık olarak etiketlenir ve bunlara göre güncellenir.

3.2.2 Yapay arı kolonisi algoritması

2005 yılında ortaya konulan yapay arı kolonisi algoritması [23] bal arılarının davranışlarını modellemiştir. Bal arıları, bir nektar kaynağı bulup kovana döndüğünde dans yaparak, diğer arılara nektarın yerini, uzaklığını ve zenginliğini dansla anlatır. Bu sayede kovan, kısa sürede maksimum kalitede bal ile dolar. Bu davranışı örnek alarak, 'Yapay Arı Kolonisi' yöntemi geliştirilmiştir. Yapay arı kolonisi algoritmasında, bir koloni de işçi arılar, gözcü arılar ve kaşif arılar olmak üzere üç grup arı bulunmaktadır. Geiştirilen modelde koloninin yarısı işçi, yarısı kaşif arı olarak seçilmiştir. Her bir nektar kaynağı için sadece bir işçi arı bulunur ve işçi arıların sayısı nektar kaynağı sayısına eşit tutulur. Tüm arılar çözüm uzayında uçuşurlar. Kaşif arılar nektar kaynağını rasgele seçerler ve nektar miktarını

hesaplarlar. İşçi arılar ise kendilerinin ve komşularının tecrübelerine göre nektar kaynağı seçerler ve konumlarını belirlerler. Ulaştıkları kaynaktaki nektar miktarı daha çok ise eski bilgileri unutup, yeni kaynağı hafızalarına alırlar.

Karınca Kolonisi Optimizayonu algoritması tezin dördüncü bölümünde ayrıntılı olarak ele alınmıştır.

4. KARINCA KOLONİSİ OPTİMİZASYONU

Arama ve eniyileme problemlerinde kullanılan sezgisel yöntemlerin belirlenmesinde, doğal süreçlerin kullanımı giderek yaygınlaşmaktadır. Biyoloji biliminden esinlenilerek, doğal süreçler gözlemlenmekte, bu süreçlerden matematiksel modeller elde edilmektedir. Bu modeller doğa esinli algoritmaları oluşturmaktadır. Doğa esinli algoritmalar, kombinatoriyel (kombinasyon hesapları içeren) eniyileme problemlerinin çözümünde kullanılmaktadır. Hayvanlar dünyasında, basit yeteneklere sahip bireylerin oluşturduğu karmaşık ve üstün yeteneklere sahip pek çok sosyal sistem örneği vardır. Çok sayıda basit bireyden oluşan bu sistemler, bir bütün olarak karmaşık bir yapı sergilemektedir. Karınca kolonileri de bu tarz sistemlere bir örnektir. Karınca Kolonileri meta sezgiseli, sürü tabanlı, rastsal arama ilkesine dayanan, en iyiye en yakın çözümü üretmek için tasarlanmış bir eniyileme yöntemidir.

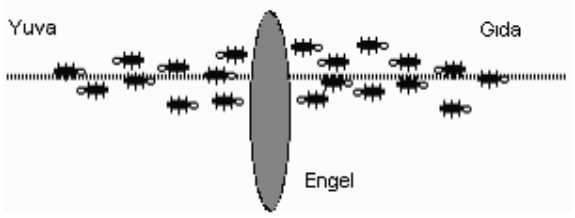
Bu tez çalışmasında, yapay sinir ağının eğitilmesinde, Karınca Kolonisi Optimizasyonu - KKO (Ant Colony Optimization) algoritması ilk kez denenmiştir. Bu bölümde Karınca Kolonisi Optimizasyon yöntemi ile ilgili bilgi verilecektir.

4.1 Karınca Kolonilerinin Doğal Özellikleri

Gerçek karıncalar, yuvalarından bir gıda kaynağına giden en kısa yolu, herhangi görsel ipucu kullanmadan bulma yetisine sahiptirler. Ayrıca, çevrelerindeki değişikliklere kısa zamanda uyum sağlayabilmektedirler. Örneğin, gıda kaynağına giden yolda herhangi bir problem meydana gelmesi, bir engelin ortaya çıkması ve yolun kullanılamaz olması durumunda, en kısa yolu yeniden bulabilirler [24]. Şekil 4.1, karıncaların yuvalarından gıda kaynağına doğrusal bir yol boyunca gidişlerini göstermektedir.

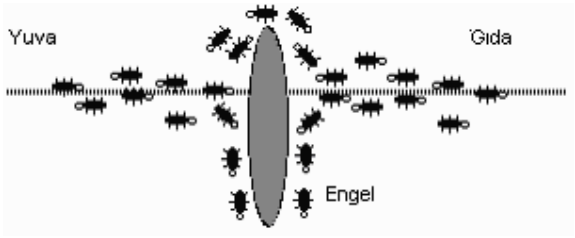
Şekil 4.1: Karıncaların doğrusal bir yolda gıda kaynağına gidişi

Bir karınca bir yiyecek kaynağı bulduğu zaman kaynağın kalitesini veya miktarını değerlendirir ve bir miktar yiyecek alarak yuvasına geri döner. Bu geri dönüş sırasında, bulduğu yiyecek kaynağının kalitesi veya miktarıyla doğru orantılı olacak şekilde kullandığı yola feromon maddesi koyar. Böylece diğer karıncalar bu yolun sonundaki yiyecek kaynağının kalitesi veya miktarı konusunda bilgi sahibi olurlar [25]. Feromon (pheromone), böcekler, örümcekler, sürüngenler gibi bazı hayvan gruplarında kendi cinslerinden diğer hayvanları etkilemek, bir uyarıyı iletmek yada iz sürmek amacıyla hücre dışına salgılanan bir tür hormondur. Karıncalar da gıda kaynağına giden en kısa yolu bulmak için feromon salgısını kullanırlar. Karıncalar ilerlerken, belirli bir miktar feromen depo ederler ve olasılığa dayanan bir yöntemle feromenin daha çok olduğu yolu az olduğu yola tercih ederler. Yuvaya yakın kaynaklara ulaşmak daha kolay olacağı için bu bölgelerde feromen maddesinin yoğunluğu daha fazla olacaktır. Depo ettikleri feromenleri, gıdaya giderken seçtikleri yola bırakarak, kendilerinden sonraki karıncalara yol seçiminde yardımcı olurlar. Bu içgüdüsel davranış, onların gıdaya giden en kısa yolu, önceden varolan bir yolun kullanılamaz olması durumunda dahi nasıl bulduklarını açıklar [26].



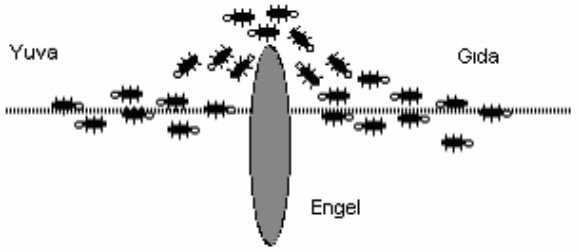
Şekil 4.2: Karıncaların gıda yolunda engel oluşması durumu

Yuva ile gıda kaynağı arasındaki yolda herhangi bir engel oluştuğunda (Şekil 4.2), engelin önüne gelen karınca feromon izini takip edemez ve yeni gıda yolunu bulmak için bir seçim yapmak zorunda kalır. Bu durumda karıncanın sağ veya sol yönü seçme olasılığı eşittir [24]. Karınca yaptığı seçime göre yoluna devam eder ve kendi yolunu çizer (Şekil 4.3).



Şekil 4.3: Karıncaların gıda yolunda engel oluşması durumunda rasgele yol seçişi

Buradaki ilginç noktalardan bir tanesi de, kolonide engel karşısında yol için eşit olasılıklı seçim yapan karıncaların seçtikleri yolun gıdaya giden en kısa yol olmaması durumunda, güzergâhlarını (koloni güzergâhı) çok hızlı bir şekilde yeniden yapılandırdıkları gerçeğidir. Yapılan seçimler de, bu yol üzerindeki feromen miktarını artıracak ve sonradan gelen karıncalar için tercih sebebi olacaktır [26]. Sonradan gelen karıncaların, yeni en kısa yolu seçmelerindeki feromen pozitif etkisinin (yol üzerindeki feromen) oluşabilmesi için, karınca ile yol üzerindeki engel arasındaki etkileşim çok hızlı bir şekilde gerçekleşmelidir. Her karıncanın, ortalama aynı hızda ve aynı miktarda feromen bıraktığı göz önüne alınırsa, karıncanın engeli fark edip en kısa yolu seçmesi, normal süreçten biraz daha uzun sürmelidir [26]. Fakat, sonradan gelen karıncaların feromene dayalı yol seçimi toplamda gıdaya giden süreci kısaltır (Şekil 4.4).



Şekil 4.4: Karınca kolonisinin en kısa yolu öğrenmesi

4.2 Yapay Karıncalar ve Karınca Kolonisi Metasezgiseli

Karınca kolonileri meta sezgiseli, doğal karıncaların yuvaları ile besin kaynakları arasında izledikleri yolların izlenmesi sonucu ortaya çıkan bilimsel gerçekler üzerine doğmuştur. Gerçek karıncalar ile ilgili deneyler Goss ve arkadaşları tarafından 1989 yılında laboratuar ortamında yetiştirilmiş karınca kolonileri üzerinde yapılmıştır. Bu çalışmalarda elde edilen sonuçlar aşağıdaki gibi özetlenmiştir [25];

- Pek çok karınca türü neredeyse kördür,
- Karıncalar yuvalarından yiyecek kaynağına veya tersi yönde hareket ederlerken geçtikleri yollara feromen adı verilen bir tür kimyasal madde bırakmaktadırlar,
- Karıncalar bir yol seçmeleri gerektiği zaman bu seçimi alternatif yollar üzerine bırakılmış olan feromen madde yoğunluğuna göre belirlemektedirler
- Karıncaların bu hareketleri merkezi bir kontrol ile sağlanmamaktadır.

Karıncaların çalışmalarla belirlenen bu özellikleri, bazı eklemelerle yapay karıncaların oluşturulmasında kullanılmıştır. Gerçek karıncalardan aşağıdaki üç özellik yapay karıncalara aynen aktarılmıştır.

- Feromen miktarı yoğun olan yolların tercih edilmesi,
- Kısa yollar üzerinde feromen miktarının daha hızlı artması.
- Feromon yoluyla karıncalar arasında kurulan iletişim,

Kombinatoryel eniyileme problemlerinin çözümünde daha faydalı olabilmeleri için yapay karıncalara bazı yeni özellikler eklenmiştir.

- Yapay karıncalar ayrık zamanda yaşarlar. Her zaman dilimide, problem tanımına göre hareket ederler.
- Yapay karıncalar tamamen kör değildirler. Problem yada amaç fonksiyonuyla ilgili bilgilere erişebilirler.
- Yapay karıncalar bir miktar hafızaya sahiptirler. Problemle ilgili ayrıntıları tutabilir ve bazı küçü hesaplamaları yapabilirler [27].

Karıncaların davranışlarının taban alındığı algoritmalar, ilk olarak Marco Dorigo tarafından ortaya atılmıştır [28]. Karınca tabanlı algoritmalarda temel fikir, basit iletişim mekanizmalarını kullanan yapay akıllı aracıların (agent), birçok karmaşık problem için çözümler üretebilmesidir.

4.3 Karınca Kolonisi Optmizasyonu Algoritması

Karınca Kolonisi Optimizasyonu-KKO algoritmalarında kullanılan yapay karıncalar ilk iterasyonda rasgele davranmaktadırlar. Bir noktadan diğerine geçerken, gidebilecekleri tüm noktaların seçilme olasılığı aynı olur. İlk turlarını bu şekilde tamamlamaktadırlar. İlk turda tüm noktalar için eşit seçilme olasılığını sağlamak için tüm yollara eşit feromon ataması yapılır. İlk turun sonunda, her bir karıncanın geçtiği yollardaki feromon miktarı güncellenir. Bu güncelleme yolun daha önce seçilmiş

olmasına, karıncaların geçiş sayısına ve yolun uzunluğuna göre yapılır. Sonraki iterasyonlarda, karıncalar gidecekleri yolu güncellenen feromon miktarlarına göre belirlerler. KKO'da yol tercihi belli bir olasılığa bağlı olarak iki şekilde gerçekleştirilir: İlk seçenek q_0 olasılıkla feromonun en yoğun olduğu yolun seçilmesidir. $\tau(i, j)$, i ve j noktaları arasındaki feromon miktarı, α ve β ayarlanabilir parametreler olmak üzere, i noktasında bulunan bir karıncanın gideceği nokta aşağıdaki gibi seçilmektedir:

$$j = \max_{u \in J_k(i)} \{ [\![\tau(\mathrm{i}, \mathrm{u})]\!]^{\alpha} x [\eta(i,u)]^{\beta} \} \quad \textbf{(4.1)}$$

Bu ifadeyede, $\eta(i, j)$ seçilebilirlik parametresini gösterir. i ve j noktaları arasındaki mesafenin tersine eşit olarak aşağıdaki gibi hesaplanır.

$$\eta(i, j) = \frac{1}{\delta(i, j)} \quad \textbf{(4.2)}$$

İkinci seçenek ise gidilmesi mümkün olan yollardan birini, yollardaki feromon izleriyle orantılı olarak seçmektir. Bu şekilde yol seçimi olasılığı 1- q_0 oranındadır. $J_k(i)$, i noktasındaki karıncanın gidebileceği noktalar yani ziyaret edilmemiş şehirleri temsil eder. Tüm şehirler için seçilme olasılıkları aşağıdaki gibi hesaplanmaktadır:

$$P_k(i, j) = \frac{\{[\tau(i,u)]^{\alpha} x [\eta(i,u)]^{\beta}\}}{\sum_{u \in J_k(i)} \{[\tau(i,u)]^{\alpha} x [\eta(i,u)]^{\beta}\}}, j \in J_k(i) \quad \textbf{(4.3)}$$

Bu olasılıklara bağlı olarak yol seçilmektedir. Feromonun yoğun olduğu yolların seçilme olasılığı yüksektir.

Tüm karıncalar turlarını bitirdiğinde, iterasyon tamamlanmış olur. İterasyon sonunda, yollara ait feromon miktarının güncellenmesi gerekir. Böylece ulaşılan çözüm uzayı da taranmış olur. Feromon güncellemesi, buharlaşma ve artırma işlmleri ile yapılır. Artırma, karıncaların kullandıkları yollarda, o karıncanın yol uzunluğuyla ters orantılı olarak yapılır. Bu şekilde iyi çözümlerin, kısa yolların, öneminin artması sağlanır. Buharlaşma ise, tüm yollardaki feromonların belirli bir oranda azaltılmasıyla sağlanır. Buharlaşma etkisi ile, önceki çözümlerin önemi azaltılır.

Tur sonunda feromon güncellemesi sırasında yeni feromon düzeyi şu şekilde hesaplanır:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \quad \textbf{(4.4)}$$

Bu ifadede τ_{ij} (t), t iterasyonuna kadar biriken feromon düzeyini, $\Delta\tau_{ij}^{k}(t+1)$, t iterasyonundaki feromon düzeyini temsil eder. ρ parametresi feromon buharlaşma katsayısını gösterir ve bu değer $(0 \le \rho \le 1)$ aralığında seçilir.

t. iterasyondaki fromon düzeyi ($\Delta\tau_{ij}^{k}(t+1)$) aşağıdaki gibi hesaplanır.

$$\Delta\tau_{ij}^{k}(t+1) = \frac{1}{L^{k}(t+1)} \qquad \textbf{(4.5)}$$

Burada, $L^{k}(t+1)$ k karıncasının toplam yol uzunluğudur. Feromon güncellemesi işlemi ile her iterasyonda daha kısa turları bulmak amaçlanmaktadır. Geçiş yapılan yolların seçilme olasılığının artırılmasıyla, turlar dinamik olarak değiştirilmektedir [29][30][31].

4.4 Geliştirilmiş Karınca Sistemleri

Bugüne kadar birçok KKO algoritması geliştirilmiştir. Bunlardan ilki M.Dorigo ve arkadaşları tarafından geliştirilmiş olan Karınca Sistemi-KS (Ant System)'dir [28] . KS öncelikle Gezgin Satıcı Problemine - GSP uygulanmıştır. Küçük ölçekli GSP problemlerinde başarılı olurken daha büyük ölçeklilerde başarılı olamamıştır. KS algoritması aslında Karınca Çevrimi (Ant-Cycle), Karınca Yoğunluğu (Ant-Density) ve Karınca Miktarı (Ant-Quantity) olmak üzere üç farklı algoritma kümesinden oluşmaktadır. Bunların arasındaki fark ise feromonun depolanmasında yatmaktadır. Karınca Çevrimi algoritmasında, feromon, sadece her karınca turunun sonunda güncellenmektedir. O karıncanın turu üzerindeki her kenar, uzunluklarıyla ters orantılı olarak feromonla depolanmaktadır. Bununla birlikte her sanal karınca gerçeğinin aksine geçtiği yolları hatırlamak için bir hafızaya sahiptir. Diğer iki algoritmada ise feromon yolu, karınca bir noktadan diğer bir noktaya geçerken güncellenmektedir. Yapılan testler sonunda Karınca Çevrimi algoritmasının daha iyi sonuçlar verdiği ortaya çıkmıştır. Karınca Çevrimi algoritması genel bir yapı kazanmış ve KS algoritması denildiğinde bu algoritma kastedilir hale gelmiştir [2].

KS algoritması üzerinde, 1996'da, Dorigo tarafından bir geliştirme yapılmış ve seçkinci izlem (elitist strategy) ortaya atılmıştır [31]. Bu yöntemde, arama sırasında

bulunan en iyi yola daha çok feromon konulması sağlanır. En iyi yol üzerindeki feromon miktarı (τ_{eiy})aşağıda hesaplanan miktar kadar artırılır [31].

$$\Delta\tau_{eiy} = e.\frac{Q}{L^*} \qquad \text{e: seçkin karınca sayısı Q: sabit } L^*\text{: en iyi yol uzunluğu} \tag{4.6}$$

Yine Dorigo tarafından 1996 yılında Karınca Kolonisi Sistemi – KKS (Ant Colony System) geliştirilmiştir [32]. KKS algoritması üç noktada farklılık gösterir. İlki, karıncalar bir noktadan diğerine giderken sözde-rastsal bir orantı kullanılmasıdır. İkincisi, feromon yalnızca en iyi kaıncalar tarafından bırakılır. Üçüncüsü ise karıncaların çözüm üretirken yerel feromon güncellemesi kuralını kullanmalarıdır. [32].

1997 yılında Stützle ve Hoos tarafından Min-Max Karınca sistemi (MM-KS) geliştirilmiştir [33]. MM-KS'de, bulunan iyi sonuçlardan daha fazla faydalanabilmek için, her tur sonunda sadece bir ve en iyi sonucu bulan karıncanın feromon eklemesine izin verilir. Ayrıca karıncaların sürekli aynı sonucu bulmasını önlemek için MM-KS algoritmasında feromon güncelleştirmesine dair bir üst sınır ve alt sınır belirlenir. Son olarak, bu algoritmada problemin başlangıcında tüm feromon miktarları belirlenen üst sınıra eşitlenir [25].

1999'da Bullnheimer, Hartl ve Stauss, Mertebe Temelli Karınca sistemi- AS_{RANK}, adı altında seçkinci izlemin gelişmiş bir çeşidini sunmuşlardır [34]. Bu algoritmaya göre her tur sonunda karıncalar tur uzunluklarına göre sıralanmakta, sadece sıralamadaki belli sayıda en iyi karıncaların ve o ana kadarki en iyi karıncanın belli bir miktara göre feromon bırakmasına izin verilmektedir.

4.5 KKO Algoritmasının Örnek Problemlere Uygulanması

KKO ile ilgili olarak, birçok çalışmada farklı alanlardaki uygulamalara yönelik, yöntemin performansını arttırmak amaçlı değişiklikler ve eklemeler yapılmıştır. Örnek olarak; karesel atama problemi, (Maniezzo *et al.*, 1994; St¨utzle & Hoos, 1998a), araç yönlendirme (Bullnheimer *et al.*, 1997a; Gambardella *et al.*, 1999), bağlantılı ve bağlantısız ağ yönlendirme (Schoonderwoerd *et al.*, 1996; Di Caro & Dorigo, 1998; Di Caro & Dorigo, 1998), ardışık sıralama (Gambardella & Dorigo, 1997), grafik renklendirme (Costa & Hertz, 1997) konularında karınca kolonisi

optimizasyon yöntemi ile çalışmalar yapılmıştır. Karınca sistemi temelli yaklaşımlar, bu problemlerin çoğunda en iyi sonucu veren sezgiseller olmuştur [29].

Bu bölümde KKO algorimasının Gezgin Satıcı Problemi ve Karesel atama Problemlerine uygulanışı anlatılacaktır.

4.5.1 KKO algoritmasında parametrelerin seçilmesi

KKO algoritmasının çeşitli problemlere uygulanmasında öncelikle paremetrelerin uygun değerleri belirlenmelidir. Algoritmanın ayarlanabilir parametreleri; karınca sayısı (m), olasılık parametresi (q_0), önem parametreleri (α ve β), feromon buharlaşma parametresi (ρ)dir. Bu parametrelerin etkin olarak seçilmesi, algoritmanın performansını ve başarısını önemli ölçüde iyileştirmektedir [6].

Karınca sayısının az olması feromon buharlaşması nedeniyle kolektif çalışma özelliğini dolayısıyla algoritmanın başarısını azaltmaktadır. Karınca sayısının artırılması ise, çözümde iyileşme sağlamakla beraber, hesaplamaları artırdığı için işlem süresini uzatmakta, performansı düşürmektedir. GSP üzerinde yapılan denemeler sonucunda, karınca sayısının şehir sayısına eşit olarak seçilmesinin en etkin sonucu verdiği görülmüştür [35] .

Olasılık parametresi, q_0, en iyi çözümün sonraki sonraki iterasyonlara aktarılmasını sağlayan parametredir. GSP'de bu parametre genellikle 0,9 olarak seçilmektedir. Böylece, karınca %90 olasılıkla en iyi çözümü sağlayan yolu tercih eder, %10 olasılıkla feromon izlerine bağlı olarak yolunu belirler.

α ve β parametreleri ise birbirleri ile ilişkili olarak belirlenir. α parametresi, ilgili yolun seçilmesinde o yoldaki feromon miktarının önemini, β parametresi ise yine seçimde o yolun uzunluğunun önemini vurgular. α parametresinin yüksek olması, feromonun yüksek olduğu yolların seçilme olasılığını artırırken, rastlantısal seçimi azaltmaktadır. β değerinin artması rastlantısal seçim yani alternatif yolların araştırılması olasılığını artırır [35]. GSP, çeşitli α ve β parametreleri ile çözülmüş ve bu parametreler için en uygun değerler α=0,625 ve β =4,375 olarak belirlenmiştir [25].

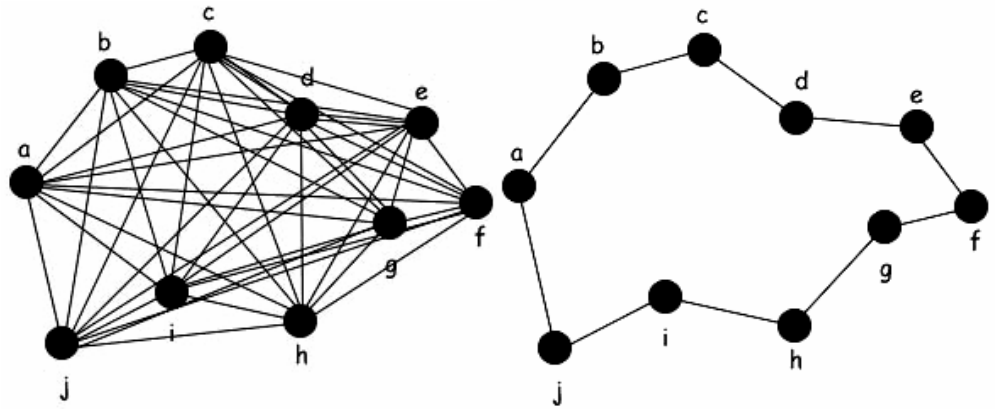
ρ parametresi ise kullanılmayan yolların sonraki iterasyonlarda seçilme olasılığını düşürmek için kullanılan feromon buharlaşma parametresidir. Yapılan denemelerle bu parametrenin en etkin değeri 0,125 olarak hesaplanmıştır [25].

KKO'nun ayarlanabilir parametreleri belirlendikten sonra, gezgin satıcı probleminin çözümü için algoritma yürütülür.

4.5.2 KKO algoritmasının gezgin satıcı problemi üzerine uygulanması

Gezgin Satıcı Problemi - GSP, seyahat eden bir satıcının, herhangi bir şehirden başlayarak, gezmesi gereken tüm şehirleri en kısa yolu kullanarak (en az maliyetle) gezip, tekrar başladığı şehre dönmesini vurgulayan bir optimizasyon problemidir. Bu ve benzeri optimizasyon problemlerinin çözümü için bir çok algoritma geliştirilmiştir. KKO Algoritması da GSP'nin çözüm yollarından birisidir.

GSP NP-Zor (makul zamanda en iyi çözüme erişilemez) bir problemdir ve şehir sayısı arttıkça karmaşıklık derecesi artmaktadır.



Şekil 4.5: Örnek GSP problemi ve çözümü

Şekil 4.5'de gösterildiği gibi GSP problemi için bir çok tur olasılığı vardır.

GSP için, KKO algoritması yürütülmeden önce tur sayısı (NC) sezgisel olarak belirlenir. Yine önceden belirlenmiş m tane karınca n tane şehir üzerine rasgele yerleştirilir. Her yol için feromon miktarı (τ_{ij}) aynı olacak şekilde feromon değerleri belirlenir. Her karınca için ziyaret ettiği şehirlerin listesi (tabu) tutulur ve tabu listelerine her karıncanın başlangıç şehri eklenir. Her karıncanın turu bitene kadar, yani tabu listesi dolana kadar, (4.1) ve (4.3) numaralı bağıntılara göre karıncanın gideceği şehir (j) seçilir. Karınca j şehrine hareket ettirilir, j şehri tabu listesine eklenir. Her turun sonunda, her karıncanın yol uzunluğu (L^k) hesaplanır. (4.4) ve (4.5) numaralı bağıntılar kullanılarak, buharlaşma parametresiyle (ρ) birlikte feromon miktarları güncellenir. Bu işlemler NC tur sayısına ulaşılana kadar tekrarlanır. NC değerine ulaşıldığında, GSP için en kısa yol KKO algoritması ile bulunmuş olur.

4.5.3 KKO algoritmasının karesel atama problemine uygulanışı

Karesel Atama Problemi - KAP (Quadratic Assignment Problem), günümüzün en karmaşık bilimsel problemlerinden biridir. Amaç belirli sayıda tesisin, belirli sayıda aday bölgelere, toplam taşıma maliyetini en azlayacak şekilde yerleştirilmesidir. Problemi tanımlamak için, n tane tesis ve n tane yerleşim bölgesi olduğu varsayılsın. Bunlar için iki tane nXn boyutunda matris oluşturulacaktır. Bölgeler matrisinde (A) matrisinde a_{ij}, i ve i bölgeleri arasındaki uzaklığı, tesisler matrisinde (B) matrisinde b_{rs}, r ve s tesisleri arasındaki iş akışını gösteriyor olsun. Bu durumda tasarlanan KAP'nin matematiksel ifadesi şu şekilde olacaktır:

$$\min_{\psi \in S(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\psi_i \psi_j} \qquad \textbf{(4.7)}$$

Burada S(n), olası bütün permutasyonların yani atamaların kümesidir. ψ_i ise, mevcut çözüm içinde ($\psi \in S(n)$) i tesisinin atandığı bölgedir. $b_{ij} a_{\psi_i \psi_j}$ ifadesi ise, eş zamanlı olarak i tesisinin ψ_i bölgesine ve j tesisinin ψ_j bölgesine atanmasının maliyetini verir. KAP, tam sayı eniyilemesi problemi olarak formüle edilirse aşağıdaki halini alır:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} a_{ij} b_{kl} x_{ik} x_{jl} \qquad \textbf{(4,8)}$$

Bu denklemde, x_{ik}, i. tesis k. bölgeye atanmışsa 1, diğer durumlarda 0 değerini alan ikili bir değişkendir. Aşağıda verilen kısıtlamayı sağlamalıdır.

