T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE BİTKİLERİN ŞEKİLLERİNİN SINIFLANDIRILMASI (YAPRAKTAN BİTKİ TANIMA)

Erdem BİLGİLİ

YÜKSEK LİSANS TEZİ

ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

84063

TEZ DANIŞMANI

Doç.Dr. Osman Nuri UÇAN

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

GEBZE

1999

Bu tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun .13/09/1999.. tarih ve .99../..22.. sayılı kararıyla oluşturulan jüri tarafından ..Elk...Mühendisliği.... Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (Tez Danışmanı) : Doç. Dr. Osman Nuri UÇAN

ÜYE : Yrd.Doç.Dr. Ertuğrul KARAÇUHA

ÜYE : Yrd. Doç. Dr. Murat ESİN

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun .12/09/1999. tarih ve ...99../..28.. sayılı kararı.

İmza/Mühür

ÖZET

Bu çalışmada, yapay sinir ağı teknikleri kullanarak bitki yapraklarına ait görüntüler sınıflandırılmıştır. Farklı beş türe ait yaprak görüntüsü ile yapay sinir ağı eğitilmiştir. Bu beş sınıf ve bunlardan farklı beş sınıfa ait toplam on türe ait yaprak görüntüleri test edilmiştir.

Sınıflandırma işleminde geriye yayılma algoritması ve hopfield ağ modeli kullanılmıştır. Bu iki yöntemin performansları karşılaştırılarak, geriye yayılma algoritmasının daha verimli olduğu gözlenmiştir.

SUMMARY

In this study, images of foliages were classified by using artificial neural networks techniques. Artificial neural networks was trained by image of model foliages of five different foliage classes. These five types and the other five types which are different were tested.

During the classification process, back propagation algorithm and hopfield net model was used. It is observed that performance of back propagation nets were highly better than the performance of hopfield nets.

TEŞEKKÜR

Bu çalışma sırasında katkılarını, desteklerini ve ilgilerini esirgemeyen danışmanım Sayın hocam, Doç.Dr. Osman Nuri UÇAN'a, Sayın hocam, Y.Doç.Dr.Ertuğrul KARAÇUHA'ya ve Sayın hocam, Y.Doç.Dr. Ragıp BAŞBUĞ'a şükranlarımı sunarım.

Benden manevi desteklerini esirgemeyen aileme, gösterdikleri anlayıştan dolayı teşekkürlerimi sunarım.

İÇİNDEKİLER

ÖZET.....IV
SUMMARY.....V
TEŞEKKÜR.....VI
İÇİNDEKİLER DİZİNİ.....VII
SİMGELER VE KISALTMALAR DİZİNİ.....IX
ŞEKİLLER DİZİNİ.....X
1.GİRİŞ.....1
1.1.Çalışmanın Amacı.....2
2.YAPAY SİNİR AĞLARI.....4
2.1.Yapay Sinir Ağlarının Üstünlükleri.....8
2.2. Basit YSA Modelleri ve Öğrenme Kuralları.....9
2.3.Eğiticili Öğrenme Yöntemleri.....18
2.3.1.Tek Tabakalı Perceptron Ağları.....19
2.3.2.Perceptron Öğrenme Kuralları.....19
2.3.3.Adaline Öğrenme Kuralları.....20
2.4.Geriye Yayılma (Back Propagation).....22
2.4.1.BP Algoritması.....26
2.4.2.Geriye Yayılma Algoritmasında Öğrenme Faktörleri.....28
2.5.Hopfield Ağları.....34
2.5.1.Ayrık Hopfield Ağları.....34
2.5.2.Sürekli Hopfield Ağları.....36
2.5.3.Çağrışımlı Bellekler.....36
3.YAPRAK DESENLERİNİN SINIFLANDIRILMASI.....39
3.1.Desen Sınıflandırma.....39
3.1.1.Bayes Sınıflandırma Yöntemi.....40
3.2.Yaprak Görüntülerinin Özniteliklerinin Çıkarılması.....41
3.2.1.Kenar Tanıma Yöntemleri.....42
3.2.1.1.Türev Yöntemi.....43
3.2.1.2.Gradient Yöntemi.....43
3.2.2.Zemin-Yaprak Sınırının Bulunması.....45
3.2.3.Sol ve Sağ El Yöntemleri.....47

3.2.3.1.Sol El Yöntemi.....47
3.2.3.2.Sağ El Yöntemi.....51
3.2.4.Yaprak Alanının Bulunması.....51
3.3.Simetri Eksenleri.....53
3.4.Özniteliklerin Çıkarılması.....55
3.5.Sınıflandırma İşlemi.....56
3.5.1.Geriye Yayılma Ağı İle Sınıflandırma..... 56
3.5.2.Hopfield Ağı İle Sınıflandırma.....65
4.SONUÇ.....68
KAYNAKLAR.....69
ÖZGEÇMİŞ.....71
EKLER.....72

SİMGELER VE KISALTMALAR DİZİNİ

$a(f)$: Aktivasyon fonksiyonu.

$d^{(k)}$: k. sınıfa ait hedef deseni.

E : Hata fonksiyonu.

$\nabla E(w)$: Hata fonksiyonun gradienti.

$G(i,j)$: Parlaklık fonksiyonu.

$\nabla G(i,j)$: Parlaklık fonksiyonunun gradienti.

$H(w)$: Hessian matrisi.

$h(i,j)$: Maske filtresi.

net_i :Hücrenin toplam giriş işareti büyüklüğü.

p_i : i. sınıfa ait olma olasılığı.

P_i : i. sınıfa ait olma koşullu olasılığı.

P_K : k. sıradaki piksel.

$q_i(x)$: x deseninin i.sınıfa ait olma ihtimali.

r^* : Risk faktörü.

r : Hata işareti.

x_i : Hücrenin i.girişi.

$x^{(k)}$: k. sınıfa ait giriş deseni.

w : Ağırlık matrisi.

Δw : Ağırlık matrisindeki değişme miktarı.

Y_K: k. piksele göre yön bilgisi.

y: Hücre çıkış işareti.

z_q : Saklı tabaka çıkışı (q. hücre).

α : Momentum faktörü.

θ_i : Hücrenin eşik (bias) değeri.

η : Öğrenme sabiti.

δ_{oi} : Çıkış katmanı hata işareti.

δ_{hq} : Saklı tabaka hata işareti.

BP: Back Propagation.

YSA: Yapay sinir ağları.

ŞEKİLLER DİZİNİ

Şekil Sayfa

2.1. Perceptron modeli.....6

2.2. Farklı ağ bağlantıları.....13

2.3. Çok katmanlı ağ yapısı.....14

2.4. Eğiticili öğrenme.....16

3.1. Değişik eşik değerlerinde yaprak kenar görüntüleri.....46

3.2. P_{K-1} Piksele komşu piksellerin yön bilgileri.....48

3.3. Y_K=1 Durumu.....50

3.4. Simetri eksenlerine göre döndürülmüş yaprak görüntüsü.....55

3.5. (3) Numaralı karakavak yaprağına ait öznitelik grafiği.....66

3.6. (1) Numaralı karaağaç yaprağına ait öznitelik grafiği.....66

3.7. (2) Numaralı akasya yaprağına ait öznitelik grafiği.....67

3.8. (1) Numaralı karaağaç yaprağının test edilmesi ile çıkışta görülen desen.....67

1. GİRİŞ

Yapay sinir ağları,günümüzde görüntü işleme ve görüntü tanıma konuları da dahil olmak üzere pek çok konuda hızla büyüyen bir ilgi alanı oluşturmaktadır. 1940'lı yılların başında Mc Culloch, Pits,Hebb ve Rosenblatt' ın çabalarıyla alışılagelmiş bilgi işleme sistemlerine bir seçenek olarak ilk yapay sinir ağı modelleri geliştirilmeye başlanmıştır [Cichocki A.,1993]. Özellikle 80'li yıllarda bu konuda ki çalışmalar hızla artış göstermiş, Hopfield,Backpropagation gibi ağ modellerinin problem çözmedeki başarıları, çeşitli bilim disiplinlerinde çalışan bilim adamlarının yapay sinir ağlarına özel ilgi göstermelerine sebep olmuştur.

Desen tanıma (pattern recognition) ve sınıflandırma işlemlerinde ilk yapılması gereken adım, görüntüye ait bir öznitelik vektörünün çıkarılmasıdır. Bundan sonraki aşama sınıflandırma aşamasıdır.Öznitelik çıkartma işlemi için çok değişik fonksiyonlar kullanılmaktadır. Bu fonksiyonların seçimi problemin yapısına bağlıdır ve doğal olarak her uygulamada farklılık göstermektedir.Burada dikkat edilmesi gerekennokta, kullanılan öznitelik çıkartma yönteminin, mümkün olduğu kadar sınıfları birbirinden ayıracak ve uzaklaştıracak olmasıdır. Öznitelik çıkartma işlemi çok sayıda matematiksel işlem yapılmasını gerektirdiği için uzun zaman almaktadır. Hatta süper bilgisayarlarda bile robot gözünden alınan görüntülerin sınıflandırılması işlemi çok uzun zaman aldığı gözlenmiştir [Karamahmut S.,1994].

Canlıların görüntü tanıma konusundaki yetenekleri, kısa bir zamanda görüntüyü algılayıp tepki vermeleri, bu konuda çalışan bilim çevrelerini beyin ve sinir sistemini incelemeye yöneltmiştir. Çalışmalar sonucunda sinir sisteminin bilgiyi paralel işlediği tesbit edilmiştir. Paralel işlemenin getirdiği çok önemli iki avantaj vardır.Birincisi bir problemin çözümünde çok sayıda hücrenin görev alması sonucu bilginin hızlı işlenmesidir.İkincisi ise problemi çözmekte görevli olan hücrelerin birkaçının görevini yapmaması veya ölmesi sonucu değiştirmemekte, sistem performansı düşse bile tatminkar sonuçlar alınmaktadır. Sinir sisteminin bir nevi simülasyonu olan yapay sinir ağı modellerinde eğitim ise, tıpkı canlıların eğitimine benzemektedir.Örnek olarak eğiticili öğrenme de ağ girişine giriş verileri verilerek,

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

bu veri kümesine karşılık ağın çıkışının alması gereken çıkış değerleri öğretilir.Sinir ağı daha sonraki aşamalarda benzer giriş verilerine karşılık daha önce öğrendiği çıkış değerini üretecektir.

1.1. Çalışmanın Amacı

Bu çalışmada bitkilerin yaprak görüntüleri vasıtasıyla sınıflandırılması gerçekleştirilmiştir.Tarımsal savaş uygulamalarında yapılan ilaçlamalarda zararlı bitkilerin yanında yararlı bitki de zarar görmektedir.Bunu en aza indirebilmek için zararlı bitkilerin yoğun olarak bulunduğu bölgeler ilaçlanırken, yararlı bitkinin yoğun olarak bulunduğu bölgelere daha az ilaç püskürten veya bitki yoğunluğuna bağlı olarak hiç ilaçlama yapmayan ilaçlama sistemleri geliştirilebilir. Bunun için bitki kimliğini tayin eden yaprak görüntülerinden faydalanılması mümkündür.Bir başka uygulama sektörü ise gıda sektörüdür. Bitkilerin yaprak görüntülerine bakılarak kalitesine göre sınıflandırma işlemleri yapılması mümkündür [Guyer,et al,1986]. Son yıllarda meyve ve sebze sınıflandırmasını otomatik olarak yapabilen sistemlerin oluşturulmasına yönelik çalışmalar hızla artmaktadır. Fındıkların olgunluklarına göre sınıflandırılmasında T.C.Pearson görüntünün kırmızı, mavi ve yeşil kopyalarından yararlandığını belirtmektedir[Pearson, et al,1996]. S. Humphries sardunya yapraklarında yaprak ve sap bölgesinin ayrılması için bir yöntem geliştirmiştir [Humphries, Simonton,1993]. Ayrıca bitki tohumlarının sınıflandırılmasına yönelik çalışmalar da yapılmaktadır [Liao, et al, 1993] . Yaprak desenlerinin sınıflandırılması alanında E.Franz, yaprağa ait genel morfolojik özellikleri çıkarmıştır[Franz, et al,1995]. Yine tohumlardaki morfolojik parametrelere ilişkin Shatadal P., tohum alanını, tohum boyunu, tohum çapını ve benzeri parametreleri kullanmıştır [Shatadal, et al, 1994].

Yapay sinir ağları (YSA) , paralel çalışmanın getirdiği hız avantajı, yazılım ve donanımla kolay gerçeklenebilir olması, hücrelerdeki bilgi kaybının veya hücre ölümlerinin sistem cevabını önemli ölçüde etkilememesi özellikleri ile sınıflandırma uygulamalarında kullanılan giriş verilerine bir esneklik kazandırmaktadır. Giriş verilerindeki gürültüler ve bozulmalar YSA tarafından elimine edilebilmektedir.

Ancak ağın öğrenme algoritmasından kaynaklanan yerel optimumlara takılması ağın performansını düşürmektedir.

Yapay sinir ağları ile yapılan bazı desen tanıma çalışmalarında giriş verileri ağa direk olarak verilmezler.Veriler önce öznitelik işlemini gerçekleştiren bir bloktan geçirilerek öz nitelik vektörü çıkarılır.

2. YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA) girişine verilen bilgiyi öğrenme,daha önce öğrendiği bilgiyi hatırlama, ve eğitim deseni veya veri kümelerini giriş olarak kabul edip bu bilgilerden yararlanarak genelleştirme yapabilme, yorum yapabilme yeteneği gösteren ümit vaat eden yeni kuşak bır bilgi işleme sistemidir.YSA bilinçli olarak insan beyninin çalışma sistemini taklit etmek maksadıyla geliştirilen bir tekniktir. Desen tanıma, sınıflandırma, fonksiyon kestirimi, optimizasyon işlemleri ve verilerin kümelendirilmesi, gruplandırılması işlemlerinin gerçekleştirilmesinde çok iyi bir performans sergilemektedir [Ölmez T.,1995]. Biyolojik sistemler, sayısal bilgisayarlar karşısında aritmetik işlemlerde kıyaslanmayacak kadar güçsüz olmalarına karşın, görüntü ve ses işleme gibi bozulmuş, tam tanımlı olmayan yüksek veri yoğunluğuna sahip bilgilerin işlenmesinde sayısal bilgisayarlara göre çok hızlı ve yüksek doğrulukta çalışırlar.Bilgisayarların ses ve görüntü gibi işaretlerin işlenmesinde yapısal olarak güçsüz olduğunun görülmesi 80'li yıllarda çeşitli bilim dallarından büyük bir araştımacı kitlesinin YSA na büyük ilgi duymalarına sebep olmuştur.

1943 yılında McCulloch ve Pits biyolojik sinir hücresinden esinlenerek geliştirdikleri bir yapay sinir hücre modeli ile VE, VEYA gibi mantık işlemlerinin yapılabileceğini gösterdiler. Böylece canlıların sinir sistemlerinin incelenmesi ve bunlara benzer şekilde çalışan yapay sinir ağı modellerinin geliştirilmesi çok farklı bilim dalından araştırmacıların ortak olarak çalıştığı bir bilim dalı olmuştur.

YSA alanında yapılan araştırmalar 60'lı yılların ortalarından 80'li yılların başına kadar bir durgunluk dönemi yaşadı.Bunun en önemli sebebi bilgi işlemede YSA 'nın alternatifi olan bilgisayarların yarı ileken teknolojisindeki gelişmeler ile kapasiteli,ucuz ve güvenilir gerçeklenme olanağı bulmasıdır.Seri olarak çalışan hızlı birimlerden oluşmuş sayısal bilgisayarlar,aritmetik işlemlerde yüksek hız,kapasite ve güvenirlik sağlamışlardır.

YSA,birbirleriyle çok sayıda iç bağlantıları olan büyük miktarda hücre adı verilen işlem elemanlarından oluşur. Bilgi ağ içerisinde paralel işlenir. İnsan

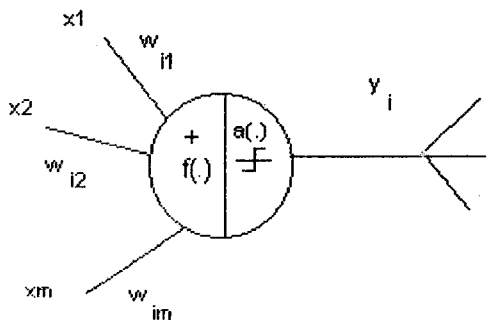
beyninde olduğu gibi YSA' nında etkileşimli ve kollektif çalışması, öğrenme ve öğrenilen bilginin hatırlanmasına, yorum yapmaya imkan tanır. YSA nın özelliklerinden bahsetmeden önce insan beyninin çalışma sistemine bir gözatmakta fayda vardır.İnsan beyninde yaklaşık 10^{11} civarında farklı yapılarda sinir hücresi bulunduğu bilinmektedir. Biyolojik sinir hücresi üç ana kısımdan oluşur. Bunlar;

1. Soma adı verilen hücre gövdesi ki hücre çekirdeğinin bulunduğu bölgedir.
2. Dendritler
3. Akson

Her bir sinir hücresine diğer hücrelerden bilgi ileten sinir tellerine dendrit adı verilir. Hücrenin akson adı verilen tek bir çıkışı vardır ve bu çıkıştan komşu sinir hücrelerine bilgi taşıyan binlerce dendrit bağlıdır. Aksona bağlı her bir dendritteki iletilen bilgi aynıdır.Yani bilginin iletim sırasında zayıflaması sözkonusu değildir. Hücre girişine bilgi taşıyan dendritler sinaps adı verilen bağlantı noktalarında hücreye bağlanırlar.

Dendritlerde iletilen ve sinapslara ulaşan sinyaller elektrik darbeleridir. Bağlantı noktası (sinaps) tarafından salınan özel iletim maddeleriyle gerçekleştirilen, karmaşık kimyasal işlemleri içeren, bir işaret iletim sistemi söz konusudur. Bu maddeler Na ve K iyonlarıdır. Bu maddelerin konsantrasyonun değişmesi ile hücre içi elektrik potansiyeli azalır veya artar. Alıcı hücrenin elektriksel potansiyeli, o hücreye ilişkin eşik değerini geçerse hücre ateşlenir, aktif hale geçer. Hücre çıkışındaki aksonda sabit şiddette ve sabit süreli bir elektrik darbesi oluşur. Bu işaret akson boyunca iletilir ve akson ucuna bağlı olan diğer hücrelere bilgi taşıyan dendritlere aktarılır. Hücreye yeni bir gelinceye kadar hücre dinlenir. Bu zamana **bekleme veya dinlenme zamanı 'refractory period'** denir. Sinapslar eğer alıcı hücrenin ateşlenmesine sebep olacak sinyalleri geçiriyorsa bu sinapslara **uyarıcı (excitory)** sinaps adı verilir.Eğer alıcı hücrenin ateşlenmesini engelleyici sinyallerin geçmesine izin veriyorlarsa bu tip sinapslara, engelleyici (inhibitory) sinaps adı verilir.

Şekil 2.1.' de yukarıda anılan özellikleri gerçekleştiren, Mc Culloch ve Pitts(1943) tarafından ileri sürülen ve genellikle M-P sinir hücresi adı verilen, biyolojik sinir hücresinin basit bir matematik modeli görülmektedir.



Şekil 2.1. Perceptron Modeli

Bu modelde ***i.*** işlem elemanı girişlerin ağırlıklı toplamını hesaplar ve bu toplam eşik değerinden büyükse çıkışta $y_i = 1$ olur yani hücre ateşlenir. Aksi halde $y_i = 0$ olur yani hücre ateşlenmez.

Burada aktivasyon fonksiyonu başka bir deyişle hücrenin transfer fonksiyonu ;

$$a(f) = \begin{cases} 1 & f \geq 0 \\ 0 & f < 0 \end{cases} \tag{2.1}$$

şeklinde tanımlı birim basamak fonksiyonudur.

W_{ij} ağırlığı, $j.$ nöronu $i.$ nörona bağlayan sinapsın uyarma şiddetini yada bağlantının ağırlık katsayısını gösterir. Pozitif ağırlıklar sinapsın uyarıcı olduğunu, negatif ağırlıklar uyarmayı engelleyici bir rol üstlendiğini gösterir. $W_{ij} = 0$ olması iki hücre arasında bir bağlantının olmadığını gösterir. Eşitlikte t ve $t+1$ anları arasında birim gecikme olduğu farzedilmiştir.

Mc Culloch ve Pitts modeli, uygun ağırlık katsayıları verilmesi durumunda AND, OR ve NOT işlemlerini gerçekleştirir. Çok değişkenli, kombinasyonel bir lojik fonksiyon, bu basit lojik işlemlerle gerçekleştirildiğinden, bu tip sinir hücrelerinin eşzamanlı çalışmasıyla geleneksel bilgisayarların yaptığı bütün işlemler YSA ile gerçeklenebilir.

Bütün bu özellikleri özet olarak aşağıdaki gibi sıralayarak YSA nın bir tanımını yapabiliriz.

1. Sinir sisteminden esinlenerek oluşturulmuş bir matematik modeldir.
2. Birbirleriyle bağlantıları olan çok sayıda işlem elemanından oluşur.
3. Bağlantıları (ağırlık katsayıları) bilgiyi tutar, hafızayı temsil eder.
4. İşlem elemanı girişindeki uyarıyı hemen yanıtlayabilir. Ve bu çıkış cevabı hücrenin yerel bilgisine bağlıdır.Başka bir deyişle hücrenin çıkış cevabı, gelen bilgiye ve hücredeki yerel bellekteki bilgiye(ağırlık katsayısı ve eşik değeri) bağlıdır
5. Yeni bağlantı ağırlıkları atayarak yada bağlantı katsayılarını değiştirerek eğitim verilerini genelleştirebilme,öğrenme ve hatırlama yeteneğine sahiptir.
6. Hücrelerin kollektif, birbirleriyle etkileşimli çalışması hesaplama gücünü artırır. Belirli bir bilgiyi taşıyan özel bir hücre yoktur. Bilgi bütün sisteme dağılmıştır.(Dağılmış yapı -distributed representation özelliği).

Bu özelliklerden dolayı YSA için *paralel,dağıtılmış bilgi işleme modeli, (paralel distributed processing models),bağlantılı model(connectionist model),kendi kendine öğrenen model(self-organizing systems),neurocomputing ve neuromorphic systems* adları da yaygın olarak kullanılır.

C. Güzeliş bugüne kadar verilen çeşitli YSA modellerinin ortak özelliklerini aşağıdaki gibi sıralamıştır.[Güzeliş C.,1993]

1. Hücreler, genelde çok girişli ve tek çıkışlı yüksek dereceden lineer olmayan alt devrelerdir.
2. Tek olan hücre çıkış işareti, çoğullanarak dışarıya çıkış olarak alınabilir veya diğer hücrelere giriş olacak şekilde (genelde gecikmeler ile) kullanılabilir.
3. Bir hücre; dış girişleri(gecikme ile ulaşmış olabilir), komşu hücre çıkışlarını, ve bir eşik değerini giriş olarak kabul eder.
4. Hücrenin lineer dinamik birimi herhangi bir dereceden olabilir.
5. Hücre çıkışında lineer olamayan cebrik işlev 1-giriş 1-çıkışlıdır ve özel olarak lineer olabilen herhangi bir işlevdir.
6. Hücre parametreleri hücreden hücreye değişebilir.
7. Ağ, genelde çok boyutlu katmanların ileri veya geribeslemeli olacak şekilde bağlanmasından oluşmuştur.
8. Ağdaki her bir katman(layer), hücrelerin çok boyutlu bir dizisidir.

9. Katman içinde, herbir hücre komşularına (belirli bir metrik uyarınca) aynı şekilde bağlıdır. Böylece katman içi bağlantı geometrisi düzgündür.
10. Katmanlar arası bağlantı bir metrik ile tanımlı düzgün bir geometriyr sahiptir.
11. Ağda yalnızca bağlantı ağırlıkları değişebilir.
12. Bağlantı ağırlıkları; ya eğiticili bir öğrenme kuralı ile yada eğiticisiz bir öğrenme kuralı değiştirilir yada önceden tasarlanan değerlerde sabit tutulur.
13. Eğiticili bir öğrenme kuralı (eğer varsa); giriş - (istenen) çıkış örnek çiftlerinin alındığı, bilinmeyen bir giriş işlevine bu örnekler yardımıyla yaklaşılmasını sağlayan bir bağlantı ağırlıkları değişim kuralıdır.
14. Eğiticisiz öğrenme kuralı(eğer varsa); giriş örneklerinin kümelendirilmesini sağlayan bir bağlantı ağırlıklarının değişim kuralıdır.

2.1. Yapay Sinir Ağlarının Üstünlükleri

YSA modelleri biyolojik sinir ağlarının çalışma biçimini açıklayan modeller olmaktan çok, biyolojik sinir ağlarından esinlenerek yaratılmıştır.YSA' lar, biyolojik olmayan yapı taşlarının düzgün bir mimari ile birbirine yoğun olarak bağlanmalarından oluşmaktadırlar.Sinir ağlarında bilgi işlemenin alternatif bilgi işleme yöntemlerine göre üstünlükleri maddeler halinde aşağıda verilmiştir.[Zurada, J.M.,1989].

Paralellik: Alışılagelmiş bilgi işlem yöntemlerinin çoğu seri işlemlerden oluşmaktadır. Bu da hız ve güvenirlik sorunlarını beraberinde getirmektedir. Seri bir işlem gerçeklenirken herhangi bir birimin yavaş oluşu tüm sistemi doğrudan yavaşlatırken, paralel bir sistemde yavaş bir birimin etkisi çok azdır. Nitekim,seri bir bilgisayarın bir işlem elemanı beyine göre binlerce kez daha hızlı işlem görmesine rağmen, beynin toplam işlem hızı seri bir bilgisayara göre kıyaslanmayacak kadar yüksektir. YSA' nın paralel çalışması hız avantajıyla birlikte yüksek hata toleransı da sağlamaktadır. Seri bilgi işlem yapan bir sistemde herhengi bir birimin hatalı çalışması, hatta bozulmuş olması tüm sistemin hatalı çalışmasına veya bozulmasına sebep olacaktır. Halbuki paralel bilgi işleme yapan bir sistemin ayrı ayrı işlem elemanlarından meydana gelecek olan hatalı çalışma veya hasar, sistemin performansında keskin bir düşüşe yol açmadan, performansın sadece hatalı birimlerin bir oranınca düşmesine sebep olur.

