T.C.
MUĞLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

YAPAY SİNİR AĞ MODELLERİ VE SINIFLANDIRMAYA UYGULANMASI

YÜKSEK LİSANS TEZİ

NİDA GÖKÇE

ŞUBAT 2005
MUĞLA

Prof.Dr.Mübariz EMİNOV danışmanlığında Nida GÖKÇE tarafından hazırlanan bu çalışma, 25 / 02 / 2005 tarihinde aşağıdaki jüri tarafından İstatistik ve Bilgisayar Bilimleri Anabilim Dalı'nda yüksek lisans tezi olarak oybirliği ile kabul edilmiştir.

Başkan	: Yrd.DoçDr.B. Taner DİNÇER	İmza :
Üye	: Prof.Dr.Mübariz EMİNOV	İmza :
Üye	: Yrd.Doç.Dr.Mahmut TENRUH	İmza :

ÖNSÖZ

Bu çalışmada Türkiye'de bulunan 81 ilin sağlık verileri kullanılarak sınıflandırılması ve illerin gelişmişlik sınıflarının belirlenmesi amaçlanmıştır. Sınıflandırma işlemi için danışmansız eğitimli sinir ağ modellerine uygulanan öğrenme algoritmalarından uyarlamalı yarışmacı öğrenme algoritması seçilmiştir.

Çalışmam sırasında değerli zamanını ve yardımlarını esirgemeyen çok değerli hocam Prof.Dr.Mübariz EMİNOV'a sonsuz teşekkürlerimi sunuyorum. Sevgili arkadaşım Araş.Gör.Nevin GÜLER'e yardımlarından dolayı teşekkür ediyorum. Ayrıca değerli hocam Prof.Dr.Mustafa DİLEK'e ve sevgili arkadaşlarım Araş.Gör.Tuba SAKARYA BEZ, Araş.Gör.Hatice GÖRGÜLÜ, Araş.Gör.Seval ERKENCİ, Araş.Gör.Aytaç PEKMEZCİ ve Araş.Gör.Serkan BALLI'ya çalışmalarım sırasında verdikleri destek için çok teşekkür ediyorum.

Son olarak hayatımın her döneminde tüm sevgi ve destekleriyle yanımda olan, beni yüreklendiren sevgili aileme sonsuz teşekkürlerimi sunuyorum.

Nida GÖKÇE

YAPAY SİNİR AĞ MODELLERİ VE SINIFLANDIRMAYA UYGULANMASI

(Yüksek Lisans Tezi)

Nida GÖKÇE

MUĞLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

2005

ÖZET

Sınıflandırma, yapılandırılmamış veri setlerindeki veri noktalarını mümkün olduğunca küme içinde homojen ve kümeler arasında heterojen olacak şekilde gruplara ayırarak veri yapısını ortaya çıkarma işlemidir. Sınıflandırma genelde, danışmanlı ve danışmansız olmak üzere iki yaklaşımla gerçekleştirilmektedir. İstatistiksel sınıflandırma yöntemleri ise danışmansız öğrenme yaklaşımını temel almaktadır.

Bu tezde yapılandırılmamış çok değişkenli verileri sınıflandırma problemine çözüm getirebilecek hem istatistiksel hem de kümeleme analizine dayanan sınıflandırma yöntemleri incelenmiş ve kümeleme analizine dayanan danışmansız yapay sinir ağ (YSA) mimarilerinin eğitiminde kullanılan yarışmacı öğrenme algoritmasının kullanımı ele alınmıştır. Girdi verileri ve ağırlık vektörünün standartlaştırılma biçiminin farklılığına bağlı olarak kullanılmakta olan uzaklığa dayanan ve açıya dayanan YÖA'larla YSA'ların eğitimi detaylı olarak incelenmiştir. Ağırlık vektörlerinin başlangıç değerlerinin rasgeleliğinden kaynaklanan sınıflandırma tutarsızlığını gidermek için bu algoritmalar geliştirilerek uyarlamalı özelliğe sahip yetenek kazandırılmıştır.

Önerilen iki farklı uyarlamalı YÖA Türkiye'deki 81 ilin sağlıkla ilgili verilerine uygulanmış ve bu illerin sağlık bazında gelişmişlik grupları tespit edilmiştir. Elde edilen sınıflandırma sonuçlarının tutarlı ve gerçekçi olduğu ve ayrıca açıya dayalı YÖA ile daha iyi sınıflandırma yapıldığı ortaya konulmuştur.

Anahtar Kelimeler: Yapay Sinir Ağları, Danışmanlı Öğrenme, Danışmansız Öğrenme, Yarışmacı Öğrenme, Ağırlık Vektörü, Sınıflandırma, Kümeleme Analizi.

Sayfa Adedi: 117

Tez Yöneticisi: Prof. Dr. Mübariz EMİNOV

NEURAL NETWORK MODELS
and
APLICATION to CLASSIFICATION

(Ph. M. Thesis)

Nida GÖKÇE

MUGLA UNIVERSITY
INSTITUTE of SCIENCE and TECHNOLOGY

2005

ABSTRACT

Classification is a process which separates the data points in the unstructured data sets into groups which are to be homogeneous in the clusters and heterogeneous among the clusters as possible and this process makes the data structure being exposed to view. Classification is generally realized by two approaches called as supervised and unsupervised. Statistical classification methods are based on supervised learning. Classification methods depending on clustering analysis is based on unsupervised learning approach.

In this thesis, in order to solve the classification problem of unstructured multivariable data, both statistical and classification analysis methods are studied. The training of unsupervised NN architectures are based on clustering analysis and among these, the competitive learning algorithm is tackled in this study. Input data and weight vector are standardized in different ways. Depending on these differences, this study investigates training of NN through CLA which is based on distance and angle. In order to remove the classification inconsistency arising from random of initial values of weight vector these algorithms have been given the ability of adaptivity features by being developed.

The two different suggested adaptivity CLA have been applied to data related to health in 81 cities in Turkey and the classification of groups in line with the extend of development in the field of health is obtained. It is put forward that the classification results obtained are valid and reliable and that a beter classification is performed by the adaptivity CLA which is based on angle.

Key Words: Neural Network, Supervised Learning, Unsupervised Learning, Competitive Learning, Weight Vector, Classification, Clustering Analysis.

Page Number: 117

Adviser : Prof. Dr. Mübariz EMİNOV

ŞEKİLLER DİZİNİ

TABLOLAR / ÇİZELGELER DİZİNİ

Tablo No

SEMBOLLER VE KISALTMALAR DİZİNİ

ρ : Çarpım moment korelasyon katsayısı

σ_{ij} : Kovaryans matrisi elemanları

$\bar{x}$: Ortalama değer

μ_p : Ortalama vektörü

η : Öğrenme oranı

μ_A :A bulanık kümesinin üyelik fonksiyon değeri

R_{ξ} :İçsel yayılma değerlerine bağlı test istatistiği

ADALINE : Uyarlanabilir doğrusal eleman

a_{kn} : gruplar içi küme merkezleri

B : Kümeler arası kovaryans matrisi

C : Calinsky katsayısı

C : Küme ayırma kriteri

D : Uzaklık matrisi

d_{ij} : İki vektör arasındaki uzaklık

$d_{k(i,j)}$: Birleştirilen i ve j kümelerinin k'ıncı kümeye uzaklığı

d_{mj} : m kümesi ile j kümesi arasındaki uzaklık

$f(x)$: Aktivasyon fonksiyonu

f_{old} : İkili sigmoid fonksiyon

k : Küme sayısı

M : Mariot katsayısı

MADALINE: İki veya daha fazla uyarlanabilir doğrusal eleman

$Max(d_{ki},d_{kj})$: Kümeler arasındaki uzaklığın maksimum olması

n : Eleman sayısı

OrtBKY: Ortalama Bağlantı Kümeleme Yöntemi

p : Değişken sayısı

${}_pSx_{o,n}$: S matrisinin istatistiksel yayılım ölçüsü

R : Range, dağılım aralığı

Sim : Benzerlik matrisi

T : Toplam kovaryans matrisi

TekBKY: Tek bağlantılı kümeleme yöntemi

tr : trace, bir matrisin izi

U : Üyelik matrisi

u_{ij} : U üyelik matrisinin elemanları

W : Küme içi kovaryans matrisi

X : Veri matrisi

x_i : İlleri temsil eden veriler

x_{ij} : Veri matrisinin elemanları

X_{max} : Bir dizideki en büyük değer

X_{min} : Bir dizideki en küçük değer

Z : Dönüştürülmüş Z matrisi

z_i : Z skorlara dönüştürülmüş veriler

SÖZLÜK

Adaptif Lineer Element: Uyarlanabilir doğrusal eleman

Agglomerative: Birleştirici

Artificial Neural Network Models: Yapay Sinir Ağ Modelleri

Associative Memory: Çağrışımsal bellek

Avarage Linkage Method: Ortalama Bağlantı Kümeleme Yöntemi

Backpropagation: Geri yayılım

Centroid Linkage Method: Küresel ortalama bağlantı kümeleme yöntemi

Centroid: Merkezi, ortalama

Classification : Sınıflandırma

Clustering Analysis: Kümeleme analizi

Competitive learning: Yarışmacı öğrenme

Data association: Veri ilişkilendirme

Data conceptualization: Veri kavramlaştırma

Data filtering: Veri filtreleme

Discriminant Analysis: Ayırma analizi

Discrimination: Ayrım

Dissimilarity : Farklılık

Distance : Uzaklık

Distributed: Dağıtık

Divisive: Ayırıcı

Feedback: Geri besleme

Feedforward: İleri besleme

Furthest Neighbor Complete Linkage (CLINK): En uzak komşuluk tam bağlantı kümeleme yöntemi

Fuzzy c-means: Bulanık c-Ortalamalar

Grup centroid: Küme merkezi, tahmini grup ortalama vektörü

Haphazard: Gelişigüzel

Hard limiter function: Eşik fonksiyonu

Heterojen: Farklı

Hidden layer: Gizli katman

Hierarchical : Hiyerarşik, Aşamalı

Hill Climbing Method: Yığın kümeleme yöntemi

Homojen: Türdeş

Impossible: Olanaksız

Input : Girdi

Input layer: Giriş katmanı

Instute of Electrical Electronic Engineering (IEEE): Elektrik Elektronik Mühendisliği Enstitüsü

Internal scatter: İçsel yayılım

Iteration: Döngü

Iterative method: Yinelemeli yaklaşım

Least mean square: En küçük kareler ortalaması

Lineer function: Doğrusal fonksiyon

MacQueen's k-Means Method: k-Ortalamalar yöntemi

Maximum gap: Büyük aralık

McQuitty Linkage Method: McQuitty bağlantı kümeleme yöntemi

Median Linkage Method: Ortanca bağlantı kümeleme yöntemi

Metric: Teknik

Multilayer perceptron: Çok katmanlı perceptron

National Academy of Science of The USA : Amerikan Ulusal Bilim Akademisi

Nearest Neighbor-Single Linkage (SLINK): En yakın komşuluk yöntemi

Neural Networks : Sinir ağları

Noise: Gürültü, Kirlilik

Nonhierarchical : Hiyerarşik olmayan, aşamalı olmayan

Non-lineer function: Doğrusal olmayan fonksiyon

Neuron: Nöron, sinir, düğüm

Numerical: Sayısal

Optimum: En iyi

Outliers: Aykırı değerler

Output : Çıktı

Output layer: Çıktı katmanı

Prediction: Tahmin

Propagation: İlerleme, yayılma

Random : Rastgele

Range: Dağılım aralığı

Recurrent: Yinelenen

Reinforcement learning: Pekiştirerek öğrenme

Seed points: Çekirdek noktalar

Self Organizing Feature Map: Özdüzenlemeli özellik haritaları

Similarity : Benzerlik

Summation function: Toplama fonksiyonu

Supervised Learning : Danışmanlı öğrenme, Denetimli öğrenme

Threshold: Eşik değeri

Trace: İz

Transfer function: Aktivasyon fonksiyonu

Unsupervised Learning : Danışmansız öğrenme, Denetimsiz öğrenme

Update: Güncelleştirmek

Variable: Değişken

Very Large Scale Integration (VLSI): Çok geniş ölçekli devre

Ward Linkage Method: Ward bağlantı kümeleme yöntemi

Weight Vektor : Ağırlık vektörü

1. GİRİŞ

Yapılandırılmamış veri setinin veri grup sayısını belirleyip benzer özelliklere sahip veri noktalarından oluşan verileri gruplara (sınıflara) ayırmak ve veri setinin yapısını ortaya çıkarmak, bilimsel araştırmalarda büyük bir önem taşımaktadır. Böyle bir hedefe ulaşma, farklı yöntemlerle yapılabilen veri analizleri vasıtasıyla gerçekleştirilmektedir. Bu yöntemler genellikle danışmanlı ve danışmansız öğrenme olmak üzere iki temel yaklaşıma dayanır. Danışmanlı öğrenme yaklaşımını temel alan istatistiksel sınıflandırma yöntemleri eğitici örnekler kümesi kullanılarak grup (sınıf) sayısı ve grup içi istatistiksel parametrelerin (ortalama, kovaryans matrisi vb.) olasılığa dayalı olarak önceden belli olduğunu öngörür. Temel Bileşenler Analizi, Diskriminant Analizi vb. çok değişkenli istatistiksel yöntemlerle veride boyut indirgeme uygulayarak, gruplar arası sapmayı maksimize edecek şekilde sınıflandrıma yapabilirler. Veri seti hakkında ön bilgilerin bulunamadığı veya daha kesin sınıflandırmanın gerektiği durumlarda istatistiksel sınıflandırma yöntemleri yetersiz kalır. Böyle zorlukların üstesinden gelmek için danışmansız öğrenme yaklaşımına dayanan kümeleme analizi yöntemleri uygulanmaktadır. Bu yöntemlerle veri analizi esnasında hiçbir ön bilgi bulunmadan veri seti küme içi homojenliği ve kümeler arası heterojenliği sağlayacak şekilde gruplara ayrılır. Kümeleme Analizi, algoritmik (k-Ortalamalar, Bulanık c-Ortalamalar, Yapay Sinir Ağ Modelli Öğrenme), hiyerarşik, sezgisel vb. farklı yöntemlerle yapılmaktadır.

Bu çalışmada gerek istatistiksel yöntemlerle gerekse kümeleme analizine dayanan yöntemlerle sınıflandırma problemleri incelenmiş ve bu yöntemlerle verilerin sınıflandırması açıklanmıştır. Bu kapsamda Yapay Sinir Ağlarının (YSA) aşağıda bahsedilen avantajları göz önünde bulundurularak sezgisel kümeleme analizi yapabilen danışmansız (unsupervised) yapay sinir ağ mimarileri ve onların eğitilmesi için kullanılan ilgili öğrenme algoritmaları sunulmuştur.

YSA'nın diğer sınıflandırma modellerine göre sağladığı avantajlar kısaca şöyle açıklanabilir. YSA'lar bulundukları ortama kolayca uyum sağlayabilme yeteneğine sahiptirler. Belli bir ortam için eğitilmiş bir sistem ortam değişikliği durumunda yeniden eğitilebilir. Bilgiyi dağıtık (distributed) saklama özelliği sayesinde bozuk ve eksik verilerle çalıştırıldıklarında bile doğruya yakın sonuçlar üretebilirler. Paralel

işlem olanağı sayesinde işlem zamanını azaltırlar. Adaptasyon yeteneği sayesinde kendi kendilerini eğitebilme ve dinamik sistemler gibi kararlı hale gelebilme özelliğine sahiptirler. YSA'ların belirtilen avantajları göz önünde bulundurularak danışmansız YSA mimarileri içerisinden veri sınıflandırma için en yaygın kullanılan yarışmacı öğrenmeye dayanan YSA modeli tezde kullanılmak üzere temel alınmış ve detaylı olarak incelenmiştir.

Yarışmacı Öğrenme (Competitive Learning) Algoritması ile YSA'nın eğitimi, girdi vektörleri ve kümeye ilişkin ağırlık vektörlerinin standartlaştırma biçimine (her bileşenin ayrı ayrı standartlaştırılması veya bileşenlerin tümünü birlikte dikkate alarak birim vektöre dönüştürme yoluyla standartlaştırılması) bağlı olarak sırasıyla uzaklığa ve açıya dayanan farklı versiyonlarının eğitim süreci detaylı araştırılmıştır. Verilerin standartlaştırılma biçimine göre sırasıyla uzaklığa dayanan ve açıya dayanan iki çeşit Yarışmacı Öğrenme Algoritması (YÖA) incelenmiş ve açıya dayanan YÖA'nın üstünlüğü dikkate alınarak bu algoritmanın sınıflandırmada kullanılması uygun görülmüştür. YÖA'da ağırlık vektörlerinin rastgele üretilen başlangıç değerlerinin kümeleme sonuçlarına olan etkisinin ortadan kaldırılması amacıyla bu algoritma geliştirilerek uyarlamalı YÖA önerilmiştir.

Bu çalışmada geliştirilen uyarlamalı YÖA Türkiye'deki 81 ilin sağlık bazında gelişmişlik gruplarına ayrılmasına uygulanmıştır. Belirlenen gelişmişlik gruplarına göre illerin sınıflandırılması, geliştirilen algoritma yardımıyla sağlıkla ilgili veriler kullanılarak gerçekleştirilmiştir.

2. SINIFLANDIRMA YÖNTEMLERİ

Sınıflandırma, birbirine benzer bireylerin bir araya getirilmesi işlemidir. Sınıflandırma işlemi iki aşamalıdır. İlk aşamada kümeleme yöntemleri kullanılarak birbirine benzer bireylerden meydana gelen kümeler oluşturulur. İkinci aşamada ise bireylerin ait oldukları sınıflar belirlenir. İlk aşamada kullanılan kümeleme yöntemlerinin amacı verileri benzerliklerine göre bir araya getirip mümkün olduğunca kendi içinde homojen, birbirleri arasında heterojen gruplar oluşturmaktır. Klasik mantık kullanılarak yapılan sınıflandırmalarda kümeleme ve sınıflandırma aynı anlamda kullanılabilir. Bulanık kümelemede ise her eleman bir üyelik fonksiyonu ile bütün gruplara dahil olabilir, fakat en büyük üyelik fonksiyonu ile girdiği kümeye atanarak sınıflandırma işlemi yapılmış olur.

Genel olarak sınıflandırma yöntemleri, kümeleme sırasında eğitici veri kümesinin kullanılıp kullanılmamasına göre danışmanlı yaklaşımla sınıflandırma ve danışmansız yaklaşımla sınıflandırma olarak ikiye ayrılır. Eğitici veri kümeleri kullanarak sınıflandırma yapan yöntemler danışmanlı sınıflandırma yöntemleridir. Danışmanlı yaklaşımla sınıflandırma yöntemlerine istatistiksel kümeleme yöntemlerinden Temel Bileşenler Analizi, Faktör Analizi ve Ayırma Analizi örnek verilebilir. Danışmansız yaklaşımla sınıflandırma yöntemleri ise Kümeleme Analizi yöntemleridir.

2.1. Temel Bileşenler Analizi

Temel Bileşenler Analizi n bireye ilişkin p değişkeni inceleyen çok değişkenli istatistiksel analiz yöntemlerinden biridir. İstatistiksel yöntemlerde değişken sayısı arttıkça işlem yükü artmakta ve sonuçların yorumlanması güçleşmektedir. Bu nedenle değişken sayısını azaltmak gerekir. Temel bileşenler analizi değişkenler arasındaki bağımlılık yapısını yok etmeyi ve boyut indirgemeyi amaçlayan bir analiz yöntemidir.

$$X = (x_{ij});\ i = 1, 2, \ldots, n\ ;\ j = 1, 2, \ldots, p \qquad (2.1)$$

n birey ve p değişkenden oluşan veri matrisi X'in p boyutlu uzaydaki durumu düşünülecek olursa, veri matrisi çok sayıda noktadan oluşan bir topluluk veya bulut olarak ifade edilebilir. Değişkenler arasında tam bağımsızlık söz konusu olamayacağı

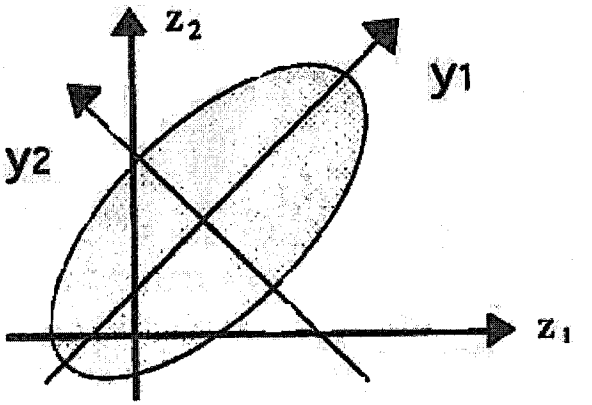
için bulut biçiminde ifade edilen geometrik şeklin eksenleri birbirlerine dik olmayacak ve tanımı da yapılamayacaktır. Oysa ki bu noktaları eksenleri birbirine dik bir elipsoid içerisine almak daha ayrıntılı ve açıklayıcı bilgi verecektir. Bu amaçla uygulanan dönüştürmede, noktaların ilk eksenler boyunca sahip oldukları toplam varyans değişmediği gibi yeni eksenler de birbirlerine dik olmaktadır. Harold Hotelling tarafından önerilen teknikte X_{pxn} ham veri matrisi doğrudan kullanılabildiği gibi Z_{pxn} biçiminde ifade edilen standartlaştırılmış değerler matrisi de kullanılabilmektedir. Ham veri matrisinin kullanılması durumunda temel bileşenlerin bulunmasında varyans-kovaryans matrisinden, standartlaştırılmış verilerin kullanılması durumunda ise korelasyon matrisinden yararlanılmaktadır. Oldukça farklı sonuçlar verebilen bu iki yoldan hangisinin seçileceği konusunda en önemli belirleyici, değişkenlerin ölçü birimleridir. Eğer değişkenlerin ölçü birimleri ve varyansları birbirine yakın ise kovaryans matrisinden değilde korelasyon matrisinde yararlanılması önerilmiştir (Tatlıdil, 2002).

2.1.1. Temel Bileşenlerin Elde Edilmesi

Değişkenlerin ölçü birimlerinin birbirine yakın olması çok karşılaşılan bir durum olmadığı için temel bileşenlerin belirlenmesinde genellikle standartlaştırılmış değerlerden oluşan Z_{pxn} standart matrisi kullanılır. Değişkenler arasındaki tam bağımlılık yapısını ortadan kaldırmak amacıyla yapılan dönüştürme, T_{pxp} bir dönüşüm matris olmak üzere,

$$Y_{pxn} = T'_{pxp} Z_{pxn} \tag{2.2}$$

biçiminde yapılmaktadır. Yani birbiriyle ilişkili z_{ij} değerlerinden dönüştürme sonucunda, birbirleriyle ilişkisiz y_{ij} değerlerine ulaşılmaktadır. Bu dönüşüm Şekil (2.1) de gösterilmiştir.



Şekil 2.1: Temel Bileşenler Dönüşümünün İki Boyutlu Uzayda Gösterimi

Dönüşüm sonunda elde edilen Y matrisinin ortalama vektörü ve kovaryans matrisi,

$$E(Y) = E(T'Z) = T'E(Z) = 0 \tag{2.3}$$

$$Var(Y) = T'E(ZZ')T = T'RT \tag{2.4}$$

dir. Burada R: pxp boyutlu değişkenler arası korelasyon matrisidir. Dönüştürülmüş Y matrisinin değişken vektörlerinin birbirine dik olabilmesi için $Var(Y)$ matrisinin köşegen olması gerekir. Bu matrisin köşegenleştirilmesinde çok sayıda T dönüşümü matrisinin kullanımı söz konusudur. Birbirlerinden farklı bu dönüşüm matrislerinden amaca en uygun olanının seçilebilmesi için y vektörleri üzerinde bazı kısıtlayıcıların konması gerekir. Bu kısıtlayıcılar şöyledir:

- y vektörlerinin ilki olan y_1 öyle seçilmelidir ki varyansı maksimum olsun.

$$Var(y_1) = \max \frac{1}{n-1} \sum_{i=1}^{n} (y_{1i})^2 \tag{2.5}$$

olmalıdır.

- y_1 vektörünün bulunmasında kullanılan t_1 vektörünün elemanlarının kareleri toplamı 1 olmalıdır.

$$t_1' t_1 = 1 \tag{2.6}$$

Bu kısıtlayıcılar yardımıyla z_i vektöründen, dönüşüm sonucu elde edilen y_1 vektörünün i'inci elemanı,

$$y_{1i} = t_1' z_i \tag{2,7}$$

biçiminde bulunur. İlk kısıtlayıcı nedeniyle y_1 vektörünün varyansı,

$$\frac{1}{n-1}\sum_{i=1}^{n}(y_{1i})^2 = \frac{1}{n-1}\sum_{i=1}^{n}(t_1' z_i)^2 = \frac{1}{n-1}\sum_{i=1}^{n} t_1' z_i z_i' t_1$$
$$Var(y_1) = t_1' \frac{1}{n-1} ZZ' t_1 = t_1' R t_1 \tag{2.8}$$

olarak bulunur. y_1 vektörünün varyans değeri olan $t_1' R t_1$'in ikinci kısıtlayıcıdan da yararlanılarak maksimum yapılması söz konusudur.

Bu amaçla,

$$\varphi_1 = t_1' R t_1 - \lambda_1 (t_1' t_1 - 1) \tag{2.9}$$

fonksiyonu verilen kısıt altında çözülür. Fonksiyonun t_1'e göre türevi alınıp sıfıra eşitlenecek olursa;

$$\frac{\partial \varphi_1}{\partial t_1} = 2Rt_1 - 2\lambda_1 t_1 = 0 \Rightarrow (R - \lambda_1 I) t_1 = 0 \tag{2.10}$$

elde edilir. Bu bağlantıda λ_1 değeri R matrisinin özdeğeri (eigen value), t_1 vektörü de R matrisinin özvektörü (eigen vector) olarak adlandırılır (Tatlıdil, 2002).

Öz değerleri elde etmek için;

$$|R - \lambda I| = 0 \tag{2.11}$$

açılımından elde edilen p'inci dereceden polinom denklemden p tane λ değeri bulunur. R matrisi pozitif tanımlı ve simetrik olduğu için elde edilecek değerlerin tümü gerçek değerler olacaktır. Yukarıdaki bağıntıdan elde edilen p tane özdeğer kullanılarak her birine karşılık gelen p tane özvektör elde edilir.

(2.11) nolu bağıntı kullanılarak elde edilen λ_j'lerden birini λ_1 olarak ve ilgili vektörü de t_1 olarak gösterilsin. Bu bağıntı soldan t_1' ile çarpılacak olursa,

$$t_1' R t_1 - \lambda_1 t_1' t_1 = 0 \tag{2.12}$$

elde edilir. Buradan (2.6) bağıntısı nedeniyle $(t_1' t_1 = 1), t_1' R t_1 = \lambda_1$ olacaktır.

Sonuç olarak;

$$Var(y_1) = Var(\sqrt{\lambda_1} t_1) = E(\sqrt{\lambda_1} t_1)(\sqrt{\lambda_1} t_1)' = \lambda_1 t_1 t_1' = \lambda_1 \tag{2.13}$$

bulunur. Yani, y_1 değişkeninin varyansı λ_1'dir. Temel bileşenler analizinde y_1'in varyansının en büyük olması istendiğinden, λ_1 değeri λ_j değerleri arasında en büyük

değerli olarak seçilir. Seçilen λ_1 değerinin kullanımı ile elde edilen t_1 vektörüne birinci özvektör adı verilir. Birinci özdeğer λ_1 ve birinci öz vektör t_1 olmak üzere; t_1 ile orijinal veri matrisi Z'nin çarpımından elde edilen $y_1 = t_1'Z$ dönüştürülmüş vektöre de birinci temel bileşen ya da birinci skor vektörü adı verilir.

İkinci temel bileşen y_2 bulunurken, y_1 vektörünün bulunmasında kullanılan iki kısıtlayıcı yanında üçüncü bir kısıtlayıcı da göz önüne alınır. Bu kısıtlayıcılar

- y_2 vektörünün varyansı y_1'den sonra en büyük olsun,
- t_2 vektörü birim normal bir vektör olsun ($t_2't_2 = 1$),
- y_1 ve y_2 vektörleri birbirine dik olsun ($t_2't_1 = 0$),

biçimindedir (Tatlıdil, 2002).

Bu biçimde devam edilecek olursa, $j = 1, 2, ..., p$ için tüm λ_j, t_j ve y_j değerleri elde edilir. Bu durumda λ_p en küçük değere sahip özdeğer ve y_p ise en küçük varyanslı temel bileşendir.

Özdeğerlerin bulunmasından sonra önemli özdeğer sayısına *(m)* karar vermek gerekir. Bu amaçla geliştirilmiş bir çok yöntem vardır. Bilinen en basit yöntemde, 1 den büyük değerli özdeğerlerin sayısı m sayısını vermektedir veya yaklaşık aynı mantığa dayanan $(\sum_{j=1}^{m} \lambda_j / p) \geq (2/3)$ koşulunun sağlandığı en küçük m değeri önemli temel bileşenlerin sayısı olarak belirlenmektedir.

2.2. Faktör Analizi

Faktör analizi p boyutlu bir uzayda birbiriyle ilişkili değişkenleri bir araya getirerek, az sayıda yeni ilişkisiz değişken bulmayı amaçlar. Temel bileşenler analizi gibi bir boyut indirgeme ve bağımlılık yapısını yok etme yöntemidir.

Faktör analizinde de yine kovaryans matrisi ya da korelasyon matrisi kullanılır. Çok sayıda ilişkili orjinal değişkenden az sayıda ilişkisiz hipotetik değişken bulmayı amaçlayan faktör analizinde, n bireyin p tane özelliğini gösteren X_{pxn} ham veri matrisinden elde edilen Z_{pxn} standartlaştırılmış veri matrisi kullanılmaktadır. Bu durumda faktör analizi modelinin z_j değişkenleri ile $f_1, f_2, ..., f_m$ ortak faktörleri arasındaki ilişkiyi gösteren doğrusal bir model olduğu söylenebilir. Bu model genel olarak aşağıdaki biçimde ifade edilir.

$$z_j = a_{j1}f_1 + a_{j2}f_2 + \ldots + a_{jm}f_m + b_j u_j \; ; j = 1,2, \ldots,p \qquad (2.14)$$

Buradaki a_{jm} katsayılarına j'inci değişkenin m'inci faktör üzerindeki yükü veya ağırlığı adı verilir. Yukarıda tanımlanan ortak faktöre hipotetik çıkarım veya hipotetik değişken adı verilir. Modeldeki u_j değişkenine özel ya da artık faktörü adı verilirken b_j ise artık faktöre ilişkin katsayıdır. Yöntemde asıl amaç pxm boyutlu $A=(a_{jm})$ yükler matrisini elde edilmesidir.

İyi bir faktör analizi sonucu indirgenmiş boyut, yaklaşık bağımsızlık ve kavramsal anlamlılık koşullarını sağlaması gerekir.

Faktör analizi temel bileşenler analizi gibi veri setini, başlangıçtaki boyuttan daha küçük sayıda boyutla açıklamayı amaçlayan çok değişkenli bir analiz tekniğidir. Temel bileşenler analizinde olduğu gibi faktör analizinde de orijinal değişkenlerden, bağımsız yeni değişkenlerin elde edilmesi çoğu kez birincil amaç olmakla birlikte bu iki teknik arasında bazı önemli farklılıklar bulunmaktadır. Bu farklılıklardan ilki temel bileşenler analizi, verilerin kovaryans matrisinin biçimi üzerinde herhangi bir varsayım yapılmaksızın verilerin dönüşümünü amaçlarken, faktör analizinde verilerin tanımlanmış bir modele uyduğu varsayılmaktadır ve bu varsayım ortak faktörler ile özel faktörlerin aşağıdaki koşulları sağlama zorunluluğu getirilmektedir.

$$E(f) = 0; \qquad E(u) = 0; \qquad Var(f) = I;$$

$$Kov(u_i,u_j) = 0; \quad i \neq j \text{ iken} \qquad (2.15)$$

$$Kov\,(f,u) = 0$$

Bu koşulların sağlanmaması durumunda faktör analizinde doğru olmayan sonuçlara ulaşılabilmektedir.