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ ve } \sum_{j=1}^{n} x_{ij} = 1 \qquad \textbf{(4,9)}$$

KAP'nin çözümünde çeşitli sezgisel yaklaşımlara başvurulmuştur. Genetik algoritmalar ve evrimsel yaklaşımlarla problem çözümlendiği gibi, karınca algoritmaları da probleme uygulanmıştır [36] [37].

KKO algoritması, KAP için yürütülürken, problem çizge ile gösterilir. G(C,L) çizgesinde, C bileşenleri, L ise bileşenler arasındaki bağlantıları gösterir. KAP'de bileşenler tesisler ve bölgelerdir. Bağlantılar ise tüm bileşenler arasında kurulabilir. KKO uygulamasında, atamalar, karıncaların çizge üzerinde, feromon izlerini takip ederek yürütülmesiyle yapılır. Karıncaların yaptığı atamalarda, her tesisin sadece bir bölgeye yerleştirilmesi ve her bölgeye sadece bir tesisin yerleştirilmesi kısıtlamalarına uyulmalıdır.

KKO uygulamasında önce atamanın yönü belirlenir. Tesislerin bölgelere, yada bölgelerin tesislere atanmasına karar verilip, tüm karıncalar bu önde çalıştırılır. Tesislerin bölgelere atanması yönünde karar verilsin. Uygulamanın başarımı için, tesisler arasındaki atama sırası da önemli olacaktır. Her bir karınca tam bir çözüm üretmek için iki aşama uygular. İlk aşamada atanacak tesisi seçer. İkinci aşamada ise, seçtiği tesisi bir bölgeye yerleştirir. Feromon izi her iki aşamada da kullanılabilir. İlk aşamada hangi atama sırasının daha iyi olduğunu feromon iziyle öğrenerek, tesis seçimini buna göre yapabilir. İkinci aşamada ise, feromon izi τ_{ij}, i tesisinin j bölgesine atanmasının ne kadar tercih edilir olduğunu gösterir. Her karınca uygun bir çözüm ürettikten sonra, çözümlerin kalitesine göre feromon izleri güncellenir [36].

Hibrid Karınca Sistemi ile KAP probleminin çözümünde (HAS-QAP) [37], her karıncanın üzerinde çalıştığı bir çözüm vardır ve karıncalar feromon izini kullanarak kendi çözümlerini geliştirirler. Karınca ilk i tesisini {1,2,…….n} kümesinden rasgele seçer. Bundan sonra başka bir j tesisini seçmeli ve i tesisinin bölgesi ψ_i'yi, j tesisinin bölgesini ψ_j ile değiştirmelidir. Burada j tesisi seçilirken feromon izine başvurulur. Aşağıdaki ifadeyi en büyükleyen j tesisi seçilir.

$\tau^k(j\psi_i) + \tau^k(i\psi_j)$ k: karınca indisi **(4.10)**

j tesisi aşağıdaki olasılık değerine göre belirlenir.

$$P_{ik}^k(t) = \frac{\tau^k(j\psi_i) + \tau^k(i\psi_j)}{\sum_{l \neq i} \tau(i\psi_l) + \tau(l\psi_i)} \quad \textbf{(4.11)}$$

Tüm karıncalar çözüm ürettiğinde, sadece en iyi çözümü bulan karınca feromonu aşağıdaki gibi günceller.

$\Delta\tau_{ij}(t) = \frac{Q}{J_\psi^{gb}}$, i tesisi j bölgesine yerleştirildiyse **(4.12)**

HAS-QAP uygulaması, diğer karınca uygulamalarından, feromon izini yeni bir çözüm oluşturmak için değil, mevcut çözümü değiştirmek için kullanmasıyla ayrılır [36].

4.5.4 KKO ile radyal tabanlı sinir ağı eğitimi

Radyal Tabanlı Sinir Ağında - RT-YSA, (Radial Basis Function NN) giriş katmanı, gizli katman ve çıkış katmanları bulunur. Giriş katmanı, giriş vektöründeki koordinatları gizli katmana aktarır. Gizli katmandaki her düğüm aldığı girişi, radyal tabanlı fonksiyon kullanarak işler ve çıkış katmanında bu sonuçların lineer kombinasyonları hesaplanır.

RT-YSA'da bir çıkış düğümünün matemaiksel ifadesi şöyledir.

$$c_j(x) = \sum_{i=1}^{k} \omega_{ji} \left\| x - \mu_i \right\|; \sigma_i \qquad \textbf{(4.13)}$$

Burada $c_j(x)$, j. çıkış düğümüne ait fonksiyona karşılık gelir ve merkezi μ_i, bant genişliği σ_i olan radyal tabanlı bir fonksiyonun lineer kombinasyonudur. ω_{ji} ise j. düğüm ile i. düğümün merkezi arasındaki ağırlıktır.

Örüntü tanıma problemlerinde radyal tabanlı fonksiyon olarak Gauss fonksiyonu kullanılır. Bu durumda $c_j(x)$'in yeni ifadesi şu şekilde olur.

$$c_j(x) = \sum_{i=1}^{k} \omega_{ji} \exp\left(\frac{\left\| x - \mu_i \right\|^2}{2{\sigma_i}^2} \right) \qquad \textbf{(4.14)}$$

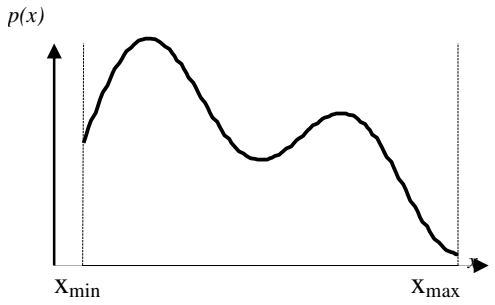
RT_YSA ağının eğitimde KKO kullanılırken, 4.4 bölümünde belirtilen denklemler kullanılarak, düğümlerin merkez ve band genişlik değerleri ve düğümler arası ağırlık değerlerinin en iyilenmesi sağlanır [38].

4.6 Sürekli eniyileme problemlerinde kko algoritması

Karınca Kolonisi Optmizasyonu, kombinatoriyel optimizasyon problemlerine uygulandığı gibi sürekli optimizayon problemlerinde (Continuous Optimization Problem - CnOP) de kullanılabilir [39]. Sürekli alan için oluşturulan bir Karınca Kolonisi modeli, karar ve kısıt, Ω, kümeleri üzerinde tanımlı bir çözüm uzayı, *S*, ve minimize edilecek bir CnOP fonksiyonundan, $f: S \to R_0^+$, oluşur. Çözüm uzayı *S*, X_i, i=1,….*n* sürekli değişkenlerin bir kümesidir. Bu kümeden seçilen bir çözüm, s∈ S, kısıt kümesindeki, Ω, tüm kısıtları sağlar ve verilen CnOP fonksiyonunun bir

çözümüdür. Çözüm uzayını oluşturan çözümlerden $f(s^*) \leq f(s)$ bağıntısını sağlayan çözümler, $s^* \in S$, genel eniyi olarak adlandırılır. CnOP çözümlenmesi, verilen fonksiyon için çözüm uzayında 1 yada daha fazla s^* bulunması ile gerçekleşir.

Sürekli KKO algoritmasında, ACO_R, sürekli olasılık yoğunluk fonksiyonu (Probability Density function, PDF) kullanılır (Şekil 4.6).

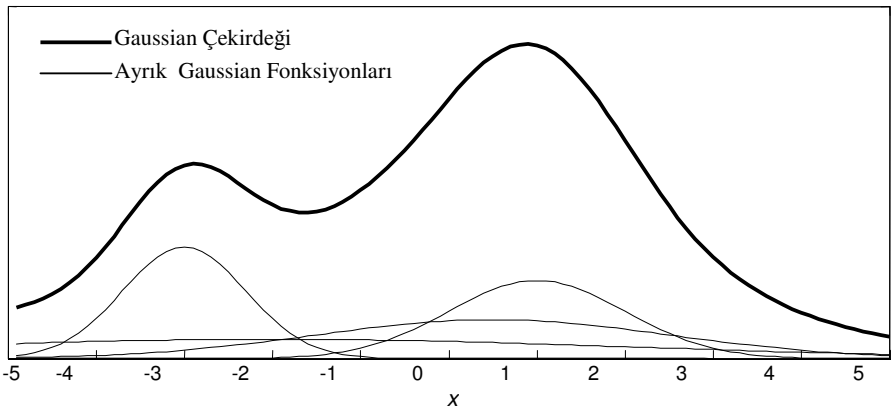


Şekil 4.6: ACO_R için sürekli olasılık –yoğunluk fonksiyonu

Socha ve Dorigo, Gaussian çekirdeği oluşturmak için çeşitli Gaussian PDF fonksiyonları kullanmıştır. Bulunan Gaussian çekirdeklerinin sayısı, problemin boyutu ile aynıdır I=1,2,…n.

$$G^i(x) = \sum_{l=1}^{k} w_l g_l^i(x) = \sum_{l}^{k} w_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i\,2}}} \qquad \textbf{(4.15)}$$

Burada w Gaussian fonksiyonu için ağırlık parametresini, μ^i ortalama vektörünü, σ^i standart sapma vektörünü gösterir.

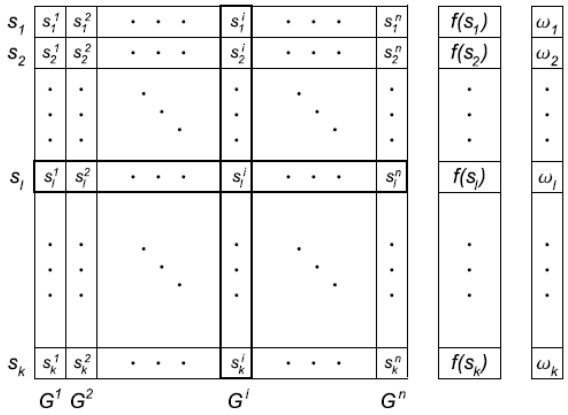


Şekil 4.7: Ayrık Gaussian fonksiyonları ve Gaussian çekirdekleri

Şekil 4.7, ayrık Gaussian fonksiyonlarını ve karşılık gelen Gaussian çekirdeklerini göstermektedir.

Bilinen KKO algoritmasındaki feromon bilgisi, sürekli algoritmada bir çizelge yapısında tutulur. Bu çizelgede, ulaşılan çözümler, s, ve bunlaın amaç fonksiyonları, f(s), bir arada tutulur. Çizelge 4.1'de bu çizelgenin yapısı görülmektedir.

Çizelge 4.1: ACO_R için feromon tablosu yapısı



Burada çözümler uygunluk değerlerine göre sıralanır ($f(s_1) \leq f(s_2) \leq f(s_3) \leq \ldots f(s_k)$).

Gaussian çekirdek PDF'nu meydana getiren k tane Gaussian fonksiyonu vardır. n problemin boyutu, G^1, G^2, .. G^n Gaussian PDF fonksiyonları olmak üzere, fonksiyonun ortalama vektörü, μ, şu şekilde bulunur.

$$\mu^i = \{\mu_1^i, \ldots . \mu_k^i\} = \{s_1^i, \ldots s_2^i\} \quad \textbf{(4.16)}$$

Ulaşılan her çözüm tabloya eklenirken, uygunluk değerine göre sıralanır. Daha sonra ağırlık vektörü, ω, aşağıdaki formüle göre hesaplanır.

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \quad \textbf{(4.17)}$$

Bu formül, ortalama değeri μ=1.0, standart sapması σ=*qk* olan Gaussian fonksiyonunun, '*l*' değişkenine uygulanmasıdır. *q* ise, KKO'da daha iyi çözümlerin seçilme olasılığını artıran parametredir.

Bir çözümün seçilmesi ve güncellenmesi için, her Gaussian fonksiyonunun seçilme olasılığı, denklem 4.17 ile hesaplanan, ağırlık değerine bağlı olarak değerlendirilir. Her fonksiyonun olasılığı şu formülle hesaplanır.

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \quad \textbf{(4.18)}$$

Hesaplanan olasılık değerine göre seçilen Gaussian fonksiyonunun parametreleri standart sapma, σ , değerine göre güncellenir. σ değeri ise şu formülle hesaplanır.

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{\left| s_e^i - s_l^i \right|}{k-1} \qquad \textbf{(4.19)}$$

σ, seçilen çözümün diğer çözümlere olan ortalama uzaklığı olarak tanımlanabilir. ξ parametresi, Karınca Kolonisi Optimizasyonunda feromon buharlaşmasını (ρ)temsil eder.

Sürekli KKO (ACO_R)algoritması çizelge 4.2 ile verilmiştir.

Çizelge 4.2: ACO_R algoritması

Rasgele çözüm uzayı oluştur
while sonlanma şartları sağlanmadıkça **do**
uygunluk hesapla, çözümleri değerlendir
çözümleri uygunluk değerlerine göre azalan sırada sırala
ağırlık vektörünü hesapla, (denklem 4.17)
olasılık değerine göre t^* Gaussian fonksiyonunu seç (denklem. 4.18) .
seçilen çözümnün σ değerini hesapla (denklem 4.19).
Seçilen çözümün σ değerini kullanarak, seçilen çözümden yeni çözüm üret
Çözüm zayının sonundaki t tane çözümü yeni üretilen çözümlerle değiştir
end while

5. EL YAZISI KARAKTER TANIMA

El yazısı tanıma, el ile yazılan harf, rakam ve sembollerin bilgisayar sistemleri tarafından tanınmasıdır. İnsanlar için oldukça kolay olmasına rağmen, bir zemin üzerindeki çizgi ve eğrilerin doğrudan harf ve rakamlar, daha ileri aşamada da anlamlı sözcükler olarak algılanması oldukça zor bir problemdir.

El yazısı tanıma probleminin öncüsü Optik Karakter Tanıma (Optical Character Recognition) problemi olmuştur. Optik Karakter Tanıma problemleri, örüntü tanıma (pattern recognition) problemlerinin en eski alanlarından biridir. Örüntü Tanıma problemlerinin ilk zamanlarında, hemen her araştırmacı Optik Karakter Tanıma ile ilgili bir konu üzerinde çalışmayı tercih etmiştir. Çünkü karakterler birçok araştırmacı tarafından bilimsel araştırmalar için elverişli bulunmuştur. Karakter tanıma ise, kolay çözümlenebilir bir problem olarak görülmüştür. Ancak umulanın aksine, bazı basit başlangıç geliştirmelerinden sonra, karakter tanıma problemlerinde büyük zorluklarla karşılaşılmıştır [40].

OKT ile ilgili ilk patentler 1929'da Tausheck (Almanya) ve 1933'te Hendel (Birleşik Devletler) tarafından alınmıştır. Bunlar Optik Karakter Tanıma ilgili ilk fikirler olmuştur. Bilgisayarların gelişimine, 1950lere, kadar harfleri ve sayıları okuyabilen makineler hayal edilmiştir [2]. 1951 yılında M. Sheppard tarafından okuyucu – yazıcı bir robotun, GISMO, icat edilmesiyle modern optik karakter tanıma yöntemlerinin gelişimi başlamıştır. 1954 yılında, sadece büyük harfleri çok yavaş bir şekilde okuyabilen bir örnek makine geliştirilmiştir (J. Rainbow). 1967de IBM ile birlikte şirketler Optik Karakter Tanıma sistemleri üretmeye başlamıştır. Başlangıçta oldukça pahalı olan bu ticari sistemler, zamanla daha da gelişmiş ve ucuzlamıştır. Çok farklı yazı tiplerinde, karakter setlerinde ve bozuk dokümanlarda problemler devam ediyorsa da, artık, araştırmalar el yazısı karakterlerinin tanınması alanına doğru kaymaktadır [41].

Optik Karakter Tanıma probleminde elde edilen başarı seviyesine, el yazısı tanıma alanında erişilememiştir. El yazısı tanıma probleminin çözümünü zorlaştıran bazı nedenler şunlardır.

1. Aynı karakterler şekil, boyut ve yazı stili olarak kişiden kişiye hatta aynı kişide dahi zamanla değişmektedir.
2. Görsel karakterlerde gürültü karışıklıklara yol açmaktadır.
3. Görsel karakterlerin görünümlerini sıkı sıkıya tarif edebilecek kurallar yoktur bu nedenle sistem görsel karakterler için gerekli kuralları çok sayıda örnekten yararlanarak sezgisel olarak çıkarmalıdır.

Şekil 5.1: Aynı kelime için farklı el yazısı örnekleri

Tüm bu zorlukları aşabilmiş olan insan görme sistemi, insan beyninin bazı üstün özellikleri sayesinde, karakter tanıma konusunda mükemmel derecede gelişmiş bir sistemdir. İnsan beyni, algıladığı görüntülerdeki küçük değişiklik ve hatalara çok çabuk uyum sağlayabilir. Böylece insanların çok farklı özelliklerdeki el yazıları başkaları tarafından rahatça okunabilir. Yine insan beyni deneyimle öğrenme yetisine sahiptir. Bu sayede, ilk defa karşılaşılan yazı tipi ve karakterleri, çok hızlı bir şekilde idrak edilebilir. Ayrıca insan görme sistemi, boyut değişikliklerinden, görüş açısından, karakterlerin renk, yön, konum ve oranlarından büyük ölçüde etkilenmez [42].

El yazısı karakter tanıma probleminin çözümünde kullanılan yöntemler, temel olarak iki grupta toplanabilir.

1. Etkileşimli (online) Yöntemler
2. Etkileşimsiz (offline) Yöntemler

Etkileşimli yöntemler, el yazısını yazı yazıldığı sırada tanıyan, özel olarak tasarlanmış sistemlerdir. Genelde elektromanyetik ya da elektrostatik tabletler kullanılır. Kalemin dokunuşları ve hareketlerin devamlılığı göz önünde tutulur. Elektronik ajandalar (PDA) günümüzde çok yaygınlaşan bir etkileşimli yöntemdir. Etkileşimsiz yöntemler ise genelde kağıt üzerine daha önceden yazılmış bilgilerin sayısallaştırılarak, sonradan tanınmaya çalışılması işlemidir. Etkileşimli sistemlerin yazı hızına yetişebilmesi için çok hızlı olması gerekirken, etkileşimsiz sistemlerde yazının tanınması için bir süre kısıtlaması gerekmemektedir. Etkileşimli sistemlerin üstünlüğü harflerin şekil özelliklerinin yanında yazılma sırasındaki hareketlerin de gözlenebilmesidir. Ayrıca kullanıcıyla etkileşimin en büyük faydalarından biri gerektiğinde yanlışların anında düzeltilebilmesidir. Etkileşimsiz sistemlerse, yazının yazılması sırasındaki hareketler hakkında hiçbir bilgi olmadığı ve özellikle eski belgeler yeterince temiz ve okunaklı olmayacağı için yanılması daha kolay sistemlerdir. Sayısallaştırıcılardan kaynaklanan gürültülerin etkisini azaltmak için çok daha detaylı bir ön işlemeye gerek duyulur. Ancak bu sistemlerin üstünlüğü özel bir donanıma gerek duyulmaması ve bu sayede yıllardır var olan bütün belgelere uygulanabilmesidir.

5.1 Etkileşimli Karakter Tanıma Yöntemleri

Etkileşimli yöntemler, bir metin yazıldığı sırada, sayısallaştırıcı yardımıyla o metni işleyen yöntemleri, metni yazan kalemin konum-zaman, hız-zaman, ivme-zaman gibi fonksiyonları hakkında bilgi toplayabilen araçları kapsar [43]. Etkileşimli sistemler genellikle elektronik tabletler tarafından kalem hareketlerinin koordinatlarının elde edilmesiyle el yazısı ya da çizimlerin otomatik olarak algılanmasını sağlar.

Etkileşimli el yazısı tanıma yöntemlerinde kullanılan kalem bilgisayarlar (pen-based computers) ilk olarak Kay tarafından keşfedilmiştir (1968). Sonrasında birçok araştırmacı tarafından geliştirilmiştir. Kalem bilgisayarlarda hedef, elektronik mürekkebin otomatik işlenmesiyle kalem ve kağıdın taklit edilebilmesidir. Kalem bilgisayarların bu konudaki başarısı, donanımsal özelliklerinin yanı sıra, el hareketlerinin ve yazılan metnin tanınabilmesini sağlayan sisteme bağlıdır. Bu

sistemlerin çoğu, zaman içindeki tüm hareket bilgisini değil, vuruşların sıklık ve düzenini inceler [43].

Tabletler için değişik teknolojiler arasında iki ana sınıftan bahsedilebilir: elektromanyetik / elektrostatik ve basınç duyarlı sistemler. Son yıllardaki gelişmeler de girdi ve çıktının aynı yüzey üzerinden olmasına olanak sağlamıştır.

Etkileşimli sistemlerde, sayısallaştırma işlemi sırasında, ani ya da kararsız el / parmak hareketleri nedeniyle, kalem hareketlerindeki (pen-up/pen-down) hatalar nedeniyle gürültü (noise) oluşur. Bu gürültü veri yumuşatma (data smoothing), işaret filtreleme, inceltme gibi işlemlerle azaltılmaya çalışılır.

Kullanılan sistemlerin çoğu el yazısı karakterlerin ve el hareketleri komutlarının düzgelenmesine ihtiyaç duyar. Düzgeleme işlemleri, karakter boyutlarının, yazım eğiminin düzeltilmesi şeklinde olabilir [43].

Bölütleme (segmentation) problemi sistemin etkileşimli olması nedeniyle farklılıklar gösterir. Etkileşimli sistemlerin en büyük üstünlüğü kullanıcının sisteme yardımcı olmasının sağlanabilmesidir. Bölütleme için ilk uygulanan yöntemlerden biri harf bittiğinde bir işaretle kullanıcının bunu belirtmesidir. Bazı sistemlerdeyse kullanıcı harfleri kutucuklara yazmaya zorlanmakta dolayısıyla bölütleme büyük bir problem olmamaktadır. Diğer bazı yöntemlerde etkileşimsiz sistemlere benzer şekilde harfler arasındaki boşluklardan yararlanılır, ancak bu sefer bu boşlukların bulunmasında harfleri yazma sırasında harcanan zaman da önem taşır. Kalemin tabletten kaldırıldığı ve yeniden konulduğu süre arasındaki fark belli bir eşik değerini aştığında yeni bir harfe başlandığı varsayılır.

Harflerin doğru tanınmasında durağan ve değişken özellikler önem taşır. Durağan özellikler etkileşimsiz sistemler tarafından da kullanılan, büyük ve küçük harflerin farklı büyüklüklere sahip olması, 'g', 'y', 'j' gibi harfler referans çizgisinin altına uzanırken 'l', 'k', 'b' gibi harflerin diğer harflerden daha uzun olması, bazı harflerin noktalı olması gibi özelliklerdir. Bunlar harflerin birbirinden ayrılması için önem taşır. Değişken özellikler ise etkileşimli sistemlere özgü olan, bir harf yazılırken kalemin ilk dokunuşunun nerede olduğuna, daha sonra kalemin nasıl bir hareket izlediğine, tek bir harf sırasında elin kaç kez kalkıp indiğine benzer özelliklerdir. Etkileşimli sistemler, el yazısı tanımak için her iki özelliği de kullanır.

Yazı karakterleri basit ve sabit harflerle kısıtlandığında tanıma oranı oldukça yükselmektedir. Etkileşimli sistemlerde, kullanıcı sisteme gün geçtikçe uyum sağlayabilir ve yazılarını tanınması daha kolay olacak şekilde yazmaya başlayabilir. Ayrıca hataları anında düzeltebilir.

5.2 Etkileşimsiz Karakter Tanıma Yöntemleri

Etkileşimsiz karakter tanıma yöntemlerinde, kağıt üzerine yazılmış olan metinler, önce sayısallaştırılarak bilgisayar ortamına aktarılır. Bundan sonra tanıma algoritmasının işleyeceği ikili ya da gri seviye karakter resmi elde edilir. Etkileşimsiz yöntemlerde yazımın yapıldığı araç ve diğer ortam değişkenleri kontrol edilemediği için, etkileşimli yöntemlere göre bir takım zorlukları vardır. Sayısallaştırma işlemi sırasında ortaya çıkan problemler de etkileşimsiz yöntemler tarafından çözülebilmelidir.

Aşağıdan yukarıya (buttom-up) yaklaşımıyla çalışan etkileşimsiz yöntemlerde tanıma işlemi, benek seviyesinde başlar ve anlamlı bir kelime bulunmasıyla sonlanır. Bu yaklaşımda hiyerarşik olarak yapılacak işlemler aşağıdaki şekilde gruplandırılır.

1. Ön İşleme (Pre-processing)
2. Bölütleme (Segmentation)
3. Öznitelik Çıkarma (Feature Extraction-Representation)
4. Tanıma (Recognition)

Etkileşimsiz yöntemler oldukça geniş bir yelpazede yer alsa da genelde benzer bir işlem sırasını izler. Bazı yöntemlerde, bazı aşamaları birleştirilebilir ya da kullanılmayabilir [44].

5.2.1 Ön işleme (pre-processing) yöntemleri

Etkileşimsiz yöntemlerin üzerinde çalışacağı ham veri, yazılı karakterlerin sayısallaştırılmasıyla oluşur. Karakterler yazılırken tanıma işlemini zorlaştıracak ve başarımını etkileyecek şekil ve boyut bozuklukları oluşmuş olabilir. Sayısallaştırma işlemi sırasında veri üzerinde problemler ortaya çıkabilir. Veri üzerinde yeni noktalar, çizgiler, boşluklar oluşabilir. Ön-işleme aşamasında, etkileşimsiz bir yöntemin kesin ve doğru bir tanıma işlemi yapabilmesini kolaylaştıracak veriyi elde

etmek ve sisteme sunmak amaçlanır. Ön-işleme aşamasında yapılması beklenenler şunlardır [44].

- Gürültünün azaltılması (noise reduction)
- Verinin düzgelenmesi (normalization)
- Verinin sıkıştırılması (compression)

5.2.1.1 Gürültü azaltma

Gürültü, yazma aracından ya da yazı bilgisayara aktarılırken kullanılan tarama cihazından kaynaklanır. Gürültü, karakterleri oluşturan bağlı çizgilerin ayrılması, çizgi üzerinde tümseklerin ya da boşlukların oluşması, karakter çizgileri dışında anlamsız nokta veya çizgilerin oluşması, karakterlerde boşluk olması gereken yerlerin dolması gibi problemler oluşturabilir. Verilerde, karakterlerin bozulması, bükülmesi, genleşmesi, kayması, köşelerin yumuşaması gibi şekil bozuklukları da sorun oluşturur.

Bahsedilen tüm bu problemlerin gürültü azaltma aşamasında elenmesi gerekmektedir. Gürültü azaltma amacıyla kullanılan yüzlerce teknik, üç ana grupta toplanabilir.

- Filtreleme (Filtering)
- Morfolojik işlemler (Morphological Operations)
- Gürültü modelleme (Noise Modelling)

Filtreleme işleminde temel fikir, önceden tanımlanmış bir maskenin, görüntü üzerinde gezdirilerek, görüntünün her pikselinin değerinin, yakın çevresindeki komşu piksellerinin değerleri arasındaki bağıntıyla yeniden hesaplanmasıdır. Filtreler, görüntüyü yumuşatmak, keskinleştirmek, eşiklemek, görüntüden arka planı kaldırmak ya da görüntüde zıtlık ayarlaması yapmak amacıyla özel olarak tanımlanabilir.