Yerel Bilgi İşleme: YSA' larda her bir bilgi işleme birimi, çözülecek problemin tümü ile ilgilenmek yerine, sadece problemin bir parçasını işlemektedir. Hücrelerin çok basit işlemler yapmalarına rağmen, sağlanan görev paylaşımı sayesinde, çok karmaşık ve zor problemler çözülebilmektedir.

Gerçeklenme Kolaylığı: YSA' ların basit işlemler gerçekleyen türdeş hücrelerden oluşması ve bağlantıların düzgün olması, bu ağların fiziksel gerçeklenmesinde bu açıdan kolaylık sağlamaktadır.

Öğrenebilirlik: Alışılagelmiş veri işleme yöntemelerinin çoğu programlama yolu ile hesaplamaya dayanmaktadır. Bu yöntemler ile tam tanımlşı olmayan bir problem çözülemez. Ayrıca herhangi bir problemin çözümü için probleme yönelik bir algoritma geliştirmek gerekir. Oysa YSA' larda örneğin eğiticili öğrenmede giriş ve çıkış çifleri verildikten sonra bunlar arasındaki ilişkiyi belirlemek üzere bir algoritma geliştirmeye gerek yoktur. İlgili bağıntıyı YSA bağlantı katsayılarını değiştirerek belirler. Zaten öğrenme dediğimiz işlemde başka bir şey değildir.YSA' da bilgi bağlantı katsayıları ile tutulur.

2.2. Basit YSA Modelleri ve Öğrenme Kuralları

Bir YSA modelinin karakteri üç temel kriterle belirlenir.

1. İşlem elemanları ,sinir modelleri
2. Sinaptik iç bağlantılar ve bağlantı yapıları
3. Bağlantı katsayılarını güncellemek için kullanılan eğitim ve öğrenme algoritmaları

İşlem Elemanları : Daha önce açıklanan M-P sinir modelinin çalışma şekli, gerçekleştirdiği fonksiyon diğer yapay sinir modellerini içine alacak şekilde genelleştirilebilir.Bir işlem elemanı girişler ve çıkışlar olmak üzere iki kısımdan oluşur. Hücre girişindeki bilgi diğer hücrelerden gelen bilgilerin ağırlıklı toplamıdır.Bu giriş işareti M-P modelinde olduğu gibi genelde X_j girişlerinin doğrusal bir fonksiyonudur.

$$f_i = net_i = \sum_{j=1}^{M} W_{ij} X_j(t) - \theta_i \tag{2.2}$$

Burada θ_i hücrenin eşik değeridir. Daha karmaşık toplama fonksiyonlarının olması da söz konusudur. Sık kullanılan fonksiyonlar şunlardır:

1. Karesel Fonksiyon (Quadratic function):

$$f_i = \sum_{j=1}^{M} W_{ij} X_j^2 - \theta_i \tag{2.3}$$

Karesel fonksiyonun kullanılması durumunda,giriş işareti komşu sinir hücrelerinden gelen işaretlerin karelerinin ağırlıklı toplamından elde edilir.

2. Silindirik Fonksiyon (Spherical function)

$$f_i = \rho^{-2} \sum_{j=1}^{M} (X_j - W_{ij})^2 - \theta_i \tag{2.4}$$

Burada ρ ,silindirin yarıçapı, w_{ij} ise merkezini göstermektedir.

3. Polinom fonksiyon(Polinomial Function)

$$f_i = \sum_{j=1}^{m} \sum_{k=1}^{m} w_{ij} x_j x_k + x_j^{\alpha} + x_k^{\beta} - \theta_i \tag{2.5}$$

Burada w_{ijk} , j. ve k. hücreyi , i. hücreye bağlayan bağlantının ağırlık katsayısı, α ve β reel sabitlerdir. Bu eşitlikte daha yüksek dereceden terimlerin olması da mümkündür. Bu fonksiyonu gerçekleştiren hücreye *sigma-pi* $(\Sigma\Pi)$ birimi adı verilir.

İşlem elemanının (hücrenin) ikinci bir işlevi, giriş işaretini transfer fonksiyonundan diğer adıyla aktivasyon fonksiyonundan $a(f)$ geçirerek, girişin fonsiyonu olan bir çıkış değeri üretmektir. Sıkça kullanılan transfer fonksiyonları ise aşağıdaki fonksiyonlardır.

1. Basamak fonksiyonu (Step function)

$$a(f)=\begin{cases}1 & f\geq 0\\ 0 & f<0\end{cases} \tag{2.6.}$$

2. Keskin sınırlayıcı fonksiyonu (Hard limiter, threshold function)

$$a(f)=\operatorname{sgn}(f)=\begin{cases}1 & f\geq 0\\ -1 & f<0\end{cases} \tag{2.7}$$

Burada sgn(•) , signum fonksiyonudur.

3. Rampa fonksiyonu (Ramp function)

$$a(f)=\begin{cases}1 & f>1\\ f & 0\leq f<1\\ 0 & f<0\end{cases} \tag{2.8.}$$

3. Unipolar sigmoid fonsiyonu

$$a(f)=\frac{1}{1+e^{-\lambda f}} \tag{2.9}$$

4. Bipolar sigmoid fonksiyonu

$$a(f)=\frac{2}{1+e^{-\lambda f}}-1 \tag{2.10}$$

Eşitlik (2.9) ve eşitlik (2.10) da $\lambda>0$ olduğunda, $f=0$ civarında, $a(f)$ sürekli fonksiyonunun adım aralığını belirler. $\lambda\to\infty$ olması durumunda eşitlik (2.9), (2.6) eşitliğine; yani unipolar sigmoid fonksiyonu ,basamak fonksiyonuna, eşitlik (2.10) ise (2.7) eşitliğine; yani bipolar sigmoid fonksiyonu, keskin sınırlayıcı fonksiyonuna dönüşür. Keskin sınırlayıcı transfer fonksiyonuna ve lineer toplayıcı birimine sahip olan bir işlem elemanına, doğrusal eşik birimi (linear threshold unit,LTU), sigmoid transfer fonksiyonuna ve doğrusal toplayıcıya sahip işlem

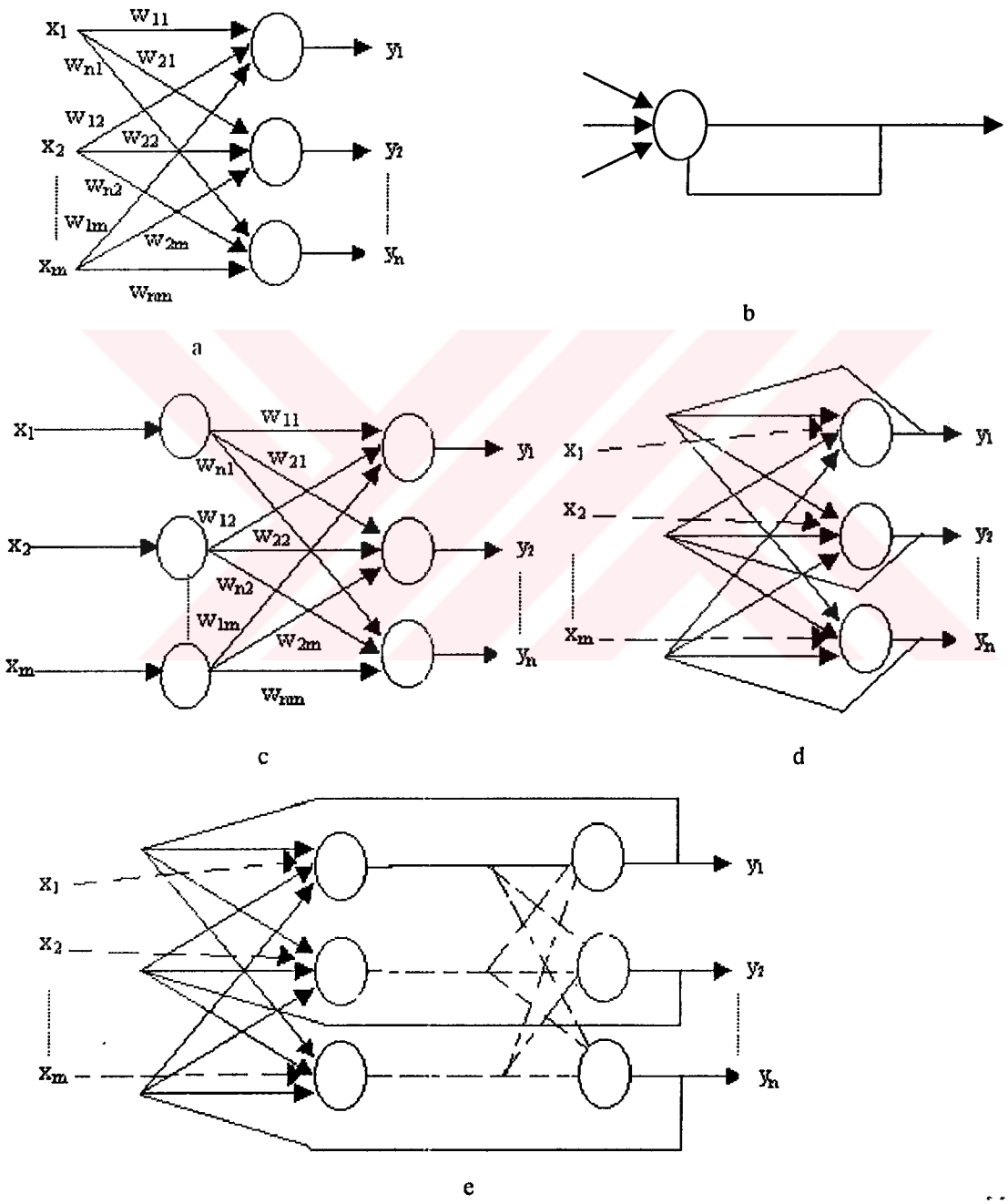
elemanına doğrusal eğimli birim (linear graded unit, LGU) adı verilir.LTU ve LGU çok yaygın kullanılan işlem elemanlarıdır.

Girişlerin toplanmasında doğrusal olmayan fonksiyonların kullanılması, karar sınırları adı verilen sınırlarla öznitelik uzayının karmaşık bölgeler şeklinde belirlenmesine imkan tanır. Bununla birlikte, toplama fonksiyonu karmaşıklaştıkça ve güçlendikçe, ağ yapısının da karmaşık olması kaçınılmazdır. Nasıl bir transfer fonksiyonu kullanılması gerektiği ve toplama fonksiyonun lineer olup olmaması, tamamen problemin yapısına bağlıdır.

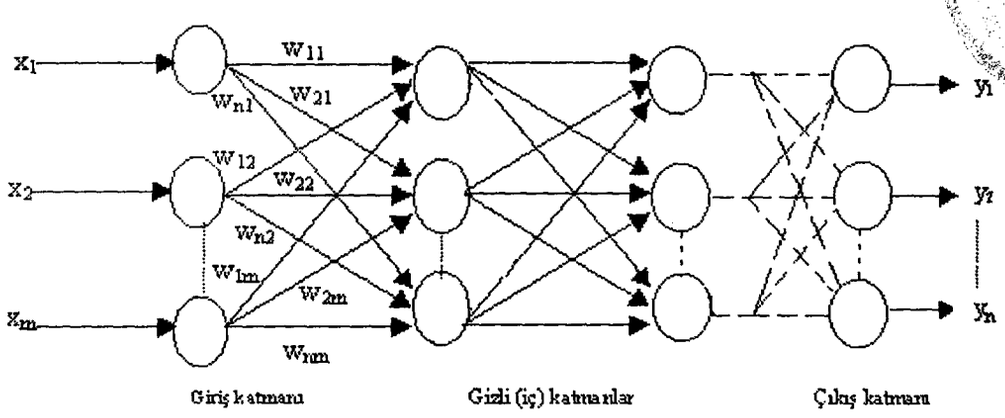
Bağlantılar: Bir YSA çok yoğun bağlantıları olan işlem elemanları (hücreler) kümesinden oluşur. Hücreler birbirlerine veya kendilerine bağlantı ağırlıkları üzerinden bağlıdırlar. Bağlantı yapısı gecikme ve geri kalmaya imkan tanır. Bu nedenle ağın bağlantı yapısının ve hücreler arası bağlantı geometrisinin, çok büyük bir önemi vardır. İşlem elmanın transfer fonksiyonunun ve toplama birimi türünün yanında bağlatıların nerede başladığı ve nerede sonlandığı, bağlantı biçimi, ağın karakterini belirlemektedir. Mümkün beş farklı ağ geometrisi söz konusudur.Ağ yapısı Şekil (2.2.a)' da görüldüğü gibi bir katmandan oluşuyorsa, bu tip bir ağa tek tabakalı ileri beslemeli ağ adı verilir. Girişler işlem elemanlarına bağlantı katsayıları üzerinden bağlıdır. Görüldüğü gibi her bir hücrenin bir tek çıkışı vardır. Bu şekilde daha fazla katman arasında bağlantılar oluşturarak çok tabakalı ileri beslemeli ağ elde edilir. Girişlerin bağlı olduğu tabakaya giriş katmanı adı verilir.Giriş tabakası veriler üzerinde herhangi bir işlemde bulunmaz. Sadece verilerin ağa girilmesinde arayüz görevi yapar. Ağın çıkışları çıkış tabakası adı verilen katmandan alınır. Giriş ve çıkış katmanları arasında bir veya birden fazla katman bulunabilir. Bu katmanlara saklı, gizli katman adı verilir. Bu katmandaki verilere doğrudan erişme imkanı yoktur. Bir katmandaki her bir hücrenin bir sonraki katmandaki bütün hücrelerle bağlantısı varsa bu tip bir ağa tam bağlantılı(fully connected) ağ adı verilir. Aksi durumda bir katmandaki hücrelerden en az birinin, bir sonraki katmandaki hücrelerden biriyle veya birkaçıyla bağlantısı yoksa bu tip bir bağlantı tam bağlantılı değildir.

Eğer bir katmandaki hücrelerin çıkışları bir önceki katmandaki hücrelere giriş olarak bağlanmıyorsa, böyle bir ağa, ileri beslemeli ağ (feedforward networks) adı

verilir. Geriye doğru bağlantıların olması geri besleme anlamına geldiğinden böyle bir ağa ise, geribeslemeli ağ (recurrent networks) denilmektedir. Eğer bir katmandaki çıkış, aynı katmana giriş olarak geri dönüyorsa, bu tip geribeslemeye yanal geri besleme veya daha doğru bir tabir ile kendi üstüne geri besleme adı verilir.



Şekil 2.2. Farklı Ağ Bağlantıları



Şekil 2.3. Çok katmanlı ağ yapısı

Öğrenme Kuralları: Bir YSA' yı tanımlamada üçüncü önemli unsur öğrenme kuralıdır. Öğrenme kuralları iki ana gruba ayrılır. Bunlardan birincisi parametrik öğrenme dediğimiz ağ parametrelerin değişmesiyle gerçekleştirilen öğrenme yöntemleri, diğeri ise yapısal öğrenme denilen ağ yapısının değiştirilmesi ile geçekleştirilen öğrenme yöntemleridir. Yapısal öğrenmede, her bir eğitim aşamasında, ağa yeni hücreler katılır veya o an için mevcut olan, fakat ağ performansını etkilemeyen hücreler ağdan çıkarılır. Bir başka yöntem de, ağda yeni bağlantıların oluşturulması veya varolan bağlantıların kaldırılmasına dayanmaktadır. Bu işlemleri gerçekleştiren algoritmalar ise genetik algoritmalardır. Tez içerisinde parametrik öğrenme yöntemleri kullanıldığından ve genelde uygulamalarda parametrik öğrenme yöntemleri yaygın olduğundan, yapısal algoritmalar üzerinde daha fazla durulmayacaktır.

Parametrik öğrenme kurallarında değiştirilen ağ paramatereleri ağın bağlantı katsayılarıdır. Bağlantı katsayılarını içeren ağırlık matrisi aşağıdaki gibi tanımlanır.

$$W \triangleq \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix} \tag{2.11}$$

Öyleki $i = 1,2,\ldots,n$ olmak üzere, $w_i = (w_{i1}, w_{12}, \ldots, w_{i3})^T$, i. işlem elemanının

ağırlık vektörüdür. w_{ij}'ler ise j. hücrenin (kaynak), i. hücreye (hedef) olan bağlantısının ağırlık katsayısıdır. Öğrenme; girişlerle, çıkışlar arasında cebrik bir ilişki sağlayan, uygun ağırlık matrisinin bulunması işlemidir. Dolayısıyla öğrenme kuralı, ağırlık matrisinin, her eğitim aşamasında güncellenmesini sağlayan bir algoritmadır.Parametrik öğrenme kurallarını üç gruba ayırmak mümkündür. Bunlar:

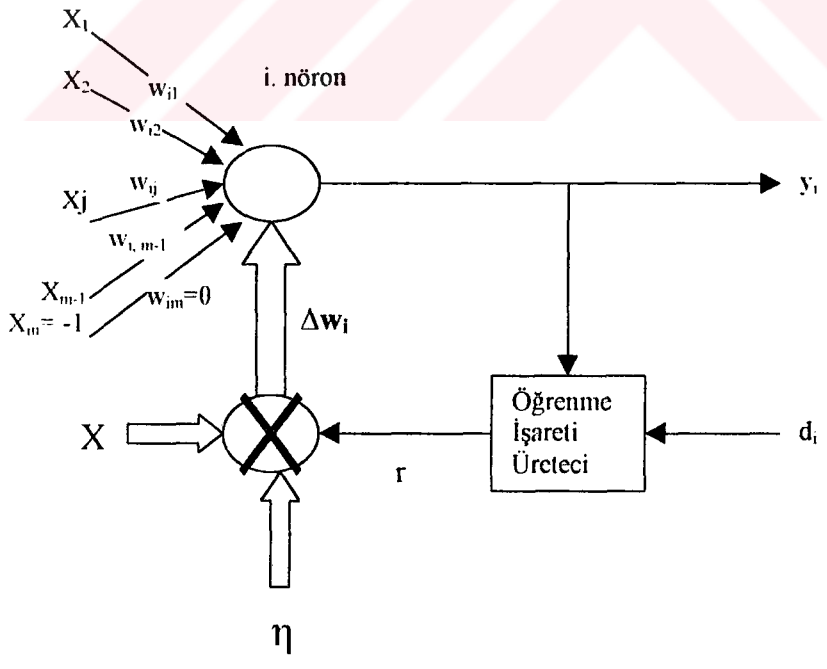
1.Eğiticili Öğrenme(Supevised learning)
2. Eğiticisiz Öğrenme(Unsupervised learning)
3.İşaretçi vasıtası ile Öğrenme (Reinforcement learning)

Eğiticili Öğrenme: Eğiticili öğrenmede her bir adımda giriş verilerini içeren bir X giriş vektörü ile bu giriş vektörüne ilişkin çıkışta görülmesi beklenen çıkış değerlerini içeren d çıkış vektörü ağa yüklenir. Dolayısıyla ağ çıkışı, ilgili girişe ilişkin çıkış değerini tam olarak bilmektedir. Daha açıkçası, eğiticili öğrenmede, YSA örnek giriş-çıkış çiftleri $\left(x^{(1)},d^{(1)}\right),\left(x^{(2)},d^{(2)}\right),\ldots,\left(x^{(k)},d^{(k)}\right),\ldots,$ ile eğitilir. Her bir $x^{(k)}$ vektörü YSA girişine verilirken aynı anda bu vektöre ilişkin çıkışta görülmesi beklenen $d^{(k)}$ vektörü de YSA' ya yüklenir. Her bir eğitim aşamasında YSA ürettiği mevcut çıkış değeri ile beklenen çıkış değerini karşılaştırarak bir hata sinyali üretir. Bu hata sinyaline göre de ağ parametreleri yani, ağırlık matrisi güncellenir. Eğitim işlemi, çıkışta arzu edilen değer belirinceye kadar ya da sözkonusu değere istenilen tolerans da ulaşılıncaya kadar devam edilir.

İşaretçi vasıtasıyla öğrenme: Eğiticili öğrenmede her giriş deseni için çıkışta ulaşılacak hedef desen, tam doğru bir şekilde bilinmektedir. Fakat bazı durumlarda, oldukça kısıtlı bilgiye sahip olunması sözkonusudur. Uç bir durum olarak , ulaşılan sonucun sadece doğru yada yanlış olduğunu gösteren ,yalnızca bir bitlik geribesleme bilgisi, ağ parametrelerini güncellemek için kullanılır. Böyle bir kritik bilgiyi temel alarak gerçekleştirilen öğrenme algoritmasına işaretçi vasıtalı(reinforced) öğrenme algoritması denir.Bu algoritma, ağın bir geribesleme sinyali ile ağırlık katsayılarını güncellemesinden dolayı eğiticili öğrenme yöntemine benzemektedir. Ancak eğiticili öğrenmede hedef deseninin kesin bir biçimde bilinmesine rağmen burada sadece o anki ağ çıkışındaki bilginin doğru mu yoksa yanlış mı olduğu bilinmektedir.

Eğiticisiz öğrenme: Eğiticisiz öğrenme de ise, eğiticili öğrenmenin aksine bir öğretici yoktur. Eğiticili öğrenmede örnek giriş-çıkış desen çiftleri bir öğretici gibi ağa verilerek ağın eğitimi yapılır. Bu öğrenme yönteminde ise herhangi bir giriş desenine karşı çıkış deseninin ne olması gerektiği belli değildir. Ağ böyle bir bilgiyle donatılmamıştır. Dolayısıyla ağın o andaki sahip olduğu ağırlık matrisiyle giriş desenine karşılık elde edilen çıkış deseninin doğru olup olmadığı bilinmemektedir. Ağ girişindeki verilere ait çıkış desenini, bu girişlere ait öznitelikleri , her bir desen arasındaki ilişkiyi, verilerin dağılımını ve söz konusu örnekleri gruplandırmayı kendi kendine keşfetmek zorundadır. Bu işlemleri yapmak, gerçekleştirmek için sürekli belli bir kural dahilinde ağ parametrelerini değiştirir. Bu işleve kendi kendini yönetme denir. Giriş desenlerinin başka bir bilgiye ihtiyaç duyulmadan gruplandırılması, kümelendirilmesi bu tür uygulamalara örnek olarak verilebilir. Desenler arasındaki benzerlikler ve farklılıklar ağ tarafından keşfedilerek her bir desenin ait olduğu sınıf belirlenir [Lin, Lee,1995].

Genelleştirilmiş Öğrenme Kuralı: Üç temel öğrenme kuralı göz önüne alınarak genelleştirilmiş bir öğrenme kuralı çıkarılabilir. Genelleştirilmiş öğrenme



Şekil 2.4. Eğiticili Öğrenme

kuralında w ağırlık matrisindeki değişme miktarı Δw, $r(t)$ öğrenme işareti ile $x(t)$ giriş işaretinin çarpımıyla orantılıdır. Yani

$$\Delta w_i(t) \propto r.x(t) \quad \text{veya} \quad \Delta w_i(t) = \eta.r.x(t) \tag{2.12}$$

şeklinde ifade edilir. Burada η öğrenme oranını gösteren pozitif bir sayıdır ve öğrenme sabiti denir. r ise , w_i, x ve eğer eğiticili öğrenme sözkonusu ise d' ye bağlı olan öğrenme işaretini gösterir. Yani;

$$r = f_r(w_i, x, d_i) \tag{2.13}$$

Dolayısıyla, genelleştirilmiş öğrenme kuralına göre her hangi bir andaki w_i ağırlık matrisi ;

$$w_i(t+1) = w_i(t) + \eta.f_r(w_i(t), x(t), d_i(t)).x(t) \tag{2.14}$$

dolayısıyla;

$$w_i^{(t+1)} = w_i^{(t)} + \eta.f_r\left(w_i^{(t)}, x^{(t)}, d_i^{(t)}\right).x^{(t)} \tag{2.15}$$

yazılabilir.

Eşitlik (2.14) öğrenmenin ayrık zamandaki ifadesidir. Sürekli zaman için herhangi bir t anındaki ağırlık matrisindeki değişim;

$$\frac{dw_i(t)}{dt} = \eta.r.x(t) \tag{2.16}$$

ifadesiyle gösterilir.

Eşitlik (2.15) ve (2.16) de görüldüğü gibi öğrenme işlemine başlamadan önce bir başlangıç ağırlık matrisi (genelde rastgele ağırlıklar içeren) belirlenmelidir.

Genelleştirilmiş öğrenme kuralı kullanılarak eğiticili veya eğiticisiz pek çok öğrenme kuralı türetilebilir. Eğiticili öğrenmede ,öğrenme işareti r, d_i' ye bağlı iken eğiticisiz öğrenmede beklenen çıkış değerleri sözkonusu olmadığından d_i belli değildir, dolayısıyla öğrenme işareti, x giriş işareti ve y çıkış işaretine bağlıdır.

Hebb öğrenme yasasının genelleştirilmiş öğrenme kuralına uyarlarsak, Hebb öğrenme kuralında öğrenme işareti r' nin tanımı aşağıdaki gibidir.

$$r = a\left(w_i^T x\right) = y_i \tag{2.17}$$

Burada $a(\bullet)$ hücrenin transfer fonksiyonunu göstermektedir.Görüldüğü gibi Hebb öğrenme yasasında , öğrenme işareti hücrenin çıkış işaretidir. Dolaysıyla ağırlık matrisindeki değişim için;

$$\Delta w_i = \eta . a\left(w_i^T x\right) x_j = \eta . y_i . x_j \qquad i = 1,2,\ldots,n; \qquad j = 1,2,\ldots,m \tag{2.18}$$

Bu bağıntıda Hebb öğrenme yasasının eğiticisiz bir öğrenme kuralı olduğu görülmektedir. Çünkü öğrenme işareti ağın giriş değerleri ile çıkış değerlerine bağlıdır. Şayet giriş ile çıkış işaretinin çarpımı pozitif ise, ağırlık katsayıları artarken, negatif olması durumunda ise ağırlık katsayıları azalacaktır.

Bu çalışmada yalnızca eğiticili öğrenme yöntemleri kullanıldığından diğer öğrenme yöntemleri üzerinde daha fazla durulmayacaktır.

2.3. Eğiticili Öğrenme Yöntemleri

YSA örnek giriş-çıkış çiftleri ile eğitilerek bir eğitim aşamasından geçiyorsa bu tip bir öğrenme yöntemine eğiticili öğrenme adı verildiği daha önceki bölümlerde belirtilmişti. Bu bölümde sıklıkla kullanılan eğiticili öğrenme yöntemine haiz YSA modelleri tanıtılacaktır. Özellikle tez içerisinde kullanılan geriye yayılma algoritması (backpropagation) ve çağrışımlı bellek yapısı gösteren Hopfield ağ modelleri üzerinde durulacaktır.