İkinci farklılık ise temel bileşenler analizi, gözlenmiş değişkenlerden temel bileşenlere $Y = T'Z$ biçimindeki bir dönüşümü hedef alırken, faktör analizinde belirlenmiş faktörlerden gözlenmiş değişkenlere $Z=AF$ biçiminde bir dönüşüm öngörülmektedir. Ayrıca faktör analizinin ölçekten bağımsız olması ve her bir faktörün varyansları 1 olacak biçimde standartlaştırılmış olması, temel bileşenler analizinden farklı olduğu noktalardır.

Faktör analizinin ikinci aşaması kavramsal anlamlılığı sağlamak için elde edilen faktörlerin döndürülmesi gerekir. Faktör döndürmesi, elde edilen faktörleri

daha iyi yorum verebilecek biçimde (kavramsal anlamlılık) yeni faktörlere çevirme olarak ifade edilebilir.

2.3. Ayırma Analizi

Araştırmalarda, p değişkenli bireylerin değişkenlerine göre sınıflandırılması istendiği durumda kullanılan bir yöntemdir. Ayırma analizinin amacı hatalı sınıflandırma olasılığını en aza indirgeyerek bireyleri ait oldukları gruplara ayırmak, çekilmiş oldukları kitleleri belirlemektir. Ayırma analizi ile kümeleme analizi arasında bazı benzerlikler var gibi görünsede, kümeleme analizinde küme sayısının tam olarak bilinmemesi ve gelecekte kullanılabilirlik özelliği olmaması nedeniyle ayırma analizinden farklılıklar göstermektedir. Ayırma analizinin temel amacı, incelenen bireyin kitlesinin belirlenmesini sağlayacak bir fonksiyonun bulunmasıdır. Bu fonksiyonun bulunmasında, belirlenecek grupların ortalamaları arasındaki farklılığın maksimum olması amaçlanmaktadır.

Ayırma analizinin amacı çok değişkenli problemin tek değişkenli biçime dönüştürülmesidir. Yani tüm değişkenlerin uygun ağırlıklarla katılacağı bir fonksiyonun elde edilmesidir.

Bir bireye ait p değişkenden bulunacak olan bağıntı,

$$y_i = a_1x_{i1} + a_2x_{i2} + \ldots + a_px_{ip} \qquad (2.16)$$

biçiminde gösterilebilir. Burada $x_1, x_2, \ldots, x_p$ orijinal değişkenleri $a_1, a_2, \ldots, a_p$ ise bu değişkenlere ilişkin ağırlıkları göstermektedir. Bu fonksiyona ayırma fonksiyonu denmektedir. Böyle bir fonksiyon bulunurken, gruplar arası varyansın grup içi varyansa göre en büyüklenmesi gerekir. Yani,

$$F = Max\ (\frac{GruplarArasıVaryans}{GrupİçiVaryans}) \qquad (2.17)$$

oranının en büyük olması istenir. Yukarıda anlatılan teknikler iki küme olması durumunda bireylerin hangi kümeden geldiğini belirlemek için kullanılan tekniklerdir. İkiden çok küme olması durumunda kullanılan ayırma analizi teknikleri, iki grup için geliştirilenin genelleştirilmiş biçimleridir. Bu durumda p değişkenli, ikiden çok sonlu sayıda kitle bulunmaktadır. Bireyler, kitleler arasında ayırma gücü en büyük olacak biçimde belli sayıda doğrusal bağıntılar yardımıyla

sınıflandırılmaktadır. Bireylere ait p değişkeni p boyutlu bir uzayda tanımlayacak öyle eksenler bulunmaya çalışılır ki veri toplulukları bu eksenler boyunca birbirlerinden olabildiğince ayrılabilsin. İkiden çok grup olması durumunda bulunacak ayırıcı fonksiyon sayısı; k grup sayısını, p değişken sayısını göstermek üzere $min(k-1,p)$ tane olacaktır.

Gerçekte ayırma analizi ile bireyler tam ölçüm uzayından uygun olan bir alt uzaya indirgenir. Bu nedenle ayırma analizi, asıl amacı dışında boyut indirgeme tekniği biçiminde de düşünülebilir.

2.4. Kümeleme Analizi

Kümeleme analizi (KA), X veri matrisinde yer alan ve doğal grupları kesin olarak bilinmeyen birimleri birbiri ile benzer olan alt kümelere ayırmaya yardımcı olan yöntemler topluluğudur (Özdamar, 2002).

Kümeleme analizinin uygulama aşamaları aşağıdaki gibi verilebilir.

1) Doğal grupları hakkında kesin bilgiler bulunmayan populasyonlardan alınan n sayıda bireyin p sayıda değişkene ilişkin gözlemlerinin elde edilmesi, kısaca veri matrisinin belirlenmesi.
2) Birimlerin birbirleri ile olan benzerliklerini ya da farklılıklarını gösteren uygun bir benzerlik ölçüsü ile birimlerin birbirlerine uzaklıklarını hesaplanması. Benzerlik ya da farklılık matrisinin belirlenmesi.
3) Uygun kümeleme yöntemi yardımı ile benzerlik veya farklılık matrislerine göre birimlerin uygun sayıda kümelere ayrılması.
4) Elde edilen kümelerin yorumlanması ve bu kümeleme yapısına dayalı olarak kurulan hipotezlerin doğrulanması için gerekli analitik yöntemlerin uygulanması.

Doğal gruplamaları açıkça bilinen toplumlarda alt kümelerin incelenmesi Diskriminant (Ayırma) Analizi ile yapılır. Alt populasyon tanımlamaları açıkça yapılamamış ya da ayrı ayrı populasyonlar oldukları kesin bilinmeyen karma toplumları birbirinden ayırmak, yeni tanımlamalar yapmak, birimler için yeni prototipler belirlemek, populasyon ya da alt populasyon profilleri tanımlamak, biyolojik materyaller için taksonomik sınıflandırma profilleri belirlemek amacıyla kümeleme analizinden yararlanılmaktadır (Özdamar, 2002).

Kümeleme analizinin genel amacı, gruplanmamış verileri benzerliklerine göre sınıflandırmak ve araştırmacıya uygun, işe yarar özetleyici bilgiler sunmaya yardımcı olmaktır. Bireylerin gruplandırılmasında kullanılması nedeniyle kümeleme ve ayırma analizleri arasında benzerlik olmakla birlikte, bu iki yöntem arasında önemli farklılıklar da bulunmaktadır. Her şeyden önce ayırma analizinde küme sayısı, bilinmekte, bu sayı analiz süresince değişmemekte ve araştırmacıdan, bireyleri bu kümelere ataması istenmektedir. Ayrıca ayırma analizinde elde edilen ayırma fonksiyonu gelecekte kullanılabilmektedir (Tatlıdil, 2002). Oysa ki kümeleme analizinde küme sayısı bilinmemekte küme sayısı kümeleme analizi aracılığıyla belirlenmeye çalışılmaktadır. Kümeleme analizinin verilerin mevcut durumuna ilişkin sonuçlar vermesi nedeniyle gelecekte kullanılabilmesi söz konusu olmamaktadır.

Kümeleme analizinin uygulanması sırasında izlenen adımlar şunlardır.

- Problemin formüle edilmesi
- Uzaklık ölçüsünün belirlenmesi
- Kümeleme prosedürünün belirlenmesi
- Küme sayısına karar verilmesi
- Kümelerin yorumlanması ve profile edilmesi
- Kümelerin geçerliliğinin belirlenmesi

Kümelemede pek çok yöntem bulunmakta ve bu yöntemler farklı başlıklar altında toplanmaktadır. Ancak, Kümeleme Analizi konusunda en çok bilinen ya da en çok kabul gören yöntemler; hiyerarşik (aşamalı) ve hiyerarşik olmayan (aşamalı olmayan) yöntemler biçiminde iki ana başlık altında toplanmaktadır (Tatlıdil, 2002).

3. ALGORİTMİK KÜMELEME METOTLARI

3.1. Kümeleme Kriterleri

Benzer verilerin bir araya getirilip gruplanması veya çok sayıdaki değişkenlerin boyut indirgemesi yoluyla kümelenmesi temeline dayanan kümeleme yöntemlerinde doğru sonuçlar elde edebilmek için doğru kriterleri seçmek gerekir.

3.1.1. Benzerlik ve Uzaklık Ölçütleri

Kümelemenin amacı, benzer bireyleri bir araya getirip gruplamak olduğundan bireylerin ne kadar benzer ya da farklı olduğunu belirleyebilmek için bazı ölçütlere ihtiyaç vardır. En sık kullanılan yöntem bireyler arasındaki mesafe bakımından benzerliğin ölçülmesidir. Birbirleri ile aralarında mesafe az olan bireyler birbirlerine daha benzerken, birbirinden uzak olanlar farklıdır. İki birey arasındaki mesafeyi hesaplamanın bir çok yolu vardır. En fazla kullanılan benzerlik ölçüsü bireyler arasındaki mesafeyi belirleyen "Öklid mesafesi" ya da onun karesidir. Öklid mesafesi her değişken için kare farklarının toplamının kare köküdür.

Farklı uzaklık ölçütleri kullanılarak faklı kümeleme sonuçlarına ulaşılabilir. Bu yüzden farklı ölçümleri kullanmak ve bunların sonuçlarını karşılaştırmak mümkündür.

X veri matrisi, n gözlem (birey) ve p değişkene ilişkin değerleri göstermek üzere değişken vektörleri

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ x_p \end{bmatrix} \tag{3.1}$$

$$\begin{aligned} \boldsymbol{x_1} &= x_{11}, x_{12}, \ldots, x_{1n} \\ \boldsymbol{x_2} &= x_{21}, x_{22}, \ldots, x_{2n} \\ &\vdots \\ \boldsymbol{x_p} &= x_{p1}, x_{p2}, \ldots, x_{pn} \end{aligned} \tag{3.2}$$

şeklinde ifade edilebilir.

x_1 ve x_2 değişken vektörleri arasındaki açı, gözlemler uzayında bu iki vektörün iç çarpımıdır (Tatlıdil, 2002).

$$< x_1,x_2> = x_1\, x'_2 = \sum_{i=1}^{n} x_{1i}x_{2i} = |x_1|\,|x_2|\,cos\alpha \qquad (3.3)$$

Buradan da,

$$cos\alpha = \frac{x_1 x'_2}{|x_1|\,|x_2|} = \sum_{i=1}^{n} x_{1i}x_{2i} / (\sum_{i=1}^{n} x_{1i}^2 \sum_{i=1}^{n} x_{2i}^2)^{1/2} \qquad (3.4)$$

yazılabilir. Veri matrisi pozitif tanımlı iken, 0 ile 1 arasında değerler alabilen vektörler arası açı ölçüm uzaklıklarını etkilemez. Bu katsayı,

$$\rho = \frac{Kov(x_1,x_2)}{[Var(x_1)Var(x_2)]^{1/2}} \qquad (3.5)$$

ρ, değişkenler arasındaki ilişkiyi ölçen en basit değer olan çarpım moment korelasyon katsayısıdır.

x_i ve x_j gözlem vektörleri arasındaki $d(x_i,x_j)=d_{ij}$ uzaklık değerlerini ifade etmek amacıyla geliştirilmiş pek çok teknik bulunmaktadır. En çok bilinen uzaklık ölçütleri şunlardır:

Mincowski Uzaklığı:

$$d_\lambda (x_i,x_j) = [\sum_{k=1}^{p} |x_{ik}-x_{jk}|^\lambda]^{1/\lambda} \; ; \lambda \geq 1 \text{ için} \qquad (3.6)$$

Manhattan City-Block Uzaklığı ($\lambda = 1$ durumu):

$$d_1(x_i,x_j) = \sum_{k=1}^{p} |x_{ik}-x_{jk}| \qquad (3.7)$$

Öklit (Euclidean) Uzaklığı ($\lambda = 2$ durumu):

$$d_2 (x_i,x_j) = [\sum_{k=1}^{p} |x_{ik}-x_{jk}|^2]^{1/2} \qquad (3.8)$$

Ölçekli Öklit Uzaklığı:

Değişkenlerin aynı ağırlıkta ölçeklenmemiş olması durumunda kullanılır.

$$d_2 (x_i,x_j) = \{\sum_{k=1}^{p} w_k^2 |x_{ik}-x_{jk}|^2\}^{1/2} \qquad (3.9)$$

Buradaki w_k, k'ıncı değişkenin standart sapma değerinin (s_k) ya da dağılım aralığı değerinin tersidir. w_k'nın s_k değerinin tersi olması durumunda elde edilen uzaklığa Karl-Pearson uzaklığı da denilmektedir (Tatlıdil, 2002).

Mahalanobis Uzaklığı :

$$d(x_i,x_j) = D^2 = (x_i-x_j)' S^{-1}(x_i-x_j) \tag{3.10}$$

ya da

$$d(\bar{x}_i, \bar{x}_j) = D^2 = (\bar{x}_i, \bar{x}_j)' S^{-1}(\bar{x}_i, \bar{x}_j) \tag{3.11}$$

Hotelling T^2 Uzaklığı:

Hotelling T^2 değeri iki grup ya da kümenin ortalama vektörlerinin karşılaştırılmasında kullanılan bir uzaklık ölçütüdür.

$$T^2 = \frac{n_1 n_2}{n}(\bar{x}_i, \bar{x}_j)' S^{-1}(\bar{x}_i, \bar{x}_j) \tag{3.12}$$

Canberra Uzaklığı:

$$d(x_i,x_j) = \sum_{k=1}^{p} |x_{ik}-x_{jk}| / \sum_{k=1}^{p} |x_{ik}+x_{jk}| \tag{3.13}$$

Değişkenlerin tümünün sürekli olmadığı durumlarda yukarıda verilen formüller doğrudan kullanılamamakta onların yerine uzaklıklar;

sürekli veriler için $w_k = 1$, (3.14)

kesikli veriler için $w_k = \dfrac{1}{\text{k'ıncı değişkenin dağılım aralığı}}$

olmak üzere,

$$d(x_i,x_j) = \frac{1}{p}\sum_{k=1}^{p} w_k |x_{ik}-x_{jk}| \tag{3.15}$$

formülünden bulunmaktadır. Sürekli veriler için geliştirilmiş formüllerin tümüne w_k değeri de eklenerek uzaklıklar yaklaşık olarak hesaplanabileceği gibi, Bhattacharyya uzaklığı başta olmak üzere özel amaçlı bazı metriklerden de yararlanılabilmektedir

Yukarıda verilen $d(x_i,x_j)$ uzaklık değerlerinin bazı özellikleri taşımaları gerekir. Bunlar kısaca şöyle sıralanabilir.

i. $d(x_i,x_j) = d(x_j,x_i)$ simetri özelliği

ii. $d(x_i,x_j) \geq 0$ negatif olmama özelliği

iii. $d(x_i,x_j) = 0$; $i=j$ iken tanım özelliği

iv. $d(x_i,x_j) \leq d(x_i,x_1) + d(x_1,x_j)$ üçgen eşitsizliği özelliği.

Bu özellikleri taşıyan uzaklık değerlerinden oluşturulan D uzaklıklar matrisi nxn boyutludur ve $n(n-1)/2$ tane uzaklık değerinden oluşmaktadır.

$$D_{nxn} = \begin{bmatrix} 0 & d_{12} & d_{13} & ..d_{1n} \\ & 0 & d_{23} & ..d_{2n} \\ & ..... & ..... & ..... \\ Sim & & & d_{n-1,n} \end{bmatrix} \qquad (3.16)$$

3.1.2. Değişkenlerin Standardizasyonu ve Dönüştürülmesi

Veri matrisinde değişkenlerin aşırı uçlarda yer alan değerleri kümeleme üzerinde olumsuz etkilerde bulunmaktadır. Bu gibi durumlarda verilerin standardize edilmesi ya da belirli aralıklarda gözlenen değerlere dönüştürülmesi uygun olmaktadır (Özdamar, 2002).

Verilerin standardize edilmesi ya da belirli aralıklara dönüştürülmesi için bir çok yöntem bulunmaktadır. Bu yöntemler; z skorlarına dönüştürme, $-1 \leq x \leq +1$ aralığına dönüştürme, $0 \leq x \leq +1$ aralığına dönüştürme, maksimum değer 1 olacak şekilde dönüştürme, ortalama 1 olacak şekilde dönüştürme, standart sapma 1 olacak şekilde dönüştürme ve birim vektöre dönüştürme gibi yöntemlerdir. Bu yöntemlere aşağıda kısaca değinilmiştir.

Z Skorlara Dönüştürme

Z skorlara dönüştürme, oransal ya da aralıklı ölçekle elde edilen ve normal dağılım gösterdiği varsayılan verilere uygulanan ve en çok tercih edilen dönüştürme yöntemidir (Özdamar, 2002). Değerler

$$z_i = \frac{X_i - \bar{x}}{S} \qquad (3.17)$$

biçiminde z skorlarına dönüştürülür.

$-1 \leq x \leq +1$ Aralığına Dönüştürme

Bu yöntem, heterojen yapıda değerlerin ve aşırı uçlarda değerlerin yer aldığı durumlarda tercih edilen bir dönüştürme yöntemidir (Özdamar, 2002).

Değerler arasında artı ve eksi değerlerin bulunması halinde uygulanan bir yöntemdir. X_{max} dizideki en büyük değer olmak üzere dönüştürme;

$$x_i = \frac{X_i}{X_{\max}} \tag{3.18}$$

biçiminde yapılır.

$0 \leq x \leq +1$ aralığına dönüştürme

$0 \leq x \leq +1$ aralığına dönüştürme, heterojen yapıda değerlerin ve aşırı uçlarda değerlerin yer aldığı durumlarda değerleri pozitif ve *0-1* aralığında değişecek biçime dönüştürmek için tercih edilen bir dönüştürme yöntemidir (Özdamar, 2002).

Dizideki en büyük değer X_{max} ve en küçük değer X_{min} olmak ve değişim genişliği (range) $R = X_{max} - X_{min}$ olarak hesaplanmak üzere dönüştürme;

$$x_i = \frac{X_i - X_{\min}}{R} \tag{3.19}$$

biçiminde yapılır.

Maksimum Değer 1 Olacak Şekilde Dönüştürme,

Maksimum değer 1 olacak şekilde dönüştürme, dizideki değerlerin maksimum değeri 1 olacak biçime dönüştürülmesi isteniyor ise uygulanan bir yöntemdir (Özdamar, 2002). Dönüşüm;

$$xi = \frac{X_i}{X_{\max}} \tag{3.20}$$

biçiminde yapılır.

Eğer dizideki maksimum sıfır ise dönüştürme;

$$x_i = \frac{X_i}{X_{\max}} + 1 \tag{3.21}$$

biçiminde yapılır.

Dönüşüm Dizisinin Ortalaması 1 Olacak Şekilde Dönüştürme

Dönüşüm dizisinin ortalaması 1 olacak şekilde dönüştürme, yeni dönüştürülmüş dizinin ortalamasının pozitif ve 1 olması istendiğinde uygulanan bir yöntemdir (Özdamar, 2002). Dönüştürme;

$$x_i = \frac{X_i}{\bar{x}} \tag{3.22}$$

biçiminde yapılır.

Eğer dizinin ortalaması sıfır ise dönüştürme;

$$x_i = \frac{X_i + 1}{\bar{x} + 1} \tag{3.23}$$

biçiminde yapılır.

Dönüşüm Dizisinin Standart Sapması 1 Olacak Şekilde Dönüştürme

Dönüşüm dizisinin standart sapması 1 olacak şekilde dönüştürme, yeni dönüştürülmüş dizinin standart sapması 1 olması istendiğinde uygulanan bir yöntemdir. Dönüştürme;

$$x_i = \frac{X_i}{S} \tag{3.24}$$

biçiminde yapılır.

Eğer dizinin değerleri (–) ve (+) değerlerden oluşuyor ve standart sapması sıfır ise verilerde dönüştürme yapılmaz. Eğer mutlaka dönüştürme yapılması gerekiyor ise yukarıda açıklanan yöntemlerden uygun birisi ile dönüştürme yapılır (Özdamar, 2002).

Birim Vektöre Dönüştürerek Standartlaştırma

Çok değişkenli verilerin standardizasyonun da birim vektöre dönüştürerek standartlaştırma yapılabilir. Birim vektör, bir birim uzunluğundaki vektördür ve

$$\sqrt{\sum_{i=1}^{p} \tilde{x}_j^{\,2}} = 1 \tag{3.25}$$

biçiminde tanımlanır. Bir vektörün bileşenlerinin, vektör uzunluğuna bölünmesi ile gerçekleştirilen dönüşümdür ve birim vektöre dönüştürme;

$$\tilde{x}_j^i = \frac{x_j^i}{\sqrt{\sum_{j=1}^{p} x_j^2}} \qquad (3.26)$$

biçiminde tanımlanır.

3.1.3. Küme Sayısının Belirlenmesi

Kümeleme analizinde sağlıklı ve anlamlı sonuçlara ulaşabilmek için iki koşulun sağlanması gerekir. Bu koşullardan ilki önemli değişkenlerin seçilmesi ikincisi ise küme sayısının belirlenmesidir.

Küme sayısının belirlenmesi konusunda son yıllarda yoğun çalışmalar yapılıyor olmakla birlikte hala 1970'li yıllarda geliştirilmiş olan bazı testlerden yararlanılmaktadır. Küme sayısına karar vermede yararlanılan en pratik yol, küme sayısı k olmak üzere,

$$k \cong (n/2)^{1/2} \qquad (3.27)$$

biçiminde hesaplanmaktadır. Küçük örneklemler için kullanılabilir gözüken bu formül örneklem hacminin büyük olması durumunda iyi sonuçlar vermemektedir (Tatlıdil, 2002).

Marriot tarafından 1971 yılında önerilen ikinci yöntem ise W, grup içi kareler toplamı matris olmak üzere küme sayısı,

$$M = k^2 \mid W \mid \qquad (3.28)$$

eşitliğinden bulunmakta ve en küçük M değerini veren küme sayısı gerçek küme sayısı olarak değerlendirilmektedir (Tatlıdil, 2002).

Bir başka yöntemde Calinsky ve Harabasz tarafından yine 1970'li yıllarda geliştirilmiştir. Bu yöntemde ise B ve W sırasıyla gruplar arası ve grup içi kareler toplamı matrisi olmak üzere,

$$C = [İz(B)/(k-1)] \,/\, [İz(W)/(n-k)] \qquad (3.29)$$

eşitliğini en büyük yapan k değeri küme sayısı olarak alınmaktadır (Tatlıdil, 2002).

Sonuç olarak, küme sayısını belirlemede bazı yöntemlerden yararlanılsa bile, bu konuda araştırmacının bilgi düzeyi, mesleki tecrübesi ve sonuçların anlamlı olup olmaması en önemli etkendir (Tatlıdil, 2002).

3.2. Aşamalı Kümeleme Yöntemleri

Aşamalı kümeleme yöntemleri, birimlerin benzerliklerini dikkate alarak birimleri belirli düzeylerde birbirleri ile birleştirmeyi veya ayırmayı amaçlayan yöntemlerdir. Bu yöntemler, birimleri küme uzaklık ölçüleri yardımıyla değişik aşamalarda bir araya getirerek ardışık biçimde kümelere ayırırlar ve kümelere girecek elemanların, hangi farklılık ya da benzerlik düzeyinde küme elemanı olduğunu belirlemeyi amaçlarlar..

Veri matrisindeki birimlerin, başlangıç aşamasında kaç küme oluşturduğuna ve küme elemanlarını belirlemede başlangıçta hangi kriterin seçildiğine göre aşamalı yöntemler iki ana gruba ayrılır.

3.2.1. Birleştirici (Agglomerative) Aşamalı Kümeleme Yöntemleri

Başlangıçta tüm birimlerin ayrı küme oluşturduğunu kabul ederek, *n* birimi aşamalı olarak sırasıyla *n, n-1, ..., n-r, ..., 3, 2, 1* kümeye yerleştirmeyi amaçlayan bir yaklaşımdır.

Birleştirici aşamalı kümeleme yönteminde, kümeleme sürecinin başlangıcında her birey bir kümedir, süreç sonunda ise tüm bireyler tek bir kümede toplanır (Tatlıdil, 2002). İşleyiş daha ayrıntılı bir şekilde aşağıdaki gibi bir algoritma ile ifade edilebilir.

1. *n* tane birey, *n* tane küme olmak üzere işleme başlanır.

2. En yakın (d_{ij} değeri en küçük olan) iki küme birleştirilir.

3. Küme sayısı bir azaltılarak yinelenmiş uzaklıklar matrisi bulunur.

4. *2* ve *3* nolu adımlar *n-1* kez tekrarlanır (Tatlıdil, 2002).

Bu süreçte birden çok gözlemli kümenin vektör olarak gösterilebilmesi amacıyla değişkenlerin ortalama değerlerinden yeni vektör oluşturulmakta ya da bu kümedeki tüm gözlemler ile başka kümedeki gözlemlerin uzaklık ortalamaları kullanılabilmektedir.

Yukarıda verilen algoritmaya dayalı yedi farklı aşamalı teknikten söz edilmektedir. Bu teknikler sırasıyla tek bağlantılı, tam bağlantılı, grup ortalama, merkezi, ortanca, minimum varyans ve Ward teknikleridir.

Birleştirici aşamalı kümeleme yöntemlerinde, birimlerin birbirleri ile birleştirilmesinde değişik yaklaşımlar uygulanmaktadır. Bu nedenle değişik isimlerle anılan birleştirici aşamalı kümeleme yöntemleri bulunmaktadır. Bu yöntemlerden sıklıkla kullanılan ve genel kabul görmüş olanları aşağıdaki gibi sayılabilir (Özdamar, 2002).

1) Tek Bağlantı Kümeleme Yöntemi

En Yakın Komşuluk (Nearest Neighbor-Single Linkage-SLINK) olarak da bilinen bu teknikte uzaklıklar matrisi kullanılarak birbirine en yakın birey ya da kümeler birleştirilmekte ve birleştirme ardı ardına tekrarlanarak sürdürülmektedir (Tatlıdil, 2002).

Johnson tarafından önerilen bu teknikte eğer *i* ve *j*'inci küme birleştirilmiş ise birleştirilen kümenin *k*'ıncı küme ile ilişkisi uzaklık ölçütü,

$$d_{k(i,j)} = Min(d_{ki}, d_{kj}) \qquad (3.30)$$

biçiminde ifade edilmektedir.

Bu yöntem aşamalı kümeleme yöntemleri içinde birimlerin oluşturdukları aşamalı kümeleri belirlemede kullanılan en basit yöntemlerden birisidir.

TekBKY ile aşamalı kümeleme yapmak için aşağıdaki işlem sırası izlenir.

1) *X* veri matrisinin *D* öklit uzaklık matrisi hesaplanır.

2) Eğer istenirse *D* matrisinden *Sim* (benzerlik) matrisi hesaplanır.

3) *D* ya da *Sim* matrisinde en küçük değerli birimler aşamalı olarak birbirileri ile birleştirilir. Küme *(ij)* oluşturulur. *i* ve *j* kümeleri birbirleri ile birleştirildikten sonra *D* (ya da *Sim*) matrisinde *j.* kümeye ilişkin satır silinir ve *ij* kümesinin uzaklığı (ya da benzerliği) *i.* kümenin uzaklığı olarak kalır. $(d_i < d_j)$

4) Tüm kümeler birbirleri ile birleştirilinceye kadar *3.* satırdaki işlemler tekrarlanır (Özdamar, 2002).

2) Ortalama Bağlantı Kümeleme Yöntemi (OrtBKY)

OrtBK yönteminde bir birimin m. küme olarak hangi birim ya da kümelerle birleştirileceği, birimlerin yeni oluşan kümelerle olan uzaklıkları dikkate alınarak belirlenir. *m.* kümenin daha önce oluşan *k.* ve *l.* kümelerden hangisi ile birleşerek oluşacağını belirlemek için *j.* küme ile *k.* ve *l.* kümelerinin uzaklıklarına bakılır. Bu uzaklıklar *k* ve *l* kümelerinin eleman sayısı ile çarpılarak ağırlıklandırılır. Elde edilen toplam yeni oluşacak m. küme eleman sayısına bölünür (Özdamar, 2002).

m. kümenin *j.* küme ile olan uzaklığı (d_{mj});

$$d_{mj} = (N_k d_{kj} + N_l d_{lj}) / N_m \qquad (3.31)$$

şeklinde belirlenir.

Birbirine en çok benzeyen çiftin bulunmasında grup içi benzerlik ortalaması *k* ve *l* kümelerine ait çiftlerin benzerlik ölçülerinden ve birim sayılarından yararlanılarak hesaplanır (Özdamar, 2002).

3) Tam Bağlantı Tekniği:

En Uzak Komşuluk (Furthest Neighbor Complete Linkage-CLINK) olarak da bilinen bu teknik yine Johnson tarafından önerilmiştir. Tek bağlantı tekniğine çok benzeyen bu teknikte tek farklılık iki küme arasındaki uzaklık olarak her kümedeki eleman çiftleri arasındaki uzaklığın en büyüğünün ele alınmasıdır. Bir birimin *m.* küme olarak hangi birim ya da kümelerle birleştirileceği, birimlerin yeni oluşan kümelerle uzaklıkları dikkate alınarak belirlenir. *m.* kümenin daha önce oluşan *k.* ve *l.* kümelerin uzaklıklarına bakılır. Bu uzaklıklardan en büyük olanı ile (*k* ya da *l*) birleştirme yapılarak *m.* küme belirlenir. *m.* kümenin *j.* küme ile olan uzaklığı (d_{mj});

$$d_{mj} = Max(d_{kj}, d_{lj}) \qquad (3.32)$$

biçiminde gösterebiliriz.

Hala kullanılmakta olan bilgisayar algoritmalarının büyük çoğunluğu tek ve tam bağlantı tekniklerini kullanmakla birlikte, tek bağlantı tekniği sağlıklı sonuçlar vermesi açısından tercih edilir ancak işlemlerin uzun sürmesi açısından sakıncalıdır. Tam bağlantı tekniği ise aynı küme içersindeki bireylerin uzaklıklarının belli bir değerden küçük olması durumunda tüm kümelerin sağlıklı oluşturulmasını garanti

etmemektedir. Son yıllarda sıkça kullanılmaya başlayan ortalama bağlantı tekniği, bu iki uç teknik arasında sonuçlar vermesi nedeniyle bir alternatif olarak önerilmektedir. Ayrıca merkezi, ortanca ve minimum varyans teknikleri de bu üç tekniğin benzerleri olup uygulamada pek sık kullanılmamaktadır (Tatlıdil, 2002).

4) McQuitty Bağlantı Kümeleme Yöntemi

m. kümenin oluşumu, *k.* ve *l.* kümelerin *j.* küme ile olan uzaklıkları ortalaması dikkate alınarak belirlenir. Ağırlıksız ortalama bağlantı yöntemi olarak da bilinmektedir. Yeni oluşan *m* ve *j* kümeleri arasındaki uzaklık;

$$d_{mj} = (d_{kj} + d_{lj}) / 2 \qquad (3.33)$$

biçiminde belirlenir (Özdamar, 2002).