Morfolojik işlemlerde temel fikir ise, görüntü üzerinde gezdirilecek filtrede mantıksal operatörlerin kullanılmasıdır. Morfolojik işlemler, karakter görüntülerinde, gürültü nedeniyle ayrılmış çizgilerin birleştirilmesinde, birleşmiş çizgilerin ayrıştırılmasında, hatların yumuşatılmasında, karakterlerin inceltilmesinde ve sınırların belirlenmesinde kullanılabilir.

Gürültü modelleme ise, gürültünün modellenebilir yapıda olması durumunda, bazı ölçümleme (kalibrasyon) teknikleri kullanılarak giderilmesidir. Çoğu durumda gürültü modellenecek yapıda olmaz. Ancak optik bozulmalarda (beneklenme, bulanıklaşma, eğrileşme gibi) gürültü modelleme üzerinde çalışılmaktadır [44].

5.2.1.2 Verinin düzgelenmesi

El yazısı tanıma problemlerindeki en büyük sorunlardan biri, yazım şeklinin, karakterlerin şekil ve boyutlarının kişiden kişiye, hatta aynı kişide zaman içinde değişebilmesidir. Düzgeleme (normalizasyon) yöntemleri, yazım şeklindeki değişiklikleri ortadan kaldırarak, ortalama bir veri elde etmeyi amaçlar. Düzgeleme işlemleri dört grupta toplanabilir [44].

- Karakterlerdeki eğikliğin düzeltilmesi
- Yatay olarak düz yazılmamış metnin düzeltilmesi ve referans çizginin belirlenmesi
- Boyut düzgelemesi
- Karakterlerin hatlarının yumuşatılması

5.2.1.3 Verinin sıkıştırılması

Karakter tanıma uygulamalarında, karakter görüntülerinin daha az yer kaplaması ve tanıma işleminin daha hızlı yapılabilmesi için verinin sıkıştırılması gerekmektedir. Klasik görüntü sıkıştırma yöntemlerinde, görüntü bir uzaydan başka bir uzaya dönüştürülür. Karakter tanıma uygulamalarında kullanılan sıkıştırma yöntemlerinin, görüntüyü sıkıştırırken şekil bilgisini koruması gerekmektedir. Bu amaçla en sık kullanılan iki yöntem şunlardır:

- Eşikleme (Thresholding)
- İnceltme (Thinning)

Karakter tanıma sistemine sunulan renkli ya da gri seviye görüntülere eşikleme işlemi uygulanarak ikili resimler (siyah-beyaz resim-binary image) elde edilir. Bu yolla, görüntü üzerinde tüm piksel değerleri siyah ya da beyaz olduğu için görüntünün boyutu azalmaktadır. Ayrıca karakterin kendisi ile arka plan birbirinden ayrılmış olmaktadır. Bunlar da tanıma uygulamasının performansını artırmaktadır. Eşikleme işlemi yerel ve genel olarak iki şekilde uygulanabilir. Genel eşikleme

sırasında, bir eşik değeri (threshold) seçilir ve tüm görüntü üzerindeki pikseller bu eşik değerinden büyük ya da küçük olmalarına göre, arka plan ya da karakter pikseli olarak işaretlenir. Yerel eşiklemede ise, görüntünün farklı alanları için farklı eşik değerleri belirlenir. [43].

İnceltme işlemi, yazım sırasında oluşan fazla çizgi ve beneklerin silinerek, hem karakterin şekil bilgisini ortaya çıkarmaya, hem de görüntünün boyutunu azaltmaya çalışır. İnceltme işleminde, bir piksel genişliğinde karakter iskeleti yakalamaya çalışılır. Ancak bu yöntem, gürültüden çok etkilenebilir ve karakterin gerçek şekil bilgisini bozabilir. Diğer yaklaşım türünde ise, inceltme işlemi sırasında, tüm pikseller işlenmeden, görüntü üzerindeki bazı genel bilgiler kullanılır. Bu bilgiler merkez çizgi (center-line), ortanca değeri (median), karakterin bitiş, geçiş, kapalı halka noktaları gibi özel noktaları olabilir.

5.2.2 Bölütleme (segmentation) yöntemleri

Önişleme aşamasının sonunda, gürültüden temizlenmiş, sıkıştırılmış, şekil bilgisi çıkarılmış, düzgelenmiş bir görüntü elde edilir. Bölütleme aşamasına ulaşan bu görüntünün kendisini oluşturan anlamlı alt parçalara ayrılması gerekir. Bölütleme aşamasında, doküman satırlara, kelimelere ve harflere ayrılır. Bu ayrımın doğruluğu, tanıma uygulamasının başarısını doğrudan etkiler

Dış bölütleme işlemi dokümanın paragraf, cümle, satır yada kelime gibi parçalara ayrılması işlemidir. Doküman analizi her ne kadar kendine ait metodolojisi olan ayrı bir araştırma alanı olsa da, karakter tanıma sistemlerinde de yazı olan ve olmayan alanların ayrıştırılabilmesi gerekmektedir.

İç bölütleme ise kelimeleri oluşturan karakterlerin ayrıştırılabilmesidir. Açık (explicit)ve kapalı (implicit) bölütleme olarak iki şekilde uygulanabilir. Açık bölütlemede, parçalar karakter temelli özellikler kullanılarak bulunur. Karakterler arası boşluklar, yatay izdüşüm analizi, bağlı birleşen analizi kullanılarak bölütleme yapılmaya çalışılır.

Kapalı bölütleme yöntemleri tanıma işlemine dayalıdır. Bu yöntemler, görüntü üzerinde önceden tanımlanmış sınıflarla eşleşen parçaları bulmaya çalışır. Uygulanan yöntemlerin bazılarında tüm görüntü, içeriğine bakılmaksızın, sistematik olarak parçalara ayrılır. Bu parçaların bazıları birbiri üzerine yayılmış olabilir. Gezgin bir pencere parçalar üzerinde gezdirilerek muhtemel bileşenler belirlenmeye çalışılır. Bu

bileşenler ancak karakter tanıma bölümünde kesinleştirilir. Bunlar genelde makine yazımı kelimelerin ayrıştırılmasında kullanılır [44].

Şekil 5.2: Bölütlenmiş el yazısı

5.2.3 Öznitelik çıkarma (feature extraction) yöntemleri

Öznitelik çıkarma karakter tanıma sisteminin en önemli bileşenlerinden biridir. Birçok tanıma sisteminde, tanıma işleminin karmaşıklığını azaltmak ve tanıma performansını artırmak amacıyla verinin daha yoğun ve karakteristik olarak tutulması sağlanır. Sınıfları birbirinden ayıracak bir takım özellikler toplanır. Bu özellikler bir sınıf içinde değişmezken, sınıftan sınıfa farklılık göstermelidir. Öznitelik bulma sorunu işlenmemiş veriden sınıflar arası değişintiyi yüksek tutan ve sınıf içi değişintiyi azaltan nitelikteki bilgilere ulaşmaktır. Bir karakter tanıma sisteminde yüksek başarıma ulaşmaktaki anahtar güçlü öznitelikler bulmaktır [45]. Öznitelik bulma problemi için geliştirilmiş yüzlerce yöntem üç ana grup altında toplanabilir.

1. Dönüşümleri ve seri açılımlarını kullanan yöntemler
2. İstatistiksel öznitelik çıkarma yöntemleri
3. Geometrik öznitelik çıkarma yöntemleri

5.2.3.1 Dönüşümleri ve seri açılımlarını kullanan yöntemler

Sürekli bir işaret, sınıflandırma işlemi için gerekenden daha fazla bilgiyi içerir. Böyle bir işaret, tanımlanmış bazı fonksiyonların serilerinin doğrusal bir birleşimi ile temsil edilebilir. Bir işaretin bu şekilde lineer kombinasyonun katsayıları ile ifade edilmesi dönüşüm ya da seri açılımı olarak isimlendirilir. Sıklıkla kullanılan dönüşüm ve seri açılım yöntemleri şunlardır.

- Fourier dönüşümü
- Gabor dönüşümü
- Wavelet dönüşümü

- Momentler
- Karhunen – Loeve Dönüşümü

5.2.3.2 İstatistiksel öznitelik çıkarma yöntemleri

Karakter imgelerinin istatistiksel yöntemlerle elde edilen özniteliklerle temsil edilmesi, karakter tanıma sisteminin karmaşıklığını azaltıp, hızını artırır. Bazı istatistiksel öznitelik çıkarma yöntemlerinde, karakteri içeren çerçeve bölgelere ayrılır. Farklı bölgelerdeki bazı özellikler (piksel yoğunluğu, çevrit yönü, ortalama sayısı, 1den 0'a geçiş sayısı…vs) incelenir ve bu özelliklerle karakter temsil edilir. Örneğin çevrit yönü (contour direction) özelliği, karakter hattının yönelimini ölçer. Görüntünün dikdörtgensel ve diyagonal bölgelere ayrılmasıyla elde edilir. Bu bölgelerdeki zincir kodların (chain code) histogramları hesaplanır ve bu yolla karakter çevritinin sağa, sola, aşağı, yukarı dönüşleri bulunur. Bu yöntemlerle karakterin kıvrılma noktalarının, son ve uç noktalarının bulunması sağlanabilir.

Bir diğer istatistiksel öznitelik, karakter hattı üzerindeki geçiş noktalarıdır. Örnek olarak karakter üzerindeki dikey çizgiler üzerinde arka plandan karaktere geçişlerin sayısı ve yeri, belirli bir alanda çizgisel bölütler arası uzaklık, öznitelik olarak alınabilir.

5.2.3.3 Geometrik öznitelik çıkarma yöntemleri

Karakterlerin değişken yerel ve genel özellikleri, bozulmalardan ve yazan kişiden kaynaklanan değişikliklerden etkilenmeyen geometrik ve topolojik özniteliklerle temsil edilebilir. Öznitelik çıkarmak için kullanılan dönüşümler, insan algılama mekanizmasına benzer hiçbir bir model kabul etmez, yalnızca istatistiksel olarak el yazısı karakterlerinin nasıl oluştuğuyla ilgilenir. Aynı karakterin farklı çeşitlerde yazılmış olanlarını tanımak için, genelleşmiş bazı ek öznitelikler kullanılmalıdır. Karakterin son noktaları ve bağlantı noktaları, özellikle Türkçe'deki 'Ç' ve 'Ş' harfleri için, karakterle ilgili sağlıklı bilgiyi taşıyabilecek şekilsel özniteliklerdir. Çoğu karakter tanıma sistemi karaktere ait beneklerle ilgilenmektedir, fakat karakteri çevreleyen arka planın da boyut, şekil ve pozisyonu ile ilgili bilgileri de şekilsel öznitelikler arasında sıralanabilir. Karakterdeki bağlantılı bileşenlerin sayısı da, özellikle 'İ', 'Ü' ve 'Ö' gibi bazı Türkçe karakterler için şekilsel öznitelik olarak kullanılabilir [45].

5.2.4 Tanıma (recognition) yöntemleri

Karakter tanıma sistemleri genel olarak örüntü tanıma (pattern recognition) yöntemlerini kullanırlar. Burada ana fikir, sayısallaştırılmış bir karakterin daha önceden belirlenmiş sembolik sınıflardan doğru olana atanabilmesidir [43]. Karakter tanıma alanında kullanılan çeşitli yöntemlerden bazıları şunlardır.

1. Şablon Eşleme Yöntemleri (Template Matching)
2. İstatistiksel Analiz Yöntemleri (Statistical Techniques)
3. Yapay Sinir Ağları (Artificial Neural Networks)

Bu yöntemlerle çalışan karakter tanıma sistemlerinin bazıları bütünsel (holistic), bazıları ise analitik yaklaşımlarla tanıma aşamasını gerçekleştirirler. Bütünsel tanıma, yukarıdan-aşağıya yaklaşımıyla, karakterlere ayırmadan tüm kelimeyi tanımaya çalışır. Bu şekilde bölütleme aşaması ve bu aşamanın zorlukları sistemden elenmiş olur. Ancak tüm kelimeyi temsil edecek özniteliklerin bulunması, karakter özniteliklerinin bulunmasından çok daha karmaşık bir problem ortaya koyar. Bu problem sistemin tanıma performansını azaltır.

Analitik yöntemler ise aşağıdan-yukarıya yaklaşımıyla, karakter seviyesinden tanıma işlemine başlar. Bu sistemlerde, başarılı bir bölütleme yapılması gerekmektedir. Bölütleme aşamasının sisteme ek bir karmaşıklık getirir ve bu aşamadaki yapılan hatalar, tanımanın sonucuna da aynen yansır.

5.2.4.1 Şablon eşleme yöntemleri

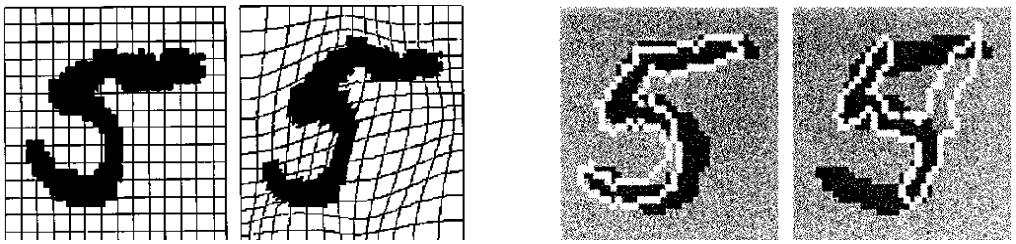
Karakter tanım sisteminde, tanıma bölümünde kullanılacak veri, tek bir karakterin gri-seviye görüntüsü gibi basit yapıda ya da karakter özelliklerinin çizgesi gibi daha karmaşık yapıda olabilir. Tanımada en basit yöntemlerden biri, tanınacak karakterlere karşılık, örneklerinin tutulmasıdır. Eşleme işlemi, tanınacak karakter ile saklanan örnekleri temsil eden özellik vektörleri arasındaki benzerlik derecesinin belirlenmesidir. Şablon eşleme yöntemleri şu şekilde gruplandırılabilir.

- Doğrudan eşleme
- Değiştirilebilir şablonlar

Doğrudan eşleme yöntemlerinde, gri-seviye ya da ikili yapıdaki karakter, tutulan örneklerle doğrudan karşılaştırılır. Seçilen benzerlik ölçütüne (Euclidean,

Mahalonobis… vb) göre şablon eşleme gerçekleştirilir. Eşleme yapılırken bire bir karşılaştırma ya da seçilen benekler (piksel) karar ağacı kullanılarak test edilir. K-en yakın komşu (k-nearest neighbour) yöntemi de doğrudan eşleme yöntemidir. Burada tanıma, tanınacak öznitelik vektörüne en yakın öznitelik vektörlerinin bulunmasına bağlıdır. En yakın öznitelik vektörlerini bulurken çeşitli metrikleri kullanılabilir. Tanıma yapılacak öznitelik vektörüne en yakın k tane komşu bulunur. Daha sonra bu k komşu en fazla hangi sınıfa ait ise, o sınıf tanıma sonucu olarak atanır [45]. Doğrudan eşleme yöntemi gürültüden fazla etkilenir.

Değiştirilebilir şablonlar kullanılarak, bilinmeyen bir karakter bilinen örneklere ait şablonların değiştirilmesiyle tanınmaya çalışılır. İki karakter, birinin şablonu diğerinin hatlarına uyması için değiştirilerek eşleştirilmeye çalışılır. Farklılık ölçütü, şablonun ne kadar değiştirildiği, kenarlara ne kadar uyum gösterdiği ile ölçülür.



Şekil 5.3: Değiştirilebilir şablonlar

5.2.4.2 İstatistiksel analiz yöntemleri

İstatistiksel örüntü tanıma yönteminde, sınıflama algoritmaları, istatistiksel analiz üzerine kurulmuştur. Aynı sınıfa ait örüntüler, istatistiksel olarak tanımlanan benzer karakteristiklere sahiptirler. Bu yöntemde, özellik olarak nitelendirilen karakteristik ölçümler, giriş örüntü örneklerinden çıkarılır. Her örüntü bir özellik vektörü ile tanımlanır. Genelde sınıflandırıcıyı oluşturan karar ve sınıflandırma yöntemleri üzerinde önemle durulur. Sınıflandırıcı tasarımı, ölçümler ve olasılıklar gibi işlenebilir örüntü bilgilerini birleştirmeyi esas alır. Böylece sınıflama, giriş veri uzayının olasılık yoğunluk fonksiyonlarının tahmini üzerine kurulu bir istatistiksel yapıdır [46].

Parametrik yaklaşımda, karakterler ilse ilgili önsel bir bilgi mevcuttur ve her karaktere karşılık parametrik bir model oluşturulabilir. Bazı olasılık değerlerine göre toplanan model parametrelerine, bazı karar kuralları uygulanarak sınıflandırma

yapılır. Maksimum olabilirlik ve Bayes tahmin edici parametrik karar vericilere örnektir.

Parametrik olmayan yöntemler çok defa dağılımlardan serbest yöntemler olarak da anılmaktadır, çünkü verilerin bilinen belirli olasılık dağılımı gösteren kaynaklardan geldiği varsayımına dayanmamaktadır.

Diğer bir istatistiksel analiz yöntemi olan Gizli Markov Modelleri,dinamik bir değişkene bağlı olarak, durağan olmayan bir şekilde değişen öznitelikleri açıklamak için kullanılan bir modeldir. GMM'lerde sonlu sayıda durum (state) tanımlanır ve özniteliklerin bulundukları duruma göre sabit bir olasılık dağılımından üretildikleri varsayılır. Böylece özniteliklerdeki durağansızlık durum değiştirme yöntemiyle açıklanmış olur. Bir Markov zinciri, sonlu sayıda durum ve bu durumlarla birleştirilmiş rasgele fonksiyonlar kümesinden oluşur. Ayrık zamanda, süreç bir durumda iken bu duruma karşılık gelen rasgele fonksiyonla bir gözlem üretilir. Markov zinciri, mevcut durumun geçiş olasılığı kullanılarak durum değiştirir. Bir karakter sistemin çıkışı olarak üretilene kadar, sistem bir durumdan diğerine geçer. Her durum olasılık değerlerine göre çıkış üretir.

Bulanık sınıflandırıcı ise, genel olarak, bir örüntünün maksimum üyelik derecesine sahip olduğu sınıfın bulunmasını sağlar [46]. Bulanık sınıflandırıcılar, önsel bir bilginin olmadığı, olasılık hesabının yapılamadığı durumlarda gerçekçi sonuçlar üretirler. Karakterleri oluşturan vuruşlar bulanık yapıda olduğu için, benzerlik ölçütü olarak bulanıklık kavramı kullanılabilir. Karakter tanıma sisteminde, bilinmeyen giriş karakteri, örnek karakterlerle eşleştirilir. En yüksek benzerlik derecesine sahip örnek karakter sınıfına atanır [44].

5.2.4.3 Yapay sinir ağları

Yapay Sinir Ağları yüksel paralellik yetenekleri nedeniyle karakter tanıma problemlerinde sıklıkla kullanılır [6, 47, 48]. Yapıları nedeniyle giriş değerlerinde oluşan değişikliklere uyarlanabilirler. YSA, örüntüler hakkındaki ilişkiyi belirli bir algoritmaya dayanarak çözmek yerine o ilişkiyi gösteren örüntü örneklerini incelemek suretiyle çözümler üretirler. Ağ, kendisine gösterilen örnekleri tekrar tekrar inceleyerek aradaki ilişkiyi kavramaya çalışır. Her yeni örnek, ağın sahip olduğu bilgiye bir yenisini ekler ve bu işlem tekrar ettikçe ilgili örüntü sınıflandırma problemi hakkında bazı genellemeler yapılır. Alışagelmiş bir takım sınıflandırma

karakteristiklerinde, istenen çıkışı üretmek için tam olarak doğru girişlere gereksinim duyulmasıdır. Diğer yandan YSA, girişlerinde değişimler olsa bile doğru çıkışı üretebilirler. YSA, kendilerine gösterilen bir örüntüyü daha önce öğrendikleri ile mukayese ederek ve aradaki benzerlikleri ortaya koyarak, belirli sınıflara ayırma özelliklerine sahiptirler. Karakter tanıma sistemlerinde en çok kullanılan YSA ileri beslemeli çok katmanlı yapay sinir ağıdır. Çok katmanlı ileri beslemeli YSA, parametrik olmayan bir sınıflandırıcıyı göz önünde tutmaktadır. Verinin temelini teşkil eden yapı hakkında tahmin yapılamayacağı gibi, aynı zamanda bu bir serbest model tahmin edicidir. Üç katmanlı ağ eğitim verisinin sonrasal olasılıkları, doğrudan tahmin etmek için yaygın olarak kullanılır. Yeterince verilen örnekler, giriş verisinden çıkış kategorisi elde edilmesiyle herhangi bir rastgele seçilmiş doğrusal olmayan eşleme ile öğrenme gerçekleşebilir [46].

6. KULLANILAN SINIFLANDIRICILAR VE ÖZNİTELİK ÇIKARMA YÖNTEMLERİ

6.1 Kullanılan Sınıflandırıcılar

Tez kapsamında sınıflandırıcı olarak kullanılan yapılar aşağıdaki gibidir.

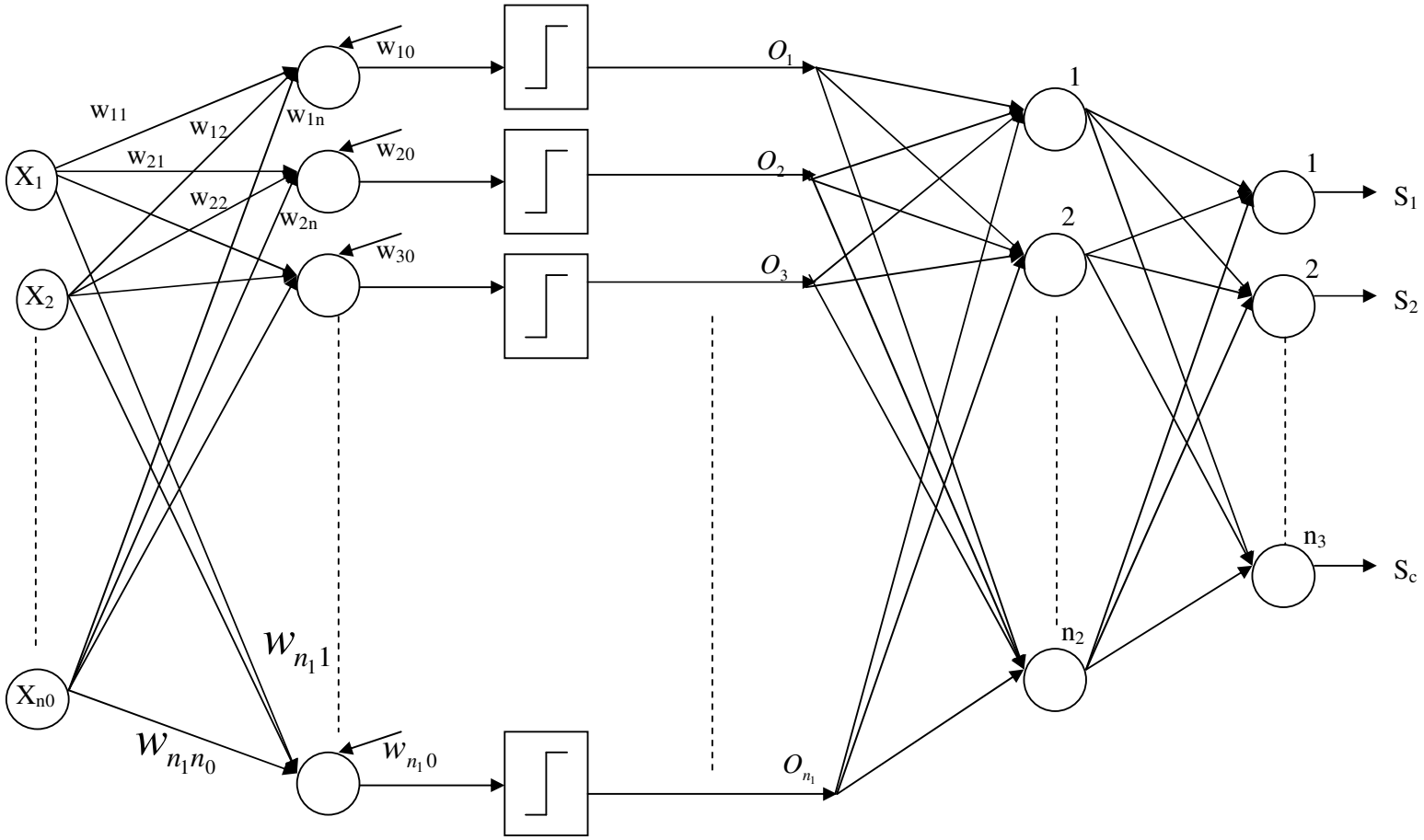
1. Karınca Kolonisi Optimizasyonu algoritması ile eğitilmiş çok katmanlı yapay sinir ağı (ACO_R-ÇKA)
2. Genetik algoritmalar ile eğitilmiş çok katmanlı yapay sinir ağı (GetÇKA)
3. Geriye yayılım algoritması ile eğitilmiş çok katmanlı yapay sinir ağı (BP-ÇKA)
4. K-En yakın komşu sınıflandırıcısı (kNN)

6.1.1 KKO ile eğitilmiş çok katmanlı yapay sinir ağı (ACO_R-ÇKA)

Tez kapsamında, sınıflandırıcı olarak kullanılacak çok katmanlı yapay sinir ağının eğitiminde KKO algoritmasından faydalanılmıştır. Çok katmanlı ağın eğitimi ile, ağın giriş katmanındaki ağırlıkların en iyi değerlerinin hesaplanması amaçlanmaktadır.

Şekil 6.1 ACO_R-ÇKA'nın yapısını göstermektedir. Ağ, giriş katmanı hariç, üç katmandan oluşmaktadır. Şekilde giriş katmanına uygulanan öznitelik vektörünün boyutu n_0 ile gösterilmiştir. n_1, 1. katmandaki nöron sayısını, n_2, 2. katmandaki nöron sayısını, n_3 ise 3 katmandaki (çıkış katmanı) nöron sayısını göstermektedir. Ağın çıkışındaki nöronlar c adet sınıfı (S_1, S_2,…, S_c) temsil etmektedir. Dolayısıyla çıkış katmanı nöron sayısı (n_3) ile sınıf sayısı (c) eşittir.