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

2.3.1 Tek Tabakalı Perceptron Ağları

Tek tabakalı perceptron ağlarında girişler ağırlık katsayıları üzerinden hücrelere bağlıdır. Eğiticili öğrenme sözkonusu olduğundan bazı örnek giriş değerleri için çıkış değerleri bilinmektedir ve ağ bu giriş çıkış çiftleri ile eğitilir. Şekilde görüldüğü gibi $x^k = \left[x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\right]^T$, giriş değerleri için beklenen çıkış deseni $d^k = \left[d_1^{(k)}, d_2^{(k)}, \ldots, d_n^{(k)}\right]^T$, olmalıdır. Eğitim işlemi sonucunda ulaşılan $y^k = \left[y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}\right]^T$, aktif çıkış değerleri, beklenen $d^{(k)}$, çıkış desenine eşit olmalıdır. Burada m girişlerin sayısı, n çıkışların sayısı, k ise örnek sayısıdır.

$$y_i^{(k)} = a\left(w_i^T . x^{(k)}\right) = a\left(\sum_{j=1}^{m} w_{ij} x_j\right) = d_i^{(k)} \quad i = 1,2,\ldots,n; \quad k = 1,2,\ldots,p \qquad (2.19)$$

$a(\bullet)$ transfer fonksiyonu bir eşik fonksiyonu veya sürekli sigmoid fonksiyonu olabilir. İşlem elemanının türüne göre uygulanacak öğrenme kuralı da değişir. İşlem elemanı LTU ise perceptron öğrenme kuralı, LGU ise Widrow-Hoff öğrenme kuralı kullanılır.

2.3.2. Perceptron Öğrenme Kuralı

Transfer fonksiyonu olarak signum fonksiyonunu kullanan percepron ağının çıkışında beklenen değerler $d_i^{(k)}$,ancak ∓ 1 olabilir. Dolayısıyla çıkış için;

$$y_i^{(k)} = \mathrm{sgn}\left(w_i^T x^{(k)}\right) = d_i^{(k)}, \qquad i = 1,2,\ldots,n; \qquad k = 1,2,\ldots,p \text{ yazılır.} \qquad (2.20)$$

Bu ilişkiyi sağlayabilmek için oluşturulacak w ağırlık matrisini güncelleyebilmek için gerekli öğrenme işareti, beklenen çıkış ile aktif çıkış arasındaki farktan elde edilir.

$$r = d_i - y_i \qquad (2.21)$$

Beklenen çıkış d_i değerleri ∓ 1 olacağından ağırlık matrisindeki değişim için ;

$$\Delta w_{ij} = \eta\left[d_i - \operatorname{sgn}\left(w_i^T x\right)\right]x_j = \begin{cases} 2\eta . d_i x_j & y_i \neq d_i \\ 0 & diger \end{cases} \qquad j = 1,2,\ldots,m \qquad (2.22)$$

bağıntısı geçerlidir.

Buradaki η öğrenme sabitinin büyük değerler alması öğrenmeyi hızlandırır. Çok fazla büyük değerler alması durumunda ise eğitim işleminin, yerel optimumlara takılması durumu ile karşılaşılır. Bu nedenle öğrenme sabitinin alacağı en uygun değer her problemde farklılıklar gösterir.

2.3.3. Adaline Öğrenme Kuralı

Eğitim kümesinin ve çıkışta beklenen değerlerin ayrık değerlerden değilde sürekli işaretlerden oluşması durumunda işlem elemanlarının (hücrelerin) aktivasyon fonksiyonları da buna uygun seçilmelidir. Bilindiği gibi sigmoid fonksiyonu gibi operasyonlar belli bir aralıkta sürekli değerler üretilmesine imkan tanırlar.

Tek tabakalı lineer aktivasyon fonksiyonuna sahip ağa adaptif lineer element denir ve kısaca ADALINE ismiyle anılır. ADALINE ağında girişlerle çıkışlar arasındaki ilişki lineerdir. Eğiticili öğrenmede $\left(x^{(1)}, x^{(2)}, \ldots, x^{(p)}\right)$ giriş örneklerine ait çıkışta $\left(d^{(1)}, d^{(2)}, \ldots, d^{(p)}\right)$ çıkış desenlerinin olacağı ağa eğitim aşamasında yüklenir. Burada ulaşılması istenen hedef, bu ilişkiyi sağlayacak w_i matrisnin bulunmasıdır. Öyleki

$$\sum_{j=1}^{m} w_j x_j^{(k)} = d^{(k)} \qquad k = 1,2,\ldots,p. \qquad (2.23)$$

eşitliği sağlanmalıdır.

Eğer giriş desenleri $\left(x^{(1)}, x^{(2)}, \ldots, x^{(p)}\right)$ lineer bağımsız olursa, eşitliği sağlayan w^* matrisi bulunur. Giriş desenleri $x^{(i)}$ lerin boyu m ise, p adet farklı desen arasındaki lineer bağımsızlık şartı için gerek şart $p \leq m$ olmalıdır.

Eşitlik (2.23) den ağırlık matrisini elde etmek için önce sistemin hatasını ölçen bir hata fonksiyonu $E(w)$ tanımlamak gereklidir.

$$E(w) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - y^{(k)}\right)^2 = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - w^T x^{(k)}\right)^2 \text{ ve}$$

$$E(w) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{j=1}^{m} w_j x_j^{(k)}\right)^2 \tag{2.24}$$

Açıkça görüldüğü gibi $E(w)$ normalde pozitiftir ve $y^{(k)}$, $x^{(k)}$'ya yaklaştıkça sıfıra yaklaşır. $E(w)$ küçüldükçe ideal w değerine ulaşılacağı görülmektedir. O halde yapılması gereken ortalama karesel hatayı yani $E(w)$ değerini minimize etmektir. Yapay sinir ağlarındaki öğrenme işlemini diğer geleneksel işlemlerden ayıran belki de en önemli özellik budur. Yani w matrisini bulmak için analitik çözümlemeler yapmak yerine w'nın rasgele bir başlangıç değerinden yola çıkarak en uygun çözümü buluncaya kadar w matrisini güncelleme işlemini gerçekleştirecek bir öğrenme algoritması kullanılmasıdır. Sık kullanılan algoritmalardan birisi Eğimdüşme (Gradient -descent) Öğrenme Algoritmasıdır. Bu yöntemle ağırlık matrisinin güncellenmesinde kullanılan Δw değişim miktarının bulunmasında, $E(w)$ fonksiyonun gradientinin ters işaretlisi kullanılır.

$$\Delta w = \eta \nabla_w E(w). \tag{2.25}$$

$$\Delta w_j = \eta \frac{\partial E}{\partial w_j} = \eta \sum_{k=1}^{p}\left(d^{(k)} - w^T x^{(k)}\right) x_j^{(k)} \quad j = 1,2,\ldots,m \tag{2.26}$$

Eğer bütün sınıflar topluca değil de sırayla ağın eğitimi için ağa sunulursa,herhangi bir $x^{(k)}$ giriş deseni için ağırlık katsayılarındaki değişim;

$$\Delta w_j = \eta\left(d^{(k)} - w^T x^{(k)}\right)x_j^{(k)} \tag{2.27}$$

eşitliğindeki gibi olacaktır.

Yukarıdaki eşitliğe Adaline öğrenme kuralı veya Widrow-Hoff öğrenme kuralı adı verilir. Bu aynı zamanda endüşük kareler ortalaması (LMS) yöntemi adını alır.Genelleştirilmiş öğrenme kuralı ile karşılaştırıldığında burada öğrenme sinyali;

$$r = d - y$$

alınmıştır.

Tez içerisinde Hopfield ve Backpropagation algoritmaları kullanıldığından bundan sonraki bölümlerde bu algoritmalar üzerinde durulacaktır.

2.4. Geriye Yayılma (Back Propagation)

Sinir ağlarının tarihsel gelişimi içinde en önemli öğrenme algoritmalarından birisi geriye yayılma (back propagation) algoritmasıdır.[Bryson ve Ho,1969; Werbos,1974; LeCun, 1985; Parker,1985; Rumelhart et al.,1986a,b]. Bu öğrenme algoritması çok tabakalı ileri beslemeli ağlar ile birleştirildiğinde elde edilen yapay sinir ağına geriye yayılma ağı adı verilir. Eğitim kümesindeki $\left\{x^{(k)}, d^{(k)}\right)$ giriş-çıkış çiftleri verildiğinde geriye yayılma ağının giriş desenlerini doğru bir şekilde sınıflandırabilmesi için öğrenme kuralı vasıtasıyla ağırlık matrisi değiştirilir. Ağırlıkların güncellenmesi, perceptron ağlarında kullanılan eğim düşümü (gradient-descent) algoritmasına dayanır. Geriye yayılma ağında iki yönlü bilgi akışı söz konusudur. İlk olarak giriş deseni $x^{(k)}$, ağ girişinden ağ çıkışına doğru gönderilir. Bu giriş desenine karşılık çıkışta oluşan $y^{(k)}$ çıkış deseni ile çıkışta görülmesi beklenen $d^{(k)}$ deseni arasındaki fark, yani hata sinyali çıkıştan geriye, giriş tabakasına doğru iletilir. Ağ parametrelerinin güncellenmesi bu hata sinyali ile yapılır. Bunları daha

açık olarak görmek amacıyla şekildeki geriye yayılma ağını dikkate alınırsa, sürekli çizgiler giriş işaretinin ileriye doğru yayılmasını, kesikli çizgiler ise hata işaretinin geriye doğru yayılmasını göstermektedir.

İfadeleri basitleştirmek için k indisi kaldırılmış bir giriş- çıkış eğitim çifti (x,d) ele alınırsa, verilen x giriş deseni için saklı tabakadaki q işlem elemanın alacağı toplam giriş işareti aşağıdaki gibi olur.;

$$net_q = \sum_{j=1}^{m} v_{qj} x_j \tag{2.28}$$

ve hücre çıkışında üretilen işaret;

$$z_q = a(net_q) = a\left(\sum_{j=1}^{m} v_{qj} x_j\right) \tag{2.29}$$

olacaktır. Çıkış katmanındaki i. hücrenin girişine ulaşan toplam giriş işareti ise;

$$net_i = \sum_{q=1}^{l} w_{iq} z_q = \sum_{q=1}^{l} w_{iq} a\left(\sum_{j=1}^{m} v_{qj} x_j\right) \tag{2.30}$$

olur ve çıkış işareti;

$$y_i = a(net_i) = a\left(\sum_{q=1}^{l} w_{iq} z_q\right) = a\left(\sum_{q=1}^{l} w_{iq} a\left(\sum_{j=1}^{m} v_{qj} x_j\right)\right). \tag{2.31}$$

Yukarıdaki eşitlikler back propagation ağında ileri doğru yayılma denklemleridir. Geriye doğru yayılma eşitliklerini elde etmek için ilk olarak hatayı belirlemek gerekir. Toplam karesel hatayı belirleyen hata fonksiyonu;

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}[d_i - a(net_i)]^2 \qquad \text{ve}$$

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}\left[d_i - a\left(\sum_{q=1}^{l} w_{iq} z_q\right)\right]^2 \tag{2.32}$$

Eğim düşümü algoritmasına göre, saklı tabakadan çıkış tabakasına doğru bütün ağırlık katsayıları aşağıdaki kuralla güncellenir.

$$\Delta w_{iq} = -\frac{\partial E}{\partial w_{iq}} \tag{2.33}$$

(2.30)-(2.32) eşitlikleri kullanılarak zincir kuralıyla $\partial E / \partial w_{iq}$ için;

$$\Delta w_{iq} = -\eta\left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial w_{iq}}\right] = \eta[d_i - y_i][a'(net_i)][z_q] = \eta\delta_{oi} z_q \tag{2.34}$$

Bu son eşitlikteki δ_{oi} hata işaretidir ve ikinci indis çıkış tabakasındaki i. hücreyi gösterir. Hata işareti aşağıdaki gibi tanımlanır;

$$\delta_{oi} = -\frac{\partial E}{\partial net_i} = -\left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right] = [d_i - y_i][a'(net_i)] \tag{2.35}$$

Bu tanım aynı zamanda delta öğrenme kuralında kullanılan hata işaretinin tanımıdır. Çıkış katmanındaki i. hücreye giren toplam işaret net_i ile gösterilirken $a'(net_i)$ için aşağıdaki tanım yapılır.

$$a'(net_i) = \frac{\partial a(net_i)}{\partial net_i} \tag{2.36}$$

Eşitlikteki z_q ise saklı tabakanın çıkış değeridir. Giriş katmanıyla, saklı tabaka arasındaki ağırlıkların güncellenmesi için de eğimdüşümü yöntemi kullanılır. Giriş tabakasındaki j. hücre ile, saklı tabakadaki q. hücre arasındaki bağlantı ağırlığındaki değişim miktarı;

$$\Delta v_{qj} = -\eta\left[\frac{\partial E}{\partial v_{qj}}\right] = -\eta\left[\frac{\partial E}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right] \text{ ve}$$

$$\Delta v_{qj} = -\eta\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right] \tag{2.37}$$

zincir kuralıyla elde edilir.

Eşitlik (2.32) ' den açıkça görüldüğü gibi her bir hata terimi $[d_i - y_i]$, z_q 'nun fonksiyonudur. Zincir kuralının uygulanmasıyla aşağıdaki ifade elde edilir.

$$\Delta v_{qj} = \eta\sum_{i=1}^{n}\left[(d_i - y_i).a'(net_i)w_{iq}\right]a'(net_q)x_j \tag{2.38}$$

Eşitlik (2.35) kullanılarak, bu son eşitlik yeniden yazılabilir;

$$\Delta v_{qj} = \eta\sum_{i=1}^{n}\left[d_{oi}w_{iq}\right]a'(net_q)x_j = \eta\delta_{hq}x_j \tag{2.39}$$

Buradaki δ_{hq}, saklı tabakadaki q. elemana ait hata işaretidir ve aşağıdaki gibi tanımlanır;

$$\delta_{hq} = -\frac{\partial E}{\partial net_q} = -\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right] = a'(net_q)\sum_{i=1}^{n}\delta_{oi}w_{iq} \tag{2.40}$$

eşitlikteki net_q saklı tabakadaki q. elemana ait toplam giriş işaretini göstermektedir. Saklı tabakadaki bir hücreye ait hata işaretinin , çıkış tabakasındaki bir hücreye ait hata sinyalinden farklı olduğu görülmektedir. Bundan dolayı (2.40) da tanımlı olan hata işaretini kullanan ağırlık güncelleme eşitliği, eşitlik (2.38) ve (2.39)'a genelleştirilmiş delta öğrenme kuralı ismi verilir. Görüldüğü gibi saklı tabakadaki hata işareti δ_{hq}, çıkış tabakasındaki hata işareti δ_{oi}'nin fonksiyonudur. Daha açıkçası hata işareti çıkış tabakasından girişe doğru yayılmaktadır. Eşitlik

(2.34) ve (2.39) genelleştirilmiş öğrenme kuralı açısından yaklaşılınca birbirinin aynıdır. Aralarındaki fark $r = \delta$ hata işaretinin tanımlarındadır. Dolayısıyla geriye yayılma ağlarında, herhangi bir katman için genelleştirilmiş ağırlık güncelleme ifadesi;

$$\Delta w_{ij} = \eta \delta_i x_j = \eta \delta_{i.cikis} x_{j.giriş} \tag{2.41}$$

Bu ifadedeki "i. çıkış" ve "j. giriş" ifadeleri j. hücreden i. hücreye doğru olan bağlantının uç noktalarını gösterir. Saklı tabakadaki veya giriş tabakasındaki giriş ucu x_j ile temsil edilirken, δ_i ise son çıkış tabakası için eşitlik (2.35) ,diğer katmanlar için ise eşitlik (2.40) ile tanımlanan hata işaretidir. Hücrelerde transfer fonksiyonu olarak bipolar sigmoid fonksiyonunun kullanılması durumunda hata işaretine ilişkin ifadeler aşağıdaki ifadeler dönüşür;

$$\delta_{oi} = \frac{1}{2}\left(1 - y_i^2\right)\left[d_i - y_i\right] \tag{2.42}$$

ve

$$\delta_{hq} = \frac{1}{2}\left(1 - z_q^2\right)\sum_{i=1}^{n} \delta_{oi} w_{iq} \tag{2.43}$$

Sonuç olarak, geri yayılma algoritması aşağıdaki BP algoritmasıyla adım adım işletilir.

2.4.1. BP Algoritması

Q adet tabakaya sahip bir ağ tasarlanmış olsun . $q = 1,2,\ldots Q$ ve ${}^{q}net_i$ ve ${}^{q}y_i$ q. katmandaki i. hücrenin sırasıyla toplam girişi ve çıkışı olsun. Ağ m adet giriş ve n adet çıkış hücresi olsun. ${}^{q}w_{ij}$ ise, ${}^{q-1}y_j$ hücresi ile ${}^{q}y_i$ hücresi arasındaki bağlantının ağırlık katsayısı olsun.

Giriş: Giriş vektörlerinin son elemanı $x_{m+1}^{k} = -1$ olan , $k = 1,2,\ldots,p$ için, p adet $\left\{x^{(k)}, d^{(k)}\right)$ örnek giriş-çıkış eğitim çiftleri kümesi, ağ için giriş verileri kabul edilir.

I. Adım (Ağın kurulması): Öğrenme sabiti $\eta > 0$ ve izin verilebilir en büyük hata E_{max} belirlenir. Ağırlık matrislerinin elemanları rastgele küçük değerlerden oluşturulur. $E = 0$ ve $k = 1$ yapılır.

II. Adım (Eğitim çevrimi): k. giriş vektörü, giriş katmanına $(q = 1)$ uygulanır:

$$^{q}y_i = {}^{1}y_i = x_i^{(k)} \qquad \forall i \text{ için}$$

III. Adım (İleri Yayılma) : Aşağıdaki eşitlik kullanılarak işaret,bütün ağ boyunca ileri doğru iletilir.

$$^{q}y_i = a\left(^{q}net_i\right) = a\left(\sum_j {}^{q}w_{ij}\, {}^{q-1}y_i\right) \tag{2.44}$$

Bütün q ve i değerleri için , çıkış katmanına ait $^{Q}y_i$ çıkışlarına ulaşıncaya kadar işleme devam edilir.

IV. Adım (Çıkış hatasının ölçülmesi): Çıkıştaki hata değeri ve hata işareti $^{Q}\delta_i$ hesaplanır.

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(d_i^{(k)} - {}^{Q}y_i\right)^2 + E \tag{2.45}$$

$$^{Q}\delta_i = \left(d_i^{(k)} - {}^{q}y_i\right)a'\left(^{Q}net_i\right) \tag{2.46}$$

V. **Adım** (Hatanın geriye yayılması): Ağırları güncellemek için hata geriye doğru yayılır ve ${}^{q-1}\delta_i$ hata sinyalleri her bir katman için hesaplanır.

$$\Delta^q w_{ij} = \eta \delta_i {}^{q-1} y_i \quad \text{ve} \quad {}^q w_{ij}^{yeni} = {}^q w_{ij}^{eski} + \Delta^q w_{ij} \tag{2.47}$$

$${}^{q-1}\delta_i = a'\left({}^{q-1}net_i\right)\sum_j {}^q w_{ji} {}^q \delta_j \qquad q = Q, Q-1, \ldots, 2. \text{ için} \tag{2.48}$$

VI. Adım (Bir eğitim çevrimi): Bütün giriş desenleri tamamlanıncaya kadar $k < p$ olduğu müddetçe $k = k + 1$ yap ve II. adıma dön. Aksi halde VII. Adıma git.

VII. Adım (Toplam hata kontrolü): Toplam hatanın izin verilen sınırlar içinde olup olmadığını kontrol et: Eğer $E < E_{\max}$ ise eğitim işlemini sonladır ve o anki bağlantı katsayılarını çıkışa gönder. Aksi halde II. adıma git $E = 0$ ve $k = 1$ yap ve yeni bir eğitim işlemi başlat.

Yukarıdaki algoritmada her bir eğitim desenin ağa sunulmasından sonra ağırlık matrisleri güncellenir. Güncelleme işleminin başka bir yöntemide bütün eğitim desenlerinin ağa sunulmasından sonra eğitim işleminin yapılmasıdır.

2.4.2.Geriye Yayılma Algoritmasında Öğrenme Faktörleri

Bu bölümde geriye yayılma ağlarında ağ performansını etkileyen önemli öğrenme faktörleri üzerinde durulacaktır. Bu faktörler şunlardır:

1. Başlangıçtaki ağırlık katsayıları
2. Öğrenme sabiti
3. Hata fonksiyonu
4. Güncelleme kuralı
5. Ağ mimarisi, katman sayısı ve katman başına düşen hücre sayısı
6. Eğitim kümesinin boyutu ve kullanılan özniteliklerin problemi temsil etme yeteneği

Başlangıç ağırlıkları: Çok tabakalı ağlarda ağırlık matrislerinin başlangıçta içerdiği ağırlıkların problemin çözümünde çok büyük önemi vardır. Genelde rasgele küçük değerler atanır. Başlangıçta atanan ağırlıklar çok büyük olamaz. Çünkü bu durumda sigmoid fonksiyonu eğitim işleminin başlangıcında doymaya girebileceğinden, hatayı ölçen hata fonksiyonunun başlangıç noktasından sonraki ilk yerel minimum noktasında veya ilk düzlükte algoritma tıkanır. Yani asıl minimuma ulaşmak imkansızlaşır. Başlangıç ağırlıklarının uygun değerlerde seçilmesi için önerilen bir yöntem, w_{ij} bağlantı ağırlıklarının $\left[-3/\sqrt{k_i}, 3/\sqrt{k_i}\right]$ aralığında olmasıdır. İfadedeki k_i değeri, i. hücre girişine gelen bağlantı sayısıdır. Başka bir deyişle ağ tam bağlantılı bir ağsa bir önceki katmandaki hücre sayısıdır.[Wessel ve Barnard,1992].

Öğrenme Sabiti: Geri yayılma algoritmasının verimliliğine ve çözüm noktasına yakınsamasına kuvvetli bir etki yapan bir diğer faktör η öğrenme sabitidir. Farklı eğitim aşamaları için uygun tek bir öğrenme sabiti değeri belirlenmemiştir.Uygun öğrenme sabitinin belirlenmesi her bir problem için deneysel olarak belirlenir. Yaniöğrenme sabitinin belirlenmesinde bir kriter yoktur. Bununla birlikte bazı genelemeler de yapılabilir. Büyük değerli öğrenme sabiti çözüme ulaşmada algoritmanın hızını artırırken, bağlantı ağırlıklarında olduğu gibi doymaya da sebebiyet verebilir. Öğrenme sabitinin η ,10^{-3} ile 10 arasında aldığı değerlerin başarılı sonuçlar ürettiği deneysel olarak gözlenmiştir.

Öğrenme sabitinin algoritma boyunca sabit tutulması yerine adaptif olarak değiştirilmesi algoritmanın performansını daha da artırmıştır.[Vogl et al.,1988; Jacobs,1988]. Bu şekilde her bir noktada en uygun öğrenme sabiti seçilerek, öğrenme sabitine bir esneklik kazandırılmıştır. Güncellenen ağırlıkların ağın karesel hatasını azaltması gerekir. Eğer hata fonksiyonu, yani karesel hata her adımda azalıyorsa, bu azalmayı muhafaza ederek, öğrenme sabitinin değerinin daha büyük değerler almasına çalışılır. Özet olarak, öğrenme sabiti aşağıda kurala göre güncellenir.[Hertz et al. ,1991.]

$$\Delta\eta = \begin{cases} +a & \Delta E < 0 \quad \textit{süreklilik} \quad \textit{gösteriyorsa} \\ -b\eta & \Delta E > 0 \\ 0 & \textit{diger} \end{cases} \tag{2.49}$$

ΔE, hata fonksiyonundaki değişimi gösterirken buradaki a ve b sembolleri ise pozitif sabitlerdir. Bir başka kural ise aşağıda ifade edilen delta-bar-delta kuralıdır.

$$\Delta\eta(t+1) = \begin{cases} +a & \bar{\lambda}(t-1)\lambda(t) > 0 \\ -b\eta(t) & \bar{\lambda}(t-1)\lambda(t) < 0 \\ 0 & \textit{diger} \end{cases} \tag{2.50}$$

Burada $\lambda(t) = \dfrac{\partial E}{\partial w_{ij}}$ ve $\bar{\lambda}(t) = (1-c)\lambda(t) + c\bar{\lambda}(t-1)$ (2.51)

Şeklinde tanımlanmıştır. Eşitlikte kullanılan sabit ise $c \in [0,1]$ aralığında bir değere sahiptir.

Hata Fonksiyonları: Yukarıdaki ifadelerde kullanılan karesel fonksiyondan başka hata fonksiyonları da vardır. Hata fonksiyonları genelde L_p normunda $1 < p < \infty$ ifade edilir. Genel tanımı aşağıdaki gibidir.

$$E = \frac{1}{p}\sum_i (d_i - y_i)^p \qquad 1 \le p < \infty \tag{2.52}$$

Özel olarak, karesel hata fonksiyonunda kullanılan en küçük kareler kriteri yani L_2 normu, basitliğinden dolayı en çok kullanılan kriterdir. Bunun yanında L_1 ve L_∞ normlarını esas alan hata fonksiyonları da yaygın olarak kullanılır. Özellikle L_∞ Chebyshev normu olarak adlandırılır.

Momentum: Eğim düşümü algoritması, eğer öğrenme sabiti küçük ise çok yavaş çalışabilir veya öğrenme sabiti çok büyük ise osilasyona sebep olabilir. Büyük değerli öğrenme sabitine sahip, eğim düşümü algoritmasının osilasyaonlara sebep

olmaması için algoritmaya momentum terimi ilave edilir [Plaut et al.,1986]. Buna göre ağırlık matrisindeki değişim;

$$\Delta w = -\eta \nabla E(t) + \alpha . \Delta w(t-1) \tag{2.53}$$

şeklinde ifade edilir.

İfadedeki α terimi momentum terimidir ve $\alpha \in [0,1]$ aralığında değerler alır. Genelde 0.9 değeri ideal bir seçimdir. Ancak problem türüne göre bu değer değişebilir.