5) Merkezsel Bağlantı Kümeleme Yöntemi

OrtBK yönteminin özel bir biçimidir. *m* kümesinin *j* kümesine olan uzaklığı;

$$D_{mj} = (N_k d_{kj} + N_l d_{lj}) / N_m - N_k N_l d_{kl} / N_m^2 \qquad (3.34)$$

biçiminde belirlenir (Özdamar, 2002).

6) Ortanca Bağlantı Kümeleme Yöntemi

McQuitty bağlantı kümeleme yönteminin özel bir biçimidir. *m* ve *j* kümeleri arasındaki uzaklık;

$$d_{mj} = (d_{kj} + d_{lj}) / 2 - d_{kl} / 4 \qquad (3.35)$$

biçiminde belirlenir (Özdamar, 2002).

7) Ward Bağlantı Kümeleme Yöntemi

Ward bağlantı yöntemi centroid ve medyan bağlantı kümeleme yöntemlerinin karma ve ağırlıklı biçimidir. *m* ve *j* kümeleri arasındaki uzaklık;

$$d_{mj} = ((N_j + N_k)d_{kj} + (N_j + N_l)d_{lj} - N_j d_{kl}) / (N_j + N_m) \qquad (3.36)$$

biçiminde belirlenir (Özdamar, 2002).

3.2.2. Ayırıcı (Divisive) Aşamalı Kümeleme Yöntemleri

Başlangıçta tüm birimlerin bir küme oluşturduğunu kabul ederek birimleri aşamalı olarak *n* birimi sırasıyla *1, 2, 3, ..., n-r, ..., n-2, n-1, n* kümeye ayırmayı

amaçlayan bir yaklaşımdır. Birleştirici aşamalı kümeleme yönteminin tersidir. Birleştirici yönteme ilişkin sonuçlardan ayırıcı yönteme ilişkin sonuçlarda elde edilebilir.

3.3. Aşamalı Olmayan Kümeleme Yöntemleri

Aşamalı kümelemede hem birimler hem de değişkenler birbirleriyle değişik benzerlik düzeylerinde kümeler oluştururken, aşamalı olmayan yöntemlerde birimlerin uygun oldukları kümelerde toplanmaları ve n birimin k sayıda kümeye parçalanması hedeflenmektedir.

Aşamalı olmayan kümeleme yöntemlerinde n birimin k kümeye parçalanması rastgele yapılabilir. Birimlerin ayrılabilecekleri küme sayısı belirlendikten sonra, küme belirleme kriterlerine göre birimlerin hangi kümelere girebileceklerine karar verilir ve atama işlemleri yapılır (Özdamar, 2002).

Aşamalı olmayan kümeleme yöntemleri, aşamalı kümeleme yöntemlerinde olduğu gibi birimlerin k kümeye parçalanmasında, kümelere atanmasında ve küme merkezlerinin belirlenmesindeki farklılıklardan dolayı farklı isimlerle anılmaktadır. Yöntemlerden bazıları küme sayısını tamamen deneme yoluyla belirlemeyi ve $k = 2, 3, 4, ..., h; h<n$ olacak şekilde bir ardışık parçalamayı önerirken, bazı yöntemler ise çekirdek noktalar (seed points) seçilerek bu noktaları, oluşacak kümenin merkezleri kabul ederek bu çekirdekler etrafında oluşacak kümelere birimlerin atanmasını önermektedir.

Aşamalı olmayan istatistiksel yöntemler *Sim* benzerlik matrisi ya da D uzaklık matrisi yerine X veri matrisini ya da dönüştürülmüş Z matrisini kullanmaktadır (Özdamar, 2002).

Çekirdek noktaların seçiminde kümeleme çözümlemesinden yararlanacak araştırıcının, verileri hakkında ayrıntılı bilgi sahibi olması önem taşımaktadır. Çekirdek noktaların belirlenmesinde bir çok yaklaşım önerilmiştir. Bu yaklaşımlar kısaca şöyle açıklanabilir.

1. Veri setindeki ilk k birim çekirdek nokta (centroid) olarak seçilir.

2. Veri setindeki birimler 1'den *n*'e kadar sıralanır ve sistematik olarak *a/k, 2a/k, ..., (k-2)a/k, (k-1)a/k* numaralı birimler çekirdek nokta olarak seçilir. Burada *a=n / k* olarak alınan rastgele sayıdır.
3. Veri setinde *k* birim gelişigüzel seçilerek çekirdek nokta olarak belirlenir.
4. 1 ile *n* arasında *k* tane rastgele sayı seçilir ve bu sayılara karşılık gelen sıra numaralarına sahip birimler çekirdek nokta olarak alınır.
5. Her bir değişken için dağılım aralığına göre koordinatlar vektörü olarak *k* nokta türetilir. Türetilen bu noktalar çekirdek nokta olarak kabul edilir. Bu noktalar veri setindeki birimlere karşılık gelen noktalardan farklı olabilirler.
6. Veri setindeki birimleri aynı anda olmamak koşuluyla *k* kümeye parçalayacak bir parçalama kalıbı seçilir ve her kümenin küresel ortalamaları çekirdek noktalar olarak alınır (Özdamar, 2002).

Küme sayısı konusunda ön bilgi var ise ya da araştırmacı anlamlı olacak küme sayısına karar vermiş ise bu durumda, çok uzun zaman alan hiyerarşik yöntemler yerine hiyerarşik olmayan (nonhierarchical) ya da aşamalı olmayan kümeleme yöntemleri tercih edilmektedir. Ayrıca bu yöntemlerin kuramsal dayanaklarının daha güçlü olması diğer bir tercih nedenidir. Hiyerarşik olmayan kümeleme başlığı altında pek çok teknikten söz edilmektedir. Ancak bunlardan en çok kullanılan yöntemler MacQueen tarafından geliştirilmiş olan k-ortalama tekniği, en çok olabilirlik tekniği ve Yığın Kümeleme Yöntemi (Hill Climbing Method)'dur. Bu yöntemler *n* sayıda birim *k* kümeye ayırırken farklı kriterleri kullanmaktadır.

3.3.1. k-Ortalamalar Yöntemi

Bu yöntem *n* birimin *k* kümeye ayrılmasında, birimin *p* boyutlu uzayda kendisine en yakın çekirdek noktalı kümeye atanmasını içerir. Bu yöntemde ortalamalar, başlangıçta ele alınan *k* noktanın yani çekirdek noktaların değerleridir. k-Ortalamalar yöntemi aşağıdaki adımları izleyerek birimleri kümelere ayırır.

a) Araştırıcının verilerden elde edeceği bilgilere göre ilk *k* birim çekirdek nokta olarak alınır. Bu noktaların her birinin *p* değişken değerleri birer küme ortalama vektörü olarak kabul edilir. Küme ortalama vektöründen her bir birimin uzaklıkları hesaplanır.

b) Geriye kalan *n-k* birim en yakın ortalama vektörlü kümeye atanır. Her atamadan sonra oluşan kümenin ortalama vektörü yeniden hesaplanır. Böylece, çekirdek noktaların verilerinden oluşan ortalama vektör değiştirilir ve birimlerin yeni oluşan küme ortalama vektörüne göre uzaklıkları hesaplanır. En büyük benzerliğe sahip birimler bir araya getirilir.

c) Küme içi varyansın minimum ve kümeler arası varyansın maksimum olduğu kümeleme yapısına ulaşıncaya kadar tüm birimler *k* kümeye atanmaya devam eder. Yinelemeli yaklaşımla uygun kümeleme sağlanır. Her birimin bu küme ortalama vektörlerine değişik aşamalarda değişik kümelerde yer alması sağlanır. Her aşamada birimlerin kümelerde yer alması olasılığı 0 ile 1 arasında değişir. Küme içi kovaryans matrisinin minimum olduğu koşul sağlanıncaya ve yakınsama kriterine eşit oluncaya ya da daha küçük varyans farkına ulaşıncaya kadar parçalanma işlemine devam edilir (Özdamar, 2002).

k-Ortalamalar yöntemi, *k=2*'den başlayarak küme sayılarını her defasında birer arttırarak deneysel olarak en uygun kümelemeyi bulmak şeklinde de uygulanabilir. Bu durumda araştırıcıya verilerini çok iyi tanıması zorunluluğu getirilmektedir. Böyle bir yaklaşımda toplam küme içi varyans matrisinin izi (trace) *tr(W)* minimize edilir.

Uygun küme sayısının belirlenmesinde oluşan küme içi varyans matrisi ve kümeler arası kovaryans matrislerinden yararlanılarak Wilk's Lamda değerleri hesaplanır. $k = 2, 3, 4, \ldots$ için hesaplanan Wilk's Lamda değerleri içinden en yüksek önemliliğe sahip olan k kümeye parçalanma çözümü, uygun kümeleme olarak alınır. Ayrıca her kümeleme sonucunda hesaplanan Mahalonobis uzaklık matrisinin Hotelling T^2 istatistikleri bulunarak bu istatistiklerin önemlilik düzeylerinin en yüksek olduğu durum uygun kümeleme olarak kabul edilebilir. Bilindiği gibi Mahalonobis D^2 uzaklık matrisi grupların birbirlerinden olan ayrımını değerlendirmeye yarayan bir uzaklıktır. Bu matrisin kümelere ayırmada, yeterince diskriminasyon sağlayıp sağlamadığı Hotelling T^2 testi ile test edilmektedir (Özdamar, 2002).

k-Ortalama tekniğinde bir başka yaklaşıma göre bireyler, küme içi kareler toplamı en küçük olacak biçimde *k* kümeye bölünmektedir. Yani, $x_1, x_2, \ldots, x_n$ her

biri p değişkenli gözlem vektörleri, çok boyutlu X uzayında birer vektör olarak düşünülecek olursa ve aynı uzayda a_{1n}, ... a_{kn} her grup birey için küme merkezleri olarak seçildiğinde

$$W_n = \frac{1}{n}\sum_{i=1}^{n} \min_{1\le j\le k} \left\| x_i - a_{jn} \right\|^2 \qquad (3.37)$$

kuralı gereğince bireyler en yakın kümeye atanarak sınıflandırma işlemi yapılmaktadır. Bu tekniğin bilgisayar algoritmalarındaki pratik işleyişi ise şöyledir (Tatlıdil, 2002).

1) İlk k gözlemin her biri bir gözlemli küme olarak alınmaktadır.

2) Kalan *n-k* gözlemin her biri, ortalamaya en yakın olan kümeye atanmakta ve her atamadan sonra küme ortalamaları yeniden hesaplanmaktadır.

3) Tüm gözlemlerin kümelere atanması bittikten sonra, *n* gözlemin son bulunmuş küme ortalamalarına göre yeniden atamaları yapılmaktadır.

4) Bir önceki kümelemeye göre son elde edilen kümelemede kümeler arası gözlem geçişi durana kadar üçüncü adım tekrarlanmaktadır (Tatlıdil, 2002).

3.3.2. Hacim İlişkisine Dayalı Kümeleme

Gözlemlerin çok boyutlu uzaydaki dağılımlarını gökyüzündeki bulutlara benzetmek mümkün. Bu gözlemler bir hacim oluşturmaktadır. Çok boyutlu uzayda kümeye en yakın gözlemin katılması halinde küme hacminde çok büyük bir artış olması gerekir. Bu düşünceden hareketle son yıllarda geliştirilen bir yöntem, daha sağlıklı sonuçlar vermesi nedeniyle tercih edilmektedir (Tatlıdil, 2002).

İlk olarak yöntemin kuramsal dayanağı olan Wilks'in çok boyutlu istatistiksel yayılım kavramından bahsetmek gerekir.

x_{1i}, x_{2i}, ..., x_{pi}; $i = 1, ..., n$; p boyutlu uzayda; ortalama vektörü $\mu_1, \mu_2, ..., \mu_p$, kovaryans matrisinin elemanları σ_{ij} olan bir kitleden çekilmiş n gözlemli bir örneklem olsun. Bu örneklem, p boyutlu Öklit uzayında n noktadan oluşan bir küme olarak tanımlanır. Eğer x_{1o}, x_{2o}, ..., x_{po} p boyutlu uzayda herhangi bir başlangıç vektörü ise gözlemlerin bu vektörlerden farkları bir matris oluşturacaktır (Tatlıdil, 2002).

$$X - X_o = \begin{bmatrix} (x_{11} - x_{1o}) & \dots & (x_{1p} - x_{po}) \\ (x_{21} - x_{1o}) & \dots & (x_{2p} - x_{po}) \\ \dots & \dots & \dots \\ (x_{n1} - x_{1o}) & \dots & (x_{n1} - x_{po}) \end{bmatrix} \quad (3.38)$$

Bu matrisin istatistiksel yayılım ölçüsü,

$$_pS_{x_o,n} = \det(X'X) = |X'X| \quad (3.39)$$

biçiminde tanımlanmaktadır. Başlangıç noktasının $\bar{x}_1, \bar{x}_2, ..., \bar{x}_p$ biçiminde örneklem ortalama vektörü olarak alınması durumunda,

$$_pS_{\bar{x}_o,n} = |X'X| = |S| = |U_{ij}| \quad ; i,j = 1, \dots, p \quad (3.40)$$

yayılma ölçüsü, içsel yayılma (internal scatter) adını almakta ve minimum yayılma değerini vermektedir (Tatlıdil, 2002).

Bu bilgiler ışığında *n* hacimli örnekleme bir gözlem daha eklenecek olursa, yeni yayılma değeri | $U_{ij\xi}$ | olarak ifade edilmekte ve bu değer kümelemede ölçüt olarak kullanılmaktadır. Eklenecek gözlemin mevcut hacim değerini bir miktar artırması gerekir. Çok artırması durumunda o gözlemin, o kümenin bir elemanı olamayacağı sonucuna ulaşılır.

İçsel yayılma değerlerine bağlı test istatistiği;

$$R_\xi = \frac{|U_{ij}|}{|U_{ij\xi}|} \; ; \xi = 1,2,\dots, t \text{ için} \quad (3.41)$$

olarak tanımlanmıştır. Bu test istatistiği, R_ξ eğer $x \sim N_p\ (0\ ;\ \sum)$ ise *(n-p-1)/2* ve *p/2* parametreleri ile beta dağılmaktadır. Ayrıca eşitlikten de anlaşılacağı gibi, bu teknikte *t* tane gözlemin herhangi bir kümeye katılıp katılamayacağı da tek adımda sınanabilmektedir. Bu durumda test istatistiğinin dağılımı;

Be ((n-p-1)/2; p/2) Be ((n-p-2)/2; p/2) ... Be ((n-p-t)/2; p/2) gibi *t* tane bağımsız beta değişkenin çarpımına eşit olmaktadır (Rao, 1973; Wilks, 1962).

R_ξ değerinin küçük olması yani sıfıra yaklaşması eklenen gözlem ya da gözlemlerin o küçülmeye ait olmadığını, 1'e yaklaşması ise o kümeye katılmaları gerektiğini işaret etmektedir.

3.3.3. Bulanık Kümeleme

Geleneksel kümeleme yöntemlerinde veriler gruplara ayrılır ve her bir gözlem sadece bir kümeye dahil edilir. Bu nedenle, kümeler kesin bir şekilde birbirinden ayrılır. Fuzzy kümeleme de ise her bir birim üyelik değerleri ile tüm kümelere dahil edilebilir (Zadeh, 1965). Bu algoritmanın çıktıları bir kümedir fakat bir sınıflandırma değildir.

Bulanık kümelerde dereceli üyelik tanımı ilk kez 1965 yılında Kaliforniya üniversitesinden Azeri kökenli Prof. Dr. Lütfü A. Zade (Lotfi Zadeh) tarafından yapılmıştır. Klasik kümelemede 0 ve 1 değerleri kullanılırken bulanık kümelemede 0 ile 1 arasında üyelik fonksiyon değerlerinden bahsedilmektedir. Klasik kümede, kümeye kesin olarak ait (1) veya kesinlikle ait değil (0) biçiminde iki grup oluşturulur. Kümeye ait olan elemanlarla olmayanlar arasında kesin bir fark vardır.

Bulanık kümeleme de ise kesin geçişleri elimine edebilmek için evrendeki bütün bireylere üyelik derecesi değerlerini atayarak matematiksel olarak tanımlanır. Böylece bireyler, bulanık küme içersinde üyelik dereceleri tarafından gösterilen daha büyük ve daha küçük değerlere sahip olabilirler. Bu üyelik değerleri [0-1] aralığında gerçel değerler ile ifade edilir.

Klasik kümenin karakteristik fonksiyonu, evrensel kümede her bireye ya 1 ya da 0 değerini atar. Bu ise üye olma veya olmama anlamındadır. Kümelerdeki her nesne bu kümenin elemanıdır. Kümeler büyük harflerle, elemanlar ise küçük harflerle gösterilir. X evrensel kümesinin her elemanı için üye olduğunun ya da olmadığının saptanması, bu kümenin karakteristiği olan özel bir fonksiyonla gerçekleştirilir. Bir A kümesi için bu fonksiyon, her $x \in X$ için $\mu_A(X)$ değerini aşağıdaki şekilde belirler:

$$\mu_A = \{ 1 \text{ yalnızca } x \in A \; ; 0 \text{ yalnızca } x \notin A \} \qquad (3.42)$$

Böylece bu fonksiyon, evrensel kümenin elemanlarını 0 ve 1'den oluşan bir kümeye çerçeveler.

$$\mu_A : X \rightarrow \{ 0,1 \} \tag{3.43}$$

Bu fonksiyon, elemanın küme içindeki üyelik derecesinin veren ve özel bir aralıkta evrensel kümenin elemanlarına değer atayan fonksiyon olarak genelleştirilebilir. Daha büyük değerler, daha yüksek dereceli üyeliğe karşılık gelirken, daha düşük değerler ise üyelik derecesinin küçük olduğunu göstermektedir. Üstte tanımlanan bu fonksiyona *üyelik fonksiyonu,* kümeye ise *bulanık küme* denilir. X bir evrensel küme ise, A bulanık kümesinin üyelik fonksiyonu $\mu_A : X \rightarrow \{ 0,1 \}$ biçiminde tanımlanır (Nabiyev, 2003).

Üyelik fonksiyonlarını oluşturmada birçok yöntem bulunmaktadır. En gelişmiş yöntemler uzman tecrübelerinden faydalanarak küme değerlerini noktalı olarak belirlemek ve analitik fonksiyon biçiminde ifade etmektir.

FCM Algoritması ile sınıflandırma

V^i ve δ^i parametrelerini belirlemek için Bezdek tarafından önerilmiş olan FCM kümeleme algoritmasının kullanımı önerilmiştir. Bu algoritmaya göre, girdi-çıktı kümesi $X \in R^{p+1}$ nin bir parçası $1 \leq m \leq c$ sayıda kümeyle temsil ediliyorsa, U fuzzy bölünme matrisi (uxc) boyutlu veya U_{ik} $u_{ik} \in [0,1]$ veri noktalarının üyelik fonksiyon değerleri, $X_k = (x_{k1}, x_{k2}, \ldots, x_{kp}, x_{(p+1)})$, c küme sayısı. Burada, $x_{i(p+1)}$, k.ıncı çıktıya karşılık gelen veri noktasını göstermektedir.

Girdi uzayını c parçaya bölmek için, uzaklığa dayalı amaç fonksiyonunu optimize etmek gerekir. Böyle bir fonksiyonla:

$$j(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \left\| V^i, X_k \right\|^2 \tag{3.44}$$

$\|.\|$ öklit uzaklığı $1<m$ bulanıklılık parametresi olarak adlandırılır ve uygun şekilde kısıtlanmalıdır.

$$\sum_{k=1}^{n} u_{ik} > 0 \qquad \text{tüm } i \in (1,\ldots,c) \text{ için}$$

$$\sum u_{ik} = 1 \qquad \text{tüm } k \in (1,...,n) \text{ için} \tag{3.45}$$

(3.44)'ün optimizasyonu için alternatif bir optimizasyon tekniği kullanılır. Bu tekniğe göre uygun küme sayısı altında, rastgele olarak başlangıç bölünme matrisi *(U0)* veya küme merkezi $V_0^i \in R^{p+1}$ değerleri her bir döngü için adım *t* (*step t*) aşağıdaki eşitlikler yoluyla güncellenir. Sırasıyla:

$$u_{ik} = \sum_{l=1}^{c} \left(\frac{\left\| V^i, X_k \right\|}{\left\| V^l, X_k \right\|} \right)^{-(2)/(m-1)} \tag{3.46}$$

$$V_i = \frac{\sum_{k=1}^{n} (u_{ik})^m X_k}{\sum_{k=1}^{n} (u_{ik})^m} \tag{3.47}$$

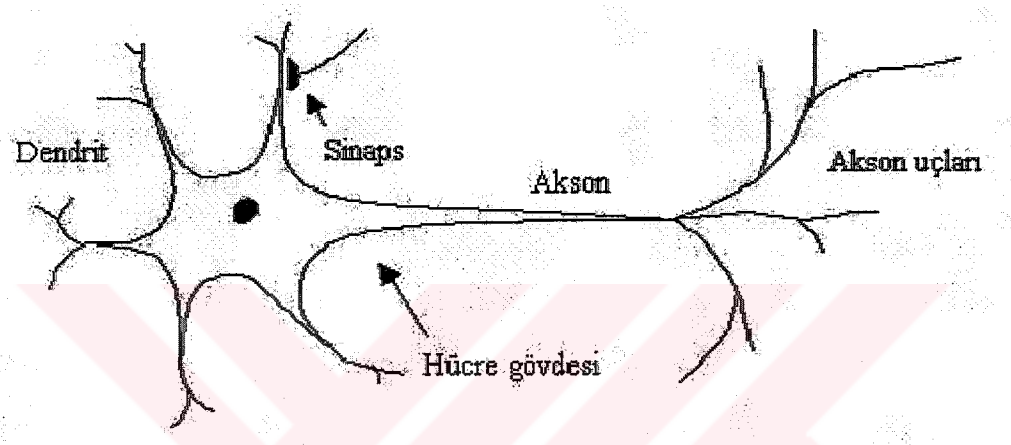
güncelleme işlemi $\varepsilon = \left\| U^t, U^{t-1} \right\|$ veya $\varepsilon = \left\| V^t, V^{t-1} \right\|$ yakınlık değerine ulaşıncaya kadar devam eder.

Öyle ki, $X \in R^{p+1}$ olarak verilen datalar için FCM kümeleme algoritmasıyla girdi-çıktı uzayı küme merkezlerine karşılık gelen c bölgeye bölümlenmiş olur.

4. YAPAY SİNİR AĞ YAPILARI VE ÖĞRENME ALGORİTMALARI

4.1. Biyolojik Sinir Ağları

YSA'lar, biyolojik sinir ağlarının bir modelidir. Bu durumda YSA'ları anlayabilmek için öncelikle biyolojik sinir ağlarını tanımak gerekir. Biyolojik sinir ağları insan beyninde bulunan milyarlarca sinir hücresinden oluşur. İnsan beyninde 10^{10} adet sinir hücresi ve hücreler arasında akışı sağlayan $6x10^{13}$'ten fazla bağlantı bulunmaktadır.



Şekil 4.1 : Biyolojik Sinir Hücresi.

Bugün hala insan beyninin nasıl çalıştığı tam olarak bilinmemektedir. Teknolojideki ve genetik bilimindeki gelişmelere paralel olarak beynin nasıl çalıştığını anlamaya yönelik yapılan çalışmaların artmasına rağmen insan beyni hala gizemini korumaktadır. YSA konusundaki çalışmaların, beynin gizemi çözülmeye başladıkça sıçramalar göstereceği düşünülmektedir. İnsan beyni, çok hızlı çalışabilen mükemmel bir bilgisayar gibi görülebilir. Bugün insan beyninin kapasitesinin çok küçük bir oranında kapasiteye sahip ve çalışabilen bir makine yapılsa olağanüstü bilgi işleme ve kontrol edebilme mekanizmaları geliştirmek ve mükemmel sonuçlar almak mümkün olabilirdi. Biyolojik sinir ağlarının performansları küçümsenemeyecek kadar yüksektir ve son derece karmaşık olayları bile işleyebilecek yetenektedir. Bu nedenle YSA modelleri kullanılarak bu yeteneğin bilgisayara kazandırılması amaçlanmaktadır.

4.2. Yapay Sinir Ağları

YSA'lar insan beyninin bazı fonksiyonlarının özellikle öğrenme fonksiyonunun benzetimine dayanan sistemlerdir. YSA'lar birbirine hiyerarşik olarak bağlı ve paralel olarak çalışabilen yapay hücrelerden oluşur. Bu hücreler nöron veya sinir olarak adlandırılan işlem elemanlarıdır. YSA'larda, işlem elemanlarının birbirine bağlandıkları ve bu bağlantıların bir ağırlığı olduğu kabul edilir. YSA'ların temel yapısı insan beynine sıradan bir bilgisayarınkinden daha çok benzemektedir fakat birimleri biyolojik sinirler kadar karmaşık değildir. Çoğu sinir ağının yapısı, beyin kabuğundaki bağlantılarla karşılaştırıldığında oldukça basit kalmaktadır.

Bir YSA'nın en temel görevi, kendisine tanıtılan bir girdi kümesine karşılık gelebilecek bir çıktı kümesi belirlemektir. Bunun için ağ, ilgili olayın örnekleri ile eğitilerek genelleme yapabilecek yeteneğe kavuşturulur. Bu genelleme ile benzer olaylara karşılık gelen çıktı kümeleri belirlenir (Öztemel, 2003).

Şimdilik sıradan bir bilgisayarda, akla uygun bir sürede taklit edilebilmesi için bir ağın oldukça küçük olması gerekir. Fakat zamanla daha hızlı ve daha paralel çalışan bilgisayarlar piyasaya çıktıkça gelişmeler sağlanması beklenmektedir. YSA'ları geliştirmekteki amaç geleneksel bilgisayarların yetersiz kaldığı sınıflandırma, kümeleme, duyu-veri işleme gibi alanlarda çalışabilen sistemler geliştirmektir.

4.2.1. YSA'nın Tarihçesi

YSA'lar ile ilgili çalışmalar 1940'lı yıllarda başlamıştır. İlk YSA modeli 1943 yılında bir sinir hekimi olan Warren McCulloch ve bir matematikçi olan Watter Pitts tarafından gerçekleştirilmiştir (McCulloch and Pitts, 1943). İnsan beyninin hesaplama yeteneğinden esinlenerek elektrik devreleriyle basit bir sinir ağı modeli oluşturmuşlardır. Bu çalışma, girdilerinin uyarmalı ve engellemeli olduğu iki durumlu basit bir nöronun tanımlandığı bir makale olarak sunulmuştur. 1948 yılında Wiener "Cybernetics" isimli kitabında, sinirlerin çalışması ve davranış özelliklerine değinmiştir.

YSA'lar için tasarlanan ilk öğrenme yöntemi McGill Üniversitesi'nden Donald Hebb tarafından önerilmiştir (Hebb, 1949). "Organization of Behavior" isimli

kitabında Hebb, öğrenebilen ve uyum sağlayabilen sinir ağları modeli için temel oluşturacak Hebb kuralını ortaya koymuştur. Hebb kuralı sinir ağının bağlantı sayısı değiştirilirse, öğrenebileceğini öngörmektedir. 1950'li yıllardan sonra bir çok araştırmacı Hebb kuralından esinlenerek YSA'ların hesaplama gücünü artırıcı yönde çalışmalar yapmıştır.

1957 yılında Frank Rosenblantt tarafından geliştiren Perceptron ilerleyen yıllarda ileri sürülen en önemli YSA modeli olmuştur (Rosenblatt, 1958, 1959, 1962). Perceptron, beyin işlevlerini modelleyebilmek amacıyla yapılan çalışmalar sonucu ortaya çıkan tek katmanlı, eğitilebilen ve tek çıkışa sahip olan bir modeldir. Perceptron modelinin Hebb modeline göre kuvvetli olan tarafı bağlantılar üzerindeki ağırlık değerlerinin güncellenebilmesidir. Daha sonra perceptron modelinin öğrenmesinde karşılaşılan problemler ve güncellemeli öğrenmenin ancak belli şartlar sağlandığında belli bir değere yaklaştığı matematiksel olarak ispat edilmiştir (Minsky and Papert, 1969).

1959 yılında, Bernard Widrow ve Marcian Hoff ADALINE ve MADALINE diye adlandırdıkları ağ modellerini geliştirmişlerdir. MADALINE, telefon hatlarında oluşan yankıları yok eden bir uyarlanabilir süzgeç olarak kullanılmış, gerçek dünya sorunlarına uygulanmış olan ilk sinir ağıdır ve hala kullanılmaktadır.

1969 yılında Minsky ve Papert Perceptron'un yetersizliğini görmüşler ve ExclusiveOR (XOR) problemlerini çözemediğini ispatlamışlardır. Bunun için iki katmanlı ileri beslemeli ağların kullanılabileceğini ileri sürmüşler ve iki katmanlı ileri beslemeli ağların tek katmanlı ağlardaki bir çok sınırlamayı ortadan kaldırdığını göstermişlerdir. Fakat gizli katmanların ağırlıklarının nasıl değiştirileceği konusunda bir yöntem önerememişlerdir. Bu probleme Rumelhart ve arkadaşları geri yayılım yöntemiyle bir çözüm getirmişlerdir.

1970'li yıllarda Kohonen çağrışımlı bellek üzerine çalışmalarda bulunmuştur. Bu çalışmalarını Özdüzenlemeli Özellik Haritaları olarak 1982'de yeniden düzenleyerek yayınlanmıştır (Kohonen, 1982). Çağrışımlı bellek üzerine çalışmalar Anderson tarafından (Anderson, 1968, 1972) İçerik Adreslenebilir Bellek ve Carpenter ve Grossberg tarafından Uyarlamalı Rezonans Teorisi konusunda devam edilmiştir (Carpenter and Grossberg, 1985, 1987, 1987b, 1990).

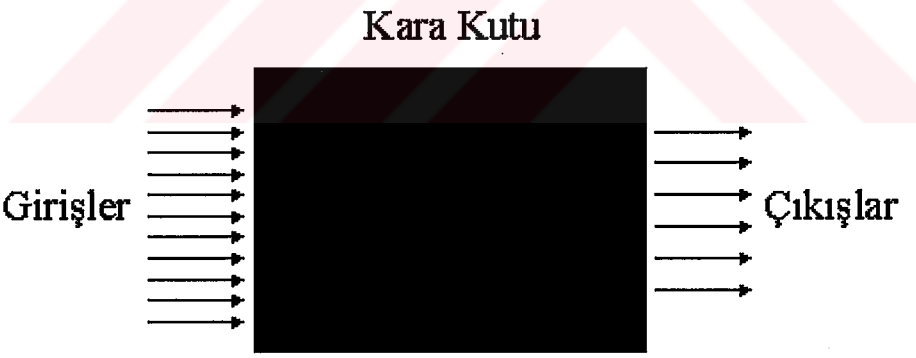
1984 yılında Kohonen danışmansız öğrenme ağlarını geliştirilmiştir. 1985 yılından sonra Amerikan Ulusal Fizik Akademisi (National Academy of Science of The USA) tarafından yapay sinir ağları ile ilgili çalışmalar izlenmeye ve desteklenmeye başlamıştır.

1987 yılında Elektrik Elektronik Mühendisliği Enstitüsü (Instute of Electrical Electronic Engineering (IEEE)) tarafından sinir ağlarını konu alan ilk uluslararası konferans 1800'ü aşkın katılımcıyla gerçekleştirilmiştir.

Optik ve dijital teknolojilerdeki gelişmeler ve VLSI gibi teknolojilerin keşfedilmesiyle bilgisayarların hızları artmış ve bu sayede YSA uygulamaları da hız kazanmıştır. Günümüzde YSA'lar teorik ve laboratuar çalışmaları olmaktan çıkmış ve günlük hayatta karşılaşılan sistemler olarak hayatımıza girmeye başlamıştır.

4.2.2. YSA'nın Özellikleri

YSA'lar giriş ve çıkışları bulunan kara kutular gibi düşünülebilir. Bir çok girişi ve bir çok çıkışı vardır. Kara kutunun işlevi basitçe matematiksel bir fonksiyonu temsil etmek şeklinde açıklanabilir. Ama aslında tam olarak bir matematiksel karşılığı yoktur. Aksi olsaydı YSA'ları kullanmaya gerek kalmazdı.



Şekil 4.2 : Yapay Sinir Ağlarının Genel Görünümü

YSA'larda her bir işlem birimi, basit anahtar görevi yapar ve şiddetine göre, kendisine gelen sinyalleri söndürür ya da iletir. Böylece sistem içindeki her birim belli bir ağırlığa sahip olmuş olur. Her birim sinyalin gücüne göre açık ya da kapalı duruma geçer ve basit bir tetikleyici görev üstlenir. Ağırlıklar karakterler arasında ilgi kurmayı sağlar. Yapay sinir ağları araştırmalarının odağındaki soru, ağırlıkların sinyalleri nasıl değiştirmesi gerektiğidir. Bu noktada her hangi bir formdaki bilgi

girişinin ne tür bir çıkışa çevrileceği, değişik modellerde farklılık göstermektedir. Bir sinir ağ tasarımında, bilgisayarda saklı olan bilgi tüm sisteme yayılmış küçük yük birimlerinin birleşmesinden oluşmaktadır. Ortama yeni bir bilgi aktarıldığında yerel büyük bir değişiklik yerine tüm sistemde küçük bir değişiklik meydana gelmektedir.

YSA'ların iki temel özelliği bu sistemlere kuvvetli bilgi işlem özelliği kazandırmaktadır. İlki bu sistemlerin paralel dağıtık yapısı, ikincisi ise öğrenebilme ve genelleme yapabilme özellikleridir. Genelleme yapabilmesi, sistemin eğitimi sırasında öğrenmediği bazı bilgileri tutarlı sonuç olarak verebilmesidir. Ancak yine de YSA'ların, insan beyninin fonksiyonlarını tam anlamıyla benzetmesi uzun bir süreç gerektirecektir. YSA'ların avantaj ve yetenekleri şu şekilde özetlenebilir.

Bir nöron doğrusal olmayan bir yapıya sahiptir. Bu sebeple nöronlardan meydana gelen YSA'larda doğrusal olmayan özelliğe sahiptir. Özellikle doğrusal olmayan girdi bilgilerini işleyen sistemler için bu önemli bir avantajdır.

Ağırlık değerleri üzerinde yapılacak değişiklikler sayesinde YSA sistemleri bulundukları ortama kolayca uyum sağlayabilme özelliğine sahiptir. Belli bir ortam için eğitilmiş olan bir sistem, ortam değişikliği durumunda tekrar eğitilebilir.

Sınıflama yapabilen bir YSA modeli sadece verilen girdinin hangi sınıfa ait olduğunu belirlemekle kalmaz, aynı zamanda öğrenme oranı kullanarak belli bir hassasiyet altında gerçekleştirir. Bu da sistemin sınıflama performansını arttırır.

YSA modelleri yapılan performans testleri ile hataya dayanıklı sistemler olarak kabul edilmektedir (Bolt, 1992). Örneğin bir nöronun veya nöronun bir bağlantısının hizmet dışı olması durumunda da sistem doğruya yakın sonuçlar verebilir. Bunda en önemli etmen, YSA sistemlerinin bilgiyi dağıtık olarak saklama özelliğidir (Yücetürk, 1999).

YSA'lar, paralel işlem olanağı sağlar ve bu sayede hesaplama zamanını azaltır. Adaptasyon kabiliyeti sayesinde, bazı ağ tipleri kendi kendini eğitme ve dinamik sistemler gibi kararlı hale gelebilme özelliğine sahiptirler. Genelleme kabiliyeti sayesinde eğitme aşamasında kullanılan giriş değerleri haricindeki değerlerce de kabul edilebilir sonuçlar verebilmektedir.

YSA'lar, geç öğrenir, çabuk düşünür ve kısa yapılandırılma zamanına ihtiyaç duyar. Fakat karmaşık ve uzun zaman gerektiren uygulamalar için, simülatörler veya hızlandırıcı kartlar mevcuttur. YSA'ların paralel yapısı, hızlı işlem görmesine ve VLSI teknolojisine uygun bir özellik göstermesine imkan tanımaktadır. Bu da kontrol veya sinyal işleme gibi gerçek zamanlı uygulamalarda VLSI teknolojisi ile kolayca çözüm getirmektedir (Yücetürk, 1999).

YSA'larda hafıza bir çok birim üzerine dağıldığı için gürültüye yani veriler içerisindeki kirliliğe karşı koyma özelliği kazandırmaktadır.

Nöron yapısı, birçok yapay sinir ağı modelinde aynı özellikleri göstermektedir. Bu genelleme farklı uygulamalar için kullanılan teori ve öğrenme algoritmalarının paylaşılabilirliğini sağlamaktadır. Böylece modüler ağlar oluşturmak mümkün olmaktadır (Yücetürk, 1999).

YSA'ların donanım bağımlı çalışmaları önemli bir sorun olarak görülebilir. Ağların özellikle, gerçek zamanlı bilgi işleyebilmeleri paralel çalışabilen işlemcilerin varlığına bağlıdır. Ayrıca bir ağın nasıl oluşturulması gerektiğini belirleyecek kuralların olmaması da başka bir dezavantajdır. Her problem farklı sayıda işlemci gerektirebilir. Bazı problemleri çözebilmek için gerekli olan paralel işlemcilerin tamamını bir arada paralel olarak çalıştırma mümkün olmayabilir.

Probleme uygun ağ yapısının genellikle deneme yanılma yoluyla belirlenebiliyor olması da önemli bir problemdir. Eğer problem için uygun bir ağ oluşturulmaz ise çözümü olan bir problemin çözülememesi veya performansı düşük çözümlerin elde edilmesi söz konusu olabilir. Bu aynı zamanda bulunan çözümün en iyi çözüm olduğunu da garanti etmez. Yani YSA'lar kabul edilebilir çözümler üretebilir.

Bazı ağlarda öğrenme katsayısı gibi ağın parametre değerleri, her katmanda olması gereken düğüm sayısı, katman sayısı gibi değerlerin belirlenmesinde bir kural olmaması da başka bir problemdir.

Ağın öğreneceği problemin ağa tanıtımı çok önemlidir. YSA'lar sadece nümerik bilgilerle çalışmaktadır. Bu nedenle de problemin nümerik gösterime dönüştürülmesi gerekir. Uygun bir gösterim mekanizmasının kurulamamış olması

problemin çözümünü engelleyebilmekte veya düşük performanslı bir çözüm elde edilmesine neden olabilmektedir.

Ağın eğitiminin ne zaman bitirileceğine karar vermek için de geliştirilmiş bir yöntem yoktur. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlanması için yeterli görülmektedir. Fakat sonuçta en iyi öğrenmenin gerçekleştiği söylenememektedir. Sadece iyi çözümler üretebilen bir ağ oluştu denilmektedir (Öztemel, 2003). Bir diğer sorunda ağın davranışlarının açıklanamamasıdır. Bir probleme çözüm ürettiği zaman bunun nasıl ve nereden üretildiği konusun da bir bilgi bulmak mümkün değildir.

4.2.3. YSA'nın Uygulama Alanları

Tüm YSA'lar, nöronlar, nöronlar arasında ki bağlantılar ve transfer fonksiyonu kavramına dayandığından, sinir ağlarının farklı yapıları veya mimarileri arasında bir benzerlik vardır. Değişimlerin çoğu, farklı öğrenim kurallarından ve bu kuralların bir ağın topolojisini değiştirme şeklinden kaynaklanmaktadır. Bu kategoriler ağların mimarisi hakkındaki karışıklıkları engellemeye yöneliktir.

Başarılı YSA uygulamaları incelendiğinde ağların, doğrusal olmayan, çok boyutlu, gürültülü, karmaşık, kesin olmayan, eksik, kusurlu, hata olasılığı yüksek sensör verilerinin olması durumunda ve problemin çözümü için özellikle bir matematik modelin ve algoritmanın bulunmaması hallerinde yaygın olarak kullanıldıkları görülmektedir. Bu amaçla geliştirilmiş ağlar genel olarak şu işlevleri yerine getirmektedir:

- Olasılıksal fonksiyon kestirimleri
- Sınıflandırma
- İlişkilendirme veya örüntü eşleştirme
- Zaman serileri analizi
- Sinyal filtreleme
- Veri sıkıştırma
- Örüntü tanıma
- Doğrusal olmayan sinyal işleme
- Doğrusal olmayan sistem modelleme
- Optimizasyon

- Zeki ve doğrusal olmayan kontrol

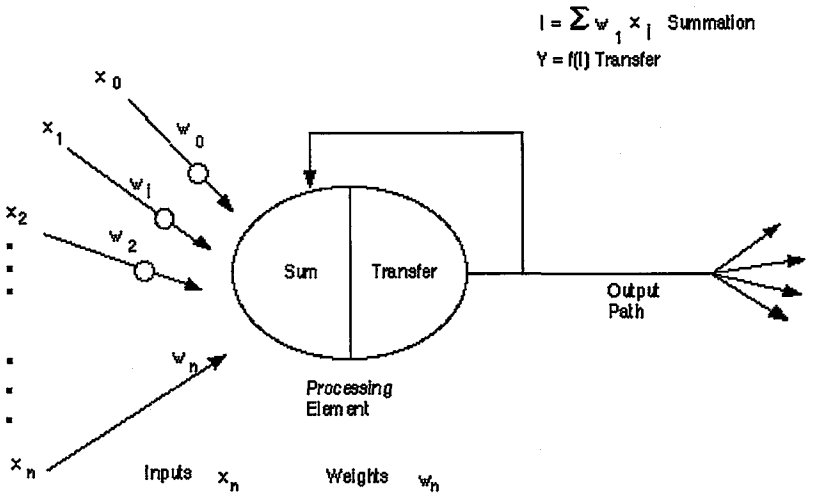
Yukarıda belirtilen teorik konular dışında finansal konulardan mühendisliklere ve tıp bilimine kadar pek çok alanda da YSA uygulamalarından bahsetmek mümkündür.

- Veri madenciliği
- Bankalardan kredi isteyen müracaatları değerlendirme
- Ürünün pazardaki performansını tahmin etme
- Kredi kartı hilelerini tahmin etme
- Zeki araçlar ve robotlar için optimum rota belirleme
- Güvenlik sistemlerinde ses ve parmak izi tanıma
- Robot hareket mekanizmalarının kontrol edilmesi
- Kalite kontrolü
- Radar ve sonar sinyallerini sınıflandırma
- Kan hücreleri reaksiyonları ve kan analizlerini sınıflandırma
- Beyin modellemesi çalışmaları

Bu çalışmalar çoğaltılabilir. Günümüzde hemen hemen her alanda YSA çalışmalarını görmek mümkündür. Çünkü gerçek hayatta karşılaşılan problemlerin çoğu doğrusal olmayan modeller gerektirmektedir. Bu durumda da geleneksel yöntemler ile çözüm üretilmesi zorlaşmakta veya imkansız olmaktadır. YSA'lar geleneksel yollarla çözümü zor olan problemlere daha doğru ve gerçekçi çözümler üretmeyi amaçlamaktadır.

4.3. Yapay Nöron Modeli ve Elemanları

Bir YSA'nın birim elemanı yapay sinirlerdir. Yapay sinirler, nöron ya da düğüm olarak da adlandırılırlar. YSA'lardaki nöronların bir çıkışları ve genelde birden fazla girişleri vardır. Bu çıkış ve girişler diğer nöronlara, YSA'nın çıkışına veya YSA'nın girişine bağlanabilir.



Şekil 4.3 : Basit Bir Yapay Nöron

x'lerin her biri girdi (input) değerleri, *w*'ler de nöronların ağırlık değerleridir. *x* değerleri *w* değerleriyle çarpılır ve toplam (summation) fonksiyonu kullanılarak ağırlıklandırılmış girdi değerleri toplanır. Toplam bir transfer fonksiyonundan geçirilerek tek bir çıktı (output) olarak verilir. Bir sinir ağında işlem elemanı olan nöronun görevi, girişindeki sayıları kendi ağırlık değerleri ile çarpıp sonra çarpımları toplayıp, toplamı bir aktivasyon (transfer) fonksiyonundan geçirdikten sonra çıkışa vermektir. Ancak giriş ve çıkış katmanındaki nöronlar bu kuraldan hariçtir.

Nöronlarının tamamı YSA'nın girişini oluşturan ve bu nöronların kendi girişi olmayan katman *girişi katmanı*, nöronlarının tamamı çıkışı oluşturan ve bu nöronların kendi çıkışı olmayan katman *çıkış katmanı* olarak adlandırılır. Giriş ve çıkış katmanları arasında kalan diğer katmanlar ise ara katmanlar ya da dışarıdan görünmedikleri için *gizli katman* olarak adlandırılırlar.

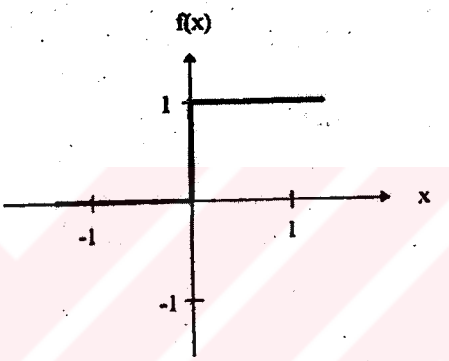
Giriş katmanındaki nöronlar sadece içerdikleri değeri çıkışa aktarırlar. Çıkış katmanındaki nöronlar ise sadece kendi girişlerindeki verilerin uygun ağırlıklarla çarpılmış hallerini toplayıp saklarlar. Bu işleme ilerleme (propagation) denilir. Bir nöronun çıkışındaki değerler genelde (0,1) sigmoid veya (-1,1) tanh arasında olması istenir. Bu değerler o çıkışın seçilme derecesini gösterir.

kombinasyonları verebilir. Aktivasyon fonksiyonu o zaman katı limitli veya adım fonksiyonu olabilir. Aktivasyon fonksiyonu, sinirin girdilerini belli bir işlemden geçirerek çıktı değerine dönüştüren fonksiyonlardır. Sıklıkla kullanılan aktivasyon fonksiyonları şunlardır.

a) Eşik Fonksiyonu

Belli bir eşik değerine göre tanımlı olan fonksiyon olup matematiksel ifadesi şöyledir:

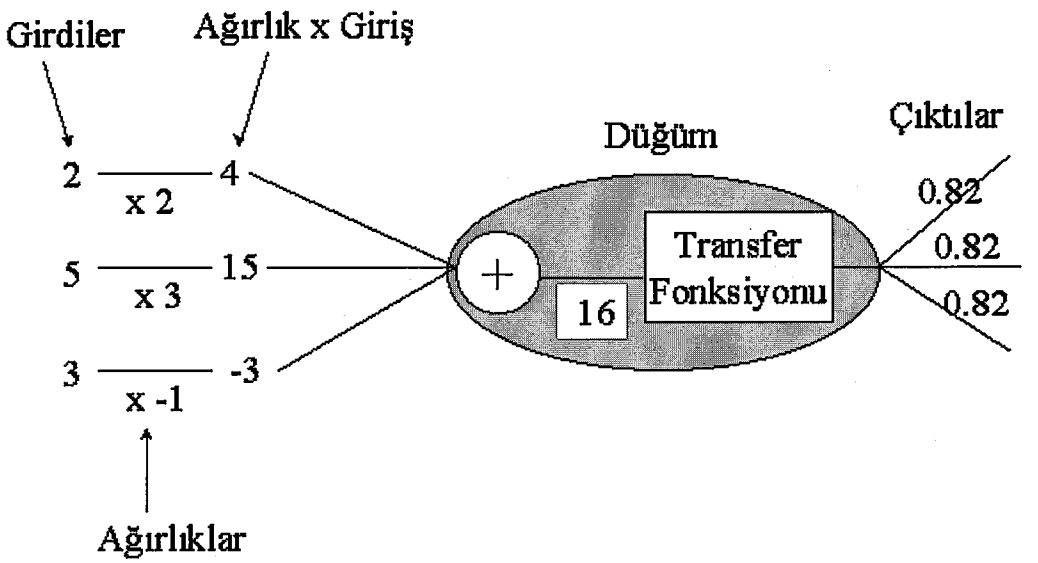
$$f(x)=\begin{cases}1 & eğer \quad x\geq 0 \quad ise\\ 0 & eğer \quad x<0 \quad ise\end{cases} \tag{4.1}$$



Şekil 4.5: Eşik Fonksiyonu

b) Kısmi Doğrusal Fonksiyon

$$f(x)=\begin{cases}1 & eğer \quad x\geq 1/2 \quad ise\\ x & eğer \quad -1/2>x>1/2 \quad ise\\ 0 & eğer \quad x\leq -1/2 \quad ise\end{cases} \tag{4.2}$$



Şekil 4.4: Nöronun Matematiksel Modeli

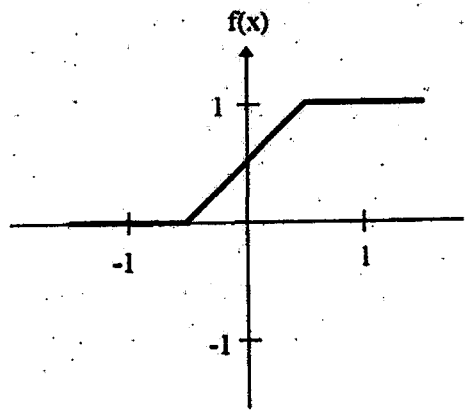
Burada aktivasyon fonksiyonunun görevi, hem çıkışları belirli değerler arasında tutmak hem de sürekli bir fonksiyon oluşturmaktır. Fonksiyonun sürekli olması türevinin alınması, türevinin alınması da eğitme aşamasındaki algoritmalar için gereklidir. Fonksiyonun türev karakteristikleri eğitme aşamasının hızını ve başarısını da etkiler.

Nöron davranışını belirleyen önemli etmenlerden biri nöronun aktivasyon fonksiyonudur. Biyolojik nöronlarda, sinire gelen sinyaller toplam belli bir değeri aştığında nöronun kısa süreli bir darbe gönderdiği bilinmektedir (Efe, 2000).

Ağırlıklar toplamına eşit olan toplam fonksiyonu, sonucu aktivasyon fonksiyonu olarak bilinen bir algoritmik işlemle çıktıya çevirir. Aktivasyon fonksiyonunda toplama sonucunun sinirin çıkışında belirtilmesi için, bir eşik değeri (threshold) ile karşılaştırılır. Toplam bu eşik değerinden daha büyük ise, işlem elemanı sinyali üretir, girdi toplamı ve ağırlık sonuçları, eşik değerinden küçük ise sinyal üretmez.

Eşik değeri ve aktivasyon fonksiyonu normalde doğrusal değildir. Doğrusal fonksiyonlar sınırlıdır. Gerçek problemleri doğrusal fonksiyonlarla ifade etmek pek mümkün olmaz. Bu da perceptron gibi ilk modellerin karşılaştığı bir problemdir.

Toplama sonucu pozitif veya negatif olsa da transfer fonksiyonu olabildiğince basit olmalıdır. Ağ çıkışında, 0 veya 1, 1 veya –1 ya da başka numara



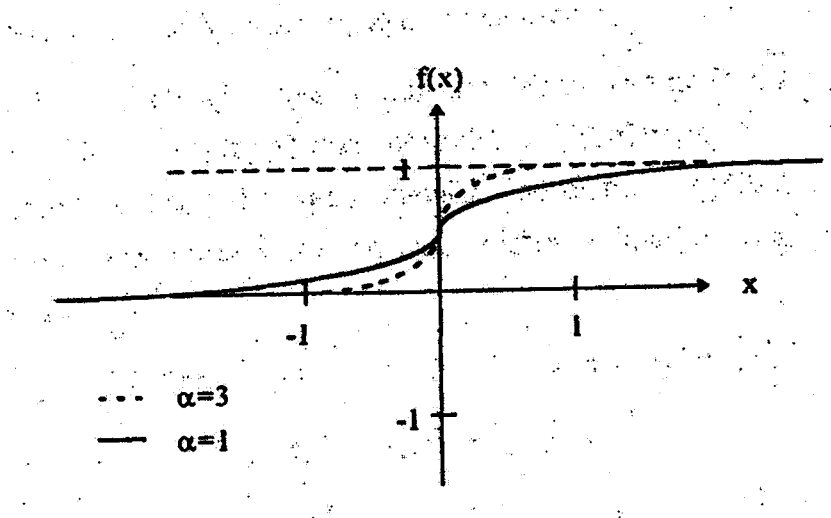
Şekil 4.6: Kısmi Doğrusal Fonksiyon

c) Sigmoid Fonksiyon

Sigmoid fonksiyonlar *S* şekilli türevi alınabilir fonksiyonlar oldukları için yapay sinir ağ modellerinde özellikle de geri yayılımlı öğrenme algoritmalarında sıkça kullanılırlar.

Sigmoid fonksiyonlar, elde edilmek istenen çıktı aralığına göre ikili sigmoid ve çift kutuplu sigmoid fonksiyonlar olarak sınıflandırılabilirler. İkili sigmoid fonksiyonlar 0 ile 1 aralığında değerler alabilen fonksiyonlardır. Matematiksel tanımları şöyledir:

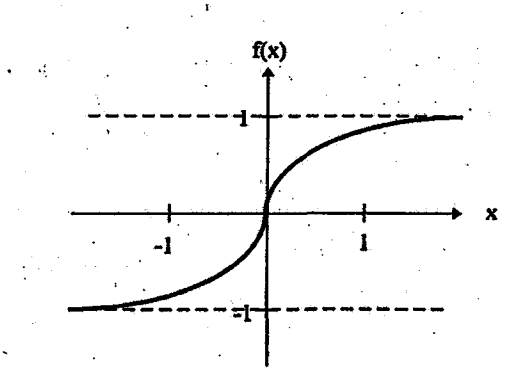
$$f(x) = \frac{1}{1+\exp(-\alpha x)} \qquad (4.3)$$



Şekil 4.7: İkili Sigmoid Fonksiyon

Çift kutuplu sigmoid fonksiyonlar ise -1 ile +1 aralığında değerler alabilen fonksiyonlardır. İkili sigmoid fonksiyonu f_{old} olarak kabul edilirse, çift kutuplu sigmoid fonksiyonu şu şekilde elde edilir.

$$f(x) = 2\,fold(x)-1 \qquad (4.4)$$



Şekil 4.8: Çift Kutuplu Sigmoid Fonksiyonu

Matematiksel olarak bu eğrilerin özelliği fonksiyon ve türevlerinin sürekli olmasıdır. Bu seçenek gayet iyi çalışır ve çoğu zaman tercih edilen aktivasyon fonksiyonudur. Özel ağ mimarileri için diğer aktivasyon fonksiyonları önerilir.

4.4. YSA'nın Temel Mimarileri

YSA'lar da nöronlar genellikle katmanlar halinde düşünülür. Bütün nöronların aynı seviyede konumlandığı tek katmanlı ağlar olduğu gibi, her katmanda farklı sayıda nöron bulunan çok katmanlı ağlar da bulunmaktadır. Genellikle aynı katmanda bulunan nöronlar aynı şekilde işlem yaparlar. Başka bir deyişle aynı katmandaki nöronların aktivasyon fonksiyonları ve diğer nöronlarla bağlantı şekilleri aynıdır. Nöronlar katmanlar halinde düzenlenmesine ve katmanlar arasındaki bağlantı düzenine ağ mimarisi adı verilir. YSA'ların hepsinde bir girdi katmanı bulunur.

YSA'ların çalışma hızlarını etkileyen en önemli etken mimarilerine ait olan katmanların ve nöronların sayısıdır. Her katmanda bir ya da daha fazla nöron bulunur ki genelde birden fazladır. Her katmandaki nöron sayısı ve katman sayısı sonuç olarak toplam nöron sayısını etkiler. Toplam nöron sayısı ve nöronlar arasındaki bağlantılar ne kadar çok ise YSA'ların çalışma hızı buna paralel olarak üstel bir şekilde azalır. Zaten şimdilik YSA'ların en büyük problemi bu hız problemidir.

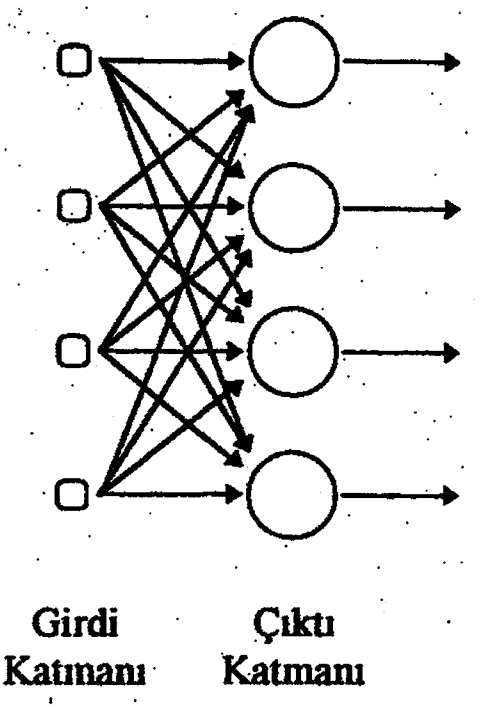
Çünkü gerçekçi problemler genelde çok sayıda nöron ve katman gerektiren karmaşıklıktadır. Dolayısıyla YSA'lar gerçek problemlerde biraz yavaş çalışmaktadır.

YSA'lar, algoritmik olmayan, uyarlanabilir, paralel programlama, dağıtılmış programlama gibi tekniklerin gelişmesine katkıda bulunmuşlardır. Bu teknikler sayesinde bilgisayarlarında öğrenebileceği gösterilmiştir. Özellikle olaylar hakkında tam ve kesin bilgilerin olmadığı fakat örneklerin bulunduğu durumlarda oldukça etkin olarak kullanılabilecek bir karar verme aracı ve hesaplama yöntemi olarak görülebilir.

YSA'larda ağ mimarisi, öğrenme algoritması ile yakından ilgilidir. Bu sebeple ağdaki nöronların birbirleriyle bağlantı yapısını belirleyen etmenlerden en önemlisi öğrenme algoritmasıdır.

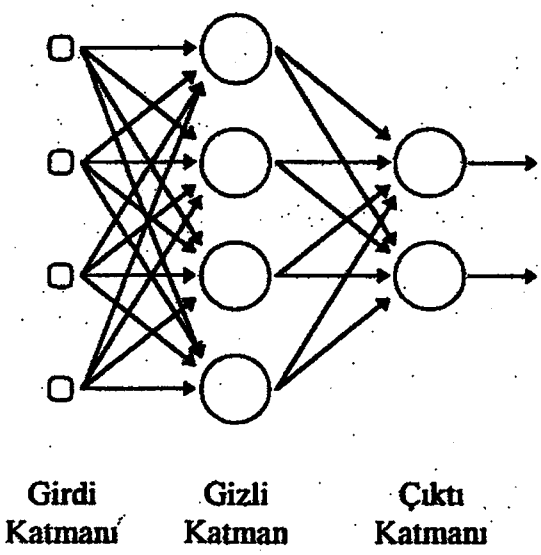
YSA'ların nöronları ve bağlantıları farklı şekillerde bir araya gelebilir ve bu farklılıklara göre değişik isimler alırlar. YSA mimarileri, nöronlar arasındaki bağlantıların yönlerine göre ileri beslemeli (feedforward) ve geri beslemeli (feedback veya recurrent) ağlar olmak üzere ikiye ayrılır.

Bilgi akışının sadece girdi katmanınıdan çıktı katmanına doğru olan ağlar ileri beslemeli ağlar olarak adlandırılır. Her katmandaki nöronlara sadece önceki katmandaki nöronlardan girişlere izin verilir. Bir nöron kendinden sonraki her hangi bir nörona bağlanabilirken kendisine asla bağlanmaz. Son katmandaki işaretler ağın çıkışıdır. İleri beslemeli ağlar, tek katmanlı ve çok katmanlı ağlar olarak ikiye ayrılır. Tek katmanlı ağlar, en basit ağ tipi olup bir çıktı katmanı ve buna bağlı bir girdi katmanı bulunmaktadır. Girdi katmanı dış dünyadan sinyallerin alındığı katman olup, çıktı katmanı ise ağın elde ettiği sonuçların dış ortama aktarıldığı katmanlardır. İleri beslemeli ağlar grubunda olan bu ağın modeli şekil (4.9) da gösterilmektedir.



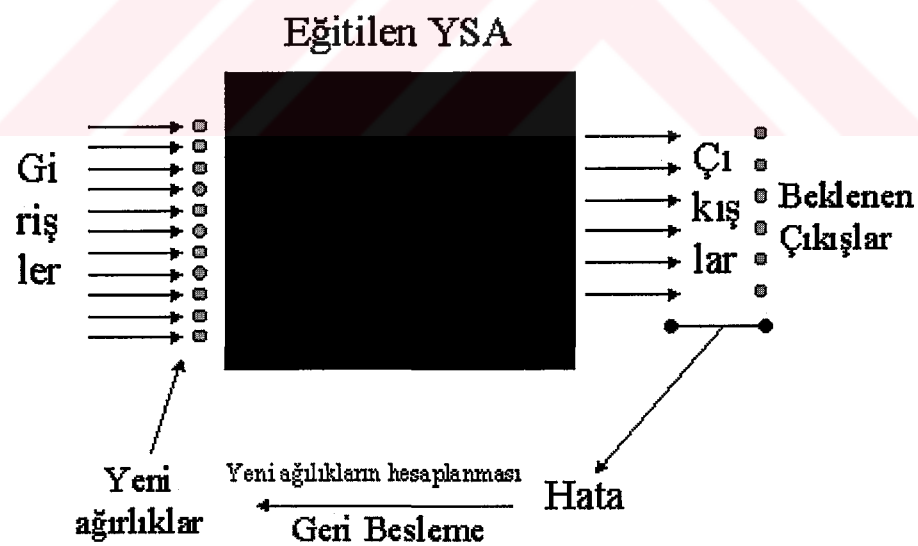
Şekil 4.9 : Tek Katmanlı İleri Beslemeli Ağ Modeli

Çok katmanlı ileri beslemeli ağlar, tek katmanlı ağlardaki girdi ve çıktı katmanından başka, bir veya daha fazla sayıda gizli katman içeren ağlara çok katmanlı ağ adı verilmektedir. Dış dünya tarafından doğrudan müdahale edilemediği için gizli katman adı verilen katmanda bulunan birimlere de gizli birimler adı verilir. İleri beslemeli çok katmanlı ağlarda her hangi bir katmanda bulunan nöronların girdisi genellikle bir önceki katmandan olmaktadır. Çok katmanlı ağlar tek katmanlı ağlara göre daha karmaşık problemlere çüzüm getirebilmekle beraber, bu tip ağların eğitilmesi daha zordur.



Şekil 4.10 : Çok Katmanlı İleri Beslemeli Ağ Modeli

Geri beslemeli veya tekrarlanan ağlarda en azından bir nöronun geriye yayıldığı bir dönüş bağlantısı vardır. Tekrarlanan ağlar tamamen veya parçalı olarak geri besleme yollarına sahiptir. Bu tür ağların tasarımları ve davranışları oldukça karmaşıktır. Geri beslemeli ağların ileri beslemeli ağlardan farkı, en az bir tane geri besleme döngüsünün bulunmasıdır.



Şekil 4.11 : Geri Beslemeli Ağ Yapısı

4.5. YSA'da Öğrenme

YSA'larda bilgi, nöronlar arasındaki bağlantılar üzerindeki ağırlık değerleri üzerinde tutulduğu için ağların eğitimi ve öğrenme bu değerlerin verilmesi ve değiştirilmesi anlamına gelmektedir. Sinir ağlarında istenen sonucun elde edilmesi için ağın uyarlanabilir olması gerekir. Bunu sağlamak için uygun ağırlık değerleri ve doğru bağlantılar seçilmelidir. Öğrenme YSA'nın ayrılmaz bir parçasıdır. Öğrenme, giriş değerlerine veya bu girişlerin çıkışlarına bağlı olarak ağın bağlantı ağırlıklarını değiştiren öğrenme kuralı ile gerçekleştirilir.

Bir sinir ağının eğitiminde ağırlık değerlerinin başlangıç değeri kadar öğrenme oranının belirlenmesi de öğrenme performansı açısından önemlidir. Öğrenme oranı ağırlıkların değişim miktarını belirler. Büyük değerler seçilirse sıçramalar meydana gelebilir, küçük değerler seçilirse de öğrenme zamanı uzayabilir. Tecrübeler genellikle 0.2-0.4 arasındaki değerlerin kullanıldığını göstermektedir. Fakat bu tamamen ilgili probleme bağlıdır. Bazı uygulamalarda öğrenme katsayısı 0.6 değerini aldığı zaman en başarılı sonuçlar verdiği görülmektedir (Öztemel, 2003).

YSA sistemlerinin girdiler ve çıktılar arasında eşleme yapabilmesi eğitim ile mümkün olur. YSA'ların eğitimi konusunda iki yaklaşım mevcuttur. Sisteme başlangıçta girdi değerleri ve bu girdi değerlerine karşılık gelen çıktı değerleri tanıtılırsa bu tür eğitime danışmalı eğitim ve bu öğrenme metoduna da danışmanlı öğrenme metodu denir. Danışmalı öğrenmede, girdi kümesinden rastgele seçilen elemanlar bir veya birden fazla defa sistemi eğitmek amacıyla sisteme geri beslenir. Eğitim sonunda sistem verilen girdiye göre istenen çıktıyı elde edebilecek konuma gelir. Sisteme sadece girdi değerleri tanıtılıp çıktı değerleri sistem tarafından üretiliyorsa, yani başlangıçta çıktı değerleri bilinmiyorsa bu tür eğitim danışmansız eğitim ve bu öğrenme metoduna da danışmansız öğrenme metodu denir.

Ağların eğitimi için kullanılan öğrenme kuralları genellikle danışmanlı öğrenme (supervised learning), danışmansız öğrenme (unsupervised learning) olmak üzere iki öğrenme yöntemi başlığı altında toplanabilir.

4.5.1. Danışmanlı Öğrenme

YSA'larda yaygın olarak kullanılan yöntem, eğitim sırasında sisteme bir girdi ve bir hedef çıktı vektörlerinin çift olarak verilmesi ve bunlara göre sistemdeki ağırlık değerlerinin güncellenmesi ve değiştirilmesidir. Bu öğrenme yöntemine Danışmanlı Öğrenme denilmektedir. Kısaca, sisteme girilmiş olan bir bilginin sonucunun da sisteme söylenmesidir. Örneğin sınıflama yapacak bir sistemde iki farklı gruptan örnekler giriliyorsa, örneklerin girilmesi sırasında hangisinin birinci gruba hangisinin ikinci gruba dahil olduğu da sisteme girilmekte, ağırlıkların güncellenmesi bu iki verinin değerlendirilmesi sonucunda yapılmaktadır.

Bu yöntemi kullanan YSA'larda gerçek çıkış istenen çıkışla kıyaslanır. Rastgele değişen ağırlıklar ağ tarafından öyle ayarlanır ki, bir sonraki döngüde gerçek çıkış ile istenen çıkış arasında daha yakın karşılaştırma üretebilsin. Öğrenme yöntemi, bütün işlem elemanlarının anlık hatalarını en aza indirmeye çalışır. Bu hata azaltma işlemi kabul edilebilir bir doğruluğa ulaşana kadar ağırlıklar devamlı olarak güncellenir (Elmas, 2003).

Danışmanlı öğrenmede her giriş değeri için istenilen çıkış sisteme tanıtılır ve sistem kendisini sinir ağının giriş-çıkış ilişkisini gerçekleştirene kadar aşama aşama ayarlar. Danışmanlı öğrenmeye çok katmanlı perceptron (multilayer perceptron), geriye yayılım (backpropagation), delta kuralı, Widrow-Hoff veya en küçük karelerin ortalaması (Least mean square) ve uyarlanabilir doğrusal eleman anlamına gelen ADALINE örnek olarak verilebilir (Elmas, 2003).

Birçok uygulamada, ağa gerçek veriler uygulanmak zorundadır. Bu nedenle de eğitme safhası uzun zaman alabilir. Sinir ağı, girişler için istenen istatistiksel doğruluğu elde ettiği zaman eğitme işlemi tamamlanmış kabul edilir ve sonlandırılır. Öğrenim aşaması tamamlandıktan sonra ağ kullanılmaya başlandığında, bulunan ağırlık değerleri sabit olarak alınır ve bir daha değiştirilmezler. Bazı ağ yapılarında ise ağ çalışırken çok düşük oranda eğitmeye izin verilmektedir. Bu işlem ağların değişen koşullara uyum sağlamasına yardımcı olmaktadır (Elmas, 2003).

Danışmanlı öğrenme yöntemini kullanan ağ yapıları şunlardır (Elmas, 2003):

- Perceptron

- Çok katmanlı perceptron
- Geri yayılım ağı
- Daha yüksek düzeyli sinir ağı
- İşlevsel bağ ağı

Danışmanlı öğrenme metodunda giriş verileri kümesi ve bu veriler kümesine uygun çıkış vektörleri önceden bellidir. Danışmanlı öğrenmede girdi ve çıktı değerleri başlangıçta sisteme verilir, sonra ağ verilen girdilerle işlemleri yürütür ve elde edilen sonuçları istenen çıktılarla kıyaslar. Girdi değerlerinin işlenmesi sonucu elde edilen sonuçlar ile istenilen sonuçlar arasındaki fark hata olarak adlandırılır. Bulunan bu hatalar sisteme girilerek sistem tarafından düzeltilir. Her defasında hesaplanan sonuçlar istenilen sonuç ile kıyaslanır ve bu işlem hata makul bir düzeye inene kadar ya da istenilen sonuca ulaşılana dek devam eder.

Genellikle başlangıçta rastgele olarak seçilen ağırlıklar daha sonra güncellenir. Öğrenme metodunun amacı, işlem elemanlarının güncel hatasını minimuma indirmektir. Ortaya çıkan hatayı azaltma, ağ tarafından kabul edilebilir kesinliğe ulaşılıncaya kadar ağırlıklarının sürekli olarak düzenlenmesi ile gerçekleştirilir.

Eğitim kümelerinin ihtiyaç duyulan tüm bilgiyi içermesi için oldukça geniş olması gerekir. Ağın niteliği ve ilişkileri öğrenme aşamasında önemlidir. Sadece veri kümeleri geniş olmak zorunda değildir, fakat eğitim dönemi geniş bir veri türünü içermelidir. Sistem, her şeyi birlikte öğrenmeli ve olaylar kümesinin tamamı için en iyi ağırlıkları bulmalıdır.

Danışmanlı öğrenme ile eğitilen bir ağda, ağın daha önce görmediği verilerle ne şekilde çalışacağını görmek önemlidir. Eğer sistem bu test kümesi için makul çıkışlar vermezse eğitim periyodu bitmez. Gerçekte bu test, ağın veri kümesini sadece ezberlemediğini görmek açısından önemlidir. Çünkü eğer sistem hep aynı tür verilerle eğitilirse ya da çok uzun süre eğitilirse aşırı eğitme denilen durum ortaya çıkar ve bu hiç istenmeyen bir durumdur.

4.5.2. Danışmansız Öğrenme

Danışmansız öğrenmede, sinir ağına girişin hangi veri sınıfına ait olabileceği söylenmez. Danışmansız öğrenme danışmanlı öğrenmeye göre çok daha hızlıdır ve

matematiksel algoritmaları da daha basittir. Danışmansız öğrenme metoduna, yarışmacı öğrenme, Kohonen'in özörgütlemeli harita ağları (Kohonen self-organizing map), Hebb öğrenme gibi öğrenme algoritmaları örnek olarak verilebilir.

Sınıflandırma işlemini, girdileri birbirleriyle karşılaştırarak sistemin kendisi yerine getiriyorsa, bu tür sistemlerdeki öğrenmeye danışmansız öğrenme adı verilir. Bu sistemlerde bir grup girdi vektörü sisteme verilir, ancak hedef çıktılar belirtilmez. Sistem, girdiler içinde birbirine en çok benzeyenleri gruplar ve her bir grup için farklı bir örüntü tanımlar (Yücetürk, 1999).

Danışmansız eğitimde eğitici kümede yalnızca giriş vektörleri mevcuttur. Eğitimin amacı ağın parametrelerini uygun şekilde değiştirerek giriş kümesine uygun çıkış değerlerini belirlemektir. Bu tür eğitimde en çok kullanılan algoritma yarışmacı öğrenme algoritmasıdır. Yarışmacı öğrenme bir çok yönüyle kümeleme analizine benzemektedir.Yarışmacı öğrenme ağları giriş vektörlerine karşılık gelen giriş nöronları, küme sayısı kadar çıkış nöronu ve her bir giriş nöronu ile çıkış nörunu arasındaki bağlantının kuvvetini belirten ağırlık değerlerinden oluşur. Bir çıkış nöronunun girişine bu kümeye dahil edilebilecek giriş vektörü verildiğinde çıkış nöronunun aktifleştirilmesi talep edilmektedir. Bu problemin çözümü için "Kazanan her şeyi alır"[1] stratejisi kullanılır. Buna uygun olarak tanım bazında "yarışma" adıyla adlandırılan durum ortaya çıkmaktadır. Sınıflandırıcı katmanın nöronları, kendi durumlarını, girişe verilen giriş vektörü ve ağırlık katsayılarına bağlı olarak belirler. Sonuçta kazanan nöron, "1" durumuna, diğerleri ise, "0" durumuna geçer.

Bir çıkış nöronunun, farklı kümelere ait olan giriş vektörleri verildiğinde aktifleşmemesi için, her bir çıkış nöronunun ağırlık toplamına bir sınırlama koymak gerekmektedir. Burada eğitim, aktif çıkış nöronu ile aktif giriş nöronları arasında ağırlık katsayı değerlerini arttırma, aktif olmayan nöronlar arasında ise ilişki ağırlıklarını azaltmak anlamına gelmektedir. Böyle bir eğitim sonucu her çıkış nöronu, yalnızca belirli kümeye ait giriş vektörleri verildiğinde aktifleşecek, diğer kümelere ait giriş vektörleri verildiğinde ise pasif kalacaktır.

Danışmansız öğrenme yöntemini kullanan ağ yapıları şunlardır (Elmas, 2003):

- Hopfield Ağı

[1] Winner takes all

- Olasılıksal Sinir Ağları
- Uyarlanabilir Rezonans Ağı
- Boltzman Makinesi
- Hamming Ağı
- İki Yönlü Çağrışım Belleği
- Yığın Ağı
- Karşı Yayma Ağı
- Yarışmacı Öğrenme Ağları
- Öğrenme Vektör Nicelendirmesi
- Özdüzenlemeli Harita Ağı

Danışmansız öğrenmede ağ çıkış verileriyle değil, girdi verileriyle çalışır. Danışmansız öğrenme için belli başlı örneklerden birisi Kohonen ağıdır. Kohonen ağında, giriş katmanına ek olarak, birbiriyle topolojik olarak ilişkili nöronlardan oluşan tek bir çıkış katmanı vardır. Her bir giriş, çıkış katmanındaki her bir nörona bağlıdır. Ağ rastgele ağırlıklarla çalışmaya başlar. Her hangi bir giriş uygulandığında, giriş vektörüne öklid uzaklığı en az olan çıkış nöronu seçilir ve bu nörona gelen bağlantı, giriş ağırlıkları giriş vektörüne yaklaşacak şekilde yenilenir. Kazanan olarak adlandırılan bu nöronla birlikte, onun topolojik komşuluğunda bulunan belli sayıda nörona karşılık gelen ağırlıkları da benzer şekilde değiştirir (Elmas, 2003).

Kazanan nöronun ne büyüklükte bir komşuluktaki nöronları etkileyeceği zaman içinde değişiklik gösterir. Komşuluk başlangıçta büyük tutulup zaman içinde azaltılır. Bu şekilde, giriş vektörlerine tek nöron değil, nöron öbeklerinin tepki vereceği şekilde bir ön örnekleme sağlanmış olur (Elmas, 2003).

Danışmansız öğrenme gelecek için büyük umutlar vadetmektedir. Bu yöntem sayesinde gelecekte bilgisayarların kendi kendilerine gerçek robotik hareketler öğrenebilecekleri öne sürülmektedir (Elmas, 2003).

4.5.2.1. Hebb Öğrenme

Donald Hebb tarafından önerilmiş olan bu öğrenim kuralı en iyi bilinen kuraldır. Donald Hebb 1949 yılında kendi yazmış olduğu "Davranış organizasyonu" adlı kitabında bu kuralı açıklamıştır. Hebb konuyu nörobiyolojik açıdan ele almış ve teorisinde birbiriyle bağlantılı iki nöron aynı anda aktif olduğunda aradaki bağlantıya ait ağırlık değerlerinin arttırılmasını öngörmüştür. Bu teori, sadece aynı anda aktif olan iki nöronu göz önünde bulundurduğu için daha sonra bazı araştırmacılar tarafından iki aşamalı bir teori haline getirilmiştir.

1. Bir sinirsel bağlantının iki tarafındaki nöronlar aynı anda aktif olursa, aradaki bağlantının kuvveti artar.

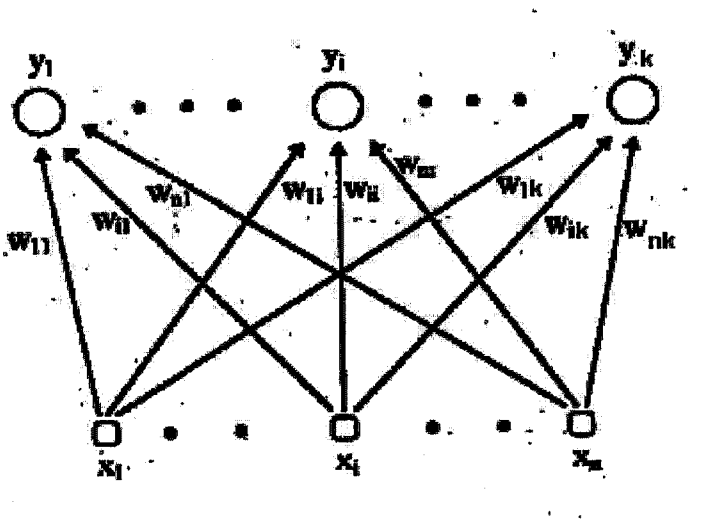
2. İki nöron farklı zamanlarda aktif oluyorsa aradaki bağlantının kuvveti azalır.

Hebb, sinir faaliyetlerini örnek alarak bunların hafizada basit bir yerde yerleşebileceğini varsaymış ve bu kurama göre sinirlerin birbirlerini ortaklaşa uyardıklarını ve bu uyarım neticesinde aralarındaki sinaptik bağlantı kuvvetinin (ağırlıklarının) kendi etkinlikleri çarpımı oranında artacağını ortaya koymuştur (Elmas, 2003).

4.5.2.2. Yarışmacı Öğrenme

Yarışmacı öğrenmenin temeli kazanan her şeyi alır stratejisine dayanır. Kazananın belirlenmesi yaklaşımındaki farklılıklara göre yarışmacı öğrenme algoritmaları farklılık göstermektedir. Yarışmacı öğrenme algoritmalarında kazanan ağırlık vektörünü belirlemek için iki farklı yaklaşım kullanılır. İlk yaklaşımda, seçilen veri vektörü ile tüm ağırlık vektörleri arasındaki uzaklıklar hesaplanır ve en az uzaklığa sahip ağırlık vektörüne karşılık gelen çıktı nöronu kazanan olarak belirlenir.

n adet *p* boyutlu girdi vektörünü *m* adet sınıfa ayıran yarışmacı ağ mimarisi Şekil 4.10 da gösterilmiştir.



Şekil 4.12 : Yarışmacı Ağ Yapısı

Kazanan çıktı nöronu j ile bu nörona ait ağırlık vektörü W_j ile gösterilirse

$\Delta W_j = \eta(X - W_j)$ şeklinde ifade edilen

ΔW_j, ağırlık düzenleme değeri ve η, öğrenme oranı ve $X = (x_1,...,x_i,...,x_n)$ giriş nöronlarına karşılık gelen verilerdir.

Her bir j çıktı nöronu için ağırlık düzenleme değeri şu şekilde tanımlanır.

$\Delta W_j = \eta O_j(X - W_j)$

O_j, sinirin aktivasyon düzeyini belirtir.

j. sinir kazanan sinir olmadığı durumda O_j=0, j. sinir kazanan sinir ise O_j=1 olarak alınır (Fu, 1994).

Eğitimin kalitesi, hata kareler kriteri olarak bilinen aşağıdaki eşitlikten hesaplanır.

$$E = \sum_p \|W_w - X_p\|^2$$

j kazanan nöron, X_p girdi vektörü ve W_j, kazanan nöron j'ye ait ağırlık vektörüdür. Yarışmacı öğrenme algoritması aslında (gradient descent) eğimdeki ani düşmeleri bu hata kareleri aracılığıyla minimum yapmaktır.

Her bir X_p girdisi için, kriter fonksiyon

$$E_p = \frac{1}{2}\sum_i (W_{ji} - X_{p,i})^2$$

şeklinde hesaplanır.

W_{ij}, *i.* birim ile *j.* birim arasındaki ağırlık

$$\Delta W_{ji} = - \eta \frac{\partial E_p}{\partial W_{ji}}$$

biçiminde hesaplanır. Eğer *j.* birimin kazanan nöronu *w* ise kısmı türevi

$$\frac{\partial E_p}{\partial W_{ji}} = W_{wi} - X_{p,i}$$

şeklindedir. Aksi taktirde, sadece türev 0 olacaktır.

$$\Delta W_{wi} = \eta (X_{p,i} - W_{wi})$$

Yarışmacı öğrenme algoritmalarında kazanan çıktı nöronunu belirlemek için kullanılan iki yaklaşım vardır. İlk yaklaşımda ağırlık vektörleri ile rastgele seçilen *x* vektörü arasındaki açı hesaplanır ve *x* vektörü ile en küçük açıyı yapan ağırlık vektörü kazanan çıktı nöronu olarak belirlenir. İkinci yaklaşımda ise tüm ağırlık vektörleri ile *x* vektörü arasındaki uzaklıklar hesaplanır. *x* vektörü ile arasındaki uzaklık en küçük olan ağırlık vektörü kazanan olarak belirlenir. Kazanan nöron minimum açı sonucunda belirleniyorsa birim vektöre dönüştürülmüş verilerden yararlanılır.

Yarışmacı Öğrenme Algoritması

Adım 1. Sınıf sayısının belirlenmesi

Adım 2. Öğrenme oranının belirlenmesi

Adım 3. Durdurma kriterinin belirlenmesi

Adım 4. Başlangıç ağırlık vektörlerinin değerlerinin belirlenmesi

Adım 5. Girdi verileri arasından rastgele bir *x* vektörü seçilir.

Adım 6. Seçilen *x* vektörü ile ağırlık vektörleri (W_k) arasında işlem yaparak kazanan ağırlık vektörü belirlenir.

Adım 7. Kazanan vektörü rastgele seçilen *x* vektöründen çıkar ve sonucu öğrenme oranı (η) ile çarparak ΔW hesaplanır.

$\Delta W = \eta (x - W_{eski})$

Adım 8. Kazanan ağırlık vektörüne ΔW'i ekleyerek kazanan ağırlık vektörünü güncelle.

$$W_{yeni} = \Delta W + W_{eski}$$

Adım 9. Durdurma kriterini sorgula.

- Durdurma kriteri gerçekleşmemiş ise 5-8 adım arasındaki işlemleri tekrarla.
- Durdurma kriteri gerçekleşmiş ise eğitme işlemini sonlandır.

4.5.2.3. Kohonen Öğrenme

Kohonen öğrenme kuralı Teuvo Kohonen tarafından geliştirilmiştir. Kohonen Özdüzenlemeli Özellik Haritaları olarak da adlandırılan bu yöntemde nöronlar öğrenme fırsatı yakalamak veya kendi ağırlıklarını güncellemek için yarışırlar.

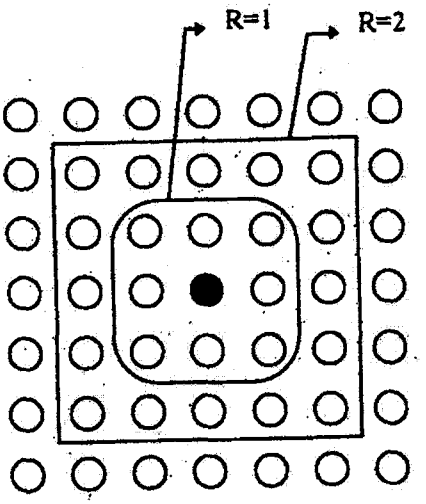
Özdüzenlemeli özellik haritaları, insan beynindeki farklı işlevleri yerine getiren işlevsel bölgelerden yola çıkan topolojik bir YSA modelidir. Benzer bilgileri tutan sinirler birbirlerine fiziksel olarak da yakın konumlanmaktadır. Başka bir deyişle herhangi bir ağırlık değerinde yapılan güncelleme işlemi, bu ağırlık değerine komşu olan nöronlar üzerinde de belli oranda etkili olmaktadır.

Kohonen mimarisi üzerinde her bir sınıfa ait ağırlık vektörü o sınıfın ve o sınıfı temsil eden çekirdek noktaların özelliklerini barındırır. Kullanılan "özdüzenlemeli" kavramı, sistemin danışmansız öğrenme yöntemiyle eğitildiğini gösterir. Sisteme uygulanan girdiler hangi sınıfa en yakın ise o sınıfa dahil edilir ve o sınıfa ait ağırlık vektörü üzerinde güncelleme yapılır. Girdilerin yakınlığını belirlemek için benzerlik ve uzaklık ölçütleri kullanılır.

Tek boyutlu bir dizinde güncelleme yapılacak komşular R kadar uzaklıkta ise, J sınıfında yapılacak güncelleme

$$max(1,J-R) \leq j \leq min(J+R,m)$$

aralığındaki bütün sınıfların ağırlık değeri üzerinde de güncellemeye neden olur. Aşağıdaki şekilde farklı R değerleri için iki boyutlu Kohonen ağ mimarisi gösterilmektedir.



Şekil 4.13: İki Boyutlu Mimaride Komşuluklar

Kohonen mimarilerinde uygulanan öğrenme algoritması aşağıdaki gibidir.

Adım 1. Ağırlıklara ilk değerlerin verilmesi (w_{ij})

Komşuluk uzaklığının belirlenmesi (R)

Öğrenme katsayısının belirlenmesi (α)

Adım 2. Durdurma kriterine kadar tekrarla

Adım 3. Her x_i girdi vektörü için 4 – 6 adımları tekrarla

Adım 4. Her j değeri için uzaklık hesapla

$$D(j) = \sum (w_{ij}-x_i)^2$$

Adım 5. $D(j)$ minimum olan j değerini bul

Adım 6. j'nin tanımlı komşuluğunda bulunan ağırlık vektörlerini güncelle

$$w_{ij}(yeni) = w_{ij}(eski) + \alpha\,(x_i - w_{ij}(eski))$$

Adım 7. Öğrenme katsayısını güncelle

Adım 8. Belli aralıklarla komşuluk uzaklığını azalt

Adım 9. Durdurma kriterini kontrol et (Yücetürk, 1999).

Sisteme gelen örnek girdilerin sayısı arttıkça girdilerin etkileyeceği komşuların sayısınında azalması beklenmektedir. Bu da algoritmada verildiği gibi komşuluk uzaklığı R'nin belli aralıklarla azaltılması ile sağlanabilir.

5. UYARLAMALI YARIŞMACI ÖĞRENME ALGORİTMASI VE SINIFLANDIRMAYA UYGULANMASI

Yarışmacı Öğrenme Algoritmasında (YÖA) başlangıç ağırlık vektörlerinin rastgele belirlenmesi, kümeleme sonuçlarını etkileyen önemli bir faktördür. Başlangıç vektörlerinin seçiminden kaynaklanabilecek bir olumsuzluğu ortadan kaldırmak amacıyla uyarlamalı YÖA önerilmiştir. Bu yaklaşımda optimum küme sayısından çok daha fazla sayıda başlangıç ağırlık vektörü belirlenir ve önceden belirlenen kriter ışığında istenilen küme sayısına ulaşana kadar ağırlık vektörlerinin bir kısmı elenir.

R^p boyutlu vektör uzayında, n eleman sayısı olmak üzere x_i vektörü

$$x_i = (x_{i1},...,x_{ip}) \qquad i=1, ...,n \tag{5.1}$$

biçiminde tanımlanır.

Her bir x_i vektörü girdi nöronlarına uygulanır. Ağ, p tane girdi nöronu, sınıf sayısı kadar çıktı nöronu (y_k) ve her bir girdi nöronunu her bir çıktı nöronu ile ilişkilendiren ağırlık vektörlerinden (w_{ik}) meydana gelir. Ağırlık vektörlerinin başlangıç değerlerinin belirlenmesi konusunda iki farklı yaklaşım bulunmaktadır. Ağırlık vektörleri R^p boyutlu vektör uzayında rastgele sayılar olarak üretilebileceği gibi x_{ij} veri kümesinden rastgele seçilen x_i vektörleri de ağırlık vektörlerinin başlangıç değeri olarak belirlenebilir. Ağırlık vektörlerinin başlangıç değerleri belirlendikten sonra benzerlik kriterlerine göre eğitim işlemine geçilir.

Yarışmacı öğrenme algoritması ile eğitilen danışmansız sinir ağlarının eğitiminin iki yaklaşımla gerçekleştirildiğinden bahsedilmişti. Bu yaklaşımlar ağırlık vektörleri ile x_i vektörünün benzerliğini belirleme noktasında farklılık gösterir; vektörler arasındaki açıya göre ve ya vektörler arasındaki uzaklığa göre benzerlik belirlenebilir. Farklı benzerlikler kullanılarak geliştirilen algoritmalar aşağıda açıklanmıştır.

5.1. Açıya Dayalı Uyarlamalı Yarışmacı Öğrenme

Benzerliğin belirlenmesinde kullanılan yaklaşımlardan biri vektörler arasındaki açıya dayalı benzerliktir. x_i vektörü ile w_k ağırlık vektörleri arasındaki açıya dayanan benzerlik yaklaşımıdır. Bu yaklaşımda kazanan ağırlık vektörü

$$k_w = \arg\max_k \{x_i.w_k\} \tag{5.2}$$

biçiminde belirlenir.

Kazanan ağırlık vektörünü güncelleme değeri

$$\Delta w_k = \eta(\frac{x_i}{p} - w_k) \tag{5.3}$$

denkleminden hesaplanır ve ağırlık vektörünün yeni değeri

$$w_{k\_yeni} = w_{k\_eski} + \Delta w_k \tag{5.4}$$

biçiminde belirlenir. Bir t anında yalnızca tek bir ağırlık vektörü eğitilir. Eğitme işlemini sonlandırmak için farklı sonlandırma kriterleri kullanılmaktadır. Bazı uygulamalarda eğitme işlemi Δw değerinin ε olarak belirlenen bir hata değerinden daha küçük oluncaya kadar eğitim sürer. Bu uygulamada maksimum döngü sayısı kullanılmıştır.

Eğitim işlemi tamamlandıktan sonra başlangıçta çok sayıda belirlenen ağırlık vektörlerinin bir kısmının elenmesi gerekir. Hangi ağırlık vektörlerinin elenmesi gerektiğine karar vermek için ilk eğitme işlemi sonunda bir sınıflandırma yapılır ve her sınıf için

$$D_k = \frac{1}{p}\sum_{x \in S_k} x.w_k \tag{5.5}$$

değeri hesaplanır ve D_k değerleri sıralanır ve bu sıralama sonucunda D_k değerleri küçük olan sınıfları temsil eden ağırlık vektörlerinden q tanesi elenir. Geriye kalan ağırlık vektörleri ile eğitme işlemi tekrarlanır. Eleme işlemi iki şekilde gerçekleştirilir. Her defasında bir ağırlık vektörü elenir veya birden fazla ağırlık vektörünün elenmesiyle kaba bir ayar yapılır daha sonra hassas eleme olarak adlandırılan birer birer elemeye geçilir. Bunun için

$$k(t) - L < q \qquad (5.6)$$

kriteri kullanılır. t anında k ile L arasındaki fark q değerinden küçükse ince ayar olan birer birer elemeye geçilir.

Açıya Dayalı Uyarlamalı YÖA

Adım 1 Başlangıç :

Eleman sayısı n, başlangıç küme sayısı $k(0)=l$, optimum küme sayısı L, öğrenme oranı η, Ağırlık vektörlerinin başlangıç değerleri $\{w_1(0),...,w_k(0),...,w_l(0)\}$, döngü sayısı t ve maksimum döngü sayısının T_{max}, elenecek vektör sayısı q'u belirle.

Adım 2 Öğrenme :

(2.1) Girdi vektörlerinden $\{x_1,...,x_i,...,x_n\}$ rastgele x_i vektörü seç.

(2.2) $k_w = \arg\max_k \{x_i.w_k\}$ denklemini kullanarak kazanan vektörü belirle.

(2.3) $\Delta w_k = \eta(\frac{x_i}{p} - w_k)$ kullanarak ağırlık güncelleme değerini belirle.

(2.4) $w_{k\_yeni} = w_{k\_eski} + \Delta w_k$ kullanarak kazanan ağırlık vektörünü güncelle.

(2.5) Döngü değerini bir arttır. $t \leftarrow t+1$

(2.6) Eğer $t = T_{\max}$ ise Adım 3'e git, değilse (2.1)'e git.

Adım 3 Eleme :

(3.1) Eğer $k \le L$ ise Adım 4'e git, değilse (3.2)'e git.

(3.2) Eğer $k - L < q$ ise (3.5)'e git, değilse (3.3)'e git.

(3.3) $D_k = \frac{1}{p}\sum_{x \in S_k} x.w_k$ denklemine göre D_k değer sıralamasında minimum olan ilk q vektörü ele.

(3.4) Küme sayısını q kadar azalt ($k \leftarrow k - q$) ve Adım 2'e git.

(3.5) $D_k = \frac{1}{p}\sum_{x \in S_k} x.w_k$ denklemine göre minimum D_k'e sahip olan w_k ağırlık vektörünü ele.

(3.6) Küme sayısını 1 azalt ($k \leftarrow k - 1$) ve Adım 2'e git.

Adım 4 Sınıflandır :

$k_w = \arg\max_k \{x_i.w_k\}$ denklemini kullanarak tüm x_i'leri sınıflandır.

5.2 Uzaklığa Dayalı Uyarlamalı Yarışmacı Öğrenme

İknici yaklaşım olan uzaklığa dayalı uyarlamalı YÖA'da, ağırlık vektörlerinin başlangıç değerleri belirlendikten sonra x_i girdi vektörü ile tüm ağırlık vektörleri arasındaki uzaklıklar hesaplanır ve

$$k_w = \arg\min_k \{\|x - w_k\|\} \tag{5.7}$$

x_i vektörü ile arasındaki uzaklık (k_w) minimum olan ağırlık vektörü kazanan olarak belirlenir. Yarışmacı öğrenme yaklaşımında yalnızca kazanan ağırlık vektörünün değeri güncellenir ve bakılan anda sadece bir ağırlık vektörü eğitilir.

Kazanan ağırlık vektörünü güncelleme değeri

$$\Delta w_k = \eta(x_i - w_k) \tag{5.8}$$

biçiminde hesaplanır ve ağırlık vektörünün yeni değeri

$$w_{k\_yeni} = w_{k\_eski} + \Delta w_k \tag{5.9}$$

denkleminden belirlenir.

Hangi ağırlık vektörlerinin elenmesi gerektiğine karar vermek için ilk eğitme işlemi sonunda bir sınıflandırma yapılır ve her k sınıf için D_k

$$D_k = \frac{1}{p}\sum_{x \in S_k} d(x, w_k) \tag{5.10}$$

biçiminde hesaplanır.

Eleme işlemleri sonlandırılıp istenen küme sayısına ulaşıldığında son bir eğitme işlemi daha gerçekleştirilir ve verilerin kendilerine en yakın olan ağırlık vektörlerinin temsil ettiği sınıflara atanmasıyla sınıflandırma işlemi gerçekleştirilir. Son olarak da yapılan sınıflandırmanın hata kareler ortalaması

$$E = \frac{1}{n}\sum_{k=1}^{L} D_k \tag{5.9}$$

biçiminde hesaplanır. Hata kareler ortalaması faklı yöntemler kullanılarak yapılan sınıflandırma sonuçlarını karşılaştırmak amacıyla kullanılabilir.

Uzaklığa Dayalı Uyarlamalı YÖA

Adım 1 Başlangıç :

Eleman sayısı n, başlangıç küme sayısı $k(0)=l$, optimum küme sayısı L, öğrenme oranı η, Ağırlık vektörlerinin başlangıç değerleri $\{w_1(0),...,w_k(0),...,w_l(0)\}$, döngü sayısı t ve maksimum döngü sayısının T_{max}, elenecek vektör sayısı q'u belirle.

Adım 2 Öğrenme :

(2.1) Girdi vektörlerinden $\{x_1,...,x_i,...,x_n\}$ rastgele x_i vektörü seç.

(2.2) $k_w = \arg\min_k \{\|x - w_k\|\}$ denklemini kullanarak kazanan vektörü belirle.

(2.3) $\Delta w_k = \eta(x_i - w_k)$ kullanarak ağırlık güncelleme değerini belirle.

(2.4) $w_{k\_yeni} = w_{k\_eski} + \Delta w_k$ kullanarak kazanan ağırlık vektörünü güncelle.

(2.5) Döngü değerini bir arttır. $t \leftarrow t+1$

(2.6) Eğer $t = T_{\max}$ ise Adım 3'e git, değilse (2.1)'e git.

Adım 3 Eleme :

(3.1) Eğer $k \le L$ ise Adım 4'e git, değilse (3.2)'e git.

(3.2) Eğer $k - L < q$ ise (3.5)'e git, değilse (3.3)'e git.

(3.3) $D_k = \frac{1}{p}\sum_{x \in S_k} d(x, w_k)$ denklemine göre D_k değer sıralamasında minimum olan ilk q vektörü ele.

(3.4) Küme sayısını q kadar azalt ($k \leftarrow k - q$) ve Adım 2'e git.

(3.5) $D_k = \frac{1}{p}\sum_{x \in S_k} d(x, w_k)$ denklemine göre maksimum D_k'e sahip olan w_k ağırlık vektörünü ele.

(3.6) Küme sayısını 1 azalt ($k \leftarrow k - 1$) ve Adım 2'e git.

Adım 4 Sınıflandır :

$k_w = \arg\min_k \{\|x - w_k\|\}$ denklemini kullanarak tüm x_i'leri sınıflandır.

Yukarıda verilen iki algoritma kullanılarak yapılan sınıflandırma işlemi sonucunda (5.11) denklemi kulanılarak her iki algoritma için sınıflandırma hataları hesaplanır. Sınıflandırma hataları, sınıflandırma sonuçlarının karşılıklı olarak değerlendirilmesi sonuçların tutarlılığının belirlenmesinde kullanılır.

6. İLLERİN SAĞLIK BAZINDA GELİŞMİŞLİK SINIFLARININ BELİRLENMESİ

Bu çalışmada, geliştirilen uyarlamalı YÖA'nın Türkiye'deki illerin sağlık bazında gelişmişlik gruplarının belirlenmesine uygulanması ele alınmıştır. Bu amaçla, danışmansız öğrenme algoritmalarından biri olan uyarlamalı YÖA geliştirilmiş ve uygulama olarak Türkiye'de 1999 yılı itibariyle bulunan 81 ilin sağlık verileri kullanılmıştır. Veriler, Devlet İstatistik Enstitüsü (DİE) yayınlarından 1999 ve 2000 nüfus sayım kitaplarından elde edilmiştir. Değişken değerleri kişi başına düşen değerler olarak verileceği için nüfus sayım sonuçlarından yararlanılmıştır. Toplam sayısı 10 olan değişkenler şu şekilde tanımlanmıştır.

X1: Kişi Başına Düşen Uzman Hekim Sayısı

X2: Kişi Başına Düşen Pratisyen Hekim Sayısı

X3: Kişi Başına Düşen Diş Hekimi Sayısı

X4: Kişi Başına Düşen Hemşire Sayısı

X5: Kişi Başına Düşen Sağlık Memuru Sayısı

X6: Kişi Başına Düşen Ebe Sayısı

X7: Kişi Başına Düşen Hastane Sayısı

X8: Kişi Başına Düşen Yatak Sayısı

X9: Kişi Başına Düşen Eczane Sayısı

X10: Kişi Başına Düşen Eczacı Sayısı

İllerin değişkenlere ilişkin değerleri Ek Tablo 1'de verilmiştir. Ek Tablo 1'deki değerler kullanılarak uyarlamalı YÖA ile illerin sınıflandırılması için, öncelikle optimum küme sayısını belirlemek gerekmektedir. Optimum küme sayısını belirlemek için (3.28) denklemi ile belirlenen Mariot katsayısı kullanılmıştır. Farklı kümeler için bu değerler, uyarlamalı YÖA kullanılarak yapılan sınıflandırma sonuçlarından hesaplanmıştır.

Tablo 6.1: Mariot Katsayıları

Sınıf sayısı	Mariot Katsayısı
2	21.10^{-8}
3	$\mathbf{19.10^{-9}}$
4	13.10^{-8}
5	20.10^{-8}
6	11.10^{-8}
7	33.10^{-9}

Mariot yöntemine göre en küçük Mariot katsayısını veren küme sayısının en uygun kümeleme olduğu kabul edilir. Tablo 6.1'de görüldüğü gibi en küçük Mariot katsayısına karşılık gelen 3, uyarlamalı YÖA'nın illerin sınıflandırılmasına uygulanmasında en uygun küme sayısı olarak belirlenmiştir. Yapılan denemeler sonucunda öğrenme oranı $\eta = 0.03$ ve durdurma kriteri maksimum döngü sayısı $T_{\max} = 10000$ olarak belirlenmiş ve bu değerler önerilen algoritmada kullanılarak sınıflandırma işlemi gerçekleştirilmiştir. Borlan Delphi 7 kullanılarak önerilen algoritmalarla sınıfladırma yapan program geliştirilmiştir.

6.1. Açıya Dayalı Uyarlamalı YÖA ile Gelişmişlik Sınıflarının Belirlenmesi

Ek Tablo 2'de verilmiş olan ve birim vektöre dönüştürülerek standartlaştırılan verilere ilişkin YSA'nın Bölüm 5.1'de verilen algoritma ile eğitilmesi sonucunda elde edilen ağırlık vektörleri Tablo 6.2'de verilmiştir.

Tablo 6.2 : Açıya Dayanan Uyarlamalı YÖA ile Eğitilen Ağın Ağırlık Vektörleri

	W1	W2	W3
X1	0.11867	0.09486	0.15062
X2	0.30856	0.22068	0.21708
X3	0.05317	0.04092	0.05028
X4	0.49880	0.39475	0.33441
X5	0.32607	0.30958	0.21737
X6	0.39183	0.33449	0.21477
X7	0.01053	0.00959	0.00599
X8	0.59969	0.74802	0.83814
X9	0.08770	0.08638	0.09450
X10	0.10858	0.09995	0.10498

YSA'nın eğitimi tamamlandıktan sonra verilerin sınıflandırılması aşamasına geçilir. Bunun için tüm veriler birer birer küme merkezlerini temsil eden tüm ağırlık vektörleri ile eğitimde olduğu gibi karşılaştırılır ve en küçük yakınlığa göre kümelere atanarak sınıflar oluşturulur. Açıya dayalı uyarlamalı YÖA kullanılarak yapılan sınıflandırma sonuçları Tablo 6.3'de verilmiştir.

Tablo 6.3 : Açıya Dayalı Uyarlamalı YÖA ile Sınıflandırma Sonuçları

Sınıf 1	**Sınıf 2**	**Sınıf 3**
Ağrı	Adıyaman	Adana
Antalya	Amasya	Afyon
Aydın	Artvin	Ankara
Bilecik	Balıkesir	Bolu
Çanakkale	Bingöl	Bursa
Denizli	Bitlis	Çorum
Edirne	Burdur	Diyarbakır
Hakkari	Çankırı	Elazığ
Kars	Giresun	Erzincan
Malatya	Hatay	Erzurum
Kahramanmaraş	İçel	Eskişehir
Mardin	Kırklareli	Gaziantep
Muğla	Kırşehir	Gümüşhane
Tunceli	Kütahya	Isparta
Uşak	Muş	İstanbul
Bayburt	Nevşehir	İzmir

Karaman	Niğde	Kastamonu
Batman	Ordu	Kayseri
Şırnak	Siirt	Kocaeli
Ardahan	Sinop	Konya
Iğdır	Tekirdağ	Manisa
Yalova	Tokat	Rize
Osmaniye	Yozgat	Sakarya
	Aksaray	Samsun
	Kırıkkale	Sivas
	Bartın	Trabzon
	Kilis	Şanlıurfa
		Van
		Zonguldak
		Karabük
		Düzce

Sınıflandırma sonucunda üç farklı gelişmişlik grubu belirlenmiş ve sınıfların gelişmişlik düzeyleri birbiri ile karşılaştırılıp onlara "Çok Gelişmiş", "Orta Derecede Gelişmiş" ve "Az Gelişmiş" şeklinde birer etiket verilmiştir. Sınıfların gelişmişlik düzeyleri, sınıfların merkezi değerleri olan ağırlık vektörlerinin uzunluklarıyla doğru orantılıdır. Açıya dayalı uyarlamalı YÖA, birim vektöre dönüştürülerek standartlaştırılan veriler üzerine uygulandığı ve eğitim sonucunda da birim vektör olarak elde edildiği için birbiri ile karşılaştırılmaları bir anlam ifade etmez. Bu sorunu ortadan kaldırabilmek için eğitim sonucunda elde edilen ağırlık vektörleri, Bölüm 5.2'de (uzaklığa dayalı uyarlamalı YÖA) kullanılan standartlaştırma biçimine uygun hale getirilmesi Tablo6.2'deki ağırlık vektörlerinin her birine en yakın olan s tane x_i vektörü belirlenmiştir. Bu x_i vektörlerinin sözü edilen biçimde standartlaştırılımış değerlerinin ortalaması alınarak ağırlık vektörlerinin değerleri bu biçimde gösterilir ve değerler Tablo 6.4'de verilmiştir.

Tablo 6.4 : Açıya Dayalı Uyarlamalı YÖA Ağırlık Vektörlerinin Uzunlukları

	Sınıf 1	**Sınıf 2**	**Sınıf 3**
Uzunluklar	0.9087	1.1061	1.4127

Elde edilen ağırlık vektörlerinin uzunlukları büyükten küçüğe sıralanır ve bu sıralamaya göre sınıflar "Çok Gelişmiş", "Orta Derece Gelişmiş" ve "Az Gelişmiş" olarak etiketlenmiştir.

Geliştirilen algoritma ile yapılan sınıflandırmanın turarlılığını uzaklığa dayalı uyarlamalı YÖA ve k-Ortalamalar algoritmaları ile yapılan sınıflandırma sonuçları ile karşılaştırmalı olarak kanıtlayacağız. Bunun için aşağıda bu algoritmalar ile sınıflandırma ve sınıflandırmaların sonuçları ele alınmıştır.

6.2. Uzaklığa Dayalı Uyarlamalı YÖA ile Gelişmişilik Sınıflarının Belirlenmesi

Ek Tablo 3'de verilmiş olan ve değişkenlerin her birinin maksimum 1 olacak şekilde standartlaştırılmış verilere ilişkin YSA'nın Bölüm 5.2'de verilen algoritma ile eğitilmesi sonucunda elde edilen ağırlık vektörleri Tablo 6.5'de verilmiştir.

Tablo 6.5 : Uzaklığa Dayalı Uyarlamalı YÖA ile Eğitilen Ağın Ağırlık Vektörleri

	W1	**W2**	**W3**
X1	0.8022	0.1541	0.2585
X2	0.7869	0.3160	0.4498
X3	0.8535	0.1661	0.2871
X4	0.7643	0.3463	0.5998
X5	0.6530	0.4213	0.7397
X6	0.2525	0.2638	0.4927
X7	0.1106	0.1988	0.3646
X8	0.5802	0.2697	0.4822
X9	0.7365	0.2835	0.4772
X10	0.8556	0.3101	0.4621

Uzaklığa dayalı uyarlamalı YÖA ile YSA'nın eğitimi tamamlandıktan sonra verilerin sınıflandırılması aşamasına geçilir. Bunun için tüm veriler birer birer Tablo 6.5'de verilen tüm ağırlık vektörleri ile eğitimde olduğu gibi karşılaştırılır ve en küçük yakınlığa göre kümelere atanarak sınıflar oluşturulur. Uzaklığa dayalı adaptif YÖA kullanılarak yapılan sınıflandırmanın sonuçları Tablo 6.6'da verilmiştir.

Tablo 6.6 : Uzaklığa Dayalı Uyarlamalı YÖA ile Sınıflandırma Sonuçları

Sınıf 1	**Sınıf 2**	**Sınıf 3**
Ankara	Adana	Amasya
İstanbul	Adıyaman	Antalya
İzmir	Afyon	Artvin
	Ağrı	Aydın
	Bilecik	Balıkesir
	Bingöl	Bolu
	Bitlis	Burdur
	Bursa	Çanakkale
	Çankırı	Denizli
	Çorum	Edirne
	Diyarbakır	Elazığ
	Erzincan	Eskişehir
	Erzurum	Giresun
	Gaziantep	Isparta
	Gümüşhane	Kastamonu
	Hakkari	Kayseri
	Hatay	Kırklareli
	İçel	Kırşehir
	Kars	Muğla
	Kocaeli	Nevşehir
	Konya	Samsun
	Kütahya	Sinop
	Malatya	Trabzon
	Manisa	Tunceli
	Kahramanmaraş	Uşak
	Mardin	
	Muş	
	Niğde	
	Ordu	
	Rize	
	Sakarya	
	Siirt	
	Sivas	
	Tekirdağ	
	Tokat	
	Şanlıurfa	
	Van	
	Yozgat	
	Zonguldak	
	Aksaray	
	Bayburt	
	Karaman	
	Kırıkkale	
	Batman	
	Şırnak	
	Bartın	

	Ardahan Iğdır Yalova Karabük Kilis Osmaniye Düzce	

Tablo 6.5'de verilen ağırlık vektörlerinin uzunlukları Tablo 6.7'de verilmiştir. Bu uzunluk değerleri büyükten küçüğe doğru sıralanarak sınıflar "Çok Gelişmiş", "Orta Derecede Gelişmiş" ve "Az Gelişmiş" olarak etiketlenmiştir.

Tablo 6.7 : Uzaklığa Dayalı Uyarlamalı YÖA Ağırlık Vektörlerinin Uzunlukları

	Sınıf 1	**Sınıf 2**	**Sınıf 3**
Uzunluklar	2.1654	0.8983	1.5192

6.3. k-Ortalamalar Yöntemi ile Gelişmişlik Sınıflarının Belirlenmesi

SPSS paket programı kullanılarak Ek Tablo 3'de verilmiş olan ve bileşenler bazında ayrı ayrı standartlaştırılmış verilerin Bölüm 3.3.1'de verilen k-Ortalamalar yöntemi ile kümelenmesi sonucunda elde edilen küme ortalama değerleri Tablo 6.8'de verilmiştir.

Tablo 6.8: k-Ortalamalar Yönteminde Küme Merkezleri.

	Kümeler		
	C1	C2	C3
X1	,619	,151	,254
X2	,648	,310	,446
X3	,610	,161	,290
X4	,760	,349	,602
X5	,644	,437	,725
X6	,322	,273	,515
X7	,227	,207	,362
X8	,645	,281	,452
X9	,771	,283	,475
X10	,847	,296	,457

İllerin sınıflandırılması Tablo 6.8'de verilen küme merkezleri kullanılarak yapılmış ve sınıflandırma sonuçları Tablo 6.9'da verilmiştir.

Tablo 6.9 : k-Ortalamalar ile Sınıflandırma Sonuçları

Küme 1	Küme 2	Küme 3
Ankara	Adana	Amasya
Bolu	Adıyaman	Antalya
Eskişehir	Afyon	Artvin
İstanbul	Ağrı	Aydın
İzmir	Bilecik	Balıkesir
	Bingöl	Burdur
	Bitlis	Çanakkale
	Bursa	Denizli
	Çankırı	Edirne
	Çorum	Elazığ
	Diyarbakır	Giresun
	Erzincan	Isparta
	Erzurum	Kastamonu
	Gaziantep	Kayseri
	Gümüşhane	Kırklareli
	Hakkari	Kırşehir
	Hatay	Muğla
	İçel	Nevşehir
	Kars	Samsun
	Kocaeli	Sinop
	Konya	Trabzon
	Kütahya	Tunceli
	Malatya	Uşak
	Manisa	
	Kahramanmaraş	
	Mardin	
	Muş	
	Niğde	
	Ordu	
	Rize	
	Sakarya	
	Siirt	
	Sivas	
	Tekirdağ	
	Tokat	
	Şanlıurfa	
	Van	
	Yozgat	
	Zonguldak	
	Aksaray	
	Bayburt	
	Karaman	
	Kırıkkale	
	Batman	
	Şırnak	

	Bartın Ardahan Iğdır Yalova Karabük Kilis Osmaniye Düzce	

Sınıfların gelişmişlik düzeylerini belirlemek için Tablo 6.8'de verilen küme ortalama değerlerinin uzunlukları hesaplanmış ve sonuçlar Tablo 6.10'da verilmiştir.

Tablo 6.10 : k-Ortalamalar Küme Merkezi Değerlerinin Uzunlukları

	Sınıf 1	**Sınıf 2**	**Sınıf 3**
Uzunluklar	2.0126	0.9064	1.5069

Tablo 6.10'da verilen sınıf ortalama değerinin uzunlukları karşılaştırılarak sınıflar "Çok Gelişmiş", "Orta Derecede Gelişmiş" ve "Az Gelişmiş" olarak etiketlenmiştir.

Yukarıda sonuçları ayrıntılı bir şekilde verilmiş olan her üç algoritma ile yapılan sınıflandırmaları birbiri ile karşılaştırmak amacıyla (5.11) denklemi kullanılarak sınıflandırma hataları hesaplanmış ve sonuçlar Tablo 6.11'de verilmiştir.

Tablo 6.11 : Sınıflandırma Hataları

	k-Ortalamalar	Açı	Uzaklık
Sınıflandırma Hatası	1.154293	0.879318	1.136285

Tablo 6.11'de verilen sonuçlar incelendiğinde açıya dayalı uyarlamalı YÖA ile yapılan sınıflandırmanın diğerlerine göre daha küçük bir hata sahip olduğu görülmüştür. Böylece önerilen açıya dayalı uyarlamalı YÖA ile yapılan sınıflandırmanın daha tutarlı sonuçlar verdiği ispatlanmıştır.

7. TARTIŞMA VE SONUÇLAR

Bu çalışmada istatistiksel ve kümeleme analizine dayanan sınıflandırma yöntemleri incelenmiş ve YSA'ların avantajları göz önünde bulundurularak sınıflandırmanın bu model kullanılarak yapılması üzerinde durulmuştur. Veri sınıflandırmada kullanılan danışmansız öğrenme yaklaşımına dayanan farklı YSA modelleri ve bu modellere uygulanan öğrenme algoritmaları incelenmiştir. Çok değişkenli verilerin sınıflandırılmasında yaygın olarak kullanılan YÖA temel alınmış ve detaylı bir şekilde incelenmiştir.

YÖA'larda küme merkezlerine karşılık gelen ağırlık vektörlerinin başlangıç değerleri iki şekilde belirlenebilir; R^p boyutlu uzayda rastgele sayılar üreterek, veri seti içinden rastgele veri noktaları seçerek. R^p boyutlu uzayda rastgele sayılar üreterek ağırlık vektörlerine başlangıç değerleri verildiğinde çok daha geniş bir alan dikkate alınmış olur ve sınıflandırma için kullanılan veriler bu uzayın küçük bir kısmında kalabilir. Böyle bir durumda veri setindeki verilerin sınıflandırılması doğru sonuçlar vermeyebilir. Bu konuda yapılan ön çalışmalar sırasında veri setinden rastgele veri noktaları seçip onları ağırlık vektörlerinin başlangıç değeri olarak belirlemenin daha sağlıklı sonuçlar verdiği görülmüştür. Başlangıç ağırlık vektörlerinin veri setinden seçilmesi durumunda, rastgelelikten kaynaklanarak uç noktaların başlangıç ağırlık vektörü olarak seçilmesi söz konusu olabilir. Böyle bir durumun da sınıflandırma sonucunu olumsuz etkilediği gözlemlenmiştir. Bu olumsuzluğu ortadan kaldırmak amacıyla, önceden belirlenen optimum küme sayısından daha fazla sayıda başlangıç ağırlık vektörlerinin üretilmesine dayanan uyarlamalı YÖA önerilmiştir.

Uyarlamalı YÖA, çok sayıda başlangıç ağırlık vektörü üretip YSA'yı bu vektörlerle eğitip daha sonra bunların bir kısmını eleyerek en uygun ağırlık vektörlerini belirlemeyi amaçlayan bir yaklaşımdır. Eleme süreci, yayılma hatası en küçük olan sınıflara ait ağırlık vektörlerinin atılması şeklinde gerçekleştirilir. Uyarlamalı YÖA'da temel YÖA'da olduğu gibi açıya dayanan ve uzaklığa dayanan yaklaşımlar kullanılarak sınıflandırma yapılabilir. Fakat açıya dayanan YÖA'nın üstünlüğü dikkate alınarak verilerin sınıflandırılmasında açıya dayanan uyarlamalı YÖA önerilmiştir.

Bu çalışmada, geliştirilen açıya dayalı uyarlamalı YÖA, Türkiye'deki 81 ilin sağlıkla ilgili 10 göstergesi dikkate alınarak onların sınıflandırılması ve gelişmişlik düzeylerinin belirlenmesine uygulanmıştır. İllere ait göstergelerin değerleri kullanılarak illerin sağlık bazında 3 gelişmişlik grubuna ayrıldığı tespit edilmiş ve gelişmişlik düzeyleri belirlenerek uygulama sonucunda oluşan sınıflar "Çok Gelişmiş", "Orta Derecede Gelişmiş" ve "Az Gelişmiş" olarak etiketlendirilmiştir.

Açıya dayanan uyarlamalı YÖA'nın sınıflandırmaya uygulanması yapılmış ve sınıflandırma hatası, uzaklığa dayanan uyarlamalı YÖA ve k-Ortalamalar ile yapılan sınıflandırma hataları ile karşılaştırmalı olarak değerlendirilmiştir. Açıya dayalı uyarlamalı YÖA ile yapılan sınıflandırmanını sınıflandırma hatası 0.8793 iken uzaklığa dayalı uyarlamalı YÖA ile yapılan sınıflandırmanın hatası 1.1363 ve k-Ortalamalar ile yapılan sınıflandırmanın hatasının 1.1543 olduğu görülmüştür. Bu sonuçlarda açıya dayalı uyarlamalı YÖA ile sınıflandırma sonucunun karşılaştırılan diğer algoritmalardan daha iyi olduğunu göstermiştir.

KAYNAKLAR

1) **Adams, R.G.; Butchart, K.; Davey, N.,** 1999, Hierarchical Classification with a Competitive Evolutionary Tree, University of Hertfordshire, Neural Networks 12, pp.541-551, UK.

2) **Akça, M.D.; Doğan, S.,** 2002, Sayısal Görüntülerde Anabileşenler Dönüşümü, Harita Dergisi, sayı 129, pp. 1-15 http://www.photogrammetry.ethz.ch/general/persons/devrim/AnaBDon.pdf

3) **Backer, S.D.; Scheunders, P.,** 1999, Texture Segmentation by Frequency - Sensitive Elliptical Competitive Learning, Image and Vision Computing 19, pp. 639-648.

4) **Barneto, G.A.; Aroujo, A.F.R.,** Improving The Performance of Differential Competitive Learning Model in Clustering Tasks, University of Sao Paulo, Department of Electrical Engineering, Brazil.

5) **Bezdek, J.C.,** 1981, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York, pp.43-93.

6) **Bolt, G.R.,** 1992, Fault Tolerance in Artificial Neural Networks, Ph.D. Thesis, York University, Ontario.

7) **Charles, D.; Fyfe, C.; McDonald, D.,** 2002, Unsupervised Neural Networks for the Identification of Minimum Overcomplete Basis in Visual Data, Neurocomputing 47, pp.119-143, Paisley, UK.

8) **Dinçer, B.,** 1996, İllerin Sosyo-Ekonomik Gelişmişlik Sıralaması Araştırması, DPT, Ankara.

9) **Efe, M.Ö.; Kaynak, O.,** 2000, Yapay Sinir Ağları ve Uygulamaları, Doktora Tezi, Boğaziçi Üni.

10) **Elmas, Ç.,** 2003, Yapay Sinir Ağları, Seçkin Yayıncılık San. ve Tic. A.Ş., Ankara, pp.192.

11) **Fan, J.L.; Zhen, W.Z; Xie, W.X.,** 2003, Suppressed Fuzzy c-Means Clustering Algorithm, Pattern Recognition Letters 24, pp.1607-1612.

12) **Fırat, M.; Güngör, M.,** 2004, Askı Madde Konsantrasyonu ve Miktarının Yapay sinir Ağları ile Belirlenmesi, IMO Teknik Dergi, Pamukkale Üniversitesi, Denizli.

13) **Fu, L.,** 1994, Neural Networks in Computer Intelligence, University of Florida, Gainesville, pp.460.

14) **Hsu,W.H,** 2001, Expectation Maximization, Unsupervised Learning and Clustering, Department of Computing and Information Sciences, KSU, http://www.cis.ksu.edu/~bhsu

15) **Koç,S.,** 2001, Türkiye'de İllerin Sosyo-Ekonomik Özelliklere Göre Sınıflandırılması, V. Ulusal Ekonomi ve İstatistik Sempozyumu, Adana. http://idari.cu.edu.tr/sempozyum/bil20.htm

16) **Jain, A.K.; Murty M.N.; Flynn P.J.,** 1999, Data Clustering: A Review, ACM Computing Surveys, Vol. 31, No. 3.

17) **Kiang, M.Y.,** 2001, Extending The Kohonen Self-organizing Map Networks for Clustering Analaysis, California State University, Computational Statistics & Data Analysis 38, pp.161-180.

18) **Koç, S.,**1999, Kümeleme Analizleri, İzmir.

19) **Koç, M.L.; Balans, C.E.; Arslan, A.,** 2004, Taş Dolgu Dalgakıranların Yapay Sinir Ağları ile Tasarımı, IMO Teknik Dergi.

20) **Kohonen, T.,** 1989, Self-organization and Associative Memory, Berlin, Springer Verlag.

21) **Maeda, M.; Miyajima, H.,** 2000, Competitive Learning Algorithms Founded on Adaptivity and Sensitivity Deletion Methods, IEICE Trans. Fundamentals, Vol.E38-A, No.12, pp. 2770-2774.

22) **Maeda, M.; Miyajima, H.; Murashima, S., 1996,** An Adaptive Learning and Self-Deleting Neural Network for Vector Quantization, IEICE Trans. Fundamentals, Vol.E79-A, pp.1886-1893.

23) **Michaud,P.,** 1997, Clustering Techniques, Future Generation Computer Systems 13, Paris, France, pp.135-147.

24) **Nabiyev, V.V,** 2003, Yapay Zeka, Seçkin Yayıncılık, pp.724, Ankara.

25) **Oja, E.,** 2002, Unsupervised Learning in Neural Computation, Helsinki University of Technology, Neural Network Research Centre, Finland.

26) **Özdamar, K.,** 2002, Paket Programlar ile İstatistiksel Veri Analizi-2, Kaan Kitabevi, pp.513, Eskişehir.

27) **Park, D.C.,** 2000, Centroid Neural Network for Unsupervised Competitive Learning, IEEE Transactions on Neural Networks, Vol.11,No.2

28) **Park, D.C.; Woo, Y.J.,** 2001, Weighted Centroid Neural Network for Edge Preserving Image Compression, IEEE Transactions on Neural Networks, Vol.12, No.5.

29) **Rumelhart, D.E.; Zipser D.,** 1985, Competitive Learning, Cognitive Science Vol 9, pp.75-112.

30) **Shapiro, J.,** 2003, Supervised Learning I: Competitive Learning, Department of Computer Science, University of Manchaster.

31) **Su, M.C.; Liu, T.K.,** 2001, Application of Neural Networks Using Quadratic Junctions in Cluster Analysis, Neurocomputing 37, pp.165-175, National Central University, Taiwan, Republic of China.

32) **Tatlıdil, H.,** 2002, Uygulamalı Çok Değişkenli İstatistiksel Analiz, Akademi Matbaası, pp.424, Ankara.

33) **Yücetürk, A. C.,** 1999, Yapay Sinir Ağları Kullanılarak Örüntü Sınıflandırma ve Tanıma, Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir.

34) **Zimmerman, J.H.,** 1996, Fuzzy Set Theory and Its Applications. Third Edition, Kluwer Academic Publishers, pp.435, USA.

EKLER

Ek Tablo 1 : Değişkenleri Kişi Başına Düşen Değerleri

İller	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Adana	0,000503	0,000666	0,000115	0,000876	0,000607	0,000604	0,000008	0,002204	0,000326	0,000351
Adıyaman	0,00012	0,000343	0,000029	0,000664	0,000532	0,000515	0,000014	0,001058	0,000127	0,000133
Afyon	0,000236	0,00049	0,000087	0,000656	0,000716	0,000679	0,000021	0,002289	0,00023	0,000255
Ağrı	0,000078	0,000236	0,000021	0,000424	0,000274	0,000301	0,000011	0,000473	0,000121	0,000125
Amasya	0,000192	0,000482	0,000118	0,001175	0,000947	0,000928	0,000022	0,002095	0,00026	0,000277
Ankara	0,001508	0,001523	0,000553	0,002089	0,001157	0,000613	0,00001	0,003538	0,000428	0,000668
Antalya	0,000637	0,000729	0,000168	0,001057	0,000473	0,000798	0,000009	0,001368	0,000349	0,000381
Artvin	0,000198	0,000881	0,000099	0,001824	0,001198	0,001553	0,000052	0,003022	0,000177	0,000198
Aydın	0,000501	0,000674	0,000271	0,001261	0,000652	0,000856	0,000014	0,001677	0,000371	0,000407
Balıkesir	0,000401	0,000498	0,00021	0,001286	0,000692	0,001141	0,000021	0,002243	0,000347	0,000386
Bilecik	0,000221	0,00071	0,000103	0,000962	0,000581	0,000705	0,000031	0,001312	0,000216	0,000242
Bingöl	0,000122	0,000441	0,000043	0,000851	0,000757	0,000623	0,000024	0,001616	0,000102	0,000126
Bitlis	0,000075	0,000298	0,000023	0,000504	0,000273	0,000324	0,000015	0,000991	0,000087	0,000095
Bolu	0,000506	0,000573	0,000166	0,001644	0,000743	0,000846	0,000044	0,004397	0,000587	0,000613
Burdur	0,000358	0,000678	0,000148	0,001371	0,001114	0,001663	0,000031	0,002784	0,00028	0,000319
Bursa	0,000548	0,000623	0,00021	0,001022	0,000582	0,000553	0,000011	0,001778	0,000297	0,00032
Çanakkale	0,000342	0,000516	0,000191	0,001486	0,000693	0,001219	0,000026	0,001843	0,000333	0,000368
Çankırı	0,000148	0,000518	0,000074	0,000795	0,000943	0,000459	0,00003	0,001775	0,000203	0,000222
Çorum	0,000221	0,000449	0,000084	0,000583	0,000638	0,000539	0,00002	0,00243	0,000218	0,000241
Denizli	0,000538	0,000631	0,000287	0,001221	0,000788	0,001195	0,000018	0,001503	0,000391	0,000419
Diyarbakır	0,000241	0,000473	0,00012	0,000953	0,000504	0,000439	0,000011	0,002088	0,000185	0,000197
Edirne	0,000847	0,001033	0,000181	0,00154	0,000738	0,00122	0,000022	0,001813	0,000313	0,000345
Elazığ	0,000432	0,000948	0,000114	0,001329	0,000685	0,000781	0,000018	0,004003	0,000193	0,000209
Erzincan	0,000183	0,000533	0,000057	0,000833	0,000685	0,000562	0,000028	0,00196	0,000133	0,000158
Erzurum	0,000393	0,000685	0,000047	0,001083	0,000788	0,000434	0,000021	0,003115	0,000093	0,000114
Eskişehir	0,000718	0,000887	0,00017	0,001708	0,000984	0,000897	0,000016	0,003693	0,000368	0,000528
Gaziantep	0,000341	0,00037	0,000117	0,000643	0,000375	0,000388	0,000006	0,001443	0,000278	0,000306
Giresun	0,000183	0,000412	0,000097	0,0013	0,000951	0,000922	0,000031	0,002186	0,000204	0,000214
Gümüşhane	0,000107	0,000733	0,000064	0,000599	0,000513	0,00061	0,000021	0,001738	0,000118	0,000123
Hakkâri	0,000042	0,000313	0,000013	0,000537	0,000338	0,000342	0,000017	0,000634	0,000051	0,000055
Hatay	0,000278	0,000374	0,000196	0,00054	0,000349	0,000497	0,00001	0,001081	0,000262	0,000288
Isparta	0,000549	0,000833	0,000191	0,001567	0,001092	0,001267	0,000033	0,005217	0,000267	0,0003
İçel	0,000354	0,000417	0,000164	0,000926	0,000585	0,000817	0,000007	0,001597	0,000251	0,000278
İstanbul	0,000914	0,000755	0,000411	0,000964	0,000381	0,000323	0,000005	0,002548	0,000424	0,000428
İzmir	0,001018	0,001197	0,000387	0,00153	0,000658	0,000643	0,00001	0,002656	0,000457	0,000521
Kars	0,000132	0,000431	0,000055	0,000791	0,00044	0,000874	0,000018	0,001154	0,000225	0,000115

Kastamonu	0,000253	0,000695	0,000099	0,000983	0,000855	0,000597	0,000056	0,00395	0,000224	0,000242
Kayseri	0,000488	0,000757	0,000112	0,000882	0,000852	0,000577	0,000013	0,002217	0,000257	0,000282
Kırklareli	0,000399	0,000511	0,000198	0,001102	0,00067	0,000947	0,00003	0,002037	0,000341	0,000374
Kırşehir	0,000269	0,000596	0,00013	0,000703	0,001102	0,001129	0,000028	0,001974	0,000253	0,000265
Kocaeli	0,000445	0,000481	0,000163	0,0009	0,000469	0,000532	0,000009	0,001652	0,000263	0,000288
Konya	0,000328	0,000509	0,000155	0,000614	0,000523	0,000433	0,000012	0,001355	0,000249	0,000269
Kütahya	0,000206	0,000437	0,000069	0,000749	0,000687	0,000594	0,000021	0,001831	0,000196	0,000218
Malatya	0,000354	0,000592	0,000125	0,001106	0,000745	0,000927	0,000011	0,001566	0,000148	0,000175
Manisa	0,000448	0,00054	0,00016	0,000909	0,000436	0,00074	0,000018	0,00198	0,000297	0,000341
K.Maraş	0,000191	0,000372	0,00009	0,00075	0,000647	0,000644	0,000007	0,000948	0,000146	0,000159
Mardin	0,000079	0,000335	0,000033	0,00052	0,000328	0,000328	0,000007	0,000674	0,000142	0,000146
Muğla	0,000535	0,000587	0,000249	0,00175	0,000629	0,001131	0,000017	0,001531	0,000366	0,000419
Muş	0,000062	0,000284	0,000018	0,000481	0,000287	0,000388	0,000011	0,000794	0,000071	0,000073
Nevşehir	0,000216	0,000532	0,000126	0,000774	0,001116	0,000694	0,000029	0,001513	0,000284	0,00031
Niğde	0,00019	0,00048	0,000078	0,000741	0,000707	0,000842	0,00002	0,001666	0,000236	0,000244
Ordu	0,000211	0,000323	0,000039	0,000892	0,000479	0,000751	0,000017	0,00163	0,000195	0,000205
Rize	0,000254	0,00047	0,000087	0,000861	0,000598	0,000588	0,000025	0,002099	0,00024	0,00026
Sakarya	0,000298	0,000353	0,000173	0,000513	0,000466	0,000518	0,000015	0,001521	0,000263	0,000275
Samsun	0,000436	0,000641	0,00017	0,001108	0,00083	0,000674	0,000014	0,002585	0,000271	0,000291
Siirt	0,000076	0,000413	0,000038	0,000667	0,000501	0,000482	0,000015	0,001081	0,000288	0,000174
Sinop	0,000213	0,000567	0,000111	0,001042	0,001095	0,000864	0,000031	0,002438	0,000226	0,000239
Sivas	0,000302	0,000751	0,000049	0,000947	0,000616	0,000608	0,000023	0,002977	0,000143	0,00016
Tekirdağ	0,000428	0,000497	0,000176	0,000898	0,000523	0,000693	0,000014	0,00143	0,000297	0,000316
Tokat	0,000143	0,000344	0,000059	0,000763	0,000678	0,000624	0,000014	0,001488	0,000145	0,000151
Trabzon	0,00041	0,000643	0,000136	0,001273	0,000916	0,000675	0,000013	0,002528	0,000197	0,000212
Tunceli	0,000075	0,001143	0,000085	0,001923	0,001197	0,002062	0,000075	0,00187	0,000214	0,000235
Ş.Urfa	0,00014	0,000272	0,000036	0,000388	0,000283	0,000229	0,000009	0,000828	0,000168	0,000174
Uşak	0,000344	0,000645	0,000192	0,00099	0,001018	0,001514	0,000022	0,001862	0,000289	0,000326
Van	0,000232	0,000364	0,000035	0,000599	0,000283	0,000302	0,000011	0,001413	0,000075	0,000085
Yozgat	0,000144	0,000387	0,000038	0,000567	0,000633	0,000496	0,000015	0,001245	0,000138	0,000146
Zonguldak	0,000385	0,000489	0,000188	0,000954	0,000427	0,000642	0,000015	0,002818	0,000288	0,000341
Aksaray	0,000192	0,000543	0,000076	0,000472	0,000487	0,000346	0,000015	0,001035	0,000212	0,00022
Bayburt	0,000134	0,000791	0,000041	0,00074	0,000555	0,000616	0,00001	0,001027	0,000092	0,000092
Karaman	0,00021	0,000617	0,000136	0,000625	0,00058	0,000576	0,000008	0,000781	0,000243	0,000263
Kırıkkale	0,00036	0,000443	0,000083	0,000712	0,001218	0,000485	0,000013	0,001591	0,000149	0,000162
Batman	0,000145	0,00025	0,000046	0,000506	0,000493	0,000254	0,000007	0,000394	0,000035	0,000094
Şırnak	0,000054	0,000323	0,000017	0,000405	0,000243	0,000297	0,000017	0,000609	0,000108	0,000108
Bartın	0,00025	0,000554	0,000147	0,000874	0,00057	0,000668	0,000027	0,001667	0,000119	0,000239
Ardahan	0,00006	0,00071	0,000045	0,000979	0,000583	0,000792	0,00003	0,001234	0,000067	0,000097
Iğdır	0,000172	0,00051	0,000059	0,000812	0,000344	0,000961	0,000006	0,000593	0,000042	0,000148
Yalova	0,000386	0,00054	0,000297	0,000575	0,00038	0,000463	0,000006	0,000593	0,000119	0,000368

Karabük	0,00032	0,000578	0,000142	0,001013	0,000702	0,000773	0,000031	0,002443	0,000053	0,000244
Kilis	0,0002	0,000715	0,000044	0,000732	0,000357	0,000758	0,000009	0,001656	0,000009	0,000279
Osmaniye	0,000129	0,000349	0,000129	0,000538	0,00068	0,000863	0,000009	0,000708	0,000037	0,000225
Düzce	0,000204	0,000296	0,000025	0,000608	0,000239	0,000356	0,000022	0,001671	0,000016	0,000006

Ek Tablo 2 : Birim Vektöre Dönüştürülerek Standartlaştırılan Veriler

İller	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Adana	0.186	0.274	0.042	0.324	0.225	0.224	0.0030	0.816	0.121	0.132
Adıyaman	0.080	0.227	0.019	0.440	0.353	0.341	0.0096	0.701	0.084	0.882
Afyon	0.089	0.184	0.033	0.247	0.269	0.256	0.0079	0.861	0.087	0.096
Ağrı	0.095	0.291	0.026	0.521	0.337	0.370	0.0140	0.581	0.149	0.153
Amasya	0.068	0.171	0.042	0.417	0.336	0.329	0.0078	0.743	0.092	0.098
Ankara	0.309	0.312	0.113	0.428	0.237	0.126	0.0022	0.724	0.088	0.086
Antalya	0.282	0.323	0.075	0.469	0.210	0.354	0.0041	0.607	0.155	0.169
Artvin	0.048	0.212	0.024	0.440	0.289	0.374	0.0126	0.729	0.043	0.048
Aydın	0.194	0.262	0.105	0.489	0.253	0.332	0.0053	0.651	0.144	0.158
Balıkesir	0.132	0.164	0.069	0.424	0.228	0.376	0.0071	0.740	0.115	0.127
Bilecik	0.109	0.348	0.051	0.472	0.285	0.346	0.0152	0.644	0.106	0.119
Bingöl	0.057	0.207	0.020	0.400	0.355	0.292	0.0111	0.759	0.048	0.059
Bitlis	0.060	0.242	0.019	0.408	0.221	0.262	0.0125	0.802	0.071	0.077
Bolu	0.102	0.115	0.034	0.331	0.150	0.170	0.0089	0.886	0.118	0.124
Burdur	0.094	0.178	0.039	0.361	0.293	0.438	0.0082	0.733	0.074	0.084
Bursa	0.228	0.259	0.087	0.425	0.242	0.23	0.0045	0.740	0.124	0.133
Çanakkale	0.119	0.180	0.067	0.518	0.241	0.425	0.0090	0.642	0.116	0.128
Çankırı	0.064	0.226	0.032	0.346	0.411	0.200	0.0129	0.773	0.089	0.097
Çorum	0.082	0.166	0.031	0.216	0.236	0.200	0.0074	0.899	0.081	0.089
Denizli	0.205	0.240	0.109	0.465	0.300	0.455	0.0067	0.572	0.149	0.159
Diyarbakır	0.098	0.192	0.049	0.386	0.204	0.178	0.0045	0.847	0.075	0.080
Edirne	0.272	0.331	0.058	0.494	0.237	0.391	0.0072	0.581	0.100	0.111
Elazığ	0.097	0.212	0.026	0.297	0.153	0.174	0.0039	0.894	0.043	0.047
Erzincan	0.077	0.224	0.024	0.349	0.287	0.236	0.0119	0.822	0.056	0.066
Erzurum	0.112	0.195	0.013	0.308	0.224	0.124	0.0061	0.887	0.026	0.033
Eskişehir	0.160	0.198	0.038	0.381	0.220	0.200	0.0035	0.824	0.082	0.118
Gaziantep	0.190	0.206	0.065	0.358	0.209	0.216	0.0035	0.804	0.155	0.170
Giresun	0.063	0.141	0.033	0.445	0.326	0.316	0.0105	0.749	0.070	0.073
Gümüşhane	0.050	0.342	0.030	0.279	0.239	0.284	0.0100	0.811	0.055	0.057
Hakkâri	0.042	0.309	0.013	0.530	0.334	0.338	0.0167	0.626	0.050	0.054
Hatay	0.186	0.250	0.131	0.361	0.233	0.332	0.0064	0.723	0.176	0.193
Isparta	0.095	0.144	0.033	0.270	0.188	0.218	0.0057	0.899	0.046	0.052
İçel	0.160	0.189	0.074	0.419	0.265	0.370	0.0033	0.723	0.114	0.126
İstanbul	0.295	0.244	0.133	0.311	0.123	0.104	0.0017	0.822	0.133	0.138
İzmir	0.278	0.327	0.106	0.418	0.180	0.176	0.0028	0.725	0.125	0.161
Kars	0.074	0.241	0.031	0.443	0.247	0.490	0.0103	0.647	0.126	0.062
Kastamonu	0.059	0.162	0.023	0.230	0.200	0.139	0.0131	0.923	0.052	0.057
Kayseri	0.193	0.272	0.040	0.316	0.305	0.207	0.0047	0.795	0.092	0.101
Kırklareli	0.146	0.188	0.073	0.404	0.246	0.347	0.0112	0.747	0.125	0.137
Kırşehir	0.099	0.218	0.048	0.257	0.403	0.413	0.0101	0.723	0.093	0.097
Kocaeli	0.206	0.223	0.076	0.417	0.218	0.247	0.0042	0.766	0.122	0.133
Konya	0.183	0.284	0.087	0.343	0.293	0.242	0.0069	0.758	0.139	0.150
Kütahya	0.091	0.194	0.030	0.333	0.305	0.264	0.0095	0.814	0.087	0.097
Malatya	0.149	0.249	0.053	0.466	0.314	0.390	0.0044	0.660	0.062	0.074

Manisa	0.180	0.217	0.064	0.365	0.175	0.297	0.0073	0.795	0.119	0.137
K.Maraş	0.120	0.234	0.057	0.472	0.407	0.406	0.0044	0.596	0.092	0.100
Mardin	0.076	0.319	0.031	0.496	0.312	0.312	0.0068	0.642	0.135	0.139
Muğla	0.188	0.206	0.088	0.616	0.221	0.398	0.0059	0.538	0.129	0.147
Muş	0.057	0.261	0.016	0.441	0.263	0.356	0.0101	0.728	0.065	0.067
Nevşehir	0.096	0.235	0.056	0.342	0.492	0.306	0.0128	0.667	0.125	0.137
Niğde	0.086	0.216	0.035	0.334	0.319	0.379	0.0091	0.751	0.106	0.110
Ordu	0.100	0.153	0.019	0.422	0.226	0.355	0.0080	0.770	0.092	0.097
Rize	0.102	0.188	0.035	0.344	0.239	0.235	0.0098	0.838	0.096	0.104
Sakarya	0.160	0.190	0.093	0.276	0.251	0.279	0.0078	0.819	0.142	0.148
Samsun	0.139	0.204	0.054	0.353	0.264	0.215	0.0045	0.824	0.087	0.093
Siirt	0.049	0.267	0.025	0.432	0.324	0.312	0.0098	0.699	0.187	0.113
Sinop	0.069	0.184	0.036	0.339	0.356	0.281	0.0101	0.793	0.074	0.078
Sivas	0.090	0.224	0.015	0.283	0.184	0.182	0.0067	0.889	0.043	0.048
Tekirdağ	0.208	0.241	0.086	0.435	0.253	0.336	0.0070	0.693	0.144	0.153
Tokat	0.073	0.176	0.030	0.390	0.346	0.319	0.0074	0.760	0.074	0.077
Trabzon	0.130	0.203	0.043	0.403	0.29	0.214	0.0042	0.800	0.062	0.067
Tunceli	0.020	0.302	0.023	0.508	0.316	0.545	0.0198	0.494	0.060	0.062
Ş.Urfa	0.132	0.257	0.034	0.366	0.267	0.216	0.0085	0.782	0.159	0.164
Uşak	0.118	0.221	0.066	0.339	0.348	0.518	0.0074	0.637	0.099	0.112
Van	0.141	0.220	0.021	0.363	0.171	0.183	0.0069	0.856	0.046	0.052
Yozgat	0.087	0.234	0.023	0.343	0.383	0.300	0.0089	0.753	0.083	0.089
Zonguldak	0.121	0.154	0.059	0.301	0.135	0.202	0.0046	0.888	0.091	0.108
Aksaray	0.133	0.376	0.053	0.327	0.338	0.240	0.0105	0.717	0.147	0.152
Bayburt	0.078	0.460	0.024	0.430	0.323	0.359	0.0060	0.598	0.054	0.054
Karaman	0.140	0.412	0.091	0.418	0.387	0.385	0.0055	0.522	0.162	0.176
Kırıkkale	0.159	0.196	0.037	0.314	0.537	0.214	0.0058	0.702	0.066	0.071
Batman	0.160	0.277	0.051	0.561	0.546	0.282	0.0073	0.437	0.039	0.104
Şırnak	0.060	0.358	0.019	0.449	0.270	0.330	0.0188	0.675	0.119	0.119
Bartın	0.114	0.253	0.067	0.400	0.261	0.306	0.0124	0.763	0.055	0.109
Ardahan	0.030	0.356	0.023	0.491	0.293	0.398	0.0150	0.619	0.034	0.049
Iğdır	0.112	0.331	0.039	0.528	0.223	0.624	0.0039	0.385	0.027	0.096
Yalova	0.294	0.412	0.226	0.439	0.290	0.353	0.0045	0.452	0.091	0.281
Karabük	0.109	0.197	0.049	0.345	0.239	0.264	0.0106	0.833	0.018	0.083
Kilis	0.093	0.333	0.020	0.341	0.166	0.353	0.0041	0.771	0.004	0.130
Osmaniye	0.087	0.235	0.087	0.363	0.458	0.581	0.0059	0.477	0.025	0.151
Düzce	0.109	0.159	0.014	0.326	0.128	0.191	0.0119	0.896	0.009	0.003

Ek Tablo 3 : Maksimum Değer 1 Olacak Şekilde Standartlaştırılan Veriler.

İller	SX1	SX2	SX3	SX4	SX5	SX6	SX7	SX8	SX9	SX10
Adana	0,334	0,437	0,208	0,419	0,498	0,293	0,107	0,422	0,555	0,525
Adıyaman	0,080	0,225	0,052	0,318	0,437	0,250	0,187	0,203	0,216	0,199
Afyon	0,156	0,322	0,157	0,314	0,588	0,329	0,280	0,439	0,392	0,382
Ağrı	0,052	0,155	0,038	0,203	0,225	0,146	0,147	0,091	0,206	0,187
Amasya	0,127	0,316	0,213	0,562	0,778	0,450	0,293	0,402	0,443	0,415
Ankara	1,000	1,000	1,000	1,000	0,950	0,297	0,133	0,678	0,729	1,000
Antalya	0,422	0,479	0,304	0,506	0,388	0,387	0,120	0,262	0,595	0,570
Artvin	0,131	0,578	0,179	0,873	0,984	0,753	0,693	0,579	0,302	0,296
Aydın	0,332	0,443	0,490	0,604	0,535	0,415	0,187	0,321	0,632	0,609
Balıkesir	0,266	0,327	0,380	0,616	0,568	0,553	0,280	0,430	0,591	0,578
Bilecik	0,147	0,466	0,186	0,461	0,477	0,342	0,413	0,251	0,368	0,362
Bingöl	0,081	0,290	0,078	0,407	0,622	0,302	0,320	0,310	0,174	0,189
Bitlis	0,050	0,196	0,042	0,241	0,224	0,157	0,200	0,190	0,148	0,142
Bolu	0,336	0,376	0,300	0,787	0,610	0,410	0,587	0,843	1,000	0,918
Burdur	0,237	0,445	0,268	0,656	0,915	0,806	0,413	0,534	0,477	0,478
Bursa	0,363	0,409	0,380	0,489	0,478	0,268	0,147	0,341	0,506	0,479
Çanakkale	0,227	0,339	0,345	0,711	0,569	0,591	0,347	0,353	0,567	0,551
Çankırı	0,098	0,340	0,134	0,381	0,774	0,223	0,400	0,340	0,346	0,332
Çorum	0,147	0,295	0,152	0,279	0,524	0,261	0,267	0,466	0,371	0,361
Denizli	0,357	0,414	0,519	0,584	0,647	0,580	0,240	0,288	0,666	0,627
Diyarbakır	0,160	0,311	0,217	0,456	0,414	0,213	0,147	0,400	0,315	0,295
Edirne	0,562	0,678	0,327	0,737	0,606	0,592	0,293	0,348	0,533	0,516
Elazığ	0,286	0,622	0,206	0,636	0,562	0,379	0,240	0,767	0,329	0,313
Erzincan	0,121	0,350	0,103	0,399	0,562	0,273	0,373	0,376	0,227	0,237
Erzurum	0,261	0,450	0,085	0,518	0,647	0,210	0,280	0,597	0,158	0,171
Eskişehir	0,476	0,582	0,307	0,818	0,808	0,435	0,213	0,708	0,627	0,790
Gaziantep	0,226	0,243	0,212	0,308	0,308	0,188	0,080	0,277	0,474	0,458
Giresun	0,121	0,271	0,175	0,622	0,781	0,447	0,413	0,419	0,348	0,320
Gümüşhane	0,071	0,481	0,116	0,287	0,421	0,296	0,280	0,333	0,201	0,184
Hakkâri	0,028	0,206	0,024	0,257	0,278	0,166	0,227	0,122	0,087	0,082
Hatay	0,184	0,246	0,354	0,258	0,287	0,241	0,133	0,207	0,446	0,431
Isparta	0,364	0,547	0,345	0,750	0,897	0,614	0,440	1,000	0,455	0,449
İçel	0,235	0,274	0,297	0,443	0,480	0,396	0,093	0,306	0,428	0,416
İstanbul	0,606	0,496	0,743	0,461	0,313	0,157	0,067	0,488	0,722	0,641
İzmir	0,675	0,786	0,700	0,732	0,540	0,312	0,133	0,509	0,779	0,885
Kars	0,088	0,283	0,099	0,379	0,361	0,424	0,240	0,221	0,383	0,166
Kastamonu	0,168	0,456	0,179	0,471	0,702	0,290	0,747	0,757	0,382	0,362
Kayseri	0,324	0,497	0,203	0,422	0,700	0,280	0,173	0,425	0,438	0,422
Kırklareli	0,265	0,336	0,358	0,528	0,550	0,459	0,400	0,390	0,581	0,560
Kırşehir	0,178	0,391	0,235	0,337	0,905	0,548	0,373	0,378	0,431	0,397
Kocaeli	0,295	0,316	0,295	0,431	0,385	0,258	0,120	0,317	0,448	0,431
Konya	0,218	0,334	0,280	0,294	0,429	0,210	0,160	0,260	0,424	0,403
Kütahya	0,137	0,287	0,125	0,359	0,564	0,288	0,280	0,351	0,334	0,326
Malatya	0,235	0,389	0,226	0,529	0,612	0,450	0,147	0,300	0,252	0,262

Manisa	0,297	0,355	0,289	0,435	0,358	0,359	0,240	0,380	0,506	0,510
K.Maraş	0,127	0,244	0,163	0,359	0,531	0,312	0,093	0,182	0,249	0,238
Mardin	0,052	0,220	0,060	0,249	0,269	0,159	0,093	0,129	0,242	0,219
Muğla	0,355	0,385	0,450	0,838	0,516	0,548	0,227	0,293	0,624	0,627
Muş	0,041	0,186	0,033	0,230	0,236	0,188	0,147	0,152	0,121	0,109
Nevşehir	0,143	0,349	0,228	0,371	0,916	0,337	0,387	0,290	0,484	0,464
Niğde	0,126	0,315	0,141	0,355	0,580	0,408	0,267	0,319	0,402	0,365
Ordu	0,140	0,212	0,071	0,427	0,393	0,364	0,227	0,312	0,332	0,307
Rize	0,168	0,309	0,157	0,412	0,491	0,285	0,333	0,402	0,409	0,389
Sakarya	0,198	0,232	0,313	0,246	0,383	0,251	0,200	0,292	0,448	0,412
Samsun	0,289	0,421	0,307	0,530	0,681	0,327	0,187	0,495	0,462	0,436
Siirt	0,050	0,271	0,069	0,319	0,411	0,234	0,200	0,207	0,491	0,260
Sinop	0,141	0,372	0,201	0,499	0,899	0,419	0,413	0,467	0,385	0,358
Sivas	0,200	0,493	0,089	0,453	0,506	0,295	0,307	0,571	0,244	0,240
Tekirdağ	0,284	0,326	0,318	0,430	0,429	0,336	0,187	0,274	0,506	0,473
Tokat	0,095	0,226	0,107	0,365	0,557	0,303	0,187	0,285	0,247	0,226
Trabzon	0,272	0,422	0,246	0,609	0,752	0,327	0,173	0,485	0,336	0,317
Tunceli	0,050	0,750	0,154	0,921	0,983	1,000	1,000	0,358	0,365	0,352
Ş.Urfa	0,093	0,179	0,065	0,186	0,232	0,111	0,120	0,159	0,286	0,260
Uşak	0,228	0,424	0,347	0,474	0,836	0,734	0,293	0,357	0,492	0,488
Van	0,154	0,239	0,063	0,287	0,232	0,146	0,147	0,271	0,128	0,127
Yozgat	0,095	0,254	0,069	0,271	0,520	0,241	0,200	0,239	0,235	0,219
Zonguldak	0,255	0,321	0,340	0,457	0,351	0,311	0,200	0,540	0,491	0,510
Aksaray	0,127	0,357	0,137	0,226	0,400	0,168	0,200	0,198	0,361	0,329
Bayburt	0,089	0,519	0,074	0,354	0,456	0,299	0,133	0,197	0,157	0,138
Karaman	0,139	0,405	0,246	0,299	0,476	0,279	0,107	0,150	0,414	0,394
Kırıkkale	0,239	0,291	0,150	0,341	1,000	0,235	0,173	0,305	0,254	0,243
Batman	0,096	0,164	0,083	0,242	0,405	0,123	0,093	0,076	0,060	0,141
Şırnak	0,036	0,212	0,031	0,194	0,200	0,144	0,227	0,117	0,184	0,162
Bartın	0,166	0,364	0,266	0,418	0,468	0,324	0,360	0,320	0,203	0,358
Ardahan	0,040	0,466	0,081	0,469	0,479	0,384	0,400	0,237	0,114	0,145
Iğdır	0,114	0,335	0,107	0,389	0,282	0,466	0,080	0,114	0,072	0,222
Yalova	0,256	0,355	0,537	0,275	0,312	0,225	0,080	0,114	0,203	0,551
Karabük	0,212	0,380	0,257	0,485	0,576	0,375	0,413	0,468	0,090	0,365
Kilis	0,133	0,469	0,080	0,350	0,293	0,368	0,120	0,317	0,015	0,418
Osmaniye	0,086	0,229	0,233	0,258	0,558	0,419	0,120	0,136	0,063	0,337
Düzce	0,135	0,194	0,045	0,291	0,196	0,173	0,293	0,320	0,027	0,009

Ek 1: Açıya Dayalı Uyarlamalı YÖA ile Sınıflandırma Yapan Programın Kaynak Kodu

```
unit OPSSB01;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, DB, StdCtrls, DBTables, Mask, DBCtrls, Grids, DBGrids;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label12: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Label13: TLabel;
    Timer2: TTimer;
    Edit3: TEdit;
    Label17: TLabel;
    Button4: TButton;
    StringGrid4: TStringGrid;
    Edit4: TEdit;
    Label1: TLabel;
    Table1: TTable;
    DataSource1: TDataSource;
    Label2: TLabel;
    StringGrid5: TStringGrid;
    Table1ILLER: TStringField;
    Table1UZMAN: TFloatField;
    Table1PRATISYEN: TFloatField;
    Table1DIS: TFloatField;
    Table1HEMSIRE: TFloatField;
    Table1SAG_MEMUR: TFloatField;
```

```
Table1EBE: TFloatField;
Table1HASTANE: TFloatField;
Table1YATAK: TFloatField;
Table1ECZANE: TFloatField;
Table1ECZACI: TFloatField;
Table1NUFUS: TFloatField;
Label3: TLabel;
StringGrid6: TStringGrid;
StringGrid3: TStringGrid;
Button5: TButton;
Button3: TButton;
Edit1: TEdit;
Table3: TTable;
DataSource3: TDataSource;
Table3X1: TFloatField;
Table3X2: TFloatField;
Table3X3: TFloatField;
Table3X4: TFloatField;
Table3X5: TFloatField;
Table3X6: TFloatField;
Table3X7: TFloatField;
Table3X8: TFloatField;
Table3X9: TFloatField;
Table3X10: TFloatField;
Label5: TLabel;
Label4: TLabel;
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
```

```
private
   { Private declarations }
 public
   { Public declarations }
 end;
var
Form1: TForm1;
cl:string;
n,z,sr,sayac,k,sayac2,j,r,i,tt,vs,kw,gkw,s,t,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s
14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32,s33,s34,
s35,s36,s37,s38,s39,s40,s41,s42,s43,s44,s45,s46,s47,s48,s49,s50:integer;
gwe,gwy,deltaw,gdeltaw,dd,we,wy:array [1..50,1..50] of real;
gveri,veri,cluster,gcluster:array [1..100,1..50] of real;
guwe,guwy,uwe,uwy,gstop,gwmat,stop,ud,wmat,bh,ss,gd,ggd,dtoplam,dk,d,ggdtopla
m,gdtoplam,stoplam,hata,eghata,x,y,gy,gx,ortwe:array [1..50]of real;
max,gmax,m,ud1,w,topbh,mariot,degis,dktop,gthata,gshata,shata,thata,mind,gdtop:re
al;
 sk:array [1..100,1..50,1..10]of real;
 mak,gmak,l,gl:array[1..100]of real;
 uveri:array[1..100] of real;
 a:array [1..50]of integer;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
   Halt;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
   Timer2.Enabled:=False;
   for k:=1 to n do
   begin
```

```
   stoplam[k]:=0;
   bh[k]:=0;
end;
w:=0;
s1:=0;s2:=0;s3:=0;s4:=0;s5:=0;s6:=0;s7:=0;s8:=0;s9:=0;s10:=0;
s11:=0;s12:=0;s13:=0;s14:=0;s15:=0;s16:=0;s17:=0;s18:=0;s19:=0;s20:=0;
s21:=0;s22:=0;s23:=0;s24:=0;s25:=0;s26:=0;s27:=0;s28:=0;s29:=0;s30:=0;
for j:=1 to 10 do
ss[j]:=0;
s:=0;
sr:=1;
thata:=0;
shata:=0;
sayac:=0;
t:=0;
a[t]:=0;
//sınıf sayısını gir.
cl:=inputbox('Sınıf Sayısı','Sınıf Sayısını Giriniz','');
n:=StrToInt(cl);
// table daki verileri çek.
vs:=81;
table1.Open;
for i:=1 to vs do
begin
  veri[i][1]:=StrToInt(Table1UZMAN.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][2]:=StrToInt(Table1PRATISYEN.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][3]:=StrToInt(Table1DIS.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][4]:=StrToInt(Table1HEMSIRE.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][5]:=StrToInt(Table1SAG_MEMUR.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][6]:=StrToInt(Table1EBE.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][7]:=StrToInt(Table1HASTANE.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][8]:=StrToInt(Table1YATAK.Text)/StrToInt(Table1NUFUS.Text);
```

```
  veri[i][9]:=StrToInt(Table1ECZANE.Text)/StrToInt(Table1NUFUS.Text);
  veri[i][10]:=StrToInt(Table1ECZACI.Text)/StrToInt(Table1NUFUS.Text);
  table1.Next;
end;
Table1.close;
Table3.Open;
for i:=1 to vs do
begin
  gveri[i][1]:=StrToFloat(Table3X1.Text);
  gveri[i][2]:=StrToFloat(Table3X2.Text);
  gveri[i][3]:=StrToFloat(Table3X3.Text);
  gveri[i][4]:=StrToFloat(Table3X4.Text);
  gveri[i][5]:=StrToFloat(Table3X5.Text);
  gveri[i][6]:=StrToFloat(Table3X6.Text);
  gveri[i][7]:=StrToFloat(Table3X7.Text);
  gveri[i][8]:=StrToFloat(Table3X8.Text);
  gveri[i][9]:=StrToFloat(Table3X9.Text);
  gveri[i][10]:=StrToFloat(Table3X10.Text);
  table3.Next;
end;
Table3.close;
//Kayıtları birim vektöre dönüştür.
for i:=1 to vs do
uveri[i]:=0;
//verilerin uzunluklarını hesapla.
for i:=1 to vs do
begin
   for j:=1 to 10 do
   uveri[i]:=uveri[i]+sqr(veri[i][j]);
   uveri[i]:=sqrt(uveri[i]);
end;
for i:=1 to vs do
```

```
    for j:=1 to 10 do
    veri[i][j]:=veri[i][j]/uveri[i];
    // ağırlık vektörlerini üret.
    r:=0;
    Randomize;
    for k:=1 to n do
    begin
      r:=random(80)+1;
      for j:=1 to 10 do
      begin
        we[k][j]:=veri[r][j];
        gwe[k][j]:=gveri[r][j];
      end;
    end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
    for k:=1 to n do
    begin
      for j:=1 to 10 do
      begin
        StringGrid2.Cells[0,k]:=IntToStr(k);
        StringGrid2.cells[j,k]:=FloatToStrF(we[k][j],ffgeneral,5,10);
        StringGrid2.Cells[k,0]:=IntToStr(k);
      end;
    end;
    sayac:=sayac+1;
    if sayac>10000 then
    begin
      Timer2.Enabled:=False;
    end
    else
```

```
begin
  Edit3.Text:=IntToStr(sayac);
  Randomize;
  //random x vektörünü seç ve ekrana yazdır.
  z:=random(80)+1;
  for j:=1 to 10 do
  StringGrid1.Cells[j,1]:=FloatToStrF(veri[z][j],ffgeneral,5,10);
  StringGrid1.Cells[0,1]:=IntToStr(z);
  for j:=1 to 10 do
  StringGrid1.Cells[j,0]:=IntToStr(j);
  //seçilen x vektörünü ağırlık vektörleriyle çarp.
  for k:=1 to n do
  begin
    y[k]:=0;
    gy[k]:=0;
  end;
  for j:=1 to 10 do
  x[j]:=0;
  for j:=1 to 10 do
  x[j]:=veri[z][j];
  for k:=1 to n do
  begin
    for j:=1 to 10 do
    y[k]:=y[k]+(x[j]*we[k][j]);
  end;
  //Vektör Çarpımlarının Ekrana Yazdırılması
  for k:=1 to n do
  begin
    StringGrid5.Cells[k,1]:=FloatToStrF(y[k],ffgeneral,5,10);
    StringGrid5.Cells[k,0]:=FloatToStr(k);
  end;
  //Birim vektörler üzerinden kazanan vektörü belirle.
```

```
max:=0;
for k:=1 to n do
begin
   if max<y[k] then
   begin
      max:=y[k];
      kw:=k;
   end;
end;
Edit4.Text:=IntToStr(kw);
//kazanan ağırlık vektörünü güncelle.
m:=0.03;
for j:=1 to 10 do
begin
   deltaw[kw][j]:=m*((veri[z][j]/10)-we[kw][j]);
   gdeltaw[kw][j]:=m*(gveri[z][j]-gwe[kw][j]);
   StringGrid6.Cells[j,1]:=FloatToStrF(deltaw[kw][j],ffgeneral,5,10);
   StringGrid6.Cells[j,2]:=FloatToStrF(gdeltaw[kw][j],ffgeneral,5,10);
   wy[kw][j]:=we[kw][j]+deltaw[kw][j];
   gwy[kw][j]:=gwe[kw][j]+gdeltaw[kw][j];
end;
//Kazanan ağırlık vektörünün uzunluğunu hesapla.
uwy[kw]:=0;
for j:=1 to 10 do
uwy[kw]:=uwy[kw]+sqr(wy[kw][j]);
uwy[kw]:=sqrt(uwy[kw]);

//kazanan ağırlık vektörünü birim vektöre dönüştür.
for j:=1 to 10 do
wy[kw][j]:=wy[kw][j]/uwy[kw];
//eğitim hatasının hesaplanması
for j:=1 to 10 do
```

```
        eghata[kw]:=eghata[kw]+(wy[kw][j]*veri[z][j]);
        eghata[kw]:=eghata[kw]/2;
        StringGrid3.Cells[kw,3]:=FloatToStrF(eghata[kw],ffgeneral,5,10);
        //Kazanan ağırlık vektörünü ekrana yazdır.
        for j:=1 to 10 do
        StringGrid2.Cells[j,kw]:=FloatToStrF(wy[kw][j],ffgeneral,5,10);
        StringGrid2.Cells[0,kw]:=FloatToStr(kw);
        //Ağırlık vektörünü güncelle.
        for j:=1 to 10 do
        begin
          we[kw][j]:=wy[kw][j];
          gwe[kw][j]:=gwy[kw][j];
        end;
        Timer2.Enabled:=True;
    end;
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
    for k:=1 to n do
    begin
        gd[k]:=0;
        ggd[k]:=0;
        dk[k]:=0;
        gdtoplam[k]:=0;
        ggdtoplam[k]:=0;
        gdtop:=0;
    end;
    ud1:=0;
    for k:=1 to n do
    begin
        for j:=1 to 10 do
        begin
```

```
        StringGrid2.Cells[0,k]:=('                ');
        StringGrid2.cells[j,k]:=('                ');
        StringGrid2.Cells[k,0]:=('                ');
    end;
end;
for i:=1 to vs do
for k:=1 to n do
begin
    cluster[i][k]:=0;
    gcluster[i][k]:=0;
end;
for i:=1 to vs do
begin
    for k:=1 to n do
    for j:=1 to 10 do
    cluster[i][k]:=cluster[i][k]+(veri[i][j]*we[k][j]);
end;
for i:=1 to vs do
begin
    for k:=1 to n do
    for j:=1 to 10 do
    gcluster[i][k]:=gcluster[i][k]+(gveri[i][j]*gwe[k][j]);
end;
for i:=1 to vs do
begin
    mak[i]:=0;
    gmak[i]:=0;
end;
for i:=1 to vs do
begin
    for k:=1 to n do
    if mak[i]<cluster[i][k]then
```

```
    begin
      mak[i]:=cluster[i][k];
      l[i]:=k;
    end;
  end;
  for i:=1 to vs do
  begin
    for k:=1 to n do
    if gmak[i]<gcluster[i][k]then
    begin
      gmak[i]:=gcluster[i][k];
      gl[i]:=k;
    end;
  end;
  mind:=100000;
  dktop:=0;
  for k:=1 to n do
  d[k]:=0;
  for k:=1 to n do
  begin
    dtoplam[k]:=0;
    gdtoplam[k]:=0;
    for i:=1 to vs do
    begin
      if l[i]=k then
      begin
        for j:=1 to 10 do
        begin
          dtoplam[k]:=dtoplam[k]+(veri[i][j]*we[k][j]);
          gdtoplam[k]:=gdtoplam[k]+(gveri[i][j]*gwe[k][j])
        end;
      end;
```

```
    end;
    d[k]:=dtoplam[k]/10;
    gd[k]:=gdtoplam[k]/10;
end;
for k:=1 to n do
begin
    StringGrid3.Cells[k,1]:=FloatToStrF(d[k],ffgeneral,4,10);
    StringGrid3.Cells[k,2]:=FloatToStrF(gd[k],ffgeneral,4,10);
    StringGrid3.Cells[k,0]:=IntToStr(k);
end;
for k:=1 to n do
begin
    for i:=1 to vs do
    begin
        if gl[i]=k then
        begin
            for j:=1 to 10 do
            begin
                ggdtoplam[k]:=ggdtoplam[k]+(gveri[i][j]*gwe[k][j]);
            end;
        end;
    end;
    ggd[k]:=ggdtoplam[k]/10;
end;
t:=0;
if n>=14 then
begin
    tt:=1;
    for k:=1 to n do
    begin
        if tt<6 then
        begin
```

```
          for k:=1 to n do
          begin
            if mind>d[k]then
            begin
              mind:=d[k];
              ud[tt]:=k;
            end;
          end;
          for k:=1 to n do
          begin
            if ud[tt]<>k then
            begin
              t:=t+1;
              for j:=1 to 10 do
              begin
                we[t][j]:=we[k][j];
                gwe[t][j]:=gwe[k][j];
              end;
            end;
          end;
          tt:=tt+1;
          mind:=100000;
        end;
    end;
    n:=n-5;
    sayac:=0;
    Timer2.Enabled:=True;
end;
mind:=100000;
if (n<=10) and (n>3) then
begin
    for k:=1 to n do
```

```
      begin
        if mind>d[k]then
        begin
          mind:=d[k];
          ud1:=k;
        end;
      end;
      for k:=1 to n do
      begin
        if ud1<>k then
        begin
          t:=t+1;
          for j:=1 to 10 do
          begin
            we[t][j]:=we[k][j];
            gwe[t][j]:=gwe[k][j];
          end;
        end;
      end;
      n:=n-1;
      sayac:=0;
      Timer2.Enabled:=True;
    end;
    if n<3 then
    Timer2.Enabled:=False;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
    for i:=1 to vs do
    l[i]:=0;
    s1:=0;s2:=0;s3:=0;s4:=0;s5:=0;s6:=0;s7:=0;s8:=0;s9:=0;s10:=0;
    s11:=0;s12:=0;s13:=0;s14:=0;s15:=0;s16:=0;s17:=0;s18:=0;s19:=0;s20:=0;
```

```
s21:=0;s22:=0;s23:=0;s24:=0;s25:=0;s26:=0;s27:=0;s28:=0;s29:=0;s30:=0;
for k:=1 to n do
ss[k]:=0;
for i:=1 to vs do
for k:=1 to n do
cluster[i][k]:=0;
for i:=1 to vs do
begin
   for k:=1 to n do
   for j:=1 to 10 do
   cluster[i][k]:=cluster[i][k]+(veri[i][j]*we[k][j]);
end;
for i:=1 to vs do
mak[i]:=0;
for i:=1 to vs do
begin
   for k:=1 to n do
   if mak[i]<cluster[i][k]then
   begin
      mak[i]:=cluster[i][k];
      l[i]:=k;
   end;
end;
for i:=1 to vs do
begin
   if l[i]=1 then s1:=s1+1;
   if l[i]=2 then s2:=s2+1;
   if l[i]=3 then s3:=s3+1;
   if l[i]=4 then s4:=s4+1;
   if l[i]=5 then s5:=s5+1;
   if l[i]=6 then s6:=s6+1;
   if l[i]=7 then s7:=s7+1;
```

```
    if l[i]=8 then s8:=s8+1;
    if l[i]=9 then s9:=s9+1;
    if l[i]=10 then s10:=s10+1;
    if l[i]=11 then s11:=s11+1;
    if l[i]=12 then s12:=s12+1;
    if l[i]=13 then s13:=s13+1;
    if l[i]=14 then s14:=s14+1;
    if l[i]=15 then s15:=s15+1;
    if l[i]=16 then s16:=s16+1;
    if l[i]=17 then s17:=s17+1;
    if l[i]=18 then s18:=s18+1;
    if l[i]=19 then s19:=s19+1;
    if l[i]=20 then s20:=s20+1;
    if l[i]=21 then s21:=s21+1;
    if l[i]=22 then s22:=s22+1;
    if l[i]=23 then s23:=s23+1;
    if l[i]=24 then s24:=s24+1;
    if l[i]=25 then s25:=s25+1;
    if l[i]=26 then s26:=s26+1;
    if l[i]=27 then s27:=s27+1;
    if l[i]=28 then s28:=s28+1;
    if l[i]=29 then s29:=s29+1;
    if l[i]=30 then s30:=s30+1;
end;
ss[1]:=s1;
ss[2]:=s2;
ss[3]:=s3;
ss[4]:=s4;
ss[5]:=s5;
ss[6]:=s6;
ss[7]:=s7;
ss[8]:=s8;
```

```
    ss[9]:=s9;
    ss[10]:=s10;
    ss[11]:=s11;
    ss[12]:=s12;
    ss[13]:=s13;
    ss[14]:=s14;
    ss[15]:=s15;
    ss[16]:=s16;
    ss[17]:=s17;
    ss[18]:=s18;
    ss[19]:=s19;
    ss[20]:=s20;
    ss[21]:=s21;
    ss[22]:=s22;
    ss[23]:=s23;
    ss[24]:=s24;
    ss[25]:=s25;
    ss[26]:=s26;
    ss[27]:=s27;
    ss[28]:=s28;
    ss[29]:=s29;
    ss[30]:=s30;
    for i:=1 to vs do
    begin
       StringGrid4.Cells[1,i]:=FloatTostr(l[i]);
       StringGrid4.Cells[0,i]:=FloatTostr(i);
    end;
    for k:=1 to n do
       StringGrid2.cells[k,n+1]:=FloatTostr(ss[k]);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
```

```
 begin
  thata:=0;
  shata:=0;
  for k:=1 to n do
  begin
     wmat[k]:=0;
     stop[k]:=0;
  end;
  for k:=1 to n do
  begin
     for i:=1 to vs do
     begin
        if l[i]=k then
        begin
           for j:=1 to 10 do
           wmat[k]:=wmat[k]+(veri[i][j]*we[k][j]);
        end;
     end;
  shata:=shata+wmat[k]/10;
  end;
  thata:=shata/81;
  edit1.Text:=FloatToStr(thata);
 end;
end;
end.
```

Ek 2: Uzaklığa Dayalı Uyarlamalı YÖA ile Sınıflandırma Yapan Programın Kaynak Kodu

```
unit OPSS308;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, DB, StdCtrls, DBTables, Mask, DBCtrls, Grids, DBGrids;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label12: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Label13: TLabel;
    Timer2: TTimer;
    Edit3: TEdit;
    Label17: TLabel;
    Button4: TButton;
    StringGrid4: TStringGrid;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    StringGrid5: TStringGrid;
    Label3: TLabel;
    StringGrid6: TStringGrid;
    StringGrid3: TStringGrid;
    Button5: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Table3: TTable;
    DataSource3: TDataSource;
```

```
    Table3X1: TFloatField;
    Table3X2: TFloatField;
    Table3X3: TFloatField;
    Table3X4: TFloatField;
    Table3X5: TFloatField;
    Table3X6: TFloatField;
    Table3X7: TFloatField;
    Table3X8: TFloatField;
    Table3X9: TFloatField;
    Table3X10: TFloatField;
    Label5: TLabel;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
Form1: TForm1;
cl:string;
n,z,sr,sayac,k,sayac2,j,r,i,tt,vs,kw,gkw,s,t,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s
14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32,s33,s34,
s35,s36,s37,s38,s39,s40,s41,s42,s43,s44,s45,s46,s47,s48,s49,s50:integer;
gwe,gwy,deltaw,gdeltaw,dd,we,wy:array [1..50,1..50] of real;
gveri,veri,cluster,gcluster:array [1..100,1..50] of real;
```

```
guwe,guwy,uwe,uwy,gstop,gwmat,stop,ud,wmat,bh,ss,gd,ggd,dtoplam,dk,d,ggdtopla
m,gdtoplam,stoplam,hata,eghata,x,y,gy,gx,ortwe:array [1..50]of real;
max,min,gmax,m,ud1,w,topbh,mariot,degis,dktop,gthata,gshata,shata,thata,mind,gdt
op:real;
sk:array [1..100,1..50,1..10]of real;
mak,mini,gmak,gmin,l,gl:array[1..100]of real;
uveri:array[1..100] of real;
a:array [1..50]of integer;
implementation
        {$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Halt;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer2.Enabled:=False;
  for k:=1 to n do
  begin
    stoplam[k]:=0;
    bh[k]:=0;
  end;
  w:=0;
  s1:=0;s2:=0;s3:=0;s4:=0;s5:=0;s6:=0;s7:=0;s8:=0;s9:=0;s10:=0;
  s11:=0;s12:=0;s13:=0;s14:=0;s15:=0;s16:=0;s17:=0;s18:=0;s19:=0;s20:=0;
  s21:=0;s22:=0;s23:=0;s24:=0;s25:=0;s26:=0;s27:=0;s28:=0;s29:=0;s30:=0;
  for j:=1 to 10 do
  ss[j]:=0;
  s:=0;
  sr:=1;
  thata:=0;
  shata:=0;
```

```
sayac:=0;
sayac2:=0;
t:=0;
a[t]:=0;
//sınıf sayısını gir.
cl:=inputbox('Sınıf Sayısı','Sınıf Sayısını Giriniz','');
n:=StrToInt(cl);
// table daki verileri çek.
vs:=81;
table3.Open;
for i:=1 to vs do
begin
  gveri[i][1]:=StrToFloat(Table3X1.Text);
  gveri[i][2]:=StrToFloat(Table3X2.Text);
  gveri[i][3]:=StrToFloat(Table3X3.Text);
  gveri[i][4]:=StrToFloat(Table3X4.Text);
  gveri[i][5]:=StrToFloat(Table3X5.Text);
  gveri[i][6]:=StrToFloat(Table3X6.Text);
  gveri[i][7]:=StrToFloat(Table3X7.Text);
  gveri[i][8]:=StrToFloat(Table3X8.Text);
  gveri[i][9]:=StrToFloat(Table3X9.Text);
  gveri[i][10]:=StrToFloat(Table3X10.Text);
  table3.Next;
end;
Table3.close;
r:=0;
Randomize;
for k:=1 to n do
begin
   r:=random(80)+1;
   for j:=1 to 10 do
   gwe[k][j]:=gveri[r][j];
```

```
  end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  for k:=1 to n do
  begin
    for j:=1 to 10 do
    begin
      StringGrid2.Cells[0,k]:=IntToStr(k);
      StringGrid2.cells[j,k]:=FloatToStrF(gwe[k][j],ffgeneral,5,10);
      StringGrid2.Cells[k,0]:=IntToStr(k);
    end;
  end;
  sayac:=sayac+1;
  if sayac>10000 then
  begin
     Timer2.Enabled:=False;
  end
  else
  begin
    Edit3.Text:=IntToStr(sayac);
    Randomize;
    //random x vektörünü seç ve ekrana yazdır.
    z:=random(80)+1;
    for j:=1 to 10 do
    StringGrid1.Cells[j,1]:=FloatToStrF(gveri[z][j],ffgeneral,5,10);
    StringGrid1.Cells[0,1]:=IntToStr(z);
    for j:=1 to 10 do
    StringGrid1.Cells[j,0]:=IntToStr(j);
    //seçilen x vektörünü ağırlık vektörleriyle çarp.
    for k:=1 to n do
    gy[k]:=0;
```

```
for k:=1 to n do
begin
   for j:=1 to 10 do
   gy[k]:=gy[k]+sqr(gveri[z][j]-gwe[k][j]);
end;
//Vektör Çarpımlarının Ekrana Yazdırılması
for k:=1 to n do
begin
   StringGrid5.Cells[k,1]:=FloatToStrF(gy[k],ffgeneral,5,10);
   StringGrid5.Cells[k,0]:=FloatToStr(k);
end;
// Kazanan vektörü belirle.
min:=100000;
for k:=1 to n do
begin
   if min>gy[k] then
   begin
      min:=gy[k];
      kw:=k;
   end;
end;
Edit4.Text:=IntToStr(kw);
//kazanan ağırlık vektörünü güncelle.
m:=0.03;
for j:=1 to 10 do
begin
  gdeltaw[kw][j]:=m*(gveri[z][j]-gwe[kw][j]);
  StringGrid6.Cells[j,2]:=FloatToStrF(gdeltaw[kw][j],ffgeneral,5,10);
  gwy[kw][j]:=gwe[kw][j]+gdeltaw[kw][j];
end;
//eğitim hatasının hesaplanması
for j:=1 to 10 do
```

```
        eghata[kw]:=eghata[kw]+sqr(gwy[kw][j]-gveri[z][j]);
        eghata[kw]:=eghata[kw]/2;
        StringGrid3.Cells[kw,3]:=FloatToStrF(eghata[kw],ffgeneral,5,10);
        //Kazanan ağırlık vektörünü ekrana yazdır.
        for j:=1 to 10 do
        StringGrid2.Cells[j,kw]:=FloatToStrF(gwy[kw][j],ffgeneral,5,10);
        StringGrid2.Cells[0,kw]:=FloatToStr(kw);
        //Ağırlık vektörünü güncelle.
        for j:=1 to 10 do
        gwe[kw][j]:=gwy[kw][j];
        Timer2.Enabled:=True;
    end;
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
    for k:=1 to n do
    begin
        gd[k]:=0;
        ggd[k]:=0;
        dk[k]:=0;
        gdtoplam[k]:=0;
        ggdtoplam[k]:=0;
        gdtop:=0;
    end;
 //   mariot:=0;
   ud1:=0;
    for k:=1 to n do
    begin
        for j:=1 to 10 do
        begin
            StringGrid2.Cells[0,k]:='                ';
            StringGrid2.cells[j,k]:='                ';
```

```
      StringGrid2.Cells[k,0]:=('                    ');
   end;
end;
for i:=1 to vs do
for k:=1 to n do
gcluster[i][k]:=0;
for i:=1 to vs do
begin
   for k:=1 to n do
   for j:=1 to 10 do
   gcluster[i][k]:=gcluster[i][k]+sqr(gveri[i][j]-gwe[k][j]);
end;
for i:=1 to vs do
gmin[i]:=100000;
for i:=1 to vs do
begin
   for k:=1 to n do
   if gmin[i]>gcluster[i][k]then
   begin
      gmin[i]:=gcluster[i][k];
      gl[i]:=k;
   end;
end;
mind:=100000;
dktop:=0;
for k:=1 to n do
gd[k]:=0;
for k:=1 to n do
begin
   gdtoplam[k]:=0;
   for i:=1 to vs do
   begin
```

```
      if gl[i]=k then
      begin
        for j:=1 to 10 do
        gdtoplam[k]:=gdtoplam[k]+sqr(gveri[i][j]-gwe[k][j])
      end;
    end;
    gd[k]:=gdtoplam[k]/10;
end;
for k:=1 to n do
begin
    StringGrid3.Cells[k,2]:=FloatToStrF(gd[k],ffgeneral,4,10);
    StringGrid3.Cells[k,0]:=IntToStr(k);
end;
t:=0;
if n>=14 then
begin
    tt:=1;
    for k:=1 to n do
    begin
      if tt<6 then
      begin
        for k:=1 to n do
        begin
          if mind>gd[k]then
          begin
            mind:=gd[k];
            ud[tt]:=k;
          end;
        end;
        mind:=100000;
        for k:=1 to n do
        begin
```

```
            if ud[tt]<>k then
            begin
               t:=t+1;
               for j:=1 to 10 do
               gwe[t][j]:=gwe[k][j];
               gd[t]:=gd[k];
            end;
         end;
         tt:=tt+1;
      end;
   end;
   n:=n-5;
   sayac:=0;
   Timer2.Enabled:=True;
end;
mind:=100000;
t:=0;
if (n<=10) and (n>3) then
begin
   for k:=1 to n do
   begin
      if mind>gd[k]then
      begin
         mind:=gd[k];
         ud1:=k;
      end;
   end;
   for k:=1 to n do
   begin
      if ud1<>k then
      begin
         t:=t+1;
```

```
            for j:=1 to 10 do
            gwe[t][j]:=gwe[k][j];
          end;
      end;
      n:=n-1;
      sayac:=0;
      Timer2.Enabled:=True;
   end;
   if n<3 then
   Timer2.Enabled:=False;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
   s1:=0;s2:=0;s3:=0;s4:=0;s5:=0;s6:=0;s7:=0;s8:=0;s9:=0;s10:=0;
   s11:=0;s12:=0;s13:=0;s14:=0;s15:=0;s16:=0;s17:=0;s18:=0;s19:=0;s20:=0;
   s21:=0;s22:=0;s23:=0;s24:=0;s25:=0;s26:=0;s27:=0;s28:=0;s29:=0;s30:=0;
   for k:=1 to n do
   ss[k]:=0;
   for i:=1 to vs do
   for k:=1 to n do
   gcluster[i][k]:=0;
   for i:=1 to vs do
   begin
      for k:=1 to n do
      for j:=1 to 10 do
      gcluster[i][k]:=gcluster[i][k]+sqr(gveri[i][j]-gwe[k][j]);
   end;
   for i:=1 to vs do
   mini[i]:=100000;
   for i:=1 to vs do
   begin
      for k:=1 to n do
```

```
    if mini[i]>gcluster[i][k]then
    begin
      mini[i]:=gcluster[i][k];
      gl[i]:=k;
    end;
end;
for i:=1 to vs do
begin
    if gl[i]=1 then s1:=s1+1;
    if gl[i]=2 then s2:=s2+1;
    if gl[i]=3 then s3:=s3+1;
    if gl[i]=4 then s4:=s4+1;
    if gl[i]=5 then s5:=s5+1;
    if gl[i]=6 then s6:=s6+1;
    if gl[i]=7 then s7:=s7+1;
    if gl[i]=8 then s8:=s8+1;
    if gl[i]=9 then s9:=s9+1;
    if gl[i]=10 then s10:=s10+1;
    if gl[i]=11 then s11:=s11+1;
    if gl[i]=12 then s12:=s12+1;
    if gl[i]=13 then s13:=s13+1;
    if gl[i]=14 then s14:=s14+1;
    if gl[i]=15 then s15:=s15+1;
    if gl[i]=16 then s16:=s16+1;
    if gl[i]=17 then s17:=s17+1;
    if gl[i]=18 then s18:=s18+1;
    if gl[i]=19 then s19:=s19+1;
    if gl[i]=20 then s20:=s20+1;
    if gl[i]=21 then s21:=s21+1;
    if gl[i]=22 then s22:=s22+1;
    if gl[i]=23 then s23:=s23+1;
    if gl[i]=24 then s24:=s24+1;
```

```
    if gl[i]=25 then s25:=s25+1;
    if gl[i]=26 then s26:=s26+1;
    if gl[i]=27 then s27:=s27+1;
    if gl[i]=28 then s28:=s28+1;
    if gl[i]=29 then s29:=s29+1;
    if gl[i]=30 then s30:=s30+1;
end;
ss[1]:=s1;
ss[2]:=s2;
ss[3]:=s3;
ss[4]:=s4;
ss[5]:=s5;
ss[6]:=s6;
ss[7]:=s7;
ss[8]:=s8;
ss[9]:=s9;
ss[10]:=s10;
ss[11]:=s11;
ss[12]:=s12;
ss[13]:=s13;
ss[14]:=s14;
ss[15]:=s15;
ss[16]:=s16;
ss[17]:=s17;
ss[18]:=s18;
ss[19]:=s19;
ss[20]:=s20;
ss[21]:=s21;
ss[22]:=s22;
ss[23]:=s23;
ss[24]:=s24;
ss[25]:=s25;
```

```
    ss[26]:=s26;
    ss[27]:=s27;
    ss[28]:=s28;
    ss[29]:=s29;
    ss[30]:=s30;
    for i:=1 to vs do
    begin
      StringGrid4.Cells[1,i]:=FloatTostr(gl[i]);
      StringGrid4.Cells[0,i]:=FloatTostr(i);
    end;
    for k:=1 to n do
    StringGrid2.cells[k,n+1]:=FloatTostr(ss[k]);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  begin
    thata:=0;
    shata:=0;
    for k:=1 to n do
    begin
      wmat[k]:=0;
      stop[k]:=0;
    end;
    for k:=1 to n do
    begin
      for i:=1 to vs do
      begin
        if gl[i]=k then
        begin
          for j:=1 to 10 do
          wmat[k]:=wmat[k]+sqr(gveri[i][j]-gwe[k][j]);
        end;
```

```
    ss[26]:=s26;
    ss[27]:=s27;
    ss[28]:=s28;
    ss[29]:=s29;
    ss[30]:=s30;
    for i:=1 to vs do
    begin
       StringGrid4.Cells[1,i]:=FloatTostr(gl[i]);
       StringGrid4.Cells[0,i]:=FloatTostr(i);
    end;
    for k:=1 to n do
    StringGrid2.cells[k,n+1]:=FloatTostr(ss[k]);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  begin
    thata:=0;
    shata:=0;
    for k:=1 to n do
    begin
       wmat[k]:=0;
       stop[k]:=0;
    end;
    for k:=1 to n do
    begin
       for i:=1 to vs do
       begin
          if gl[i]=k then
          begin
             for j:=1 to 10 do
             wmat[k]:=wmat[k]+sqr(gveri[i][j]-gwe[k][j]);
          end;
```

```
    ss[26]:=s26;
    ss[27]:=s27;
    ss[28]:=s28;
    ss[29]:=s29;
    ss[30]:=s30;
    for i:=1 to vs do
    begin
       StringGrid4.Cells[1,i]:=FloatTostr(gl[i]);
       StringGrid4.Cells[0,i]:=FloatTostr(i);
    end;
    for k:=1 to n do
    StringGrid2.cells[k,n+1]:=FloatTostr(ss[k]);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  begin
    thata:=0;
    shata:=0;
    for k:=1 to n do
    begin
       wmat[k]:=0;
       stop[k]:=0;
    end;
    for k:=1 to n do
    begin
       for i:=1 to vs do
       begin
          if gl[i]=k then
          begin
             for j:=1 to 10 do
             wmat[k]:=wmat[k]+sqr(gveri[i][j]-gwe[k][j]);
          end;
```

```
      end;
    shata:=shata+wmat[k]/10;
    end;
    thata:=shata/81;
    edit1.Text:=FloatToStr(thata);
  end;
end;
end.
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	: Nida GÖKÇE
Doğum Yeri ve Tarihi	: Polatlı / 21.11.1976

EĞİTİM VE AKADEMİK BİLGİLER

Lise	: Polatlı Lisesi 1990-1993
Lisans	: 1993 – 1995 Ankara Üniversitesi, Fen Fakültesi, Astronomi ve Uzay Bilimleri
Lisans	: 1996 – 2001 Muğla Üniversitesi, Fen Edebiyat Fakültesi, İstatistik ve Bilgisayar Bilimleri
Y. Lisans	: 2002 – 2005 Muğla Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik ve Bilgisayar Bilimleri ABD.
Yabancı Dil	: İngilizce

MESLEKİ BİLGİLER

2002 – 2003	: Muğla Üniversitesi, Fen Bilimleri Enstitüsü, Araştırma Görevlisi
2003 -	: Muğla Üniversitesi, Fen Edebiyat Fakültesi, Araştırma Görevlisi