ACO_R-ÇKA'nın 1. katmanında öz nitelik uzayı hiper düzlemler ile bölgelere ayrılır. Örneğin öz nitelik boyutu 2 iken (n_0=2) iki boyutlu düzlemde doğrular oluşacaktır. Aşağıdaki denklem n_0 boyutlu öznitelik uzayının bir hiper düzlem ile iki bölgeye ayrılmasını göstermektedir. Denklemde [x_1……x_{n0}] vektörü giriş vektörünü



Şekil 6.1: Karınca kolonisi optimizasyon algoritması ile eğitilmiş çok katmanlı ağ (ACO_R-ÇKA) yapısı

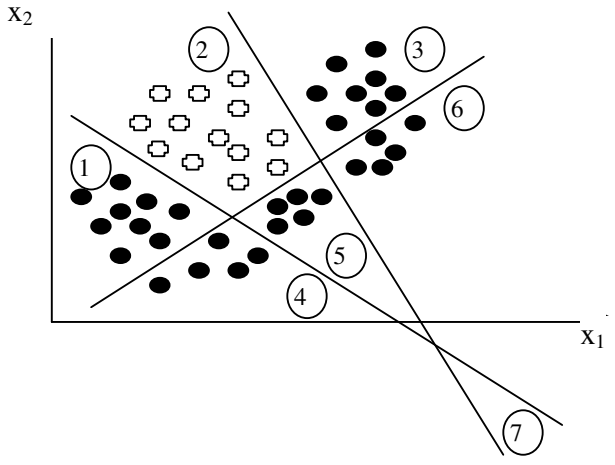
w_{i1}…….w_{in0} i. nörona ait ağırlık vektörünü göstermektedir. Nöron çıkışına aktivasyon fonksiyonu (hard-limiter) uygulandığında, çıkış (O_i), denklem sonucunun 0'dan büyük ya da küçük olmasına göre iki ayrık değerden {+1,-1} birini alır.

$$O_i = \begin{cases} +1 & w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \ldots\ldots\ldots + w_{in_0}x_{n_0} > 0 \\ -1 & w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \ldots\ldots\ldots + w_{in_0}x_{n_0} \le 0 \end{cases} \qquad \textbf{(6.1)}$$

İlk katmandaki her nöron, bir hiper düzeleme karşılık gelmektedir. Böylece öznitelik uzayı, nöron sayısı (n_0) kadar hiper düzlemle, belirli sayıda ayrık bölgeye ayrılır. İlk katmandaki nöronların ağırlıkları (w_{i1},…….w_{in0}), *W* ağırlık matrisinde toplanmıştır.

$$W = \begin{bmatrix} w_{10} & w_{11} & \ldots & \ldots & w_{1n_0} \\ : & & & & : \\ w_{i0} & w_{i1} & .. & .. & w_{in_0} \\ : & & & & : \\ w_{n_1 0} & w_{n_1 1} & \ldots & \ldots & w_{n_1 n_0} \end{bmatrix} \qquad w_{ij}\begin{cases} i: & 1.\textit{katman nöron indisi} \\ j: & \textit{giriş vektörü indisi} \end{cases} \qquad \textbf{(6.2)}$$

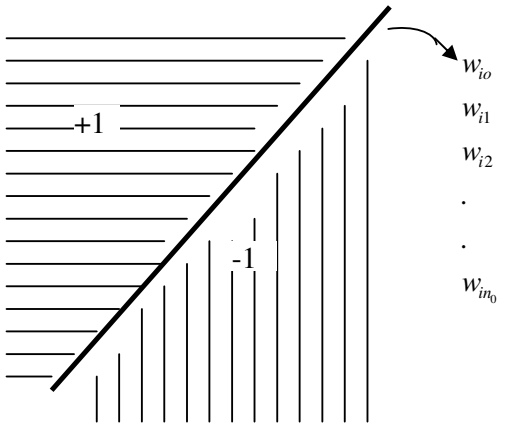
Sınıflandırmanın gerçekleştirilebilmesi için, farklı sınıflara ait giriş elemanlarını farklı bölgelerde bulunduran hiper düzlemlere ihtiyaç vardır. Bu hiper düzlemler, ağın 1. katmanındaki nöronların ağırlıklarının uygun şekilde değiştirilmesiyle elde edilir. Ağırlık değerlerinin değiştirildiği bu aşama, eğitim aşamasıdır. Eğitim aşamasında uygun sayıda nöron ve uygun hiper düzlem katsayıları bulunacaktır. Örneğin, aşağıda 2 boyutlu öznitelik uzayında 2 sınıf, 3 nöron kullanılarak birbirinden ayrılmışlardır. Oluşan hiper düzlemler (doğrular) öz nitelik uzayını (x_1, x_2), 7 bölgeye ayırmıştır.



Şekil 6.2: 3 nöronun 2 boyutlu öznitelik uzayını bölmelemesi

6.1.1.1 ACO_R-ÇKA'nın öznitelik uzayını bölmelemesi

ACO_R-ÇKA'nın 1. katmanında, nöronların çıkışlarına aktivasyon fonksiyonu uygulanacaktır. Bu aktivasyon fonksiyonunun sonucunda her nöron +1 ya da -1 sonucunu üretecektir (denklem 6.1).



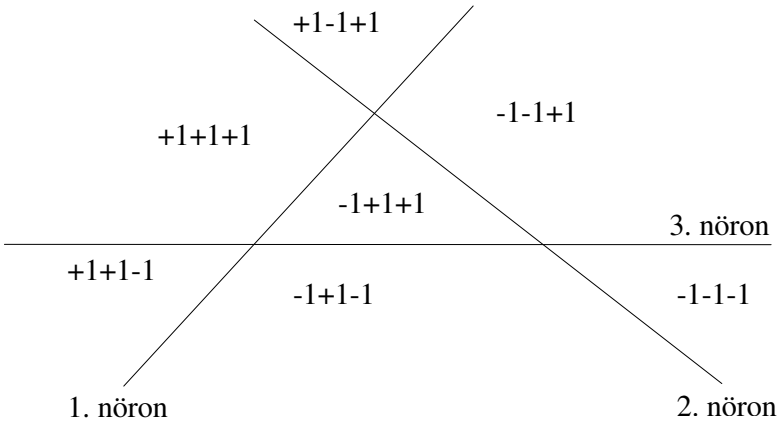
Şekil 6.3: i. nöronun öznitelik uzayını iki farklı bölgeye ayırması

Şekil 6.3'de, i. nöron ile uzay iki farklı bölgeye ayrılmıştır. -1 ve +1 bu iki bölgenin kodlarını göstermektedir. Yeni nöronların eklenmesiyle, uzayda her bölge farklı kodlara sahip olacaktır. Nöron sayısıyla uzaydaki oluşan bölge sayısı arasındaki ilişki aşağıdaki gibidir.

$$C(p,d) = \begin{cases} 2^p & p \le d \\ 2\sum_{i=0}^{d-1}\binom{p-1}{i} + \binom{p-1}{d} & p > d \end{cases} \quad \textbf{(6.3)}$$

Bu eşitlikte, p nöron sayısını, d ise öz nitelik uzayının boyutunu göstermektedir. Örneğin 2 boyutlu uzayda 3 nöron aşağıda hesaplandığı gibi en çok 7 bölge oluşturulabilir.

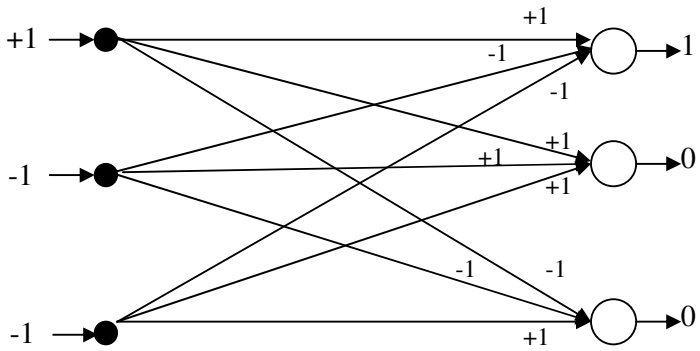
$$2.\left[\binom{2}{0} + \binom{2}{1}\right] + \binom{2}{2} = 7 \quad \textbf{(6.4)}$$



Şekil 6.4: 3 nöronun 2 boyutlu öznitelik uzayını bölmelemesiyle oluşan bölgelerin kodları

Ağın 1. katmanı ile oluşturulan hiper düzlemler kullanılarak, farklı sınıflar birbirinden ayrılırlar. Fakat Şekil 6.2'de görüldüğü gibi bir sınıf farklı bölgelerde bulunabilir. Bu durumda, aynı sınıftan örnekler içeren farklı bölgelerin birleştirilmesi işlemi sonraki katmanlarda gerçekleştirilir.

ACO_R-ÇKA'nın 2. katmanında, 1. katmanda oluşan ve eğitim kümesi elemanlarını içeren bölgelerin kodları saklanmaktadır. 2. katman sonucunda, giriş vektörünün içine düştüğü bölgeye ait nöron 1 değerini, geri kalanlar 0 değerini vermelidir. Bu katman, 1. katman çıkışını alır, sakladığı bütün bölge kodları ile karşılaştırır. Karşılaştırma yaparken Hamming mesafesini [49] hesaplar. En küçük Hamming mesafesine sahip olan bölgeye ilişkin çıkışı 1, geri kalanlarını 0 yapar. Böylece ağın 2. katmanı sonucunda, içinde bulunulan bölge belirlenmiş olur.



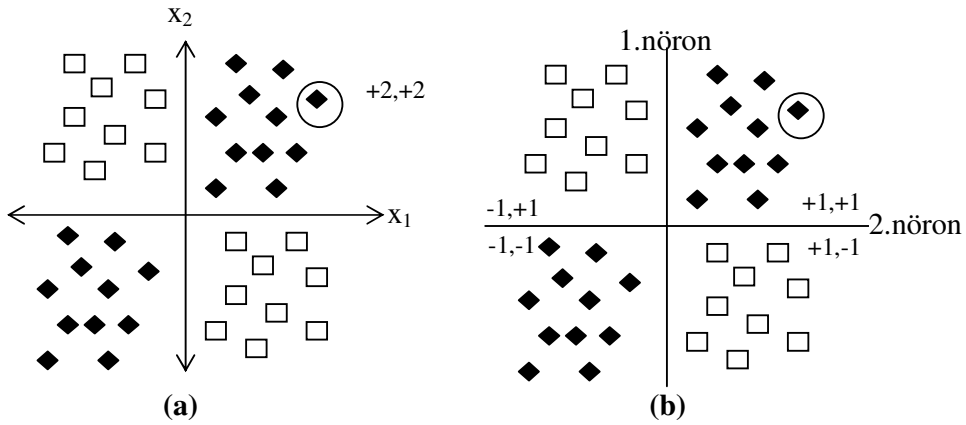
Şekil 6.5: ACO_R-ÇKA'da 1. katman çıkışında bir bölge kodunun 2. katmanda nöron çıkışına aktarılması

Şekil 6.5'de ağın 1. katman çıkışının [+1,-1,-1] olduğu varsayılmıştır. 2. katmandaki nöronlardan her birinin ağırlıkları ilgili bağlantılar üzerinde gösterilmiştir. 2. katman

çıkışları hesaplanırken her nöronun ağırlıkları ile 2. katman girişi arasındaki hamming mesafesi hesaplanmıştır. 2. katmandaki 1. nöronun ağırlıkları [+1,-1,-1] ile giriş vektörü aynı olduğu için hamming mesafesi 0 olarak hesaplanacaktır. Böylece 1. nöronun çıkışı 1, diğer nöronların çıkışları 0 olacaktır.

ACO_R-ÇKA'da her sınıf, her zaman tek bir bölge ile temsil edilemeyebilir. Bu durumda sınıf bilgisinin elde edilebilmesi için, aynı sınıftan örneklerin bulunduğu farklı bölgelerin birleştirilmesi gerekir. Bu birleştirme işlemi ağın 3. katmanında gerçekleştirilir. Bu katmanda, ağın 2. katman çıkışlarına mantıksal VEYA işlemi uygulanır. 3. katmanda sınıf sayısı (c) kadar nöron (n_3) bulunmaktadır. 2. katmandan gelen işaretler 1 veya 0 ile çarpılır ve nöronlarda VEYA işlemine tabi tutulur. 3. katmanda her defasında bir nöron 1 değerini verebilir, diğer nöronlar 0 değerini vermelidir.

Örnek olarak Şekil 6.6a'da sınıflandırılacak iki küme gösterilmektedir. □ ile gösterilenler 1. sınıfa, ♦ ile gösterilenler ise 2. sınıfa ait elemanları belirtmektedir. Ağın 1. katmanına 2 nöron eklenmesiyle, farklı sınıflar Şekil 6.6b'de gösterildiği gibi birbirinden ayrılmıştır. Öznitelik uzayı 4 farklı bölgeye bölünmüştür. Her bölgenin kodları şekilde gösterilmektedir.



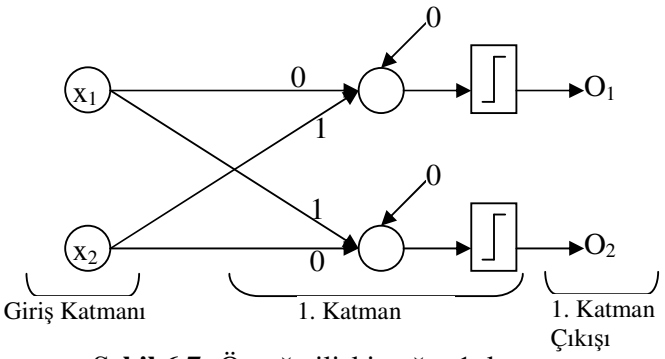
Şekil 6.6 a) Sınıflandırılacak kümeler **b)**1.katmanda oluşan nöronlar ve bölge kodları

1. ve 2. nöronlara ait çıkış ifadeleri (6.5a) ile W ağırlık matrisi (6.5b) ile verilmiştir.

$$O_1 = \begin{cases} +1 & 0 + 0x_1 + 1x_2 > 0 \\ -1 & 0 + 0x_1 + 1x_2 \le 0 \end{cases} \quad O_2 = \begin{cases} +1 & 0 + 1x_1 + 0x_2 > 0 \\ -1 & 0 + 1x_1 + 0x_2 \le 0 \end{cases} \qquad \textbf{(6.5a)}$$

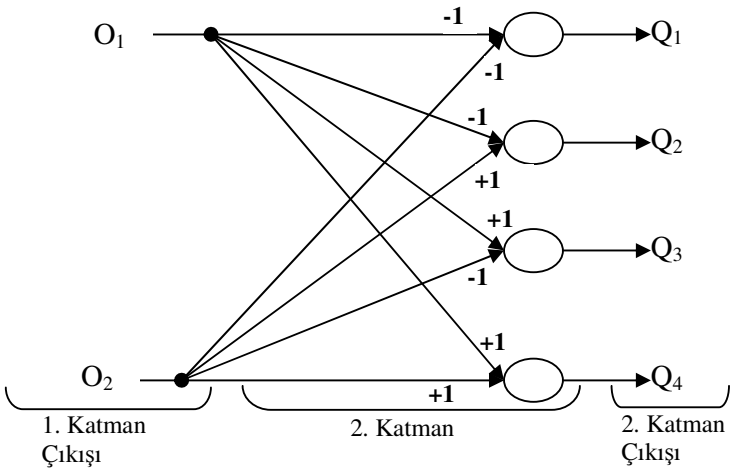
$$W = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \textbf{(6.5b)}$$

Bu durumda ağın 1 katmanı Şekil 6.7'da gösterildiği gibi oluşacaktır.



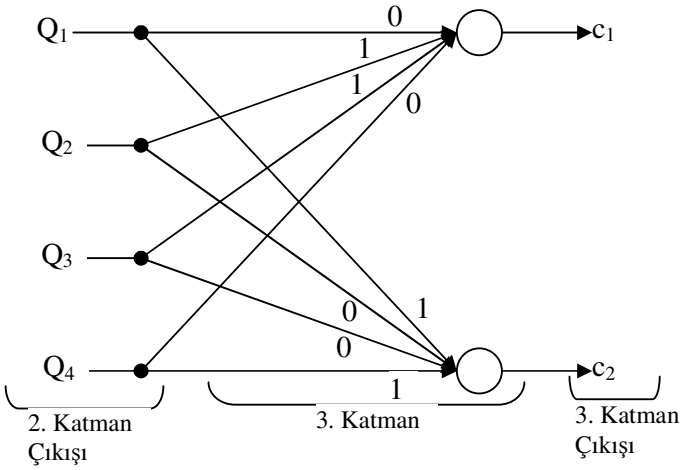
Şekil 6.7: Örneğe ilişkin ağın 1. katmanı

Şekil 6.6b de görüldüğü gibi öznitelik uzayı 4 farklı bölgeye ayrılmıştır. Bu bölgeler {(-1,-1); (-1,+1); (+1,-1); (+1,+1)} kodları ile temsil edilmektedir. 2. katman ağırlıklarında bu bölge kodları bulunmaktadır. Buna göre ağın 2. katmanı aşağıdaki gibi olacaktır.



Şekil 6.8: Örneğe ilişkin ağın 2. katmanı

Şekil 6.6b de görüldüğü gibi aynı sınıfa ait elemanlar farklı bölgelerde bulunuyor olabilir. Örneğin 1. sınıf elemanları (-1,+1) ve (+1,-1) kodları ile temsil edilen bölgelerde bulunmaktadır. 3. katmanda bu bölgelerin mantıksal VEYA işlemiyle birleştirilmesi gerçekleştirilecektir. 3. katmandaki ağırlıklar, 2. katmandaki bölgelerin temsil ettikleri sınıflara göre belirlenir. 3. katmanda, 2. katmandaki bir nöronun temsil ettiği sınıfa ait ağırlık 1, diğer sınıflara ait ağırlıklar 0 değerini alır. Buna göre ağın 3. katmanı aşağıdaki gibi olacaktır.



Şekil 6.9: Örneğe ilişkin ağın 3. katmanı

6.1.1.2 ACO_R-ÇKA'nın eğitimi

ACO_R-ÇKA'nın eğitimi 1. katmandaki ağırlıkların en iyi değerlerinin hesaplanıp, ağırlık matrisinin *(W)* bulunması demektir. Bulunan *W* matrisinin uygunluğunun ölçülebilmesi için bir uyum fonksiyonu (fitness function) belirlenmelidir. Belirlenen uyum fonksiyonu, farklı sınıftan elemanları farklı bölgelerde tutmayı ve bir bölgede aynı sınıftan en çok sayıda eleman saklamayı sağlamalıdır. Eğitim aşamasında uyum fonksiyonu istenen değere gelene kadar ağa yeni nöron eklenecektir. Ayrıca tüm sınıflar hiper düzlemlerle birbirinden ayrıldığı zaman uyum fonksiyonu 1, diğer durumlarda 1'den küçük değer almalıdır. Sınıflar birbirinden ayrıldıkça uyum fonksiyonunun değeri de artış göstermelidir.

Bu bölümde, yukarıdaki şartları sağlayan bir uyum fonksiyonu tanıtılacaktır.

- ACO_R-ÇKA'da, 1. katman çıkışında, toplam M tane içi boş olmayan bölge oluşmuş olsun: $\{C_1\},\{C_2\},\{C_3\}.....\{C_M\}$
- Toplam E adet eğitim kümesi elemanı olsun :

$$E = \#(\{C_1\} \cup \{C_2\} \cup \{C_3\} ..... \cup \{C_M\}) \quad \textbf{(6.6)}$$

- Toplam S adet sınıf olsun:

$$\{C_i\} = \bigcup_{k=1}^{S} \{C_{ik}\} = \{C_{i1}\} \cup \{C_{i2}\} \cup ..... \cup \{C_{iS}\} \quad \textbf{(6.7)}$$

- Her bölgenin temsil ettiği sınıfın (o bölgedeki) eleman sayısı MAX_i olsun. Temsil edilen sınıf, bölgede en çok sayıda eleman bulunduran sınıftır

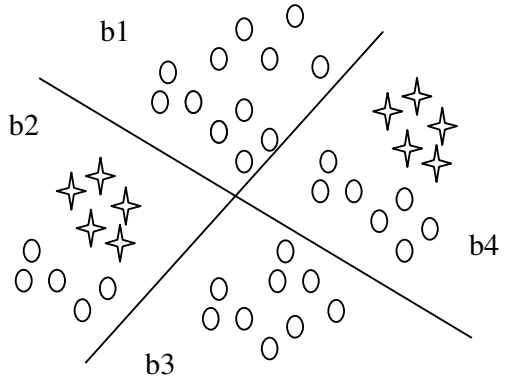
$$MAX_i = Max(\#\{C_{i1}\}, \#\{C_{i2}\}, ..., \#\{C_{iS}\}) \quad \textbf{(6.8)}$$

c_{ik}, i. bölgede, k. sınıfa ait elemanların kümesini göstermektedir.

- Uyumluluk fonksiyonu aşağıdaki gibi tanımlanmıştır:

$$UF = \sum_{i=1}^{M} \frac{MAX_i}{E} * \frac{MAX_i}{\sum_{k=1}^{S} C_{ik}} \quad \textbf{(6.9)}$$

Oluşturulan her nöron uzayı farklı bölgelere ayıracağı için, bu bölgeler ile sınıfların bu bölgelere dağılımı bölge tablosu ile gösterilir. Örneğin aşağıdaki bölgeleme için bölge tablosu ve uyumluluk fonksiyonu hesaplanmıştır.



Şekil 6.10: Örnek bir dağılım ve oluşan bölgeler

Çizelge 6.1 : Örnek dağılıma ilişkin bölge tablosu

Bölge no	Sınıf 1	Sınıf 2
b1	12	0
b2	5	5
b3	9	0
b4	7	5

Eğitim kümesinde toplam $E = 12 + 10 + 9 + 12 = 43$ adet örnek bulunmaktadır. Buna göre bu dağılımın uyumluluk fonksiyonu (UF), çizelge 6.1'den yararlanarak aşağıdaki gibi hesaplanır.

$$UF = \frac{12}{43} * \frac{12}{12} + \frac{5}{43} * \frac{5}{10} + \frac{9}{43} * \frac{9}{9} + \frac{7}{43} * \frac{7}{12} = 0{,}641 \quad \textbf{(6.10)}$$

Uyumluluk fonksiyonunun değeri, eğitim kümesi elemanlarının bölgelere ayrılması ile değişecektir. Bölgeleme ise birinci katmandaki nöronlar tarafından oluşturulan hiper düzlemler ile gerçekleşmektedir. Buna göre, her nöronun ağırlık vektörü (w_i), karşı düşen hiper düzlemin pozisyonunu ve yönünü değiştirecektir. Ağın eğitimi, yüksek değerde uyum fonksiyonuna sahip bir bölgeleme oluşturmak için birinci katmandaki nöron ağırlıklarının uygun biçimde değiştirilmesidir.

Bir nöron için (i. nöron) giriş vektörü boyutunun bir fazlası kadar (n_0+1) w_{ij} ağırlık değeri vardır. Birinci katmandaki nöron sayısı n1 ise, eğitim sırasında bulunması gereken toplam ağırlık değeri sayısı n1*(n0+1) olacaktır. Fakat eğitim işlemi, ağırlıkların tamamının tek seferde bulundurmasını gerektirmez. Artımsal olarak ağa nöronlar eklenerek öz nitelik uzayı bölgelere ayrıldıkça Uyum Fonksiyonu değeri artış gösterecektir. Eğitim sırasında 1. katmana yeterli sayıda nöron eklenerek ve eklenen yeni nöron için ağırlıklar değiştirilerek Uyum Fonksiyonunun istenilen değere gelmesi sağlanır. Uyum fonksiyonu istenilen değere erişince eğitim tamamlanmış ve sınıflama için uygun karar yüzeyleri oluşturulmuş olacaktır.

Her eklenen nöron için belirlenmesi gereken W_{ij} parametreleri ACO_R yöntemi ile belirlenmektedir. Belirli sayıda iterasyon boyunca ACO_R algoritması çalıştırılarak, eklenen nöron parametrelerinin Uyum Fonksiyonunu maksimize etmesi beklenir. İterasyon tamamlandığında, nöron ağırlıkları artık kalıcı olacaktır. Daha sonra eklenen nöronlar için bu nöronun oluşturduğu bölge sınırı sabit kalacaktır. Nöron ekleme işlemi, istenilen değerde Uyum Fonksiyonu elde edilene kadar devam eder.

Aşağıda, ACO_R ile eğitilen çok katmanlı ağın eğitim algoritması verilmiştir.

Çizelge 6.2: ACO_R ile eğitilmiş çok katmanlı ağın eğitim algoritması

while Uygunluk değeri < İstenen uygunluk değeri **do**
- Ağın birinci katmanına bir nöron ekle
- ACO_R için rasgele çözümlerin saklandığı çizelgeyi oluştur
- İterasyon sayısını belirle

while iterasyon sayısı >0 **do**

For Çizelgedeki her satır elemanı için
- Önceden belirlenen nöronlar (sabitleşmiş nöronlar) ile birlikte aday nöron için eğitim kümesindeki her elemanın bulunduğu bölgeyi belirle
- Boş olmayan bölgeler için bölge matrisini oluştur
- Bölge matrisinden yararlanarak uygunluk değerini hesapla (Denklem 6.9)

End for
- Çizelgeyi uygunluk değerine göre büyükten küçüğe sırala
- Çizelgeden yararlanarak ACO_R'ye ait ağırlık değerlerini hesapla (Denklem 4.17)
- Olasılık değerine göre t* adet çözüm seç
- Seçilen çözümün σ değerlerini hesapla (Denklem 4.19)
- σ değerinden yaralanarak seçilen çözümlerden yeni çözümler üret
- Oluşan yeni çözümleri, tablonun en altındaki çözümler ile değiştir
- İterasyon sayısını bir azalt

End while
- En yüksek uyumluluk değerini veren aday nöronu yeni nöron olarak ağa kabul et, uygunluk değerini bu nöronun sağladığı uygunluk değeri olarak belirle

For Yeni eklenen nöron haricindeki sabitleşmiş diğer nöronlar için
- Ağdan bu nöronu çıkart ve uyumluluk değerindeki değişime bak
- eğer değişim yoksa bu nöronu ağdan kalıcı olarak çıkart

End for

End While

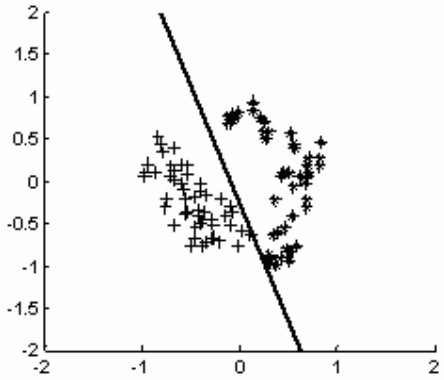
Birinci katmandaki ağırlıklar, Çizelge 6.2'de verildiği gibi bulunduktan sonra, ikinci ve üçüncü katmanlar kolayca bulunacaktır. Bu katmanlar aynı sınıfı temsil eden farklı bölgelerin birleştirilmesi görevini yerine getirecektir. Birinci katman ağırlıkları dışındaki ağırlıkların eniyileme yöntemleri ile araştırılmasına gerek yoktur. Birinci katman sonucu oluşan bölgeleme sonucundan yararlanılarak her sınıfın hangi bölgeler tarafından temsil dildiği bulunur ve karşı gelen bölgeler VEYA işlemi ile birleştirilir.

Önerilen ağ ayrıca, unutma algoritması da sunmaktadır. Bazı durumlarda, yeni bir nöron eklendiğinde, önceki nöronlardan biri ya da birkaçının, bölgeleme başarımı üzerinde etkisi kalmamış olabilir. Bu durumda, bu nöron ağdan atılabilir. Bu amaçla, her yeni nöron eklendikten sonra, varsa ağdaki gereksiz nöronlar kalıcı olarak atılır.

6.1.1.3 ACO_R-ÇKA ile elde edilen örnek sonuçlar

Bu kısımda, iki boyutta örnek dağılımlar, oluşan karar doğruları ile birlikte verilmiş ve elde edilen bölge matrisleri gösterilmiştir. Oluşan karar doğrusu sayısı 1. katmandaki nöron sayısını verir. Farklı her sınıf, farklı semboller ile (+,*, x…) belirtilmiştir. İstenen uyum fonksiyonu değeri 0.95 olarak belirlenmiştir.

Aşağıda örnek dağılım 1 nöron ile ayrıştırılmıştır



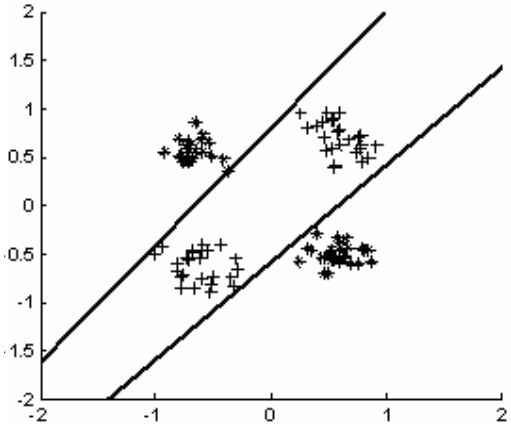
Şekil 6.11: Örnek dağılım ve ayrıştırılması–1

Yukarıdaki örnek için elde edilen bölge matrisi ise aşağıdaki gibidir.