Güncelleme Kuralları: Taylor serisini kullanılarak $E(w)$ fonksiyonu w_0 noktasında seriye açılırsa ;

$$E(w) = E(w_0) + (w - w_0)^T \nabla E(w_0) + \frac{1}{2}(w - w_0)^T H(w)(w - w_0) + \ldots \tag{2.54}$$

elde edilir. Eşitlikteki $H(w)$'e Hessian matrisi adı verilir ve w_0 noktasındaki ikinci türevdir.

$$H(w) = \nabla^2 E(w) \quad \text{veya} \quad H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{2.55}$$

$E(w)$' nin minimum değeri bu fonksiyonun türevini sıfır yapan noktada olacağından;

$$\nabla E(w) = \nabla E(w_0) + H(w)(w - w_0) + \ldots = 0 \tag{2.56}$$

Bu ifadedeki üçüncü ve daha yüksek dereceli terimler ihmal edilirse;

$$w = w_0 - H^{-1}(w) \nabla E(w_0) \tag{2.57}$$

ifadesi elde edilir ve k indisi k. eğitim adımını göstermek üzere aşağıdaki bağıntı yazılabilir.

$$w^{(k+1)} = w^{(k)} - H^{-1}(w^{(k)})\nabla E\left(w^{(k)}\right) \tag{2.58}$$

Bu eşitlik, ağırlıkların güncellenmesinde Newton yöntemi olarak bilinir. Çözümün konveks bir fonksiyona benzemesi halinde Newton Yöntemi hızla çözüme yaklaşır. Ancak bu yöntemin bazı mahsurları da vardır. İlk olarak çözüme yaklaşmak için başlangıçta çözüm için iyi bir kestirim yapmak gereklidir.Yani başlangıç ağırlıkların seçimi çok önem taşımaktadır. Bir diğer husus, konveks olmayan bir fonksiyon söz konusu olduğunda, bu yöntemde yerel minimumlara takılmak kaçınılmazdır. En önemli mahsur ise her eğitim aşamasında Hessian matrisinin ve tersinin hesaplanmasının gerekmesidir. Bu oldukça fazla hesaplama yükü demektir ve bunların depolanması büyük yer tutar. Dolayısıyla bu yöntem pek pratik olmayan bir yöntemdir.

Eğitim Kümesi: Her alanda olduğu gibi yapay sinir ağlarında da doğru çözüme ulaşmak için doğru verilere ihtiyaç vardır. Eğer girişteki veri kümesinin problemi temsil etme yeteneği güçlü değilse, hangi algoritma kullanılırsa kullanılsın doğru sonuca ulaşmak mümkün değildir. Yaygın bir kural , bütün giriş uzayını kapsayan verileri kullanarak, bu verilerden rasgele seçilmiş eğitim vektörlerinin eğitim işleminde kullanılmasıdır. Örnek olarak, hiper düzlemlerle lineer olarak ayrıştırlabilir M adet bölgenin sınırlarını belirlemek için kullanılan eğitim vektörlerinin eleman sayısı P ise, $P/M >> 1$ olmalıdır. Ancak bu tek başına yeterli değildir. Eğitim vektöründe kullanılan elemanların giriş uzayındaki değerlerden seçilmesinde izlenen yol önemlidir. Hatta giriş uzayındaki verilerin kullanılması yerine, bu verileri kullanarak çıkarılan öznitelik vektörlerinin kullanılması eğitim sürecinin daha verimli olmasını sağlar. Özniteliklerin çıkarılmasında her probleme cevap verebilecek genel bir yöntem yoktur. Bu çalışmadaki yaprak şekillerinin temsil edilmesinde yaprak çevresini oluşturan noktaların yaprak merkezine olan uzaklıkları öznitelik vektörlerinin elemanlarını oluşturmaktadır.

İç Katmandaki hücre sayısı: Saklı katmandaki hücre sayısının çözüme etki ettiği kesindir. Ancak bu etkinin analitik olarak gözlemlenmesi pek mümkün değildir. Dolayısıyla saklı katmandaki hücre sayısı deneysel olarak belirlenir. Özellikle çok sayıda giriş içeren, yani giriş vektörlerinin boyu dolayısıyla giriş katmanındaki hücre sayısı yüzlerce olan problemlerde deneysel olarak elde edilen sonuölara göre, iç katmandaki hücre sayısı giriş tabakasındaki hücre sayısının çok küçük bir kesri olmalıdır. Eğer mevcut hücre sayısı ile çözüme yaklaşılamıyorsa iç katmandaki hücre sayısı makul ölçülerde artırılmalıdır. Çözüme hızlı bir yakınsama söz konusu olduğunda ise iç tabakadaki hücre sayısı azaltılır. Bu şekilde en optimum çözüm bulununcaya kadar devam edilir [Freeman ve Skapura, 1991].

Giriş katmanı m elemanlı olan M adet lineer olarak ayrıştıralabilir farklı bölge içeren ve girişteki verinin bu M sınıftan hangisine ait olduğunun bulunmasını sağlayacak bir YSA 'daki gizli katman sayısı N_m ise, bu büyüklükler arasında aşağıdaki ilişkiler mevcuttur.[Mirchandini ve Cao,1989; Huang ve Huang, 1991].

$$N_m < j \text{ için } \binom{N_m}{j} = 0 \text{ olmak üzere } \quad N_m + 1 \le M \le \sum_{j=0}^{m} \binom{N_m}{j} \tag{2.59}$$

O halde saklı tabakada N_m adet hücre içeren bir ağın sınıflandırabileceği maksimum sınıf sayısı $M_{\max}$, $N_m > m$ olması halinde aşağıdaki gibidir.

$$M_{\max} = \sum_{j=0}^{m} \binom{N_m}{j} \quad \text{ve}$$

$$M_{\max} = 1 + N_m + \frac{N_m(N_m - 1)}{2!} + \ldots + \frac{N_m(N_m - 1)\ldots(N_m - j + 1)}{m!} \tag{2.60}$$

$N_m \le m$ olması halinde ise;

$$M_{\max} = 2^{N_m} \quad \text{ve } N_m = \log_2 M_{\max} \quad \text{olur.} \tag{2.61}$$

2.5. Hopfield Ağları

Hopfield' in 1982 ve 1984 yıllarındaki yayınları ile YSA' da önemli bir gelişme olmuş ve ileri sürülen ağ modeli Hopfield ağı adını almıştır. Hopfield ağı bir tür çağrışımlı bellek görevi yapmaktadır. Bu ağlar sürekli Hopfiel ağı ve ayrık Hopfield ağı olmak üzere ikiye ayrılır.

2.5.1. Ayrık Hopfield Ağları

Hopfield ağı Şekil (2.2.d)' de görüldüğü gibi tek katmanlı geri beslemeli bir ağdır. Şekilde her bir hücrenin kendi üzerine de geri besleme elemanı olmasına rağmen Hopfield ağında hücrenin kendi üzerine geri beslemesi yoktur veya sözkonusu bağlantı ağırlığı sıfırdı. Eğer ayrık zamanda çalışma söz konusu ise ayrık Hopfield ağı adı verilir. Girişe verilen bir patern ağ çıkışında değişiklik olmasına sebep olur. Daha sonra giriş deseni kaldırılır ve o andaki ağ çıkışı geri besleme yapılarak girişe verilir. Birinci giriş deseni birinci çıkış desenini, ikinci giriş deseni ikinci çıkış deseninin oluşmasına sebep olur ve bu işlem böylece devam eder. Her adımda giriş deseni bir önceki adımdaki çıkış desenidir. Ağ çıkışında bir önceki çıkışa göre bir değişiklik olmayıncaya kadar işlem sürer.

Hopfield ağında her bir hücre harici bir x_j girişine ve θ_j eşik değerine sahiptir. Hücrelerin kendileri üzerine geri besleme bağlantısı yoktur. Bir hücrenin çıkışı kendisi dışındaki diğer hücrelerin girişine geri besleme yolu ile w_{ij} ağırlık katsayısı üzerinden bağlıdır. Dolayısıyla $w_{ii} = 0$ dır. W matrisinin köşegen elemanlarının sıfır olması dışında bir başka özelliği simetrik olmasıdır. Yani $w_{ij} = w_{ji}$ eşitliği matrisin belirgin bir özelliğidir. Ayrık Hopfield ağında i. hücrenin k. güncelleme aşamasındaki çıkışı $y_i^{(k)}$ olmak üzere, bir sonraki $(k+1)$. çıkış değeri ;

$$y_i^{(k+1)} = \operatorname{sgn}\left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i \right) \quad , \qquad i = 1,2,\ldots,n \qquad (2.62)$$

ifadesindeki gibidir. Ağdaki hücre sayısı ise n adettir.

Hopfield ağının iyi sonuç verebilmesi için kararlılığının iyi olması gerekir. Ağın enerji fonksiyonun davranışı ile kararlılık gözlemlenebilir.

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_i y_j - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i \tag{2.63}$$

Ağın kararlılığının gösterilmesi için yukarıdaki eşitlikteki E enerji fonksiyonunun değerinin, hücre çıkışlarının her değişmesinde, azaldığının gösterilmesi gereklidir. Ağdaki i. hücre çıkışının -1' den +1'e değiştiğini veya tam tersi yönde değiştiği farzedilsin. Bu durumda enerji fonksiyonundaki değişim,

$\Delta E = E\left(y_i^{(k+1)}\right) - E\left(y_i^{(k)}\right)$ ve buradan

$$\Delta E = -\left(\sum_{\substack{i=1 \\ j\neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i\right)\left(y_i^{(k+1)} - y_i^{(k)}\right) \tag{2.64}$$

bağıntısı elde edilir. İfadeyi basitleştirirsek;

$$\Delta E = -(net_i)\Delta y_i \tag{2.65}$$

olur. Burada net_i, i. hücrenin net giriş işaretidir.

$$net_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i \tag{2.66}$$

Eşitlik (2.60)' a göre hücre çıkışı $-1 \rightarrow 1$ yönünde değiştiğinde $\Delta y_i = 2$ olur. Bu durumun gerçekleşmesi için net_i pozitif bir değere sahip olmak zorundadır.

Dolayısıyla ΔE negatif olacaktır. Benzer şekilde çıkıştaki eğişim tersi yönde $+1 \rightarrow -1$ olduğunda ise $\Delta y_i = -2$ olacaktır. Bu değişimin olabilmesi için ise net_i negatif bir değere sahip olmalıdır. Dolayısıyla ΔE yine negatif olacaktır. Enerjideki değişimin her zaman negatif olması ağın kararlılığını gösterir. Sonuçta ağ enerjisini minimum yapan değerde karar kılacaktır. Bu minimum değer fonksiyonun asıl minimumu olabileceği gibi büyük bir olasılıkla yerel minimum noktası olacaktır. Ağ çalışmaya başladığı noktadan itibaren çıkışında sonlu sayıda güncelleme yapıldığında bir kararlılık noktasına ulaşır. Yerel minimumlara takılmaması için desen sınıflandırma uygulamalarında, sınıfları temsil eden desenlerin öznitelikleri birbirinden mümkün olduğunca uzak olmalıdır. Aksi halde bir yerel minimuma takılan ağ çıkışında üretilen desen hiçbir sınıfa ait olmaz.

2.5.2. Sürekli Hopfield Ağları:

Ayrık Hopfield Ağları, zamana göre sürekli değişen değerler alan ve hücreleri ayrık ikili değerler değilde, sürekli işaretler üreten, sürekli zaman modeline genelleştirilebilir. Hopfield , Sürekli Hopfield ağının da, ayrık Hopfield ağının sahip olduğu üstünlüklere sahip olduğunu göstermiştir. Sürekli Hopfield ağında transfer fonksiyonu olarak signum fonksiyonu yerine sigmoid fonksiyonu kullanılır.

Hopfield ağlarının en çok kullanıldığı alanlardan birisi daha önce değinildiği gibi çağrışımlı bellek (associative memory) uygulamalarıdır. Tez içerisinde Hopfield ağının bu özelliği kullanıldığından bundan sonraki bölümde bu konu üzerinde durulacaktır.

2.5.3. Çağrışımlı Bellekler

Bir çağrışımlı bellek, bir desen kümesini bir bellek gibi saklar. Çağrışımlı bellek girişine bir test deseni sunulduğunda, çıkışında daha önce sakladığı desenlerden bu test desenine en çok benzeyen deseni üretir. Buna geri çağırma işlemi adı verilir. Geleneksel bilgisayarlarda kullanılan hafıza yönteminde bilgi belli adreslerde byte formunda saklı desenler şeklinde tutulur. Bu tip hafızalara içerik

adreslemeli bellek (content-addressable memory) adı verilir. Geri çağırma işleminde bilginin saklandığı adres vasıtasıyla bilgiye ulaşılır.

Direk ve dolaylı olmak üzere iki farklı çağrışımlı bellek türü vardır (autoassociative memory, heteroassociative memory). Sırasıyla giriş ve çıkış desen vektörleri $x^i \in \Re^n$ ve $y^i \in \Re^m$ olsun, $\{(x^1, y^1), (x^2, y^2), \ldots, (x^p, y^p)$ desen çiftleri verilsin. Direk çağrışımlı bellekte $x^i = y^i$ bağıntısı vardır. Yani bir sınıfı temsil eden giriş deseni ile o sınıfa ait çıkış deseni aynı desendir. Belleğin gerçekleştirdiği fonksiyonun tanımı eşitlik (2.67)' de gösterilmiştir.

$$\phi(x^i) = x^i \ , \ \ \phi : \Re^n \rightarrow \Re^n \tag{2.67}$$

Dolaylı çağrışımlı bellekte ise giriş deseni ile çıkış deseni farklıdır. Buna ilişkin fonksiyon ise eşitlik (2.68)'da gösterilmiştir.

$$\phi(x^i) = y^i \ , \ \ \phi : \Re^n \rightarrow \Re^m \tag{2.68}$$

Bu çalışmada direk çağrışımlı bellek görevi yapan Hopfield ağı kullanılmıştır. Başlangıçta sınıfları temsil eden desenlerin hafıza yüklenme kuralı aşağıdaki gibidir

$$W = \sum_{k=1}^{p} x^{(k)} x^{(k)T} - pI \tag{2.69}$$

Daha açık bir ifadeyle;

$$w_{ij} = \sum_{k=1}^{p} x_i^k x_j^k \ , \qquad i \neq j; w_{ii} = 0 \tag{2.70}$$

elde edilir.

Belleğe yüklenen p adet desene ait bilgi, W ağırlık matrisinde tutulmaktadır. Bu matrisin elemanlarının değerleri değişmeyip sabit kalmaktadır. Yani Hopfield ağında öğrenme algoritması yoktur. Öğrenme algoritmasına karşılık eşitlik (2.70)'de ifade edilen desenlerin belleye depolanma kuralı vardır.

Hopfield ağında doğru bir sınıflandırma işlemi için sınıf sayısı p ile giriş vektörünün eleman sayısı başka bir değişle hücre sayısı n arasındaki ilişki aşağıdaki eşitlikte ifade edilen şarta uymak zorundadır.[Hopfield, 1983; Amit et al.,1985].

$$p \leq 0.14n \tag{2.71}$$

Sınıf sayısının bu değerden fazla olması durumunda bazı test desenleri için çıkışta beklenmeyen, ağa yüklenmeyen desenler görülebilir. Aynı durum biribirine çok yakın desenlerin sınıflandırılmasında da görülebilir. Bu nedenle iyi bir sınıflandırma işlemi için, yukarıdaki oran gözününde tutulmalı ve öznitelik vektörlerini belirlerken sınıfların farklarını ortaya koyan özelliklere ait değerlerden oluşturmaya özen gösterilmelidir.

3.YAPRAK DESENLERİNİN SINIFLANDIRILMASI

3.1. Desen Sınıflandırma

Görüntü ve ses işaretlerinin tanınması uygulamalarında desen sınflandırma tekniklerinden yararlanılmaktadır. Desen, bir görüntü parçasına, bir ses işaretine veya herhangi bir elektriksel işarete ait bilgileri içeren veri kümesine verilen isimdir. Desen sınıflandırma ise bu tür verileri belli sınıflara, veri kümelerine ayırma işlemidir. Deseni temsil eden veri kümesi bir boyutlu, yani vektör ise buna bir boyutlu desen, bir matris ise iki boyutlu,eğer uzayda bir bölgeyi temsil ediyorsa üç boyutlu desen adı verilir. Herhangi bir fiziksel bilgiyi taşıyan bir elektriksel işareti çok boyutlu bir desen kümesi olarak ifade etmek mümkündür.

Diğer işaretlerde olduğu gibi, görüntü bilgisi taşıyan desenlerde de sınıflandırma işlemi yapmak için desene ait özniteliklerin (features) çıkarılması gerekir. Desene ait verilerden hangilerinin öznitelik olarak seçileceği ilgilenilen probleme bağlıdır. Genelde öznitelik olarak ham veri kümesinin belli elemanlarının seçilmesi yerine, bu değerlerin fonksiyonu olan , şekil hakkında anlamlı bilgiler ifade eden parametreler kullanılır. Bu çalışmada yaprak görüntülerine ait öznitelikler çıkarılırken, yaprak kenarındaki her bir noktanın yaprak merkezine olan uzaklığı kullanılmıştır.

İki boyutlu görüntülerde, görüntü desenini oluşturan veriler bir matris olup her bir pikseldeki parlaklık değerini veya renk bilgisini ihtiva ederler. Siyah-beyaz görüntülerde yalnızca parlaklık bilgisi işlenirken, renkli görüntülerde resmin kırmızı,yeşil ve mavi temel renklerine ait üç ayrı matris işlenir.

YSA ile desen sınıflandırma tekniklerine geçmeden önce, uygulamalarda yaygın olarak kullanılan istatistiksel desen tanıma yöntemlerinden biri olan Bayes Sınıflandırma yönteminden bahsetmek faydalı olacaktır.

3.1.1. Bayes Sınıflandırma Yöntemi

Bir desene ait öznitelik vektörü X olsun. Bu desenin w_1 ve w_2 sınıflarından hangisine ait oladuğu araştırılsın. Her bir sınıfa benzerlik oranı $q_1(X)$ ve $q_2(X)$ olsun. Bu durumda Bayes Sınıflandırıcı aşağıdaki gibi tanımlanır.

$$q_1(X) \underset{w_2}{\overset{w_1}{\gtrless}} q_2(X) \tag{3.1}$$

Bu eşitliğe göre eğer $q_1(X) > q_2(X)$ ise X deseni w_1 sınıfına aittir. Aksi halde yani, $q_1(X) < q_2(X)$ ise X deseninin w_2 sınıfına ait olduğuna karar verilir. Bu yöntemdeki risk ise ;

$$r^*(X) = \min[q_1(X), q_2(X)] \tag{3.2}$$

şeklinde tanımlanır. Toplam hata büyüklüğü ise eşitlik (3.3) de verilmiştir.

$$\varepsilon^*(X) = E\{r^*(X)\} = P_1 \int_{\Gamma_2} p_1(X)dX + P_2 \int_{\Gamma_1} p_2(X)dX \tag{3.3}$$

Her bir sınıf bölgesi için hata değeri ise

$$\varepsilon_1(X) = \int_{\Gamma_2} p_1(X)dX \qquad \text{ve} \qquad \varepsilon_2(X) = \int_{\Gamma_1} p_2(X)dX \tag{3.4}$$

olacakır. Bu eşitliklerde kullanılan Γ_1 ve Γ_2 sırasıyla w_1 ve w_2 sınıflarına ait bölgeleri, P_i , i. sınıfın olasılığını, p_i i. sınıfın koşullu yoğunluk fonksiyonunu $q_i(X)$ ise X deseninin i. sınıfa ait olma olasılığını göstermektedir. Bu büyüklüklerin tanımı aşağıdaki gibidir.

$$p(X) = \sum_{i=1}^{L} P_i p_i(X) \tag{3.5}$$

yoğunluk fonksiyonu olmak üzere;

$$q_i(X) = P_i \; p_i(X) \; p(X) \tag{3.6}$$

3.2. Yaprak Görüntülerinin Özniteliklerinin Çıkarılması

Bundan önceki bölümlerde bahsedildiği gibi bir görüntünün YSA' ya sunulmasından önce görüntüye ait özniteliklerin belirlenmesi gerekir. Görüntüyü YSA' da temsil eden görüntüye ait öznitelik vektörü olacaktır. Bazı uygulamalarda öznitelik çıkarma işlemi de YSA tarafından yapılmaktadır. Böylece iki katlı bir sistem tasarlanması gerekir. Birinci ağ öznitelik vektörlerini çıkarırken ikinci ağ sınıflandırma işlemini gerçekleştirir. Bu çalışmada, öznitelikler YSA ile değil doğrudan şeklin geometrisinden çıkarılmıştır. YSA sadece sınıflandırıcı olarak kullanılmıştır.

Özniteliklerin çıkarılmasında önce eşik değeri küçük tutulmuş Sobel kenar tanıma algoritmasıyla yapraklardaki kenarlar bulunmuştur. Kenar görüntüsüne ait veriler A matrisinde tutulur. Geliştirilen bir yöntemle kenar görüntüsünden, yaprak çevresini çevreleyen sınır görüntüsü elde edilmiştir. Burada sözü edilen sınır, yaprak ile zemin görüntüsünün ara kesitidir. Bu görüntü B matrisinde tutulur. Böylece görüntüde yaprağın bulunduğu bölge belirlenmiş olmaktadır. Bu görüntü sınır piksellerinin değeri '1' olan diğer piksellerin değeri '0' olan ikili (binary) bir görüntüdür. Başka bir deyişle görüntüdeki '1' lerin sayısı yaprağın çevre uzunluğunu vermektedir. Bir sonraki aşamada, görüntüde sınır içerisinde kalan yaprağa ait bölgedeki bütün piksellere '1' değeri verilip zemine ait piksellere '0' değeri verilir. Bu görüntüdeki '1' değeri içeren piksellerin sayısı yaprağın alanını vermektedir. Alan görüntüsüne ait piksel değerleri C isimli bir matriste tutulur. Yaprağın alanı bulunduktan sonra yaprak görüntüsünün ağırlık merkezi ve simetri eksenleri bulunur. Herhangi bir yaprağa ait verilerin görüntünün dönmesinden etkilenmemesi için, yalnızca en dış kenar gösteren B matrisine ait görüntü simetri eksenlerine göre döndürülür. Böylece simetri eksenleri, aynı zamanda yeni oluşturulan döndürülmüş görüntünün yatay ve düşey eksenleri olur. Bu son görüntü E matrisinde tutulur. Yaprak çevresini gösteren bu görüntüdeki her bir sınır pikselinin, ağırlık merkezine

olan uzaklığı ve yatay simetri ekseni ile yaptığı açı bulunur. Bulunan bu uzaklık değerleri görüntüdeki en büyük uzaklık değerine bölünerek normalize edilmiş uzaklık değerleri elde edilmiş olur. Böylece özniteliklerin yaprak görüntüsünün büyüklüğünden bağımsız olması sağlanır. Öznitelikler oluşturulurken 2° lik adımlarla uzaklık değerlerinden örnekler alınır. Böylece her bir yaprak için 180 adet öznitelik çıkarılmış olur.

Başlangıçta her bir türden 10 adet olmak üzere 10 türe ait yapraklar tarayıcı ile taranarak, elde edilen görüntüler 256 gri seviyeli BMP formatında kaydedilmiştir. Öznitelik çıkarma işlemi sırasında her bir dosyadaki görüntüye ait veriler önce *ARES* isimli bir matrise atanmıştır. Bu işlem *bmpokuma.m* isimli program yazılarak yapılmıştır. Daha sonra *ARES* matrisi kullanılarak, kenar tanıma işleminden başlayarak yukarıda anlatılan işlemler sırasıyla yapılarak öznitelikler çıkarılmıştır. Bu sırada kullanılan bütün yazılımlar *yaprakoku.m* dosyası altında toplanmıştır.

Bundan sonraki bölümde kenar tanıma algoritmalarından kısaca bahsedilecektir.

3.2.1. Kenar Tanıma Yöntemleri

Bir görüntünün en belirgin özelliklerini kenarlarından elde etmek mümkündür. Özellikle en dış kenarı, yani cismin sınırını belirleyen sınır kenarı, şekil tabanlı bir sınıflandırma işlemi yapılıyorsa, kesinlikle belirlenmesi gerekir. Kenarlar, görüntüde parlaklık fonksiyonun ani değişme gösterdiği noktalardan geçer. Kenarların belirlenmesinde yaygın olarak kulanılan beş farklı yöntem vardır.

1. Türev Yöntemleri
2. Yöresel İstatistik Yöntemi
3. Filtreleme Yöntemi
4. Stokastik Gradient Yöntemi
5. Morfolojik Yöntem

Bu yöntemlerden özellikle türev yöntemi üzerinde durulacaktır.

3.2.1.1. Türev Yöntemi

Kenarları bulmak için en etkili yöntemlerden birisi ani gri seviyesi değişimlerini tesbit etmektir.İki boyutlu, 256 gri seviyeli bir görüntü desenini temsil eden G matrisinin boyutları mxn olsun. $G(i,j)$ değeri görüntünün i. satır ve j. sütundaki piksele ait parlaklık değerini içerir. Bu fonksiyonun birinci türevi kenar bölgelerinde yöresel maksimumlar yapar. Bu nedenle görüntünün gradyenti alınarak uygun bir eşik değerinden geçirilirse kenar noktaları bulunmuş olur. Bu Gradient yöntemidir. Diğer bir yöntem ise bu fonksiyonun ikinci türevini alarak sıfır geçişlerinin bulunmasına dayanır. Bu yönteme Laplasyen yöntemi adı verilir. Bir diğer yöntem ise pencere operatörlerinin kullanıldığı Sobel, Prewitt ve Robert yöntemleridir.Yaprak desenlerinin kenarlarının bulunmasında Sobel maskesi kullanılmıştır.

3.2.1.2. Gradient Yöntemi:

Parlaklık bilgisini içeren G matrisinin gradienti aşağıdaki gibi tanımlanır.

$$\nabla G(i,j) = \frac{\partial G(i,j)}{\partial x}\cos\theta + \frac{\partial G(i,j)}{\partial y}\sin\theta \qquad (3.6)$$

Burada θ gri seviyelerinin değişim açısıdır. Bu ifadeyi sıfır yapan değer bir yöresel maksimum değeri olacağından, bu denklemin sıfıra eşitlenmesi kenar büyüklüğünü ve kenar yönünü verir. Buradan gradientin büyüklüğü;

$$(\nabla G(i,j))_{\max} = \sqrt{\frac{\partial G(x,y)^2}{\partial x}} + \sqrt{\frac{\partial G(x,y)^2}{\partial y}} \qquad (3.7)$$

ve gradient yönü;

$$\theta_g = \tan^{-1}\left(\partial G_y / \partial G_x\right) \qquad (3.8)$$

olacaktır.

