# TIBBİ BİLGİLERİN YAPAY SİNİR AĞLARI KULLANARAK İNCELENMESİ

Alper Kürşat UYSAL

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Ağustos – 2005

**JÜRİ VE ENSTİTÜ ONAYI**

**Alper Kürşat UYSAL'ın "Tıbbi Bilgilerin Yapay Sinir Ağları Kullanarak İncelenmesi" başlıklı Bilgisayar Mühendisliği Anabilim Dalındaki, Yüksek Lisans tezi.....tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.**

| | Adı-Soyadı | İmza |
|---|---|---|
| Üye (Tez Danışmanı) | : Prof. Dr. Ali GÜNEŞ | …… |
| Üye | : Doç. Dr. Ahmet BABANLI | ..…..…… |
| Üye | : Yard. Doç. Dr. Serkan TAPKIN | ..……. |

**Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.**

**Enstitü Müdürü**

# ÖZET

**Yüksek Lisans Tezi**

**TIBBİ BİLGİLERİN YAPAY SİNİR AĞLARI KULLANARAK İNCELENMESİ**

**ALPER KÜRŞAT UYSAL**

**Anadolu Üniversitesi**
**Fen Bilimleri Enstitüsü**
**Bilgisayar Mühendisliği Anabilim Dalı**

**Danışman: Prof. Dr. Ali GÜNEŞ**
**2005, 97 sayfa**

**Tıbbi bilgilerin bilgisayar analizleri eskiden beri yapay sinir ağları algoritmalarıyla yapılmaktadır. Bunun sebebi bu tip problemlerin çözümünde insan tecrübesi ve bilgi birikiminin son derece önemli olmasıdır. Bu çalışmada Osmangazi Üniversitesi Nöroloji Anabilim Dalı'ndan alınan baş ağrısı hastalığı verileri böyle bir teşhis problemini çözmek için kullanılmıştır.**

**Problem, alınan hasta verilerini 4 farklı hastalık türüne ayrılacak şekilde sınıflandırmaktır. Bu hastalık türleri aurasız migren, auralı migren, gerilim tipi baş ağrısı ve aurasız migrenden deforme baş ağrısıdır. Yapay sinir ağlarının bu tip problemleri çözmek için son derece uygun olduğu bilinmektedir. 3 öğreticili ve 1 öğreticisiz yöntemi kapsayan 4 farklı yapay sinir ağı metodu kullanılmıştır. Bunlar perseptron, geri yayılım ağları, vektör nicemleme ağları ve kendini düzenleyen ağlardır. Sonuç olarak geri yayılım ağlarının baş ağrısı hastalıklarını sınıflandırmada en etkili metot olduğu görülmüştür.**

**Anahtar Kelimeler: Yapay Sinir Ağları, Baş Ağrısı Hastalıkları, Teşhis Problemleri**

## ABSTRACT

**Master of Science Thesis**

**ANALYSIS OF MEDICAL INFORMATION USING NEURAL NETWORKS**

**ALPER KÜRŞAT UYSAL**

**Anadolu University**
**Graduate School of Applied Sciences**
**Computer Engineering Program**

**Supervisor: Prof. Dr. Ali GÜNEŞ**
**2005, 97 pages**

**The computer analyses of medical data to solve diagnosis problems are historically performed by neural network algorithms. This is due to the importance of human experience and information aggregations in solving these types of problems. In this work, headache disease data taken from Osmangazi University Neurology Department are used to solve such a diagnosis problem.**

**The problem is to classify the given patient data into four types of diseases: Migraine without aura, migraine with aura, tension-type headache and transformed migraine. Since neural networks are well known to solve such classification problems, 4 methods consisting of 3 supervised and 1 unsupervised method are used. These are perceptron, backpropagation, learning vector quantization and self-organizing maps. As a result, it was seen that the most efficient neural network method in classification of headache disease data is the backpropagation network.**


**Keywords: Neural Networks, Headache Diseases, Diagnostic Problems**

## TEŞEKKÜR

Bu tezin hazırlanmasında bana yol gösteren danışmanım Anadolu Üniversitesi'nden Sayın Prof. Dr. Ali GÜNEŞ'e, bu konudaki birikim ve deneyimlerini benimle paylaşarak tezime çok önemli katkılarda bulunan Anadolu Üniversitesi'nden Sayın Doç. Dr. Ahmet BABANLI'ya, hastalık belirtilerinin çıkarılmasında ve hastalıklarla eşleştirilmesinde değerli vaktini bizim için harcayan Osmangazi Üniversitesi Tıp Fakültesi'nden Sayın Dr. Demet İLHAN'a, teknik yardımlarından istifade ettiğim arkadaşlarım Anadolu Üniversitesi Bilgisayar Mühendisliği Bölümü'nden Sayın Araş.Gör.Muzaffer DOĞAN'a, Sayın Araş.Gör.Ahmet ARSLAN'a ve tez çalışmam sırasında bana maddi manevi her türlü desteğini sunan aileme teşekkür ederim.

Alper Kürşat UYSAL
akuysal@anadolu.edu.tr
Ağustos - 2005

# İÇİNDEKİLER

# ŞEKİLLER DİZİNİ

# ÇİZELGELER DİZİNİ

# SİMGELER ve KISALTMALAR DİZİNİ

| | |
|---|---|
| IHS | Uluslarası Baş ağrısı Derneği |
| YSA | Yapay Sinir Ağları |
| SOM | Kendini Düzenleyen Ağlar |
| LVQ | Destekleyici Öğrenmeli Vektör Nicemleme Ağları |

# 1. GİRİŞ VE AMAÇ

Yapay sinir ağları kabaca insanın düşünme yapısının sistematik bir şekilde bilgisayar ortamına aktarılmasıdır. Bu sebepten yapay sinir ağlarının uygulama alanları arasında hastalık teşhisleri de önemli yer tutmaktadır. Doktorlar teşhis için kararlarını sayıca çok belirti ve bunlar arasındaki ilişkiler sayesinde verebilmektedir. Bu tezde, seçilen belli bir hastalık tipi ve bunun 4 farklı türü arasında hastalık teşhisi çalışması yapılmıştır. Yapay sinir ağları ve bunun yanı sıra hastalık teşhisi söz konusu olduğunda sınıflandırma probleminden bahsedilebilmektedir.

## 1.1. Problemin Tanımı

Sınıflandırma için kullanılan farklı yöntemler bulunmaktadır. Yapay sinir ağları ana başlık olarak bunlardan sadece bir tanesidir. Bu tezin konusu itibariyle çalışmada metot olarak yapay sinir ağları, uygulama alanı olarak ta nöroloji ana bilim dalı altında toplanan baş ağrısı hastalıkları alınmıştır. Baş ağrısı hastalıklarının diğer nöroloji hastalıklarından en önemli farkı ağırlıklı olarak belirtiler eşliğinde teşhis yapılabilmesinin mümkün olmasıdır. Buna karşın nöroloji anabilim dalında önemli bir yere sahip olan Alzheimer, Parkinson gibi hastalıkların teşhisi daha çok beynin belli kısımlarının dijital ortamdaki görüntülerinin incelenmesi şeklinde gerçekleşmektedir [1,2]. Bu gibi bilgisayarlı teşhis yöntemlerinin çıkış amacı insansız teşhisten ziyade insana bağlı olarak ortaya çıkabilecek hataların ve bunlardan doğabilecek yanlış teşhislerin en aza indirgenmesidir. Sonuçta insan faktörü ortaya çıktığında dikkatsizlik veya bazı şeylerin gözden kaçırılabilmesi gibi hatalarla karşılaşmak mümkündür.

Baş ağrısı hastalıkları 13 ana başlık altında toplanmıştır. Bu hastalıklar da kendi içlerinde toplam 127 alt gruba ayrılmıştır. Baş ağrısı hastalıkları temel olarak iki gruba ayrılabilir. İlk grup, nöroloji anabilim dalını doğrudan ilgilendiren ve başka bir hastalık veya tıbbi durumla ilgili olmayan ve **primer (birincil) baş ağrıları** şeklinde isimlendirilen baş ağrılarıdır. Bu tezde incelenen 4 baş ağrısı tipi bu gruba aittir. İkinci grup baş ağrıları ise darbeler, göz hastalıkları, kulak burun boğaz hastalıkları, boyun hastalıkları, beyin kanaması, beyin tümörü gibi tıbbi

durumlar sebebiyle oluşan ve **sekonder (ikincil) baş ağrıları** şeklinde isimlendirilen baş ağrılarıdır [3].

Bu çalışmada Osmangazi Üniversitesi Nöroloji Ana Bilim Dalı'ndan alınan 4 farklı baş ağrısı hastalığı verileri kullanılmış ve farklı yapay sinir ağları yöntemlerinin denenmesi sonucunda daraltılmış bir belirti kümesi üzerinde başarılı sonuçlar elde edilmiştir. Sonuç bölümünde de bunlar arasında karşılaştırılmaya gidilmiştir.

### 1.2. Katkılar

Bu tezde amacımız Osmangazi Üniversitesi Tıp Fakültesi Nöroloji Anabilim Dalı'ndan alınan baş ağrısı hastalıkları verilerini yapay sinir ağları yöntemleriyle incelemektir. Hastalık verileri söz konusu olduğunda inceleme kelimesi hastalıkların teşhisi anlamına gelmektedir. Yapay sinir ağları söz konusu olduğunda da bu bir sınıflandırma problemi olarak düşünülebilir. Sonuç olarak eldeki hasta verileri ve bunlara karşılık gelen teşhis sonuçları eşliğinde gelecek yeni hastaların teşhisi için bir sistem geliştirilmiştir. Bunu yaparken matlab ortamından yararlanılmıştır. Daha somut bir bakış açısı olması itibariyle bu teşhis sistemi için bir de kullanıcı grafik ara yüzü oluşturulmuştur.

### 1.3. Tez Organizasyonu

Bölüm 2'de sınıflandırmanın genel anlamda ne olduğu açıklanmış ve sınıflandırma için kullanılan yöntemlerden bahsedilmiştir. Bölüm 2'nin sonunda bu çalışmada kullanılan yöntem olan yapay sinir ağları da kısaca ele alınmıştır. Bölüm 3'te yapay sinir ağları hakkında detaylı bilgi verilmiştir. Bu kısım yapay sinir ağlarının ne olduğundan başlayarak uygulama alanlarına kadar olan geniş bir bilgi kümesini içermektedir. Bölüm 4'te bu çalışmanın ana teması olan baş ağrısı hastalıkları hakkında genel bilgiler verilmiştir. Verilen bilgiler ağırlıklı olarak bu çalışmada kullanılan 4 farklı hastalık çeşidi üzerinedir. Bölüm 5'te bu çalışmada yapılan uygulamalar ve alınan sonuçlardan bahsedilmektedir. Son kısmında da geliştirilen uygulama ara yüzü hakkında bilgi verilmiştir. Bölüm 6'da elde edilen sonuçlar değerlendirilmiş ve gelecekte bu konuda hangi çalışmaların yapılabileceğinden bahsedilmiştir.

# 2. SINIFLANDIRMA ve SINIFLANDIRMA İÇİN KULLANILAN YÖNTEMLER

## 2.1. Sınıflandırma Nedir?

Yüzeysel olarak sınıflandırma terimi belli karar veya tahminin mevcut olan bilginin üzerine yapıldığı herhangi bir durumu ifade etmektedir. Sınıflandırma prosedürü ise tekrarlanan yeni durumlardaki benzer kararların verilebilmesi için kullanılan muntazam bir metottur. Problem, her bir yeni durumun gözlenen belirleyici niteliklere dayanılarak önceden belirlenmiş sınıflara atanacağı bir prosedürün oluşturulmasıyla ilgilidir. Doğru sınıfların bilindiği bir veri kümesinden sınıflandırma prosedürünün oluşturulması genellikle örüntü tanıma (pattern recognition), ayrım (discrimination) veya öğreticili öğrenme (supervised learning) yöntemleriyle mümkün olmaktadır. Mali durum ve diğer bilgilerine göre insanlarının kredi alabilme şartını sağlayıp sağlamadığının belirlenmesi, belirleyici test sonuçları gelmeden önce acil tedavi yöntemlerinin seçimi için hastalıklara başlangıç teşhisinin konması gibi sınıflandırma görevinin temel içerik olduğu durumlar vardır. Nitekim bilimde, endüstride ve ticarette çıkan acil problemlerin bazıları karmaşık ve çoğunlukla büyük veri toplulukları kullanan sınıflandırma veya karar problemleri olarak ele alınır [4].

## 2.2. Sınıflandırma Üzerine Farklı Bakış Açıları

Sınıflandırma işlerine yönelik olarak istatistiksel yöntemler, makine öğrenimi yöntemleri ve yapay sinir ağları yöntemleri olmak üzere temelde 3 farklı çözüm yöntemi kullanılmaktadır.

### 2.2.1. İstatistiksel yaklaşımlar

İstatistik topluluğu içinde sınıflandırma üzerine iki çalışma aşaması tanınmıştır. Birincisi, Fisher'ın **doğrusal ayrım (linear discrimination)** üzerine önceki çalışmalarının bir uzantısı olan "**klasik"** aşamadır. İkincisi karışık sınıf modellerinde başarılı olan "**modern**" sınıflandırma aşamasıdır. Birçoğu, sınıflandırma kuralı belirlemek için her örneğin belli bir sınıfa yakınlığını belirlemeye çalışır. İstatistiksel yaklaşımlar genelde açık bir olasılık teorisi

üzerine oluşturulmuştur. Basit bir sınıflandırma yerine herhangi bir sınıfa ait olma ihtimalini verir [4].

### 2.2.2. Makine öğrenimi

Makine öğrenimi, genellikle mantıksal veya ikili işlemlere dayalı otomatik hesaplama prosedürlerini içine alan, bir işi birçok örnek duruma göre öğrenen bir yöntemdir. Sınıflandırma sonuçlarının birkaç sıralı mantıksal adımdan çıkarıldığı karar ağaçları yaklaşımı üzerine ilgi yoğunlaşmıştır. Karar ağaçları yöntemi ile elde yeterli miktarda veri olduğu takdirde çok karmaşık problemler temsil edilebilmektedir. Genetik algoritmalar ve mantıksal tümevarım prosedürleri (inductive logic procedures) gibi diğer teknikler de halen aktif olarak geliştirilmektedir. Makine öğrenimi, insanlar tarafından kolayca anlaşılabilecek sınıflandırma ifadeleri üretmeyi amaçlar. Bu sınıflandırma ifadeleri insan düşünme sistemini taklit eden karar süreci şeklinde olmalıdır. İstatistiksel yaklaşımlarda olduğu gibi önceki bilgilerden gelişim sürecinde yararlanılmalıdır fakat işlemler insan müdahalesi olmadan gerçekleşecektir [4].

### 2.2.3. Yapay sinir ağları

Yapay sinir ağları alanının ortaya çıkışının temeli, insan beynini anlamak ve ona benzer bir sistem oluşturmaya dayanmaktadır. Teknoloji takibi bilimin, mühendisliğin birçok alanında akademik ve endüstriyel araştırmacılar için itici bir kuvvet olmaktadır. Yapay sinir ağlarında da makine öğreniminde olduğu gibi zekâyı kendi üretme gösterişi ile oluşan bir teknolojik ilerleme heyecanı vardır. Yapay sinir ağları birbirine bağlı düğüm noktalarından oluşan katmanlar içerir. Her düğüm kendi girdisinin bir doğrusal olmayan fonksiyonunu üretir. Bir düğüm noktasına gelen girdi başka düğüm noktalarından veya doğrudan girdi verilerinden gelebilir. Ayrıca bazı düğüm noktaları yapay sinir ağının çıktısıyla birlikte de düşünülebilir. Bu yüzden bütün ağ herhangi bir derece doğrusal olmayan fonksiyonları birleştiren ve çok genel fonksiyonların modellenmesine imkân tanıyan karmaşık ilişkiler kümesi olarak tasvir edilebilir. Basit ağlarda, bir düğümün çıkışı birbiriyle ilişkili düğümlerden oluşan katmanlar arasında mesajların iletilmesi gibi bir şekilde diğer düğümü besler. Çok karmaşık davranışlar, son çıkış düğümlerinin önceki düğümlere bağlı olduğu ve sistemin

geri besleme ile yüksek derecede doğrusal olmayan sistem olma özelliğine sahip olduğu ağlar tarafından modellenebilir. Yapay sinir ağlarının, beyindeki nöron ağlarının belirli büyüklükteki davranışlarını yansıttığı kanıtlanmıştır. Yapay sinir ağları yaklaşımı bazı istatistiksel yöntemlerin karmaşıklığı ile makine öğrenimi yönteminin insan beynine benzeme amacını birleştirmektedir [4].

# 3. YAPAY SİNİR AĞLARI

## 3.1. Yapay Sinir Ağları Hakkında Genel Bilgiler

### 3.1.1. Yapay sinir ağları nedir?

Yapay sinir ağları, beynin sinirsel yapısına dayalı kabataslak elektronik modellerdir. Beyin, temel anlamda tecrübeye dayalı bir şekilde öğrenmektedir. Bu beyin modellemesi makine çözümleri üretmek için küçük bir teknik yöntem sağlamaktadır [5].

Biyolojiden esinlenen bu hesaplama yöntemleri bilgisayar endüstrisinde bir sonraki büyük ilerleme olarak düşünülebilir. Basit hayvan beyinleri bile bilgisayar için şu anda gerçekleştirilmesi mümkün olmayan fonksiyonlara sahip olabilmektedir. Bilgisayarlar, karışık matematiksel işlemleri yapabilme ve hafızada tutabilme gibi alışılmış hareketleri çok iyi yapabilmektedirler. Fakat geçmişteki benzerlerine bakılarak genelleştirilmesi zor olup gelecekte karşılaşılabilecek basit örüntüleri bile tanımada sıkıntı yaşarlar [5].

Biyolojik araştırmalardaki gelişmeler doğal düşünme mekanizmasının anlaşılmaya başlandığının bir belirtisidir. Bu araştırmalar beynin bilgileri örüntü halinde sakladığını göstermektedirler. Bu örüntülerden bazıları çok karmaşıktır ve bize farklı yüzleri farklı açılardan tanıma yeteneği sağlar. Bilgileri örüntü olarak saklama, bu örüntülerden yararlanma ve sonra problemleri çözme bilgisayar dalında yeni bir çalışma sahası ortaya çıkmasına neden olmaktadır. Bu alan, geleneksel programlamadan yararlanmamaktadır fakat büyük paralel ağların yaratılmasına ve bu ağların belli problemleri çözmek üzere eğitilmesine katkıda bulunmaktadır [5].

### 3.1.2. Beyin ile mukayese

İnsan beyninin tamamen nasıl çalıştığı hala bir sırdır. Buna rağmen bu ilginç işlemcinin bazı yönleri bilinmektedir. Beynin en temel elementi vücudun diğer kısımları gibi yenilenemeyen özel bir hücre tipidir. Bu hücre tipinin vücudun yavaşça değiştirilemeyen tek parçası olmasının sonucunda bize hatırlamak, düşünmek ve yeni gelişen olaylarda eski olayların tecrübelerinden yararlanmak gibi yetenekleri sağladığı düşünülmektedir. Sayıları 100 milyar kadar olan bu

hücreler nöronlar olarak bilinmektedirler. Bu nöronlardan her biri 200.000'e kadar başka nöronlara bağlanabilmektedir fakat 1000 ile 10000 arasındaki kısmı kendine özgüdür.

İnsan zekâsının gücü sayısız miktardaki bu basit bileşenlerden ve onların arasındaki çoklu bağlantılardan gelmektedir.

Nöronlar tek başlarına oldukça karmaşık yapıdadırlar. Sayısız parçaya, alt sisteme ve kontrol mekanizmalarına sahiptirler. Elektrokimyasal yollardan bilgi aktarmaktadırlar. 100'ün üzerinde farklı sınıfta nöronlar vardır.

### 3.1.3. YSA' nın uygulama alanları

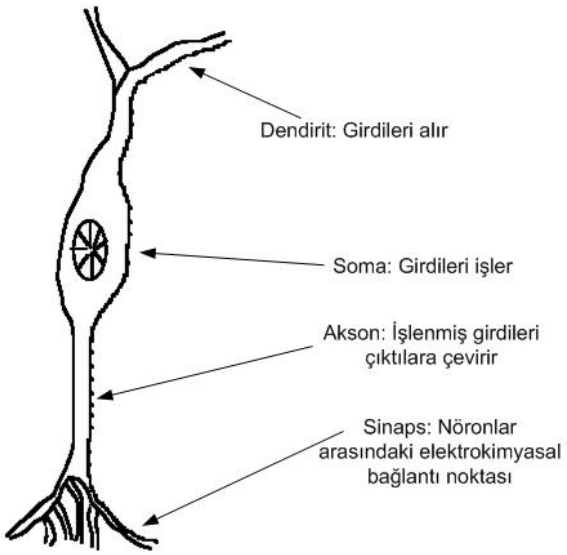
Bugüne kadar yapay sinir ağları, çözümü güç ve karmaşık olan ya da ekonomik olmayan, çok farklı alanlardaki birçok probleme uygulanmış ve genellikle başarılı sonuçlar alınabilmiştir. Başlıca uygulama alanlarını şu şekilde sıralayabiliriz [6]:

1. **Endüstriyel Uygulamalar**:
   - Bir endüstriyel proseste fırınların ürettiği gaz miktarının tahmini
   - İmalatta, ürün tasarımı, proses ve makinelerin bakımı ve hataların teşhisinde görsel kalite kontrolü
   - Kimyasal proseslerin dinamik modellenmesi
   - Otomobillerde otomatik rehber sisteminin geliştirilmesi
   - Araba pistonlarının üretim şartlarının belirlenmesi
2. **Finansal Uygulamalar**:
   - Makro ekonomik tahminler
   - Borsa benzetim çalışmaları endekslerinin tahmin edilmesi
   - Kredi kartları hilelerinin tespiti
   - Banka kredilerinin değerlendirilmesi
   - Risk analizleri
3. **Askeri Uygulamalar**:
   - Hedef tanıma ve takip sistemleri
   - Yeni sensörlerin performans analizleri
   - Radar ve görüntü sinyalleri işleme
   - Mayın detektörleri

4. **Sağlık Uygulamaları**:
   - Solunum hastalıklarının teşhisi
   - EEG ve ECG analizleri
   - Hastalıkların teşhisi ve resimlerden tanınması
   - Kardiovascular sistemlerin modellenmesi ve teşhisi
   - CTG izleme
   - Hamile kadınların karınlarındaki çocukların kalp atışlarının izlenmesi
   - Üroloji uygulamaları (prostat analizleri, sperm analizleri)
5. **Diğer Alanlar**:
   - Uçak parçalarının hata teşhislerinin yapılması
   - Petrol ve gaz aramasının yapılması
   - Hava alanlarındaki bomba detektörleri ve uyuşturucu koklayıcıları
   - Karakter, el yazısı ve imza tanıma sistemleri

### 3.1.4. Nöronlar ve çalışma şekilleri

YSA'nın temel işlevsel elemanları nöronlardır. İnsan bilincini oluşturan bu parça birkaç yeteneği içine almaktadır. Temel olarak bir biyolojik nöron başka kaynaklardan girdi alır, bunları bir şekilde birleştirir, sonucun üzerinde doğrusal olmayan bir takım işlemler yapar ve sonucu çıktı olarak verir [5].



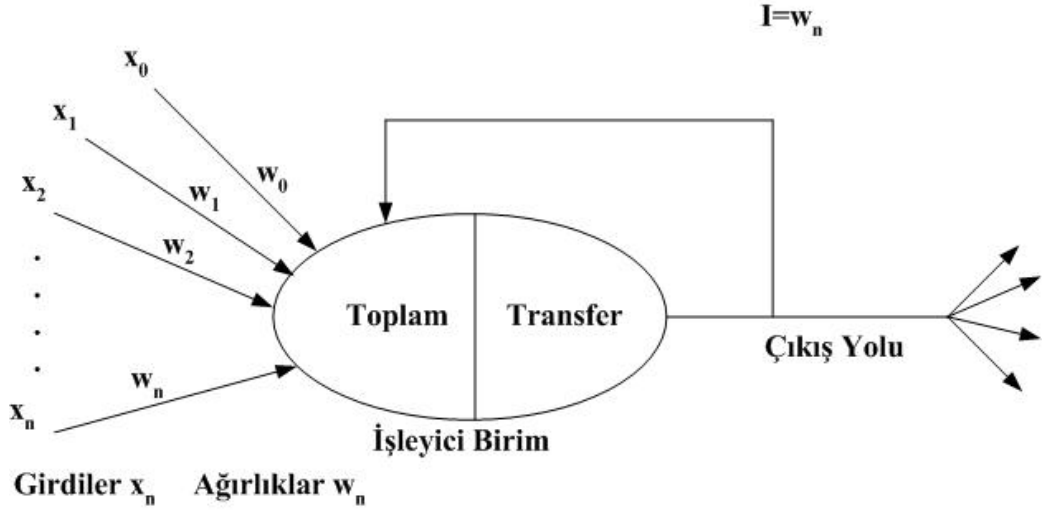

**Şekil 3.1 Biyolojik nöronun yapısı**

İnsanlarda bu basit tipteki nöronun birçok çeşitlemeleri mevcuttur. Fakat bütün biyolojik nöronlar aynı 4 temel bileşene sahiptir. Bu bileşenler biyolojik isimleriyle bilindikleri şekilde dendrit, soma, akson ve sinapslardır. Dendritler, somanın giriş kanalı vazifesi gören saça benzeyen uzantılarıdır. Bu giriş kanalları sinyalleri başka nöronların sinapslarından alır. Soma daha sonra bu sinyalleri zaman içinde işler. Sonunda işlenmiş değeri çıktı haline getirir ve bunu da diğer nöronlara iletilmek üzere akson ve sinapslara verir [5].