Çizelge 6.3: Örnek bölge matrisi-1

Bölge\Sınıf	1	2
1	50	0
0	0	50

Aşağıda örnek dağılım 2 nöron ile ayrıştırılmıştır



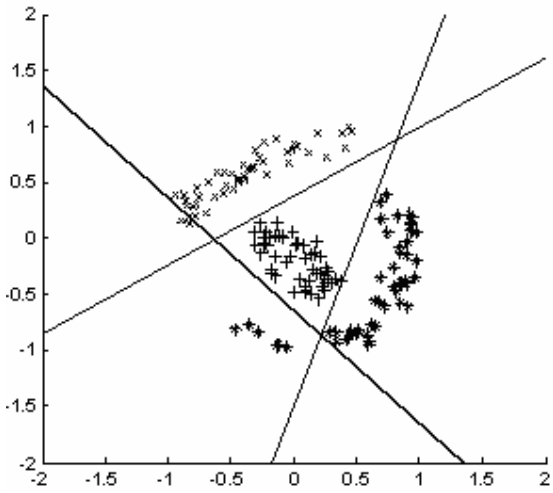
Şekil 6.12: Örnek dağılım ve ayrıştırılması–2

Yukarıdaki örnek için elde edilen bölge matrisi ise aşağıdaki gibidir.

Çizelge 6.4: Örnek bölge matrisi-2

Bölge\Sınıf	1	2
3	50	0
2	0	22
1	0	28

Aşağıda örnek dağılım 3 nöron ile ayrıştırılmıştır



Şekil 6.13: Örnek dağılım ve ayrıştırılması–3

Yukarıdaki örnek için elde edilen bölge matrisi ise aşağıdaki gibidir.

Çizelge 6.5: Örnek bölge matrisi- 3

Bölge\Sınıf	1	2	3
4	50	0	0
6	0	44	0
5	0	6	0
0	0	0	47
1	0	0	3

6.1.2 Genetik çok katmanlı yapay sinir ağı (Get-ÇKA)

Tez kapsamında sınıflandırıcı olarak kullanılan diğer bir yapay sinir ağı Genetik Çok Katmanlı Ağ (Get-ÇKA)dır. Get-ÇKA'nın eğitiminde amaç, ağın birinci katmanındaki düğümlere ait ağırlıkların en iyi değerlerinin hesaplanmasıdır. ACO_R-ÇKA'da ağırlıkların hesaplanmasında KKO kullanılırken, Get-ÇKA da genetik algoritmalardan yararlanılır.

Genetik havuzdaki diziler ağın tüm parametrelerini temsil etmez. Genetik algoritmalar, birinci katmandaki düğümlerin ağırlık değerlerini bulmaktadır, diğer

katmandaki düğümlerin ağırlıklarını bulmak için kullanılmamaktadır. Havuz içinde uyumluluk fonksiyonunu en iyi sağlayacak tek bir hiper düzlem araştırılır

Ağın birinci katmanındaki düğümler ihtiyaca göre belirlenmektedir. Böylece çok katmanlı ağın düğüm sayısı eğitim sırasında belirlenmektedir. Birinci katmandaki düğümler, giriş uzayındaki bölgeleri kodlamaktadır. Eğitim sonunda, bir bölge içine düşen eğitim kümesi vektörlerinin hepsi aynı sınıftan olmalıdır. Aksi halde, eğitim sonrası bazı giriş değerleri için doğru sınıflama sonuçları üretilemeyecektir. Dolayısıyla uyumluluk fonksiyonu, eğitim sırasında seçilen hiper düzlemler ile oluşan hacimsel bölgelerin içinde aynı sınıftan vektörlerin bulunmasını temin edecek şekilde tanımlanır. Amaç, yeni bir hiper düzlem ilavesi ile oluşan bölgelerde en çok sayıda aynı sınıftan vektör elde etmektir. Amaç ölçütü için uyumluluk fonksiyonunun (UF) matematiksel ifadesi aşağıda verilmiştir.

$$\{\mathrm{E}\}=\bigcup_{i=1}^{M}\{C_i\} \tag{6.11a}$$

$$\{C_i\}=\bigcup_{k=1}^{S}\{C_{ik}\}=\{C_{i1}\}\cup\{C_{i2}\}\cup.....\cup\{C_{iS}\} \tag{6.11b}$$

$$MAX_i = Max(\#\{C_{i1}\},\#\{C_{i2}\},.....,\#\{C_{iS}\}) \tag{6.11c}$$

$$UF=\sum_{i=1}^{M}\frac{\mathrm{MAX}_i}{\#\{\mathrm{E}\}}*\frac{MAX_i}{\#\{C_i\}} \tag{6.11d}$$

Denklemlerde görülen {E}; eğitim kümesini, $\{c_{ik}\}$; *i*-inci bölge içinde *k*-ıncı sınıfa ait vektörlerin kümesini, *M*; oluşan toplam bölge sayısını ve *S*; sınıf sayısını göstermektedir. ACO_R-ÇKA ile Get-ÇKA ağında aynı uyum fonksiyonu kullanılmış, böylelikle benzer şartlar altında algoritmaların başarımının ölçülmesi sağlanmıştır.

Başlangıçta havuz, birinci katmandaki ilk düğümün ağırlıklarını temsil eden dizilerden oluşturulur. Her nesilde havuz içindeki tüm dizilere dört genetik işlem uygulanır. Nesil sayısı sonunda en yüksek uyumluluk değerinin 0.95'in üstünde olup olmadığı incelenir. Bu başarım sağlanmışsa, en iyi uyumluluk değeri veren düzlem ağın ilk düğümü olarak havuzdan alınır ve ağın sabitleşmiş düğümü olarak atanır. Gelecek nesillerde bu düğümün ağırlıkları araştırılmaz, ancak uyumluluk değeri hesabında kullanılır. İkinci düğümü temsil etmek üzere rasgele değerler ile havuz yeniden oluşturulur. İkinci düğümü bulmak için uyumluluk değeri hesabında birinci

sabit düğüm ile ikinci aday düğümlerin temsil ettikleri hiper düzlemler tek tek kesiştirilir. Kesişim sonucu oluşan bölgeler için uyumluluk değeri hesaplanır.

GetÇKA'nın eğitiminde kullanılan adımlar aşağıda verilmiştir.

Adım 1: Rastgele değerler ile genetik havuzu oluştur. Nesil sayısını belirle.

Adım 2: Önceden bulunan ağ hiper düzlemleri (sabitleşmiş düğümler) ile havuzdaki dizilerin temsil ettiği hiper düzlemleri tek tek kesiştir ve her bir dizi için uyumluluk değeri hesapla.

Adım 3: Havuz içindeki tüm dizilere kopyalama, çaprazlama ve mutasyon işlemlerini uygula. Nesil sayacını 1 azalt. Nesil sayacı sıfıra eşit değilse adım 2'ye git.

Adım 4: Nesil sayacını başlangıç değeri ile kur. Havuz içinde en yüksek uyumluluk değerini veren diziyi bul.

Adım 5: Uyumluluk değeri 0.95 değerinden büyük ise eğitim algoritmasını sonlandır.

Adım 6: En iyi uyumluluk değerini veren diziyi, ağın sabitleşmiş bir düğümünü oluşturmak üzere havuzdan al. Adım 1'e git.

6.1.3 Geriye yayılım algoritması ile eğitilmiş çok katmanlı yapay sinir ağı

Tez kapsamında sınıflandırıcı olarak geriye yayılım algoritmasıyla eğitilmiş çok katmanlı yapay sinir ağı da kullanılmıştır. ÇKA'ya ait ağırlıkların en iyi değeri geri yayılım (backpropogation) algoritması kullanılarak hesaplanmaya çalışılır.

ÇKA'nın eğitimine başlanmadan önce ağın yapısının ve başlangıç parametrelerinin kullanıcı tarafından belirlenmesi gerekmektedir. Ağın yapısı belirlenirken şu parametreler belirlenir:

- Giriş nöron sayısı
- Çıkış nöron sayısı
- Gizli katman sayısı
- Gizli katmandaki nöron sayıları

Tez kapsamında kullanılan ÇKA için bu değerler belirlenirken, farklı katman sayıları ve nöron sayıları ile çeşitli denemeler yapılmıştır. Denemeler sonucunda katman ve nöron sayısı parametreleri aşağıdaki gibi seçilmiştir.

- Giriş nöron sayısı = öznitelik vektörü boyutu kadar
- Çıkış nöron sayısı = 14
- Gizli katman sayısı =1
- Gizli katmandaki nöron sayıları =14

ÇKA'nın eğitimine başlamadan önce belirlenmesi gereken diğer başlangıç parametresi öğrenme katsayısı (η) ise 0,2 olarak seçilmiştir.

Çok Katmanlı ağın eğitim algoritmasını şu şekilde özetlenebilir.

Adım 1: Kabul edilebilir hata oranını belirle (ε)

Adım 2: Tüm ağırlıklara [-1,+1] aralığında rasgele değer ata

Adım 3: Eğitim kümesinden rasgele örnek seç

Adım 4: Örnek eğitim vektörü için beklenen çıkış değerini ($\overline{T}$) belirle

Adım 5: Seçilen örnek vektör için ağın ürettiği çıkışı ($\overline{O}^{C}$) bul

(ileri doğru hesaplama- Denklem 2.10, 2.11, 2.12, 2.13)

Adım 6: Ağın çıkışı ($\overline{O}^{C}$) ile beklenen çıkış ($\overline{T}$) arasındaki farkı (hata) hesapla

(Denklem 2.15)

Adım 7: Hesaplanan hata (E), kabul edilebilir hatadan (ε) küçükse adım 10a git

Adım 8:Çıkış katmanından geriye doğru ağırlık değerlerini (w_{ij}) güncelle

ağırlık değeri çıkış katmanı ile ara katman arasında ise

(Denklem 2.19-2.20)

ağırlık değeri ara katmanlar veya ara katman ile giriş katmanı arasında ise

(Denklem 2.23-2.25)

Adım 9: Adım 3 e git

Adım 10: Eğitimi sonlandır

6.1.4 K- En yakın komşu (k-nearest neighbour) sınıflandırıcısı

Eğitim kümesindeki $\mathbf{x_i}$ örüntüleri, $\mathbf{y_j}$ sınıfları ile doğru olarak ilişkilendirilmiş örneklerden oluşur. Sezgisel olarak, aynı **j** sınıfı ile ilişkili olan **x** örüntülerinin birbirlerine benzer (*yakın*) oldukları söylenebilir. Aynı sınıf örüntüleri genellikle bir bölgede kümeleşirler. Bu durum, sınıflandırılmamış **x** örüntülerini en yakın komşusu olan **j** sınıfına atamanın olumlu sonuçlar üreteceğine işaret eder. Bu sınıflandırma kuralı en yakın komşu yaklaşımı olarak adlandırılır ve matematiksel olarak şu şekilde ifade edilir. Aşağıdaki eşitliği sağlayan, x'_n, x'in en yakın komşusudur.

$$\min d(x_i,x) = d(x'_n,x) \qquad \textbf{(6.12)}$$

En yakın komşu yaklaşımı x örüntüsünün sınıfını, x örüntüsüne en yakın olan örüntünün sınıfı olarak belirleme yaklaşımıdır. Bu yaklaşımda, sadece tek uzaklıktaki komşuluk yerine, bir örneğin k tane komşusuna bakılarak sınıfının belirlenmesi k- en yakın komşu kuralı olarak adlandırılır [50].

K-en yakın komşu algoritması, istatistiksel teoriye dayanan bir sınıflandırma yöntemidir. Tanıma, tanınacak öznitelik vektörüne en yakın öznitelik vektörlerinin bulunmasına bağlıdır. Tanıma yapılacak öznitelik vektörüne en yakın k komşu bulunur. Daha sonra bu k komşu en fazla hangi sınıfa ait ise, o sınıf tanıma sonucu olarak atanır [45].

k-en yakın komşu algoritması, eğitim ve test kümesindeki tüm örneklerin n boyutlu bir öznitelik uzayında bir noktaya karşılık geldiğini kabul eder. Bir örneğin en yakın komşuları bulunurken çeşitli metrikler kullanılabilir. Bunlar Euclidean ölçeği, Hamming ölçeği, nokta mesafesi ölçeği, kosinüs benzerliği olabilir.

Euclidean ölçeği seçildiğinde, iki örnek arasındaki benzerlik şu şekilde bulunur.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2} \qquad \textbf{(6.13)}$$

Burada x_i, x_j test kümesinden alınan iki örneği temsil eder. Her bir x örneği, öznitelik vektörü ile tanımlanır.

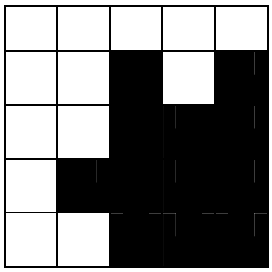
$$x_i=[a_1(x_i), a_2(x_i),\ldots\ldots, a_n(x_i)] \qquad \textbf{(6.14)}$$

$a_r(x_i)$, x_i örneğinin r. özniteliğinin değeridir [51].

Sınıflandırma Hamming ölçeği kullanılarak yapılırsa, seçilen iki örneğin farklılığı, karşılıklı piksel değerlerine mantıksal YADA işlemi uygulanarak hesaplanır [52].

$$d(x_i, x_j) = \sum_{k=1}^{M} x_{ik} \oplus x_{jk} \qquad \text{M:toplam piksel sayısı} \qquad \textbf{(6.15)}$$

Piksel mesafesi ölçeği kullanılarak iki örneğin farklılığı hesaplanırken, her pikselin, örüntüdeki diğer bölgeye (arka plan veya nesne) ait en yakın piksele olan uzaklığının tutulduğu uzaklık matrisinden yararlanılır.



B_i

3	2	1	2	1
2	1	1	1	1
2	1	1	1	2
1	1	2	2	3
2	1	1	2	3

$D(B_i)$

Şekil 6.14: Piksel Matrisi (B_i)ve Piksel Uzaklık Matrisi ($D(B_i)$)

Buna göre uzaklık şu şekilde hesaplanır [51].

$$d(B_i, B_j) = \sum (B_i \oplus B_j)\left[D(B_i) + D(B_j)\right] \qquad \textbf{(6.16)}$$

k-en yakın komşu yaklaşımında, k değerinin seçimi, sınıflandırmanın sonucunu büyük oranda etkiler. Farklı k değerleri ile farklı doğrulukta sonuçlar elde edilir [51]. K değerinin büyük seçilmesi hatalı sınıflandırma oranını azaltırken, küçük seçilmesi daha yakın komşuların sınıflandırmadaki etkisinin artırır [50].

6.2. Kullanılan Öznitelikler

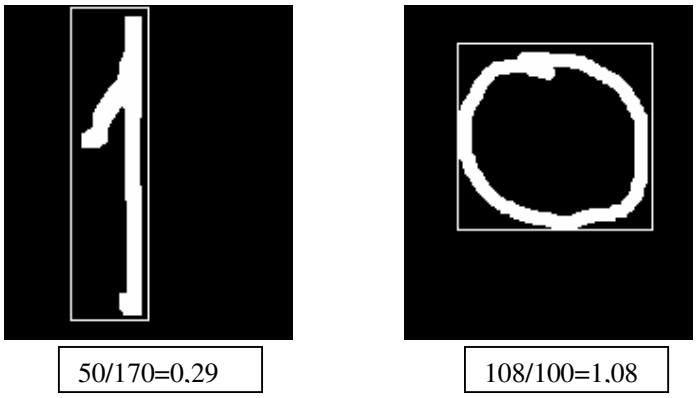
Tez kapsamında öznitelik çıkarmak amacıyla iki yöntem kullanılmıştır. İlkinde giriş verilerinin şekilsel özelliklerinden yararlanılarak öznitelik vektörü oluşturulmuştur. İkinci yöntemde ise Karhunen-Loeve Dönüşümünden yararlanılmıştır.

6.2.1 Şekilsel öznitelikler

Tanıma sistemine sunulan her örüntü için aşağıda ayrıntıları verilen şekilsel özelliklerin sayısal değerleri kullanılarak bir öznitelik vektörü oluşturulmuştur.

6.2.1.1 En boy oranı

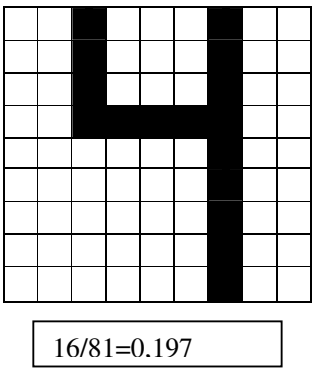
Şekilsel öznitelik vektörünün ilk elemanı giriş örüntüsünde yer alan karakterin dış hatlarını çevreleyen dikdörtgenin en boy oranıdır. Bu öznitelik, özellikle '1' karakteri için, diğer rakamlara oranla daha küçük değerler alacaktır. Dolayısıyla '1' karakteri için ayırt edici olabilir.



Şekil 6.15: 1 ve 0 karakterlerinin en-boy oranı değeri

6.2.1.2 Örüntü beyaz piksel oranı

Örüntünün tamamındaki beyaz piksel sayısının, tüm piksel sayısına oranı hesaplanarak bir diğer şekilsel öznitelik elde edilmiştir.

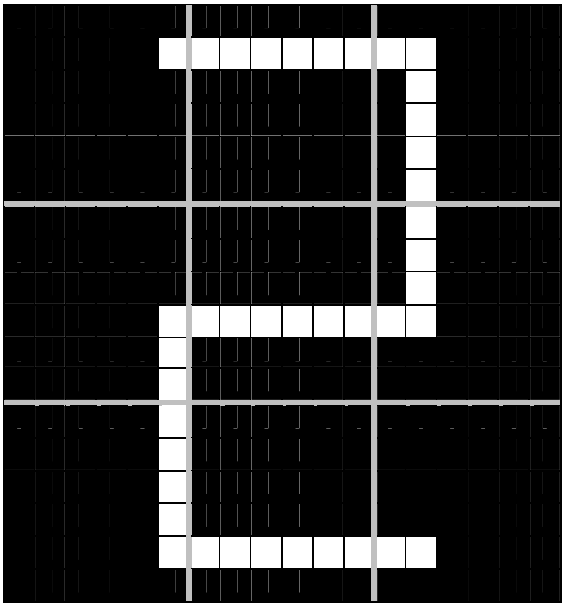


Şekil 6.16: Örüntü beyaz piksel oranını özniteliğinin çıkarımı

Örüntü beyaz piksel oranı şekilsel öznitelik vektörünün 11. elemanını oluşturur.

6.2.1.3 Bölgesel beyaz piksel oranı

Öznitelik vektöründe kullanılacak diğer bir şekilsel öznitelik, örüntünün 9 bölgeye ayrılması ve her bölgedeki beyaz piksel sayısının, örüntünün tamamındaki beyaz piksel sayısına oranı bulunarak elde edilmiştir.



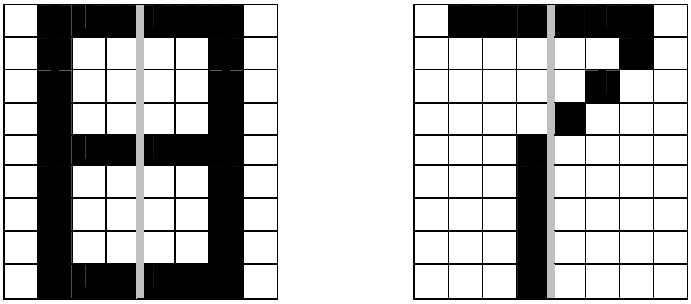
1. Bölge →1/40=0,025	2. Bölge→6/40=0,15	3. Bölge→6/40=0,15
4. Bölge→3/40=0,075	5. Bölge→6/40=0,15	6. Bölge→5/40=0,125
7. Bölge→5/40=0,125	8. Bölge→6/40=0,15	9. Bölge→2/40=0,05

Şekil 6.17: Bölgesel beyaz piksel oranı özniteliğinin çıkarımı

Bu şekilde elde edilen 9 değer, şekilsel öznitelik vektörünün 2-10'uncu elemanlarını oluşturur.

6.2.1.4 Dikey simetri değeri

Dikey simetri değeri, örüntü alanı dikey olarak iki eşit parçaya bölündüğünde, iki bölgedeki karşılıklı piksel değerlerinin benzerliğini gösterir. Dikey simetri örneğin '8' karakterinde yüksek bir değer alırken, '7' gibi bir karakterde küçük değer alır. Bu gibi karakterlerin ayrımında dikey simetri değeri yararlı olur.



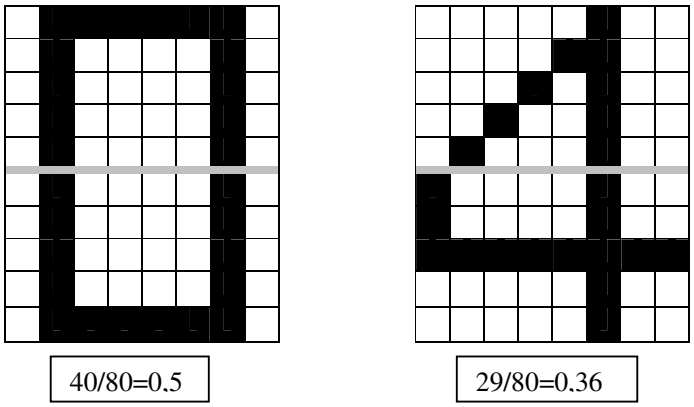
36/72=0,5	28/72=0,38

Şekil 6.18: Dikey simetri değeri özniteliğinin çıkarımı

Örüntünün iki dikey bölgesindeki aynı değerdeki piksel sayısının, tüm piksel sayısına oranı ile elde edilen dikey simetri değeri şekilsel öznitelik vektörünün 12. elemanını oluşturur.

6.2.1.5 Yatay simetri değeri

Yatay simetri değeri, örüntü alanı yatay olarak iki eşit parçaya bölündüğünde, iki bölgedeki karşılıklı piksel değerlerinin benzerliğini gösterir. Yatay simetri örneğin '0' karakterinde yüksek bir değer alırken, '4' gibi bir karakterde küçük değer alır. Bu gibi karakterlerin ayrımında dikey simetri değeri yararlı olur.



Şekil 6.19: Yatay simetri değeri özniteliğinin çıkarımı

Örüntünün iki yatay bölgesindeki aynı değerdeki piksel sayısının, tüm piksel sayısına oranı ile elde edilen yatay simetri değeri şekilsel öznitelik vektörünün 13. elemanını oluşturur.

6.2.2 Karhunen – Loeve dönüşümü (KLD)

KLD, verilen bir dağılımda örnek işaretleri en iyi temsil eden öznitelikleri ortaya çıkarmak için kullanılır. Böylece üzerinde çalışılan öznitelik uzayının boyutu azaltılır. Bu yolla sistemin tanıma aşamasında yapacağı hesaplamalar basitleştirilir ve başarım artışı sağlanır.

Karakter tanıma sistemine sunulacak eğitim kümesindeki görüntülere KLD uygulanırken, öncelikle görüntü matrisi –G- elde edilir. G matrisinin her bir satırı eğitim kümesindeki bir görüntünün piksel değerlerini saklar. Bu matris oluşturulurken, öncelikle (m*n) boyutundaki her bir eğitim görüntüsü (1*mn) boyutunda vektöre dönüştürülür. Elde edilen görüntü vektörleri, 1 satır halinde G matrisine eklenir.

G matrisi elde edildikten sonra, 'ortak değişinti matrisi' (Covariance Matrix) – CovG-nin elde edilmesi gerekir. G matrisinden, CovG matrisine geçişte, önce G matrisinin ortalama değeri hesaplanır ve G matrisinden çıkarılır. Elde edilen fark G matrisinin tersyüzü (transpose) kendisi ile çarpılarak CovG matrisi elde edilir.

G= G- ortalama(G) **(6.17a)**

CovG= G' * G **(6.17b)**

KLD uygulamasının devamında CovG matrisine ait özvektörlerin (eigen vector) bulunması gerekir. CovG kare matrisine ait özdeğer (eigen value)ve özvektörlerin aşağıdaki eşitliği sağlaması gerekir.

$$CovGx = \lambda x \quad \textbf{(6.18)}$$

Bu eşitlikte x matrisin özvektörlerini, λ ise özdeğerlerini göstermektedir. Aşağıdaki eşitliği sağlayan ve sütunları CovG matrisinin özvektörlerini oluşturan bir T matrisi bulunur.

$$T^{-1}CovGT = \Lambda \quad \textbf{(6.20a)}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & .. & .. & 0 \\ 0 & \lambda_2 & .. & .. & 0 \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & .. & \lambda_n \end{bmatrix} \quad \textbf{(6.20b)}$$

T matrisi, en büyük t adet özdeğere (eigenvalue) ilişkin t adet özvektörden oluşmuştur.

G görüntü matrisi, T matrisinin tersyüzü (transpose) ile çarpılır ve KLD öznitelik vektörü elde edilmiş olur.

Tez çalışmasında kullanılan karakter veri kümelerinde bazı örnekler düşük boyutlu özelliklerden oluşurken, bazı örnekler doğrudan karakter görüntülerinden oluşmaktadır. Karakter görüntüleri, sınıflandırıcı olarak kullanılan ağlara sunulursa, giriş uzayının boyutu büyük olur ve sınıflandırıcıların başarımı düşer. Başarımın düşmesini önlemek için, bu veri kümeleride giriş uzayının boyutunun azaltılması gerekir. Bu amaçla öznitelik çıkarma yöntemlerine başvurulmuştur. Şekilsel öznitelik

çıkarma yöntemi ile boyutu 13'e indirgenmiş bir öznitelik vektörü elde edilir. Karhunen-Loeve Dönüşümünde ise, 16 adet özvektör tercih edildiği için, giriş uzayının boyutu 16 olmuştur. Üçüncü bir öznitelik vektörü ise, iki öznitelik vektörünün birleştirilmesi ile elde edilmiştir.

Kullanılan veri kümelerinin özellikleri ve üç öznitelik vektörünün başarıma yaptıkları katkı, tezin 7. bölümünde verilmiştir.

7. KULLANILAN VERİ KÜMELERİ VE DENEYSEL SONUÇLAR

7.1. Kullanılan Veri Kümeleri

Tez kapsamında oluşturulan sınıflandırıcılar, iki grup veri kümesi kullanılarak test edilmiş ve başarımları ölçülmüştür.

7.1.1 I.Grup veri kümeleri

Sınıflandırıcılar, önce, sınıflamanın daha kolay yapılabildiği, bilinen bazı veri kümeleri üzerinde çalıştırılmıştır. Kullanılan veri kümeleri;

- Iris veri kümesi
- Tic – Tac – Toe veri kümesi

7.1.1.1 Iris veri kümesi

Iris veri kümesi, üç çeşit iris bitkisinin sınıflandırılması için kullanılır. Bitkiye ait 3 sınıfın her birinden 50 adet olmak üzere 150 adet örnek veriden oluşur. Ayırt edilecek 3 sınıf, Iris Setosa, Iris Versicolour ve Iris Virginica sınıflarıdır. Her bir örnek için 4 özellik ölçülmüş ve öznitelik olarak kullanılmıştır. Öznitelik olarak alınan 4 ölçüm şunlardır.

1. Sepal uzunluğu (cm)
2. Sepal genişliği (cm)
3. Petal uzunluğu (cm)
4. Petal genişliği (cm)

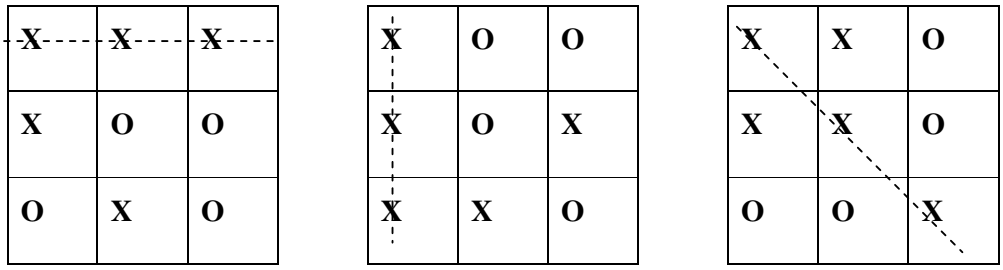
Iris veri kümesinde örnek girişler Çizelge 7.1 ile gösterilmiştir.