Bu hesaplanan gradientin değeri belli bir eşik değerinden büyükse ilgili nokta kenar noktası olarak alınır. Uygun eşik değeri maksimum gradientin %5' i ile %10' u arasında seçilmelidir. Yukarıdaki ifadeler sayısal görüntüler için aşağıdaki biçime dönüşür [Gonzalez R.,1974; Fu K.S.,1985].

$$\frac{\partial G(i,j)}{\partial x} = (G(i+1,j+1) - G(i-1,j+1) + G(i+1,j) - G(i-1,j) + G(i+1,j-1) - G(i-1,j-1) \tag{3.9}$$

$$\frac{\partial G(i,j)}{\partial y} = (G(i+1,j+1) - G(i+1,j-1) + G(i,j+1) - G(i,j-1) + G(i-1,j+1) - G(i-1,j-1) \tag{3.10}$$

Yukarıdaki fark denklemlerinin $G(i,j)$ fonksiyonu ile impuls cevabı $h(i,j)$ olan filtrelerin konvolüsyonu olduğu görülmektedir. Buradaki $h(i,j)$ filtreleri yönlü veya yönsüz kenar operatörleridir. Bu filtrelere maske, konvolüsyon işlemine de maskeleme denilmektedir.

Yönlü kenar maskelerinin her biri ancak kendi doğrultularındaki kenarları tanımakta, diğer yönlerdeki kenarları tanıyamamaktadırlar. Yatay, düşey ve çapraz yöndeki kenarlara duyarlı olan 4 farklı yönlü kenar maskesi tanımlıdır [Erdön Altuğ,1992; Fu,1985].

Yönsüz kenar operatörleride fark denklemleri yaklaşımıyla elde edilebilir. Yönsüz kenar operatörlerinde hem x hem de y olmak üzere iki ayrı filtre ile konvolusyon işlemi uygulanır. Bu konvolusyonların kareleri toplamının karekökü, gradientin mutlak değerini verir. Sobel, Prewitt ve Robert maskeleri bu maskelerdendir. Yaprak kenarlarının bulunmasında sobel maskesi kullanılmıştır. Bütün kenar noktalarının görünmesi için eşik değeri küçük tutulmuştur. Büyük eşik değerlerinin seçilmesi durumunda bazı kenarların kaybolduğu gözlenmiştir. Her yöntemde yatay ve düşey olmak üzere iki maske kullanılmaktadır. Bu yöntemlerde resmin her yöndeki kenarlarının bulunması amaçlanmaktadır. Ancak Robert maskesinin çapraz yöndeki kenarlara daha duyarlı olduğu gözlenmektedir. Fakat resimleri bir bütün olarak ele aldığımızda Sobel maskesinin daha etkili olduğu

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

bilinmektedir. Aslında bu çalışmada asıl amaç yaprağı çevreleyen dış kenarın bulunmasıdır. Dolayısıyla içteki kenarlarla ilgilenilmemiştir. Eşik değerinin küçük tutulmasındaki maksat en dış kenarın kopuk kopuk değil de sürekli olarak bulunmasını sağlamaktır. Yatay ve düşey Sobel, Prewitt ve Robert maskeleri aşağıda gösterilmiştir.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.11}$$

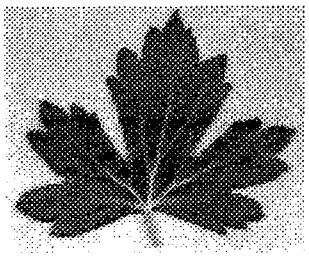
$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.12}$$

$$R_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad , \quad R_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.13}$$

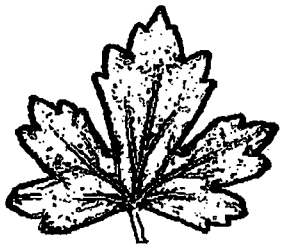
Kenar tanıma işlemi için önce *bmpokuma.m* dosyasındaki *sobel1.m* fonksiyonu işletilerek *ARES* görüntü matrisi ile sobel maskelerinin konvolüsyonu bulunur. Sonuçlar y matrisine atanır. Bundan sonraki aşamada *esik.m* program parçaçığı işletilerek önce y matrisi ile aynı boyutta bir A sıfır matrisi oluşturulur. y matrisinin elemanları eşik değeri ile karşılaştırılarak, bu değerden daha büyük değerler alan elemanların satır ve sütun numaralarına A matrisinde karşılık düşen elemanlar '1' yapılır. Böylece A matrisi sıfır ve birlerden oluşan, bir olan elemanları kenarlara karşılık düşen bir matris olarak elde edilir. Çalışmada kullanılan bir akasya yaprağına ait ARES matrisi görüntüsü (Şekil 3.1.a) ve A matrisine ait kenar görüntüsü (Şekil 3.1.b) aşağıda gösterilmiştir.

3.2.2. ZEMİN -YAPRAK SINIRININ BULUNMASI

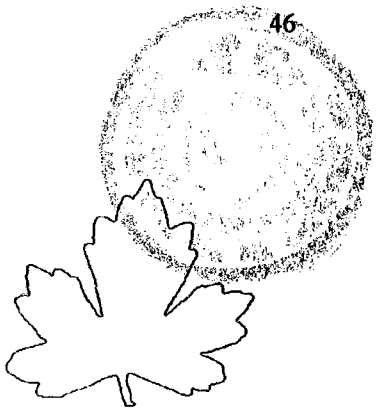
Kenarların bulunmasından sonra cismin en dış kenarı yani yaprak görüntüsü ile zeminin arakesitini oluşturan kenarın bulunması gereklidir. Aslında kenar tanıma



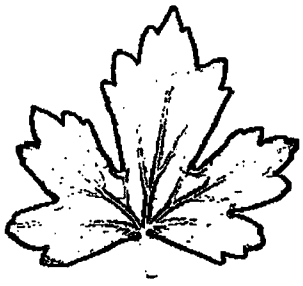
(a) Orijinal görüntü



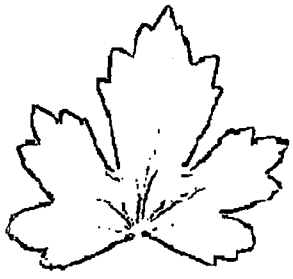
(b) eşik=0.1



(c) Sol el yöntemi ile elde edilen cisim-zemin sınırı



(d) eşik=0.3



(e)- eşik=1

Şekil 3.1. Değişik eşik değerlerinde yaprak kenar görüntüleri

işlemi sırasında eşik değer büyük tutulursa iç kenarlar elimine edilerek en dış kenar tek başına elde edilebilir. Ancak bu yöntem her zaman doğru değildir. Bu durumda elde edilen sınır sürekli olmayabilir. Çünkü bazı iç kenar noktalarına ait gradientler , bazı sınır piksellerine ait gradientlerden büyük olur. Bu nedenle en dış kenarı sürekli olacak şekilde elde etmek şartıyla, iç kenarları tamamen yok etmek mümkün değildir. Bu durum yukarıda değişik eşik seviyeleri için elde edilmiş frenk üzümü yaprağına ait kenar görüntülerinde görülmektedir.(Şekil 3.1.b,d,e).

Şekilde *b* ve *d* görüntülerinde dış kenar sürekliliğini korumasına rağmen iç kenarlar tamamen kaldırılamamıştır. Daha büyük bir eşik değeri için elde edilen *e* görüntüsünde ise dış kenarda süreksizlik noktaları görülmeye başlamasına rağmen iç kenar noktaları halen daha varlıklarını korumaktadırlar. En son görüntü ise bu çalışma sırasında geliştirilen, cismin çevresinin bulunmasında son derece etkili sol el yöntemiyle elde edilmiştir. Görüldüğü gibi hiç bir iç kenar noktası mevcut değildir ve dış kenarda süreksizlik söz konusu değildir. Bu bölümde bu yöntem üzerinde durulacaktır.

3.2.3. Sol ve Sağ El Yöntemleri

Yaprağı çevreleyen en dış kenarın bulunması için ilk olarak, cisim-zemin sınırı üzerinde bulunan bir piksel başlangıç pikseli olarak seçilmiştir.Bu pikselin bulunması için kenarları içeren resmin sol üst köşesinden itibaren ilk kenar pikseli bulununcaya kadar, yani matriste ilk '1' değeri bulununcaya kadar kenar görüntüsü matrisinin birinci satır ve birinci sütunundan itibaren matrisin elemanları taranmıştır. Bulunan bu başlangıç pikseli içerdiği değer '1' olmasından dolayı bir kenar pikseli olduğu gibi aynı zamanda, cisim-zemin sınırına ait bir pikseldir. Bu nokta aynı zamanda cisim-zemin sınırının görüntünün üst kenarına en yakın noktasıdır. Algoritma başlangıcında bütün elemanları '0' olan, A matrisinin boyutunda bir B matrisi oluşturulmuştur. B matrisi cisim-zemin sınırının görüntüsünü içeren bir matris olacaktır. Algoritma sonunda B matrisi cisim-zemin sınırına ait olduğu belirlenen piksellere karşılık düşen elemanları '1' diğer elemanları '0' olacak şekilde elde edilecektir. İlk olarak başlangıç pikselinin A matrisindeki bulunduğu yerin satır ve sütun numaraları bulunur ve B matrisinde aynı satır ve sütun numaralı elemanın değeri '1' yapılır. Aynı işlem daha sonraki aşamalarda da cisim-zemin sınırı üzerinde bulunan pikseller içinde uygulanacaktır. Böylece B matrisinde değeri '1' olan elemanlar cisim-zemin sınırını gösterecektir. Buraya kadar anlatılan işlemler *basla1.m* dosyasındaki yazılımla gerçekleştirilmiştir.

Başlangıç pikselinden itibaren cisim-zemin sınırına ait pikseller ardışıl olarak elde edilecektir. Bu noktada algoritma boyunca esas alınacak bir seçim yapmak gereklidir.

1. Tarama yönünün cismin görüntüsü sol elin içerisinde kalacak şekilde seçilmesi

2. Tarama yönünün cismin görüntüsü sağ elin içerisinde kalacak şekilde seçilmesi

3.2.3.1. Sol El Yöntemi

Herhangi bir k adımında tesbit edilen cisim-zemin sınırına ait P_K pikselinden sonraki komşu sınır pikseli P_{K+1}' i bulmak için, yön bilgisine ihtiyaç vardır. Bu yön

bilgisi P_K pikselinin P_{K-1} pikseline göre hangi konumda olduğunu göstermektedir. Bir piksele 1-komşuluğundaki piksellerin sayısı 8 dir.Dolayısıyla 8 farklı yön bilgisi mümkündür.

7 Sol üst	8 Üst	1 Sağ üst
6 Sol yan	P_{K-1} Pikseli	2 Sağyan
5 Sol alt	4 Alt	3 Sağalt

Şekil 3.2. P_{K-1}. piksele komşu piksellerin yön bilgileri

Cisim-zemin sınırını kullanarak yapılan görüntü tanıma uygulamalarında kullanılan zincir kodunda olduğu gibi, her komşu pikselin bulunduğu yeri belirlemek için 1-8 arasında kodlama işlemi yapılır. Zincir kodunda her bir sınır pikseli bir önceki sınır pikseline göre bulunduğu konumun numarasını alır .

P_{K+1}. Pikselin bulunması için P_K nın P_{K-1} 'e göre bulunduğu konuma göre 8 değişik arama yönteminden birisi seçilir.

1. Pk için yönbilgisi sağüst yani 1 ise;

P_{K+1} in aranmasına A matrisinde P_K nın 7 numaralı sol üst komşusundan başlanır.Komşu piksellerden 1 değerine sahip piksel bulununcaya kadar sırasıyla 7-8-1-2-3-4 numaralı pikseller taranır. '1' değerini içeren piksel bulununca arama durdurulur.Bu piksele ait yön bilgisi ,yön bilgisini tutan değişkene aktarılır.Pikselin bulunduğu satır ve sütun numaraları alınarak B matrisinde bu satır ve sütun numarasına sahip eleman '1' yapılır.

2. P_K yönbilgisi, sag yan yani 2 ise;

P_{K+1} in aranmasına A matrisinde P_K nın 8 numaralı üst komşusundan başlanır. 8-1-2-3-4-5 sıralamasına göre arama yapılır. 1.maddede P_{K+1} pikseli bulunduğunda B matrisi ve yönbilgisi değişkeni üzerinde yapılan işlemler yapılarak yönbilgisi değişkeni güncelleştirilir, B matrisinde ilgili elemanın değeri '1' yapılır.

Görüldüğü gibi her adımda P_{K+1} in aranması farklı bir sıralamaya göre olmaktadır. Her bir durum için takip edilmesi gereken sıralama Tablo 1 de verilmiştir. Yukarıda da belirtildiği gibi her arama işleminde 1 değeri içeren piksel bulunduğunda, bu pikselin satır ve sütun numaraları alınarak B matrisinde bu satır ve sütunda bulunan eleman 1 yapılır. Yön bilgisi de her adımda güncellenir. Başlangıç pikseline tekrar ulaşıldığında cisim-zemin sınırına ait bütün pikseller bulunmuş demektir. Arama işlemi durdurulur. Bu yöntemde cisim-zemin sınırındaki hareket yönü sol elin parmaklarının gösterdiği yön olur ve cisim devamlı sol elin iç tarafında kalır.

Y_K	YÖN BİLGİSİ	ARAMA SIRASI
1	SAĞÜST	7 - 8 - 1 - 2 - 3 - 4
2	SAGYAN	8 - 1 - 2 - 3 - 4 - 5
3	SAGALT	1 - 2 - 3 - 4 - 5 - 6
4	ALT	2 - 3 - 4 - 5 - 6 - 7
5	SOLALT	3 - 4 - 5 - 6 - 7 - 8
6	SOLYAN	4 - 5 - 6 - 7 - 8 - 1
7	SOLÜST	5 - 6 - 7 - 8 - 1 - 2
8	ÜST	6 - 7 - 8 - 1 - 2 - 3

Tablo 3.1. P_K pikseline ilişkin yön bilgisine göre P_{K+1} pikselinin bulunması için izlenmesi gereken arama sırası

Tablodan görüldüğü gibi P_{K+1}. pikseli bulmak için P_K. pikselin etrafında altı piksel aranır. Diğer iki komşu pikselden birisi P_{K-1}. pikseldir.Diğer piksel için ise iki durum söz konusudur.

a-) Bir içkenar pikseli olabilir ama sınır pikseli değildir.

b-)Eğer sınır pikseli ise k. adımdan önceki adımlarda sınır pikseli olarak zaten bulunmuştur.

Dolayısıyla bu iki pikselin kontrol edilmesine gerek yoktur. Bu bir örnekle gösterilirse;

k. adımda P_K pikseli bulunmuş olsun ve buna ilişkin yön bilgisi Y_K=1(sağüst) olsun. Bu durumda cisim-zemin sınırındaki arama yönü cismi sol elin içinde bırakacak şekilde olması için, P_{K+1} için arama sırası 7-8-1-2-3-4 olacaktır. P_K in yönbilgisi 5 (solalt) olan komşu pikseli bir önceki aşamada bulunan P_{K-1} pikselidir.Yönbilgisi 6 (alt) olan piksel ise; Eğer bir kenar pikseli ise, P_{K-1} den P_K 'a doğru giderken sol elin dış tarafında kaldığından, k. adımdan daha önce tesbit edilmiş bir sınır pikselidir. Aksi düşünülürse yani bu piksel bir sınır pikseli olmasına rağmen daha önce tesbit edilmemiş olsa, algoritmanın yapısına göre Pk-1 den sonra PK pikseli değil bu piksel k+1. adımda bulunacak sınır pikseli olacaktı.O halde böyle bir durum sözkonusu değildir.Yani bu nokta cisim-zemin sınırına ait değildir.Algoritmanın daha hızlı çalışması açısından ve yukarıdaki gerekçelerden bu iki piksel arama dışı bırakılmıştır. (Şekil 3.3).

7	8	1
6	P_K	2
5.P_{K-1}	4	3

Şekil 3.3. Y_K=1 durumu

O halde herhangi bir k adımında bir Y_K yönbilgisi için P_{K+1} in aranmasına başlanması gereken pikselin P_K ya göre yönbilgisi Y_{K+1}bas;

$$\mathbf{Y_{K+1}bas = Y_K + 6 \bmod (8)} \qquad (3.14)$$

olmalıdır.

Kenar pikseli bulununcaya kadar yönbilgisi her defasında 1 artılarak yapılan aramada en son aranacak piksel yönbilgisi Y_{K+1} son;

$$\mathbf{Y_{K+1}son = Y_K + 3 \ mod(8)} \qquad (3.15)$$

olan pikseldir.

Eğer bu pikselde sınır pikseli değilse bu noktada cisim-zemin sınırının süreksizlik gösterdiği anlaşılır. Kenar tanıma algoritması işletilirken eşik değer düşük tutulursa böyle bir durumla karşılaşılmaz.

Çalışma sırasında sol el yöntemi kullanılmıştır. Bu yöntem *asil2.m* dosyasındaki yazılımla gerçekleştirilmiştir.

3.2.3.2 Sağ El Yöntemi

Sağ el yönteminde ise cisim-zemin sınırı tesbit edilirken, cisim sağ elin iç tarafında kalacak şekilde sınır çizgisi boyunca hareket edilir. Dolayısıyla P_{K+1} bulunurken izlenmesi gereken sıralama sol el yöntemine göre farklıdır.

Diğer işlemler sol el yöntemindekilerle aynıdır. Tablo 3.2.' de Y_K yönbilgisine göre takip edilmesi gereken arama sırası gösterilmiştir.

3.2.4 Yaprak Alanının Bulunması

Yaprak alanının bulunması için yaprak sınırı içindeki bütün piksellerin '1' yapılarak bu piksellerin sayısının bulunması gereklidir. Bu işlemde yine sol el yönteminden istifade edilerek yapılmıştır. Bu sırada kullanılan program kodu *asil2.m'* e benzeyen *asil3.m* programıdır.

Yaprağın alanı C matrisindeki '1' değerine sahip elemanların sayısına eşittir. Buraya kadar anlatılan A, B ve C matrislerinin boyutları eşittir. C matrisinin elde edilmesinde B matrisinden faydalanılır ve *alanl.m* isimli program kodu çalıştırılır. Bu görüntünün bulunmasından sonra yaprak alanı ve şeklin ağırlık merkezi

bulunabilir. Matrisin satır sayısı m, sütun sayısı n olmak üzere yaprak alanı aşağıdaki gibi tanımlanır.

Y_K	YÖN BİLGİSİ	ARAMA SIRASI
1	SAĞÜST	3 - 2 - 1 - 8 - 7 - 6
2	SAGYAN	4 - 3 - 2 - 1 - 8 - 7
3	SAGALT	5 - 4 - 3 - 2 - 1 - 8
4	ALT	6 - 5 - 4 - 3 - 2 - 1
5	SOLALT	7 - 6 - 5 - 4 - 3 - 2
6	SOLYAN	8 - 7 - 6 - 5 - 4 - 3
7	SOLÜST	1 - 8 - 7 - 6 - 5 - 4
8	ÜST	2 - 1 - 8 - 7 - 6 - 5

Tablo 3.2. Sağ el sınır tanıma yöntemine göre P_K pikseline ilişkin yön bilgisine göre, P_{K+1} pikselinin bulunması için izlenmesi gereken arama sırası

Matrisin satır sayısı m, sütun sayısı n olmak üzere yaprak alanı aşağıdaki gibi tanımlanır. Matrisde yaprak içerisine dahil elemanların değerleri '1', diğer elemanlar '0' olduğundan, yaprak alanı matrisin bütün elemanlarının toplamıdır. Dolayısıyla alan ifadesi;

$$alan = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} \qquad \text{olacaktır.} \tag{3.16}$$

Ağırlık merkezinin koordinatları ise ;

$$satmerk = \frac{1}{alan}\sum_{i=1}^{m}\sum_{j=1}^{n} iC_{ij}\,, \quad \text{ve} \quad sutmerk = \frac{1}{alan}\sum_{i=1}^{m}\sum_{j=1}^{n} jCij \tag{3.16}$$

ifadeleri ile tanımlanır.

Alanın ve ağırlık merkezinin satır ve sütun numaralarının bulunması *agmerkz.m* programıyla gerçekleştirilmiştir.

3.3. Simetri Eksenleri

Görüntü tanıma uygulamalarında, görüntülerin oryantasyon, boyut ve öteleme işlemlerinden bağımsız olacak şekilde modellenmesi en çok dikkat edilmesi gereken husustur. Aksi taktirde aynı görüntünün farklı pozisyonlardaki görünüşleri cismin doğru sınıflandırılmasına engel olur. Çünkü bu durumda görüntü matrisi farklı olacağından,orijinalinden tamamen farklı özniteliklerle karşılaşmak sözkonusu olacaktır.

Bu nedenle yaprak görüntülerinin modellenmesi, görüntünün simetri eksenlerine göre yapılmıştır. Böylece yaprak hangi pozisyonda bulunursa bulunsun özniteliklerin değeri ve sırası değişmemektedir.

Simetri eksenlerinin bulunmasında *simeksen.m* isimli program kullanılmıştır. Görüntünün mevcut pozisyondaki yatay ekseniyle simetri eksenlerinin yapmış olduğu açı bulunarak, bir sonraki *cevir23.m* programıyla B matrisindeki kenar görüntüsü bu açı kadar döndürülerek E matrisi elde edilmiştir. Bu işlemler sırasında kullanılan bağıntılar aşağıda verilmiştir.

$$cmi = \sum_{i=1}^{m}\sum_{j=1}^{n}\left((satmerk - i)^2 + (j - sutmerk)^2\right)C_{ij} \tag{3.17}$$

$$m_{02} = \sum_{i=1}^{m}\sum_{j=1}^{n}(satmerk - i)^2 . C_{ij} \tag{3.18}$$

$$m_{20} = \sum_{i=1}^{m}\sum_{j=1}^{n}(j - sutmerk)^2 . C_{ij} \tag{3.19}$$

$$m_{11} = \sum_{i=1}^{m}\sum_{j=1}^{n}((satmerk - i)(j - sutmerk)) C_{ij} \tag{3.20}$$

$$tg^2\theta + \frac{m_{20} - m_{02}}{m_{11}} tg\theta - 1 \tag{3.21}$$

denkleminin kökleri simetri eksenlerinin yatay eksenle yapmış olduğu açıyı vermektedir. B matrisine ait görüntü θ kadar döndürülürse simetri eksenleri yatay ve düşey eksenle çakışmış olur. Bunun için önce B matrisindeki '1' değerine sahip dış kenar pksellerinin her birinin ağırlık merkezine olan uzaklıkları ve yatayla yapmış oldukları açı bulunarak, dış kenardaki noktalar kutupsal koordinatlarda modellenir. Dış kenarın bulunması sırasında çalıştırılan *asil2.m* programı dış kenar piksellerinin satır numaralarını *SATIR* vektöründe, sütun numaralarını ise *SUTUN* vektörüne kaydetmektedir. Bu vektörlerin boyu aynı zamanda çevre üzerindeki piksel sayısına eşittir. Çevre uzunluğu için bir ölçek olarak alınabilir. Bu vektörlerden yararlanarak *acigen.m* programı ile hesaplanan uzunluklar *UZUNLUK* matrisine, simetri eksenini baz olarak alan açılar ise *TETA4* matrisine atanır.

$$UZAKLIK(i) = \sqrt{\left((SATIR(i) - satmerk)^2 + (SUTUN(i) - sutmerk)^2\right)} \tag{3.22}$$

$$TETA4(i) = atan2((satmerk - SATIR(i)),(SUTUN(i) - sutmerk)) - teta1 \tag{3.23}$$

Eşitlikteki $atan2(\bullet)$ fonksiyonu $[0,\ 2\pi]$ aralığında değerler alan özel arctangent fonksiyonudur. Diğer terim $teta1$ ise denkleminin pozitif köküdür.

Bu sonuçlar elde edildikten sonra şeklin döndürülmesi mümkündür. Dönme işleminde sonraki elde edilecek görüntünün dış kenar piksellerinin satır ve sütün numaralarını kaydetmek üzere $SATIR1$ ve $SUTUN1$ vektörleri tanımlanır.

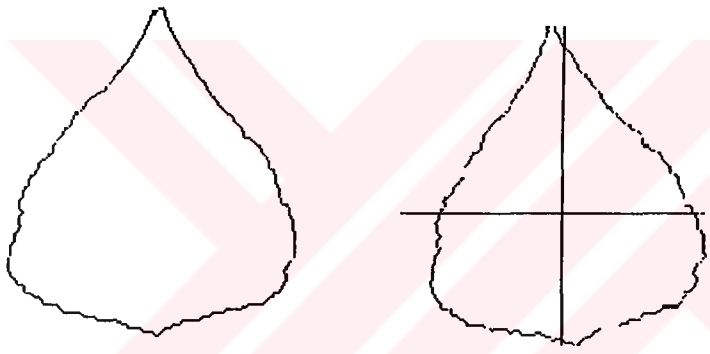
$$SATIR1(i) = UZAKLIK(i) * \cos(TETA4(i)) \tag{3.24}$$

$$SUTUN1(i) = -UZAKLIK(i) * \sin(TETA4(i)) \tag{3.25}$$

Bu vektörler vasıtasıyla döndürülmüş görüntüye ait başlangıçta bütün elemanları sıfır olan E matrisi aşağıdaki bağıntı ile güncellenir.

$$E(SATIR1(i) + satmerk, SUTUN1(i) + sutmerk) = 1 \tag{3.26}$$

Matrisin indislerinin tamsayı olmaması durumunda, tamsayıya yuvarlatma işlemi yapılır.



Şekil 3.4. Simetri eksenlerine göre döndürülmüş yaprak görüntüsü

3.4. Özniteliklerin Çıkarılması

Elde edilen *UZUNLUK* ve *TETA4* matrisindeki elemanların sıralanışı dış kenarın bulunmasında izlenen sıraya göredir. Uzunluklar *TETA4* matrsindeki açı değerlerine göre küçükten büyüğe doğru, 2° lik adımlarla sıralanır. Her aralıktaki maksimum uzuluğa sahip piksel örneklenir.Bu amaçla *egitim* matrisinin birinci sütunu kullanılır. Matrisin diğer sütunlarında ise her bir açı derecesine karşı düşen minimum uzunluklar ile nokta sayısı kaydedilir. Ancak bu sütunlardaki verilere

sınıflandırma aşamasında ihtiyaç duyulmamıştır. Maksimum uzunluklar sınıflandırma işleminde yeterli olmuştur.