Şu andaki deneysel veriler biyolojik nöronların bugünün yapay sinir ağlarında oluşturulan yapay nöronlardan daha karışık olduğuna dair kanıt sağlamıştır. Biyolojinin nöronların anlaşılması için daha fazla imkân sağladığı ve teknolojinin geliştiği sürece yapay sinir ağı tasarımcıları sistemlerini geliştirmeye devam edeceklerdir [5].

Fakat şu anda yapay sinir ağlarının amacı görkemli bir şekilde beyni baştan yaratmak değildir. Bunun aksine yapay sinir ağları araştırmacıları insanların şu ana kadar geleneksel yöntemlerle çözülemeyen problemleri doğal yetenekleriyle nasıl çözdüğünü bulmaya çalışmaktadır.

Bunu yapmak için beynin temel birimi olan nöronlar doğal nöronların 4 farklı fonksiyonunu simule etmektedir.




**Şekil 3.2 Basit bir yapay nöron**

Şekil 3.2'de ağa gelen çeşitli girdiler x(n) matematiksel sembolüyle ifade edilmiştir. Bu girdilerin her biri bir bağlantı ağırlığıyla çarpılmaktadır. Ağırlıklar

w(n) şeklinde ifade edilmektedir. Basit anlamda bu çarpımlar toplanmaktadır ve sonuç üretmek üzere bir transfer fonksiyonuna sokulmaktadır [5].

### 3.1.5. YSA'nın temel bileşenleri

Bir yapay nöronu oluşturan aşağıdaki gibi 7 temel bileşen vardır [5]:

1. **Ağırlık faktörü:** Bir nöron genellikle birçok eşzamanlı girdiler alır. Her bir girdi işleyici birimin toplama fonksiyonunda etki göstermesi için gerekli olan kendi bağıl ağırlığına sahiptir. Bu ağırlıklar biyolojik nöronların sinaptik şiddetlerini değiştiren aynı tipte fonksiyonu gerçekleştirir. Her iki durumda da bazı girdiler diğerlerinden daha önemli olmaktadır. Bir tepki vermek üzere birleştiklerini düşünürsek önemli olanların işleyici birim üzerinde etkileri daha fazladır [5].

   Ağırlıklar, yapay nörona doğru giriş sinyallerinin şiddetini belirleyen ağa uygun katsayılardadırlar. Onlar, girişin bağlantı dayanıklılığının bir ölçüsüdür. Bu dayanıklılıklar çeşitli eğitim setlerine verilen cevaba ve ağ topolojisine göre değiştirilebilir.
2. **Toplama fonksiyonu:** İşleyici birimin işlemindeki ilk adım ağırlıkları belirlenmiş girdilerin toplamını hesaplamaktır. Matematiksel olarak girdiler ve onlara bağlı ağırlıklar ($i_1,i_2,i_3, \ldots,i_n$ ve $w_1,w_2,w_3,\ldots,w_n$) vektördürler. Toplam giriş sinyali bu vektörlerin iç çarpımı şeklindedir. Bu basit toplama fonksiyonu her i vektörünü ilgili w vektörüyle çarpmak ve sonuçta hepsini toplamak şeklinde gerçekleştirilir.

$$Girdi1=i_1*w_1$$
$$Girdi2=i_2*w_2$$
$$.$$
$$.$$
$$GirdiN=i_n*w_n$$
$$Toplam\ Girdi=Girdi1+Girdi2+\ldots+GirdiN$$

Sonuç ise tek bir rakamdır, çok elemanlı bir vektör değildir.

Geometrik olarak, iki vektörün iç çarpımı onların birbirlerine benzerlik ölçütü olarak düşünülebilir. Eğer vektörler aynı yönlüyse, iç çarpım maksimum; eğer vektörler zıt yöndeyse iç çarpımları minimumdur.

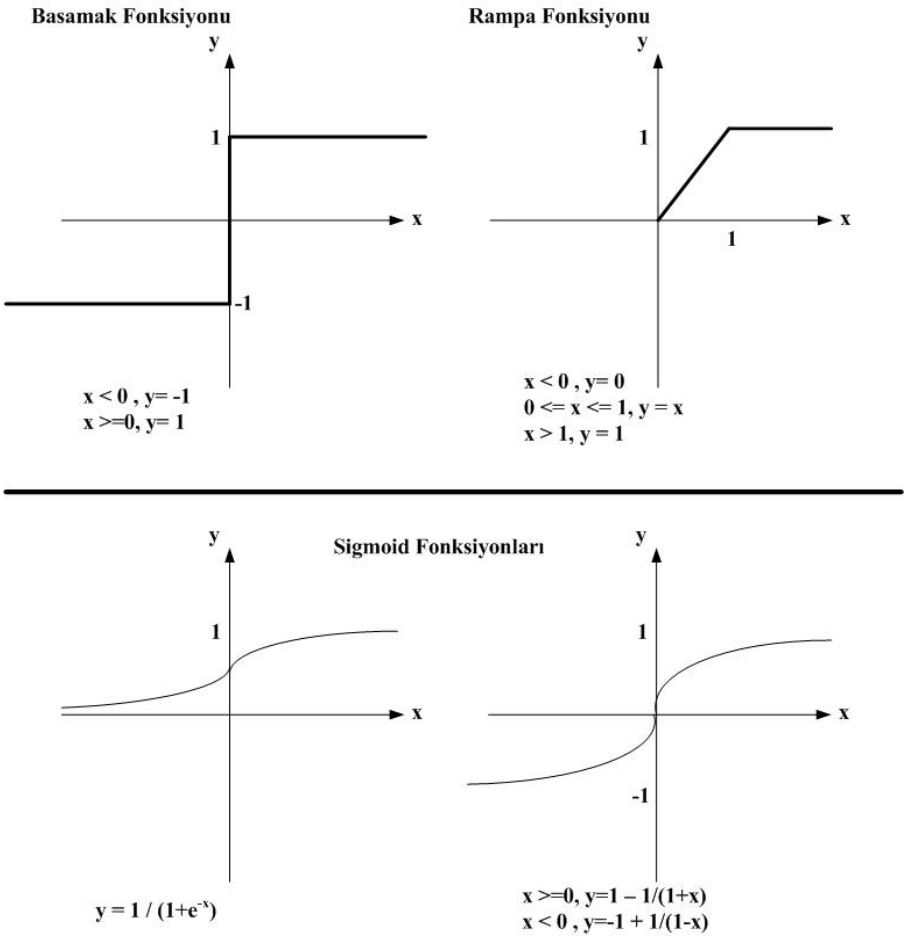
Toplama fonksiyonu basit bir girdi ve onların ağırlıkları çarpımı toplamından çok daha karmaşık olabilir. Girdi ve ağırlık katsayısı transfer fonksiyonuna sokulmadan önce birçok değişik şekillerde birleştirilebilir. Basit çarpım toplamına ek olarak toplama fonksiyonu minimum, maksimum, çoğunluk veya birkaç normalizasyon algoritmasını seçebilir. Yapay girdileri birleştiren özel algoritma seçilen ağ mimarisi tarafından belirlenmektedir [5].

Bazı toplama fonksiyonları transfer fonksiyonuna sokulmadan önce sonuca uygulanan ek bir işleme sahiptir. Bu işlem bazen aktivasyon fonksiyonu olarak isimlendirilir. Bir aktivasyon fonksiyonu kullanmanın amacı zamana bağlı olarak toplam çıktının çeşitlenmesine izin vermektir. Aktivasyon fonksiyonları halen araştırma açısından oldukça sınırlanmıştır. İlaveten bir fonksiyon ayrı ayrı her bir işleyici birimin bileşeni olmaktansa bütün olarak ağın bileşeni olabilir.

3. **Transfer fonksiyonu:** Ağırlıklı toplam olarak adlandırılan toplama fonksiyonunun sonucu algoritmik bir işlem olarak bilinen transfer fonksiyonu aracılığıyla işleyen bir çıktıya dönüştürülür. Transfer fonksiyonu içinde toplam sonucu yapay çıktıyı belirlemek için bir eşik değeriyle karşılaştırılabilir. Eğer toplam, eşik değerinden büyükse işleyici birim bir sinyal üretir. Girdiler ve ağırlık çarpımlarının toplamı eşik değerinden küçükse, sinyal üretilmez. İki çeşit tepki de önemlidir.

   Eşik değeri veya transfer fonksiyonu genellikle doğrusal değildir. Doğrusal fonksiyonlar sınırlıdır çünkü çıktı girdiyle basit bir şekilde orantılıdır. Doğrusal fonksiyonlar çok faydalı değildir.

   Transfer fonksiyonu basitçe toplam sonucunun pozitif veya negatif olmasına göre herhangi bir şey olabilir. Ağ; 1 ve 0, 1 ve -1 veya başka rakamsal kombinasyonlarda çıktı üretebilir. Transfer fonksiyonu bir basamak veya rampa fonksiyonu olabilir.




**Şekil 3.3 Örnek transfer fonksiyonları**

Bir başka çeşit transfer fonksiyonu olan eşik veya rampa fonksiyonu girdiyi belirli sınır aralıkları arasına sokabilir ve bu sınırların dışında bir basamak fonksiyonu gibi davranabilir. Bu fonksiyon minimum ve maksimum değerlerine kırpılmış bir doğrusal fonksiyondur. Bu, onu doğrusal olmayan bir fonksiyon yapar. Diğer bir seçenek te sigmoid veya S-şekilli eğridir. Bu eğri, asimptotlarından bir minimuma ve maksimuma yaklaşmaktadır. Değer aralıkları 0 ve 1 olduğunda bu eğri sigmoid olarak, -1 ve +1 olduğunda ise hiperbolik tanjant olarak adlandırılmaktadır. Matematiksel olarak hem fonksiyonların hem de türevlerinin sürekli olması bu eğrilerin özelliğidir. Bu seçenek oldukça düzgün çalışmaktadır ve bu fonksiyon sıklıkla seçilen transfer fonksiyonudur. Diğer transfer fonksiyonları özel ağ mimarilerine ithaf

edilmiştir. Şekil 3.3'te 4 adet transfer fonksiyonu örnek olarak gösterilmiştir [5].

4. **Ölçeklendirme ve sınırlandırma:** İşleyici birimin transfer fonksiyonundan sonra sonuç, ölçeklendirilmek ve sınırlandırılmak üzere ek işlemlerden geçer. Bu ölçeklendirme basit anlamda ölçeklendirme faktörü ile transfer değerini çarpar ve sonra bir ofset ekler. Sınırlandırma ölçeklendirilmiş sonucun üst veya alt sınırı geçmeyeceğini garanti altına alan mekanizmadır. Bu sınırlandırma orijinal transfer fonksiyonunun gerçekleştirilmiş olabileceği katı sınırlandırmaya bir ektir [5].
5. **Çıktı fonksiyonu (rekabet):** Her işleyici birimin yüzlerce nörona çıktı olarak gidebilecek tek bir çıktı sinyali üretmesine izin verilmektedir. Bunlar tıpkı biyolojik nöronlar gibidir. Normalde sonuç doğrudan transfer fonksiyonunun sonucuna eşittir. Bununla birlikte bazı ağ topolojilerinde rekabeti komşu işleyici birimler arasına dâhil etmek için transfer sonucu değiştirilir. Yeterli şiddette olmadıkça işleyici birimlere engel olmak için nöronların birbiriyle rekabet etmesine izin verilir. Rekabet bir seviyede veya seviyelerin her birinde gerçekleşebilir. Birinci olarak rekabet hangi yapay nöronun aktif olacağını belirler veya bir çıktı üretilmesini sağlar. İkinci olarak rekabetçi girişler hangi işleyici birimin öğrenme veya uyum sürecine katılacağını belirlemeye yardımcı olur [5].
6. **Hata fonksiyonu ve geriye yayılan değer:** Birçok öğrenen ağda anlık çıktı ile beklenen çıktı arasındaki fark hesaplanmaktadır. Bu ham hata daha sonra hata fonksiyonu tarafından belirli bir ağ mimarisi ile eşleştirilmek üzere dönüştürülür. Birçok ağ mimarisi bu hatayı doğrudan kullanır fakat bazıları işaretini korurken hatanın karesini alır, bazıları hatanın küpünü alır ve diğer modeller kendi özel amaçlarına uydurabilmek için ham hatayı değiştirir. Yapay nöronun hatası daha sonra bir başka işleyici birimin öğrenme fonksiyonuna doğru yayılır. Bu hata kavramı anlık hata olarak ta adlandırılabilir [5].

   Anlık hata bir önceki katmana da yayılabilir. Geriye yayılan değer anlık hata da olabilir. Anlık hata sıklıkla transfer fonksiyonunun bir türevi tarafından olmak üzere bazı biçimlerde ölçeklendirilebilir. Genellikle bu

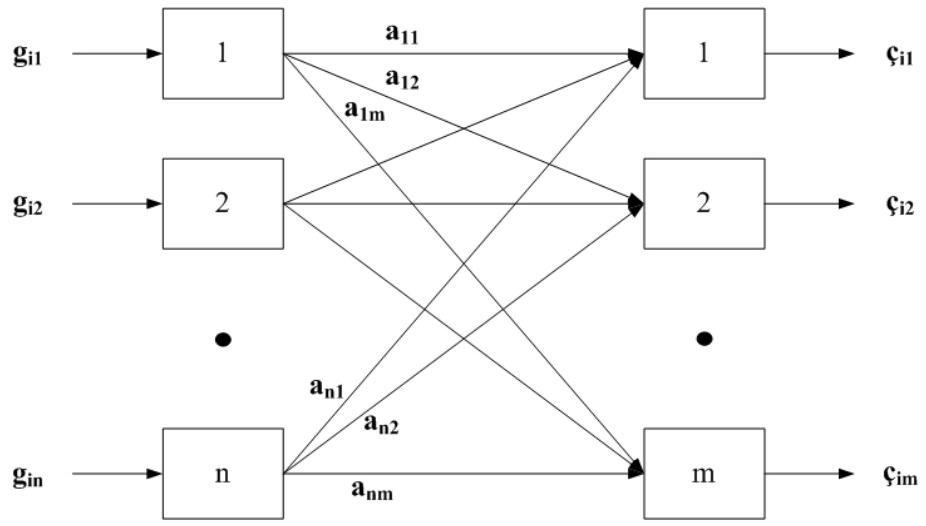
geriye yayılmış değer, öğrenme fonksiyonu tarafından ölçeklendirildikten sonra diğer öğrenme döngüsüne girmeden önce değiştirilmek için her bir gelen bağlantı ağırlığıyla çarpılır [5].

7. **Öğrenme fonksiyonu:** Öğrenme fonksiyonunun amacı bazı yapay sinir ağları kökenli algoritmalara göre her işleyici birimin girişindeki değişken bağlantı ağırlıklarını düzenlemektir. Beklenen sonuç elde etmek için giriş bağlantılarının ağırlıklarını değiştirme işlemi öğrenme şekline ek olarak uyum fonksiyonu olarak ta adlandırılabilir. Öğreticili ve öğreticisiz olmak üzere iki çeşit öğrenme vardır. Öğreticili öğrenme bir öğretmen gerektirir. Öğretmen bir veri topluluğunun eğitim seti veya ağ sonuçlarını değerlendiren bir gözleyici olabilir. Başka bir deyişle bir öğreticiye sahip olmak destek yoluyla öğrenmektir. Harici bir öğretici olmadığında ağ içinde tasarlanmış bazı iç ölçütlere göre sistem kendi kendini düzenlemelidir [5].

### 3.1.6. Yapay sinir ağı yapıları

a) **Tek tabakalı ileri beslemeli YSA:**

Tek tabakalı YSA şekil 3.4'te görüldüğü gibi bir tek giriş ve çıkış tabakasından meydana gelir. Eğriselliği temin edecek orta tabakanın bulunmaması dolayısı ile bu mimari yapı daha çok doğrusal tasvirler için kullanılır [7].

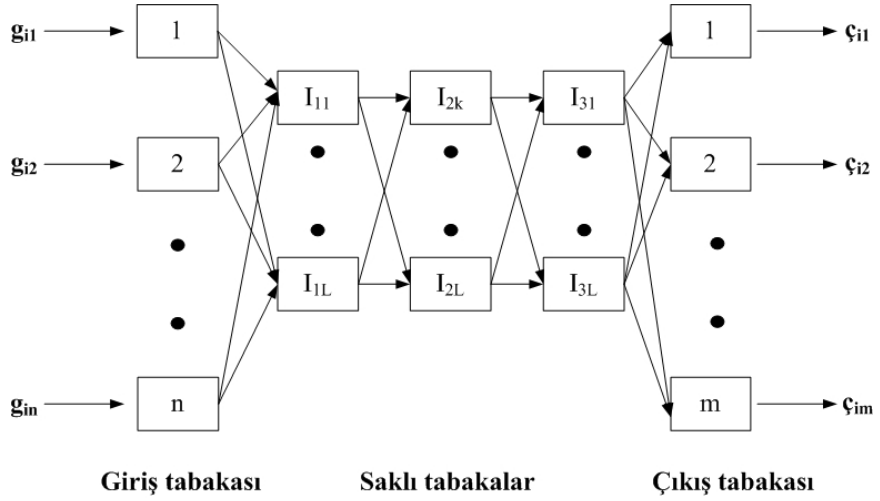



**Şekil 3.4 Tek tabakalı ileri beslemeli yapay sinir ağı**

b) **Çok tabakalı ileri beslemeli YSA:**

Giriş ve çıkış tabakaları arasında en az bir ara tabakaya sahip ileri beslemeli bir ağdır. Ara tabakalar giriş ve çıkış tabakalarındaki sinir hücreleri ile doğrudan bağlantısı olan saklı hücre adı verilen sinir hücrelerinden meydana gelir. Saklı hücrelerin meydana getirdiği bu tabakalara ara veya gizli tabaka adı da verilir. Çok tabakalı YSA'nın yeteneği tabakalı bir yapıya sahip olmasından ve ara tabaka sinir hücre çıkışlarında doğrusal olmayan işlemcilerin kullanılmasından kaynaklanır. Çok tabakalı YSA'ya örnek Şekil 3.5'te gösterilmiştir [7].

Giriş ve çıkış tabakaları arasına gizli tabaka konması veya gizli tabaka sayısının artırılmasının verimi artırıp artırmayacağı konusunda kesin bir cevap vermek yanlış olur. Ancak genel kanaat tek tabakalı YSA doğrusal tasvirlerde ve çok tabakalı YSA'nın doğrusal olmayan tasvirlerde iyi sonuçlar verdiği yönündedir [7].




**Şekil 3.5 Çok tabakalı ileri beslemeli yapay sinir ağı**

c) **Kaskat bağlantılı (yinelemeli) YSA:**

Bu yapıdaki YSA, en az bir tane geri besleme döngüsü içermesi ile ileri beslemeli YSA'dan ayrılır. Kaskat bağlantılı YSA'lar geri beslemeli ağlar olarak ta isimlendirilir [8].

### 3.1.7. YSA öğrenme stratejileri

Yapay sinir ağları gibi örneklerden öğrenen sistemlerde değişik öğrenme stratejileri kullanılmaktadır. Öğrenmeyi gerçekleştirecek olan sistem ve kullanılan öğrenme algoritması bu stratejilere bağlı olarak değişmektedir. Genel olarak 3 öğrenme stratejisinin ve 1 tane de bunların birleşimi sonucunda oluşan öğrenme stratejisinin uygulandığı görülmektedir. Bunlar aşağıdaki gibidir [6]:

a) **Öğretmenli (Supervised) Öğrenme:** Bu tür stratejide öğrenen sistemin olayı öğrenebilmesi için bir öğretmen yardımcı olmaktadır. Öğretmen sisteme öğrenilmesi istenen olay ile ilgili örnekleri girdi/çıktı seti olarak verir. Yani, her örnek için hem girdiler hem de o girdiler karşılığında oluşturulması gereken çıktılar sisteme gösterilirler. Sistemin görevi girdileri öğretmenin belirlediği çıktılara haritalamaktır. Bu sayede olayın girdileri ile çıktıları arasındaki ilişkiler öğrenilmektedir.

b) **Destekleyici (Reinforcement) Öğrenme:** Bu tür stratejide de öğrenen sisteme bir öğretmen yardımcı olur. Fakat öğretmen her girdi seti için olması gereken (üretilmesi gereken) çıktı setini sisteme göstermek yerine sistemin kendisine gösterilen girdilere karşılık çıktısını üretmesini bekler ve üretilen çıktının doğru veya yanlış olduğunu gösteren bir sinyal üretir. Sistem, öğretmenden gelen bu sinyali dikkate alarak öğrenme sürecini devam ettirir. LVQ ağı buna örnek olarak verilebilir.

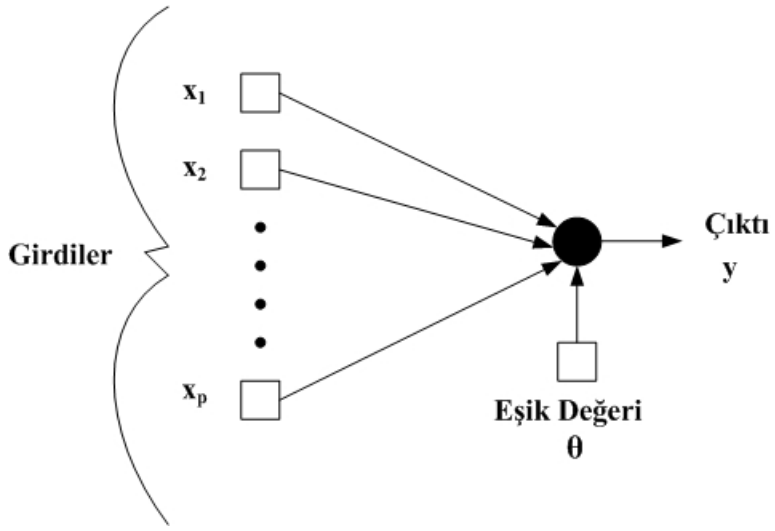
c) **Öğretmensiz (Unsupervised) Öğrenme:** Bu tür stratejide sistemin öğrenmesine yardımcı olan herhangi bir öğretmen yoktur. Sisteme sadece girdi değerleri gösterilir. Örneklerdeki parametreler arasındaki ilişkileri sistemin kendi kendisine öğrenmesi beklenir. Bu, daha çok sınıflandırma problemleri için kullanılan bir stratejidir. Yalnız sistemin öğrenmesi bittikten sonra çıktıların ne anlama geldiğini gösteren etiketlendirmenin kullanıcı tarafından yapılması gerekmektedir.

d) **Karma Stratejiler:** Yukarıdaki 3 stratejiden birkaçını birlikte kullanarak öğrenme gerçekleştirilen ağlar da vardır. Burada kısmen öğretmenli, kısmen ise öğretmensiz olarak öğrenme yapan ağlar kastedilmektedir. Radyal tabanlı yapay sinir ağları (RBNN) ve olasılık tabanlı ağlar (PBNN) bunlara örnek olarak verilebilir.

### 3.1.8. Kullanılan Yapay Sinir Ağı Yöntemleri

Bu çalışmada 3 adet öğreticili, 1 adet de öğreticisiz öğrenme yöntemi uygulanmıştır. Bu 4 yöntem yapay sinir ağlarının sınıflandırma işlemlerinde sıklıkla kullanılan elemanlarıdır. Perseptron, geri yayılım ağı, destekleyici öğrenmeli vektör nicemleme ağları ve kendini düzenleyen ağlar üzerinde sınıflandırma çalışması yapılmıştır.

## 3.2. Perseptron



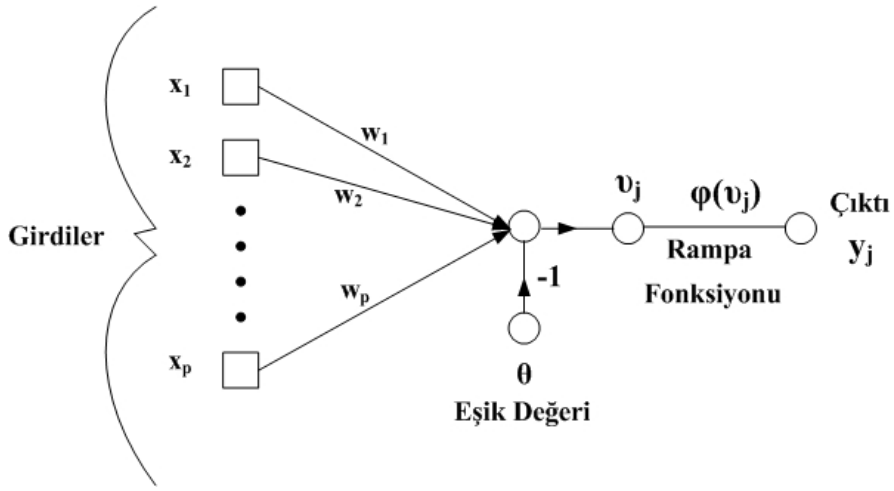

**Şekil 3.6 Tek katmanlı perseptron**

Perseptron, doğrusal ayrılabilen örüntülerin sınıflandırılması için kullanılan basit bir yapay sinir ağı yapısıdır. Basit anlamda şekil 3.6'da görüldüğü gibi ayarlanabilen sinaptik ağırlıklar ve eşik değerlerine sahip tek bir nöron içermektedir. Bu yapay sinir ağının serbest parametrelerini ayarlamak için kullanılan algoritma ilk olarak Rosenblatt (1958,1962) tarafından geliştirilmiş bir öğrenme süreci içinde görülmüştür. Rosenblatt, eğer perseptronu eğitmek için kullanılan örüntüler (vektörler) doğrusal ayrılabilen iki sınıftan çizilirse perseptron algoritmasının yakınsayacağını ve karar yüzeyini iki sınıf arasında bir hiperdüzlem şeklinde konumlandıracağını ispatlamıştır [8].