Çizelge 7.1: Örnek IRIS veri kümesi girişleri

Sepal Uzunluk	Sepal Genişlik	Petal Uzunluk	Petal Genişlik	Sınıf
5.1	3.5	1.4	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
5.8	2.7	5.1	1.9	Virginica

7.1.1.2 Tic Tac Toe veri kümesi

Tic-Tac-Toe veri kümesi, tic-tac-toe oyun panosunun muhtemel bitiş görüntülerini sınıflandırır. Oyun panosu 9 bölmeden oluşur, her bölme ya boştur, ya da X ve O işaretlerinden biri ile doludur. Yatay, dikey veya köşegende 3 'X' işareti yan yana ise X oyuncusu kazanmış olur. Örnek oyun panoları şekil 7.1'de verilmiştir.



Şekil 7.1: Örnek Tic-Tac-Toe oyun panosu

Tic-Tac-Toe veri kümesinde sınıflandırma, X oyuncusunun kazanma (positive) ve kazanamama (negative) durumuna göre yapılır.

Bu durumda veri kümesi için 2 sınıf ve 9 giriş değeri bulunmaktadır. Giriş değerleri x, o ya da b (boş) değerlerinden birini alabilir.

1. Üst sol bölme: {x,o,b}
2. Üst orta bölme: {x,o,b}
3. Üst sağ bölme: {x,o,b}
4. Orta sol bölme: {x,o,b}
5. Orta orta bölme: {x,o,b}
6. Orta sağ bölme: {x,o,b}
7. Alt sol bölme: {x,o,b}

8. Alt orta bölme: {x,o,b}

9. Alt sağ bölme: {x,o,b}

Tic-Tac-toe veri kümesinde örnek girişler şu şekildedi

Çizelge 7.2: Örnek Tic-Tac-Toe veri kümesi girişleri

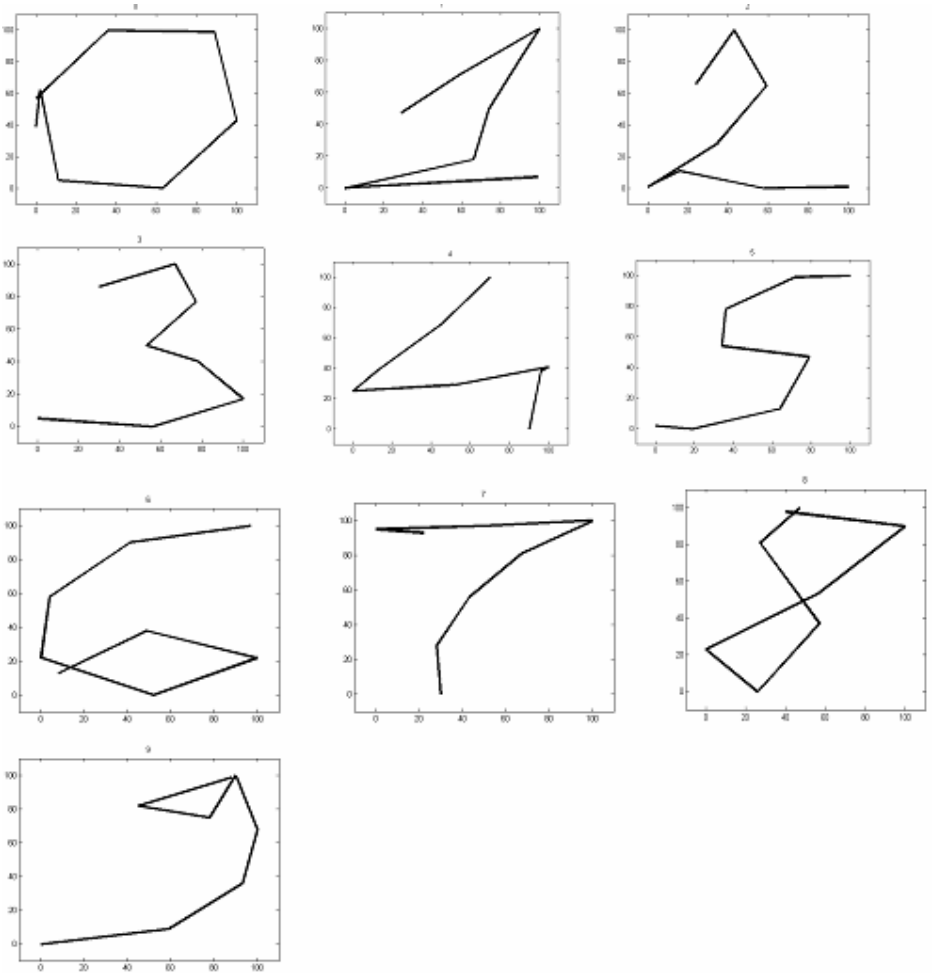
1. Blm	2. Blm	3. Blm	4. Blm	5. Blm	6. Blm	7. Blm	8. Blm	9. Blm	Sınıf
X	X	X	X	O	O	X	O	O	positive
X	X	X	O	X	B	O	O	B	positive
O	O	O	B	X	X	X	X	B	negative

7.1.2 II. Grup veri kümeleri

2. Grup veri kümeleri, çeşitli özelliklerdeki el yazısı rakamları içeren veri kümeleridir.

7.1.2.1 PenDigit veri kümesi (pen-based recognition of handwritten digits dataset)

PenDigit veri kümesi, 44 ayrı kişiden toplanan 250 rakam örneğinin bir araya getirilmesiyle oluşturulmuş bir veri kümesidir. Örneklerin yazımı sırasında WACOM PL-100V tablet ve kablosuz tablet kalemi kullanılmıştır. Her yazıcı tableti kullanarak rasgele sırada 250 rakam yazmıştır. Tabletler her 100ms'de yapılan vuruşların x ve y koordinatlarını saklar. Karaktere ait noktalar bu şekilde örneklenmiş olur. Uzay boyutunun azalması için karakteri oluşturan noktalar, belirli bir yay uzunluğuna göre yeniden örneklenmiş ve her bir karakter için alınan 8 nokta öznitelik olarak seçilmiştir. Bu noktalara karşılık gelen x ve y koordinatlarıyla 16 giriş değeri elde edilmiştir. Şekil 7.2'de bazı giriş değerleri kullanılarak yeniden oluşturulmuş rakamlar gösterilmiştir.



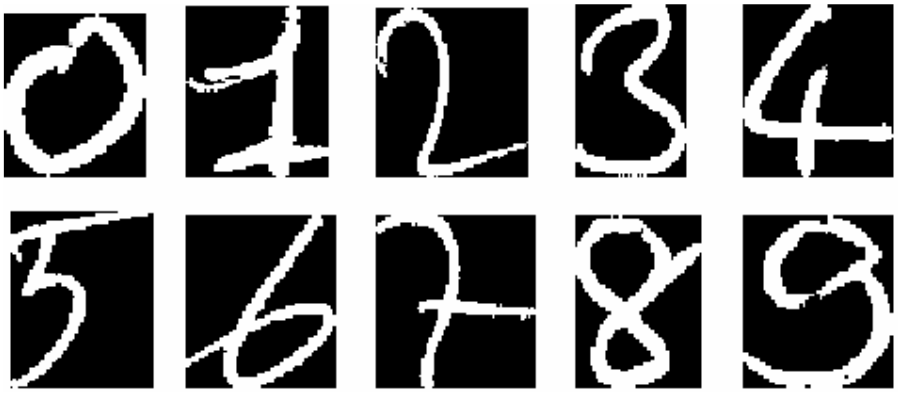
Şekil 7.2: PenDigit veri kümesinden örnekler

7.1.2.2 OpticalDigit veri kümesi

OpticalDigit veri kümesinde 32x32 boyutundaki karakterler 4x4 boyutundaki bölümlere ayrılmıştır. Her bölmedeki beyaz piksellerin sayısı alınarak 64 adet giriş elde edilmiştir. Bu girişler 0-16 aralığında değer alabilirler.

7.1.2.3 İ.T.Ü Türkçe karakter veri kümesi

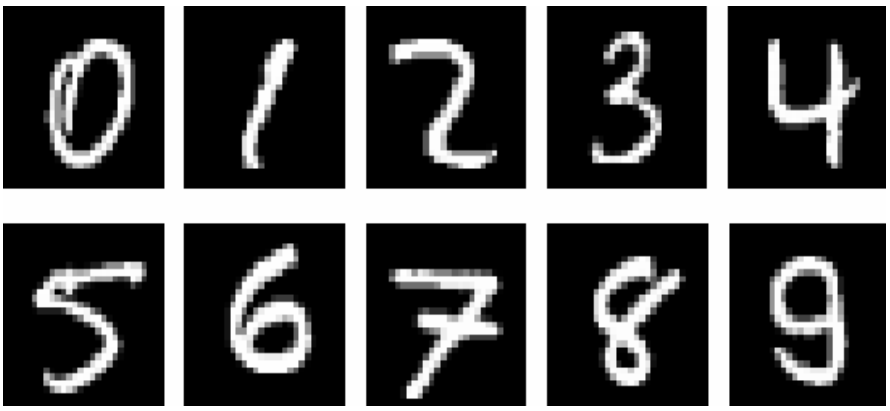
İTÜ Türkçe karakter veri kümesi 0-9 rakamlarını ve büyük Türkçe harflerin ikili görüntülerini içerir. Görüntüler karakter çerçevesi şeklinde olduğu için farklı boyutlardadır. Yöntem bölümünde anlatılan öznitelik çıkarma yöntemleri ile bu görüntülerin öznitelikleri alınmış ve giriş olarak kullanılmıştır. İTÜ veri kümesinden alınan rakam görüntülerin örnekleri aşağıda verilmiştir.



Şekil 7.3: İTÜ rakam veri kümesinden örnekler

7.1.2.4 NIST rakam veri kümesi

NIST Rakam veri kümesi 0-9 rakamlarının ikili görüntülerini içerir. Görüntüler karakter çerçevesi şeklinde olduğu için farklı boyutlardadır. Yöntem bölümünde anlatılan öznitelik çıkarma yöntemleri ile bu görüntülerin öznitelikleri alınmış ve giriş olarak kullanılmıştır. NIST veri kümesinden alınan rakam görüntülerin örnekleri aşağıda verilmiştir.



Şekil 7.4: NIST rakam veri kümesinden örnekler

7.2. Deneysel Sonuçlar

Bu bölümde, kullanılan sınıflandırıcıların, her bir veri kümesi üzerindeki sınıflandırma başarımları çizelgeler ile verilmiştir. Bu başarım çizelgelerinde satırlar, veri kümesindeki sınıfları gösterir. Satırlar ise, sınıflandırıcının ilgili sınıftan örnekleri hangi sınıflara yerleştirdiği görülebilir. Son satır ise yüzdesel başarımı göstermektedir. Örneğin aşağıdaki çizelgede, X sınıfına ait örneklerin 10 tanesi X sınıfı, 5 tanesi Y sınıfı, Y sınıfına ait örneklerin 3 tanesi X sınıfı, 12 tanesi Y sınıfı olarak sınıflandırılmıştır.

Çizelge 7.3: Örnek sınıflandırıcı başarım çizelgesi

	X	Y
X	10	5
Y	3	12

7.2.1 IRIS veri kümesi ile elde edilen sonuçlar

Bölüm 7.1.1.1 ile özellikleri verilen IRIS veri kümesi üzerinde 3-en yakın komşu, 5-en yakın komşu, BP-ÇKA, Get-ÇKA ve ACO_R-ÇKA sınıflandırıcıları test edilmiştir. IRIS veri kümesinde 3 sınıf ve her sınıfta 50 örnek bulunmaktadır. Testlerde her sınıftan rasgele seçilen 10 elemanla eğitim kümesi, geriye kalan 40 elemanla test kümesi oluşturulmuştur.

3-NN sınıflandırıcısı elde edilen sonuçları Çizelge 7.4 ile verilmiştir.

Çizelge 7.4: 3-NN sınıflandırıcısının IRIS veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2	Sınıf 3
Sınıf 1	40	0	0
Sınıf 2	0	37	3
Sınıf 3	0	1	39
% Başarım		%96	

5-NN sınıflandırıcısı elde edilen sonuçları Çizelge 7.5 ile verilmiştir.

Çizelge 7.5: 5-NN sınıflandırıcısının IRIS veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2	Sınıf 3
Sınıf 1	40	0	0
Sınıf 2	0	38	2
Sınıf 3	0	3	37
% Başarım		%95	

BP-ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.6 ile verilmiştir.

Çizelge 7.6: BP-ÇKA sınıflandırıcısının IRIS veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2	Sınıf 3
Sınıf 1	40	0	0
Sınıf 2	0	35	5
Sınıf 3	0	0	40
%Başarım		%95	

Get-ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.7 ile verilmiştir.

Çizelge 7.7: Get-ÇKA sınıflandırıcısının IRIS veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2	Sınıf 3
Sınıf 1	40	0	0
Sınıf 2	0	35	5
Sınıf 3	0	2	38
%Başarım		%94	

ACO_R-ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.8 ile verilmiştir.

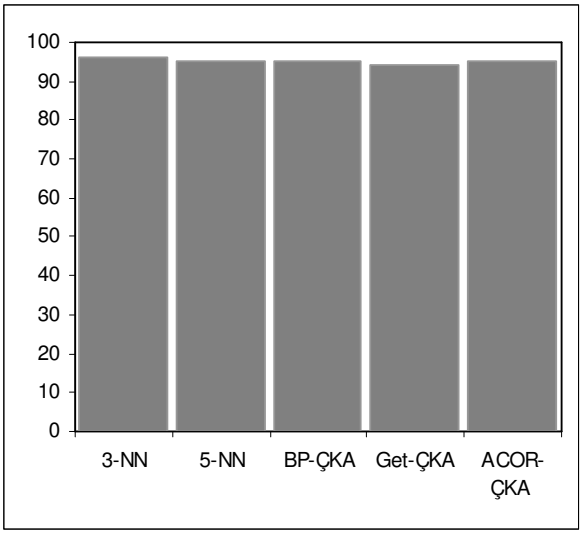
Çizelge 7.8: ACO_R -ÇKA sınıflandırıcısının IRIS veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2	Sınıf 3
Sınıf 1	40	0	0
Sınıf 2	0	36	4
Sınıf 3	0	2	38
% Başarım		%95	

Kullanılan sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.9, başarım grafikleri şekil 7.3 ile verilmiştir.

Çizelge 7.9: IRIS veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	96	95	95	94	95



Şekil 7.5: IRIS veri kümesi için sınıflandırıcıların başarımları

7.2.2 Tic-Tac-Toe veri kümesi ile elde edilen sonuçlar

Bölüm 7.1.1.B ile özellikleri verilen Tic-Tac-Toe veri kümesi üzerinde 3-en yakın komşu, 5-en yakın komşu, BP-ÇKA, Get-ÇKA ve ACO_R-ÇKA sınıflandırıcıları test edilmiştir. Tic-Tac-Toe veri kümesinde 2 sınıf ve sınıf-1'de 332, sınıf-2'de 626

örnek bulunmaktadır. Testlerde her sınıftan rasgele seçilen 100 elemanla eğitim kümesi, geriye kalan elemanlarla test kümesi oluşturulmuştur.

3-NN sınıflandırıcısı elde edilen sonuçları Çizelge 7.10 ile verilmiştir.

Çizelge 7.10: 3-NN sınıflandırıcısının Tic-Tac-Toe veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2
Sınıf 1	218	14
Sınıf 2	42	484
% Başarım	%92	

5-NN sınıflandırıcısı elde edilen sonuçları Çizelge 7.11 ile verilmiştir.

Çizelge 7.11: 5-NN sınıflandırıcısının Tic-Tac-Toe veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2
Sınıf 1	221	11
Sınıf 2	53	473
% Başarım	%91	

BP-ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.12 ile verilmiştir.

Çizelge 7.12: BP-ÇKA sınıflandırıcısının Tic-Tac-Toe veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2
Sınıf 1	223	9
Sınıf 2	4	522
% Başarım	%98	

Get-ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.13 ile verilmiştir.

Çizelge 7.13: Get-ÇKA sınıflandırıcısının Tic-Tac-Toe veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2
Sınıf 1	194	38
Sınıf 2	62	464
% Başarım	%86	

ACO_R -ÇKA sınıflandırıcısı elde edilen sonuçları Çizelge 7.14 ile verilmiştir.

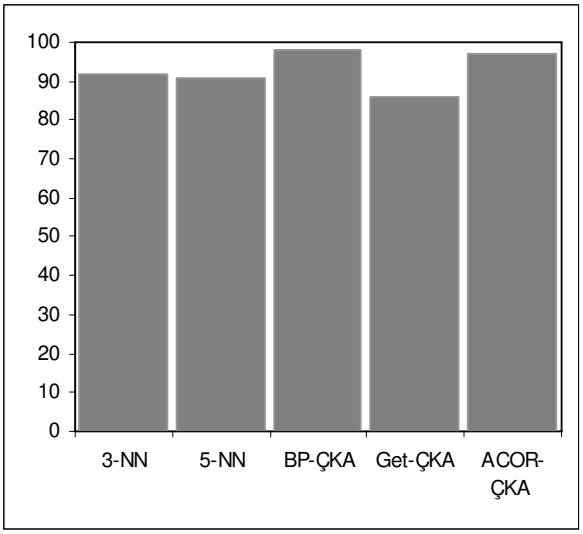
Çizelge 7.14: ACO_R-ÇKA sınıflandırıcısının Tic-Tac-Toe veri kümesi üzerindeki başarımı

	Sınıf 1	Sınıf 2
Sınıf 1	218	14
Sınıf 2	2	524
% Başarım	%97	

Kullanılan sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.15, başarım grafikleri şekil 7.6 ile verilmiştir.

Çizelge 7.15: Tic-Tac-Toe veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	92	91	98	86	97



Şekil 7.6: Tic-Tac-Toe veri kümesi için sınıflandırıcıların başarımları

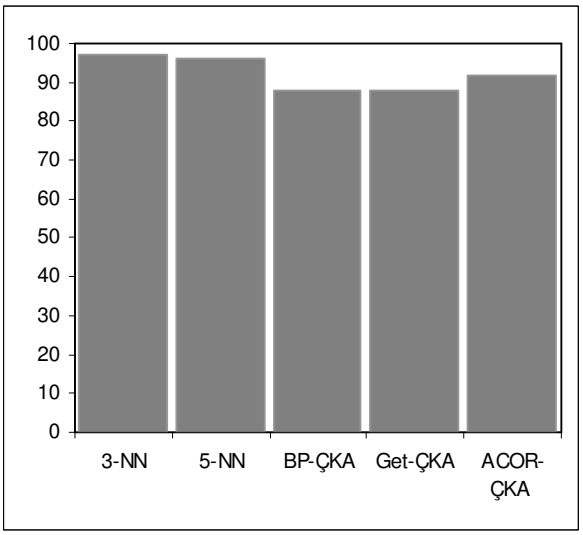
7.2.3 PenDigit veri kümesi ile elde edilen sonuçlar

Bölüm 7.1.2.1 ile özellikleri verilen PenDigit veri kümesi üzerinde 3-en yakın komşu, 5-en yakın komşu, BP-ÇKA, Get-ÇKA ve ACO_R-ÇKA sınıflandırıcıları test edilmiştir. PenDigit veri kümesinde {0,1,2,3,4,5,6,7,8,9} rakamlarını içeren 9 sınıf bulunmaktadır. 0 sınıfından 780, 1 sınıfından 779, 2 sınıfından 780, 3 sınıfından 719, 4 sınıfından 780, 5 sınıfından 720, 6 sınıfından 720, 7 sınıfından 778, 8 sınıfından 719, 9 sınıfından 719 örnek bulunmaktadır. Testlerde her sınıftan rasgele seçilen 100 elemanla eğitim kümesi, geriye kalan elemanlarla test kümesi oluşturulmuştur.

PenDigit veri kümesi üzerinde test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.16, başarım grafikleri Şekil 7.7 ile verilmiştir.

Çizelge 7.16: PenDigit veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	97	96	88	88	92



Şekil 7.7: PenDigit veri kümesi için sınıflandırıcıların başarımları

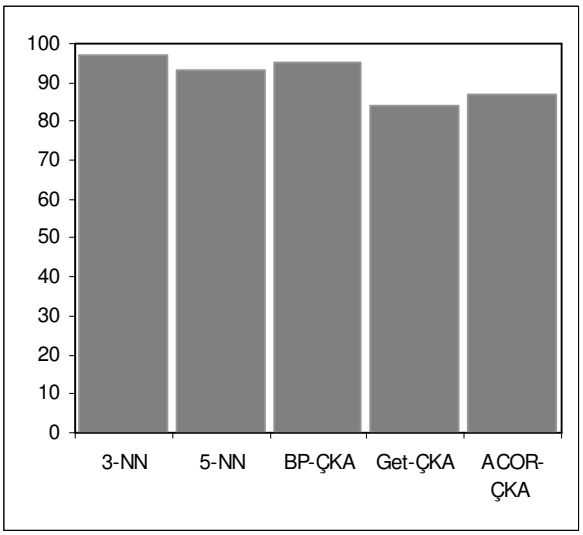
7.2.4 OptDigit veri kümesi ile elde edilen sonuçlar

Bölüm 7.1.2.B ile özellikleri verilen OptDigit veri kümesi üzerinde 3-en yakın komşu, 5-en yakın komşu, BP-ÇKA, Get-ÇKA ve ACO_R-ÇKA sınıflandırıcıları test edilmiştir. OptDigit veri kümesinde {0,1,2,3,4,5,6,7,8,9} rakamlarını içeren 9 sınıf bulunmaktadır. 0 sınıfından 376, 1 sınıfından 389, 2 sınıfından 380, 3 sınıfından 389, 4 sınıfından 387, 5 sınıfından 376, 6 sınıfından 377, 7 sınıfından 387, 8 sınıfından 380, 9 sınıfından 382 örnek bulunmaktadır. Testlerde her sınıftan rasgele seçilen 100 elemanla eğitim kümesi, geriye kalan elemanlarla test kümesi oluşturulmuştur.

OptDigit veri kümesi üzerinde test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.17, başarım grafikleri Şekil 7.8 ile verilmiştir.

Çizelge 7.17: OptDigit veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	97	93	95	84	87



Şekil 7.8: PenDigit veri kümesi için sınıflandırıcıların başarımları

7.2.5 İTU veri kümesi ile elde edilen sonuçlar

Bölüm 7.1.2.C ile özellikleri verilen İTU Karakter veri kümesi üzerinde 3-en yakın komşu, 5-en yakın komşu, BP-ÇKA, Get-ÇKA ve ACO_R-ÇKA sınıflandırıcıları test edilmiştir. İTU karakter veri kümesinde {0,1,2,3,4,5,6,7,8,9} rakamlarını içeren 9 sınıf bulunmaktadır. Tüm sınıflardan 500 örnek alınmıştır. Testlerde her sınıftan rasgele seçilen 100 elemanla eğitim kümesi, geriye kalan elemanlarla test kümesi oluşturulmuştur.

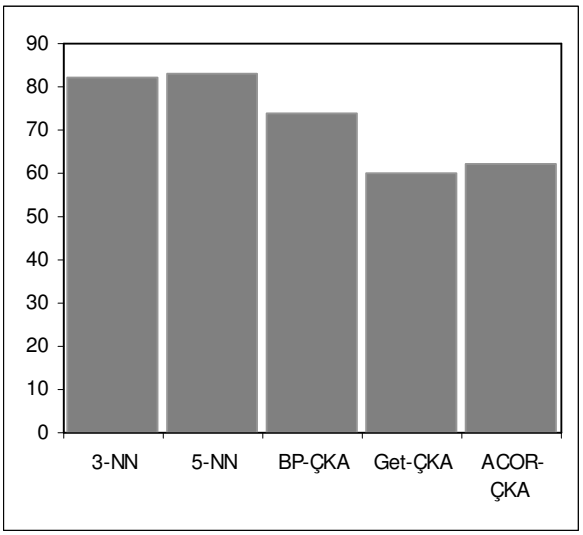
İTU veri kümesinden alınan örneklerine 3 çeşit öznitelik çıkarma yöntemi uygulanmıştır. Her bir öznitelik vektörüyle ayrı sonuçlar elde edilmiştir.

7.2.5.1 İTU veri kümesi ve kld (karhunen- loeve dönüşümü) öznitelik vektörü ile elde edilen sonuçlar

İTU veri kümesinden KLD ile elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.18, başarım grafikleri Şekil 7.9 ile verilmiştir.

Çizelge 7.18: İTU+KLD veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	82	83	74	60	62



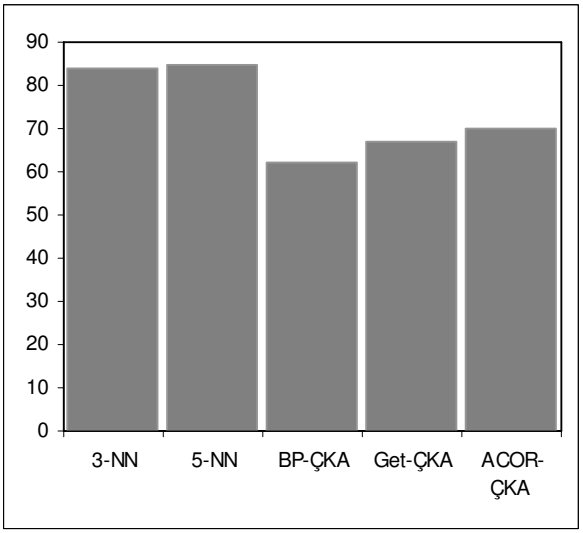
Şekil 7.9: İTU+KLD veri kümesi için sınıflandırıcıların başarımları

7.2.5.2 İTU veri kümesi ve şekilsel öznitelik vektörü ile elde edilen sonuçlar

İTU veri kümesinden şekilsel özellikler kullanılarak elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.19, başarım grafikleri Şekil7.10 ile verilmiştir.

Çizelge 7.19: İTU+Şekilsel veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	84	85	62	67	70



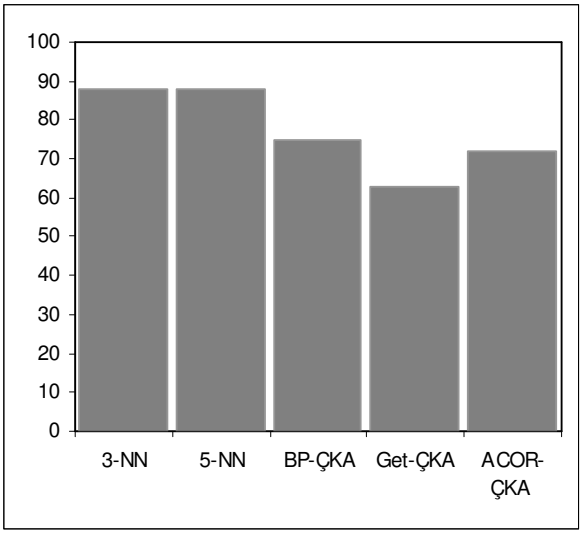
Şekil 7.10: İTU+Şekilsel veri kümesi için sınıflandırıcıların başarımları

7.2.5.3 İTU veri kümesi ve kld+şekilsel öznitelik vektörü ile elde edilen sonuçlar

İTU veri kümesinden şekilsel öznitelik vektörü ve KLD ile elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.20, başarım grafikleri Şekil 7.11 ile verilmiştir.