3.5. Sınıflandırma İşlemi

Sınıflandırma işlemine başlamadan önce bütün yapraklar okutularak öznitelikler çıkarılmış ve kaydedilmiştir. Daha sonra altı değişik sınıftan birer örnek seçilerek YSA eğitilmiştir. Geriye yayılma (back propagation) ve Hopfield ağlarında sınıflandıma işlemi yapılmıştır. Test işleminde on türe ait yaprak görüntüleri kullanılmıştır.

Mevcut bütün yaprak görüntülerine ait öznitelikler *vermatris.m* programı işletilerek her bir türe ait, türün kendi adıyla anılan *akasya, akca, citlen, gul, igde, kagac, kartopu, kkavak, kusburnu, leylak, fruz* öznitelik matrisleri elde edilmiştir. Bu matrislerin her bir kolonunda değişik bir yaprağa ait veriler bulunmaktadır.

3.5.1. Geriye Yayılma Ağı İle Sınıflandırma

Geriye yayılma ağında giriş katmanı 90, saklı katman 35 çıkış katmanı ise 8 hücreden oluşmaktadır. Saklı katman hücrelerinin sayısı çok sayıda denemeden sonra en optimum olacak şekilde seçilmiştir. Bütün katmanlar için transfer fonksiyonu olarak $\log sig(\bullet)$ fonksiyonu kullanılmıştır. Sigmoid fonksiyonundan daha iyi sonuç verdiği gözlenmiştir.

Ağın eğitiminde adaptif momentum geriye yayılma öğrenme kuralı kullanılmıştır. Bu yöntemin birbirine çok benzeyen desenleri tanımadaki performansının diğer yöntemlere göre çok iyi olduğu gözlenmiştir. Çünkü diğer yöntemlerle yapılan denemelerde doğru sınıflandırma işlemine pek az rastlanıldığından bu uygulama için kullanışsız oldukları görülmüştür. Oysa bu yöntemde % 100 'e yakın doğrulukta bir sınıflandırma gerçekleştirilmiştir.

Eğitim işlemi için geçen süre yaklaşık 4 dakika olmuştur. Giriş kümesinin büyüklüğü ve istenilen hatanın çok küçük değerde tutulması, aynı zamanda sınıf sayısının fazlalığı gözönüne alındığında bu süre çok büyük değildir. Bu ağ modeli ile yapılan eğitim işleminin çıktısı aşağıda gösterilmiştir.

```
» negitim=[takasya(:,5) tkkavak(:,4) tkagac(:,4) tkartopu(:,2) tkusburnu(:,2)];
» yapsin
TRAINBPX: 0/4000 epochs, lr = 0.1001, SSE = 10.4381.
TRAINBPX: 50/4000 epochs, lr = 0.550127, SSE = 4.07991.
TRAINBPX: 100/4000 epochs, lr = 0.253607, SSE = 3.32991.
TRAINBPX: 150/4000 epochs, lr = 0.0285914, SSE = 2.11096.
TRAINBPX: 200/4000 epochs, lr = 0.52449, SSE = 1.60858.
TRAINBPX: 250/4000 epochs, lr = 0.0789458, SSE = 1.0274.
TRAINBPX: 300/4000 epochs, lr = 0.608523, SSE = 0.48994.
TRAINBPX: 350/4000 epochs, lr = 0.16327, SSE = 0.409723.
TRAINBPX: 400/4000 epochs, lr = 0.105566, SSE = 0.202152.
TRAINBPX: 450/4000 epochs, lr = 0.676554, SSE = 0.164203.
TRAINBPX: 500/4000 epochs, lr = 0.133484, SSE = 0.0777113.
TRAINBPX: 550/4000 epochs, lr = 2.80349, SSE = 0.056526.
TRAINBPX: 600/4000 epochs, lr = 0.483124, SSE = 0.0305093.
TRAINBPX: 650/4000 epochs, lr = 13.3004, SSE = 0.0170962.
TRAINBPX: 700/4000 epochs, lr = 2.62416, SSE = 0.0120748.
TRAINBPX: 750/4000 epochs, lr = 5.66345, SSE = 0.00524598.
TRAINBPX: 800/4000 epochs, lr = 11.6351, SSE = 0.00431955.
TRAINBPX: 850/4000 epochs, lr = 3.21971, SSE = 0.00167837.
TRAINBPX: 900/4000 epochs, lr = 48.2134, SSE = 0.00135717.
TRAINBPX: 950/4000 epochs, lr = 10.1784, SSE = 0.000598508.
TRAINBPX: 1000/4000 epochs, lr = 261.879, SSE = 0.000410333.
TRAINBPX: 1050/4000 epochs, lr = 45.1294, SSE = 0.000189735.
TRAINBPX: 1100/4000 epochs, lr = 1329.38, SSE = 0.000104106.
TRAINBPX: 1150/4000 epochs, lr = 214.103, SSE = 5.85266e-005.
TRAINBPX: 1200/4000 epochs, lr = 6306.85, SSE = 2.47077e-005.
TRAINBPX: 1250/4000 epochs, lr = 949.302, SSE = 1.61487e-005.
TRAINBPX: 1300/4000 epochs, lr = 27963.6, SSE = 5.58883e-006.
TRAINBPX: 1350/4000 epochs, lr = 4503.69, SSE = 3.89612e-006.
TRAINBPX: 1400/4000 epochs, lr = 86790.4, SSE = 1.24973e-006.
TRAINBPX: 1450/4000 epochs, lr = 28007.1, SSE = 1.06629e-006.
TRAINBPX: 1500/4000 epochs, lr = 25869.5, SSE = 3.45498e-007.
TRAINBPX: 1550/4000 epochs, lr = 108463, SSE = 2.67119e-007.
TRAINBPX: 1600/4000 epochs, lr = 250470, SSE = 7.27e-008.
TRAINBPX: 1650/4000 epochs, lr = 550592, SSE = 5.83759e-008.
TRAINBPX: 1700/4000 epochs, lr = 332708, SSE = 1.82337e-008.
TRAINBPX: 1750/4000 epochs, lr = 2.44124e+006, SSE = 1.34667e-008.
TRAINBPX: 1800/4000 epochs, lr = 5.63748e+006, SSE = 3.14204e-009.
TRAINBPX: 1850/4000 epochs, lr = 1.326e+007, SSE = 2.48541e-009.
TRAINBPX: 1900/4000 epochs, lr = 1.22479e+007, SSE = 6.60448e-010.
TRAINBPX: 1950/4000 epochs, lr = 6.73117e+007, SSE = 5.00397e-010.
TRAINBPX: 2000/4000 epochs, lr = 2.37602e+008, SSE = 9.95432e-011.
TRAINBPX: 2050/4000 epochs, lr = 4.79245e+008, SSE = 7.81881e-011.
TRAINBPX: 2100/4000 epochs, lr = 7.24013e+008, SSE = 1.59096e-011.
TRAINBPX: 2150/4000 epochs, lr = 3.1889e+009, SSE = 1.20855e-011.
TRAINBPX: 2200/4000 epochs, lr = 1.12564e+010, SSE = 2.12175e-012.
TRAINBPX: 2250/4000 epochs, lr = 1.7321e+010, SSE = 1.64264e-012.
TRAINBPX: 2300/4000 epochs, lr = 5.10225e+011, SSE = 2.50358e-013.
TRAINBPX: 2350/4000 epochs, lr = 1.23322e+011, SSE = 1.69469e-013.
TRAINBPX: 2400/4000 epochs, lr = 3.63269e+012, SSE = 3.39763e-014.
TRAINBPX: 2450/4000 epochs, lr = 1.07008e+014, SSE = 1.06608e-015.
```

TRAINBPX: 2500/4000 epochs, lr = 3.15215e+015, SSE = 3.57112e-017.
TRAINBPX: 2550/4000 epochs, lr = 9.28529e+016, SSE = 1.19156e-018.
TRAINBPX: 2600/4000 epochs, lr = 2.73517e+018, SSE = 3.98699e-020.
TRAINBPX: 2611/4000 epochs, lr = 5.75713e+018, SSE = 1.88878e-020.

Bu işlemin başında tanımlanan *negitim* matrisinde her bir sınıfı temsil eden model yapraklara ait veriler bulunmaktadır. Akasya yaprakları için 5 numaralı yaprak, karakavak ve karaağaç için bu türlere ait 4 numaralı yapraklar, kartopu ve kuşburnu türleri için ise 2 numaralı yapraklar model olarak seçilmiştir. Hedeflenen karesel hata (SSE), $2x10^{-20}$ alınmıştır. Toplam karesel hata değeri bu değerin altına düşünce eğitim işlemi durdurulmuştur. Bu hedefe 2611 eğitim adımı sonunda ulaşılmıştır.

Eğitim işleminde, çıkış desenine ait bilgi ise *nhedef* matrisinde tutulmuştur. Bu matris aşağıda verimiştir.

$$nhedef = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

bu matristeki sütunlar sırasıyla akasya , karakavak, karaağaç, kartopu ve kusburnu türlerine ait çıkış desenleridir.

Eğitim işleminin ardından her bir türden 5 yaprak olmak üzere toplam 10 tür üzerinde yapılan test işlemi sonucunda bulunan çıkış desenleri aşağıda verilmiştir.

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9511	1.0000	1.0000	1.0000	1.0000	1.0000
0.0086	0.0000	0.0000	0.0000	0.0000	0.0000
0.1933	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Tablo 3.3. Akasya Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	0.0132	1.0000	0.9983	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0016	0.0000	0.0000	0.0000
0.0000	0.0000	0.7404	0.0000	0.0000	0.0000
1.0000	1.0000	0.9999	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tablo 3.4. Karakavak Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0000	0.0000	0.0002	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	0.9999	1.0000	1.0000	1.0000
1.0000	1.0000	0.9637	1.0000	1.0000	1.0000
0.0000	0.0000	0.9999	0.0000	0.5476	0.0000
1.0000	1.0000	0.0000	1.0000	0.3131	1.0000

Tablo 3.5. Karaağaç Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	0.0002	0.1627	1.0000
0.0000	0.0000	0.0000	0.9120	0.6217	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	1.0000	0.9869	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tablo 3.6. Kartopu Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	0.9999	1.0000
0.0000	0.0000	0.0000	0.0000	0.0001	0.0000

Tablo 3.7. Kusburnu Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9997	0.9938	1.0000	0.9742	0.9100	
0.0889	1.0000	1.0000	1.0000	0.9990	
1.0000	1.0000	0.0276	1.0000	1.0000	BELİRLİ
0.0000	0.0000	0.0000	0.0001	0.0012	DEĞİL
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	

Tablo 3.8. Akçaağaç Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9995	1.0000	1.0000	0.0001	0.0000	
0.0001	0.0000	0.0000	0.7722	1.0000	
0.0000	0.0000	0.0000	0.9993	1.0000	BELİRLİ
1.0000	1.0000	1.0000	1.0000	1.0000	DEĞİL
0.0000	0.0000	0.0000	0.0000	0.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	

Tablo 3.9. Çitlenbik Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0001	0.0000	0.0000	0.0000	0.0071	
0.9997	1.0000	0.7978	1.0000	0.9999	
1.0000	1.0000	1.0000	1.0000	1.0000	BELİRLİ
1.0000	0.9999	1.0000	1.0000	0.3090	DEĞİL
1.0000	1.0000	0.1706	0.9081	1.0000	
0.0000	0.0000	0.9690	0.3701	0.0000	

Tablo 3.10. Gül Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	BELİRLİ
1.0000	1.0000	1.0000	1.0000	1.0000	DEĞİL
0.0000	0.0000	0.0055	0.0000	0.0000	
1.0000	1.0000	0.9979	1.0000	1.0000	

Tablo 3.11. İğde Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9845	0.0000	0.0000	0.5927	0.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0004	0.5336	1.0000	0.0044	1.0000	BELİRLİ
0.0000	1.0000	1.0000	0.0008	1.0000	DEĞİL
1.0000	0.6890	0.0000	1.0000	0.0000	
0.0000	0.0551	1.0000	0.0000	1.0000	

Tablo 3.12. Leylak Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	1.0000	1.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	BELİRLİ
0.0000	0.0000	0.0000	0.0000	0.0000	DEĞİL
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	

Tablo 3.13. Frenk Üzümü Yaprakları

Test işlemi sonucunda Akasya yaprakları %100, karakavak yaprakları % 80 karaağaç yaprakları %60, kartopu yaprakları %60 kusburnu yaprakları ise %100 doğrulukla sınıflandırılmıştır. Sistemin toplam sınıflama doğruluğu %80 olmaktadır.

Eğitim işlemi sırasında ağa tanıtılmayan diğer 5 türe ait çıkış desenlerine bakıldığında akçaağaç; 011101 çıkış deseni ile karaağaca benzetilmiştir. Çitlenbiğe

ait görüntülerden ilk üç tanesi ;100101 çıkış deseni ile akasyaya benzetilirken, diğer ikisi; 011101 çıkış deseni ile karaağaca benzetilmiştir. Gül ise 011110 çıkış deseni ile kuşburnuya benzetilmiştir. İğde yapraklarının tamamı 100101 çıkış deseni ile akasya yaprakları ile ilişkilendirilmiştir. En zor sınıf leylak olarak göze çarpmaktadır. Leylak yaprakları karaağaç ve karakavak yapraklarına benzetilmiştir. Fren üzümü yapraklarının tamamı ise;110010 çıkış deseni ile karakavak yapraklarına benzetilmiştir.

Model seçiminin öğrenmeye etkisinin gözlenmesi amacıyla ikinci bir eğitim işleminde, akasya türünü temsil etmek üzere 5 numaralı yaprak yerine 2 numaralı yaprak kullanılmıştır. Bu eğitim işleminin ve arkasından yapılan test işleminin sonuçları aşağıda gösterilmiştir.

```
» negitim=[takasya(:,2) tkkavak(:,4) tkagac(:,4) tkartopu(:,2) tkusburnu(:,2)];
» yapsin
TRAINBPX: 0/4000 epochs, lr = 0.1001, SSE = 11.0712.
TRAINBPX: 50/4000 epochs, lr = 1.17942, SSE = 4.23274.
TRAINBPX: 100/4000 epochs, lr = 0.177525, SSE = 3.53758.
TRAINBPX: 150/4000 epochs, lr = 0.0305928, SSE = 2.11941.
TRAINBPX: 200/4000 epochs, lr = 0.842217, SSE = 1.29313.
TRAINBPX: 250/4000 epochs, lr = 0.135644, SSE = 1.10735.
TRAINBPX: 300/4000 epochs, lr = 0.0267625, SSE = 0.693409.
TRAINBPX: 350/4000 epochs, lr = 0.562078, SSE = 0.527322.
TRAINBPX: 400/4000 epochs, lr = 0.0690616, SSE = 0.339255.
TRAINBPX: 450/4000 epochs, lr = 1.77688, SSE = 0.205394.
TRAINBPX: 500/4000 epochs, lr = 0.218323, SSE = 0.168566.
TRAINBPX: 550/4000 epochs, lr = 0.141161, SSE = 0.0975811.
TRAINBPX: 600/4000 epochs, lr = 0.968009, SSE = 0.0824334.
TRAINBPX: 650/4000 epochs, lr = 0.178493, SSE = 0.0459609.
TRAINBPX: 700/4000 epochs, lr = 3.50355, SSE = 0.0355527.
TRAINBPX: 750/4000 epochs, lr = 0.564266, SSE = 0.0223106.
TRAINBPX: 800/4000 epochs, lr = 16.6216, SSE = 0.0125426.
TRAINBPX: 850/4000 epochs, lr = 2.3382, SSE = 0.0105387.
TRAINBPX: 900/4000 epochs, lr = 2.15973, SSE = 0.00584174.
TRAINBPX: 950/4000 epochs, lr = 7.90909, SSE = 0.00491753.
TRAINBPX: 1000/4000 epochs, lr = 1.6697, SSE = 0.00268442.
TRAINBPX: 1050/4000 epochs, lr = 32.7735, SSE = 0.0021402.
TRAINBPX: 1100/4000 epochs, lr = 5.27836, SSE = 0.00121235.
TRAINBPX: 1150/4000 epochs, lr = 135.806, SSE = 0.000820717.
TRAINBPX: 1200/4000 epochs, lr = 19.1042, SSE = 0.000567095.
TRAINBPX: 1250/4000 epochs, lr = 562.752, SSE = 0.000291218.
TRAINBPX: 1300/4000 epochs, lr = 79.1635, SSE = 0.000257612.
TRAINBPX: 1350/4000 epochs, lr = 47.8364, SSE = 0.000139395.
TRAINBPX: 1400/4000 epochs, lr = 286.52, SSE = 0.000113159.
TRAINBPX: 1450/4000 epochs, lr = 60.4874, SSE = 6.27061e-005.
TRAINBPX: 1500/4000 epochs, lr = 1359.31, SSE = 4.77735e-005.
TRAINBPX: 1550/4000 epochs, lr = 218.925, SSE = 2.79239e-005.
TRAINBPX: 1600/4000 epochs, lr = 6026.98, SSE = 1.78721e-005.
TRAINBPX: 1650/4000 epochs, lr = 740.525, SSE = 1.22614e-005.
TRAINBPX: 1700/4000 epochs, lr = 21813.7, SSE = 6.27778e-006.
TRAINBPX: 1750/4000 epochs, lr = 3068.58, SSE = 5.40063e-006.
```

TRAINBPX: 1800/4000 epochs, lr = 3032.77, SSE = 2.71401e-006.
TRAINBPX: 1850/4000 epochs, lr = 12715.5, SSE = 2.34622e-006.
TRAINBPX: 1900/4000 epochs, lr = 2872.29, SSE = 1.14728e-006.
TRAINBPX: 1950/4000 epochs, lr = 52690.3, SSE = 9.20036e-007.
TRAINBPX: 2000/4000 epochs, lr = 9715.69, SSE = 4.82015e-007.
TRAINBPX: 2050/4000 epochs, lr = 249974, SSE = 3.29505e-007.
TRAINBPX: 2100/4000 epochs, lr = 43077.9, SSE = 1.87016e-007.
TRAINBPX: 2150/4000 epochs, lr = 1.26895e+006, SSE = 9.55566e-008.
TRAINBPX: 2200/4000 epochs, lr = 178505, SSE = 6.69374e-008.
TRAINBPX: 2250/4000 epochs, lr = 5.25824e+006, SSE = 2.66185e-008.
TRAINBPX: 2300/4000 epochs, lr = 846868, SSE = 2.21031e-008.
TRAINBPX: 2350/4000 epochs, lr = 1.95565e+006, SSE = 8.74533e-009.
TRAINBPX: 2400/4000 epochs, lr = 4.59989e+006, SSE = 7.26465e-009.
TRAINBPX: 2450/4000 epochs, lr = 1.2729e+006, SSE = 2.58643e-009.
TRAINBPX: 2500/4000 epochs, lr = 2.4985e+007, SSE = 2.02675e-009.
TRAINBPX: 2550/4000 epochs, lr = 5.27461e+006, SSE = 6.99534e-010.
TRAINBPX: 2600/4000 epochs, lr = 1.26832e+008, SSE = 4.95293e-010.
TRAINBPX: 2650/4000 epochs, lr = 2.50239e+007, SSE = 1.39574e-010.
TRAINBPX: 2700/4000 epochs, lr = 6.43837e+008, SSE = 9.92673e-011.
TRAINBPX: 2750/4000 epochs, lr = 1.78165e+008, SSE = 1.95926e-011.
TRAINBPX: 2800/4000 epochs, lr = 4.2841e+009, SSE = 1.43098e-011.
TRAINBPX: 2850/4000 epochs, lr = 1.26197e+011, SSE = 1.19367e-012.
TRAINBPX: 2900/4000 epochs, lr = 3.71738e+012, SSE = 3.76926e-014.
TRAINBPX: 2950/4000 epochs, lr = 1.09503e+014, SSE = 1.27136e-015.
TRAINBPX: 3000/4000 epochs, lr = 3.22563e+015, SSE = 4.27611e-017.
TRAINBPX: 3050/4000 epochs, lr = 9.50175e+016, SSE = 1.4395e-018.
TRAINBPX: 3100/4000 epochs, lr = 2.79893e+018, SSE = 4.83595e-020.
TRAINBPX: 3113/4000 epochs, lr = 6.745e+018, SSE = 1.99531e-020.

Görüldüğü gibi burada eğitim işlemi daha uzun sürmüş ve 3113 adımda sonlanmıştır.

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0293	1.0000	1.0000	1.0000	1.0000	1.0000
0.9887	0.0000	0.0000	0.0000	0.0000	0.0000
0.3926	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Tablo 3.14. Akasya Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	0.0001	1.0000	0.0846	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0146	0.8237	0.0000	0.0023	0.0000
0.0000	0.0000	0.9998	0.0000	0.2677	0.0000
1.0000	1.0000	0.9963	1.0000	1.0000	1.0000
0.0000	0.0000	0.0076	0.0000	0.0000	0.0000

Tablo 3.15. Karakavak Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	0.9996	1.0000	1.0000	1.0000
0.0000	0.0003	1.0000	0.0000	0.9999	0.0000
1.0000	0.9998	0.0000	1.0000	0.0002	1.0000

Tablo 3.16. Karaağaç Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	0.3568	1.0000	1.0000
0.0000	0.0000	0.0000	0.0355	0.0019	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.9592	0.0027	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tablo 3.17. Kartopu Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0000	0.0000	0.0029	0.0000	0.0000	0.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tablo 3.18. Kusburnu Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	1.0000	0.9998	
0.0000	0.9895	1.0000	0.9821	0.9251	
1.0000	1.0000	0.9058	1.0000	1.0000	BELİRLİ
0.0000	0.0000	0.0000	0.0000	0.0002	DEĞİL
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	

Tablo 3.19. Akçaağaç Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9764	1.0000	0.9997	0.0000	0.0000	
0.0742	0.0000	0.0014	0.9999	1.0000	
0.0000	0.0000	0.0000	1.0000	1.0000	BELİRLİ
1.0000	1.0000	1.0000	1.0000	1.0000	DEĞİL
0.0000	0.0000	0.0000	0.0000	0.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	

Tablo 3.20. Çitlenbik Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.9112	0.0000	0.0004	0.0000	0.9999	
0.6038	1.0000	0.9971	1.0000	0.0005	
1.0000	1.0000	1.0000	1.0000	1.0000	BELİRLİ
0.9429	0.9997	1.0000	1.0000	0.0000	DEĞİL
1.0000	1.0000	0.0004	0.0002	1.0000	
0.0000	0.0000	0.9995	0.9997	0.0000	

Tablo 3.21. Gül Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	BELİRLİ
1.0000	1.0000	1.0000	1.0000	1.0000	DEĞİL
0.0000	0.0000	0.0000	0.0000	0.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	

Tablo 3.22. İğde Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.1717	0.0000	0.0000	0.0000	0.0000	
1.0000	1.0000	1.0000	1.0000	1.0000	
0.7800	0.9999	1.0000	0.9996	1.0000	BELİRLİ
0.0356	1.0000	1.0000	0.9954	1.0000	DEĞİL
1.0000	0.2230	0.0000	1.0000	0.0000	
0.0000	0.8907	1.0000	0.0000	1.0000	

Tablo 3.23. Leylak Yaprakları

1. Yaprak	2.Yaprak	3. Yaprak	4.Yaprak	5. Yaprak	Hedef
0.0000	1.0000	1.0000	0.1454	0.7794	
1.0000	1.0000	1.0000	1.0000	1.0000	
1.0000	0.0000	0.0000	0.0004	0.0273	BELİRLİ
0.9999	0.0000	0.0000	0.0606	0.0034	DEĞİL
1.0000	1.0000	1.0000	1.0000	1.0000	
0.0000	0.0000	0.0000	0.0000	0.0000	

Tablo 3.24. Frenk Üzümü Yaprakları

Test işlemi sonunda akasya yapraklarının sınıflandırma doğruluğu % 80' e düşmüştür. İlk akasya desenine ait çıkış deseni doğru sınıfı göstermemektedir. Karakavak yapraklarında ise doğruluk %60 olup, 3 numaralı yaprak kuşburnu çıkış desenini vermektedir. Diğer 5 numaralı desen ise eğitim aşamasında tanıtılan çıkış desenlerinden hiçbirine benzememektedir. Desen tanımada böyle çıkış desenlerine hayali, saçma desen (spiritual pattern) adı verilmektedir. Karaağaç yapraklarında da doğruluk %60 olup, 3 ve 5 numaralı desenler; 011110 çıkış deseni ile kuşburnu olarak görülmektedir. Kartopu yaprakları %80 doğrulukla sınıflandırılmıştır. Bu yapraklarda da 4 numaralı yaprak için elde edilen çıkış deseni hayali bir desendir. Kuşburnu yaprakları yine %100 doğrulukla sınıflandırılmıştır. Toplam doğruluk %76 olarak elde edilmiştir. Görüldüğü gibi model desen seçimi sistem performansını önemli ölçüde etkilemektedir. Bu çalışma sırasında farklı modeller seçilerek çok sayıda deneme yapılmış, en iyi sonucu veren bu iki deneme burada verilmiştir. Özet olarak seçilen modelin, ait olduğu sınıfı temsil etme yeteneği ile orantılı olarak ağ performansı artmaktadır.

3.5.2. Hopfield Ağı İle Sınıflandırma

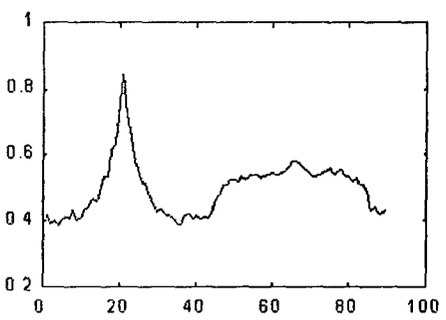
Hopfield ağı ile sınıflandırma işlemi yapmadan önce, geriye yayılma ağında olduğu gibi önce vermatris.m ve normatris.m programları çalıştırılarak ağa girilecek veriler elde edilir. Hopfield ağı ile yapılan çalışmadan verimli bir sınıflandırma işlemi yapılamadığından sınıf sayısı üç ile sınırlandırılmıştır. Model yaprak olarak 3 numaralı karakavak yaprağı, 1 numaralı karaağaç yaprağı ve 2 numaralı akasya yaprağı seçilmiştir.

Her iki geri besleme adımında alınan sonuçlar aşağıda verilmiştir. Toplam 20 defa geribesleme yapılmasına rağmen, çıkış deseni hedeflenen desene yaklaşma yerine uzaklaşma eğilimi göstermiştir.