Şekil 3.6'daki tek katmanlı perseptron tek bir nörona sahiptir. Bu şekildeki perseptron sadece 2 sınıfa ait örüntüleri sınıflandırabilmektedir. Birden fazla nörondan oluşacak şekilde perseptronun çıkış katmanını genişletilirse ikiden fazla

sayıda sınıfı ayırabilmektedir. Bununla birlikte sınıflar perseptronun düzgün çalışabilmesi için doğrusal ayrılabilir olmalıdır [8].

### 3.2.1. Perseptronun temel işleyiş yapısı




**Şekil 3.7 Perseptronun sinyal akış grafiği**

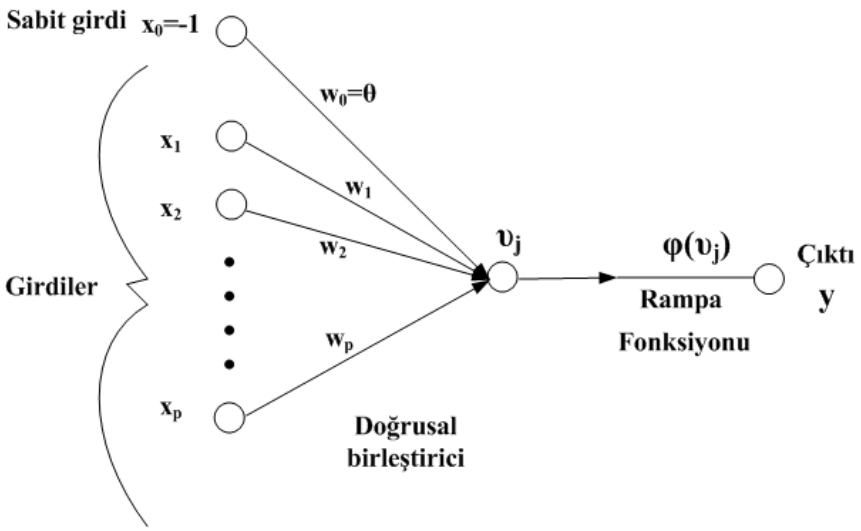
Nöron modelinin toplayıcı birimi, sinapslarına uygulanan girdilerin doğrusal bir kombinasyonunu hesaplar. Sonuç toplam bir basamak fonksiyonundan geçer. Buna göre nöron basamak fonksiyonunun girdisi pozitif ise 1, değilse -1 değerini üretir.

Şekil 3.7'deki sinyal akış grafiğinde tek katmanlı perseptronun sinaptik ağırlıkları $w_1,w_2,\ldots,w_p$ şeklinde ifade edilmiştir. Buna bağlı olarak perseptrona uygulanan girdiler $x_1,x_2,\ldots,x_p$ ile gösterilmiştir. θ, uygulanan harici eşik değerini ifade etmektedir. Şekil 3.7'deki modelden doğrusal birleştirici çıktısı aşağıdaki gibi bulunmaktadır [8]:

$$\nu = \sum_{i=1}^{p} w_i x_i - \theta \tag{3.1}$$

Perseptronun amacı harici uygulanmış $x_1,x_2,\ldots,x_p$ kümesini $c_1$ ve $c_2$ sınıflarından birisine atamaktır. Sınıflandırmanın karar kuralı, $x_1,x_2,\ldots,x_p$ girdileriyle temsil edilen noktaları eğer perseptronun çıktısı +1 ise $c_1$ sınıfına, -1 ise $c_2$ sınıfına atayacaktır [8].

### 3.2.2. Perseptron yakınsama teoremi




**Şekil 3.8 Perseptronun eşit sinyal akış grafiği**

Tek katmanlı perseptronun hata düzeltmeli öğrenme algoritmasının geliştirilmesi için değiştirilmiş şekil 3.8'deki sinyal akış grafik modeliyle çalışmak çok daha uygun görülmektedir. Şekil 3.7'dekiyle aynı olan bu ikinci modelde eşik değeri θ(n), değeri -1 olan sabit bir girdiye bağlı sinaptik ağırlık gibi davranmaktadır. (p+1)*1'lik girdi vektörü şu şekilde tanımlanabilir [8]:

$$x(n) = [-1, x_1(n), x_2(n), \ldots, x_p(n)]^T \tag{3.2}$$

Buna bağlı olarak (p+1)*1'lik ağırlık vektörü de şu şekilde tanımlanmaktadır:

$$w(n) = [\theta(n), w_1(n), w_2(n), \ldots, w_p(n)]^T \tag{3.3}$$

Buna göre doğrusal birleştirici çıktısı şu şekilde oluşmaktadır:

$$\nu(n) = w^T(n)x(n) \tag{3.4}$$

Sabit bir n değeri için $w^Tx=0$ denklemi, $x_1,x_2,\ldots,x_p$ koordinatlı p boyutlu uzayda çizildiğinde iki sınıfın girdileri arasındaki karar yüzeyi olarak bir hiperdüzlemi tanımlamaktadır.

Tek katmanlı perseptronun girdi değişkenlerinin hiperdüzlemin zıt taraflarına düşen doğrusal ayrılabilen iki sınıftan oluştuğu varsayılsın. $X_1$, $c_1$ sınıfına ait olan $x_1(1)$, $x_1(2)$, …,$x_1(n)$ eğitim vektörlerinin alt kümesi, $X_2$ de $c_2$ sınıfına ait olan $x_2(1),x_2(2),...,x_2(n)$ eğitim vektörlerinin alt kümesi olsun.

Sınıflandırıcıyı eğitmek için verilmiş olan $X_1$ ve $X_2$ vektör kümeleriyle, eğitim süreci $c_1$ ve $c_2$ sınıfının doğrusal ayrılabilir olduğu durumda ağırlık vektörü w'nın ayarlanmasına katkıda bulunmaktadır. Bunun tersine $c_1$ ve $c_2$ sınıfı doğrusal ayrılabilir olarak biliniyorsa, bunun gibi bir w ağırlık vektörü bulunmaktadır [8]:

$$w^T x \geq 0 \qquad c_1 \text{ sınıfına ait olan her x girdi vektörü için}$$

ve (3.5)

$$w^T x < 0 \qquad c_2 \text{ sınıfına ait olan her x girdi vektörü için}$$

Eğitim vektörlerinin alt kümeleri olan $X_1$ ve $X_2$ ile basit bir perseptron için eğitim problemi, yukarıdaki eşitsizliklerin sağlandığı bir w ağırlık vektörünün bulunmasıdır.

Basit bir perseptronun ağırlık vektörünü uyarlamak için aşağıdaki formüller kullanılmaktadır:

1. Eğer eğitim vektörünün n.ci üyesi x(n), n.ci iterasyonda hesaplanan ağırlık vektörü w(n) tarafından doğru sınıflandırılırsa perseptronun ağırlık vektörüne hiçbir düzeltme yapılmaz.

$$w(n+1) = w(n) \quad \text{eğer } w^T(n)x(n) \geq 0 \text{ ve } x(n) \; c_1 \text{ sınıfına ait ise}$$

ve (3.6)

$$w(n+1) = w(n) \quad \text{eğer } w^T(n)x(n) < 0 \text{ ve } x(n) \; c_2 \text{ sınıfına ait ise}$$

2. Aksi takdirde perseptronun ağırlık vektörü aşağıdaki kurala bağlı olarak güncelleştirilir. Öğrenme katsayısı η(n), n.ci iterasyonda ağırlık vektörüne uygulanan düzenlemeyi kontrol etmektedir.

$$w(n+1) = w(n) - \eta(n)x(n) \qquad \text{eğer } w^t(n)x(n) \geq 0 \text{ ve } x(n) \; c_2 \text{ sınıfına ait ise}$$

ve (3.7)

$$w(n+1) = w(n) + \eta(n)x(n) \qquad \text{eğer } w^t(n)x(n) < 0 \text{ ve } x(n) \; c_1 \text{ sınıfına ait ise}$$

Eğer $\eta(n) = \eta > 0$ durumu söz konusu ise, η'nün iterasyon sayısı n'den bağımsız bir sabit olduğu durumda perseptron sabit bir artış uyum kuralına sahiptir.

Neticede ilk olarak η=1 olduğu durumda sabit artışlı uyum kuralının yakınsaması ispatlanmış oldu. Pozitif olduğu sürece η'nün değeri önemsizdir. η≠1 değeri, örüntü vektörlerinin ayrılabilirliklerini etkilemeden yalnızca ölçeklendirir.

İspat ilk olarak w(0)=0 olduğu durum için gösterilmiştir. n=1,2,… şeklindeyken $w^t(n)x(n) < 0$ olduğunu ve girdi vektörü x(n)'in $X_1$ alt kümesine ait

olduğu varsayılsın. Perseptron, denklem (3.5)'in ikinci koşulu sağlanmadığı takdirde x(1),x(2),… vektörlerini yanlış şekilde sınıflandırır. η(n)=1 sabitiyle, denklem (3.7)'nin ikinci kısmı kullanılırsa:

$$w(n+1) = w(n) + x(n) \quad x(n)\text{'in } c_1 \text{ sınıfına ait olduğu durumda} \quad (3.8)$$

w(0)=0 ilk koşulunu verilirse, w(n+1) için denklemi adım adım çözülebilir ve aşağıdaki sonucu elde edilir.

$$w(n+1) = x(1) + x(2) + \ldots + x(n) \quad (3.9)$$

$c_1$ ve $c_2$ doğrusal ayrılabilir olarak düşünülürse, $X_1$'in alt kümesi x(1),…,x(n) vektörleri için $w^T x(n)>0$ olduğu durumda $w_0$ çözümü vardır. $w_0$ sabit çözümü için aşağıdaki bağıntıyla bir α pozitif rakamı tanımlanacaktır:

$$\alpha = \min w_0^T x(n) \qquad x(n) \in X_1 \quad (3.10)$$

Denklem (3.9)'un iki tarafı da $w_0^T$ ile çarpıldığında denklem şu şekle gelmektedir:

$$w_0^T w(n+1) = w_0^T x(1) + w_0^T x(2) + \ldots + w_0^T x(n) \quad (3.11)$$

Buna uygun olarak (3.10) denkleminden de yararlanılarak (3.12) elde edilmektedir:

$$w_0^T w(n+1) \geq n\alpha \quad (3.12)$$

Daha sonra Cauchy-Schwarz eşitsizliğinden faydalanılacaktır. Cauchy-Schwarz eşitsizliği $w_0$ ve w(n+1) için aşağıdaki (3.13) denklemini ortaya koymaktadır:

$$\|w_0\|^2 \|w(n+1)\|^2 \geq [w_0^T w(n+1)]^2 \quad (3.13)$$

$\|.\|$, eklenmiş argüman vektörünün öklit normunu göstermektedir ve $w_0^T w(n+1)$ iç çarpımı ölçeklenebilir miktardır. Denklem (3.2)'den $[w_0^T w(n+1)]^2$'nin $n^2\alpha^2$'ye eşit veya büyük olduğu görülebilmektedir. Ayrıca, denklem (3.13)'ten $\|w_0\|^2 \|w(n+1)\|^2$'in $[w_0^T w(n+1)]^2$'ye eşit veya daha büyük olduğu görülebilmektedir. Sonuçta aşağıdaki gibi devam edilir:

$$\|w_0\|^2 \|w(n+1)\|^2 \geq n^2\alpha^2$$

veya eşit şekilde, (3.14)

$$\|w(n+1)\|^2 \geq \frac{n^2\alpha^2}{\|w_0\|^2}$$

Daha sonra başka bir geliştirme yolu takip edilecektir. Denklem (3.8)'i aşağıdaki formda yazarız:

$w(k+1) = w(k) + x(k)$ her k=1,…,n ve x(k)$\in X_1$ için (3.15)

Bu yüzden denklem (3.15)'in iki tarafın da karesi alınmış öklit normu;

$$\|w(k+1)\|^2 = \|w(k)\|^2 + \|x(k)\|^2 + 2w^T(k)x(k) \qquad (3.16)$$

Fakat perseptronun x(k) girdi vektörünü yanlış sınıflandırdığını varsayımı altında, $w^T(k)x(k)<0$ eşitsizliği oluşmaktadır. Bu yüzden denklem (3.16)'dan aşağıdaki denklem çıkarılır:

$$\|w(k+1)\|^2 = \|w(k)\|^2 + \|x(k)\|^2$$

veya denk şekilde, (3.17)

$$\|w(k+1)\|^2 - \|w(k)\|^2 \leq \|x(k)\|^2, \quad k=1,\ldots,n$$

Bu eşitsizlikler k=1,…,n için eklenirse ve w(0)=0 başlangıç koşulu olduğu varsayılırsa, aşağıdaki durum elde edilir:

$$\|w(n+1)\|^2 \leq \sum_{k=1}^{n} \|x(k)\|^2$$

$$\leq n\beta \qquad (3.18)$$

Eğer β aşağıdaki şekilde tanımlanmış pozitif bir sayı ise

$$\beta = \max \|x(k)\|^2 \qquad x(k)\in X_1 \text{ için} \qquad (3.19)$$

Denklem (3.18)'de, ağırlık vektörü w(n+1)'in öklit normunun karesinin iterasyon sayısı olan n ile birlikte doğrusal olarak arttığını gösterilmektedir.

Açıkçası, denklem (3.18)'in ikinci sonucu büyük n değerleri için denklem (3.14)'ün önceki sonucuyla çelişmektedir. Gerçekten, n'in denklem (3.14) ve denklem (3.18)'in eşitlik işaretini sağladığı $n_{max}$ adı verilen bir değerinden büyük olamayacağı görülmektedir. $n_{max}$, denklemin çözümüdür.

$$\frac{n_{max}^2\alpha^2}{\|w_0\|^2} = n_{max}\beta \qquad (3.20)$$

$n_{max}$ için bir $w_0$ vektörü çözülürse, denklem (3.21) bulunur:

$$n_{\max} = \frac{\beta \left\| w_0 \right\|^2}{\alpha^2} \tag{3.21}$$

Bütün n'ler için η(n)=1 olduğu durumda w(0)=0 olduğu ve bir çözüm vektörü $w_0$ olduğu ispatlanmıştır. Birleştirici birimleri perseptronun tepki birimine bağlayan sinaptik ağırlıkların ayarlanması kuralı $n_{max}$ iterasyondan sonra sona ermelidir.

### 3.3. Geri Yayılım Algoritması (Back Propagation)

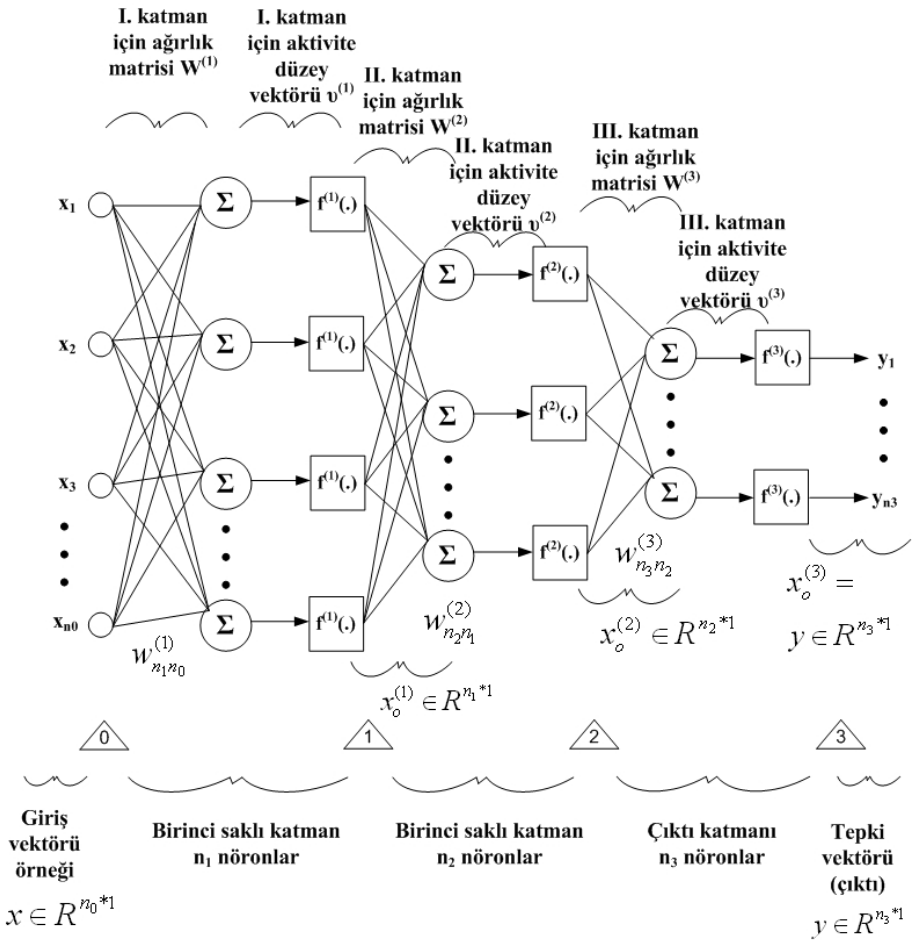
Geri yayılım ağı, çok katmanlı ileri beslemeli perseptronların eğitilmesi için kullanılır. Bu yönteme aynı zamanda genelleştirilmiş delta kuralı da denilmektedir. Werbos tarafından 1974'te geliştirilmiş olup günümüzde sıkça kullanılan bir yöntemdir. Girdi ve çıktı veri setleri ile geri yayılım ağı, doğrusal olmayan bir haritalama yapmak için sinaptik ağırlıkları düzenleyebilmektedir. Eğitim sürecinden sonra sabit ağırlıkları olan çok tabakalı perseptron sınıflandırma, örüntü tanıma, teşhis vb. problemler için çözüm yolu olabilmektedir [9].

#### 3.3.1. Geri yayılım algoritmasının temel işleyişi

Çok katmanlı ileri beslemeli perseptronları eğitmek için kullanılan dereceli azalma eğimi (steepest descent gradient) yaklaşımına dayanan standart geri yayılım algoritması, anlık hatayı temsil eden enerji fonksiyonunun minimize edilmesi için uygulanmaktadır.

Başka bir deyişle, (3.22) bağıntısında gösterilen fonksiyon minimize edilmek istenmektedir:

$$E_q = \frac{1}{2}(d_q - x_o^{(3)})^T (d_q - x_o^{(3)}) = \frac{1}{2}\sum_{h=1}^{n} (d_{qh} - x_{o.h}^{(3)})^2 \tag{3.22}$$




**Şekil 3.9 Üç katmanlı ileri beslemeli yapay sinir ağı yapısı**

$d_q$, q. girdi için beklenen ağ çıktısını temsil etmektedir. $x_o^{(3)} = y_q$ ise şekil 3.9'da gösterilen çok katmanlı perseptronun gerçek çıktısını temsil etmektedir. Denklem (3.22)'yi minimize ederek ağırlık güncelleştirmelerinin yapılması çevrimiçi (online) metot olarak isimlendirilir [9].

Dereceli azalma eğimi (steepest descent gradient) yaklaşımı kullanılarak herhangi bir katmandaki ağ ağırlıkları için öğrenme kuralı denklem (3.23)'deki gibidir:

$$\Delta w_{ji}^{(s)} = -\mu^{(s)} \frac{\partial E_q}{\partial w_{ji}^{(s)}} \tag{3.23}$$

s=1,2,3 uygun ağ yapısını tasarlar ve $\mu^{(s)} > 0$ da ilgili öğrenme katsayısı parametresidir. Çok katmanlı ileri beslemeli perseptron ağının saklı katmanları ve çıktı katmanı için farklı öğrenme algoritmaları türetilir. Çıkış katmanındaki ağırlıklar denklem (3.24)'e göre düzenlenir.

$$\Delta w_{ji}^{(3)} = -\mu^{(3)} \frac{\partial E_q}{\partial w_{ji}^{(3)}} \tag{3.24}$$

Kısmi türevler için zincir kuralı yöntemi kullanılarak denklem aşağıdaki gibi yeniden yazılır:

$$\Delta w_{ji}^{(3)} = -\mu^{(3)} \frac{\partial E_q}{\partial v_j^{(3)}} \frac{\partial v_j^{(3)}}{\partial w_{ji}^{(3)}} \tag{3.25}$$

(3.25)'teki ayrılmış terimler aşağıdaki gibi değerlendirilebilir:

$$\frac{\partial v_j^{(3)}}{\partial w_{ji}^{(3)}} = \frac{\partial}{\partial w_{ji}^{(3)}} \left( \sum_{h=1}^{n_2} w_{jh}^{(3)} x_{o.h}^{(2)} \right) = x_{o.i}^{(2)} \tag{3.26}$$

ve

$$\frac{\partial E_q}{\partial v_j^{(3)}} = \frac{\partial}{\partial v_j^{(3)}} \{ \frac{1}{2} \sum_{h=1}^{n_2} \left[ d_{qh} - f(v_h^{(3)}) \right]^2 \} = -\left[ d_{qi} - f(v_j^{(3)}) \right] g(v_j^{(3)}) \tag{3.27}$$

veya

$$\frac{\partial E_q}{\partial v_j^{(3)}} = -(d_{qj} - x_{o.j}^{(3)}) g(v_j^{(3)}) = -\delta_j^{(3)} \tag{3.28}$$

g(.), doğrusal olmayan aktivasyon fonksiyonu f(.)'in birinci türevini temsil etmektedir. (3.28)'de tanımlanan terim, yerel hata veya delta olarak isimlendirilmektedir.

(3.25), (3.26) ve (3.28)'i birleştirerek çıkış katmanındaki ağırlıklar için öğrenme kuralı denklemi aşağıdaki gibi yazılır:

$$\Delta w_{ji}^{(3)} = \mu^{(3)} \delta_j^{(3)} x_{o.i}^{(2)} \tag{3.29}$$

veya

$$w_{ji}^{(3)}(k+1) = w_{ji}^{(3)}(k) + \mu^{(3)} \delta_j^{(3)} x_{o.i}^{(2)} \tag{3.30}$$

Saklı katmanlardaki ağırlıklar için güncelleştirme denklemleri aynı yolla türetilebilir. Dereceli azalma eğimi (steepest descent gradient) yaklaşımını kullanarak aşağıdaki denkleme ulaşılır:

$$\Delta w_{ji}^{(2)} = -\mu^{(2)} \frac{\partial E_q}{\partial w_{ji}^{(2)}} = -\mu^{(2)} \frac{\partial E_q}{\partial v_j^{(2)}} \frac{\partial v_j^{(2)}}{\partial w_{ji}^{(2)}} \tag{3.31}$$

Denklem (3.31)'in sağ tarafında bulunan ikinci kısmi türev aşağıdaki gibi değerlendirilebilir:

$$\frac{\partial \nu_j^{(2)}}{\partial w_{ji}^{(2)}} = \frac{\partial}{\partial w_{ji}^{(2)}}\left(\sum_{h=1}^{n_1} w_{jh}^{(2)} x_{o.h}^{(1)}\right) = x_{o.i}^{(1)} \tag{3.32}$$

Denklem (3.31)'deki birinci kısmi türevin değerlendirilmesi, $\nu_j^{(2)}$'deki değişimin ağın çıkış katmanına doğru ilerlemesinden ve bütün ağ çıktılarını etkilemesinden itibaren çok daha karmaşıktır. Bu çıkarım, niceliklerin bilinen nicelikler fonksiyonu olarak açıklanabilmesi durumunda izlenebilir. İleri gitmek için aşağıdaki gibi yazılır [9]:

$$\frac{\partial E_q}{\partial \nu_j^{(2)}} = \frac{\partial}{\partial x_{o.h}^{(2)}}\left\{\frac{1}{2}\sum_{h=1}^{n_3}\left[d_{qh} - f\left(\sum_{p=1}^{n_2}\ \right)\right]^2\right\}\frac{\partial x_{o.j}^{(2)}}{\partial \nu_j^{(2)}} \tag{3.33}$$

veya

$$\begin{aligned}\frac{\partial E_q}{\partial \nu_j^{(2)}} &= -\left[\sum_{h=1}^{n_3}\left(d_{qh} - x_{o.h}^{(3)}\right)g\left(\nu_h^{(3)}\right)w_{hj}^{(3)}\right]g(\nu_j^{(2)}) \\ &= -\left(\sum_{h=1}^{n_3}\delta_h^{(3)} w_{hj}^{(3)}\right)g(\nu_j^{(2)}) = -\delta_j^{(2)}\end{aligned} \tag{3.34}$$

Denklem (3.31), (3.32) ve (3.34) birleştirilirse:

$$\Delta w_{ji}^{(2)} = \mu^{(2)}\delta_j^{(2)} x_{o.i}^{(2)} \tag{3.35}$$

veya

$$w_{ji}^{(2)}(k+1) = w_{ji}^{(2)}(k) + \mu^{(2)}\delta_j^{(2)} x_{o.i}^{(2)} \tag{3.36}$$

(3.30) ve (3.36) karşılaştırılırsa, çıktı katmanı ve saklı katmanlardaki ağırlıkların güncelleştirilmesi kuralının aynı yapıya sahip olduğu görülmektedir. Aradaki tek fark yerel hatanın hesaplanması şeklidir. Çıktı katmanı için yerel hata, beklenen çıktı ile gerçek ağ çıktısı arasındaki fark ile orantılıdır. Aynı konsepti saklı katmanların çıkışlarına uygularsak saklı katmandaki bir nöron için yerel hatanın beklenen çıktı ile her bir nöronun gerçek çıktısı arasındaki fark ile orantılı olduğu görülmektedir. Tabii ki eğitim süreci boyunca saklı katmandaki nöronların beklenen çıktıları bilinmemektedir. Bu yüzden yerel hatalar bağlı olan bütün

nöronların sinyalleri süresince tekrarlamalı olarak hesaplanmalıdır. Denklem (3.36), keyfi sayıda saklı katman içeren çok katmanlı ileri beslemeli ağlar için genelleştirilebilir. Bunun için aşağıdaki gibi yazılır [9]:

$$w_{ji}^{(s)}(k+1) = w_{ji}^{(s)}(k) + \mu^{(s)}\delta_j^{(s)}x_{o.i}^{(s)} \tag{3.37}$$

Aynı zamanda çıktı katmanı için aşağıdaki durum sağlanmaktadır:

$$\delta_j^{(s)} = \left(d_{qh} - x_{o.i}^{(s)}\right)g\left(v_j^{(s)}\right) \tag{3.38}$$

Saklı katmanlar için de aşağıdaki durum sağlanmaktadır:

$$\delta_j^{(s)} = \left(\sum_{h=1}^{n_{s-1}} \delta_h^{(s+1)} w_{hj}^{(s+1)}\right) g(v_j^{(s)}) \tag{3.39}$$

### 3.4. Destekleyici Öğrenmeli Vektör Nicemleme Ağları (LVQ)

LVQ ağı Kohonen tarafından 1984 yılında geliştirilmiştir. Temel felsefesi n boyutlu bir vektörü bir vektörler setine haritalamaktır. Aslında bir vektörün belirli sayıda vektörler ile gösterimi amaçlanmaktadır. Öğrenme ile de girdi vektörünün hangi vektör seti tarafından temsil edilmesi gerektiğinin bulunması kastedilmektedir. Bu vektör setine referans vektörleri denirse LVQ ağının görevi öğrenme yolu ile bu referans vektörleri belirlemektir. Yani, girdi vektörlerinin üyesi olabilecekleri vektör sınıfını belirlemektir [6].