Çizelge 7.20: İTU+KLD+Şekilsel veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	88	88	75	63	72



Şekil 7.11: İTU+KLD+Şekilsel veri kümesi için sınıflandırıcıların başarımları

7.2.6 NIST veri kümesi ile elde edilen sonuçlar

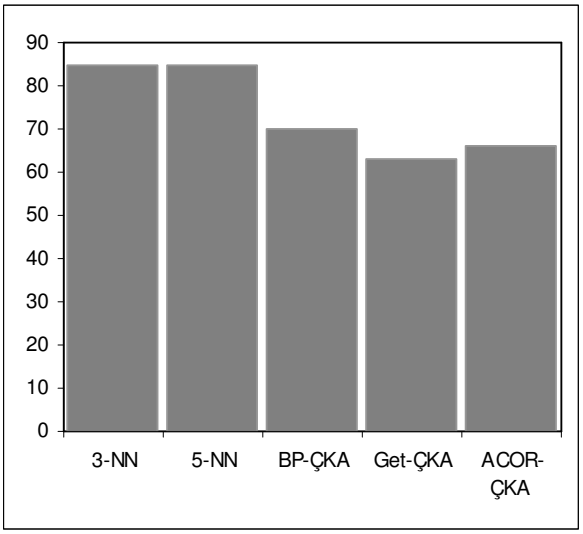
NIST veri kümesinden alınan örneklerine 3 çeşit öznitelik çıkarma yöntemi uygulanmıştır. Her bir öznitelik vektörüyle ayrı sonuçlar elde edilmiştir.

7.2.6.1 NIST veri kümesi ve KLD öznitelik vektörü ile elde edilen sonuçlar

NIST veri kümesinden KLD ile elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.21, başarım grafikleri Şekil 7.12 ile verilmiştir.

Çizelge 7.21: NIST+KLD veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	85	85	70	63	66



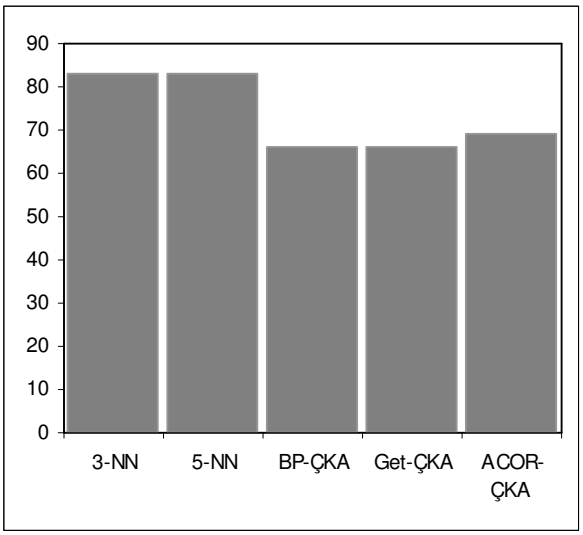
Şekil 7.12: NIST+KLD veri kümesi için sınıflandırıcıların başarımları

7.2.6.2 NIST veri kümesi ve şekilsel öznitelik vektörü ile elde edilen sonuçlar

NIST veri kümesinden şekilsel özellikler ile elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.22, başarım grafikleri Şekil7.13 ile verilmiştir

Çizelge 7.22: NIST+ Şekilsel veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	83	83	66	66	69



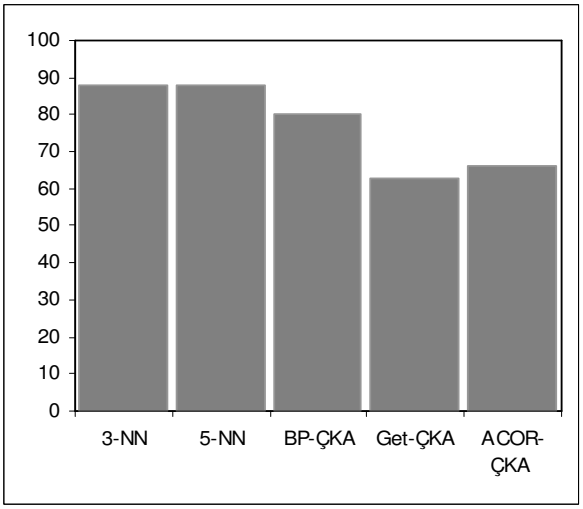
Şekil 7.13: NIST+Şekilsel veri kümesi için sınıflandırıcıların başarımları

7.2.6.3 NIST veri kümesi ve kld+şekilsel öznitelik vektörü ile elde edilen sonuçlar

NIST veri kümesinden şekilsel öznitelik vektörü ve KLD ile elde edilen öznitelik vektörü ile test edilen sınıflandırıcıların başarım çizelgeleri ekler bölümünde verilmiştir. Sınıflandırıcıların karşılaştırmalı başarımları Çizelge 7.23, başarım grafikleri Şekil 7.14 ile verilmiştir

Çizelge 7.23: NIST+ KLD+Şekilsel veri kümesi üzerinde sınıflandırıcıların karşılaştırmalı başarımı

Sınıflandırıcı	3-NN	5-NN	BP-ÇKA	Get-ÇKA	ACO_R-ÇKA
Başarım	88	88	80	63	66



Şekil 7.14: NIST+KLD+Şekilsel veri kümesi için sınıflandırıcıların başarımları

8. SONUÇ VE ÖNERİLER

Çalışmada yeni bir eniyileme yöntemi olan sürekli bölge karınca koloni optimizasyonu (ACO_R) çok katmanlı ağa uygulanmıştır. Gerçekleştirilen çok katmanlı ağ (ACO_R-ÇKA) ile el yazısı rakam veri kümeleri sınıflandırılmış ve sonuçlar literatürde kullanılan diğer yapay sinir ağları ile karşılaştırılmıştır.

El yazısı karakterlerinin sınıflandırılması sınıflandırıcılar için zor bir süreçtir. Karakterlerin kişiden kişiye büyük farklılıklar göstermesi, sınıf sayısının çok olması ve yüksek boyutta öz niteliklerin kullanılması yapay sinir ağlarının daha fazla sayıda nöron elemanı kullanmasını gerektirmekte, ağın genelleme yeteneğini azaltmakta, çalışma hızını ve sınıflama başarımını düşürmektedir. Bu nedenlerden dolayı, el yazısı rakamlarının sınıflandırılması, örüntü tanımada zor bir görev olarak bilinmektedir [53]

ITU ve NIST veri kümelerine ait el yazısı rakam karakterleri sınıflandırılırken öz nitelik olarak, şekilsel öznitelikler, Karhounen Loeve Dönüşümü (KLD) yöntemi ile şekilsel ve KLD yöntemi birlikte kullanılmıştır. Şekilsel öz niteliğin KLD'ye göre daha yüksek başarım vermiştir. Öz niteliklerin birlikte kullanılmasıyla başarımda artış sağlandığı görülmektedir.

ACO_R yöntemi, sürekli değerler alan parametreleri bir uyum fonksiyonunu optimize edecek biçimde değiştirir. Bir tabloda saklanan çözümler, uyum fonksiyonunda aldıkları değerlere göre sıralanırlar ve iyi çözümlerden yeni çözümler üretilir. Üretilen yeni çözümler belirli sayıda kötü çözüm ile değiştirilir. Böylece global çözüme yaklaşılır. Çözüm sayısının artırılması ise yerel minimumda kalma riskini azaltacaktır çünkü algoritma bu sayede birçok farklı noktada çözüm arayışındadır. Genetik ağlar optimizasyonda sıkça kullanılmaktadır fakat parametrelerin bitler ile temsil edilmesi, nicemleme hatasına neden olacaktır. Ayrıca parametrelerin üst ve alt limitleri her zaman belirlenemez.

ACO_R ile eğitilmiş çok katmanlı ağın yapısı, genetik çok katmanlı ağ ile benzerlik göstermektedir. Fakat Bu çalışmada önerilen yapay sinir ağının optimizasyon

algoritmasının genetik algoritmalara karşı üstünlükleri vardır. Socha ve Dorigo [38] çalışmalarında ACO_R ve genetik ağları karşılaştırmışlardır. Bunun yanında, bu çalışmada kullanılan uyumluluk fonksiyonu, her bölgenin temsil ettiği sınıf dışındaki elemanlar nedeniyle daha düşük değerler almasına neden olmaktadır. Bu da farklı sınıflara ait elemanların daha az nöronla ve kolay ayrılmasına sebep olacaktır.

Rakam sınıflandırma sonuçları incelendiğinde, KNN sınıflandırıcısı ile diğer ağlara göre daha yüksek başarımlar elde edildiği görülür. KNN sınıflandırma yaparken bütün eğitim kümesi elemanlarını hafızasında tutar. Bu nedenle bu sınıflandırıcının sınıflandırma yeteneği diğer ağlara göre yüksektir. KNN algoritması basit bir algoritma olmasına rağmen eğitim kümesi boyutu arttıkça işlem yükü ve eğitim kümesinin saklanması için gereken bellek ihtiyacı da oldukça artmaktadır.

Klasik çok katmanlı yapay sinir ağının önerilen ağa göre üstünlüğü, içerdiği gizli katman ve lineer olmayan aktivasyon fonksiyonu sayesinde oluşan hiper düzlemlerin uzayda çok değişik biçimlerde bulunabilmesi, böylece ağın karmaşık sınıf dağılımlarını da öğrenebilir olmasıdır. Örneğin, önerilen ağ iki boyutlu öznitelik uzayı için yalnızca doğrular ile bölgeler oluştururken, klasik çok katmanlı ağ, eğriler kullanarak karmaşık sınıf dağılımlarını öğrenebilir. Bu nedenle, (ACO_R_ÇKA) ağı aynı sınıf dağılımını daha fazla sayıda nöron ile öğrenecektir.

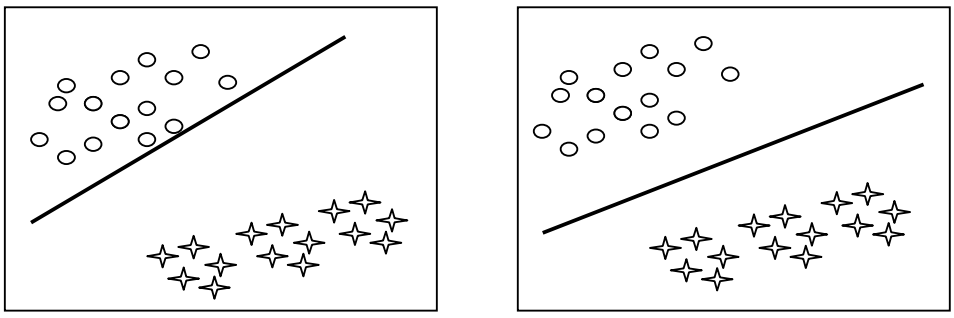
Önerilen ağın klasik yapay sinir ağına karşı önemli avantajlarından biri, ağın topolojisinin önceden belirlenmesine ihtiyaç duymamasıdır. Ağ, artımsal olarak gerektiği kadar nöron ekleyerek eğitimini tamamlamaktadır. Klasik yapay sinir ağında katman sayısı ve her katmandaki nöron sayısı genellikle birkaç deneme ile belirlenmektedir. ACO_R_ÇKA ağında ise gerektiği kadar nöron eğitim aşamasında ağa eklenmektedir.

Bu çalışmada yeni bir çok katmanlı ağ eğitim yöntemi önerilmektedir. Önerilen ağ ile çeşitli deneme veri kümelerinin sınıflama performansları da incelenmiştir. Sonuçlar incelenirse ACO_R_ÇKA ağının düşük boyutlu veri kümelerinde daha iyi başarım sergilediği görülmektedir.ACO_R_ÇKA ağının eğitimi bir optimizasyon yöntemini temel almaktadır ve bu optimizasyon yöntemi boyut sayısının artışından kötü olarak etkilenmektedir. Bu nedenle, öznitelik uzayının boyutu arttıkça ağın verimli bir biçimde eğitimi ve sınıflandırma başarımı da düşmektedir. Düşük

boyutlarda ACO_R_ÇKA ağının başarımının daha yüksek olması ve boyut arttıkça düşmesinin nedeni budur.

İleriki çalışmalarda, aynı ağ yapısı yeni optimizasyon yöntemleri ile eğitilmesi incelenebilir. Örneğin Karaboğa ve Baştürk [23] tarafından önerilen Yapay arı kolonisi (Artificial Bee Colony, ABC) bal arılarının yiyecek bulma davranışlarını taklit eden bir algoritmadır. Bu yöntem yapay sinir ağına uygulanarak ağın başarım artışı incelenebilir.

Eniyileme yönteminin başarımını doğrudan etkileyen bir faktör, kullanılan uyum fonksiyonudur. Bu çalışmada, genetik çok katmanlı ağda kullanılandan farklı bir uyum fonksiyonu kullanılmıştır. Bölüm 6.1.1.2'de anlatılan uyum fonksiyonu incelenirse, uygunluk değeri, bölge aitlik tablosundan yararlanılarak elde edilmektedir. Bu tablo, eğitim kümesindeki her elemanının hangi bölgede bulunduğuna bakılarak elde edilir. Fakat tablo oluşturulurken, eğitim kümesi elemanının bölge sınırına yakınlığı ile ilgilenilmez. Diğer bir deyişle, bir sınıfa ait elemanların bölge sınırına yakın ya da uzak oluşu uygunluk değerini değiştirmez. Aşağıda bu durumu gösteren iki şekil vardır. Her iki şekilde de elde edilen bölge matrisi aynıdır ve dolayısıyla uyum fonksiyonunun ürettiği değer aynı olacaktır (iki sınıfta tam olarak ayrıldığı için değer 1 olacaktır) . Fakat yakın bir dağılıma sahip bu iki sınıf için tam ortadan geçen bir karar doğrusu, ağın genelleme yeteneğini artıracağından test aşamasında daha yüksek başarım sağlayabilir. İleriki çalışmalarda, bölge tablosu hazırlanırken elemanların bir bölgeye ait olup olmadıkları değil de, bu bölgeye ilişkin aitlik değerleri (0-1 arası değişen bir reel sayı) hesaplanabilir. Bu sayede ağın genelleme yeteneği artırılabilecektir.

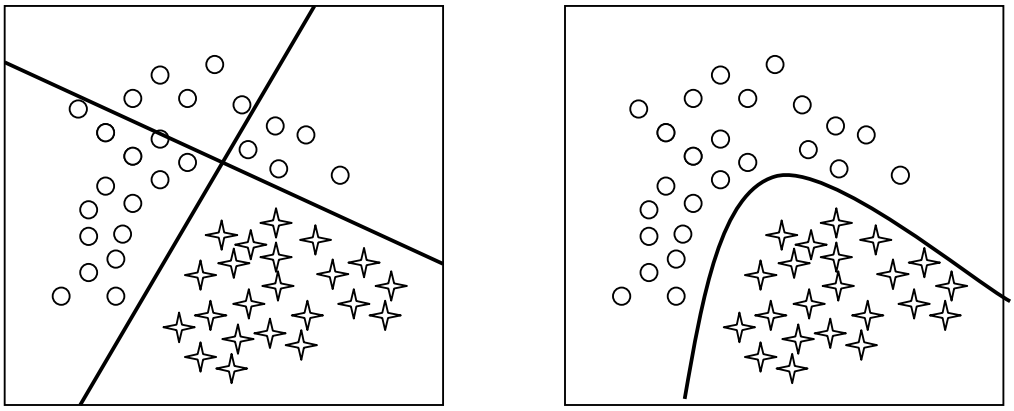


Şekil 8.1: Belirlenen uyum fonksiyonuna göre aynı uygunluk değerini üretecek iki farklı düzlem

ACO_R_ÇKA uzayı hiper düzlemler ile bölgelere ayırır. Düzlem denklemi, doğrusal bir denklemdir, aşağıdaki eşitlik bir doğru denklemidir:

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots\ldots\ldots + w_n x_n = 0 \qquad \textbf{(8.1)}$$

Bu, iki boyutta bir doğruya, 3 boyutta bir düzleme karşılık gelir. Fakat çalışmada önerilen yapay sinir ağı eğri bir hiper düzlem oluşturamaz. Bu da karmaşık ve iç içe sınıf dağılımları için ağın eğitilmesini güçleştirir. Örneğin aşağıdaki iki şekilde aynı dağılım önce doğrular ile farklı bölgelere ayrılması, ardından bir eğri ile ayrılması gösterilmiştir.



Şekil 8.2: Farklı sınıflara ait eğitim kümesi elemanlarının a) doğrular ile, b)bir eğri yardımıyla ayrıştırılması

Karar yüzeyinin, bir eğri olması için, doğrusal olmayan bir nöron denklemi kullanılmalıdır. Bunun için örneğin, nöron denklemi aşağıdaki gibi seçilebilir:

$$w_0 + w_1 x_1 + \ldots\ldots + w_n x_n + w_{n+1} x_1^2 + w_{n+2} x_2^2 + \ldots\ldots + w_{2n} x_n^2 = 0 \qquad \textbf{(8.2)}$$

Yukarıdaki denklem ile eğri yüzeyler elde edilebilir fakat, optimize edilecek parametre sayısı (w_0 … w_{2n}) bir kat daha artmıştır. İleriki çalışmalarda, doğrusal olmayan denklemler kullanılarak karmaşık dağılımların sınıflama başarımlarındaki artış incelenebilir.

Kullanılan öz nitelikler, sınıflandırıcı başarımını doğrudan etkilemektedir. Öz nitelik araştırma amacıyla, öz niteliklerin gerekliliklerinin araştırılması ve gereksiz olanlarının kullanılmaması daha iyi başarım sağlayacaktır. Diverjans analizi yöntemi [54] öznitelikleri, sınıfları birbirinden ayırabilme yeteneğine göre sınıflandırarak öz

nitelik seçimine yardımcı olur. İleriki çalışmalarda, yeni öz nitelikler araştırılarak daha verimli sınıflama yapılması sağlanabilir.

Sonuç olarak, çalışmada çok katmanlı yapay sinir ağının eğitimine ilişkin bir yöntem geliştirilmiş ve sınıflama başarımları deneme amaçlı veri kümelerinin ve rakam görüntülerinin sınıflandırılmasında incelenmiştir. Gerçekleştirilen ağın kuvvetli ve zayıf yönleri incelenmiş ve zayıf yönlerinin iyileştirilmesi için yapılabilecekler önerilmiştir. Tez çalışmasında önerilen ağın, topolojisini kendi kendine oluşturması ve eğitiminin kolay olması bakımından birçok uygulamada tercih edilebilir olacağı düşünülmektedir.

KAYNAKLAR

[1] **Brunette E. S., Flemmer R. C. and Flemmer C. L.**, 2009, A Review of Artificial Intelligence, *Proceedings of the 4th International Conference on Autonomous Robots and Agents*, Wellington, New Zealand

[2] **Uğur A. ve Aydın D.**, Ant System Algoritmasının Java ile Görselleştirilmesi, 2006, *Akademik Bilişim*, Bildiri No: 53, http://ab.org.tr/ab06/bildiri/53.pdf alındığı tarih: 10.09.2009

[3] **Deneubourg J.L., Pasteels J.M.and Verhaeghe J.C.**, 1983, Probabilistic Behaviour in Ants: a Strategy of Errors, *Journal of Theoretical Biology*, **105**, 259–271.

[4] **Sankar K. P. and Sushmita M.**, 1992, Multilayer Perceptron, Fuzzy Sets and Classification, *IEEE Transaction on Neural Networks*, **Vol. 3**, No. 5.

[5] **Ergezer H., Dikmen M. ve Özdemir E**., 2003, Yapay Sinir Ağları ve Tanıma Sistemleri, Başkent Üni. İktisadi ve İdari Bilimler Fak., Elyad-Dal Ar. Lab. *Pivolka* **Sayı:6** Sayfa:14-15.

[6] **Erdem O. A. ve Uzun E.,** 2005, Yapay Sinir Ağları ile Türkçe Tımes New Roman, Arial ve Elyazısı Karakterleri Tanıma, *Gazi Üniv. Müh. Mim. Fak. Der*. **Cilt 20**, No 1, 13-19.

[7] **Abdi H., Valentine D., Edalman B.**, 1999, Neural Networks, Sape Publication.

[8] **Bernard I.**, 1993, Multilayer Perceptron and Uppercase Handwritten Chracters Recognition, *Document Analysis and Recognition, Proceedings of the Second International Conference* pp:935 - 938

[9] DTREG, Software For Predictive Modeling and Forecasting, http://www.dtreg.com/mlfn.htm, alındığı tarih 18.10.2009

[10] **Widrow B. And Hoff M. E.,** 1960, Adaptive Switching Circuits, 1960 *IRE WESCON Convention Reccord*, Part 4, New York, IRE, pp. 96-104

[11] **Nittu T.**, 2003, Solving The XOR Problem And The Detection Of Symmetry Using A Single Complex-Valued Neuron, ScienceDirect, *Neural Networks* **16**,1101-1105.

[12] XOR problem Theory, http://home.agh.edu.pl/~vlsi/AI/xor_t/en/main.htm alındığı tarih 15.10.2009

[13] **Yüksel A. ve Korürek M.,** 2008, EMG İşaretlerinin Genetik Algoritmalar ve Çok Katmanlı Yapay Sinir Ağı ile Sınıflandırılması, *Eleco-2008*, Bursa

[14] **Salerno, J.**, 1997, Using the Particle Swarm Optimization Technique to Train a Recurrent Neural Model, *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*.

[15] **Zhang, C., Shao, H., Li, Y.,** 2000, Particle Swarm Optimisation For Evolving Artificial Neural Netowrk, *Proceedings of IEEE International Conference on Systems, Man,and Cybernetics* **4,** 2487-2490

[16] **Xiaorong P., Zhongjie F., and Yongguo L.,** 2007, Multilayer Perceptron Networks Training Using Partical Swarm Optimization with Minimum Velocity Constratints , ISNN 2007, Part III, LNCS **4493**, pp. 237-245.

[17] **Pham D.T. and Sağıroğlu S.,** 2001, Training Multilayered Perceptrons For Pattern Recognition: A Comparative Study Of Four Training Algorithms, *International Journal of Machine Tools & Manufacture* **41,** 419–430.

[18] **Schaffer J. D., Whitley D., Esshelman L.J.**, 1992, Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art, *Int. Wks on Combinations of Genetic Algorithms and Neural Networks, (COGANN-92),*1 – 37.

[19] **Seiffert U.,** 2001, Multiple Layer Perceptron Training Using Genetic Algorithms, *European Symposium on Artificial Neural Networks*, pp. 159-164.

[20] Karınca Çiftliği Benzetimi Yazılımı, http://www.forgefx.com/casestudies /prenticehall/ph/ants/ants.htm, alındığı tarih 25.10.2009

[21] **Onet V.E., Vladu E.,** 'Nature Inspıred Algorithms and Artificial Intelligence', http://electroinf.uoradea.ro/reviste%20CSCS/documente/JCSCS_2008 /JCSCS_2008_13_Onet_1.pdf, alındığı tarih 12.10.2009

[22] **Kennedy and R. C. Eberhart**, 1995, Particle Swarm Optimization,, *in Proc.IEEE Int. Conf. Neural Netw. IV,* pp. 1942–9148.

[23] **Karaboga D. and Basturk B.,** 2007, A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, *Journal of Global Optimization*, **Volume:39**, Issue:3, pp:459-171.

[24] Ant Colony Optimization, http://iridia.ulb.ac.be/~mdorigo/ACO/RealAnts.html, alındığı tarih 22.09.2009

[25] **Alaykıran K. ve Engin O.,** 2005, Karınca Kolonileri Metasezgiseli ve Gezgin Satıcı Problemleri Üzerinde Bir Uygulaması , *Gazi Üniv. Müh. Mim. Fak. Der.*, **Cilt 20**, No 1, 69-76.

[26] **Dalkılıç G. ve Türkmen F.,** *Karınca Kolonisi Optimizasyonu*, http:// www.bilmuh.gyte.edu.tr/~ypbs/pdfler/A-I-2-GokhanDalgic.pdf, alındığı tarih 23.09.2009

[27] **Yakhno T. , Ekin E.**, 2002, Ant Systems: Another Alternatine for Optimization Problems, *Advances in Information Systems*, 324-326

[28] **Dorigo M., Maniezzo V. and Colorni A.**, 1991, The Ant System: An Autocatalytic Optimizing Process, *Technical Report* No. 91-016, Italy.

[29] **Dorigo M., Di Caro G.** , 1999, Ant Colony Optimization: A New Meta-Heuristic, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, **Vol: 2**, on pp: 1477

[30] **Keskintürk T. ve Söyler H.**, 2006, Global Karınca Kolonisi Optimizasyonu, *Gazi Üniv. Müh. Mim. Fak. Der.* , **Cilt 21**, No 4, 689-698.

[31] **Dorigo M., Maniezzo V. and Colorni A**, 1996, The Ant System, Optimization by a Colony Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, **Vol.26**, No.1, pp.1-13.

[32] **Dorigo M., Gambardella L.M.,** 1997, 'Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem'*, , IEEE Transactions on Evolutionary Computation*, **Vol.1**, No.1.

[33] **Stützle T., Hoos TH.,** 1997, The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem, *In Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC'97*), 308-313.

[34] **Bullnheimer B., Hartl R.F. and Strauss C.**, 1997, A New Rank Based Version of the Ant System: A Computational Study, *Central European Journal for Operations Research and Economics.*

[35] **Dorigo M., Maniezzo V. and Colorni A.**, 1991, The Ant System: An autocatalytic optimizing process, *Technical Report 91-016 Revised*, Dipartimento di Elettronica, Italy

[36] **Stützle T., Dorigo M.**, 1999, ACO Algorithms For The Quadratic Assignment Problem, *New Ideas in Optimization*, pp:33-50

[37] **Gambardella L.M., Dorigo M.** , Ant Colonies For The Quadratic Assignment Problem, http://www.idsia.ch/~luca/tr-idsia-4-97.pdf, alındığı tarih 27.10.2009

[38**] Chun-tao M., Xiao-xia L. and Li-yong Z.**, 2007, Radial Basis Function Neural Netwoek Based On Ant Colony Optimization, *Int. Conf. On Comp. Intelligence and Security Workshop.*

[39] **Socha K., Dorigo M.,** 2008, Ant Colony optimization for Continous Domains, *European Jornal of Operational Research,* **185**, 1155-1173

[40] **Mori S., Suen C.Y. and Yamamoto K.**, 1992, Historical Review Of OCR Research and Development, *Proceedıngs Of The IEEE.* **Vol 80**. No 7**.** pp.1029-1058

[41] **Verma B., Blumensteın M. and Kulkarnı S**., Recent Achievements In Off-Lıne Handwrıtıng Recognıtıon Systems, http://ww98.griffith.edu.au /dspace/bitstream/10072/15597/1/9787.pdf, alındığı tarih 07.10.2009

[42] **Araokar S.**, 2005, Visual Character Recognition Using Artificial Neural Networks, http://cogprints.org/4334/1/VCRANNSA.pdf, alındığı tarih 28.09.2009

[43] **Plamondon R. and Srihari S.N.**, 2000, On-Line and Off-Line Handwriting Recognition:A Comprehensive Survey, *IEEE Transactıons On Pattern Analysıs And Machıne Intellıgence*. **Vol. 22**, No. 1.

[44] **Arıca N. and Yarman Vural F.T.** , 2001, An Overview of Character Recognition Focused on Off-Line Handwriting, *IEEE Transactıons On Systems, Man, And Cybernetıcs—Part C: Applıcatıons And Revıews,* **Vol. 31**, No. 2.