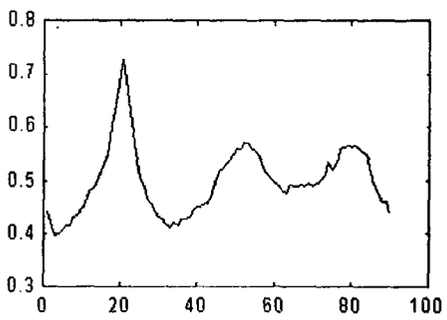
GÜNCELLEME SAYISI	KARAKAVAK 3 .YAPRAK	KARAAĞAÇ 1. YAPRAK	AKASYA 2.YAPRAK
0	0.1457	0.0000	0.9000
2	0.5284	0.2458	1.0381
4	0.8399	0.3989	1.0305
6	1.2685	0.6532	1.0450
8	1.8655	1.0533	1.1021
10	2.6332	1.6086	1.1860
12	3.7094	2.4437	1.3883
14	4.9523	3.4937	1.6889
16	5.9896	4.3549	1.9978
18	6.9538	5.2206	2.3431
20	7.7750	5.9843	2.6803

Tablo 3.25. Hopfield Ağında Modellere Göre Karesel Hatanın Değişimi

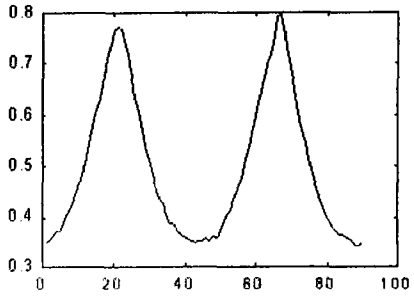
Yukarıdaki veriler test aşamasında, ağ girişine 1 numaralı karaağaç deseni verildiğinde her bir adımda çıkış deseni ile, model desenler arasındaki toplam karesel hatayı göstermektedir. Test deseni model desenlerden biri olmasına rağmen ağ kararlı bir noktaya ulaşamamakta ve denge noktasından sürekli uzaklaşmaktadır. Yani bir nevi pozitif geri besleme söz konusudur. Son olarak 20. adım bittiğinde test deseni, karaağaca değil akasyaya daha fazla benzemektedir. Aşağıda bu durum grafik olarak gösterilmiştir



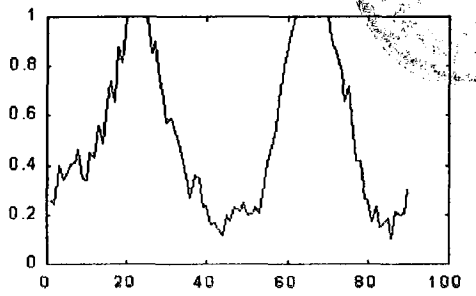
Şekil 3.5. (3) Numaralı Karakavak Yaprağına ait Öznitelik Grafiği



Şekil 3.6. (1) Numaralı Karaağaç Yaprağına ait Öznitelik Grafiği



Şekil 3.7. (2) Numaralı Akasya Yaprağına ait Öznitelik Grafiği



Şekil 3.8. (1) Numaralı Karaağaç Yaprağının Test Edilmesiyle Çıkışta Görülen Desen

4. SONUÇ

Bu çalışmada geriye yayılma yapay sinir ağı ile 10 farklı bitki türüne ait yaprak görüntüleri sınıflandırılarak bitki türleri tesbit edilmiştir. Eğitim işlemi sırasında tanıtılan 5 farklı bitki türünün sınıflandırılmasında %80 doğruluk elde edilmiştir.

Sınıflandırma işleminde yaygın olarak kullanılan geriye yayılma algoritması ve Hopfield ağlarının, yaprak görüntüleri gibi birbirine çok yakın özellikler gösteren desenlerin sınıflandırılmasındaki performansları karşılaştırılmıştır. Hopfield ağlarının bu tip desenlerin sınıflandırılmasında başarısız olduğu gözlenmiştir. Geriye yayılma algoritması ise beklenen performansı göstermiştir.

Çalışma sırasında ayrıca bir görüntüde cisim ile zeminin sınırının tesbit edilmesinde sağ el ve sol el yöntemleri geliştirilmiştir.

KAYNAKLAR

[1] Amit D.J., Gutfrent G., Sompolinsky H. Spin-glass models of neural networks, Phys. Rev. A32: 1007-1018, 1985.

[2] Bryson A.E., Ho Y.C. Applied optimal control, Blaisdell, New York, 1969.

[3] Cichocki A. Neural network for optimization and signal processing, New York: JohnWiley,1993.

[4] Erdön A. Sayısal görüntülerde kenar tanıma metotları ,İTÜ Merkez Kitaplığı no:27363, 1992

[5] Humphries S., Simonton W. Identification of plant part using color and geometric data, ASAE Volume 36(5):1493-1500, 1993.

[6] Franz E., Gebhardt M.R., Unklesbay K.B. Algorithms for extracting leaf boundary information from digital images of plant foliage, ASAE Volume 38(2): 625-633, 1995.

[7] Fu K.S.,Young T.Y. Handbook of pattern recognition and image processing, Academic Press, California, 1986.

[8] Freeman J.A., Skapura D.M. Neural networks:algorithms,applications and programming techniques, MA, Addison-Wesley, 1991.

[9] Hertz J.A., Palmer R.G. Introduction to the theory of neural computation, Adisson-Wesley, New York, 1991.

[10] Hopfield J.J. Unlearning has a stabilizing effects in collective memories, Nature 304:158, 1983.

[11] Huang S.C., Huang Y.F. Bounds on the number fo hidden neurons in multilayer perceptrons, IEEE Trans. Neural Networks Volume 2(1):47-55,1991.

[12] Gonzalez R.C., Tou J.T. Pattern recognition principles, Addison-Wesley Co., Massachusets, 1974.

[13] Güzeliş C. Genelleştirilmiş hücresel yapay sinir ağları, EMO 5. Ulusal Kongresi, Trabzon, pp:7-12,1993.

[14] Guyer D.E., Miles G.E. Schreiber M.M., Mitchell O.R., Vanderbilt V.C. Machine vision and image processing for plant identification, ASAE Volume 29(6),1986.

[15] Jacobs R.A. Increased raised of convergence through learning rate adaptation. Neural Networs Volume 1: 295-307,1988.

[16] Karamahmut S. Hücresel yapay sinir ağları için iki öğrenme algoritması ve görüntü işleme uygulamaları, İTÜ Merkez Kitaplığı, no: 39003, 1994.

[17] Mirchandini G., Cao W. On hidden nodes in neural nets, IEEE Trans. Circuit, Systems Volume 36(5): 641-664, 1989.

[18] Lee G.C.S., Lin C.T. Neural fuzzy systems, Prentice Hall, New Jersey, 1995.

[19] LeeCun Y. Alearning procedure for assymetric network, Cognitiva 85: 599-604, Paris, 1985.

[20] Liao K., Paulse M.R., Reid J.F., Ni B.C., Bonifacio E.P. Corn kernel breakage classification by machine vision using neural network classifier, ASAE Volume 36(6): 1949-1953,1993.

[21] Ölmez T. Yapay sinir ağları yardımıyla biyomedikal dokuların sınıflandırılması, İTÜ Merkez Kitaplığı no:44082, 1995.

[22] Parker D.B. Learning logic, MIT, no:TR-47, Cambridge.

[23] Pearson T.C., Slaugher D.C. Machine vision detection of early split pistachio nuts, ASAE Volume 39(3): 1203-1207, 1996.

[24] Rumelhart D.E., Hinton G.E., Willias R.J. Learning represantions by back propagation errors, Nature 323:533-536, 1986(a).

[25] Rumelhart D.E., Hinton G.E., Willias R.J. Learning internal represations by error propagation, Paralell distrubuted processing Volume 1: Chapter 8, Cambridge, MA, 1986(b).

[26] Shatadal P., Jayas D.S., Bulley N.R. Digital image analysis for software seperation and classification of touching grains: II. Classification, ASAE Volume 38(2): 645-649,1995.

[27] Vogl T.P., Mangis J.K., Rigler A.K. Accelarating the convergence of the backpropagation method, Biol. Cybern. Volume 59:257-263,1988.

[28] Werbos P.J. Beyond regression: New tools for prediction and analysis in the behavioral sciences, Harward University, 1974.

[29] Wessel L.F., Barnard E. Avoiding false local minima by proper initialization of connections, IEEE Trans. Neural Network Volume 3(1):73-86,1992.

[30] Wolfe R.R., Sandler W.E. An algorithm for stem detection using digital image analysis, ASAE October (1985): 641-644,1985

[31] Zurada J.M. Introduction to artifical neural systems, West Publishing Co., St. Paul,1991.

ÖZGEÇMİŞ

Erdem BİLGİLİ, 1972 yılında Şenkaya'da doğdu. İlk,orta ve lise öğrenimini Erzurum'da tamamladı. 1996 yılında Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümünden mezun oldu. Aynı yıl Atatürk Üniversitesi Tarım Makinaları Bölümünde araştırma görevlisi olarak çalışmaya başladı. 1996 yılı güz yarılında Gebze Yüksek Teknoloji Enstitüsünde yüksek lisans öğrenimine başladı. 1998 yılında Gebze Yüksek Teknoloji Enstitüsünde araştırma görevliliğine atandı. Halen araştırma görevliliği görevini yürütmektedir.

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

EKLER

BİLGİSAYAR PROGRAMLARI

BMPYAPRAKOKU.M

```
okuma
esik
basla1
alan1
agmerkz
simeksen
acigen
cevir23
```

OKUMA.M

```
%%%%%%%BMP FORMATLI DOSYALARIN PALETİNİ SATIR*SÜTUN BOYUTLU%%%%%%%
            %%%%MATRİSE DÖNÜŞTÜRÜR%%%%%%%%%%%
fid=fopen(s,'r');
BK=fread(fid,'char');
fclose(fid);
satir=BK(24)*256+BK(23);
sutun=BK(20)*256+BK(19);
uzunluk=length(BK);
test=0
  while BK(uzunluk-test)==0
     test=test+1;
  end;
  atla=0;
    for i=1:satir
       atla=atla+test;
       for j=1:sutun
          ARES(i,sutun-j+1)=BK(uzunluk-(j-1)-(i-1)*sutun-atla);
       end;
    end;
ARES=ARES/256;
y=sobel1(ARES);
```

SOBEL1.M

```
function y=sobel1(x)
h=[-1,-2,-1;0,0,0;1,2,1];
v=h';
xh=conv4(x,h);
xv=conv4(x,v);
y=xh.*xh+xv.*xv;
```

ESIK.M

```
[m,n]=size(y);
A=zeros(m,n);
enbuyuk=0;
```

```
 esikdeg=.1;
for i=3:m-2
  for j=3:n-2
     if y(i,j)>esikdeg
       A(i,j)=1;
     end;
  end;
end;
```

BASLA1.M

```
clear SATIR SUTUN UZAKLIK TETA4;
cevre=0;
satbas=1;
sonbas=1;
while cevre<50
sonbas=satbas+1;
kos1;
noktasay=cp+t;
cevre=cp*1.414+t
end
```

ALAN1.M

```
%%%*************ŞEKLİN ALANINI VE AĞIRLIK MERKEZİNİ
BULMA*************%%%
C=B;
[m,n]=size(C);
sat=satbas
sut=sutbas;
alt1;
asil3;
```

AGMERKZ.M

```
topalan=0;
sattop=0;
suttop=0;
for i=1:m
   for j=1:n
       sattop=sattop+i*C(i,j);
       suttop=suttop+j*C(i,j);
       topalan=topalan+C(i,j);
       end
end

satmerk=sattop/topalan
sutmerk=suttop/topalan
topalan=topalan
```

SIMEKSEN.M

```
m02=0;
ccmi=0;
cm11=0;
cm20=0;
cm02=0;
pmi1=0;
pmi2=0;
```

```
  for i=1:m
     for j=1:n
        satfark=satmerk-i;
        sutfark=j-sutmerk;
     cmi=cmi+(satfark^2+sutfark^2)*C(i,j);
     m02=m02+satfark^2*C(i,j);
     m20=m20+sutfark^2*C(i,j);
     m11=m11+satfark*sutfark*C(i,j);
    end
  end

 denkb=(m20-m02)/m11;
 tteta1=(-denkb+sqrt(denkb^2+4))/2;
 tteta2=(-denkb-sqrt(denkb^2+4))/2;
 teta1=atan(tteta1);
 teta2=atan(tteta2);
 aci1=180*teta1/pi;
 aci2=180*teta2/pi;
 if teta1>pi/4
    teta1=teta2;
  end
 bcmi=cmi/topalan;
 bm20=m20/topalan;
 bm02=m02/topalan;
 bm11=m11/topalan;

 pmi1=pmi1/topalan;
 pmi2=pmi2/topalan;
 sonuc=[pmi1 pmi2 aci1 aci2;
        bcmi bm20 bm02 bm11]



 compact=4*pi*topalan/cevre^2
 round=4*topalan/(pi*cevre^2)
```

ACIGEN.M

```
[k,boyut]=size(SATIR);
for i=1:boyut
 UZAKLIK(i)=sqrt((SATIR(i)-satmerk)^2+(SUTUN(i)-sutmerk)^2);
 teta3=atan2((satmerk-SATIR(i)),(SUTUN(i)-sutmerk));
 TETA4(i)=mod(teta3-teta1,2*pi);
end
```

CEVIR23.M

```
F=zeros(180,10);
caci=zeros(180,1);
for i=1:boyut
SUTUN1(i)=UZAKLIK(i)*cos(TETA4(i));
SATIR1(i)=-UZAKLIK(i)*sin(TETA4(i));
end
satmerk1=-min(SATIR1)+1;sutmerk1=-min(SUTUN1)+1;
c1=fix(90*TETA4(2)/pi);
UZAKLIK(1)=0;
maxuzaklik=max(UZAKLIK);
for i=2:boyut
   a=fix(SUTUN1(i)+sutmerk1);
```

```
b=fix(SATIR1(i)+satmerk1);
  if (SUTUN1(i)+sutmerk1-a)>.5
    a=a+1;
  end
  if (SATIR1(i)+satmerk1-b)>.5
    b=b+1;
  end
E(b,a)=1;
c=fix(TETA4(i)*90/pi);

% if (TETA4(i)*180/pi-c)>.5
%   c=c+1;
%end
% if (c==0)
%   c=180;
%end
if abs(c-c1)>=1 & abs(c-c1)<=4
   for kar=1:abs(c-c1)
      ind=mod(c1+kar*(c-c1)/abs(c-c1),180);
      if ind==0
         ind=180;
      end
      if caci(ind,1)<10;
         caci(ind,1)=caci(ind,1)+1;
         F(ind,caci(ind,1))=UZAKLIK(i)/maxuzaklik;
      end
   end
end
c1=c;
end
for i=1:180
   if F(i,1)==0
      F(i,1)=F(i+1,1)/2+F(i-1,1)/2;
      caci(i,1)=1;
   end
   egitim(i,1)=max(F(i,:));
   egitim(i,3)=caci(i,1);
   enkd=1;
   for j=1:10
      if F(i,j)<enkd & F(i,j)~=0
         enkd=F(i,j);
      end
   end
   egitim(i,2)=enkd;
end
[eboy1,eboy2]=size(E);
EKY=zeros(eboy1,1);
EKA=zeros(1,eboy2+1);
   E=[E EKY;
      EKA];
```

KOS1.M

```
global sat sut satbas sutbas
cevrbul1;
endis1;
p
cp=1;
t=1;
asil2;
```

CEVRBUL1.M

```
%%%%%%%%%%%%%%%%%%%YAPRAĞIN EN DIŞ KENARININ BULUNMASI%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%BAŞLANGIÇ NOKTASININ TESBİTİ%%%%%%%%%%%%%%%%%%%
t=1;
if sonbas==2

par1=1;
satbas=0;
sutbas=0;
[m,n]=size(A);
A21=zeros(m,1);
A22=zeros(1,n+2);
A=[A22;
   A21 A A21;
   A22];
[m,n]=size(A);
B=zeros(m,n);
TEKRAR=zeros(m,n);
end
for i=sonbas:m
   for j=1:n
      if A(i,j)==par1
        sat=i;
        satbas=i;
        sut=j;
        sutbas=j;
        B(satbas,sutbas)=par1;
        return
      end
   end
  end
```

ENDIS1.M

```
p=1;
t=1
sagyan1;
```

ASIL2.M

```
while (sat~=satbas|sut~=sutbas)
  if p==1
     SATIR(cp+t)=sat;
     SUTUN(cp+t)=sut;
     saustca1;

    elseif p==2
       SATIR(cp+t)=sat;
       SUTUN(cp+t)=sut;
       sagyan1;

    elseif p==3
       SATIR(cp+t)=sat;
       SUTUN(cp+t)=sut;
        saaltca1;

    elseif p==4
```

```
        SATIR(cp+t)=sat;
        SUTUN(cp+t)=sut;
         alt1;

    elseif p==5
        SATIR(cp+t)=sat;
        SUTUN(cp+t)=sut;
         soaltca1;

    elseif p==6
        SATIR(cp+t)=sat;
        SUTUN(cp+t)=sut;
         solyan1;

    elseif p==7
        SATIR(cp+t)=sat;
        SUTUN(cp+t)=sut;
         soustca1;

    elseif p==8
        SATIR(cp+t)=sat;
        SUTUN(cp+t)=sut;
        ust1;
    else
    disp 'uygun değil';
    sat
    sut
    pause

    end
  end
```

ASIL3.M

```
while (sat~=satbas|sut~=sutbas)
  if p==1
    if TEKRAR(sat+1,sut-1)~=1
       if p1~=6
       asat=sat+1;
       asut=sut;
       while B(asat,asut)~=par1
         C(asat,asut)=par1;
         asat=asat+1;
       end
       asat=sat+1;
       asut=sut;
       while B(asat,asut)~=par1
         C(asat,asut)=par1;
         asut=asut+1;
      end
    end
   end
       p1=1;
       saustca2;

  elseif p==2
    if TEKRAR(sat,sut-1)~=1
        if p1~=7
       asat=sat+1;
       asut=sut-1;
       while B(asat,asut)~=par1
```

```
        C(asat,asut)=par1;
        asat=asat+1;
      end
      end
end
      p1=2;
      sagyan2;

elseif p==3
  if TEKRAR(sat-1,sut-1)~=1
       if p1~=8
       asat=sat;
      asut=sut-1;
      while B(asat,asut)~=par1
        C(asat,asut)=par1;
        asat=asat+1;
      end
      asat=sat;
      asut=asut-1;
      while B(asat,asut)~=par1
        C(asat,asut)=par1;
        asut=asut-1;
      end
      end
      end
      p1=3;
   saaltca2;

  elseif p==4
  if TEKRAR(sat-1,sut)~=1
      if p1~=1
      asat=sat-1;
      asut=sut-1;
      while B(asat,asut)~=par1

      C(asat,asut)=par1;
      asut=asut-1;
      end
      end
      end
    p1=4;
    alt2;

  elseif p==5
  if TEKRAR(sat-1,sut+1)~=1
       if p1~=2
      asat=sat-1;
      asut=sut;
      while B(asat,asut)~=par1
        C(asat,asut)=par1;
        asat=asat-1;
      end
      asat=sat-1;
      asut=sut;
      while B(asat,asut)~=par1
        C(asat,asut)=par1;
        asut=asut-1;
      end
      end
    end
      p1=5;
```

```
        soaltca2;

     elseif p==6
     if TEKRAR(sat,sut+1)~=1
         if p1~=3
        asat=sat-1;
        asut=sut+1;
        while B(asat,asut)~=par1
          C(asat,asut)=par1;
          asat=asat-1;
        end
        end
      end
        p1=6;
        solyan2;

     elseif p==7
     if TEKRAR(sat+1,sut+1)~=1
        if p1~=4
          asat=sat;
          asut=sut+1;
          while B(asat,asut)~=par1
            C(asat,asut)=par1;
            asut=asut+1;
          end
          asat=sat;
          asut=sut+1;
          while B(asat,asut)~=par1
            C(asat,asut)=par1;
            asat=asat-1;
          end
        end
     end
     p1=7;
     soustca2;

     elseif p==8
     if TEKRAR(sat+1,sut)~=1
         if p1~=5
         asat=sat+1;
        asut=sut+1;
        while B(asat,asut)~=par1
          C(asat,asut)=par1;
          asut=asut+1;
        end
       end
     end
         p1=8;
       ust2;
   else

   disp 'uygun değil';
   end
  end
```

ALT1

```
%**************alt*********

%%%%%%%%%%%
```

```
t=t+1;
if A(sat,sut+1)==par1
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=2;

%%%%%%%%%%%%%%%
elseif A(sat+1,sut+1)==par1
    sat=sat+1;
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%%%%%%
  elseif A(sat+1,sut)==par1
    sat=sat+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=4;

%%%%%%%%%%%%%%
   elseif A(sat+1,sut-1)==par1
     sat=sat+1;
     sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=5;

 %%%%%%%%%%%%%

 elseif A(sat,sut-1)==par1
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%%%

 elseif A(sat-1,sut-1)==par1
    sat=sat-1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=7;
%%%%%%%%%%%%%%
 elseif A(sat,sut)==par1
    disp 'alt'
```

```
    pause
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat+1,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    %A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    %A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;

  p=4;
  end
```

SAALCA1.M

```
%**********sağalt çapraz**************

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 cp=cp+1;
 if A(sat-1,sut+1)==par1
    sat=sat-1;
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=1;

%%%%%%%%
 elseif A(sat,sut+1)==par1
     sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=2;
```

```
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%
 elseif A(sat+1,sut)==par1
    sat=sat+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=4;

%%%%%%%%%%%
 elseif A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=5;

%%%%%%%%%%

 elseif A(sat,sut-1)==par1
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=6;
%%%%%%%%%%
 elseif A(sat,sut)==par1
    p=3;
    disp 'saalt'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    %A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    %A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
```

```
  end
```

SAALTCA2.M

```
%**********sağalt çapraz**************

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 cp=cp+1;
 if A(sat-1,sut+1)==par1
    sat=sat-1;
    sut=sut+1;
     B(sat,sut)=par1;
     p=1;

%%%%%%%%
 elseif A(sat,sut+1)==par1
     sut=sut+1;
     B(sat,sut)=par1;
     p=2;

%%%%%%%%
 elseif A(sat+1,sut+1)==par1
    sat=sat+1;
    sut=sut+1;
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%
 elseif A(sat+1,sut)==par1
    sat=sat+1;
    B(sat,sut)=par1;
    p=4;

%%%%%%%%%%%
 elseif A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    B(sat,sut)=par1;
    p=5;

%%%%%%%%%%

 elseif A(sat,sut-1)==par1
    sut=sut-1;
    B(sat,sut)=par1;
    p=6;
%%%%%%%%%%%
 elseif A(sat,sut)==par1
    p=3;
    disp 'saalt'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
```

```
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
```

SAGYAN1.M

```
%**************sağyan*********

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   t=t+1;
   if A(sat-1,sut)==par1
      sat=sat-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=8;

%%%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat-1,sut+1)==par1
     sat=sat-1;
     sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat,sut+1)==par1
      sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=2;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat+1,sut+1)==par1
      sat=sat+1;
      sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=3;

%%%%%%%%%%%%%%%%%%%%%%%
```

```
   elseif A(sat+1,sut)==par1
       sat=sat+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
       B(sat,sut)=par1;
       p=4;

%%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
      sat=sat+1;
      sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=5;
  else
    p =2;
    disp 'sayan'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    %A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    %A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
 %%%%%%%%%%%%%%%%%%%%%%%%
```

SAGYAN2.M

```
%**************sağyan*********

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   t=t+1;
   if A(sat-1,sut)==par1
      sat=sat-1;
      B(sat,sut)=par1;
      p=8;

%%%%%%%%%%%%%%%%%%%%%%%%
```

```
   elseif A(sat-1,sut+1)==par1
     sat=sat-1;
     sut=sut+1;
     B(sat,sut)=par1;
     p=1;

%%%%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat,sut+1)==par1
      sut=sut+1;
      B(sat,sut)=par1;
      p=2;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat+1,sut+1)==par1
      sat=sat+1;
      sut=sut+1;
      B(sat,sut)=par1;
      p=3;

%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat+1,sut)==par1
       sat=sat+1;
       B(sat,sut)=par1;
       p=4;

%%%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
      sat=sat+1;
      sut=sut-1;
      B(sat,sut)=par1;
      p=5;
  else
    p =2;
    disp 'sayan'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
```

```
%%%%%%%%%%%%%%%%%%%%%%%
```

SAUSTCA1.M

```
%**************sağüstçapraz********

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  cp=cp+1;
  if A(sat-1,sut-1)==par1
     sat=sat-1;
     sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=7;

%%%%%%%%%%%%%%%%%%%%%%%%%

  elseif A(sat-1,sut)==par1
      sat=sat-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=8;

%%%%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat-1,sut+1)==par1
      sat=sat-1;
      sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=1;

%%%%%%%%%%%%%%%%%%%%%%%%
 elseif A(sat,sut+1)==par1
      sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
      B(sat,sut)=par1;
      p=2;

%%%%%%%%%%%%%%%%%%%%%%%
 elseif A(sat+1,sut+1)==par1
    sat=sat+1;
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%%%%%%%%%%%%%

 elseif A(sat+1,sut)==par1
     sat=sat+1;
    if B(sat,sut)==par1
```

```
      TEKRAR(sat,sut)=1;
     end
      B(sat,sut)=par1;
      p=4;
%%%%%%%%%%%%%%%%%%
 elseif A(sat,sut)==par1
      p=1;
      disp 'saust'
     pause
     A(sat+1,sut+1)=par1;
     %A(sat+1,sut)=par1;
     A(sat,sut+1)=par1;
     A(sat-1,sut+1)=par1;
     A(sat+1,sut-1)=par1;
     A(sat,sut-1)=par1;
     A(sat-1,sut-1)=par1;
     A(sat-1,sut)=par1;

     A(sat+2,sut-2)=par1;
     A(sat+2,sut-1)=par1;
     %A(sat+2,sut)=par1;
     A(sat+2,sut+1)=par1;
     A(sat+2,sut+2)=par1;
     A(sat-2,sut+2)=par1;
     A(sat-1,sut+2)=par1;
     A(sat,sut+2)=par1;
     A(sat+1,sut+2)=par1;
     A(sat-2,sut-2)=par1;
     A(sat-2,sut-1)=par1;
     A(sat-2,sut)=par1;
     A(sat-2,sut+1)=par1;
     A(sat-1,sut-2)=par1;
     A(sat,sut-2)=par1;
     A(sat+1,sut-2)=par1;
  end
%%%%%%%%%%%%%%%%%%%%%%
```

SAUSTCA2.M

```
%**************sağüstçapraz********

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  cp=cp+1;
  if A(sat-1,sut-1)==par1
     sat=sat-1;
     sut=sut-1;
     B(sat,sut)=par1;
     p=7;

%%%%%%%%%%%%%%%%%%%%%%%%%

  elseif A(sat-1,sut)==par1
      sat=sat-1;
      B(sat,sut)=par1;
      p=8;

%%%%%%%%%%%%%%%%%%%%%%%%%
   elseif A(sat-1,sut+1)==par1
      sat=sat-1;
      sut=sut+1;
      B(sat,sut)=par1;
```

```
      p=1;

%%%%%%%%%%%%%%%%%%%%%%%%
 elseif A(sat,sut+1)==par1
      sut=sut+1;
      B(sat,sut)=par1;
      p=2;

%%%%%%%%%%%%%%%%%%%%%%%%
 elseif A(sat+1,sut+1)==par1
    sat=sat+1;
    sut=sut+1;
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%%%%%%%%%%%%%%

 elseif A(sat+1,sut)==par1
     sat=sat+1;
     B(sat,sut)=par1;
     p=4;
%%%%%%%%%%%%%%%%%%%
 elseif A(sat,sut)==par1
     p=1;
     disp 'saust'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat,sut+1)=par1;
    A(sat-1,sut+1)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
%%%%%%%%%%%%%%%%%%%%%%%%
```

SOALTCA1.M

```
%**************solaltçapraz*********

  %%%%%%%%%%%%%%%%%%
 cp=cp+1;
 if A(sat+1,sut+1)==par1
```

```
      sat=sat+1;
      sut=sut+1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
      B(sat,sut)=par1;
      p=3;

%%%%%%%%%%%%%%%%%
 elseif A(sat+1,sut)==par1
      sat=sat+1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
       B(sat,sut)=par1;
       p=4;
  %%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
       sat=sat+1;
       sut=sut-1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
       B(sat,sut)=par1;
       p=5;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
       sut=sut-1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
       B(sat,sut)=par1;
       p=6;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
       sat=sat-1;
       sut=sut-1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
       B(sat,sut)=par1;
       p=7;

%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
       sat=sat-1;
     if B(sat,sut)==par1
       TEKRAR(sat,sut)=1;
     end
       B(sat,sut)=par1;
       p=8;
%%%%%%%%%%%%%%%%%%
 else
      p=5;
     disp 'soalt'
     pause
     A(sat+1,sut+1)=par1;
     A(sat+1,sut)=par1;
     A(sat+1,sut-1)=par1;
```

```
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    %A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    %A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
end
```

SOALTCA2.M

```
%**************solaltçapraz*********

  %%%%%%%%%%%%%%%%%%
 cp=cp+1;
 if A(sat+1,sut+1)==par1
     sat=sat+1;
     sut=sut+1;
     B(sat,sut)=par1;
     p=3;

%%%%%%%%%%%%%%%%%
 elseif A(sat+1,sut)==par1
      sat=sat+1;
      B(sat,sut)=par1;
      p=4;
  %%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
      sat=sat+1;
      sut=sut-1;
      B(sat,sut)=par1;
      p=5;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
      sut=sut-1;
      B(sat,sut)=par1;
      p=6;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
      sat=sat-1;
      sut=sut-1;
      B(sat,sut)=par1;
      p=7;
```

```
%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
      sat=sat-1;
      B(sat,sut)=par1;
      p=8;
%%%%%%%%%%%%%%%%%%%
 else
     p=5;
    disp 'soalt'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
end
```

SOLYAN1

```
%***************solyan*********

 %%%%%%%%%%%%%%%%%%%%%%%
 t=t+1;
 if A(sat+1,sut)==par1
    sat=sat+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=4;

%%%%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=5;
```

```
%%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
     sat=sat-1;
     sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=7;

%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
     sat=sat-1;
     if B(sat,sut)==par1


      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=8;

%%%%%%%%%%%
  elseif A(sat-1,sut+1)==par1
    sat=sat-1;
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=1;
%%%%%%%%%%%%%%%%%
 else
     p=6;
    disp 'solyan'
    pause
    A(sat+1,sut+1)=par1;
    %A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    %A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
```

```
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
end
```

SOLYAN2.M

```
%***************solyan*********

 %%%%%%%%%%%%%%%%%%%%%%
 t=t+1;
 if A(sat+1,sut)==par1
    sat=sat+1;
    B(sat,sut)=par1;
    p=4;

%%%%%%%%%%%%%%
  elseif A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    B(sat,sut)=par1;
    p=5;

%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
    sut=sut-1;
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
     sat=sat-1;
     sut=sut-1;
     B(sat,sut)=par1;
     p=7;

%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
     sat=sat-1;
    B(sat,sut)=par1;
    p=8;

%%%%%%%%%%%%%
  elseif A(sat-1,sut+1)==par1
    sat=sat-1;
    sut=sut+1;
    B(sat,sut)=par1;
    p=1;
%%%%%%%%%%%%%%%%%%
 else
     p=6;
    disp 'solyan'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
```

```
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
end
```

SOLUSTCA1.M

```
***************solüstçapraz*********

%%%%%%%%%%%%%%%%%%%%%%%
 cp=cp+1;
 if A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=5;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
    sat=sat-1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=7;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
    sat=sat-1;
```

```
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=8;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut+1)==par1
     sat=sat-1;
     sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=1;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut+1)==par1
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=2;
%%%%%%%%%%%%%%%%%%
 else
     p=7;
    disp 'soust'
    pause
    A(sat+1,sut+1)=par1;
    %A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    %A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
%%%%%%%%%%%%%%%%%%%%%%%%
```

SOLUSTCA2.M

```
%**************solüstçapraz*********

%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 cp=cp+1;
 if A(sat+1,sut-1)==par1
    sat=sat+1;
    sut=sut-1;
    B(sat,sut)=par1;
    p=5;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut-1)==par1
    sut=sut-1;
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
    sat=sat-1;
    sut=sut-1;
    B(sat,sut)=par1;
    p=7;

%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
    sat=sat-1;
    B(sat,sut)=par1;
    p=8;

%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat-1,sut+1)==par1
     sat=sat-1;
     sut=sut+1;
     B(sat,sut)=par1;
     p=1;

%%%%%%%%%%%%%%%%%%%%%%%%
  elseif A(sat,sut+1)==par1
    sut=sut+1;
    B(sat,sut)=par1;
    p=2;
%%%%%%%%%%%%%%%%%%%
 else
     p=7;
    disp 'soust'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
```

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

```
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
  end
%%%%%%%%%%%%%%%%%%%%%%%%
```

UST1.M

```
***************üst*********

%%%%%%%%%%%%%%%%%%%%
  t=t+1;
 if A(sat,sut-1)==par1
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
    sat=sat-1;
    sut=sut-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=7;

%%%%%%%%%%%%%
  elseif A(sat-1,sut)==par1
     sat=sat-1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=8;

%%%%%%%%%%%%%5
  elseif A(sat-1,sut+1)==par1
     sat=sat-1;
     sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=1;

%%%%%%%%%%%%%
  elseif A(sat,sut+1)==par1
     sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
     B(sat,sut)=par1;
     p=2;
```

```
%%%%%%%%%%%%%%
  elseif A(sat+1,sut+1)==par1
    sat=sat+1;
    sut=sut+1;
    if B(sat,sut)==par1
      TEKRAR(sat,sut)=1;
    end
    B(sat,sut)=par1;
    p=3;

%%%%%%%%%%%%%%%%%%
 else
    p=8;
disp 'ust'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    %A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    %A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
 end
```

UST2.M

```
%**************üst*********

%%%%%%%%%%%%%%%%%%
 t=t+1;
 if A(sat,sut-1)==par1
    sut=sut-1;
    B(sat,sut)=par1;
    p=6;

%%%%%%%%%%%%
  elseif A(sat-1,sut-1)==par1
```

```
      sat=sat-1;
      B(sat,sut)=par1;
      p=8;

%%%%%%%%%%%%%5
  elseif A(sat-1,sut+1)==par1
      sat=sat-1;
      sut=sut+1;
      B(sat,sut)=par1;
      p=1;

%%%%%%%%%%%%%%
  elseif A(sat,sut+1)==par1
      sut=sut+1;
      B(sat,sut)=par1;
      p=2;

%%%%%%%%%%%%%%
  elseif A(sat+1,sut+1)==par1
     sat=sat+1;
     sut=sut+1;
     B(sat,sut)=par1;
     p=3;

%%%%%%%%%%%%%%%%%%
 else
    p=8;
disp 'ust'
    pause
    A(sat+1,sut+1)=par1;
    A(sat+1,sut)=par1;
    A(sat+1,sut-1)=par1;
    A(sat,sut-1)=par1;
    A(sat-1,sut-1)=par1;
    A(sat-1,sut)=par1;
    A(sat-1,sut+1)=par1;
    A(sat,sut+1)=par1;

    A(sat+2,sut-2)=par1;
    A(sat+2,sut-1)=par1;
    A(sat+2,sut)=par1;
    A(sat+2,sut+1)=par1;
    A(sat+2,sut+2)=par1;
    A(sat-2,sut+2)=par1;
    A(sat-1,sut+2)=par1;
    A(sat,sut+2)=par1;
    A(sat+1,sut+2)=par1;
    A(sat-2,sut-2)=par1;
    A(sat-2,sut-1)=par1;
    A(sat-2,sut)=par1;
    A(sat-2,sut+1)=par1;
    A(sat-1,sut-2)=par1;
    A(sat,sut-2)=par1;
    A(sat+1,sut-2)=par1;
 end
```

VERMATRIS.M

```
yartop=.0001*ones(12,12);
yisim='c:\erdem\egitim\akasya\dos1.dat';
dosya=fopen(yisim,'r');
```

```
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,1)=eger(ornek*i,1);
   yartop(1,1)=yartop(1,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,2)=eger(ornek*i,1);
   yartop(2,1)=yartop(2,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos5.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,3)=eger(ornek*i,1);
   yartop(3,1)=yartop(3,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos6.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,4)=eger(ornek*i,1);
   yartop(4,1)=yartop(4,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos7.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,5)=eger(ornek*i,1);
   yartop(5,1)=yartop(5,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,6)=eger(ornek*i,1);
   yartop(6,1)=yartop(6,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akasya\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akasya(i,7)=eger(ornek*i,1);
   yartop(7,1)=yartop(7,1)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akca\dos52.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akca(i,1)=eger(ornek*i,1);
   yartop(1,2)=yartop(1,2)+eger(ornek*i,1);
```

```
end
yisim='c:\erdem\egitim\akca\dos53.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akca(i,2)=eger(ornek*i,1);
   yartop(2,2)=yartop(2,2)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akca\dos54.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akca(i,3)=eger(ornek*i,1);
   yartop(3,2)=yartop(3,2)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akca\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akca(i,4)=eger(ornek*i,1);
   yartop(4,2)=yartop(4,2)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\akca\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   akca(i,5)=eger(ornek*i,1);
   yartop(5,2)=yartop(5,2)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\citlen\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   citlen(i,1)=eger(ornek*i,1);
   yartop(1,3)=yartop(1,3)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\citlen\dos28.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   citlen(i,2)=eger(ornek*i,1);
   yartop(2,3)=yartop(2,3)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\citlen\dos71.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   citlen(i,3)=eger(ornek*i,1);
   yartop(3,3)=yartop(1,3)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\citlen\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
```

```
for i=1:egitsec/ornek
   citlen(i,4)=eger(ornek*i,1);
   yartop(4,3)=yartop(4,3)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\citlen\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   citlen(i,5)=eger(ornek*i,1);
   yartop(5,3)=yartop(1,3)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos1.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,1)=eger(ornek*i,1);
   yartop(1,4)=yartop(1,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,2)=eger(ornek*i,1);
   yartop(2,4)=yartop(2,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos11.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,3)=eger(ornek*i,1);
   yartop(3,4)=yartop(3,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos13.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,4)=eger(ornek*i,1);
   yartop(4,4)=yartop(4,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos2.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,5)=eger(ornek*i,1);
   yartop(5,4)=yartop(5,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos23.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,6)=eger(ornek*i,1);
   yartop(6,4)=yartop(6,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos6.dat';
```

```
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,7)=eger(ornek*i,1);
   yartop(7,4)=yartop(7,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos7.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(8,1)=eger(ornek*i,1);
   yartop(8,4)=yartop(8,4)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\gul\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   gul(i,9)=eger(ornek*i,1);
   yartop(9,4)=yartop(9,4)+eger(ornek*i,1);
end

yisim='c:\erdem\egitim\igde\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   igde(i,1)=eger(ornek*i,1);
   yartop(1,5)=yartop(1,5)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\igde\dos11.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   igde(i,2)=eger(ornek*i,1);
   yartop(2,5)=yartop(2,5)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\igde\dos12.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   igde(i,3)=eger(ornek*i,1);
   yartop(3,5)=yartop(3,5)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\igde\dos14.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   igde(i,4)=eger(ornek*i,1);
   yartop(4,5)=yartop(4,5)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\igde\dos16.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
```

```
    igde(i,5)=eger(ornek*i,1);
    yartop(5,5)=yartop(5,5)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\igde\dos17.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    igde(i,6)=eger(ornek*i,1);
    yartop(6,5)=yartop(6,5)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\igde\dos3.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    igde(i,7)=eger(ornek*i,1);
    yartop(7,5)=yartop(7,5)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\igde\dos9.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    igde(i,8)=eger(ornek*i,1);
    yartop(8,5)=yartop(8,5)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\kagac\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    kagac(i,1)=eger(ornek*i,1);
    yartop(1,6)=yartop(1,6)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\kagac\dos10.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    kagac(i,2)=eger(ornek*i,1);
    yartop(2,6)=yartop(2,6)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\kagac\dos12.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    kagac(i,3)=eger(ornek*i,1);
    yartop(3,6)=yartop(3,6)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\kagac\dos14.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    kagac(i,4)=eger(ornek*i,1);
    yartop(4,6)=yartop(4,6)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\kagac\dos6.dat';
 dosya=fopen(yisim,'r');
```

```
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kagac(i,5)=eger(ornek*i,1);
   yartop(5,6)=yartop(5,6)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kagac\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kagac(i,6)=eger(ornek*i,1);
   yartop(6,6)=yartop(6,6)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos1.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,1)=eger(ornek*i,1);
   yartop(1,7)=yartop(1,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,2)=eger(ornek*i,1);
   yartop(2,7)=yartop(2,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos12.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,3)=eger(ornek*i,1);
   yartop(3,7)=yartop(3,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos18.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,4)=eger(ornek*i,1);
   yartop(4,7)=yartop(4,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos2.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,5)=eger(ornek*i,1);
   yartop(5,7)=yartop(5,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,6)=eger(ornek*i,1);
   yartop(6,7)=yartop(6,7)+eger(ornek*i,1);
```

```
end
yisim='c:\erdem\egitim\kartopu\dos5.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,7)=eger(ornek*i,1);
   yartop(7,7)=yartop(7,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos6.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,8)=eger(ornek*i,1);
   yartop(8,7)=yartop(8,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kartopu\dos7.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kartopu(i,9)=eger(ornek*i,1);
   yartop(9,7)=yartop(9,7)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos1.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,1)=eger(ornek*i,1);
   yartop(1,8)=yartop(1,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos12.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,2)=eger(ornek*i,1);
   yartop(2,8)=yartop(2,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos13.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,3)=eger(ornek*i,1);
   yartop(3,8)=yartop(3,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos15.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,4)=eger(ornek*i,1);
   yartop(4,8)=yartop(4,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos16.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
```

```
for i=1:egitsec/ornek
   kkavak(i,5)=eger(ornek*i,1);
   yartop(5,8)=yartop(5,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos18.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,6)=eger(ornek*i,1);
   yartop(6,8)=yartop(6,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos2.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,7)=eger(ornek*i,1);
   yartop(7,8)=yartop(7,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,8)=eger(ornek*i,1);
   yartop(8,8)=yartop(8,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos4.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,9)=eger(ornek*i,1);
   yartop(9,8)=yartop(9,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos5.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,10)=eger(ornek*i,1);
   yartop(10,8)=yartop(10,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,11)=eger(ornek*i,1);
   yartop(11,8)=yartop(11,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kkavak\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kkavak(i,12)=eger(ornek*i,1);
   yartop(12,8)=yartop(12,8)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos1.dat';
```

```
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,1)=eger(ornek*i,1);
   yartop(1,9)=yartop(1,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos2.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,2)=eger(ornek*i,1);
   yartop(2,9)=yartop(2,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,3)=eger(ornek*i,1);
   yartop(3,9)=yartop(3,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos4.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,4)=eger(ornek*i,1);
   yartop(4,9)=yartop(4,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos7.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,5)=eger(ornek*i,1);
   yartop(5,9)=yartop(5,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\kusburnu\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   kusburnu(i,6)=eger(ornek*i,1);
   yartop(6,9)=yartop(6,9)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\leylak\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   leylak(i,1)=eger(ornek*i,1);
   yartop(1,10)=yartop(1,10)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\leylak\dos11.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   leylak(i,2)=eger(ornek*i,1);
```

```
    yartop(2,10)=yartop(2,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\leylak\dos13.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    leylak(i,3)=eger(ornek*i,1);
    yartop(3,10)=yartop(3,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\leylak\dos15.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    leylak(i,4)=eger(ornek*i,1);
    yartop(4,10)=yartop(4,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\leylak\dos17.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    leylak(i,5)=eger(ornek*i,1);
    yartop(5,10)=yartop(5,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\leylak\dos4.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    leylak(i,6)=eger(ornek*i,1);
    yartop(6,10)=yartop(6,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\leylak\dos5.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    leylak(i,7)=eger(ornek*i,1);
    yartop(7,10)=yartop(7,10)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\fruz\dos10.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    fruz(i,1)=eger(ornek*i,1);
    yartop(1,11)=yartop(1,11)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\fruz\dos11.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    fruz(i,2)=eger(ornek*i,1);
    yartop(2,11)=yartop(1,11)+eger(ornek*i,1);
 end
 yisim='c:\erdem\egitim\fruz\dos14.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
```

```
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,3)=eger(ornek*i,1);
   yartop(3,11)=yartop(3,11)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\fruz\dos15.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,4)=eger(ornek*i,1);
   yartop(4,11)=yartop(4,11)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\fruz\dos2.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,5)=eger(ornek*i,1);
   yartop(5,11)=yartop(5,11)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\fruz\dos3.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,6)=eger(ornek*i,1);
   yartop(6,11)=yartop(6,11)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\fruz\dos8.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,7)=eger(ornek*i,1);
   yartop(7,11)=yartop(7,11)+eger(ornek*i,1);
end
yisim='c:\erdem\egitim\fruz\dos9.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   fruz(i,8)=eger(ornek*i,1);
   yartop(8,11)=yartop(8,11)+eger(ornek*i,1);
end
```

NORMATRIS.M

```
[aksmax,aks]=max(akasya(5:35,:));
[akcmax,akc]=max(akca(5:35,:));
[citmax,cit]=max(citlen(5:35,:));
[gulmax,glu]=max(gul(5:35,:));
[igdmax,igd]=max(igde(5:35,:));
[kacmax,kac]=max(kagac(5:35,:));
[karmax,krt]=max(kartopu(5:35,:));
[kkvmax,kkv]=max(kkavak(5:35,:));
[kusmax,kus]=max(kusburnu(5:35,:));
[leymax,ley]=max(leylak(5:35,:));
[frzmax,frz]=max(fruz(5:35,:));

for j=1:5
   for i=1:egitsec/ornek
```

```
   takasya(i,j)=45*akasya(mod(i+aks(j)-18,90)+1,j)/yartop(j,1);
   takca(i,j)=45*akca(mod(i+akc(j)-18,90)+1,j)/yartop(j,2);
   tcitlen(i,j)=45*citlen(mod(i+cit(j)-18,90)+1,j)/yartop(j,3);
   tgul(i,j)=45*gul(mod(i+glu(j)-18,90)+1,j)/yartop(j,4);
   tigde(i,j)=45*igde(mod(i+igd(j)-18,90)+1,j)/yartop(j,5);
   tkagac(i,j)=45*kagac(mod(i+kac(j)-18,90)+1,j)/yartop(j,6);
   tkartopu(i,j)=45*kartopu(mod(i+krt(j)-18,90)+1,j)/yartop(j,7);
   tkkavak(i,j)=45*kkavak(mod(i+kkv(j)-18,90)+1,j)/yartop(j,8);
   tkusburnu(i,j)=45*kusburnu(mod(i+kus(j)-18,90)+1,j)/yartop(j,9);
   tleylak(i,j)=45*leylak(mod(i+ley(j)-18,90)+1,j)/yartop(j,10);
   tfruz(i,j)=45*fruz(mod(i+frz(j)-18,90)+1,j)/yartop(j,11);
   end
end

 frakasya=10*diff(takasya);
 frakca=10*diff(takca);
 frcitlen=10*diff(tcitlen);
 frgul=10*diff(tgul);
 frigde=10*diff(tigde);
 frkagac=10*diff(tkagac);
 frkartopu=10*diff(tkartopu);
 frkkavak=10*diff(tkkavak);
 frkusburnu=10*diff(tkusburnu);
 frleylak=10*diff(tleylak);
 frfruz=10*diff(tfruz);
```

KAYDET.M

```
%yisim='c:\erdem\'+ yisim + '.dat';
dosya=fopen(yisim,'w');
fwrite(dosya,egitim,'float');
fclose(dosya);
```

YAPSIN.M

```
for i=1:egitsec/ornek
   pe(i,1:2)=[0 1];
end
nhedef=[ 1  1  0  1  0 ;
         0  1  1  0  1 ;
         0  0  1  1  1 ;
         1  0  1  0  1 ;
         0  1  0  1  1 ;
         1  0  1  0  0];
[w1,b1,w2,b2,w3,b3]=initff(pe,egitsec/ornek,'logsig',35,'logsig',6,'
logsig');
[w1,b1,w2,b2,w3,b3,te,tr]=trainbpx(w1,b1,'logsig',w2,b2,'logsig',w3,
b3,'logsig',negitim(1:egitsec/ornek,1:5),nhedef,[50,4000,0.000000000
00000000002,0.1001,1.07,0.7,0.9,1.04]);
[cik1,cik2,cik3]=simuff(negitim(1:egitsec/ornek,1),w1,b1,'logsig',w2
,b2,'logsig',w3,b3,'logsig');
```

YAPHOP.M

```
%vermatris;
%normatris;
negitim=[tkkavak(:,3) tkagac(:,1) takasya(:,2)];% gul(:,1) igde(:,1)
kagac(:,1) kartopu(:,1) kkavak(:,1) kusburnu(:,1) leylak(:,1)];
[w,b]=solvehop(negitim);

denek=tkagac(:,1);

f5=sumsqr(denek-negitim(:,1))
```

```
f6=sumsqr(denek-negitim(:,2))
f7=sumsqr(denek-negitim(:,3))

indeks=1:egitsec/ornek;
figure(4);
plot(indeks,denek);
a=denek;
for i=1:10
  [a,aa]=simuhopt(a,w,b,2);
  fark=negitim(:,1)-a;
  f1=sumsqr(fark)
  fark=negitim(:,2)-a;
  f2=sumsqr(fark)
  fark=negitim(:,3)-a;
  f3=sumsqr(fark)
  f4=sumsqr(a-denek)
  pause;

figure(5);
plot(indeks,negitim(:,1));
figure(6);
plot(indeks,negitim(:,2));
figure(7);
plot(indeks,negitim(:,3));
figure(8);
plot(indeks,a);
end
```

EGITVEK.M

```
disp 'eğitim kümeniz hangi verilerden oluşturulsun ?';
disp 'yalnız maximum uzunluklarla   (180)';
disp 'maximum ve minimum uzunluklarla  (360)';
egitsec=input('seciminiz:  ');
disp 'kaç derecede bir örnek alacaksınız ?(çift sayı
giriniz...2,4,6...)';
ornek=input('seciminiz:  ');
ornek=ornek/2;
yisim='c:\erdem\egitim\akasya\dos7.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
egnormalt=zeros(10,1);
for i=1:egitsec/ornek
   negitim(i,1)=eger(ornek*i,1);
   egnormalt(1,1)=egnormalt(1,1)+negitim(i,1);
end
yisim='c:\erdem\egitim\akca\dos52.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   negitim(i,2)=eger(ornek*i,1);
   egnormalt(2,1)=egnormalt(2,1)+negitim(i,2);
end
yisim='c:\erdem\egitim\citlen\dos10.dat';
dosya=fopen(yisim,'r');
eger=fread(dosya,'float');
fclose(dosya);
for i=1:egitsec/ornek
   negitim(i,3)=eger(ornek*i,1);
```

```
    egnormalt(3,1)=egnormalt(3,1)+negitim(i,3);
 end
 yisim='c:\erdem\egitim\gul\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,4)=eger(i*ornek,1);
    egnormalt(4,1)=egnormalt(4,1)+negitim(i,4);
 end
 yisim='c:\erdem\egitim\igde\dos10.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,5)=eger(ornek*i,1);
    egnormalt(5,1)=egnormalt(5,1)+negitim(i,5);
 end
 yisim='c:\erdem\egitim\kagac\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,6)=eger(ornek*i,1);
    egnormalt(6,1)=egnormalt(6,1)+negitim(i,6);
 end
 yisim='c:\erdem\egitim\kartopu\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,7)=eger(ornek*i,1);
    egnormalt(7,1)=egnormalt(7,1)+negitim(i,7);
 end
 yisim='c:\erdem\egitim\kkavak\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,8)=eger(ornek*i,1);
    egnormalt(8,1)=egnormalt(8,1)+negitim(i,8);
 end
 yisim='c:\erdem\egitim\kusburnu\dos1.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,9)=eger(ornek*i,1);
    egnormalt(9,1)=egnormalt(9,1)+negitim(i,9);
 end
 yisim='c:\erdem\egitim\leylak\dos10.dat';
 dosya=fopen(yisim,'r');
 eger=fread(dosya,'float');
 fclose(dosya);
 for i=1:egitsec/ornek
    negitim(i,10)=eger(ornek*i,1);
    egnormalt(10,1)=egnormalt(10,1)+negitim(i,10);
 end
 for i=1:10
    negitim(:,i)=(.5*egitsec/ornek)*negitim(:,i)/egnormalt(i,1);
 end
```

```
nhedef=[.1  .2  .3  .4  .5  .6  .7  .8  .9  1 ;
        .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ;
        .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ;
        .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ;
        .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ];
```