LVQ ağları genel olarak sınıflandırma problemlerinin çözümünde kullanılmaktadır. Çıktılardan sadece birisi 1 diğerleri 0 değerlerini almaktadır. Çıktı değerinin 1 olması girdinin ilgili çıktının temsil ettiği sınıfa ait olduğunu göstermektedir.

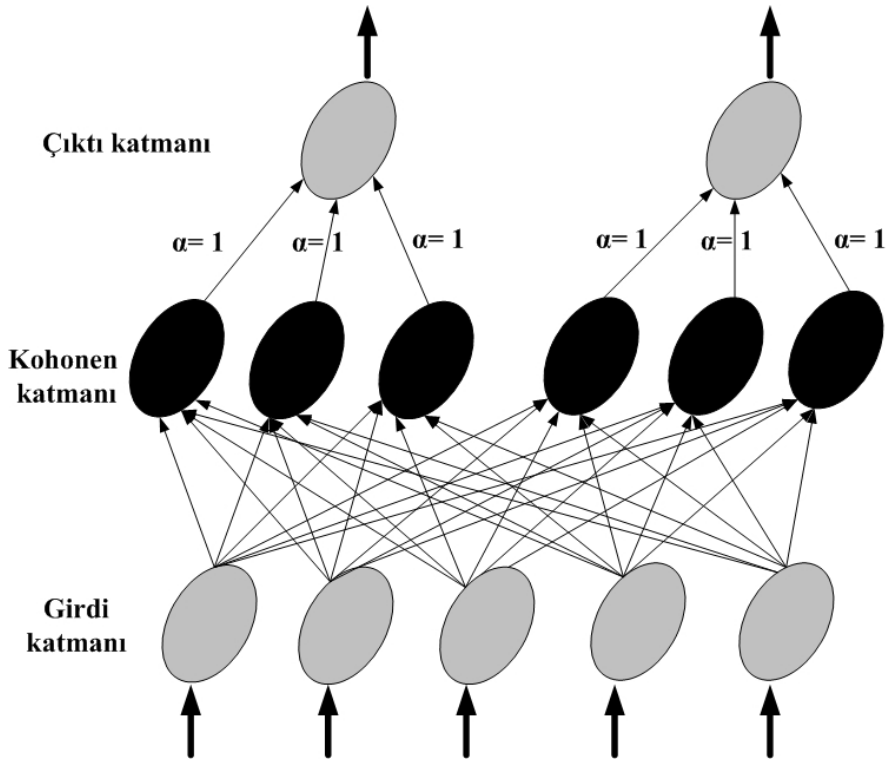
Eğitim sırasında girdilerin sınıflara ayrılması en yakın komşu (nearest neighbour) kuralına göre gerçekleştirilmektedir. Girdi vektörü ile referans vektörleri arasında en kısa mesafe aranmakta ve girdi vektörünün en kısa mesafede bulunan vektör grubuna ait olduğu varsayılmaktadır. Ağın ağırlıklarını değiştirmek yolu ile girdileri doğru sınıflara ayıracak referans vektörleri belirlenmektedir. Kullanılan öğrenme stratejisi destekleyici (reinforcement learning) öğrenmedir. Çıktı değerlerinin belirlenmesinde ise “kazanan her şeyi alır” (winner takes all) stratejisi uygulanmaktadır. Ağ eğitilirken her iterasyonda ağın ürettiği çıktının değeri yerine sadece doğru olup olmadığı söylenir. Sadece

girdi vektörüne en yakın olan vektörün (kazanan vektör) değerleri (ağın bu vektöre ait ağırlıkları) değiştirilir [6].

Diğer ağlarda olduğu gibi LVQ ağında da ağın ağırlıkları öğrenme katsayısına göre değiştirilmektedir.

Kullanılan öğrenme katsayısının zaman içerisinde sıfır olacak şekilde monoton olarak azaltılması istenmektedir.

LVQ ağının 3 katmandan oluşan yapısı şekil 3.10'da görülmektedir.




**Şekil 3.10 Lvq ağının yapısı**

Lvq ağları girdi katmanı ile Kohonen katmanı arasında tam bağlantılı, Kohonen katmanı ile çıktı katmanı arasında ise kısmi bağlantılıdır. Yani girdi katmanındaki her proses elemanı Kohonen katmanındaki her proses elemanına bağlıdır. Kohonen katmanındaki proses elemanları ise sadece çıktı katmanındaki bir tek proses elemanına bağlıdır. Kohonen katmanı ile çıktı katmanı arasındaki ağırlıklar (α) sabit olup 1'e eşittir. Sadece girdi katmanı ile Kohonen katmanı arasındaki ağırlıklar değiştirilirler. Öğrenme bu ağırlıklar üzerinden gerçekleştirilir. Kohonen katmanında kaç tane proses elemanı var ise o kadar referans vektörü oluşacaktır. **Referans vektörü,** girdi değerlerini Kohonen

katmanındaki proses elemanlarına bağlayan bağlantıların ağırlık değerlerinden oluşur. Dolayısı ile referans vektörünün eleman sayısı girdi katmanındaki eleman sayısı kadardır [6].

Kohonen ve çıktı katmanlarındaki proses elemanlarının çıktıları ikili (binary) değerler olup sadece bir proses elemanının çıktısı 1 diğerlerininki ise 0'dır. Kohonen katmanında proses elemanları birbirleri ile yarışırlar. Yarışı kazanan proses elemanının çıktısı 1 değerini alırken diğerinin çıktısı ise 0 olur. Hangi proses elemanının yarışı kazandığına öğrenme kuralına göre karar verilir. Kohonen katmanında hangi proses elemanının çıktısı 1 olursa onun bağlı olduğu çıktı katmanındaki proses elemanının çıktısı 1 değerini alır, diğerlerinin çıktısı da 0 olur. Böylece ağa sunulan bir girdi için çıktı katmanında sadece bir proses elemanının çıktısı 1 olmakta ve girdi vektörü o çıktının gösterdiği sınıfın üyesi kabul edilmektedir. Öğrenme ile girdi için doğru sınıfın belirlenmesi sağlanmaktadır [6].

### 3.4.1. LVQ ağının temel işleyişi

LVQ ağlarını kullanmak için şu prosedürü izlemek gerekmektedir:

1. Örneklerin belirlenmesi
2. Ağın topolojisinin belirlenmesi (girdi ve çıktı sayısının belirlenmesi, referans vektör sayısının belirlenmesi)
3. Ağın öğrenme parametrelerinin belirlenmesi (öğrenme katsayısı ve istenen sabit değerlerin belirlenmesi)
4. Ağırlıkların başlangıç değerlerinin atanması
5. Öğrenme setinden bir örneğin ağa gösterilmesi
6. Kazanan proses elemanının bulunması
7. Ağırlıkların değiştirilmesi
8. Bütün örnekler doğru sınıflandırılıncaya kadar yukarıdaki adımları (5-7) tekrar etmek

Bir LVQ ağının performansı doğru sayıda referans vektörünün belirlenmesi, ağırlıkların başlangıç değerleri ve öğrenme katsayısının belirlenmesi ile yakından ilgilidir.

### 3.4.2. LVQ ağının öğrenme kuralı

LVQ ağının öğrenme kuralına Kohonen öğrenme kuralı da denmektedir. Öğrenme kuralı, Kohonen tabakasındaki proses elemanlarının birbirleri ile yarışmaları ilkesine dayanır. Yarışma girdi vektörü ile ağırlık vektörleri (referans vektörleri) arasındaki öklid (euclid) mesafesinin hesaplanmasına dayanmaktadır. Hangi proses elemanının referans vektörü girdi vektörüne en yakın ise o yarışmayı kazanmaktadır. Girdi vektörü X ile referans vektörü A arasındaki mesafe d ile gösterilirse; i. proses elemanının mesafesi şu şekilde hesaplanmaktadır [6].

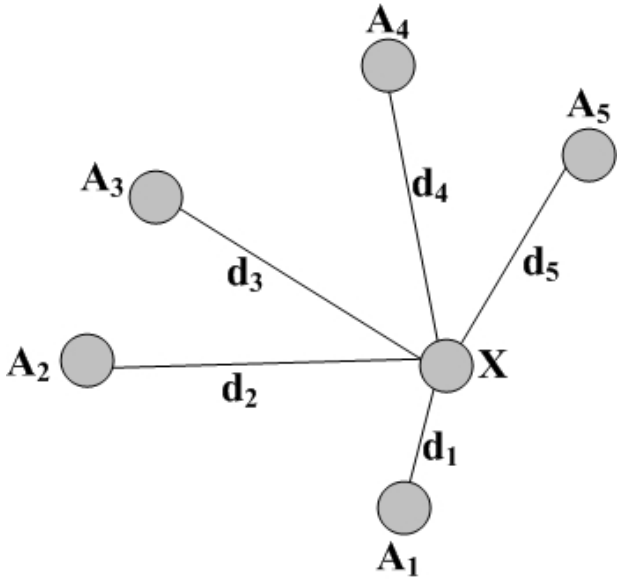
$$d_i = \|A_i - X\| = \sqrt{\sum_j (A_{ij} - x_j)^2} \tag{3.40}$$

Burada $A_{ij}$ ve $x_j$ ağırlık vektörü ve girdi vektörünün j. değerlerini göstermektedir. Girdi vektörü ile referans vektörlerinin hepsinin arasındaki mesafesi tek tek hesap edildikten sonra hangi proses elemanın referans vektörü girdi vektörüne en yakın ise o yarışmayı kazanmaktadır. Öğrenme sırasında, sadece girdi katmanını bu proses elemanına bağlayan ağırlık değerleri değiştirilir. Diğer ağırlıklar değiştirilmezler. Kazanan proses elemanı için iki durum söz konusudur [6]:

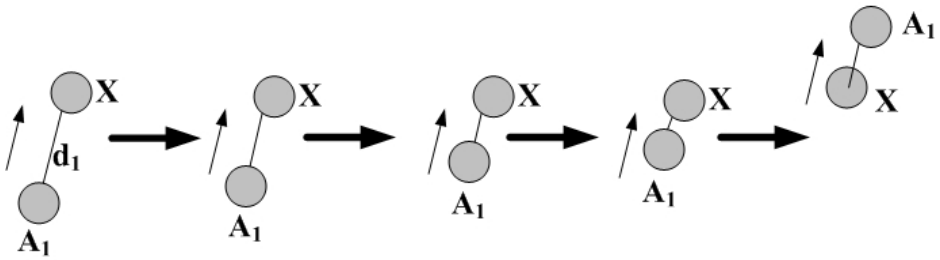
a) Kazanan proses elemanı doğru sınıfın bir üyesidir. Bu durumda ilgili ağırlıklar girdi vektörüne biraz daha yaklaştırılırlar. Bu, aynı örnek ağa tekrar gösterildiğinde yine aynı proses elemanının kazanması için yapılmaktadır. Bu durumda, ağırlıkların değiştirilmesi şu formüle göre yapılmaktadır.

$$A_y = A_e + \lambda(X - A_e) \tag{3.41}$$

Burada λ öğrenme katsayısıdır. Zaman içerisinde de sıfır değerini alacak şekilde monoton olarak azaltılır. Bunun nedeni girdi vektörünün referans vektörüne çok yaklaştığında durması ve aksi yönde tekrar uzaklaşmaması içindir. Aksi takdirde ters yönde tekrar uzaklaşma olacaktır. Bu durum şekil 3.11 ve şekil 3.12'de gösterilmiştir.




**Şekil 3.11 Girdi vektörüne en yakın ağırlık (referans vektörü (A1))**




**Şekil 3.12 Ağırlık vektörünün girdi vektörüne yaklaşması**

Şekil 3.11'de görüldüğü gibi, girdi vektörüne en yakın vektör $A_1$ vektörüdür. Bu vektörün sürekli X vektörüne yaklaştırılması zaman içinde onu geçerek ters yönde uzaklaşması anlamına gelecektir. Şekil 3.12'de bu durum vurgulanmak istenmektedir. Görüldüğü gibi $A_1$ vektörü sürekli X vektörüne yaklaşmaktadır. Belirli bir süre sonra bu iki vektör birbirine çok yakın (bazen üst üste) olmakta ve daha sonra $A_1$ vektörü tekrar X vektöründen uzaklaşabilmektedir. O nedenle, girdi ve ağırlık vektörleri birbirine çok yakın olduğunda ağırlıkların değişmemesi için öğrenme katsayısı sıfır değerine indirilmektedir. Burada tasarımcıların çok dikkatli olması gerekmektedir. Öğrenme katsayısının ne çok erken ne de çok geç sıfıra indirgenmemesi lazımdır. Eğitim süreci çok ayrıntılı incelenerek öğrenmenin ne zaman durdurulacağına karar verilmelidir [6].

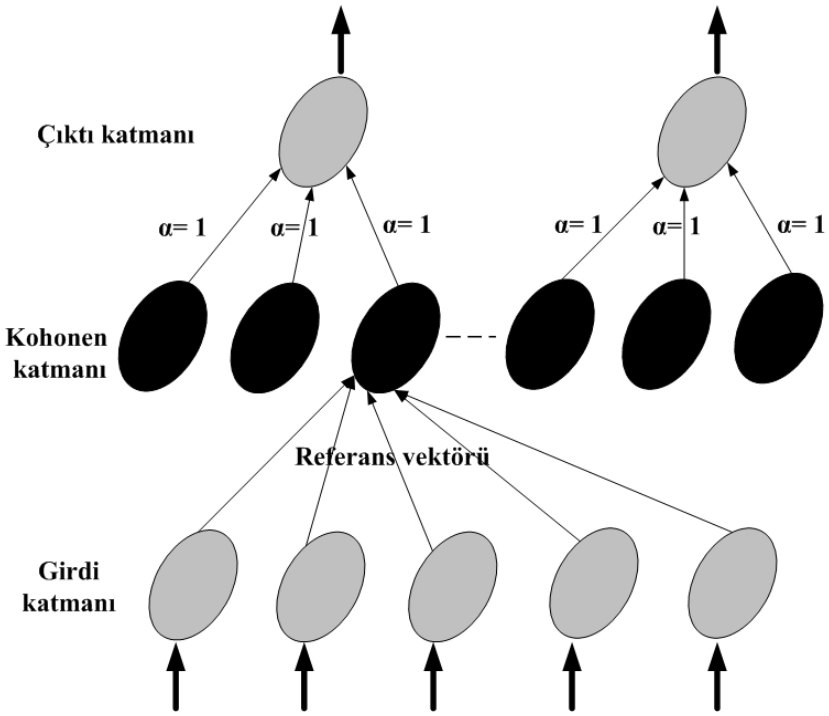
b) Kazanan proses elemanı yanlış sınıftandır. Bu durumda ağırlık vektörü girdi vektöründen uzaklaştırılır. Bir daha aynı örnek geldiğinde aynı proses elemanı kazanmasın diye bu uzaklaştırılma yapılmaktadır. Şu formül ile ağırlıklar değiştirilmektedir.

$$A_y = A_e - \lambda(X - A_e) \tag{3.42}$$

Öğrenme katsayısının zaman içinde sıfır değerini alacak şekilde monotonik olarak azalması burada da geçerlidir.

### 3.4.3. LVQ ağının eğitilmesi

LVQ ağının eğitilmesinde amaç her iterasyonda girdi vektörüne en yakın referans vektörünü bulmaktır. Referans vektörleri Kohonen katmanındaki proses elemanlarını girdi katmanındaki proses elemanlarına bağlayan ağırlık değerleridir. Şekil 3.13'de kohonen katmanındaki 3. proses elemanını girdi katmanına bağlayan referans vektörü görülmektedir.




**Şekil 3.13 Referans vektörü örneği**

Öğrenme esnasında sadece referans vektörlerinin ağırlık değerleri değiştirilir. Şekil 3.13 aynı zamanda girdi katmanı ile çıktı katmanı arasındaki

ağırlıkların (α) sabit ve 1 değerine sahip olduklarını da göstermektedir. Bu ağırlıklar eğitim sırasında da değiştirilmezler.

Öğrenme sırasında girdi katmanından gösterilen bir örnek sonucu referans vektörlerinin ağırlık değerleri kullanılarak girdi vektörü ile arasındaki mesafeleri hesaplanır. Bu mesafelerden en küçük değeri hangi proses elemanına ait referans vektörü üretiyor ise o proses elemanının çıktısı 1 diğerininki 0 olur. Yani kohonen katmanındaki her proses elemanının çıktısı $Ç^k$ ise,

$Ç_i^k = 1$ → Eğer i.nci proses elemanı yarışı kazanırsa

0 → Aksi halde

Kohonen katmanındaki proses elemanlarının çıktıları bu proses elemanlarını çıktı katmanına bağlayan ağırlık değerleri ile çarpılarak ağın çıktısı hesaplanır.

$$Ç_i = \sum_j Ç_j^k \alpha_{ki} \tag{3.43}$$

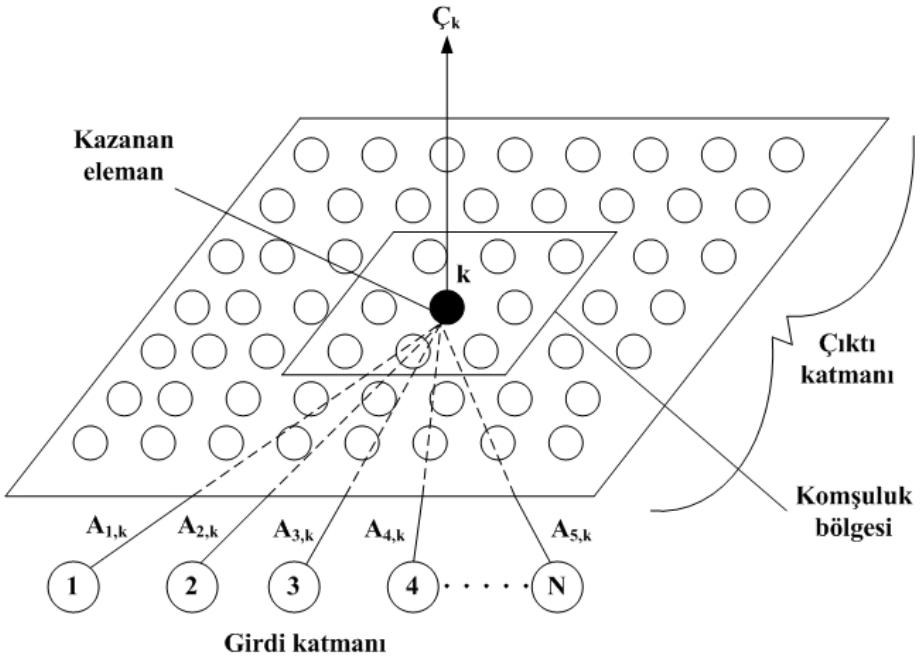
Bu, kohonen katmanında yarışmayı kazanan proses elemanına bağlı olan çıktı elemanının değerinin 1, diğerlerinin değerinin 0 olması anlamına gelmektedir.

### 3.5. Kendini Düzenleyen ağlar (SOM)

Som ağları Kohonen tarafından geliştirilmiştir. Genel olarak sınıflandırma yapmak için kullanılmaktadır. Bu ağların girdi vektörlerini sınıflandırma ve girdi vektörlerinin dağılımını öğrenebilme yetenekleri çok yüksektir.

Som ağlarının en temel özelliği olayları öğrenmek için bir öğretmene ihtiyaç duymaması veya ağın üretmesi gereken çıktıların ağa söylenme zorunluluğunun olmamasıdır. Özellikle beklenen çıktıların belirlenemediği problemler için kullanılmaktadır.

Yapısal olarak da bu ağlar diğerlerinden farklıdır. Ağ, girdi ve çıktı katmanından oluşmaktadır. Çıktı katmanı 2 boyutlu bir düzlemi göstermektedir. Proses elemanları bir düzlem üzerine dağılmış vektörleri gösterirler. SOM ağları yarışmayı kazanma ve kazanan elemanın 1, diğerinin 0 değerini alması ilkesine dayanmaktadır. Bir girdi verildiğinde çıktı uzayında yarışmayı kazanan ve onun etrafındaki komşuları eğitim sırasında ağırlıklarını değiştirmektedir. Som ağının yapısı şekil 3.14'te gösterilmiştir [6].




**Şekil 3.14 Som ağının gösterimi**

Şekilde sadece kazanan eleman ile girdi elemanlarının arasındaki bağlantılar gösterilmiştir. Aslında girdi elemanları çıktı elemanlarının tamamına bağlıdır.

### 3.5.1. SOM ağının eğitilmesi

Herhangi bir *t* zamanında örnek setinden bir örnek ağa gösterilir. Girdi vektörü X ve ağırlık vektörü A normalize edilmiş olmalıdır. Çıktı elemanlarından kazanan eleman bulunur. Bunun için iki yoldan birisi kullanılmaktadır:

1. Her elemanın çıktısı (Ç) ağırlıklarla girdilerin çarpımının toplamı ile bulunur. Yani;

$$Ç_i = \sum_i A_{i0} X_i \tag{3.44}$$

Bu çıktı değerinden en yüksek değere sahip olan proses elemanı yarışmayı kazanmaktadır. Bu elemanın k. eleman olması durumunda,

$Ç_k=1$

$Ç_i=0$ i=1,2,… ve i≠k

2. Öklit mesafesi (d) kullanılarak girdi vektörüne en yakın ağırlık vektörüne sahip eleman kazanan elemandır. İki vektör arasındaki mesafe aşağıdaki gibi hesaplanır.

$$d_j = \|X - A_j\| \tag{3.45}$$

Her çıktı elemanı için bu mesafeler hesaplanır ve en küçük mesafe değerine sahip eleman kazanan eleman olarak belirlenir.

Kazanan eleman belirlendikten sonra bu eleman ve komşularının ağırlıkları aşağıdaki formüle göre değiştirilmektedir:

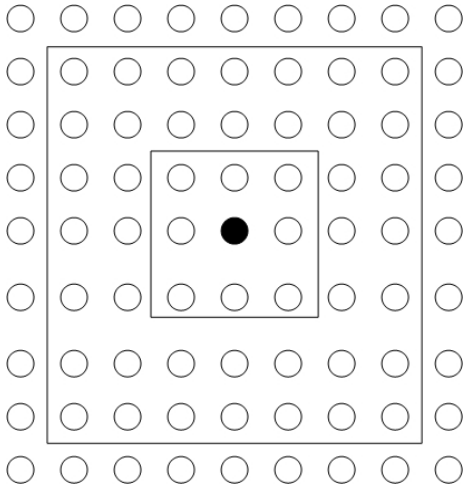
$$A(t+1) = A(t) + \lambda g(i,k)(X(t) - A(t)) \tag{3.46}$$

Burada λ öğrenme katsayısıdır. Öğrenme anında zamanla küçültülmektedir. g(i,k) ise komşuluk fonksiyonudur. i ve k elemanlarının komşuluklarını belirler. i=k durumunda g(i,k)=1 olur. Bu fonksiyon zaman içerisinde azalan bir fonksiyondur. Genel olarak g(i,k) aşağıdaki formüldeki gibi hesaplanmaktadır:

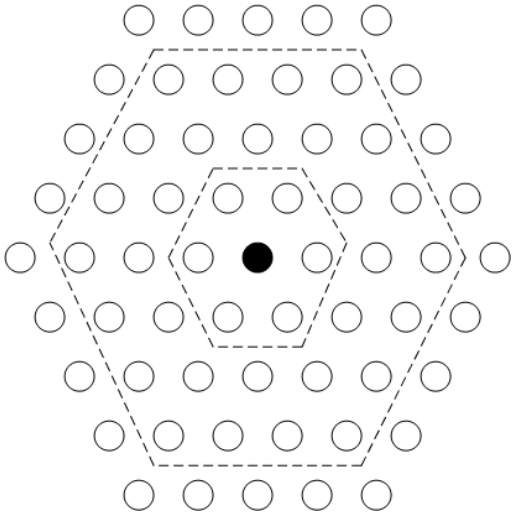
$$g(k) = (\exp(-\|d_i - d_k\|)^2 / (2\sigma^2)) \tag{3.47}$$

Formülde, $d_i$ ve $d_k$ i. ve k. elemanların pozisyonunu gösteren vektörlerdir. σ ise komşuluk alanının genişliğini göstermektedir ve zaman içerisinde azalmaktadır.

Bir proses elemanının komşularının (onunla beraber) ağırlıkları değişecek olanları belirlemede iki yöntem kullanılmaktadır. Biri Şekil 3.15'te görüldüğü gibi proses elemanının etrafındaki elemanları bir kare/dikdörtgen içinde belirlemek, diğeri de Şekil 3.16'da görüldüğü gibi proses elemanının etrafındakileri bir çokgen içinde belirlemektir.



**Şekil 3.15 Dörtgen komşuluk alanı**



**Şekil 3.16 Çokgen komşuluk alanı**

# 4. BAŞ AĞRISI HASTALIKLARI ve SINIFLANDIRILMALARI

Yaşamının herhangi bir döneminde baş ağrısından yakınmayan insan neredeyse yoktur. Ancak baş ağrılarını iki şekilde değerlendirmek gerekir. Birincisi başlı başına bir hastalık olarak baş ağrısı (**primer baş ağrısı**), ikincisi ise çeşitli hastalıkların bulgusu olan baş ağrısıdır (**sekonder baş ağrısı**). İkinci gruptaki baş ağrıları genellikle gözlerden, kulak, burun, boğaz hastalıklarından, dişlerden kaynaklanan baş ağrılarıdır. Genellikle bu tür baş ağrılarının teşhis ve tedavisi daha kolaydır [10].

Başağrıları, Uluslararası Başağrısı Derneği (IHS=International Headache Society) tarafından 1988 yılında 13 ana sınıfa ayrılmış ve o zamandan bu yana çok az değişikliğe uğramıştır. Bu sınıflandırma şu şekildedir [3]:

1. Migren
2. Gerilim tipi baş ağrıları
3. Küme baş ağrıları ve kronik paroksismal hemikrania
4. Yapısal lezyonlarla ilişkisi olmayan diğer baş ağrıları
5. Kafa travmalarına bağlı baş ağrıları
6. Damarsal bozukluklarla ilişkili baş ağrıları
7. Damarsal olmayan olaylarla ilişkili baş ağrıları
8. Madde bağımlılığı veya yoksunluklarına bağlı baş ağrıları
9. Ekstrakranial enfeksiyonlara bağlı baş ağrıları
10. Metabolik bozukluklara bağlı baş ağrıları
11. Kranium, boyun, kulaklar, sinüsler, dişler, ağız ve diğer kafa yapılarının bozukluklarına bağlı kafa ve yüz ağrıları
12. Kranial nevraljiler, sinir ağrıları ve deafferentasyon ağrıları
13. Sınıflandırılamayan baş ağrıları

1–4 numaralı baş ağrıları, primer baş ağrılarıdır. 5–13 numaralı baş ağrıları da sekonder baş ağrıları grubunu oluşturmaktadır. Yukarıdaki sınıflandırma, alt gruplarla birlikte EK-A'da detaylı biçimde verilmiştir.

## 4.1. Migren

Migren, farklı biçimlerde ortaya çıkan bir rahatsızlıktır. Migren rahatsızlığı, **"en şiddetli baş ağrısı"** olarak adlandırılmaktadır. Tekrarlayan ve şiddetiyle

süresi değişen bir baş ağrısı nöbetidir. 20 ile 40 yaş arası sık görülmekte olup elli yaşından sonra azalma eğilimi gösterir. **Migren** kalıtsal bir hastalıktır [11].

### 4.1.1. Migren belirtileri

Genellikle başın bir tarafında olan zonklayıcı ağrı öncesinde görme bozuklukları olabilir. Ardından sıkça mide bulantısı ve kusmalar görülür. Ağrı, göz çevresi, alın ya da şakakta duyulur.

İlk migren atağı normal olarak ergenlik döneminde olmakla beraber, bazıları daha **çocuk yaşta** bunu yaşarlar.

İki migren tipi vardır [11]:

1. **Önceden uyaran (Auralı) migren tipi**: Bu tip migren daha seyrektir ve tüm vakaların % 10-20' si kadarında görülür. Auralı migren öncesinde değişik görsel fenomenler (ışık çakmaları, zikzaklı çizgiler, boş noktalar, titreşimli görüntüler) olur. Anlaşılmaz şekilde konuşma, iğnelenme ile uyuşmalar (felç belirtisi), susama ve esnemeler de görülebilir. Bu durum yaklaşık 30 dak. sürebilir. Aura sonrasında gelen bu şiddetli baş ağrısı, aura ile birlikte ya da öncesinde de gelebilir.
2. **Aurasız migren:** Bu baş ağrısı öncesinde belli bulgular yoktur. Çokları kendilerini kötü hissederler ve çabuk sinirlenirler. Bazılarında iştah değişmeleri görülür. Migren ataklarının süresi yarım saatten üç güne kadar uzayabilir.

### 4.1.2. Migreni başlatan faktörler

- Stres (psikolojik ve fiziksel)
- Uyku (çok ya da az uyumak)
- Düzensiz yemek saatleri ve perhiz
- İçinde fazla tiramin maddesi olan yiyecekler (kırmızı şarap ve sert peynir çeşitleri)
- Narenciye ve çikolata gibi yiyecekler
- Kanda şekerin düşmesi
- Hormonlarla ilgili etmenler (âdet görme, yumurtacık oluşumu ya da doğum kontrol hapı)

- Çevre koşulları (sıcak, soğuk, ışık, ses, koku ve hava basıncındaki değişmeler)
- Ağrı (gerilim tipi baş ağrısı)
- Bedensel zorlanma
- Alerji

### 4.2. Gerilim Tipi Baş Ağrısı (Tension Type Headache)

Çok sık görülen bir rahatsızlıktır. İnsanların yaklaşık %80-90 kadarı, bunu yaşamışlardır. Gençlerle ve çocuklarda da görülür. Hafif ve tazyikli olan gerilim tipi baş ağrısı, ağır ağır yerleşir, kademeli olarak artar ve yine ağır ağır kaybolur. Başın etrafında çember tarzında gergin bir şerit ya da çok dar bir kasket gibi hissedilir. Basınç alın, şakaklar ve boyunda hissedilir. Saç dipleriyle çenede hassas noktalar oluşabilir. Boyun kasları gerili ve ağrıyor olabilir. Ağrılar, 30 dakikadan yedi güne kadar sürebilir [11].

#### 4.2.1. Gerilim tipi baş ağrısını başlatan faktörler

- Değişik türde stresler: Aile yaşamı, ilişkiler, başarısız olma endişesi v.s.
- Yorgunluk, uykusuzluk
- Kötü pozisyonda çalışmak
- Bilgisayarın önünde uzun süre kalmak
- Görme bozukluğu
- Kötü/yetersiz ışıklandırma
- Kullanılan herhangi bir ilaç
- Depresyon hali

### 4.3. Aurasız Migrenden Deforme Baş Ağrısı (Transformed Migraine)

Bazı hastalar hem migren hem de gerilim tipi baş ağrısına sahip olarak düşünülebilmektedir. Migrenden deforme baş ağrısı bu hastalara ne olduğunu daha iyi açıklamaktadır. Basit anlamda migrenli önemli bir azınlık hasta ileride günlük gelişen baş ağrıları (migrenden deforme baş ağrısı) çekebilmektedir. Bu teşhisin yapılması için hastanın kronik günlük gelişen baş ağrısından önceki baş ağrısı ataklarının karakteri belirlenmelidir. Çoğu hasta kronik günlük baş ağrısı gelişiminden önceki dönemde açık bir migren tipi baş ağrısı hikâyesi vermektedir.

Ayrıca teşhiste yardımcı olabilecek bir diğer şey de baş ağrısının şiddetlenme karakteridir. Eğer bu şiddetlenmeler atakların oldukça uzun olması dışında migren teşhis kriterlerini sağlarsa migrenden deforme baş ağrısı hastalığından söz edilebilmektedir [12].

### 4.3.1. Aurasız migrenden deforme baş ağrısı için tanı kriterleri

- 1 aydan fazla bir zaman içinde 15 günden fazla meydana gelen baş ağrıları
- Ortalama baş ağrısı devam etme süresi günde 4 saatten fazla
- Aşağıdakilerden birini sağlayan koşullar
    1. Episodik migren belirtilerin olması
    2. Baş ağrısı şiddetlerinin migren tanı kriterlerini sağlaması (uzun süreli olması dışında)
- Başka açık bir şikâyetin olmaması

## 4.4. Teşhis

Baş ağrısı hastalıklarının teşhisindeki en güçlü yöntem, hastanın iyice bir hikâyesinin alınmasıdır. Hastanın rahat anlayıp açıklıkla cevap verebileceği şekilde teşhise götürücü bilgilerin alınması gerekir. Hikâye almada özellikle baş ağrısının başlangıç şekli, gün içindeki zamanı, süresi, ağrı tipi, sıklığı, eşlik eden bulguları, yeri ve yayılımı, ağrıyı azaltan ve artıran faktörler sorulmalıdır [3].

# 5. BAŞ AĞRISI HASTALIKLARININ SINIFLANDIRILMA ÇALIŞMASI

## 5.1. Belirtilerin Sınıflandırılması

Uluslar arası Baş Ağrısı Derneği (IHS) tarafından yapılan sınıflandırma, Ek-A'da verilmiştir. Bu tez çalışmasında, baş ağrısı hastalıklarının bütün belirtileri sistemli bir şekle kavuşturularak belirlenmeye çalışılmıştır. Belirtiler üç ana başlık altında toplanmıştır:

1. **Şikâyetler**: Hastanın hikâyesinden çıkarılan sonuçlardır. Baş ağrısının karakteri, şiddeti, yerleşimi, süresi, periyodu, aura belirtileri, aura belirtilerinin gelişimi, baş ağrısının ortaya çıkış zamanı, pozisyon ile ilişkisi, başlama şekli, eşlik ettiği durumlar, prodromal ve postdromal belirtiler, baş ağrısını düzelten tedavi yöntemleri ve aile hikâyesi; hastanın anlattıklarını dinleyen doktor tarafından dinlenir. Her insanın ağrıya karşı tutumu farklıdır. Bazıları ağrısı çok şiddetli olmasına rağmen fazla şiddetli olmadığını söylemekte, bazıları da hafif baş ağrılarını çok şiddetli olarak tarif etmektedirler. Hasta şikâyetleri, uzman bir doktor tarafından yöresel ve kişisel faktörler de göz önünde tutularak dikkatlice öğrenilmelidir. Hastanın derdini rahatça ve açıklıkla anlatması sağlanmalıdır. Baş ağrısı hastalıklarının teşhisinde en büyük yardımcı, hastanın hikâyesidir.
2. **Bulgular**: Ataklar arasında hastanın tetkik edilmesiyle uzman doktorların gözlemlediği, şikâyetlerden daha teknik tarafı olan ve daha bilimsel belirtilerdir.
3. **Laboratuar Bulguları**: Ataklar arasındaki nörolojik muayenede anormallikler belirlenirse laboratuar sonuçlarına ihtiyaç duyulabilir.

Eskişehir Osmangazi Üniversitesi Nöroloji Bölümü doktorlarından Prof. Dr. Nevzat UZUNER ile birlikte yapılan belirtileri belirleme ve sınıflandırma çalışmasının sonucunda bulunan ve hasta verileri toplamak için form haline getirilen başlangıç belirtiler kümesi Ek-B'de gösterilmiştir. Bu çalışmanın amacı, hastalık belirtilerinin sistematik bir biçimde listelenmesi, kodlanması, gruplanması ve tutarlı bir veri topluluğuna ulaşabilmek için gerekli alt yapının sağlanmasıdır. Hastalık belirtilerinin kodlanması sayesinde verilerin sayısallaştırılması işlemi kolaylaşmıştır.

## 5.2. Hastalıklar ve Belirtileri

Baş ağrısı hastalıklarının belirtilerinin oluşturulmasından sonra, baş ağrısı hastalıklarının bu belirtilerle eşleştirilmesi üzerinde çalışılmıştır. Bunu yaparken de eldeki 4 hastalık türüne ait tanı kriterlerinden ve uzman doktordan alınan bilgilerden yararlanılmıştır. Eldeki hastalıkların temel özellikleri ve tanı kriterlerinden 4. bölümde bahsedilmektedir.

## 5.3. Yapay Sinir Ağları ile Teşhis

### 5.3.1. Örneklerin Toplanması

Baş ağrısı hastalıklarının teşhisi için düşünülen yapay sinir ağı çözümü için bol miktarda örneğe ihtiyaç duyulmaktaydı. Eskişehir Osmangazi Üniversitesi Nöroloji Polikliniği'ne gelen 47 hastanın uzman doktorlar tarafından toparlanan belirtileri, teşhisleriyle birlikte alındı.

İkincil baş ağrıları hastalıkları, nöroloji polikliniğini direkt olarak ilgilendirmediği için başka polikliniklere havale edildiğinden ve zaten bu tür başvurular hemen hemen hiç yapılmadığından, örnekler arasında ikincil baş ağrısı teşhisi konulan hiçbir hasta yoktu.

Böylece yapay sinir ağları çözümü için en sık rastlanan dört hastalığın incelenmesi gereği ortaya çıktı. Burada belirtmek gerekir ki, daha fazla örnek temin edilebilirse sadece bu 4 hastalık değil, bütün baş ağrısı hastalıkları yapay sinir ağları ile sınıflandırmaya tabi tutulabilir.

Az sayıda hastalıkla çalışmanın avantajlarından biri, giriş ve çıkış sayıları azaldığı için yapay sinir ağının daha güzel ve çabuk sonuçlar verebilecek olmasıdır. Ayrıca nöroloji polikliniğine gelen hastaların büyük bir kısmı bu hastalıklardan birine yakalandığı için bir kayıp söz konusu değildir.

### 5.3.2. Belirtilerin Daraltılması

Hastalıklar daraltılınca, belirtilerin de daraltılması gündeme geldi. Teşhiste payı olmayan belirtilerin giriş verileri arasına alınması sistemin eğitimini yavaşlatacaktı. Elimizdeki verileri oluşturan dört hastalık olan aurasız migren, auralı migren, gerilim tipi baş ağrısı ve aurasız migrenden deforme baş ağrısı için

teşhiste önemli olan belirtiler 1 kodlu şikâyetler ve 2 kodlu bulguların birleşimi şeklinde aşağıdaki gibi listelenmiştir:

1.01 Baş ağrısının karakteri
1.02 Baş ağrısının şiddeti
1.03 Tutulan taraf
1.04 Baş ağrısının yerleşimi
1.05 Bir baş ağrısının süresi
1.06 Aynı tipteki iki baş ağrısı arasındaki süre
1.07 Baş ağrısından önce veya beraber (Aura belirtileri)
1.08 Aura belirtileri süresi
1.09 Baş ağrısının ortaya çıkış zamanı
1.10 Baş ağrısının pozisyon ile ilişkisi
1.11 Baş ağrısının başlama şekli
1.12 Baş ağrısının şiddeti her defasında nasıl değişiyor
1.13 Prodromal belirtiler
1.14 Postdrom belirtiler
1.15 Baş ağrısı ne ile düzeliyor
1.16 Aynı şekildeki baş ağrılarının sayısı
1.17 Aynı şekildeki baş ağrılarının toplam sayısı
1.18 Ailede aynı tipte baş ağrısı
2.01 Ataklar arası nörolojik muayene
2.02 Baş ağrısı ile beraber gerçekleşen belirtiler

Belirtiler değer kümelerine göre sayısal veriler haline dönüştürülebilmektedir. Daha sonra yapay sinir ağının uyumunu kolaylaştırabilmek için her bir belirtinin içerdiği değerler kümesi kendi içinde karşılaştırılıp aşağıdaki şekillerde sadeleştirmeye gidilmiştir. Aşağıdaki sınıflandırmalara gidilmesinin bir nedeni de bir belirtinin birden fazla değer alma durumu olduğunda da belirtinin değerinin sayısal verilere dönüştürülebilmesinin ve dolayısıyla yapay sinir ağının oluşturulabilmesinin sağlanmasıdır.

Uzman doktordan alınan bilgiler ve bunlara bağlı olarak veriler üzerinde yapılan incelemeler sayesinde belirtiler ve değer kümeleri içinde aşağıdaki değişiklikler gerçekleştirilmiştir:

- 1.01 belirtisi için "zonklayıcı" karakterinin değer kümesindeki diğer elemanlara göre belirgin derecede önemli olduğu saptanmıştır. Eldeki hasta verilerinin de incelenmesi sonucu 1.01 belirtisi için şu şekilde bir sınıflandırmaya gidilmiştir:

  0 → Zonklayıcı değil ise

  1 → Zonklayıcı ise

- 1.04 belirtisi için en önemli özelliğin "ağrının başın tamamında olup olmaması" olduğu şeklinde bilgi alınmış ve buna göre de 1.04 belirtisi için şu şekilde bir sınıflandırmaya gidilmiştir:

  0 → Baş ağrısı yerleşimi başın tamamında değil ise

  1 → Baş ağrısı yerleşimi başın tamamında ise

- 1.05 belirtisinde migren için önemli olan zaman dilimi "30 dakika-7 gün" iken gerilim tipi için "4-72 saat" olmaktadır. Bu ikisi de düşünülerek 1.05 belirtisi için veriler sayısallaştırılırken aşağıdaki gibi bir gruba ayrılmıştır.

  1 → 0.5-4 saat (4 dahil değil)

  2 → 4-72 saat (4-72 dahil)

  3 → 72 saat-7 gün (7 gün dahil)

  4 → 7 gün-1 ay

- 1.06 belirtisi için ise eldeki veriler incelendikten sonra iki baş ağrısı arasındaki sürenin "14 günden fazla" veya "14 günden az" olmasının en etkili özellik olduğuna karar verilmiş ve aşağıdaki şekilde sınıflandırılmaya gidilmiştir:

  0 → Aynı tipte iki baş ağrısı arası süre 14 günden az ise

  1→ Aynı tipte iki baş ağrısı arası süre 14 günden fazla ise

- 1.07 belirtisinin auralı migren'i ayırmak için kullanılabileceği gözlendiğinden bu belirti sadece var ya da yok şekline sokulmuştur:

  0 → Aura belirtileri yok ise

  1 → Aura belirtiler var ise

- 1.08'deki aura belirti süresi için ise aşağıdaki sınıflandırılmaya gidilmiştir:

0 → 1 saatten uzun sürmüyor

1 → 1 saatten uzun sürüyor

- 1.09'daki baş ağrısının ortaya çıkış zamanı eldeki veriler göz önünde bulunarak incelenmiş ve hastaların büyük çoğunluğunda "gün içinde uyanıkken" şeklinde değer aldığı saptanmıştır. Sonuçta eldeki 4 hastalık için ayırt edici bir özellik olmadığı gözlenmiş ve kaldırılmasına karar verilmiştir.
- 1.13'teki prodromal belirti olarak hastaların hemen hemen hepsinde "duyarlılığın artması" özelliğinin, 1.14'teki postdrom belirti için ise "başında ağırlık hissi" özelliğinin sağlandığı görülmüş ve bunların ayırt edici özellik olmadığına karar verilmiştir. Bunun sonucunda da bu iki belirtinin sistemden çıkartılmasına karar verilmiştir.
- 1.15'teki baş ağrısının düzelme şekli olarak çoğu hastada "uyku ile düzeliyor" ve "tekoin ile düzeliyor" özelliklerinin birlikte sağlandığı görülmüştür. Bunun da sistem için ayırt edici bir özellik olamayacağı düşünüldüğünden kaldırılmasına karar verilmiştir.
- 1.16 ile 1.17 belirtilerinin birinin diğerinin tutarlılığını kontrol etme amaçlı olduğu uzman doktordan öğrenildikten sonra bunlardan birinin yani 1.16 belirtisinin kaldırılmasına karar verilmiştir. 1.17 belirtisi de daha dar bir değer kümesine sahip olabilmesi açısından veriler incelendikten sonra şu şekle sokulmuştur:

  1 → 1-10 arası

  2 → 11-50 arası

  3 → 51-100 arası

  4 → 101-150 arası

  5 → 151-200 arası

  6 → 201-300 arası

  7 → 301 – 500 arası

  8 → 501 ve sonrası

- 1.18 belirtisinde ise değer kümesi önce ailede var ya da yok şeklinde iki değer alır hale getirilmiştir. Ancak bu durumda da belirtinin teşhis için önemli olamayacağı saptanmış ve çıkartılmasına karar verilmiştir.

- 2.01 belirtisinin uygulanan temel bileşenler analizi [16] sonrasında genellikle aynı değeri alması sonucunda sistem tarafından atıldığı gözlenmiş ve çıkartılmasına karar verilmiştir.
- 2.02 belirtisi için ise eldeki veriler doğrultusunda uzman doktora danışılmış ve bulantı, kusma, fotofobi ve fonofobi özelliklerinin migren tipi hastalıkların tanısı için kullanıldığı gözlenmiştir. Sonuç olarak eldeki veriler de daha çok bu özellikleri taşıdığından bu belirti 0 ve 1 değerlerini alan 4 farklı belirti şeklinde ayrılmıştır.

Çizelge 5.1 ve 5.2'de 16 belirti içeren hasta verileri gözlenmektedir:

**Çizelge 5.1 Hasta Örnekleri (Eğitim İçin Kullanılan)**

| | Baş ağrısının karakteri | Baş ağrısının şiddeti | Tutulan taraf | Baş ağrısının yerleşimi | Bir baş ağrısının süresi | Aynı tipteki iki baş ağrısı arasındaki süre | Baş ağrısından önce veya beraber aura belirtileri | Aura belirtileri şu kadar sürede gelişiyor | Baş ağrısının pozisyon ile ilişkisi | Baş ağrısı başlama şekli | Baş ağrısının şiddeti her defasında | Aynı şekilde baş ağrılarının toplam sayısı | Bulantı var mı? | Kusma var mı? | Fotofobi var mı? | Fonofobi var mı? | Teşhis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 1 | 2 | 1 | 4 | 0 | 0 | 0 | 4 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | **Aurasız Migren** |
| **2** | 1 | 3 | 1 | 1 | 3 | 0 | 0 | 0 | 1 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | **Aurasız Migren** |
| **3** | 1 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 4 | 3 | 1 | 8 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **4** | 1 | 3 | 1 | 0 | 3 | 1 | 0 | 0 | 4 | 1 | 1 | 5 | 1 | 0 | 0 | 0 | **Aurasız Migren** |
| **5** | 1 | 4 | 2 | 0 | 2 | 0 | 0 | 0 | 1 | 1 | 1 | 8 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **6** | 1 | 3 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 1 | 1 | 7 | 1 | 1 | 1 | 0 | **Gerilim Tipi** |
| **7** | 1 | 2 | 2 | 0 | 2 | 1 | 0 | 0 | 4 | 2 | 1 | 1 | 1 | 0 | 1 | 1 | **Aurasız Migren** |
| **8** | 0 | 3 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | **Gerilim Tipi** |
| **9** | 1 | 3 | 2 | 0 | 4 | 0 | 0 | 0 | 4 | 1 | 1 | 7 | 1 | 0 | 0 | 1 | **Gerilim Tipi** |
| **10** | 1 | 2 | 1 | 0 | 2 | 1 | 0 | 0 | 2 | 2 | 1 | 7 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **11** | 1 | 2 | 1 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | 1 | 0 | 0 | 1 | **Gerilim Tipi** |
| **12** | 1 | 3 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 1 | 4 | 3 | 0 | 0 | 0 | 1 | **Gerilim Tipi** |
| **13** | 1 | 4 | 1 | 0 | 3 | 1 | 0 | 0 | 4 | 1 | 1 | 6 | 1 | 0 | 1 | 1 | **Aurasız Migren** |
| **14** | 1 | 3 | 1 | 0 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | **Aurasız Migren** |
| **15** | 1 | 3 | 1 | 0 | 3 | 0 | 0 | 0 | 4 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | **Gerilim Tipi** |

**Çizelge 5.1 Hasta Örnekleri (Eğitim İçin Kullanılan) (devamı)**

| | Baş ağrısının karakteri | Baş ağrısının şiddeti | Tutulan taraf | Baş ağrısının yerleşimi | Bir baş ağrısının süresi | Aynı tipteki iki baş ağrısı arasındaki süre | Baş ağrısından önce veya beraber aura belirtileri | Aura belirtileri şu kadar sürede gelişiyor | Baş ağrısının pozisyon ile ilişkisi | Baş ağrısı başlama şekli | Baş ağrısının şiddeti her defasında | Aynı şekilde baş ağrılarının toplam sayısı | Bulantı var mı? | Kusma var mı? | Fotofobi var mı? | Fonofobi var mı? | Teşhis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **16** | 1 | 3 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 3 | 3 | 4 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **17** | 1 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 2 | 1 | 6 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **18** | 1 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 2 | 1 | 3 | 0 | 0 | 1 | 0 | **Aurasız Migren** |
| **19** | 1 | 3 | 2 | 0 | 2 | 0 | 0 | 0 | 4 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | **Aurasız Migren** |
| **20** | 0 | 2 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | **Gerilim Tipi** |
| **21** | 1 | 3 | 1 | 1 | 2 | 0 | 1 | 1 | 1 | 3 | 1 | 5 | 1 | 0 | 1 | 1 | **Auralı Migren** |
| **22** | 1 | 3 | 1 | 1 | 3 | 1 | 1 | 1 | 4 | 1 | 1 | 4 | 1 | 0 | 1 | 1 | **Auralı Migren** |
| **23** | 1 | 2 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 3 | 1 | 2 | 1 | 0 | 1 | 1 | **Aurasız Migren** |
| **24** | 1 | 3 | 1 | 0 | 2 | 0 | 1 | 1 | 1 | 3 | 1 | 3 | 1 | 0 | 1 | 1 | **Auralı Migren** |
| **25** | 1 | 3 | 2 | 0 | 2 | 1 | 0 | 0 | 4 | 1 | 3 | 4 | 1 | 0 | 0 | 1 | **Aurasız Migren** |
| **26** | 0 | 2 | 2 | 1 | 2 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | **Gerilim Tipi** |
| **27** | 0 | 2 | 2 | 0 | 3 | 0 | 0 | 0 | 4 | 1 | 1 | 8 | 1 | 0 | 0 | 1 | **Aurasız Migren** |
| **28** | 1 | 3 | 2 | 1 | 3 | 0 | 0 | 0 | 1 | 1 | 1 | 8 | 1 | 1 | 1 | 1 | **A.sız Migrenden Deforme** |
| **29** | 1 | 4 | 2 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **Auralı Migren** |
| **30** | 1 | 1 | 1 | 1 | 3 | 0 | 0 | 0 | 1 | 3 | 4 | 8 | 1 | 1 | 1 | 1 | **A.sız Migrenden Deforme** |
| **31** | 1 | 2 | 2 | 1 | 2 | 1 | 0 | 0 | 1 | 3 | 1 | 4 | 1 | 0 | 0 | 1 | **Aurasız Migren** |
| **32** | 1 | 2 | 2 | 1 | 3 | 0 | 0 | 0 | 4 | 3 | 1 | 2 | 0 | 0 | 0 | 0 | **Gerilim Tipi** |
| **33** | 0 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 3 | 1 | 8 | 0 | 0 | 0 | 1 | **Gerilim Tipi** |
| **34** | 1 | 1 | 1 | 1 | 2 | 0 | 0 | 0 | 1 | 3 | 1 | 2 | 1 | 0 | 0 | 1 | **Aurasız Migren** |
| **35** | 0 | 2 | 2 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | 1 | 0 | 0 | 0 | **Aurasız Migren** |
| **36** | 0 | 2 | 2 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | **Aurasız Migren** |
| **37** | 1 | 3 | 1 | 0 | 1 | 1 | 1 | 0 | 4 | 3 | 3 | 2 | 0 | 0 | 1 | 1 | **Aurasız Migren** |
| **38** | 1 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | **Gerilim Tipi** |

**Çizelge 5.2 Hasta Örnekleri (Test İçin Kullanılan)**

| | Baş ağrısının karakteri | Baş ağrısının şiddeti | Tutulan taraf | Baş ağrısının yerleşimi | Bir baş ağrısının süresi | Aynı tipteki iki baş ağrısı arasındaki süre | Baş ağrısından önce veya beraber aura belirtileri | Aura belirtileri şu kadar sürede gelişiyor | Baş ağrısının pozisyon ile ilişkisi | Baş ağrısı başlama şekli | Baş ağrısının şiddeti her defasında | Aynı şekilde baş ağrılarının toplam sayısı | Bulantı var mı? | Kusma var mı? | Fotofobi var mı? | Fonofobi var mı? | Teşhis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 3 | 1 | 0 | 3 | 1 | 0 | 0 | 4 | 3 | 1 | 2 | 1 | 1 | 1 | 1 | **Aurasız Migren** |
| **2** | 1 | 2 | 1 | 0 | 2 | 1 | 0 | 0 | 4 | 2 | 3 | 4 | 1 | 1 | 1 | 1 | **Aurasız Migren** |
| **3** | 0 | 2 | 0 | 1 | 2 | 0 | 0 | 0 | 4 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | **Aurasız Migren** |
| **4** | 1 | 3 | 1 | 0 | 2 | 0 | 1 | 1 | 4 | 3 | 1 | 6 | 1 | 0 | 1 | 1 | **Auralı Migren** |
| **5** | 1 | 3 | 2 | 0 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | **Auralı Migren** |
| **6** | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 4 | 3 | 1 | 3 | 0 | 0 | 0 | 0 | **Gerilim Tipi** |
| **7** | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 4 | 0 | 0 | 0 | 0 | **Gerilim Tipi** |
| **8** | 1 | 4 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 3 | 6 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |
| **9** | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 3 | 3 | 8 | 1 | 0 | 1 | 1 | **A.sız Migrenden Deforme** |

Hasta verileri içinde 4 hastalık türüne ait 47 adet veri bulunmaktadır. Bunlardan 19 tanesi aurasız migrene, 6 tanesi auralı migrene, 13 tanesi gerilim tipi baş ağrısına ve 9 tanesi de aurasız migrenden deforme baş ağrısına aittir.

### 5.3.3. Perseptron ile çözüm

Baş ağrılarının teşhisinde kullanılabilmek için 16 girişli 2 nöronlu yapıda bir ağ tasarlanmıştır. 16 girişin her birinin değer kümesine göre değer aralıkları mevcuttur. Nöronların çıkışları da basamak transfer fonksiyonu nedeniyle 0 veya 1 değerini alabildiği için 2 nöronun çıkışlarında dört durum söz konusudur: 0-0, 0-1, 1-0, 1-1. Teşhis edilecek hastalık sayısı da 4 olduğu için 2 nöronlu yapı yeterli olmuştur.

Sistemin eğitilmesi için Çizelge 5.1'de verilen 38 adet örnek kullanılmıştır. Çizelge 5.2'de verilen 9 örnek de test için ayrılmıştır. 100 epok sonucunda hata oranı 0.1 civarlarında kalmış ve bu şekilde devam etmiştir. Eğitim verileri sisteme geri verildiğinde bunlardan 28'inin doğru 10'unun ise yanlış teşhis edildiği görülmüştür. Test verileri için ise 8 tanesine doğru 1 tanesine yanlış teşhis konulduğu gözükmektedir. Bunun yanı sıra test verilerinde değişiklik yapıldığında başarı oranının daha da düştüğü gözlenmiştir. Perseptronun, yapısı itibariyle doğrusal ayrılabilir veri toplulukları üzerinde çok iyi sonuç verdiği bilinmektedir. Test verilerine iyi sonuç vermesine karşın eğitim verilerine verdiği yanıt ve hata oranı itibariyle eldeki veri topluluğunun yapı itibariyle tam olarak doğrusal ayrılabilir olmadığı anlaşılmaktadır.

### 5.3.4. Geri yayılım ağları ile çözüm

Geri yayılım algoritmasında eğitim için farklı fonksiyonlar kullanılabilmektedir. Bunların çeşitleri ve matlab ortamında kullanılan fonksiyon isimleri Çizelge 5.3'te verilmiştir.

**Çizelge 5.3 Geri yayılım algoritmaları**

| **İsim** | **Fonksiyon** |
|---|---|
| **Gradient Descent (Widrow-Hoff)** | traingd |
| **Gradient Descent with Momentum** | traingdm |
| **Variable Learning** | traingda |
| **Variable Learning with Momentum Extension** | traingdx |
| **Resilient Backpropagation** | trainrp |
| ***Conjugate Gradient Algorithms*** | |
| **Fletcher-Reeves Update** | traincgf |
| **Polak-Ribiere Update** | traincgp |
| **Powell-Beale Restarts** | traincgb |
| **Scaled Conjugate Gradient** | trainscg |
| ***Quasi-Newton Algorithms*** | |
| **BFGS Algorithm** | trainbgf |
| **One Step Secant Algorithm** | trainoss |
| **Levenberg-Marquardt Algorithm** | Trainlm |

Oluşturulacak geri yayılım ağının çıkış katmanında doğrusal transfer fonksiyonlu 4 adet nöron kullanılmış ve hastalıklardan hangisi teşhis edilmişse ona karşılık gelen nöronun 1 değerini alması istenmiştir. 1'e en yakın değer veren nöronun değeri 1'e yuvarlanmış ve bu nöron hangi hastalık türünü temsil ediyorsa onun teşhis edildiği yargısına varılmıştır.

Hastalık teşhislerinin örüntü tanıma problemlerine dâhil olması, bu tip problemlerde 2 adet gizli katman kullanılmasının daha verimli olduğunun bilinmesi nedeniyle bu şekilde bir yapı oluşturulmuş [16] ve gizli katman transfer fonksiyonu olarak sigmoid tanjant fonksiyonu seçilmiştir. Gizli katmanlarda ise 15 nöronlu bir yapı kurmanın olumlu sonuçlar verdiği yapılan denemeler sonucunda görülmüş ve Çizelge 5.4'de verilen algoritmalar için bu yapı kullanılmıştır. Bu algoritmaların 100, 1000 ve 10000 epok sonundaki ortalama kare hataları Çizelge 5.4'te verilmiştir.

**Çizelge 5.4 Geri yayılım algoritmaları ile eğitimde ortalama kare hataları**

| Fonksiyon | Epok | MSE | Epok | MSE | Epok | MSE |
|---|---|---|---|---|---|---|
| **traingd** | 100 | 0.83 | 1000 | 0.235 | 10000 | 0.0056 |
| **traingdm** | 100 | 0.79 | 1000 | 0.229 | 10000 | 0.0039 |
| **traingda** | 100 | 0.192 | 1000 | 0.00148 | 10000 | 0.00000054 |
| **traingdx** | 100 | 0.103 | 1000 | 0.000162 | 10000 | 0.00000164 |
| **trainrp** | 100 | 0.00080 | 1000 | 0.0000062 | 10000 | 0.00000002 |
| **traincgf** | 100 | 0.000195 | 560 | 0 | - | |
| **traincgp** | 100 | 0.000208 | 560 | 0 | - | |
| **traincgb** | 100 | 0.000162 | 469 | 0 | - | |
| **trainscg** | 100 | 0.000387 | 1000 | 0.0000007 | 4245 | 0 |
| **trainbfg** | 100 | 0.000018 | 387 | 0 | - | |
| **trainoss** | 100 | 0.00140 | 957 | 0 | - | |
| **trainlm** | 12 | 0 | - | | - | |

Geri yayılım algoritmalarının eğitim sonuçlarına bakıldığında, en yavaş algoritmanın Widrow-Hoff (traingd) algoritması olduğu görülmektedir. 10000 epok sonunda bile ortalama kare hata yeterince düşmemektedir. Sonuca en hızlı

ulaşan algoritma ise Levenberg-Marquardt (trainlm) algoritmasıdır. Sadece 12 epok sonunda eğitim sona erebilmektedir.

10, 100 ve 1000 maksimum epok sayıları verildiğinde test için ayrılan 9 örnekten rastgele 10 eğitim sonunda ortalama kaç tanesinin yanlış teşhis edildiği, Çizelge 5.5'te gösterilmiştir. İlk iki metod olan traingd ve traingdm fonksiyonları sonuca yavaş ulaşmasına rağmen epok sayısı arttırıldığında gittikçe daha doğru sonuçlar vermektedirler. Trainrp fonksiyonunda ise daha hızlı bir eğitim süreci ve bu ikisinden daha başarılı sonuçlar ortaya çıktığı görülmektedir. Diğer fonksiyonlar sonuca daha hızlı yaklaşmalarına rağmen maksimum epok sayıları arttırıldığında yanlış teşhis sayısında çarpıcı bir azalma görülmemektedir. Kare hata oranı kabul edilebilir orana gelip eğitim tamamlandığında eğitim için kullanılan 38 veri de sisteme sınıflandırılmak üzere verilmiş ve bunların hatasız sınıflandırıldığı görülmüştür.

Yukarıdaki yapay sinir ağlarının oluşturulduğu ve sonuçları almakta kullanılan MATLAB program kodları Ek-C'de verilmiştir.

**Çizelge 5.5 Geri yayılım fonksiyonlarının test örneklerine verdiği ortalama yanlış sayıları**

| **Fonksiyon ismi** | **Maksimum Epok Sayıları** | | |
|---|---|---|---|
| | **100** | **1000** | **10000** |
| **Traingd** | 5.1 | 3.0 | 2.0 |
| **Traingdm** | 4.8 | 2.8 | 2.3 |
| **Traingda** | 2.5 | 2.1 | 2.5 |
| **Traingdx** | 2.4 | 2.1 | 2.3 |
| **Trainrp** | 2.5 | 1.6 | 1.5 |
| **Traincgf** | 2.3 | 2.0 | - |
| **Traincgp** | 2.4 | 1.8 | - |
| **Traincgb** | 2.6 | 1.8 | - |
| **Trainscg** | 2.1 | 1.9 | 2.3 |
| **Trainbfg** | 1.7 | 1.6 | - |
| **Trainoss** | 2.2 | 2.1 | - |
| **Trainlm** | 2.4 | - | - |

### 5.3.5. Destekleyici öğrenmeli vektör nicemleme ağları (LVQ) ile çözüm

Lvq ile yapılan denemeler sonucunda da hatanın 0.1 dolaylarında kaldığı görülmüştür. Bu yüzden eğitimin tamamlanması için hedef olarak 0.1 kare hata oranı seçilmiş ve bunun sonucunda da 0.092 hata oranı ile 213 epok sonunda eğitim sona ermiştir. Ortaya çıkan sonuçlara bakıldığında 38 eğitim verisinin 32'sine ve 9 test verisinin 6'sına doğru cevap alındığı gözükmektedir.

### 5.3.6. Kendini düzenleyen ağlar (SOM) ile çözüm

Som ile yapılan denemelerde 47 verinin hepsi sisteme verilmiş ve bunların 4 sınıfa ayrılması istenmiştir. Denemeler çok sayıda farklı epok değerleri üzerinde yapılmasına karşın sürekli olarak verilerin yaklaşık yarısının yanlış sınıflandırıldığı görülmektedir. Bunun üzerine yapılan araştırmalarda Som'un başarılı olduğu veri kümeleri için yapılan tanımlama özelliklerin eşit derecede ölçeklendirilmesi ve eşit derecede öneme sahip olması yönündedir [4]. Elimizdeki verilerdeki 16 özelliğin birbiriyle çok farklı önem derecelerine sahip olduğunu söylemek mümkündür. Aynı zamanda değer kümeleri de farklılıklar göstermektedir. Örneğin bir belirtinin değeri sadece 0 ve 1 iken bir başkasının değeri 0-8 aralığında olabilmektedir. Sonuç olarak Som'un bu veri topluluğu üzerinde başarısız olduğunu söylemek mümkündür.

## 5.4. Karşılaştırma

Kullanılan yöntemlerden perseptron, geri yayılım ağları, yinelenen vektör nicemleme ağları ve kendini düzenleyen ağlar arasında en güzel sonucun geri yayılım ağları ile alındığı söylenebilir. Perseptronun test verilerine olumlu sonuç vermiş olmasına karşın eğitim verilerine verdiği sonuçlar ve hata oranı itibariyle bu yapıda bir veri kümesini sınıflandırmak için yetersiz kaldığını söylemek mümkündür. Bir diğer öğreticili yöntem olan lvq ağları da yeterince başarılı olmamasına rağmen sınıflandırma problemlerinde genel olarak başarılı olması ve perseptron gibi sadece doğrusal problemlerin çözümüne hitap etmemesi açısından daha büyük veri toplulukları üzerinde başarılı olabileceği düşünülmektedir.

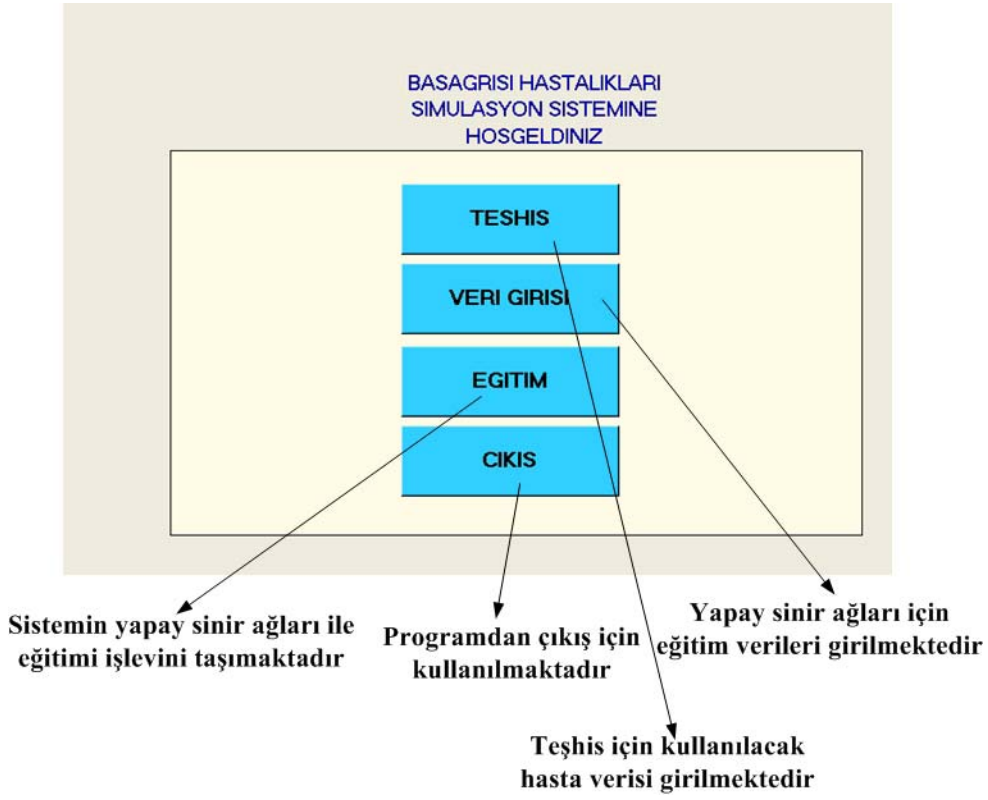
Kendini düzenleyen ağlar ise perseptron, geri yayılım ağları ve vektör nicemleme ağlarından farklı düşünülmelidir. Kendini düzenleyen ağın eğitimi öğreticisiz olurken diğerleri öğreticili eğitimden geçmektedirler. Kendini

düzenleyen ağlar, eğitim ve test verilerine verdiği sonuçlar itibariyle bu şekilde bir sınıflandırma problemini çözmekten son derece uzak gözükmektedir.

Eldeki örnekler daha kesin ve amaca yönelik olarak uzman bir doktor gözetiminde tekrar alınabilirse ve örneklerin sayısı arttırılabilirse bu sonuçların daha güzel olacağı açıktır.

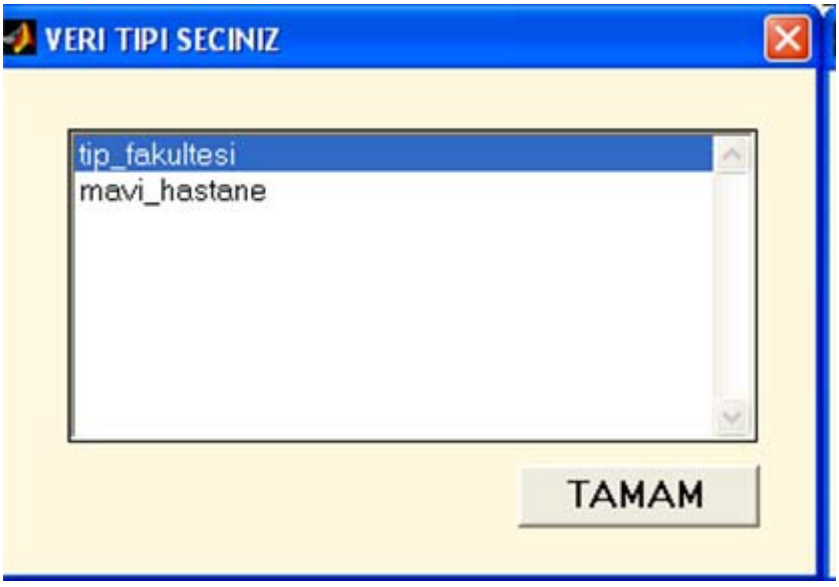
### 5.5. Grafik Arayüz Hakkında Bilgi

Tezin giriş kısmında bahsedildiği gibi teşhis problemine daha somut bir bakış açısı oluşturmak için grafik arayüze sahip bir yazılım da oluşturulmuştur. Programı ilk çalıştırdığımızda karşımıza aşağıdaki şekildeki bir menü çıkmaktadır.



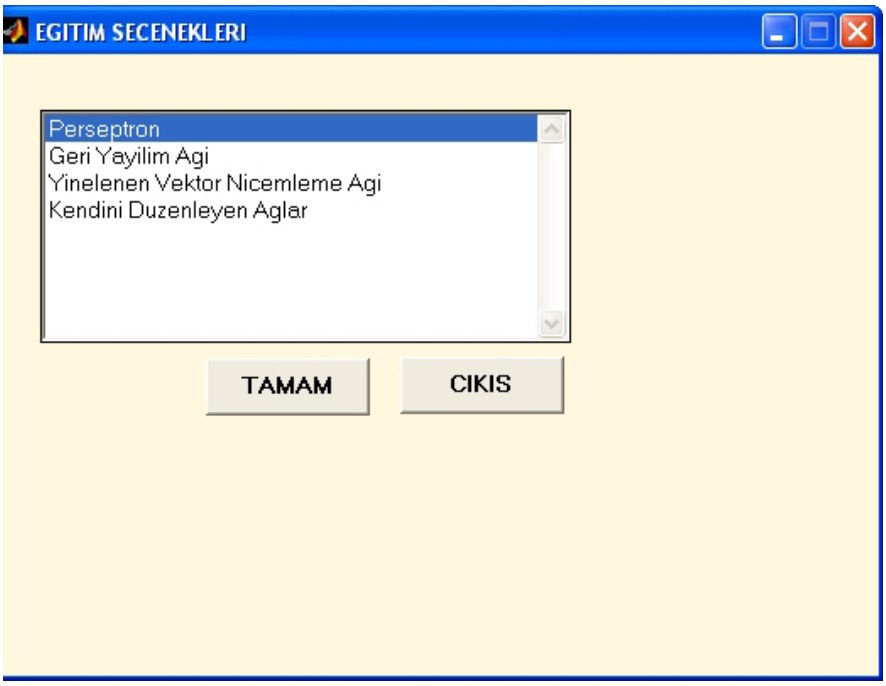

**Şekil 5.1 Hazırlanan yazılımın ana menüsü**

Sonraki adımda veri girişine tıklanmalı ve YSA'nın eğitim için kullanılacak veri kümesi girilmelidir. Bu verilerin girişi de önceden saklanan veri topluluğunun toplu halde girişi şeklindedir. Örnek olması açısından bu veri topluluklarına iki adet isimlendirme yapılmıştır. Bunlar da Şekil 5.2'de görülmektedir:



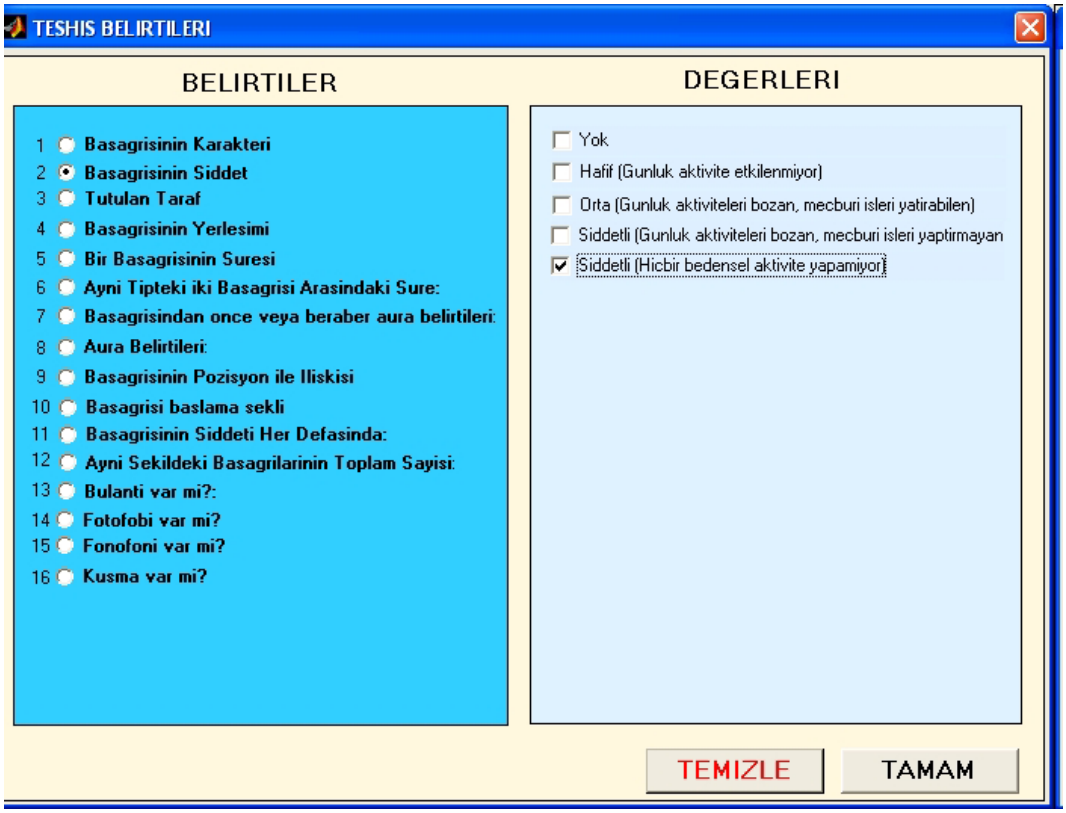

**Şekil 5.2 Veri tipi girişi formu**

Bir sonraki adımda ana menüdeki "eğitim" seçeneği kullanılmalı ve eğitimin yapılacağı YSA metotlarından biri seçilmelidir:



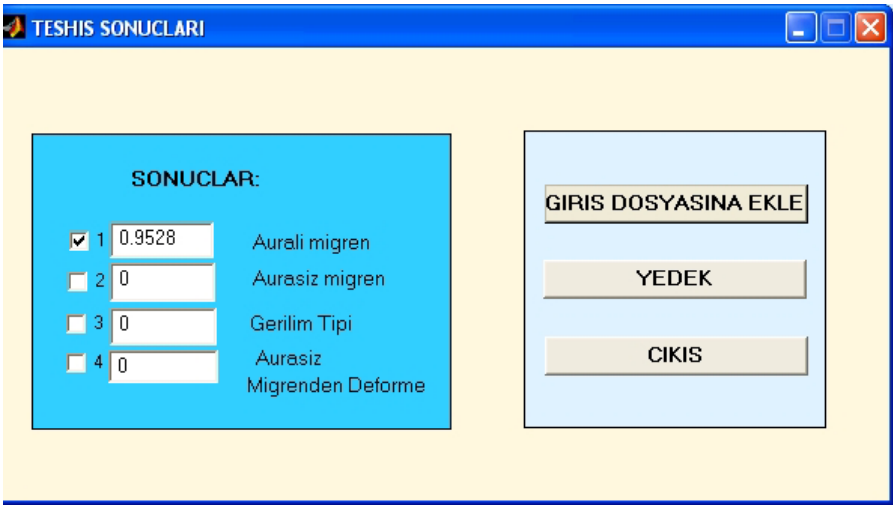

**Şekil 5.3 Eğitim metodu seçim formu**

Eğitim aşaması tamamlandıktan sonra ana menüdeki teşhis butonuna tıklanarak aşağıdaki form açılmalıdır ve buradan teşhis yapılacak hasta verileri girilmelidir.




**Şekil 5.4 Veri giriş formu**

Şekilde "TAMAM" butonuna tıklandığında sonuçlar Şekil 5.5'te görüldüğü gibi gelmektedir. "GİRİŞ DOSYASINA EKLE" butonu, alınan sonuçların da eğitim verileri arasına katılması anlamını taşımaktadır. "YEDEK" butonu ise veri kümesinin dosya şeklinde belli bir yerde saklanmasını sağlamaktadır.




**Şekil 5.5 Sonuç formuu**

# 6. SONUÇ VE ÖNERİLER

Bu tez çalışmasının sonunda ileriki araştırmalara öncülük edebilecek bir bilgi birikimi ortaya çıkarılmıştır. Yapay sinir ağları kullanılarak incelenecek tıbbi veri çeşidi olarak baş ağrısı hastalığı verileri seçilmiştir. Hastalık belirtileri indirgenirken mümkün olduğunca eldeki belirtiler topluluğu ve bunlara ait değer kümelerinin en geniş haliyle ifade edilmesine çalışılmıştır. Çalışmanın başlangıç safhasını oluşturan ve baş ağrısı hastalıklarının belirtilerini sistematik bir şekilde kodlayarak gösteren Ek-B, baş ağrısı hastalıkları üzerine ileride yapılabilecek istatistiksel çalışmalarda kullanılabilir.

Tıbbi veriler üzerinde 3 adet öğreticili, 1 adet öğreticisiz olmak üzere 4 farklı yapay sinir ağı yöntemi uygulanmıştır. Bu yöntemlerden biri olan geri yayılım ağının bu tip karmaşık veri toplulukları üzerinde başarılı olabileceği görülmektedir. Eğer çok sayıda örnek elde edilebilirse, sadece dört hastalık üstünde değil, daha fazla sayıda hastalık üzerinde geri yayılım ağı kullanılarak çalışmalar yapılabileceği düşünülmektedir.

Perseptron yönteminin, problemin karmaşıklık düzeyinin yüksekliği ve bunun sonucunda da verilerin doğrusal bir yapıda olmaması sonucu çok iyi sonuçlar vermediği görülmektedir.

Lvq yönteminin şu an için çok başarılı gözükmemesine karşın sınıflandırma problemlerindeki başarısı düşünüldüğünde eldeki veri sayısındaki artış sonrası daha verimli bir hale gelebileceği düşünülmektedir.

Kendini düzenleyen ağların öğreticisiz öğrenme yöntemi olması itibariyle diğerlerinden farklı yapıda olduğu bilinmektedir. Bu yöntemin böyle bir tipteki veri kümesi üzerinde başarısız sonuçlar verdiğini görmekteyiz. Eldeki veri topluluğunun yaklaşık yarısının yanlış sınıflandırılması buna bir kanıttır. Baş ağrısı hastalıklarında belirtilerin önem derecelerinin kendi aralarında oldukça farklı olduğunu problemin yapısı itibariyle söylemek mümkündür. Bunun da bu başarısızlığın ortaya çıkışındaki temel etmen olduğu düşünülmektedir. Bütün bunlardan yola çıkılarak bu tipteki ileriki çalışmalarda kendini düzenleyen ağların olumlu sonuç verme olasılığının düşük olduğu söylenebilir.

Bunun yanı sıra oluşturulan grafik arayüz de, gelecekte oluşturulabilecek bir uygulama yazılımının genel işleyişi açısından uzman doktorlara fikir vermektedir.

## KAYNAKLAR


[1]. A. PETROSIAN, D. PROKHOROV, W. LAJARA-NANSON, ve B. SCHIFFER, *Recurrent Neural Network based Approach for Early Recognition of Alzheimer's Disease in EEG*, Clinical Neurophysiology, **112/8**, ss. 1378-1387 (2001).

[2]. AXELSON, D., BAKKEN, I.J., GRIBBESTAD, I.S., EHRNHOLM,B., NILSEN , G., AASLY, J., *Applications of neural network analyses to in vivo $^{1}H$ magnetic resonance spectroscopy of Parkinson disease patients*, *Journal of Magnetic Resonance Imaging*, **16/1** , ss. 13 – 20, Wiley Press (2002).

[3]. DOĞAN, M., *Yapay Sinir Ağları Temelli Tıbbî Teşhis Sistemi*, Yüksek Lisans Tezi, Anadolu Üniversitesi, (2003).

[4]. MICHIE, D., SPIEGELHALTER, D.J. , TAYLOR, C.C., *Machine Learning, Neural and Statistical Classification*, ss. 1-3, Ellis Horwood Limited (1994).

[5]. ANDERSON, D., MCNEILL, G., *Artificial Neural Network Technology*, A DACS State-of-the-Art Report, Kaman Sciences Corporation (1992).

[6]. ÖZTEMEL, E., *Yapay Sinir Ağları*, Papatya Yayıncılık, İstanbul (2003).

[7]. ŞEN, Z., *Yapay Sinir Ağları İlkeleri*, Su Vakfı Yayınları, İstanbul (2004).

[8]. HAYKIN, S., *Neural Networks A Comprehensive Foundation*, Prentice Hall, New Jersey (1994).

[9]. HAM, F. ve KOSTANIC, I., *Principles of Neurocomputing for Science and Engineering*, Mc Graw Hill, Singapore (2001)

[10]. http://www.ent.com.tr/hastaliklar/basagrilari.htm

[11]. http://www.migran.org/mahdavi/Turkiska.doc

[12]. www.stacommunications.com/journals/pdfs/cme/CMEfebruary2003/ headache.pdf

[13]. DEMUTH, H. ve BEALE M., *Neural Network Toolbox For Use With MATLAB*, The Matworks Inc., MA, USA (2002).

[14]. BABANLI, A., *Kurala Dayalı Tıbbi Uzman Sistem*, 8. Türk Yapay Zeka ve Yapay Sinir Ağları Sempozyumu, 23-25 Haziran 1999, Boğaziçi Üniversitesi, İstanbul, Türkiye.

[15]. BABANLI, A., KARACA, H. ve GÜNER, Ö., *Teşhis Problemlerinde Belirtiler Uzayının Küçültülmesi*, A.Ü. Bilim ve Teknoloji Dergisi, **5/1**, ss. 177-182 (2004).

[16]. BABANLI, A. ve UYSAL, A.K., *Yapay Sinir Ağları Temelli Tıbbi Teşhis Problemi*, ELECO'2004, 8-12 Aralık 2004, Bursa.

[17]. BARAS, J.S. ve LAVIGNA, A., *Convergence of Kohonen's learning vector quantization*, International Joint Conference On Neural Networks, **3**, ss. 17-20, San Diego (1990).

[18]. BAXT, W., *The applications of the artificial neural network to clinical decision making*, Conference On Neural Information Processing Systems-Natural and Synthetic, Denver, 30 Kasım-3 Aralık (1992).

[19]. HARRISON, D., MARSHALL, S. ve KENNEDY, R., *the early diagnosis of heart attacks: A neurocomputational approach*, International Joint Conference On Neural Networks, 1, ss. 1-5, Seattle (1991).

[20]. HUANG, W. Y. ve LIPPMANN, R. P., *Comparisons between neural net and conventional classifiers*, Proceedings of the IEEE First International

Conference On Neural Networks, ss. 485–494, Piscataway, NJ. IEEE., (1987).

[21]. SILVA, F. M. ve ALMEIDA, L. B., *Acceleration techniques for the backpropagation algorithm*, Lecture Notes in Computer Science, **412**, ss. 110–119. Springer-Verlag, Berlin, (1990).

[22]. MINAI, A. A. ve WILLIAMS, R.D., *Acceleration of back-propagation through learning rate and momentum adaptation*, Proceedings of the International Joint Conference on Neural Networks, ss. 676-679, Washington (1990).

[23]. www.mathworks.com

## EK-A BAŞ AĞRISI HASTALIKLARI ve SINIFLANDIRILMALARI

**1. Migraine**

- 1.1. Migraine without aura
- 1.2. Migraine with aura
  - 1.2.1. Migraine with typical aura
  - 1.2.2. Migraine with prolonged aura
  - 1.2.3. Familial hemiplegic migraine
  - 1.2.4. Basiliar migraine
  - 1.2.5. Migraine aura without headache
  - 1.2.6. Migraine with acute onset aura
- 1.3. Ophthalmoplegic migraine
- 1.4. Retinal migraine
- 1.5. Childhood periodic syndromes that may be precursors to or associated with migraine
  - 1.5.1. Benign paroxysmal vertigo of childhood
  - 1.5.2. Alternating hemiplegia of childhood
- 1.6. Complications of migraine
  - 1.6.1. Status migrainosus
  - 1.6.2. Migrainous infarction
- 1.7. Migraionus disorder not fulfilling above criteria

**2. Tension-type headache**

- 2.1. Episodic tension-type headache
  - 2.1.1. Episodic tension-type headache associated with disorder of pericranial muscles
  - 2.1.2. Episodic tension-type headache unassociated with disorder of pericranial muscles
- 2.2. Chronic tension-type headache
  - 2.2.1. Chronic tension-type headache associated with disorder of pericranial muscles
  - 2.2.2. Chronic tension-type headache unassociated with disorder of pericranial muscles
- 2.3. Headache of the tension-type not fulfilling above criteria

**3. Cluster headache and chronic paroxysmal hemicrania**

- 3.1. Cluster headache
  - 3.1.1. Cluster headache periodicity undetermined
  - 3.1.2. Episodic cluster headache
  - 3.1.3. Chronic cluster headache
    - 3.1.3.1. Unremitting from onset
    - 3.1.3.2. Evolved from episodic
- 3.2. Chronic paroxysmal hemicrania
- 3.3. Cluster headache-like disorder not fulfilling above criteria

**4. Miscellaneous headaches unassociated with structural lesion**

- 4.1. Idiopathic stabbing headache
- 4.2. External compression headache
- 4.3. Cold stimulus headache
  - 4.3.1. External application of a cold stimulus
  - 4.3.2. Ingestion of a cold stimulus
- 4.4. Benign cough headache
- 4.5. Benign exertional headache
- 4.6. Headache associated with sexual activity
  - 4.6.1. Dull type
  - 4.6.2. Explosive type
  - 4.6.3. Postural type

**5. Headache associated with head trauma**

- 5.1. Acute post-traumatic headache
  - 5.1.1. With significant head trauma and/or confirmatory signs
  - 5.1.2. With minor head trauma and no confirmatory signs
- 5.2. Chronic post-traumatic headache
  - 5.2.1. With significant head trauma and/or confirmatory signs
  - 5.2.2. With minor head trauma and no confirmatory signs

**6. Headache associated with vascular disorders**

- 6.1. Acute ischemic cerebrovascular disease
  - 6.1.1. Transient ischemic attack (TIA)
  - 6.1.2. Thromboembolic stroke

6.2. Intracranial hematoma

6.2.1. Intracerebral hematoma

6.2.2. Subdural hematoma

6.2.3. Epidural hematoma

6.3. Subarachnoid hemorrhage

6.4. Unrupted vascular malformation

6.4.1. Arteriovenous malformation

6.4.2. Saccular aneurysm

6.5. Arteritis

6.5.1. Giant cell arteritis

6.5.2. Other systematic arterides

6.5.3. Primary intracranial arteritis

6.6. Carotid or certebral artery pain

6.6.1. Carotid or vertebral dissection

6.6.2. Carotidynia (idiopathic)

6.6.3. Post endarterectomy headache

6.7. Venous thrombosis

6.8. Arterial hypertension

6.8.1. Acute pressor response to exogenous agent

6.8.2. Pheochromocytoma

6.8.3. Malignant (accelerated) hypertension

6.8.4. Pre-eclampsia and eclampsia

6.9. Headache associated with other vascular disorder

**7. Headache associated with non-vascular intracranial disorder**

7.1. High cerebrospinal fluid pressure

7.1.1. Benign intracranial hypertension

7.1.2. High pressure hypocephalus

7.2. Low cerebrospinal fluid pressure

7.2.1. Post-lumbar puncture headache

7.2.2. Cerebrospinal fluid fistula headache

7.3. Intracranial infection

7.4. Intracranial sarcoidosis and other non-infectious inflammatory diseases

7.5. Headache related to intrathecal injections

7.5.1. Direct effect

7.5.2. Due to chemical meningitis

7.6. Intracranial neoplasm

7.7. Headache associated with other intracranial disorder

**8. Headache associated with substances or their withdrawal**

8.1. Headache induced by acute substance use or exposure

8.1.1. Nitrate/nitrite induced headache

8.1.2. Monosodium glutamate induced headache

8.1.3. Carbon monoxide induced headache

8.1.4. Alcohol induced headache

8.1.5. Other substances

8.2. Headache induced by chronic substance use or exposure

8.2.1. Ergotamine induced headache

8.2.2. Analgesics abuse headache

8.2.3. Other substances

8.3. Headache from substance withdrawal (acute use)

8.3.1. Alcohol withdrawal headache (hangover)

8.3.2. Other substances

8.4. Headache from substance withdrawal (chronic use)

8.4.1. Ergotamine withdrawal headache

8.4.2. Caffeine withdrawal headache

8.4.3. Narcotics abstinence headache

8.4.4. Other substances

8.5. Headache associated with substances but with uncertain mechanism

8.5.1. Birth control pills or estrogens

8.5.2. Other substances

**9. Headache associated with non-cephalic infection**

9.1. Viral infection

9.1.1. Focal non-cephalic

9.1.2. Systematic

9.2. Bacterial infection

9.2.1. Focal non-cephalic

9.2.2. Systematic (septicemia)

9.3. Headache related to other infection

**10. Headache associated with metabolic disorder**

10.1. Hypoxia

10.1.1. High altitude headache

10.1.2. Hypoxic headache

10.1.3. Sleep apnoea headache

10.2. Hypercapnia

10.3. Mixed hypxia and hypercapnia

10.4. Hypoglycemia

10.5. Dialysis

10.6. Headache related to other metabolic abnormality

**11. Headache or facial pain associated with disorder of cranium, neck, eyes, ears, nose, sinuses, teeth, mouth or other facial or cranial structures**

11.1. Cranial bone

11.2. Neck

11.2.1. Cervical spine

11.2.2. Retropharyngeal tendonitis

11.3. Eyes

11.3.1. Acute glaucoma

11.3.2. Refractive errors

11.3.3. Heterophoria or heterotraopia

11.4. Ears

11.5. Nose and sinuses

11.5.1. Acute sinus headache

11.5.2. Other diseases of nose or sinuses

11.6. Teeth, jaws and related structures

11.7. Temporomandibular joint disease

**12. Cranial neuralgias, nerve trunk pain and deafferentation pain**

12.1. Persistent (in contrast to tic-like) pain of cranial nerve origin

12.1.1. Compression or distortion of cranial nerves and second or third cervical roots

12.1.2. Demyelination of cranial nerves

12.1.2.1. Optic neuritis (retrobulbar neuritis)

12.1.3. Infarction of cranial nerves

12.1.3.1. Diabetic neuritis

12.1.4. Inflammation of cranial nerves

12.1.4.1. Herpes zoster

12.1.4.2. Chronic post-herpetic neuralgia

12.1.5. Tolosa-Hunt syndrome

12.1.6. Neck-tongue syndrome

12.1.7. Other causes of persistent pain of cranial nerve origin

12.2. Trigeminal neuralgia

12.2.1. Idiopathic trigeminal neuralgia

12.2.2. Symptomatic trigeminal neuralgia

12.2.2.1. Compression of trigeminal root or ganglion

12.2.2.2. Central lesions

12.3. Glossopharyngeal neuralgia

12.3.1. Idiopathic glossopharyngeal neuralgia

12.3.2. Symptomatic glossophayngeal neuralgia

12.4. Nervus intermedius neuralgia

12.5. Superior leryngeal neuralgia

12.6. Occipital neuralgia

12.7. Central causes of head and facial pain other than tic douloureux

12.7.1. Anaesthesia dolorosa

12.7.2. Thalamic pain

12.8. Facial pain not fulfilling criteria in groups 11 or 12

**13. Headache not classifiable**

## EK-B BAŞAĞRISI HASTALIKLARININ BELİRTİLERİ

# 1 Şikayetler

**1.01 Başağrısının Karakteri:**

1.01.0 Yok

1.01.1 Zonklayıcı (Damar atışı gibi, davul gibi)

1.01.2 Patlayıcı (Çok şiddetli ve ani, yakıcı)

1.01.3 Sersemletici (Başta sersemlik hissi gibi, sarhoşvari, yorgun)

1.01.4 Şimşek gibi (Elektrik çarpması gibi, bıçak saplanır tarzı)

1.01.5 Baskı hissi (Sürekli sıkıştırılıyormuş gibi)

1.01.6 Yakıcı

**1.02 Başağrısının Şiddeti:**

1.02.1 Hafif (Günlük aktivite etkilenmiyor)

1.02.2 Orta (Günlük aktiviteleri bozan, mecburi işleri yatırabilen)

1.02.3 Şiddetli (Günlük aktiviteleri bozan, mecburi işleri yaptırmayan)

1.02.4 Şiddetli (Hiçbir bedensel aktivite yapamıyor)

**1.03 Tutulan Taraf**

1.03.01 Tek taraflı

1.03.02 İki taraflı

**1.04 Bağağrısının Yerleşimi**

1.04.01 Yüzün üst bölümü

1.04.02 Başın arkasında

1.04.03 Yüzün ön ve iç kısmı

1.04.04 Başın Tamamında

**1.05 Bir Başağrısının Süresi: .....**

1.05.1 Saniye

1.05.2 Dakika

1.05.3 Saat

1.05.4 Gün

1.05.5 Hafta

1.05.6 Ay

1.05.7 Yıl

**1.06 Aynı Tipteki İki Başağrısı Arasındaki Süre:**

1.06.1 Uyku hariç 4 saatten kısa

1.06.2 Uyku hariç 4 saatten uzun

1.06.3 14 günden az

1.06.4 14 günden fazla

1.06.5 Ortalama 8 hafta

1.06.6 Düzensiz

1.06.7 24 saat

1.06.8 1 yıl

**1.07 Başağrısından Önce veya Beraber (Aura Belirtileri): (R/L/B/T) [(R)ight, (L)eft, (B)ilateral, (T)araf değiştiren]**

1.07.1 Yok

1.07.2 Hemisferik

1.07.3 Beyin sapı

1.07.4 Kranial sinir

**1.08 Aura Belirtileri**

1.08.1 Birkaç dakikada gelişiyor

1.08.2 5 dakikadan uzun sürede gelişiyor

1.08.3 1 saatten uzun sürmüyor

1.08.4 1 haftadan kısa sürüyor

1.08.5 Tamamen düzeliyor

1.08.6 Tamamen düzelmiyor

1.08.7 Birlikte veya sonra başağrısı yok

**1.09 Başağrısının Ortaya Çıkış Zamanı**

1.09.1 Belli değil

1.09.2 Uyandıktan sonra

1.09.3 Gün içinde uyanıkken

1.09.4 Gece uyurken

**1.10 Başağrısının Pozisyon İle İlişkisi**

1.10.1 Hareket ederken fazla

1.10.2 Otururken fazla

1.10.3 Yatarken fazla

1.10.4 Değişmiyor

1.10.5 Ayaktayken fazla (ayağa kalktıktan sonra 15 dk. içinde)

1.10.6 Yattıktan sonra 30 dk içinde düzeliyor

**1.11 Başağrısı başlama şekli**

1.11.1 Ani

1.11.2 Yavaş yavaş

1.11.3 Bazen ani, bazen yavaş

**1.12 Başağrısının Şiddeti Her Defasında**

1.12.1 Aynı

1.12.2 Azalıyor

1.12.3 Artıyor

1.12.4 Değişken

**1.13 Prodromal**

1.13.01 Duyarlılığın artması

1.13.02 Duyarlılığın azalması

1.13.03 Dikkat azalması

1.13.04 Düşünce yavaşlığı

**1.14 Postdrom**

1.14.1 Halsizlik

1.14.2 Başında ağırlık hissi

1.14.3 Işıktan rahatsız olma

1.14.4 Acıkma

1.14.5 Tatlı yeme isteği

1.14.6 Sık idrara çıkma

**1.15 Başağrısı:**

1.15.01 Uyku ile düzeliyor

1.15.02 2 hafta içinde spontan düzeliyor

1.15.03 Tekoin ile düzeliyor

**1.16 Aynı şekildeki başağrılarının sayısı: .....**

1.16.1 Bir günde

1.16.2 Bir haftada

1.16.3 Bir ayda

1.16.4 Bir yılda

**1.17 Aynı şekildeki Başağrılarının Toplam Sayısı: .....**

**1.18 Ailede aynı tipte başağrısı:**

1.18.1 Annede

1.18.2 Babada

1.18.3 Kardeşlerde

## 2 Bulgular

**2.1 Ataklar arasında nörolojik muayene:**

2.1.1 Normal

2.1.2 Anormal

**2.2 Başağrısı İle Beraber**

2.2.01 Ozmofobi

2.2.02 Bulantı

2.2.03 Kusma

2.2.04 Fotofobi

2.2.05 Fonofobi

2.2.06 Otonomik

2.2.07 Kranial sinir paralizisi

2.2.08 Ense Sertliği

2.2.09 Ateş

2.2.10 Şis ve hassas kafa derisi arteri

2.2.11 Çenede Ağrı

2.2.12 Yaygın kas ağrıları

2.2.13 Epileptik nöbetler

2.2.14 Anskiyete

2.2.15 Göz dibi bulguları

**3 Laboratuvar Bulguları**

**3.01 Laboratuvar**

3.01.00 Yapılmadı

3.01.01 Normal

3.01.02 Anormal

**3.02 Kafa Grafisi:**

3.02.0 Yapılmadı

3.02.1 Normal

3.02.2 Anormal

**3.03 Servikal Grafi**

3.03.0 Yapılmadı

3.03.1 Normal

3.03.2 Anormal

**3.04 Evoked Potansiyeller:**

3.04.0 Yapılmadı

3.04.1 Normal

3.04.2 Anormal

**3.05 BOS Tetkiki:**

3.05.0 Yapılmadı

3.05.1 Normal

3.05.2 Anormal

3.05.3 Kültür (-)

**3.06 Vestibüler Fonksiyon Testleri**

3.06.0 Yapılmadı

3.06.1 Normal

3.06.2 Anormal

**3.07 Nörofizyolojik Testler**

3.07.0 Yapılmadı

3.07.1 Normal

3.07.2 Anormal

**3.08 EEG:**

3.08.0 Yapılmadı

3.08.1 Normal

3.08.2 Anormal

**3.09 Doppler Ultrasonografi:**

3.09.0 Yapılmadı

3.09.1 Normal

3.09.2 Anormal

**3.10 TCD:**

3.10.0 Yapılmadı

3.10.1 Normal

3.10.2 Anormal

**3.11 BT:**

3.11.00 Yapılmadı

3.11.01 Normal

3.11.02 Anormal

**3.12 MRI:**

3.12.00 Yapılmadı

3.12.01 Normal

3.12.02 Anormal

**3.13 Anjiografi:**

3.13.00 Yapılmadı

3.13.01 Normal

3.13.02 Anormal

**3.14 EMG**

3.14.0 Yapılmadı

3.14.1 Normal

3.14.2 Anormal

**3.15 MR anjiografi**

3.15.0 Yapılmadı

3.15.1 Normal

3.15.2 Anormal

**3.16 Biopsi**

3.16.0 Yapılmadı

3.16.1 Normal

3.16.2 Anormal

**3.17 Water's grafi**

3.17.0 Yapılmadı

3.17.1 Normal

3.17.2 Anormal

**3.18 Nörolojik Muayene**

3.18.0 Yapılmadı

3.18.1 Normal

3.18.2 Anormal

## EK-C MATLAB PROGRAM KODLARI

```
% PERSEPTRON İLE HASTALIKLARIN SINIFLANDIRILMASI.
%egitim icin kullanilan veri kumesi
noro_train_data=[0 1 2 1 4 0 0 0 4 3 3 1 0 0 0 0
                 1 3 1 1 3 0 0 0 1 3 1 1 1 0 1 1
                 1 2 2 0 2 0 0 0 4 3 1 8 1 0 1 1
                 1 3 1 0 3 1 0 0 4 1 1 5 1 0 0 0
                 1 4 2 0 2 0 0 0 1 1 1 8 1 0 1 1
                 1 3 1 0 2 0 0 0 1 1 1 7 1 1 1 0
                 1 2 2 0 2 1 0 0 4 2 1 1 1 0 1 1
                 0 3 1 0 1 1 0 0 1 2 1 2 1 0 0 0
                 1 3 2 0 4 0 0 0 4 1 1 7 1 0 0 1
                 1 2 1 0 2 1 0 0 2 2 1 7 1 0 1 1
                 1 2 1 1 3 1 0 0 1 1 1 3 1 0 0 1
                 1 3 1 1 1 0 0 0 3 1 4 3 0 0 0 1
                 1 4 1 0 3 1 0 0 4 1 1 6 1 0 1 1
                 1 3 1 0 1 1 0 0 4 1 1 2 1 1 1 1
                 1 3 1 0 3 0 0 0 4 2 1 3 1 1 1 1
                 1 3 1 0 2 0 1 0 2 3 3 4 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 6 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 3 0 0 1 0
                 1 3 2 0 2 0 0 0 4 1 1 2 1 1 1 1
                 0 2 2 0 1 0 0 0 1 2 1 1 0 0 0 0
                 1 3 1 1 2 0 1 1 1 3 1 5 1 0 1 1
                 1 3 1 1 3 1 1 1 4 1 1 4 1 0 1 1
                 1 2 1 0 2 0 0 0 1 3 1 2 1 0 1 1
                 1 3 1 0 2 0 1 1 1 3 1 3 1 0 1 1
                 1 3 2 0 2 1 0 0 4 1 3 4 1 0 0 1
                 0 2 2 1 2 0 0 0 1 1 1 1 0 0 0 1
                 0 2 2 0 3 0 0 0 4 1 1 8 1 0 0 1
                 1 3 2 1 3 0 0 0 1 1 1 8 1 1 1 1
                 1 4 2 0 1 1 0 1 1 1 1 1 1 1 1 1
                 1 1 1 1 3 0 0 0 1 3 4 8 1 1 1 1
                 1 2 2 1 2 1 0 0 1 3 1 4 1 0 0 1
                 1 2 2 1 3 0 0 0 4 3 1 2 0 0 0 0
                 0 2 2 1 1 0 0 0 1 3 1 8 0 0 0 1
                 1 1 1 1 2 0 0 0 1 3 1 2 1 0 0 1
                 0 2 2 1 3 1 0 0 1 1 1 3 1 0 0 0
```

```
                0 2 2 0 1 1 0 0 2 1 1 1 1 1 0 0
                1 3 1 0 1 1 1 0 4 3 3 2 0 0 1 1
                1 3 2 0 1 0 0 0 4 1 1 3 1 1 1 1
            ];
%test icin kullanilan veri kumesi
 noro_test_data=[1 3 1 0 3 1 0 0 4 3 1 2 1 1 1 1
                1 2 1 0 2 1 0 0 4 2 3 4 1 1 1 1
                0 2 0 1 2 0 0 0 4 2 1 2 1 0 0 0
                1 3 1 0 2 0 1 1 4 3 1 6 1 0 1 1
                1 3 2 0 1 1 1 1 2 1 1 1 1 0 1 1
                0 2 1 0 1 0 0 0 4 3 1 3 0 0 0 0
                0 2 1 0 1 0 0 0 1 1 1 4 0 0 0 0
                1 4 1 0 1 0 0 0 1 1 3 6 1 0 1 1
                1 1 1 0 1 0 0 0 1 3 3 8 1 0 1 1
            ];
%hastaliklar asagidaki gibi ifade edilmektedir
    % 00 aurasiz migren
    % 01 aurali migren
    % 10 gerilim tipi
    % 11 aurasiz migrenden deforme

 %egitim verileri sonuclari
 noro_train_rs=[0 0;
               0 0;
               1 1;
               0 0;
               1 1;
               1 0;
               0 0;
               1 0;
               1 0;
               1 1;
               1 0;
               1 0;
               0 0;
               0 0;
               1 0;
               1 1;
               1 1;
```

```
                0 0;
                0 0;
                1 0;
                0 1;
                0 1;
                0 0;
                0 1;
                0 0;
                1 0;
                0 0;
                1 1;
                0 1;
                1 1;
                0 0;
                1 0;
                1 0;
                0 0;
                0 0;
                0 0;
                0 0;
                1 0
            ];
%test verileri sonuclari
noro_test_rs=[0 0;
              0 0;
              0 0;
              0 1;
              0 1;
              1 0;
              1 0;
              1 1;
              1 1
          ];

T=noro_train_rs';
ttest=noro_test_rs';
P=noro_train_data';
ptest=noro_test_data';
%perseptron agi olusturuldu
```

```
net=newp(minmax(P),2);
net.trainParam.epochs=100;
net=init(net);
net=train(net,P,T);
sonuc_egitim=sim(net,P);
sonuc_test=sim(net,ptest);

dogru_egitim=0;
dogru_test=0;

%egitim verileri icin dogru yanit sayilari kontrol edilecek
for i = 1:38,
    if (sonuc_egitim(1,i)==T(1,i)) & (sonuc_egitim(2,i)==T(2,i))
          dogru_egitim=dogru_egitim+1;
    end
end

%test verileri icin dogru yanit sayilari kontrol edilecek
for i = 1:9,
  if (sonuc_test(1,i)==ttest(1,i)) & (sonuc_test(2,i)==ttest(2,i))
           dogru_test=dogru_test+1;
     end
end
```

**% GERİ YAYILIM AĞI İLE HASTALIKLARIN SINIFLANDIRILMASI.**

```
%egitim icin kullanilan veri kumesi
noro_train_data=[0 1 2 1 4 0 0 0 4 3 3 1 0 0 0 0
                 1 3 1 1 3 0 0 0 1 3 1 1 1 0 1 1
                 1 2 2 0 2 0 0 0 4 3 1 8 1 0 1 1
                 1 3 1 0 3 1 0 0 4 1 1 5 1 0 0 0
                 1 4 2 0 2 0 0 0 1 1 1 8 1 0 1 1
                 1 3 1 0 2 0 0 0 1 1 1 7 1 1 1 0
                 1 2 2 0 2 1 0 0 4 2 1 1 1 0 1 1
                 0 3 1 0 1 1 0 0 1 2 1 2 1 0 0 0
                 1 3 2 0 4 0 0 0 4 1 1 7 1 0 0 1
                 1 2 1 0 2 1 0 0 2 2 1 7 1 0 1 1
                 1 2 1 1 3 1 0 0 1 1 1 3 1 0 0 1
                 1 3 1 1 1 0 0 0 3 1 4 3 0 0 0 1
                 1 4 1 0 3 1 0 0 4 1 1 6 1 0 1 1
```

```
                1 3 1 0 1 1 0 0 4 1 1 2 1 1 1 1
                1 3 1 0 3 0 0 0 4 2 1 3 1 1 1 1
                1 3 1 0 2 0 1 0 2 3 3 4 1 0 1 1
                1 3 2 0 1 0 0 0 4 2 1 6 1 0 1 1
                1 3 2 0 1 0 0 0 4 2 1 3 0 0 1 0
                1 3 2 0 2 0 0 0 4 1 1 2 1 1 1 1
                0 2 2 0 1 0 0 0 1 2 1 1 0 0 0 0
                1 3 1 1 2 0 1 1 1 3 1 5 1 0 1 1
                1 3 1 1 3 1 1 1 4 1 1 4 1 0 1 1
                1 2 1 0 2 0 0 0 1 3 1 2 1 0 1 1
                1 3 1 0 2 0 1 1 1 3 1 3 1 0 1 1
                1 3 2 0 2 1 0 0 4 1 3 4 1 0 0 1
                0 2 2 1 2 0 0 0 1 1 1 1 0 0 0 1
                0 2 2 0 3 0 0 0 4 1 1 8 1 0 0 1
                1 3 2 1 3 0 0 0 1 1 1 8 1 1 1 1
                1 4 2 0 1 1 0 1 1 1 1 1 1 1 1 1
                1 1 1 1 3 0 0 0 1 3 4 8 1 1 1 1
                1 2 2 1 2 1 0 0 1 3 1 4 1 0 0 1
                1 2 2 1 3 0 0 0 4 3 1 2 0 0 0 0
                0 2 2 1 1 0 0 0 1 3 1 8 0 0 0 1
                1 1 1 1 2 0 0 0 1 3 1 2 1 0 0 1
                0 2 2 1 3 1 0 0 1 1 1 3 1 0 0 0
                0 2 2 0 1 1 0 0 2 1 1 1 1 1 0 0
                1 3 1 0 1 1 1 0 4 3 3 2 0 0 1 1
                1 3 2 0 1 0 0 0 4 1 1 3 1 1 1 1
            ];
 %test icin kullanilan veri kumesi
 noro_test_data=[1 3 1 0 3 1 0 0 4 3 1 2 1 1 1 1
                1 2 1 0 2 1 0 0 4 2 3 4 1 1 1 1
                0 2 0 1 2 0 0 0 4 2 1 2 1 0 0 0
                1 3 1 0 2 0 1 1 4 3 1 6 1 0 1 1
                1 3 2 0 1 1 1 1 2 1 1 1 1 0 1 1
                0 2 1 0 1 0 0 0 4 3 1 3 0 0 0 0
                0 2 1 0 1 0 0 0 1 1 1 4 0 0 0 0
                1 4 1 0 1 0 0 0 1 1 3 6 1 0 1 1
                1 1 1 0 1 0 0 0 1 3 3 8 1 0 1 1
            ];
%hastaliklar asagidaki gibi ifade edilmektedir
    % 1000 aurasiz migren
```

```
    % 0100 aurali migren
    % 0010 gerilim tipi
    % 0001 aurasiz migrenden deforme

%egitim verileri sonuclari
noro_train_rs=[ 1 0 0 0
                1 0 0 0
                0 0 0 1
                1 0 0 0
                0 0 0 1
                0 0 1 0
                1 0 0 0
                0 0 1 0
                0 0 1 0
                0 0 0 1
                0 0 1 0
                0 0 1 0
                1 0 0 0
                1 0 0 0
                0 0 1 0
                0 0 0 1
                0 0 0 1
                1 0 0 0
                1 0 0 0
                0 0 1 0
                0 1 0 0
                0 1 0 0
                1 0 0 0
                0 1 0 0
                1 0 0 0
                0 0 1 0
                1 0 0 0
                0 0 0 1
                0 1 0 0
                0 0 0 1
                1 0 0 0
                0 0 1 0
                0 0 1 0
                1 0 0 0
```

```
                1 0 0 0
                1 0 0 0
                1 0 0 0
                0 0 1 0
            ];
%test verileri sonuclari
noro_test_rs=[1 0 0 0
              1 0 0 0
              1 0 0 0
              0 1 0 0
              0 1 0 0
              0 0 1 0
              0 0 1 0
              0 0 0 1
              0 0 0 1
          ];

T=noro_train_rs';

P=noro_train_data';
%normalizasyon islemi yapiliyor
[pn,meanp,stdp,tn,meant,stdt] = prestd(P,T);
net=newff(minmax(P),[15 15 4],{'tansig' 'tansig'
'purelin'},'traingd');
net=init(net);
net.trainParam.epochs = 1000;
net=train(net,pn,tn);

[p2n] = trastd(noro_test_data',meanp,stdp);
sonuc_test=sim(net,p2n);
[sonuc_test] = poststd(sonuc_test,meant,stdt);

max_sonuc_test=max(sonuc_test);

[pn] = trastd(P,meanp,stdp);
sonuc_egitim=sim(net,pn);
[sonuc_egitim]=poststd(sonuc_egitim,meant,stdt);

max_sonuc_egitim=max(sonuc_egitim);
```

```
%test verilerine verilen sonuclarda 1'e en yakin
%deger 1'e digerleri 0'a yuvarlandi
for i = 1:9,
        for j=1:4,
            if sonuc_test(j,i)==max_sonuc_test(i)
                sonuc_test(j,i)=1;
            else
                sonuc_test(j,i)=0;
            end
        end
end

%egitim verilerine verilen sonuclarda 1'e en yakin
%deger 1'e digerleri 0'a yuvarlandi
for i = 1:38,
        for j=1:4,
            if sonuc_egitim(j,i)==max_sonuc_egitim(i)
                sonuc_egitim(j,i)=1;
            else
                sonuc_egitim(j,i)=0;
            end
        end
end

noro_train_rs=noro_train_rs';
noro_test_rs=noro_test_rs';

dogru_test=0;
dogru_train=0;

%test verilerine verilen dogru cevap sayisi bulundu
for i = 1:9,
        for j=1:4,
            if sonuc_test(j,i)==noro_test_rs(j,i)
               if (sonuc_test(j,i)==1)
                   dogru_test=dogru_test+1;
               end
            end
```

```
        end
end

%egitim verilerine verilen dogru cevap sayisi bulundu
for i = 1:38,
        for j=1:4,
            if sonuc_egitim(j,i)==noro_train_rs(j,i)
                if (sonuc_egitim(j,i)==1)
                   dogru_train=dogru_train+1;
                end
            end
        end
end
```

**% LVQ İLE HASTALIKLARIN SINIFLANDIRILMASI.**

```
%egitim icin kullanilan veri kumesi
noro_train_data=[0 1 2 1 4 0 0 0 4 3 3 1 0 0 0 0
                 1 3 1 1 3 0 0 0 1 3 1 1 1 0 1 1
                 1 2 2 0 2 0 0 0 4 3 1 8 1 0 1 1
                 1 3 1 0 3 1 0 0 4 1 1 5 1 0 0 0
                 1 4 2 0 2 0 0 0 1 1 1 8 1 0 1 1
                 1 3 1 0 2 0 0 0 1 1 1 7 1 1 1 0
                 1 2 2 0 2 1 0 0 4 2 1 1 1 0 1 1
                 0 3 1 0 1 1 0 0 1 2 1 2 1 0 0 0
                 1 3 2 0 4 0 0 0 4 1 1 7 1 0 0 1
                 1 2 1 0 2 1 0 0 2 2 1 7 1 0 1 1
                 1 2 1 1 3 1 0 0 1 1 1 3 1 0 0 1
                 1 3 1 1 1 0 0 0 3 1 4 3 0 0 0 1
                 1 4 1 0 3 1 0 0 4 1 1 6 1 0 1 1
                 1 3 1 0 1 1 0 0 4 1 1 2 1 1 1 1
                 1 3 1 0 3 0 0 0 4 2 1 3 1 1 1 1
                 1 3 1 0 2 0 1 0 2 3 3 4 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 6 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 3 0 0 1 0
                 1 3 2 0 2 0 0 0 4 1 1 2 1 1 1 1
                 0 2 2 0 1 0 0 0 1 2 1 1 0 0 0 0
                 1 3 1 1 2 0 1 1 1 3 1 5 1 0 1 1
                 1 3 1 1 3 1 1 1 4 1 1 4 1 0 1 1
                 1 2 1 0 2 0 0 0 1 3 1 2 1 0 1 1
```

```
                1 3 1 0 2 0 1 1 1 3 1 3 1 0 1 1
                1 3 2 0 2 1 0 0 4 1 3 4 1 0 0 1
                0 2 2 1 2 0 0 0 1 1 1 1 0 0 0 1
                0 2 2 0 3 0 0 0 4 1 1 8 1 0 0 1
                1 3 2 1 3 0 0 0 1 1 1 8 1 1 1 1
                1 4 2 0 1 1 0 1 1 1 1 1 1 1 1 1
                1 1 1 1 3 0 0 0 1 3 4 8 1 1 1 1
                1 2 2 1 2 1 0 0 1 3 1 4 1 0 0 1
                1 2 2 1 3 0 0 0 4 3 1 2 0 0 0 0
                0 2 2 1 1 0 0 0 1 3 1 8 0 0 0 1
                1 1 1 1 2 0 0 0 1 3 1 2 1 0 0 1
                0 2 2 1 3 1 0 0 1 1 1 3 1 0 0 0
                0 2 2 0 1 1 0 0 2 1 1 1 1 1 0 0
                1 3 1 0 1 1 1 0 4 3 3 2 0 0 1 1
                1 3 2 0 1 0 0 0 4 1 1 3 1 1 1 1
            ];
%test icin kullanilan veri kumesi
 noro_test_data=[1 3 1 0 3 1 0 0 4 3 1 2 1 1 1 1
                1 2 1 0 2 1 0 0 4 2 3 4 1 1 1 1
                0 2 0 1 2 0 0 0 4 2 1 2 1 0 0 0
                1 3 1 0 2 0 1 1 4 3 1 6 1 0 1 1
                1 3 2 0 1 1 1 1 2 1 1 1 1 0 1 1
                0 2 1 0 1 0 0 0 4 3 1 3 0 0 0 0
                0 2 1 0 1 0 0 0 1 1 1 4 0 0 0 0
                1 4 1 0 1 0 0 0 1 1 3 6 1 0 1 1
                1 1 1 0 1 0 0 0 1 3 3 8 1 0 1 1
            ];
%hastaliklar asagidaki gibi ifade edilmektedir
     %1 Aurasiz migren
     %2 Aurali migren
     %3 Gerilim tipi
     %4 Aurasiz migrenden deforme
%egitim verileri sonuclari
noro_train_rs=[ 1
                1
                4
                1
                4
                3
```

```
                1
                3
                3
                4
                3
                3
                1
                1
                3
                4
                4
                1
                1
                3
                2
                2
                1
                2
                1
                3
                1
                4
                2
                4
                1
                3
                3
                1
                1
                1
                1
                3
            ];
%test verileri sonuclari
noro_test_rs=[1
              1
              1
              2
              2
```

```
            3
            3
            4
            4
        ];

T=noro_train_rs';
P=noro_train_data';
T=ind2vec(T);
%lvq agi olusturuldu
net=newlvq(minmax(P),20,[.42 .12 .28 .18]);
net=init(net);

net.trainParam.epochs = 500;
net.trainParam.goal=0.1;
net=train(net,P,T);

sonuc_test=sim(net,noro_test_data')
sonuc_test=vec2ind(sonuc_test);

sonuc_egitim=sim(net,P)
sonuc_egitim=vec2ind(sonuc_egitim);

ed=noro_train_rs';
td=noro_test_rs';

dogru_egitim=0;
dogru_test=0;
%egitim verilerindeki dogru sayisi belirlendi
for i = 1:38,
            if sonuc_egitim(i)==ed(i)
                dogru_egitim=dogru_egitim+1;
            end
end
%test verilerindeki dogru sayisi belirlendi
for i = 1:9,
            if sonuc_test(i)==td(i)
                dogru_test=dogru_test+1;
            end
```

```
end

% SOM İLE HASTALIKLARIN SINIFLANDIRILMASI.
%egitim icin kullanilan veri kumesi
noro_train_data=[0 1 2 1 4 0 0 0 4 3 3 1 0 0 0 0
                 1 3 1 1 3 0 0 0 1 3 1 1 1 0 1 1
                 1 2 2 0 2 0 0 0 4 3 1 8 1 0 1 1
                 1 3 1 0 3 1 0 0 4 1 1 5 1 0 0 0
                 1 4 2 0 2 0 0 0 1 1 1 8 1 0 1 1
                 1 3 1 0 2 0 0 0 1 1 1 7 1 1 1 0
                 1 2 2 0 2 1 0 0 4 2 1 1 1 0 1 1
                 0 3 1 0 1 1 0 0 1 2 1 2 1 0 0 0
                 1 3 2 0 4 0 0 0 4 1 1 7 1 0 0 1
                 1 2 1 0 2 1 0 0 2 2 1 7 1 0 1 1
                 1 2 1 1 3 1 0 0 1 1 1 3 1 0 0 1
                 1 3 1 1 1 0 0 0 3 1 4 3 0 0 0 1
                 1 4 1 0 3 1 0 0 4 1 1 6 1 0 1 1
                 1 3 1 0 1 1 0 0 4 1 1 2 1 1 1 1
                 1 3 1 0 3 0 0 0 4 2 1 3 1 1 1 1
                 1 3 1 0 2 0 1 0 2 3 3 4 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 6 1 0 1 1
                 1 3 2 0 1 0 0 0 4 2 1 3 0 0 1 0
                 1 3 2 0 2 0 0 0 4 1 1 2 1 1 1 1
                 0 2 2 0 1 0 0 0 1 2 1 1 0 0 0 0
                 1 3 1 1 2 0 1 1 1 3 1 5 1 0 1 1
                 1 3 1 1 3 1 1 1 4 1 1 4 1 0 1 1
                 1 2 1 0 2 0 0 0 1 3 1 2 1 0 1 1
                 1 3 1 0 2 0 1 1 1 3 1 3 1 0 1 1
                 1 3 2 0 2 1 0 0 4 1 3 4 1 0 0 1
                 0 2 2 1 2 0 0 0 1 1 1 1 0 0 0 1
                 0 2 2 0 3 0 0 0 4 1 1 8 1 0 0 1
                 1 3 2 1 3 0 0 0 1 1 1 8 1 1 1 1
                 1 4 2 0 1 1 0 1 1 1 1 1 1 1 1 1
                 1 1 1 1 3 0 0 0 1 3 4 8 1 1 1 1
                 1 2 2 1 2 1 0 0 1 3 1 4 1 0 0 1
                 1 2 2 1 3 0 0 0 4 3 1 2 0 0 0 0
                 0 2 2 1 1 0 0 0 1 3 1 8 0 0 0 1
                 1 1 1 1 2 0 0 0 1 3 1 2 1 0 0 1
                 0 2 2 1 3 1 0 0 1 1 1 3 1 0 0 0
```

```
                0 2 2 0 1 1 0 0 2 1 1 1 1 1 0 0
                1 3 1 0 1 1 1 0 4 3 3 2 0 0 1 1
                1 3 2 0 1 0 0 0 4 1 1 3 1 1 1 1
            ];
%test icin kullanilan veri kumesi
 noro_test_data=[1 3 1 0 3 1 0 0 4 3 1 2 1 1 1 1
                1 2 1 0 2 1 0 0 4 2 3 4 1 1 1 1
                0 2 0 1 2 0 0 0 4 2 1 2 1 0 0 0
                1 3 1 0 2 0 1 1 4 3 1 6 1 0 1 1
                1 3 2 0 1 1 1 1 2 1 1 1 1 0 1 1
                0 2 1 0 1 0 0 0 4 3 1 3 0 0 0 0
                0 2 1 0 1 0 0 0 1 1 1 4 0 0 0 0
                1 4 1 0 1 0 0 0 1 1 3 6 1 0 1 1
                1 1 1 0 1 0 0 0 1 3 3 8 1 0 1 1
            ];
%hastaliklar asagidaki gibi ifade edilmektedir
     %1 Aurasiz migren
     %2 Aurali migren
     %3 Gerilim tipi
     %4 Aurasiz migrenden deforme
%egitim verileri sonuclari
noro_train_rs=[ 1
                1
                4
                1
                4
                3
                1
                3
                3
                4
                3
                3
                1
                1
                3
                4
                4
                1
```

```
                1
                3
                2
                2
                1
                2
                1
                3
                1
                4
                2
                4
                1
                3
                3
                1
                1
                1
                1
                3
            ];
%test verileri sonuclari
noro_test_rs=[1
              1
              1
              2
              2
              3
              3
              4
              4
          ];

T=[noro_train_rs;noro_test_rs];
T=T';
%egitim ve test verileri birlestiriliyor
P=[noro_train_data;noro_test_data];
P=P';
%som agi olusturuldu
```

```
net=newc(minmax(P),4);
net=init(net);
net.trainParam.epochs = 1000;
net=train(net,P);
a=sim(net,P);
a=vec2ind(a);

%som'um ayirdigi siniflarin
%gercekte hangi sinifi temsil ettigi
%kontrol ediliyor
kontrol=zeros(4,4);

for J=1:4
    for I=1:47
        if T(1,I)==J
            if a(1,I)==1
                kontrol(J,1)=kontrol(J,1)+1;
            elseif a(1,I)==2
                kontrol(J,2)=kontrol(J,2)+1;
            elseif a(1,I)==3
                kontrol(J,3)=kontrol(J,3)+1;
            elseif a(1,I)==4
                kontrol(J,4)=kontrol(J,4)+1;
            end
        end
    end
end
```