[45] **Gokmen M. , Çapar A. , Taşdemir K. , Kılıç Ö.**, 2002, Türkçe Elyazısı Tanıma Sistemlerinde Öznitelik Çıkarma Ve Sınıflandırma Yöntemlerinin Karşılaştırması, *URSI-2002*, http://www.ursi.org.tr/2002-1.Ulusal%20Kongre/ursicd1/C14.pdf, alındığı tarih 22.09.2009

[46] **Turkolu İ.** 'Örüntü Tanıma Sistemleri- Ders Notları', Fırat Üniv., Fen Bilimleri Enstitüsü, 2003 http://perweb.firat.edu.tr/personel/yayinlar/fua_69/69_22586.pdf, alındığı tarih 12.10.2009

[47] **Brown E W**, Applying Neural Networks to Character Recognition, http://www.ccs.neu.edu/home/feneric/charrecnn.html, alındığı tarih 22.10.2009

[48] **Öz C. and Köker R,** 2001, Vehicle Licence Plate Recognition Using Artificial Neural Networks *, ELECO'2001*, Bursa,Turkey

[49] **Hamming R.W,** 1950, Error Detecting and Error Correcting codes, *The Bell System Technical Journal*, **Vol:XXVI**, No:2, pp:147-161

[50] Nearest Neighbour Rule: A short Tutorial http://cgm.cs.mcgill.ca/~soss/cs644/projects/simard/, alındığı tarih 27.10.2009

[51] **Zhan Y., Chen H. and Zhang G.**, 2006, An Optımızatıon Algorıthm Of K-nn Classıfıcatıon, *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*

[52] **Smith J. S., Bourgoin M., Sims K. and Voorhees H. L.**, 1994, Handwritten Character Classification Using Nearest Neighbor in Large Databases, *IEEE Transactıons On Pattern Analysıs And Machıne Intellıgence,* **Vol. 16**, No. 9, 915-919

[53] **Sahel Ba-Karait, N.O. and Shamsuddin S.M.**, 2008, Handwritten Digits Recognition Using Particle Swarm Optimization, *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference,* pp.615-619

[54] **Dokur Z. and Olmez T.**, 2009, Feature Determination For Heart Sounds Based On Divergence Analysis*, Digital Signal Processing*, **Volume 19**, Issue 3, Pages 521-531

EKLER

EK A : Sınıflandırıcılara ait başarım çizelgeleri

Çizelge A.1: 3-NN sınıflandırıcısının PenDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	673	1	0	0	4	0	1	0	0	1
1	0	608	32	24	1	1	11	2	0	0
2	0	7	670	0	0	0	0	3	0	0
3	0	7	1	607	1	1	0	2	0	0
4	0	0	1	0	675	0	0	2	0	2
5	0	0	0	6	0	607	0	0	1	6
6	0	0	1	0	0	1	618	0	0	0
7	0	3	5	0	0	0	0	670	0	0
8	2	10	0	0	0	4	0	8	595	0
9	0	0	0	6	2	5	0	0	0	606
% Başarım						%97				

Çizelge A.2: 5-NN sınıflandırıcısının PenDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	660	1	0	0	16	0	1	0	1	1
1	0	597	45	23	1	3	10	0	0	0
2	0	7	672	0	0	0	0	1	0	0
3	0	9	0	608	1	1	0	0	0	0
4	0	1	0	0	672	0	0	3	0	4
5	0	1	0	7	0	609	0	0	3	0
6	0	0	1	0	0	1	618	0	0	0
7	0	15	1	1	0	0	0	657	4	0
8	5	4	0	0	0	6	1	2	601	0
9	17	7	0	3	2	25	0	0	0	565
% Başarım						%96				

Çizelge A.3: BP-ÇKA sınıflandırıcısının PenDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	640	6	0	0	9	0	2	0	22	1
1	1	472	113	36	15	0	6	9	10	17
2	0	19	655	0	0	0	0	5	0	1
3	0	12	0	605	0	0	0	1	0	1
4	0	6	1	2	640	0	1	0	0	30
5	0	0	0	43	0	395	27	3	3	149
6	0	0	0	0	0	1	614	0	5	0
7	0	44	0	20	0	0	3	584	1	26
8	28	11	0	1	0	1	0	2	569	7
9	4	17	9	10	4	6	1	4	1	563
% Başarım						%88				

Çizelge A.4: Get-ÇKA sınıflandırıcısının PenDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	617	0	0	0	24	0	6	11	17	5
1	0	562	39	11	10	23	1	17	4	12
2	0	15	647	0	0	7	0	11	0	0
3	0	40	1	552	4	1	3	0	8	10
4	6	9	1	0	607	8	7	14	0	28
5	2	4	1	3	2	562	5	17	2	22
6	6	0	0	0	16	28	570	0	0	0
7	0	25	0	20	1	1	0	607	24	0
8	13	24	0	15	0	9	0	20	518	20
9	0	11	2	0	20	55	0	3	2	526
% Başarım						% 88				

Çizelge A.5: ACO_R-ÇKA sınıflandırıcısının PenDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	647	1	0	0	9	14	0	0	8	1
1	0	575	46	20	0	21	1	7	5	4
2	0	24	648	0	0	0	0	6	1	1
3	0	3	1	606	1	3	0	5	0	0
4	0	6	1	2	652	2	0	0	0	17
5	1	1	0	13	0	592	1	0	5	7
6	0	0	0	1	3	4	596	1	15	0
7	0	3	0	18	0	4	0	638	9	6
8	25	3	2	8	1	24	2	10	536	8
9	0	6	0	3	19	33	0	10	5	543
% Başarım						%92				

Çizelge A.6: 3-NN sınıflandırıcısının OptDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	274	0	0	0	1	0	1	0	0	0
1	0	284	1	0	0	0	0	1	0	3
2	0	0	279	0	0	0	0	1	0	0
3	0	0	1	280	0	3	0	1	0	4
4	1	0	0	0	273	0	3	1	1	8
5	0	0	0	1	0	262	2	0	0	11
6	0	0	0	0	0	0	277	0	0	0
7	0	2	0	1	0	0	0	284	0	0
8	0	5	3	2	0	1	4	0	265	0
9	0	1	0	4	0	0	0	5	1	271
% Başarım						%97				

Çizelge A.7: 5-NN sınıflandırıcısının OptDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	270	1	2	0	0	2	1	0	0	0
1	1	249	3	3	6	2	2	1	18	4
2	0	2	262	6	0	1	1	1	5	2
3	0	1	3	274	0	3	0	3	2	3
4	1	6	0	0	271	0	3	0	0	6
5	0	0	1	0	0	265	0	0	0	10
6	0	2	0	0	0	0	275	0	0	0
7	0	1	1	0	0	0	0	285	0	0
8	0	12	3	7	8	2	1	1	245	1
9	0	3	4	9	6	3	0	2	5	250
% Başarım						%93				

Çizelge A.8: BP-ÇKA sınıflandırıcısının OptDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	274	0	0	0	1	0	1	0	0	0
1	0	284	1	0	0	0	0	0	1	3
2	0	0	278	0	0	0	0	1	0	1
3	0	1	1	280	0	3	0	2	1	1
4	1	1	0	1	278	0	2	1	0	3
5	0	0	0	3	0	268	0	0	1	4
6	0	2	0	0	0	0	275	0	0	0
7	0	1	0	1	0	0	0	284	0	1
8	0	9	2	0	0	1	2	0	263	3
9	0	2	0	2	1	2	0	6	1	268
% Başarım						%95				

Çizelge A.9: Get-ÇKA sınıflandırıcısının OptDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	245	0	0	0	2	2	0	0	20	7
1	0	236	4	4	16	0	2	1	5	21
2	0	2	251	1	0	6	2	4	10	4
3	0	8	16	235	0	5	0	5	10	10
4	2	20	0	0	216	1	18	13	10	7
5	3	2	9	8	0	222	3	4	10	15
6	0	4	0	0	6	2	252	0	13	0
7	1	9	0	1	5	0	0	265	5	1
8	0	10	0	8	4	5	4	3	243	3
9	1	2	1	22	4	11	1	6	23	211
% Başarım						% 84				

Çizelge A.10: ACO_R-ÇKA sınıflandırıcısının OptDigit veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	244	1	3	0	7	4	3	3	5	6
1	0	249	9	0	4	0	7	0	13	7
2	0	4	247	14	2	0	1	0	8	4
3	0	4	5	259	0	8	0	3	2	8
4	0	5	1	0	266	0	4	0	3	8
5	0	0	0	7	6	239	3	2	0	19
6	2	3	2	0	6	0	263	0	1	0
7	0	6	2	9	4	3	0	251	0	12
8	0	11	5	3	6	6	5	1	227	16
9	1	4	0	11	9	15	1	0	2	239
% Başarım						%87				

Çizelge A.11: 3-NN sınıflandırıcısının İTU+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	382	1	0	4	0	0	13	0	0	0
1	2	381	4	0	4	0	5	2	2	0
2	11	7	361	2	1	1	0	8	8	1
3	11	15	9	321	0	12	0	8	19	5
4	9	9	2	1	331	3	29	1	6	9
5	16	17	7	8	2	315	4	0	10	21
6	20	8	1	0	19	1	348	1	2	0
7	0	23	19	11	1	0	4	330	7	5
8	30	17	6	13	5	13	20	2	290	4
9	25	28	9	23	11	26	2	5	16	255
% Başarım						%82				

Çizelge A.12: 5-NN sınıflandırıcısının İTU+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	389	1	0	0	0	0	10	0	0	0
1	3	369	8	3	3	1	7	0	4	2
2	9	2	368	4	0	3	1	7	6	0
3	8	14	8	327	0	7	1	9	12	14
4	9	8	1	0	319	4	43	2	7	7
5	18	7	2	9	3	332	6	0	5	18
6	12	11	0	0	10	0	364	0	3	0
7	2	25	15	9	2	0	4	336	3	4
8	10	17	6	19	1	11	24	3	303	6
9	22	18	1	24	15	42	2	9	18	249
% Başarım						%83				

Çizelge A.13: BP-ÇKA sınıflandırıcısının İTU+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	360	8	8	0	1	6	15	0	0	2
1	1	354	9	5	6	1	3	4	1	16
2	9	10	314	19	0	15	4	20	8	1
3	8	12	11	318	3	17	4	21	5	1
4	18	22	1	0	287	1	40	18	2	11
5	16	21	5	19	3	304	9	4	0	19
6	19	42	4	0	29	0	305	1	0	0
7	1	26	7	5	2	0	1	339	7	12
8	6	40	37	28	5	36	29	24	187	8
9	21	20	15	35	18	52	3	19	13	204
% Başarım						%74				

Çizelge A.14: Get-ÇKA sınıflandırıcısının İTU+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	162	3	0	3	4	6	11	0	8	3
1	0	134	3	7	7	1	3	18	14	13
2	17	4	118	13	5	9	11	12	7	4
3	3	9	9	142	4	4	0	7	7	15
4	4	23	1	0	134	5	15	4	8	6
5	10	5	6	14	4	115	15	6	12	13
6	24	6	6	2	31	5	102	10	14	0
7	0	14	7	5	15	3	4	133	10	9
8	10	18	12	12	10	9	24	8	86	11
9	7	8	3	18	13	29	2	15	11	94
% Başarım						%60				

Çizelge A.15: ACO_R -ÇKA sınıflandırıcısının İTU+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	157	4	2	7	2	3	9	3	7	6
1	1	155	3	6	8	1	13	1	10	2
2	13	3	123	14	2	6	5	8	13	13
3	6	4	11	129	2	9	4	16	8	11
4	1	9	4	2	111	4	21	6	24	18
5	10	5	13	29	1	100	4	3	11	24
6	3	11	1	2	8	1	163	1	8	2
7	5	3	16	21	8	2	4	122	4	15
8	24	9	7	17	1	17	14	5	91	15
9	10	4	5	26	2	11	1	19	20	102
% Başarım						%62				

Çizelge A.16: 3-NN sınıflandırıcısının İTU+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	295	0	0	1	0	1	2	0	0	1
1	0	271	9	7	3	1	1	2	2	4
2	5	2	281	3	3	6	0	0	0	0
3	1	4	5	263	0	4	0	2	9	12
4	2	2	4	0	253	2	24	3	0	10
5	7	1	21	16	6	224	1	0	8	16
6	2	1	0	2	10	1	280	0	4	0
7	0	9	10	6	2	2	0	252	2	17
8	4	3	8	11	2	27	6	4	231	4
9	4	9	3	28	2	40	0	11	8	195
%Başarım						%84				

Çizelge A.17: 5-NN sınıflandırıcısının İTU+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	294	0	1	0	0	1	2	0	1	1
1	0	257	12	9	2	1	0	6	8	5
2	4	0	271	4	0	11	1	5	4	0
3	1	4	6	266	1	2	1	1	7	11
4	1	2	0	0	256	4	28	3	2	4
5	4	2	2	18	9	224	2	3	19	17
6	2	1	0	1	4	4	285	0	3	0
7	0	2	11	12	1	3	2	251	4	14
8	1	2	1	13	0	28	9	1	239	6
9	0	4	1	24	4	19	0	24	17	207
%Başarım						%85				

Çizelge A.18: BP-ÇKA sınıflandırıcısının İTU+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	286	1	4	1	2	0	3	0	3	0
1	1	246	14	10	1	0	4	0	20	4
2	11	2	237	29	7	1	1	0	7	5
3	3	9	14	241	2	0	0	0	18	13
4	4	4	7	1	223	0	50	0	6	5
5	22	2	65	45	59	2	2	0	73	30
6	6	6	3	1	9	0	270	0	5	0
7	0	14	76	7	40	54	0	0	59	50
8	3	16	7	16	4	0	13	0	228	13
9	5	16	2	51	52	0	0	0	28	146
% Başarım						%62				

Çizelge A.19: Get-ÇKA sınıflandırıcısının İTU+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	258	2	1	5	1	19	8	1	1	4
1	0	218	6	17	6	10	5	5	17	16
2	2	5	199	21	7	25	3	19	18	1
3	3	1	10	199	0	26	7	9	19	26
4	1	2	2	2	237	9	31	1	3	12
5	6	9	25	19	22	123	11	22	33	30
6	8	3	5	2	34	5	236	0	7	0
7	0	11	6	10	5	13	1	227	10	17
8	9	3	9	19	6	25	6	12	187	24
9	2	7	2	42	9	44	1	20	28	145
% Başarım						%67				

Çizelge A.20: ACO_R-ÇKA sınıflandırıcısının İTU+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	246	1	7	6	5	15	2	0	17	1
1	2	231	8	13	0	7	2	16	8	13
2	6	5	235	10	3	22	3	6	9	1
3	1	18	18	218	2	24	4	4	2	9
4	12	8	2	0	216	17	19	7	7	12
5	3	9	14	43	12	139	2	6	36	36
6	5	8	7	2	27	7	225	1	18	0
7	0	21	5	5	0	12	1	226	5	25
8	3	12	2	22	1	52	6	2	184	16
9	0	15	5	30	4	40	0	9	14	183
% Başarım						%70				

Çizelge A.21: 3NN sınıflandırıcısının İTU+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	393	0	1	0	0	0	4	0	0	2
1	1	372	10	8	3	1	0	4	1	0
2	6	2	370	6	0	3	2	6	4	1
3	2	6	7	362	0	2	0	5	7	9
4	7	3	2	0	338	2	38	4	1	5
5	9	0	5	15	6	339	3	0	5	18
6	2	3	0	1	8	0	383	0	3	0
7	1	4	12	8	0	0	2	358	0	15
8	7	4	6	20	2	24	11	0	324	2
9	7	10	10	47	6	20	0	7	6	287
% Başarım						%88				

Çizelge A.22: 5NN sınıflandırıcısının İTU+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	391	0	1	0	0	0	6	0	0	2
1	0	373	13	5	2	3	0	0	2	2
2	3	0	380	3	0	2	0	6	6	0
3	0	4	1	371	0	4	0	1	13	6
4	0	5	3	1	340	3	32	5	3	8
5	7	0	2	21	5	332	3	0	8	22
6	1	1	0	1	6	1	386	0	4	0
7	0	12	8	12	0	0	2	356	1	9
8	0	6	4	12	2	24	16	4	324	8
9	3	9	7	50	4	17	0	15	6	289
% Başarım						%88				

Çizelge A.23: BP-ÇKA sınıflandırıcısının İTU+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	371	0	0	1	2	17	8	0	1	0
1	0	355	13	8	6	0	12	0	4	2
2	9	12	319	12	0	13	3	11	20	1
3	9	42	5	313	0	6	7	5	10	3
4	3	12	0	0	308	1	50	17	5	4
5	15	5	4	47	16	275	9	7	6	16
6	3	18	3	1	69	1	301	1	3	0
7	2	15	4	14	2	4	0	330	9	20
8	6	33	9	8	6	20	21	8	279	10
9	9	21	1	92	16	34	4	19	29	175
% Başarım						%75				

Çizelge A.24: Get-ÇKA sınıflandırıcısının İTU+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	341	4	6	5	11	4	6	2	1	20
1	2	304	25	12	12	2	14	7	8	14
2	8	25	218	22	29	12	42	10	28	6
3	5	27	24	216	11	31	9	18	16	43
4	12	14	4	7	264	3	31	5	9	51
5	7	8	8	46	12	218	11	12	33	45
6	14	7	4	6	31	1	318	0	16	3
7	0	31	9	6	12	9	1	309	10	13
8	11	33	37	48	19	26	36	17	160	13
9	3	24	4	50	22	43	1	24	19	210
%Başarım						%63				

Çizelge A.25: ACO_R-ÇKA sınıflandırıcısının İTU+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	301	5	22	1	9	23	26	0	11	2
1	1	264	5	23	14	8	9	13	51	12
2	11	2	319	9	8	21	3	5	17	5
3	4	17	17	241	1	30	1	8	34	47
4	23	13	11	10	235	8	49	14	17	20
5	14	3	31	25	8	206	5	11	37	60
6	15	3	6	5	66	6	277	0	21	1
7	0	9	14	27	6	10	1	301	12	20
8	2	25	15	25	26	49	10	10	207	31
9	1	10	18	26	15	41	0		44	227
% Başarım						%72				

Çizelge A.26: 3-NN sınıflandırıcısının NIST+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	285	3	0	0	0	0	8	0	4	0
1	0	297	2	0	0	0	1	0	0	0
2	8	16	251	3	0	0	2	14	5	1
3	1	14	4	254	0	4	0	4	18	1
4	3	14	1	1	244	0	4	1	2	30
5	0	19	1	25	2	231	2	0	13	7
6	5	8	0	0	1	2	284	0	0	0
7	0	10	3	1	3	0	0	246	4	33
8	7	32	3	9	8	6	1	4	221	9
9	1	6	2	5	12	1	0	5	3	265
% Başarım						% 85				

Çizelge A.27: 5-NN sınıflandırıcısının NIST+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	291	0	0	0	2	0	5	0	2	0
1	0	295	2	0	0	0	0	0	3	0
2	3	16	234	10	0	2	5	22	6	2
3	1	10	2	261	0	3	0	2	15	6
4	1	16	0	0	227	0	2	0	1	53
5	4	7	0	21	1	236	6	2	17	6
6	1	17	0	1	0	6	274	0	1	0
7	0	11	1	1	2	0	0	249	2	34
8	3	6	1	10	2	11	1	3	252	11
9	1	9	0	2	15	2	1	13	4	253
% Başarım						% 85				

Çizelge A.28: BP-ÇKA sınıflandırıcısının NIST+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	279	0	0	5	0	4	11	1	0	0
1	4	96	3	55	9	3	3	91	36	0
2	1	1	240	9	1	3	24	10	10	1
3	4	0	21	223	0	17	9	12	3	11
4	0	2	2	0	215	7	9	2	11	52
5	33	5	4	27	1	188	13	1	13	15
6	3	2	3	1	1	8	281	0	1	0
7	2	5	5	5	4	0	0	231	12	36
8	6	3	18	16	29	30	18	17	151	12
9	1	3	1	7	44	1	1	21	14	207
% Başarım						% 70				

Çizelge A.29: Get-ÇKA sınıflandırıcısının NIST+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	240	0	5	10	5	13	12	1	13	1
1	0	220	12	8	20	10	8	7	6	9
2	8	5	180	25	24	18	4	27	3	6
3	18	8	27	138	9	40	6	25	14	15
4	5	5	8	10	190	7	4	8	17	46
5	29	3	8	28	3	170	9	8	30	12
6	20	10	17	15	12	9	201	6	11	0
7	12	11	14	13	10	13	0	185	12	30
8	13	6	5	17	8	31	1	17	181	21
9	6	5	1	7	22	15	1	19	29	195
%Başarım						%63				

Çizelge A.30: ACO_R-ÇKA sınıflandırıcısının NIST+KLD veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	240	4	3	3	4	6	24	6	10	0
1	4	234	8	5	4	3	20	5	15	2
2	3	6	204	6	4	4	25	12	34	2
3	7	4	24	198	9	13	11	6	24	4
4	7	10	4	2	196	5	13	9	13	41
5	14	7	15	41	9	139	15	9	33	18
6	28	7	16	0	4	8	231	1	5	0
7	3	7	17	5	6	5	5	168	20	64
8	15	11	31	19	17	32	22	19	120	14
9	6	1	10	4	18	10	0	47	9	195
% Başarım						%64				

Çizelge A.31: 3-NN sınıflandırıcısının NIST+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	292	0	2	2	0	0	2	0	2	0
1	0	282	3	7	0	2	1	0	5	0
2	18	2	240	15	0	4	12	4	5	0
3	1	3	25	231	0	14	2	7	15	2
4	2	0	1	4	235	6	10	1	0	41
5	7	2	22	19	7	223	2	0	16	2
6	1	5	7	2	2	6	273	0	4	0
7	0	1	3	2	4	5	0	246	5	34
8	2	8	5	40	1	30	4	3	206	1
9	2	2	0	5	8	2	0	18	1	262
% Başarım						%83				

Çizelge A.32: 5-NN sınıflandırıcısının NIST+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	293	1	0	1	0	1	0	2	2	0
1	0	278	4	5	0	3	1	0	8	1
2	14	0	246	20	1	6	7	4	2	0
3	4	4	13	242	1	14	0	6	15	1
4	1	3	1	2	219	5	8	1	10	50
5	6	0	15	26	0	226	3	1	22	1
6	4	2	3	1	3	6	278	0	3	0
7	2	3	3	5	4	4	0	260	4	15
8	8	2	2	34	2	31	0	5	210	6
9	3	1	0	4	9	4	0	16	2	261
% Başarım						%83				

Çizelge A.33: BP-ÇKA sınıflandırıcısının NIST+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	284	0	1	0	0	2	11	1	0	1
1	2	276	2	0	1	4	11	2	0	2
2	8	2	249	0	0	5	26	6	1	3
3	21	59	121	0	1	3	6	33	3	53
4	1	2	0	0	259	1	9	0	0	28
5	7	13	64	0	15	143	23	2	0	33
6	2	7	1	0	2	1	287	0	0	0
7	2	12	2	0	3	5	0	238	2	36
8	10	100	20	0	10	95	12	19	4	30
9	4	8	0	0	20	1	0	19	1	247
% Başarım						%66				

Çizelge A.34: Get-ÇKA sınıflandırıcısının NIST+Şekilsel veri kümesi üzerindeki Başarımı

	0	1	2	3	4	5	6	7	8	9
0	247	1	3	4	4	24	4	2	9	2
1	2	227	11	11	2	14	2	2	26	3
2	10	6	197	28	0	20	18	5	16	0
3	4	8	43	181	9	22	0	8	18	7
4	12	7	1	7	190	11	7	5	15	45
5	17	6	16	28	1	176	7	2	36	11
6	14	10	8	5	3	18	236	0	6	0
7	1	6	3	20	7	7	0	225	12	19
8	12	24	21	35	3	57	2	14	117	15
9	9	2	1	12	15	13	0	36	23	189
%Başarım						%66				

Çizelge A.35: ACO_R-ÇKA sınıflandırıcısının NIST+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	256	0	14	3	0	9	3	1	11	3
1	1	206	4	20	4	10	7	12	34	2
2	14	3	197	26	0	17	20	11	12	0
3	14	16	21	153	9	29	3	14	36	5
4	2	5	1	9	214	10	10	9	12	28
5	9	4	27	28	16	151	12	4	43	6
6	1	4	21	1	9	17	236	1	10	0
7	1	2	0	12	14	4	1	234	10	22
8	5	13	5	15	18	22	11	16	178	17
9	0	0	0	18	34	5	0	26	32	185
% Başarım						%67				

Çizelge A.36: 3-NN sınıflandırıcısının NIST+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	295	2	0	0	0	1	1	0	1	0
1	0	291	2	0	0	1	1	3	0	2
2	3	2	266	5	0	1	6	8	9	0
3	1	5	6	264	0	6	0	1	12	5
4	4	4	2	1	245	0	6	1	1	36
5	2	2	0	29	0	248	5	0	12	2
6	1	11	0	2	0	5	281	0	0	0
7	1	6	2	1	5	0	0	252	0	33
8	4	10	1	22	0	9	1	6	235	12
9	1	2	0	2	13	1	1	7	4	269
% Başarım						% 88				

Çizelge A.37: 5-NN sınıflandırıcısının NIST+KLD+Şekilsel veri kümesi üzerindeki Başarımı

	0	1	2	3	4	5	6	7	8	9
0	291	1	0	0	0	3	2	0	3	0
1	0	287	3	1	0	1	2	2	3	1
2	3	1	264	3	1	0	4	11	12	1
3	0	5	4	271	0	7	0	0	10	3
4	2	4	0	1	240	0	9	0	4	40
5	1	2	0	30	2	242	5	1	14	3
6	1	1	1	0	0	6	290	0	1	0
7	0	4	4	6	3	0	0	245	0	38
8	0	11	1	18	4	8	1	3	242	12
9	2	5	1	3	6	2	0	7	3	271
% Başarım						% 88				

Çizelge A.38: BP-ÇKA sınıflandırıcısının NIST+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	291	0	0	2	1	1	2	0	3	0
1	0	264	4	10	1	6	5	0	1	9
2	3	5	251	9	2	5	7	8	9	1
3	1	3	9	254	0	5	1	9	11	7
4	2	3	4	0	252	8	4	0	5	22
5	11	2	9	63	10	167	20	7	2	9
6	3	7	2	1	1	4	281	0	1	0
7	0	4	1	4	2	4	0	253	2	30
8	9	27	4	30	15	34	9	5	154	13
9	4	5	0	7	18	5	0	14	4	243
% Başarım						% 80				

Çizelge A.39: Get-ÇKA sınıflandırıcısının NIST+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	201	1	16	16	0	21	24	0	18	3
1	0	197	15	12	11	19	7	10	18	11
2	7	8	182	16	1	16	35	13	19	3
3	2	2	39	160	2	36	4	11	33	11
4	1	8	1	9	196	19	15	3	5	43
5	5	0	20	12	3	212	13	5	16	14
6	7	4	16	5	4	11	243	0	9	1
7	0	10	8	9	1	9	0	222	3	38
8	3	7	32	33	3	57	2	12	127	24
9	0	9	7	16	22	35	0	37	27	147
% Başarım						% 63				

Çizelge A.40: ACO_R-ÇKA sınıflandırıcısının NIST+KLD+Şekilsel veri kümesi üzerindeki başarımı

	0	1	2	3	4	5	6	7	8	9
0	229	4	3	6	12	14	17	5	9	1
1	1	215	11	14	2	17	7	3	24	6
2	8	2	200	23	0	29	20	6	12	0
3	6	14	27	187	4	20	5	12	21	4
4	3	6	1	9	194	24	17	6	23	17
5	8	9	31	26	9	149	14	9	29	16
6	3	3	20	0	4	10	256	0	3	1
7	1	5	13	11	6	7	0	229	9	19
8	6	22	27	44	18	37	13	6	114	13
9	4	7	2	19	33	8	0	25	6	196
% Başarım						% 65				

ÖZGEÇMİŞ

Ad Soyad: Hale Hilal DODURGALI

Doğum Yeri ve Tarihi: İSTANBUL – 05.02.1983

Adres: Bulgurlu Mah. Gürpınar Sit. 2. Ksm B/5 B D/17 Üsküdar/İSTANBUL

Lisans Üniversite: İstanbul Teknik Üniversitesi

Yayın Listesi: