BİYOMEDİKAL İŞARETLERİN SINIFLANDIRILMASI

İ. Selçuk BÜYÜKYILMAZ
Yüksek Lisans Tezi
Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Danışman : Yrd. Doç. Dr. Hasan DEMİR

2012

T.C.

NAMIK KEMAL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BİYOMEDİKAL İŞARETLERİN SINIFLANDIRILMASI

İ. Selçuk BÜYÜKYILMAZ

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Yrd. Doç. Dr. Hasan DEMİR

TEKİRDAĞ – 2012

Yrd. Doç. Dr. Hasan DEMİR danışmanlığında, İ. Selçuk BÜYÜKYILMAZ tarafından hazırlanan bu çalışma aşağıdaki jüri tarafından Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda yüksek lisans tezi olarak kabul edilmiştir.

Jüri Başkanı : Yrd.Doç.Dr. Rafet AKDENİZ

Üye : Yrd.Doç.Dr. Hasan DEMİR (Danışman)

Üye : Yrd.Doç.Dr. Hale Pınar ZENGİNGÖNÜL

Fen Bilimleri Enstitüsü Yönetim Kurulu adına

Prof. Dr. Fatih KONUKCU

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

BİYOMEDİKAL İŞARETLERİN SINIFLANDIRILMASI

İ. Selçuk BÜYÜKYILMAZ

Namık Kemal Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Hasan DEMİR

Bu tezde, PhysioNet veritabanından alınan elektroensefalografi (EEG) işaretleri için, açık kaynak kodlu programlar kullanılarak öznitelikler elde edilmiş ve sara krizi tespit edilmeye çalışılmıştır. EEG işaretlerine ait spektral entropi, Hjorth parametreleri, tekil değer ayrıştırma entropisi, Fisher bilgisi, yaklaşık entropi, Hurst katsayısı, örnek entropisi, Petrosian fraktal boyutu, Katz fraktal boyutu, Sevcik fraktal boyutu ve Hjorth fraktal boyutu hesaplanarak, yapay sinir ağları, destek vektör makineleri ve en yakın k-komşu algoritması ile sınıflandırılmıştır. Aynı işlemler EEG işaretlerinin dalgacık katsayıları için de tekrar edilmiştir. Böylelikle her bir sınıflayıcı ve parametre için en iyi durumlar elde edilmeye çalışılmıştır.

Anahtar kelimeler : Elektroensefalografi, en yakın k-komşululuk, yapay sinir ağları, destek vektör makineleri, fraktal boyut, EEG sınıflama

2012, 104 sayfa

ABSTRACT

MSc. Thesis

CLASSIFICATION OF BIOMEDICAL SIGNALS

İ. Selçuk BÜYÜKYILMAZ

Namık Kemal University
Graduate School of Natural and Applied Sciences
Main Science Division of Electronics and Telecommunication Engineering

Supervisor : Assist. Prof. Dr. Hasan DEMİR

In this thesis, features for electroencephalography (EEG) signals obtained from PhysioNet database are extracted using open source coded programs and epileptic sezures are tried to detect. Spectral entropy, Hjorth parameters, singular value decomposition entropy, Fisher information, approximate entropy, Hurst exponent, sample entropy, Petrosian fractal dimension, Katz fractal dimension, Sevcik fractal dimension and Hjorth fractal dimension of the EEG signals are calculated and they are classified by artificial neural networks, support vector machines and k-nearest neighbor algorithm. The same processes are repeated for the wavelet coefficients of the EEG signals. Thus, best results are tried to achieve for each classifier and parameter. Open source coded programs are used for extracting features and classifying.

Key words : Electroencephalography, k-nearest neighborhood, artificial neural networks, support vector machines, fractal dimension, EEG classification

2012, 104 pages

TEŞEKKÜR

Yüksek lisans tezimin planlanması ve yürütülmesindeki bilgi ve desteğini esirgemeyen Sayın Hocam Yrd. Doç. Dr. Hasan DEMİR'e tüm çalışmalarım boyunca yapmış olduğu değerli yardım ve katkılarından dolayı teşekkür etmeyi bir borç bilirim.

Tezimin yürütülmesi sırasında desteklerini gördüğüm Elektronik ve Haberleşme Bölümündeki bütün hocalarıma ve öğretim elemanlarına teşekkür ederim.

Ayrıca bana daima destek olan değerli aileme teşekkürlerimi sunarım.

İÇİNDEKİLER **Sayfa No.**

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler

$\underline{a}$	Dalga Şeklinin Ortalama Adımı
B	Boyut, Fraktal Boyut
C	Santral
d	Eğrinin Çapı, Komşuya Olan Uzaklık
F	Frontal
H	Hurst Katsayısı
L	Eğrinin Toplam Uzunluğu, Lagrange Çarpanı
N	Yeni Çubukların Sayısı, Nokta Sayısı
O	Oksipital
P	Parietal
sn	Saniye
T	Temporal
Z	Orta Çizgi Üzerine Yerleştirilmiş Bir Elektrot
μV	Mikrovolt
Є	Ölçekleme Faktörü

Kısaltmalar

ADVM	Ağ Tabanlı Destek Vektör Makinesi
AHİ	Analitik Hiyerarşi İşlemi
ASCII	American Standard Code for Information Interchange
AUÖ	Ayırt Edici Uzamsal Örüntüler
BBA	Beyin-Bilgisayar Arayüzü
CHB-MIT	Children's Hospital Boston - Massachusetts Institute of Technology
DÖM	Destek Öznitelik Makinesi
DVM	Destek Vektör Makinesi
DVR	Destek Vektör Regresyonu
EDF	European Data Format
EEG	Elektroensefalogram, Elektroensefalografi
EKG	Elektrokardiyogram, Elektrokardiyografi

FB	Fraktal Boyut
HH	Hjorth Hareketliliği
HİP	Hareketle İlgili Potansiyeller
HK	Hjorth Karmaşıklığı
HKKO	Hata Kareleri Karekökünün Ortalaması
İVM	İlişki Vektör Makinesi
KP	Kuadratik Programlama
LM	Levenberg-Marquardt
MEG	Magnetoensefalografi
mlpy	Machine Learning Python
MR	Manyetik Rezonans
OİA	Olayla İlgili Asenkronizasyon
OUÖ	Ortak Uzamsal Örüntüler
ÖB	Özbağlanım
ÖE	Örnek Entropisi
PSG	Polisomnogram, Polisomnografi
pyeeg	Python + EEG/MEG
REN	Renyi Entropisi
SEN	Spektral Entropi
sklearn	scikit-learn
TDA	Tekil Değer Ayrıştırma
UÖM	Uç Öğrenme Makinesi
USBK	Uyarlamalı Sinir-Bulanık Karışma
VNS	Vagal Nerve Stimulus
YE	Yaklaşık Entropi
YSA	Yapay Sinir Ağı

ŞEKİLLER DİZİNİ **Sayfa No.**

ÇİZELGELER DİZİNİ **Sayfa No.**

1. GİRİŞ

Bu çalışmada, Physiobank arşivinden elde edilmiş olan CHB-MIT (Children's Hospital Boston - Massachusetts Institute of Technology) Kafa Derisi EEG (Elektroensefalogram) Veritabanı hakkında bilgi verilmiş olup uygulamada bu veritabanının verileri kullanılmıştır. Spektral entropi, Hjorth parametreleri, tekil değer ayrıştırma entropisi, Fisher bilgisi, yaklaşık entropi, Hurst katsayısı, örnek entropisi, Petrosian fraktal boyutu, Katz fraktal boyutu, Sevcik fraktal boyutu gibi çeşitli öznitelik elde etme yöntemleri incelenmiştir. Ayrıca elde edilen öznitelik vektörlerine uygulanmak üzere sınıflandırma yöntemleri olarak yapay sinir ağları, destek vektör makineleri ve en yakın k-komşu algoritması kullanılmıştır.

Khorshidtalab ve Salami (2011)'nin çalışmasında beyin-bilgisayar arayüz (BBA) tabanlı EEG sinyallerinin gerçek zamanlı sınıflandırılması için farklı algoritmaların performansları karşılaştırılmıştır. BBA, kullanıcının herhangi bir kas hareketi yerine beyin sinyallerini kullanmaktadır. Bu sistem, ciddi motor rahatsızlıkları olan insanların beyin dalgaları yardımıyla elektronik cihazlara komut göndermesini sağlamaktadır. Sinyaller belirlenmelidir, işlenmelidir ve özel komutlara sınıflandırılmalıdır. Öznitelik çıkarma ve sınıflandırma yöntemleri BBA sistemlerinde en önemli rolü oynamaktadır. Çünkü herhangi bir yanlış sınıflandırma ve hata yanlış bir komutla sonuçlanabilmektedir.

Naderi ve Nasab (2010)'ın çalışması, EEG sinyallerindeki sara krizlerinin otomatik tespiti için üç aşamalı bir teknik önermektedir. Örüntü tanımanın pratik uygulamalarında genellikle, tanınması gereken, satır verilerinden elde edilen farklı öznitelikler vardır. Önerilen yöntem zaman dizisi işareti, spektral analiz ve tekrarlayıcı sinir ağlarına dayalıdır. Karar verme üç aşamada gerçekleştirilmiştir: a) Welch yöntemi güç spektral yoğunluğu hesaplaması kullanarak öznitelik elde etme, b) Elde edilen öznitelikler ve zaman dizisi işareti örnekleri üzerinde istatistikler kullanarak boyut azaltma, c) Tekrarlayıcı sinir ağları kullanarak EEG sınıflandırması. Bu çalışma göstermiştir ki Welch yöntemi güç spektral yoğunluğu hesaplaması EEG işaretlerini iyi temsil eden uygun bir özniteliktir. Bu çalışma EEG işaretlerini sınıflandıran diğer araştırmalarla karşılaştırıldığında tam %100'lük bir başarıyla diğer çalışmalardan daha yüksek belirleyicilik, hassasiyet ve sınıflandırma doğruluğu elde etmiştir.

Boashash ve ark. (2011)'nın çalışması işaret işlemede zaman-frekans yöntemlerine bir giriş ile, işaretle ilgili öznitelikler ve görüntü ile ilgili özniteliklerin birleşimine dayalı EEG anormallikleri tespiti ve sınıflandırması için yeni bir yöntem sunmaktadır. EEG işaretlerinin

durağan olmayan yapısı ve çok bileşenli karakteristiğini karakterize eden bu öznitelikler, işaretlerin zaman-frekans gösteriminden elde edilmektedir. İşaretle ilgili öznitelikler EEG işaretlerinin zaman-frekans gösteriminden elde edilir ve anlık frekans, tekil değer ayrışımı ve enerji tabanlı öznitelikleri içermektedir. Görüntü ile ilgili öznitelikler, zaman-frekans görüntü işleme teknikleri kullanılarak bir görüntü olarak ele alınan zaman-frekans gösteriminden elde edilmektedir. Bu birleştirilmiş işaret ve görüntü öznitelikleri, bir işaretten, daha fazla bilgi elde edilmesini sağlamaktadır. Yeni doğan ve yetişkin EEG verileri üzerinde elde edilen sonuçlar, görüntü ile ilgili özniteliklerin, çoklu destek vektör makinesi sınıflandırıcısına dayalı sınıflandırma sistemlerinde EEG kriz tespitinin performansını geliştirdiğini göstermektedir.

Çınar ve Şahin (2010)'in çalışması EEG sinyalleri için farklı sınıflandırma tekniği uygulamalarını incelemiştir. Fuzzy Fonksiyonları Destek Vektör Sınıflandırıcısı, Gelişmiş Fuzzy Fonksiyonları Destek Vektör Sınıflandırıcısı ile Parçacık Yığın Eniyileştirmesi ve Radyal Temel Fonksiyon Ağlarını kullanarak tasarlanmış olan yeni bir hibrit tekniği üzerinde çalışılmıştır. Tekniklerin sınıflandırma performansı, kamuya açık olan ve birçok BBA araştırmacısı tarafından kullanılan aynı standart veri kümeleri üzerinde karşılaştırılmıştır. Sonuçlar göstermiştir ki, önerilen sınıflandırıcılar, en gelişmiş sınıflandırıcıların sınıflandırma performansına ulaşabilmektedir ve EEG sinyallerinin sınıflandırma uygulamalarında alternatif teknikler olarak kullanılabilmektedir.

EEG örüntü sınıflandırma, beyin bilgisayar ara yüzünde önemli bir rol oynamaktadır. Bununla birlikte, EEG verisi, gürültüyü ve insan tarafından yapılan etkileri kapsayan çok değişkenli bir zaman dizisi verisidir. Oh ve ark. (2006)'ın çalışmalarında, temel bileşen analizi ve sinir ağlarını beraber kullanan EEG örüntü sınıflandırma için yöntemler sunulmuştur ve bu hibrit yönteminin, güvenilir EEG sinyali sınıflandırması için daha iyi bir imkan sunduğuna inanılmaktadır.

Stastny ve ark. (2001)'nın çalışması, Saklı Markov Modellerine dayalı bir sistem kullanarak basit hareketlerin sınıflandırılmasını açıklamaktadır. İşaret parmağını hızlı bükme ve uzatmalar ile omuz ve parmağın hareketleri kafa derisi EEG işaretleri kullanılarak sınıflandırılmıştır. Bu çalışmanın amacı, beyin yarımkürelerinin aynı kafa derisi elektrotları üzerinde EEG değişimleri gösteren hareketlerin sınıflandırılması için bir sistem geliştirmek olmuştur. Bir elin hareketleri ile ilgili EEG örüntülerinin sınıflandırılması zordur çünkü hareketlerin çözülmesi bir kayıt yerindeki EEG değişiklerinin geçici oluşumuna

dayanabilmektedir. EEG dalga şekillerinin çok değişken olması içerik bilgisini kullanmayı gerektirmektedir.

Fukuda ve ark. (1995)'nın çalışmasında, EEG sinyallerinin bir insan arayüz aracı olması ihtimalini değerlendirmek için, basit ve kullanışlı bir elektroensefalograf ile ölçülen EEG sinyallerinin bir örüntü sınıflandırma yöntemi önerilmiştir. Deneklerden göz durumlarını değiştirmeleri istenmiştir veya 450 saniyelik sözde rastgele diziye göre sürekli yanıp sönen bir flaş ışığına maruz bırakılmışlardır. Deneyler boyunca EEG işaretleri ölçülmüştür ve sınıflandırma için kullanılmıştır. Gözün açılması ve kapanması ile flaş ışığının varlığı ve yokluğu gibi iki uyarım durumuna bağlı olarak her EEG işaretinin farklı dağılımı olabilmektedir. Bu nedenle bir istatistiksel model ile birlikte log-doğrusallaştırılmış Gauss Karışım Sinir Ağı kullanılmaktadır. Deneylerden gösterilmiştir ki, EEG işaretleri, yeterli olarak sınıflandırılabilmektedir ve sınıflandırma hızları eğitim verisinin sayısına ve öznitelik vektörlerinin boyutuna bağlı olarak değişmektedir.

İstatistikte bilgisayarların artan kullanımı, yeni bir çok değişkenli istatistik teknikleri jenerasyonu meydana getirmiştir. Bunların en önemlisi, sınıflandırma ve regresyon analizi için ağaç yapılı bir yöntemdir. Sınıflandırma ve Regresyon Ağaçları programı, en iyi ikili veri bölümleri için tekrarlayıcı bir araştırmaya dayalı olan özyinelemeli bir bölümlendirme işlemi uygulamaktadır. Sonuçta elde edilen sınıflandırıcılar, yaprakları sınıf etiketlemeyi belirleyen ikilik ağaçlardan oluşmaktadır. Veri yeniden örnekleme tekniklerinin kapsamlı kullanımı, kutuplanmış sınıflandırıcı performans ölçülerinin yerine geçmektedir. Grajski ve ark. (1986)'nın çalışmasında, ağaç yapılı yöntem ve veri yeniden örnekleme teknikleri incelenmiştir. Bir durum çalışması, veri keşfi ve sınıflandırılmasında, Sınıflandırma ve Regresyon Ağaçlarının kullanımını vurgulamaktadır. Veriler, tavşanların koklama haznesinden kaydedilen EEG'nin uzamsal örüntülerinden oluşmaktadır. Sınıflandırma ve Regresyon Ağaçları, önceki örüntü analizlerini doğrulamak ve EEG'nin koku belirliliğinin, evrensel bir dalga şeklinin uzamsal olarak modüle edilmiş bir genlik örüntüsünde yattığını göstermek için kullanılmaktadır. Bu çalışma, EEG aktivitesinin uzamsal örüntülerinin sınıflandırılması için ilk ağaç yapılı yöntem uygulamasıdır.

Ko ve Sim (2011)'in çalışması, Uyarlamalı Sinir-Bulanık Karışma (USBK) modeline dayalı yöntem kullanarak gerçekleştirilen motor imgesel işi ile ilgili EEG sinyalleri gibi insan beyin aktivitesinin sınıflandırılması için yeni bir yöntem açıklamaktadır. Önerilen yöntem, motor imgesel EEG sinyallerinin sınıflandırılması için harmoni araştırma algoritması kullanarak USBK modelinin eniyilemesinin kullanılabilirliğinin gösterilmesine odaklanmıştır.

Eniyilemeden önce, USBK modeli sınıflandırıcısının öznitelikleri Hjorth parametreleri ile belirlenmiştir. Harmoni araştırma algoritması; geriye yayılma, dereceli azaltma yöntemi gibi diğer USBK modeli eğitim teknikleri ile beraber kullanıma izin vermek için yeterli derecede uyarlanabilirdir. Önerilen yöntemi simüle etmek için üç tip motor imgesel işi gerçekleştirilmektedir ve EEG sinyallerinin sınıflandırılma sonuçları önceki yöntemlerle karşılaştırıldığında iyi performans göstermektedir.

EEG sinyallerinin sınıflandırılması, EEG'ye dayalı BBA'da en önemli konudur. Tipik olarak böyle bir sınıflandırma, bir seçilmiş EEG algılayıcıları kümesinden elde edilen işaretler kullanılarak gerçekleştirilmektedir. EEG algılayıcı sinyalleri, düşük sinyal-gürültü oranına sahip olan, etkin sinyalle gürültünün karışımları olduğu için, motor imgesel EEG işaretlerinin sınıflandırılması zor olabilmektedir. Xiao ve ark. (2009)'nın çalışmasında, Enerji entropisi, motor imgesel EEG verilerinin ön işlemesi için kullanılmıştır ve Fisher sınıf ayrılabilme kriteri öznitelikleri belirlemek için kullanılmıştır. Sonunda, bir doğrusal ayırt etme yöntemi ya da çok katmanlı geriye yayılma sinir ağları ve destek vektör makineleri ile, dört tip motor imgesel EEG'sinin sınıflandırılması gerçekleştirilmiştir. Sonuçlar göstermiştir ki, bu çalışmada önerilen yöntem kullanılarak elde edilen sınıflandırma doğruluğu, üç deneğin herhangi bir çeşit kombinasyonundaki geriye yayılma sinir ağları veya destek vektör makinesi kullanılarak elde edilen sınıflandırma doğruluğundan çok daha yüksektir.

Skinner ve ark. (2007)'nın çalışması, geniş durum dizileri (108 bit) kullanılarak temsil edilen, gürültüyü ve insan tarafından yapılan etkileri içeren insan EEG işaretlerinin sınıflandırılması için, genetik tabanlı öğrenen sınıflandırıcı sistemin etkinliğini araştırmaktadır. Üç katılımcıdan elde edilen EEG işaretleri, yarı küreye ait cevapları ortaya çıkarmak için tasarlanan dört zihinsel işi gerçekleştirirken kaydedilmiştir. Özbağlanımlı modeller ve Fast Fourier Dönüşümü yöntemleri, zihinsel işlerin ayırt edilebileceği öznitelik vektörlerini oluşturmak için kullanılmıştır. Genetik tabanlı öğrenen sınıflandırıcı sistem, %99,3'lük bir maksimum sınıflandırma doğruluğu ve %88,9'luk bir en iyi ortalama elde etmiştir. Genetik tabanlı öğrenen sınıflandırıcı sistemin göresel sınıflandırma performansı, farklı öğrenme tekniklerinden ortaya çıkan evrimsel olmayan dört sınıflandırıcı sistemle karşılaştırılmıştır. Deneysel sonuçlar, felçli kişilerin elektrikli tekerlekli sandalye veya diğer cihazları kontrol etmelerini sağlamak için bir arayüz olarak EEG işaretlerini kullanmanın uygunluğunu araştırırken daha büyük bir çalışmanın bir parçası olarak kullanılacaktır.

Tekli deneme EEG sınıflandırması BBA geliştirmede önemlidir. Bununla beraber ortak uzamsal örüntüler gibi popüler sınıflandırma algoritmaları, bilgi pratik uygulamalarda

her zaman bilinmemesine rağmen, gürültüyü kaldırmak için, genellikle yüksek derecede, öncelikli nörofizyolojik bilgiye bağlıdır. Li ve ark. (2009)'nın çalışmasında, tekli deneme EEG sınıflandırması için, öncelikli nörofizyolojik bilgi olmaksızın iyi çalışan yeni bir tensör tabanlı yöntem önerilmiştir. Bu yöntemde, EEG işaretleri, dalgacık dönüşümü ile uzamsal spektral geçici etki alanında gösterilmektedir; çoklu doğrusal ayırt edici alt uzay, genel tensör diskriminant analizi ile ayrılmaktadır; gereksiz ayırt edici olmayan örüntüler Fisher skoru ile çıkarılmaktadır; ve sınıflandırma destek vektör makinesi ile yapılmaktadır. Özellikle öncelikli nörofizyolojik bilginin olmadığı durumlarda, üç veri kümesindeki uygulamalar, EEG işaretleri analizinde önerilen tensör yönteminin etkinliğini ve gücünü doğrulamaktadır.

Ortak uzamsal örüntüler BBA bağlamında EEG işaretlerini sınıflandırmak için popüler bir algoritmadır. Lu ve ark. (2010)'nın çalışması, küçük örnek ayarında, ortak uzamsal örüntü için bir düzenleme ve birleştirme tekniği sunmaktadır. Klasik ortak uzamsal örüntü, örnek tabanlı kovaryans matrisi hesaplamasına dayalıdır. Bu nedenle, eğitim örneklerinin sayısı az olursa, EEG sınıflandırmasındaki performansı kötüleşmektedir. Bu konuda, hesaplama kutbunu azaltırken hesaplama varyansını azaltmak için iki parametre ile kovaryans matrisi hesaplamanın düzenlendiği bir düzenlenmiş ortak uzamsal örüntü algoritması sunulmaktadır. Düzenleme parametresi belirleme problemini çözmek için, topluluk tabanlı bir çözüm veren bir dizi düzenlenmiş ortak uzamsal örüntünün bir araya toplandığı, birleştirmeli düzenlenmiş ortak uzamsal örüntü ayrıca önerilmektedir. Önerilen algoritma, rekabet eden diğer dört algoritmaya karşı, Üçüncü BBA Yarışmasının dördüncü veri kümesi üzerinde değerlendirilmektedir. Deneyler göstermektedir ki, küçük örnek ayarındaki özel üstünlük ile, çeşitli test senaryolarında, üç deney kümesindeki ortalama sınıflandırma performansında, birleştirmeli düzenlenmiş ortak uzamsal örüntü, diğer yöntemlere göre önemli ölçüde daha üstün performans sergilemektedir.

Han ve Sun (2010)'un çalışmasında, İlişki Vektör Makinesi (İVM) ve Özbağlanım (ÖB) modeline dayalı yeni bir EEG işareti sınıflandırma yöntemi önerilmektedir. Bu yöntem, kriz anındaki EEG işaretleriyle krizler arası EEG işaretlerini iyi ayırt edebilmektedir. Bu, sara teşhisinde çok önemlidir. Bu çalışma üç kısma ayrılabilmektedir: İlk olarak, ÖB modellerine dayalı olarak işaretlerden EEG öznitelikleri elde edilir ve sonra bu özniteliklerin performansı değerlendirilmektedir. İkinci olarak, özniteliklerin performansına bağlı olarak, öznitelik belirleme ile sınıflandırıcılar arasında öznitelik seçimi yapılmaktadır. Son olarak, yöntemi gözden geçirmek için, farklı ÖB modelleri, farklı çekirdek genişlikleri ve farklı öznitelik alt kümeleri ile, İVM uygulanmaktadır. Sonuçlar göstermektedir ki: 1) Sara teşhisi için EEG

sinyali sınıflandırma görevinde, ÖB modellerine dayalı olarak elde edilen öznitelikler, EEG işaretlerini iyi temsil edebilmektedir. 2) Öznitelik belirleme ve sınıflandırıcılar arasında öznitelik seçimine ihtiyaç duyulmaktadır. 3) İVM ve ÖB modeline dayalı olan yöntem, iki EEG işaret tipini iyi ayırt edebilmektedir.

1921'de, Berger, tüm kafatası üzerine yerleştirilen elektrotlarla, beynin elektriksel aktivitesinin ilk kayıtlarını gerçekleştirmiştir. Kaydedilen işaretlerin frekans içeriğinin, bu işaretleri ve beynin durumunu tanımlamada önemli bir rol oynadığı hemen anlaşılmıştır. Isaksson ve ark. (1981)'ın çalışması, EEG'nin temel özelliklerini araştırmaktadır ve çeşitli etkili faktörleri belirtmektedir. Görsel olarak ekranda göstermeyi tamamlamak için EEG'yi nicelemek üzere birçok yöntem geliştirilmiştir; bunlar parametrik ve parametrik olmayan şeklinde sınıflandırılmıştır. Bu çalışma, işaret analizinin, gözlemlenen işlemin matematiksel bir modeline dayalı olduğu parametrik yöntemlerin üzerinde durmaktadır. Skaler ya da çok değişkenli model, zamanla değişmeyen ya da zamanla değişen parametrelerle, tipik olarak doğrusaldır. Modeli gözlemlenen veriye uydurmak için algoritmalar araştırılmıştır. Analiz sonuçları, karakteristik değişkenlerin zamanla değiştiği şekli de içeren, EEG'nin spektral özelliklerini açıklamak için kullanılabilmektedir. Parametrik modeller, ani yükseliş ve keskin dalga denilen, epileptik orijin ile geçişlerin meydana gelişini tespit etmek için başarılı bir şekilde uygulanmıştır. Beynin önemli işlevsel durumlarını anlamak için parametre hesaplamasını sınıflandırma algoritmasıyla birleştirerek ilginç sonuçlar da elde edilmiştir.

EEG beynin elektriksel aktivitesinin kayıtlarıdır ve epilepsi gibi nörolojik hastalıkların teşhisi için vazgeçilmez bir araçtır. EEG'ler gibi durağan olmayan işaretlerin analizi için dalgacık dönüşümü etkili bir araçtır. Dalgacık analizi EEG'yi delta, teta, alfa, beta ve gama alt bantlarına ayırmak için kullanılmaktadır. Lyapunov katsayısı, işaretin doğrusal olmayan kaotik dinamiklerini nicel olarak değerlendirmek için kullanılmaktadır. Ayrıca, beyin aktivitesinin farklı durumları, Lyapunov katsayıları gibi, doğrusal olmayan değişmez ölçülerle belirlenen farklı kaotik dinamiklere sahiptir. Olasılıksal sinir ağı ve radyal taban işlev sinir ağı test edilmiştir ve bunların sınıflandırma hızı performansları, ölçüt veri kümesi kullanılarak değerlendirilmiştir. Karar verme iki aşamada gerçekleştirilmiştir: Lyapunov katsayılarını ve dalgacık katsayılarını hesaplayarak öznitelik belirleme ve belirlenen öznitelikler üzerinde eğitilen sınıflandırıcıları kullanarak sınıflandırma. Murugavel ve ark. (2011)'nın çalışması göstermiştir ki, Lyapunov katsayıları ve dalgacık katsayıları EEG işaretlerini iyi temsil eden özniteliklerdir ve bu öznitelikler üzerinde eğitilen çok sınıflı destek vektör makinesi ve olasılıksal sinir ağı %96 ve %94 gibi yüksek sınıflandırma doğrulukları elde etmiştir.

Bir denek değişen hayali zihinsel işleri gerçekleştirirken kaydedilen EEG'yi sınıflandırma kabiliyeti, kullanılabilir beyin bilgisayar arayüzleri oluşturmak için temel ortaya koyabileceği gibi klinik ayarlarda kullanılan EEG analizi yazılımının performansını da geliştirebilmektedir. Birçok araştırma grubu EEG sınıflandırıcıları üretmiş olmasına rağmen, bu yöntemler birçok pratik uygulamada kullanım için kabul edilebilir bir performans seviyesine henüz ulaşamamıştır. Forney ve Anderson (2011), günümüzdeki yöntemlerin, EEG'de içerilen geçici ve uzamsal örüntüleri yakalama kabiliyetleri tarafından sınırlanmış olduğunu ileri sürmektedirler. Bu problemleri çözebilmek için, Elman tekrarlı sinir ağlarını kullanan, EEG sınıflandırması için yeni üretken bir teknik önermektedirler. Bir denek çeşitli hayali zihinsel işlerden birini gerçekleştirirken kaydedilen EEG ilk olarak, işareti zamanda bir adım önde öngörmek için bir ağ eğiterek modellenmiştir. Bu modeller 0,110 kadar düşük bir hataların ortalama karekökü ile EEG'yi iyi öngörebilmektedir. Daha sonra ayrı bir model, her sınıfa ait EEG üzerinden eğitilmektedir. Önceden görülmeyen verinin sınıflandırılması her modeli uygulayarak ve en düşük öngörü hatasını üreten ağ ile ilgili sınıf etiketini atayarak gerçekleştirilmektedir. Bu yöntem, iki sağlam vücutlu denek ve yüksek derecede omurilik zedelenmesi olan bir denekten elde edilen EEG üzerinde test edilmiştir. Dakikada 38,7 bitlik bir bit oranı veren her saniye verilen kararlar ile iki görevli bir problem için %99,3 kadar yüksek sınıflandırma oranları elde edilmiştir.

BBA, EEG gibi bir beyin işareti ile yansıtılan insan niyetini bir çıkış cihazı için bir kontrol işaretine çeviren bir haberleşme kanalı sağlamak içindir. Son yıllarda, Olayla İlgili Asenkronizasyon (OİA) ve Hareketle İlgili Potansiyeller (HİP), motorla ilgili BBA sisteminde önemli öznitelikler olarak kullanılmaktadır ve Ortak Uzamsal Örüntüler (OUÖ) algoritmasının, OİA tabanlı sınıflandırma için çok kullanışlı olduğu gösterilmiştir. Bununla birlikte, HİP'ler salınmayan yavaş EEG potansiyel kaymaları olduğu için, HİP tabanlı sınıflandırıcı için OUÖ uygun bir yöntem değildir. Liao ve ark. (2007)'nın çalışmasında, başka bir uzamsal filtreleme algoritması olan Ayırt Edici Uzamsal Örüntüler (AUÖ), HİP'lerin genliklerindeki farkın daha iyi belirlenmesi için bir yenilik olarak sunulmaktadır ve AUÖ, bilinçli olarak yapılan sola karşı sağ parmak hareket görevleri boyunca kaydedilen EEG işaretlerinden elde edilen öznitelikleri belirlemek için OUÖ ile birlikte kullanılmaktadır. Öznitelikler için sınıflandırıcı olarak bir destek vektör makinesi tabanlı sistem tasarlanmıştır. Sonuçlar göstermektedir ki, HİP'ler ve OİA öznitelikleri için, birleştirilmiş uzamsal filtreler, tekli deneme EEG sınıflandırmasını, AUÖ ve OUÖ'nün her birinin tek başına gerçekleştirdiğinden daha iyi gerçekleştirebilmektedir. Bu çalışmada, destek vektör makinesi

ile sınıflandırılan, biri OUÖ (OİA)'ya dayalı olan ve diğeri AUÖ (HİP)'ye dayalı olan iki öznitelik kümesi ile bir EEG tabanlı BBA sistemi önerilmektedir.

Nörologların, bir hastanın saralı mı yoksa gerçekte farklı bir hastalıktan kaynaklandığı halde sadece sarayla bağlantılı belirtiler mi gösteriyor olduğunu belirlemelerine ve teşhis etmelerine yardımcı olabilecek hızlı bir görüntüleme işlemi için acil bir ihtiyaç bulunmaktadır. Yanlış bir teşhisin, özellikle ameliyat odalarında ve yoğun bakım ünitelerinde ölümcül sonuçları olabilmektedir. Sara ve diğer beyin hastalıklarına uyabilen beyin işlevlerini değerlendirerek hastaları teşhis etmek için altın bir standart olarak EEG geleneksel olarak kullanılmaktadır. Bu nedenle, Chaovalitwongse ve ark. (2011)'nın araştırması, çok kanallı EEG kayıtları için yeni sınıflandırma teknikleri geliştirmek üzerine odaklanmıştır. İki zaman dizisi sınıflandırma tekniği, Destek Öznitelik Makinesi (DÖM) ve Ağ Tabanlı Destek Vektör Makinesi (ADVM), bir kişinin saralı olup olmadığını EEG okumalarından öngörmek için bu çalışmada önerilmektedir. DÖM yöntemi, zaman dizisi benzerlik ölçülerine dayalı olan güçlü sınıf ayırabilme kabiliyetine sahip bir elektrot grubu (öznitelikler) seçerek sınıflandırma doğruluğunu en üst seviyeye çıkaran bir eniyileme modelidir ve eğitim aşamasında EEG örneklerini doğru olarak sınıflandırmaktadır. ADVM yöntemi, EEG verisinin hem uzamsal hem de geçici karakteristiklerinden yararlanmak için, geleneksel Destek Vektör Makineleri (DVM) ile çok boyutlu zaman dizisi verileri için yeni bir ağ tabanlı modeli birleştirmektedir. Önerilen teknikler, biri on diğeri beş adet hastadan elde edilen iki EEG kümesi üzerinde test edilmiştir. DVM ve karar verme ağaçları gibi diğer yaygın kullanılan sınıflandırma teknikleri ile karşılaştırıldığında, önerilen DÖM ve ADVM teknikleri, çok umut vaat edici ve pratik sonuçlar sağlamaktadır ve geleneksel tekniklerden daha az zaman ve bellek kaynağı gerektirmektedir. Bu çalışma, insan sara teşhisi ve tedavisinin gelişmesi için gerekli bir veri madenciliği uygulamasıdır.

Sara, beyin fonksiyon bozukluğu ve idrak ile ilgili hastalıklarla sonuçlanabilen sık karşılaşılan beyin hastalıklarından biridir. Sara krizleri, beynin geçici ve beklenmeyen elektriksel kesintilerine bağlı olarak meydana gelebilmektedir. EEG, korteksle ilgili davranışların doğrudan bir değerlendirmesini yapabilen, insan beyin dinamiklerini analiz etmek için girişimsel olmayan yöntemlerden biridir. Krizler, çok büyük genlikli, kısa ve süreksiz, sinir hücreleri ile ilgili eş zamanlı boşalmalar ile belirtilmektedir. Bu düzgün olmayan eş zamanlılık, benzer şekilde beyinde, sadece birkaç kanalda görülebilen kısmi krizler ya da bütün beyinle ilgili EEG işaretinin her kanalında görülen genel krizler olarak meydana gelebilmektedir. Bugünkü sara analizi, son derece eğitimli klinik tedavi uzmanları

tarafından yapılan zor olan bir görsel taramaya bağlıdır. Verilerin kaydedilmesi çok uzun veriler meydana getirmektedir ve bu nedenle saranın araştırılması ve belirlenmesi teşhis için daha fazla zaman almaktadır. Günümüzde bilgisayarlaştırılmış sistemlere genellikle teşhisi kolaylaştırmak için yer verilmektedir. Geetha ve Geethalakshmi (2011)'nin çalışması, sinir ağları kullanarak otomatik sara EEG tespitinin bir uygulamasını tartışmaktadır. Bu çalışmada, normal, kriz anındaki ve krizler arası EEG işaretlerini sınıflandırma işini gerçekleştirmek için bir öznitelik belirleme yöntemi olarak örnek entropisi denilen bir istatistiksel parametre kullanılmıştır. Burada örnek entropisinin değeri sara krizlerinin tespitinde önemli bir rol oynamaktadır. Örnek entropisinin değerinin sara krizi sırasında ani olarak düştüğü gözlemlenmektedir ve bundan dolayı bu bilgi saranın tam olarak meydana gelişini açıklamaktadır. Sınıflandırma stratejileri, Geriye Yayılma Sinir Ağı, Destek Vektör Makinesi, Uç Öğrenme Makinesi (UÖM) gibi modellere dayalıdır. Hibrit UÖM, bu çalışmada icat edilen sınıflandırma yöntemidir ve EEG sınıflandırması için uygulanmak üzere türünün ilk örneğidir. Bu sınıflandırma yöntemi, giriş ağırlıklarını ve saklı kutupları seçmek için Analitik Hiyerarşi İşlemini (AHİ), çıkış ağırlıklarını analitik olarak belirlemek için UÖM algoritmasını, ağı öğrenmek için Levenberg-Marquardt (LM) algoritmasını kullanmaktadır. Deneysel sonuçlar göstermektedir ki, örnek entropisi kullanarak otomatik sara tespiti ve hibrit UÖM daha az zamanda daha iyi doğruluk elde etmektedir.

Destek vektör ağı, iki gruplu sınıflandırma problemleri için yeni bir öğrenme makinesidir. Makine kavramsal olarak şu düşünceyi uygulamaktadır: Giriş vektörleri, çok yüksek boyutlu bir öznitelik uzayına doğrusal olmayarak dönüştürülmektedir. Bu öznitelik uzayında bir doğrusal karar verme yüzeyi oluşturulmaktadır. Karar verme yüzeyinin belirli özellikleri, öğrenme makinesinin yüksek genelleştirme kabiliyetini sağlamaktadır. Destek vektör ağının ardında yatan düşünce, eğitim verisinin hatasız olarak ayrılabildiği sınırlandırılmış durum için geçmişte uygulanmıştır. Bunun sonuçları ise Cortes ve Vapnik (1995)'in çalışmasında ayrılamayan eğitim verisi için genişletilmiştir. Çok terimli giriş dönüşümlerinden yararlanan destek vektör ağlarının yüksek genelleştirme kabiliyeti gösterilmiştir. Ayrıca, hepsi bir optik karakter tanıma kriter çalışmasında yer alan çeşitli klasik öğrenme algoritmaları ile destek vektör ağının performansı karşılaştırılmıştır.

2. ELEKTROENSEFALOGRAFİ (EEG)

İnsanın sinir sistemi, yaklaşık 10 milyar sinir hücresi içerir. Bunların çoğu beyinde, geri kalanı omurgada ve bedenin öbür kesimlerinde, ilgili sinirlerde yer alır. Her beyin hücresi 5.000-50.000 sinir hücresiyle bağlantılıdır. Sinir akıları sinir lifleri boyunca taşınır ve beyinde elektrik dalgalarına yol açar. Bu elektrik dalgaları kafa derisinde ölçülebilir.

Elektroensefalografi ya da EEG, beyin dalgaları aktivitesinin elektriksel yöntemle izlenmesini ölçen yöntemdir. Kafatasının üzerine yerleştirilmiş elektrotlarla, beyinde bulunan nöronların ürettiği elektriksel potansiyellerin kaydedilmesi işlemidir (Seymen 2007).

Elektroensefalografla elde edilen kayıt da, elektroensefalogram (EEG) diye adlandırılır. Elektroensefalografi diye adlandırılan bu teknik, 1929'da Alman ruh hekimi Hans Berger tarafından geliştirilmiştir.

EEG beyin hastalıklarını tespit etmede kullanılır. En yaygın olarak sara krizi esnasında beyindeki aktivitenin tipini ve yerini belirlemek için kullanılır. Ayrıca beyin fonksiyonlarıyla ilgili problemleri olan hastaları tespit etmek için de kullanılır. Bu problemler, akıl karışıklığı, koma, tümörler, düşünme ve hafıza ile ilgili uzun süreli zorluklar ya da felce bağlı olarak vücudun belli bölgelerinin güçsüzleşmesini içerebilir (Vrocher ve Lowell 2005).

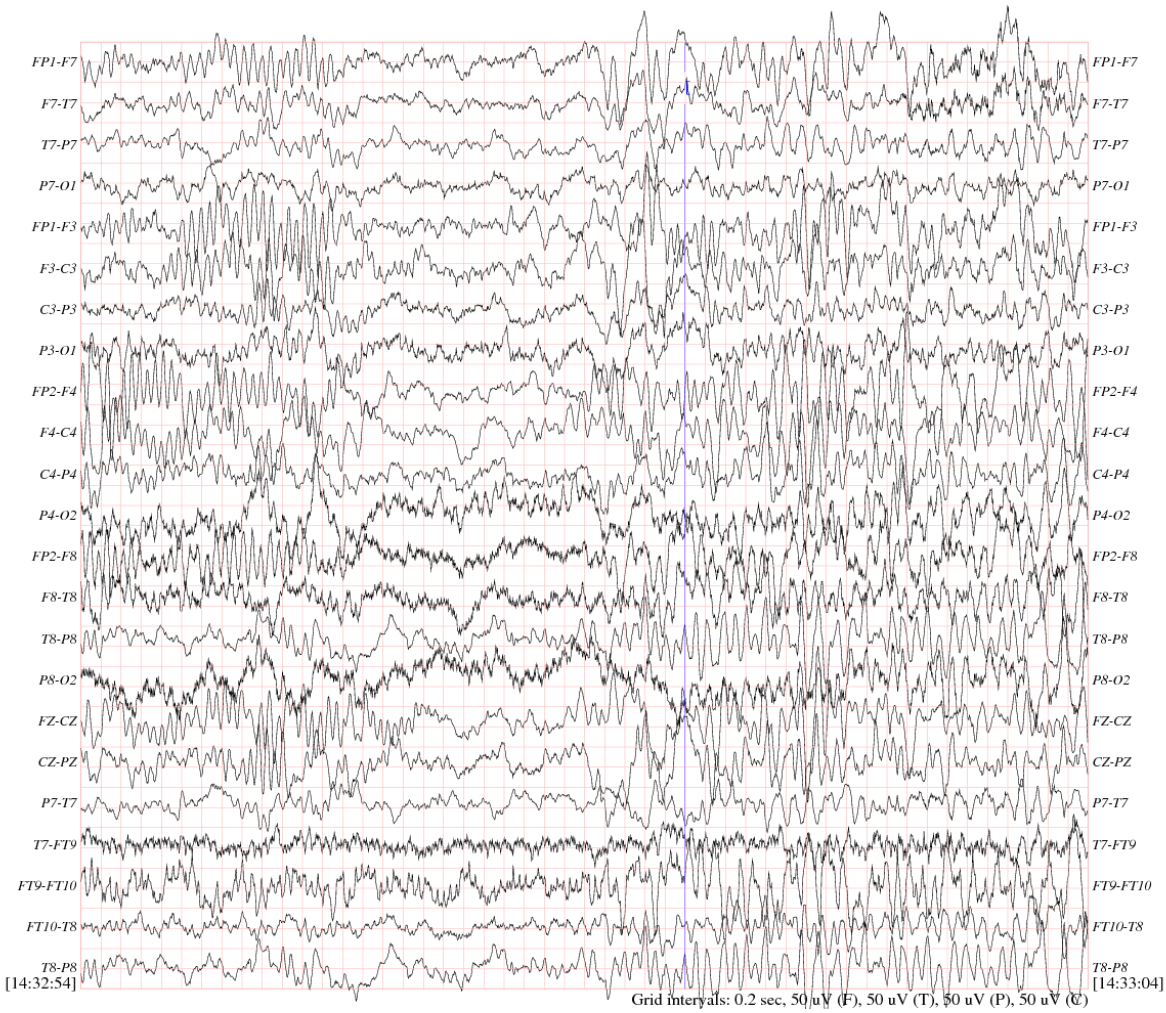
EEG'de çekim, küçük elektrotların saçlı deriye yerleştirilmesiyle, "pasta" denilen iletken bir madde aracılığı ile yapıştırılmasıyla olur. Bu elektrotların ikisi arasındaki elektriksel potansiyel değişiklikleri bilgisayara kayıt edilir. Elde edilen kaydın incelenmesinde, normale oranla sapmalar bulunmasına dayanılarak, beynin birçok çalışma bozukluğu (sara vb.) teşhis edilebilir.

Klinik olarak nöbeti olan her hastada EEG anormalliği gösterilemeyebileceği gibi nöbet veya epilepsisi olmayan kişilerde de EEG anormalliği görülebilir. Nöbeti veya epilepsisi olan hastalarda nöbetler arasında EEG'lerde ortalama % 70 oranında anormallik gösterilebilir (Anonim 2011a).

2.1. CHB-MIT Kafa Derisi EEG Veritabanı

Boston Çocuk Hastanesi'nde toplanan bu veritabanı, inatçı felci olan pediatrik hastalardan alınan EEG kayıtlarından oluşur. Hastalar, felçlerinin karakterize edilmesi ve cerrahi müdahaleye adaylıklarının değerlendirilmesi için, anti-felç ilaç tedavisinin bırakılmasından sonra birkaç gün için gözlemlenmiştir.

23 duruma gruplanan kayıtlar, yaşları 3 ile 22 arasında olan 5 erkek ve yaşları 1,5 ile 19 arasında olan 17 bayandan oluşan 22 hastadan toplanmıştır. (chb21 durumu, aynı bayan hastadan alınan chb01 durumundan 1,5 yıl sonra elde edilmiştir.) SUBJECT-INFO dosyası her hastanın cinsiyetini ve yaşını içerir. (chb24 durumu bu koleksiyona Aralık 2010'da eklenmiştir ve şu anda SUBJECT-INFO'nun içinde yoktur.) Şekil 2.1'de EEG elektrot kanallarından elde edilen grafikler gösterilmiştir.



Şekil 2.1. Kanallardaki EEG grafikleri (Anonim 2011b)

Her durum (chb01, chb02 vs.), tek bir hastadan elde edilen ve sayıları 9 ile 42 arasında değişen sürekli .edf (European Data Format) dosyalarını içerir.

EDF, farklı donanım ve laboratuarlar arasında EEG ve polysomnogram (PSG) verilerinin alışverişi için büyük ölçüde kabul görmüş bir standarttır. Ama diğer araştırmalar

için pek uyumlu değildir (Kemp ve Olivan 2003). Polisomnografi, uyku veya uykusuzluk bozuklukları ile uyku veya uykusuzluk üzerinde etkisi olan diğer bozuklukları tespit etmeye yarayan bir yöntemdir (Amon 2012).

EDF 1992 yılında yayınlanmıştır ve her sinyal için farklı örnekleme hızlarına izin vererek çok kanallı veriyi depolar. İçerisinde bir başlık ve bir veya daha fazla veri kaydı bulunur. Başlık; hasta kimliği, başlama zamanı gibi genel bilgileri ve kalibrasyon, örnekleme hızı, filtreleme gibi her sinyalin teknik özelliklerini, ASCII karakterleriyle kodlanmış olarak içerir. Veri kayıtları ise little-endian 16-bit tamsayıları biçimindeki örnekleri içerir (Anonim 2012a).

Donanım kısıtlamaları ardışık olarak numaralanmış olan ve işaretlerin kaydedilmediği .edf dosyaları arasında boşlukların oluşmasıyla sonuçlanmıştır. Çoğu durumda boşluklar 10 saniye ya da daha azdır, ama bazen daha uzun boşluklar da vardır. Hastaların gizliliğini korumak için, orijinal .edf dosyaları içindeki bütün korunmuş sağlık bilgileri, bu bilgilerin yerini tutan ve burada sağlanmış olan dosyaların içinde bulunan başka bilgilerle değiştirilmiştir. Orijinal .edf dosyalarındaki tarihler başka tarihlerle değiştirilmiştir, ama her duruma ait olan farklı dosyalar arasındaki zaman ilişkileri korunmuştur. chb10 durumuna ait olan .edf dosyaları 2 saat uzunluğunda ve chb04, chb06, chb07, chb09 ve chb23 durumlarına ait olan .edf dosyaları 4 saat uzunluğunda olmasına rağmen, çoğu durumda .edf dosyaları tam 1 saatlik sayısallaştırılmış EEG işaretleri içerir, bazen felçlerin kaydedildiği dosyalar daha kısadır.

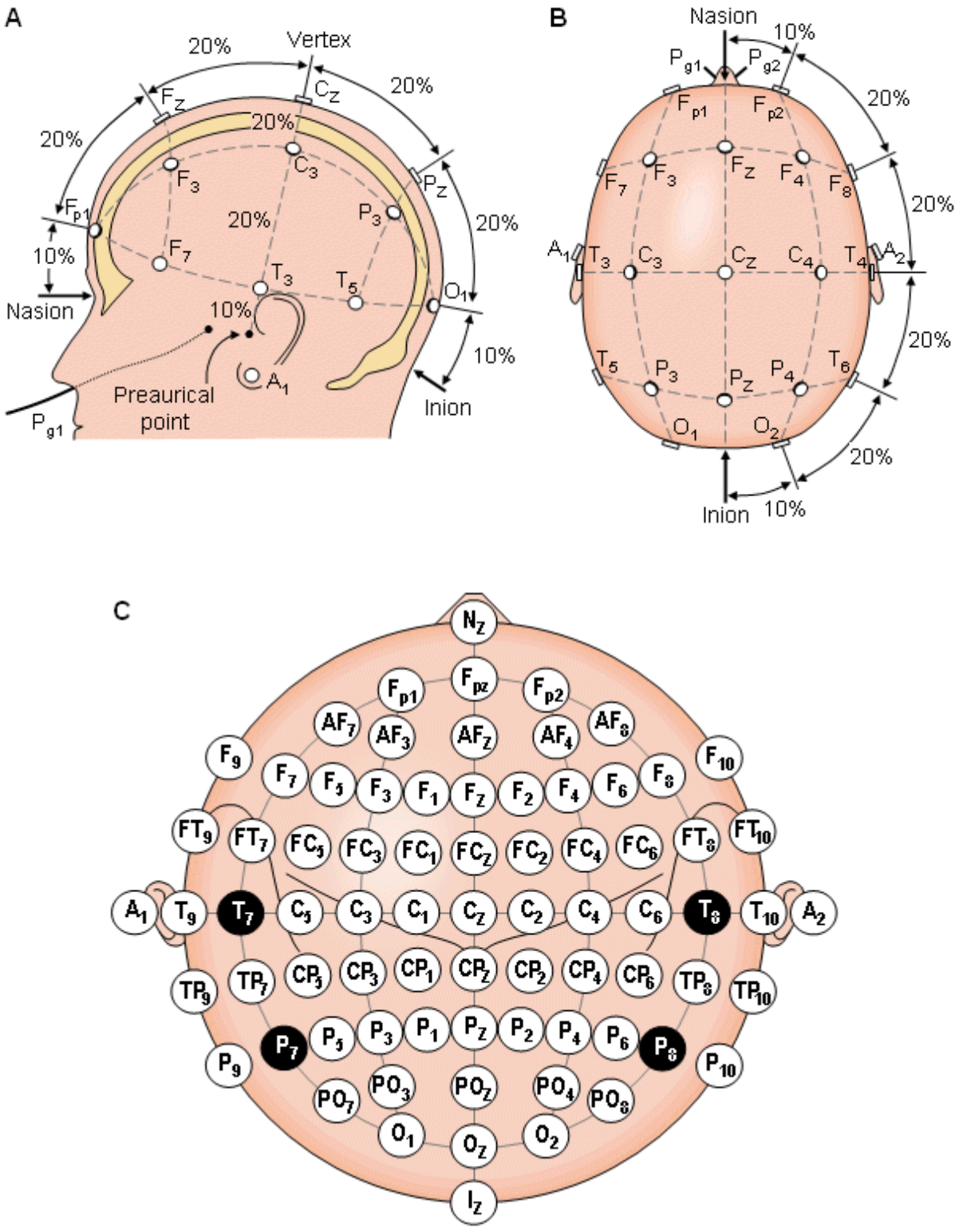
Bütün işaretler 16-bit çözünürlük ile saniyede 256 örnekte örneklenmiştir. Çoğu dosya 23 EEG işareti (birkaç durumda 24 ya da 26 EEG işareti) içerir. EEG elektrod pozisyonları ve terminolojisi için Uluslararası 10-20 Sistemi bu kayıtlar için kullanılmıştır. Birkaç kayıtta diğer işaretler de kaydedilmiştir, örneğin son 36 dosyada chb04 durumuna ait olan bir EKG (elektrokardiyogram) işareti ve son 18 dosyada chb09 durumuna ait olan bir VNS (vagal nerve stimulus) işareti. Bazı durumlarda, 5 taneye kadar yapay işaretler ("-" ile isimlendirilmiştir) EEG işaretlerinin arasına okuması kolay bir görüntü formatı elde etmek için serpiştirilmiştir. Bu eklenen yapay işaretler ihmal edilebilir.

Uluslararası Standart 10-20 sistemi genellikle anlık EEG kaydı için kullanılır. Bu sistemde, 21 adet elektrot, Şekil 2.2.a ve b'de gösterildiği şekilde kafa derisi yüzeyinin üzerine yerleştirilmiştir. Konumlar şu şekildedir: Referans noktaları, gözlerin hizasındaki, burnun üst kısmındaki çukur ve kafatasının arka kısmındaki boynun hemen üzerindeki küçük çıkıntıdır. Bu noktalardan, enine düzlem üzerinden ve kafatasını simetrik olarak ikiye ayıran

düzlem üzerinden kafatasının çevre uzunlukları ölçülür. Elektrot yerleşim konumları bu çevre uzunluklarının %10 ve %20 aralıklarla bölünmesiyle belirlenir. Ayrıca Şekil 2.2.b'de gösterildiği gibi, kendilerine komşu olan noktalara eşit uzaklıkta olarak, her iki tarafa üç elektrot yerleştirilir. Uluslararası 10-20 sisteminin elektrotlarına ek olarak, arada bulunan %10 elektrot konumları da kullanılır. Bu elektrotların yerleşimleri ve adlandırmaları Amerikan Ensefalografi Derneği tarafından standartlaştırılmıştır. Burada 10-20 sistemine kıyasla, dört elektrotun farklı isimleri vardır. Bunlar T7, T8, P7 ve P8 olup Şekil 2.2c'de siyah zemin üzerine beyaz yazıyla gösterilmiştir (Malmivuo ve Plonsey 1995).

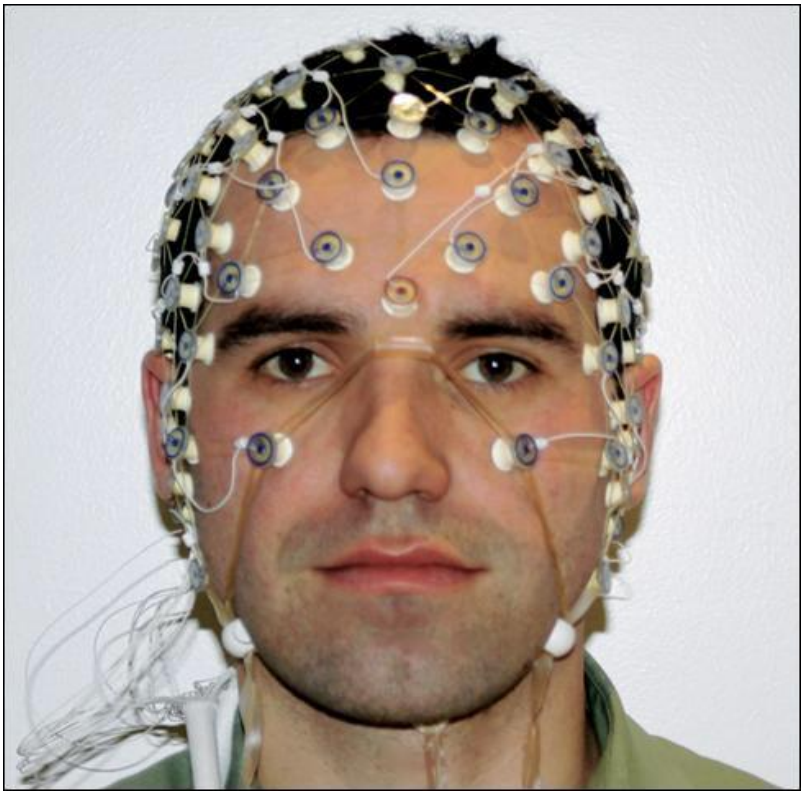
Şekil 2.2'deki F, T, C, P, O harfleri, sırasıyla frontal, temporal, santral, parietal, oksipital lobları ifade eder. Z harfi ise orta çizgi üzerine yerleştirilmiş bir elektrotu ifade eder.

RECORDS dosyası, bu koleksiyonda kapsanan tüm 664 .edf dosyasının bir listesini içerir ve RECORDS-WITH-SEIZURES dosyası bu dosyalardan bir ya da daha fazla felç içeren 129'unu listeler. Toplamda bu kayıtlar 198 felç içerir (182'si orijinal 23 durumluk set içindedir). RECORDS-WITH-SEIZURES dosyasında listelenen dosyaların her birine eşlik eden .seizure notasyon dosyalarında, her felcin başlangıç ([ ) ve bitişinin ( ]) notasyonu yapılmıştır. Ayrıca chbnn-summary.txt isimli dosyalar her kayıt için kullanılan montaj hakkında ve her .edf dosyasının başlangıcından o dosyada içerilen her felcin başlangıç ve bitişine kadar geçen saniye cinsinden zaman hakkında bilgi içerir (Goldberger 2000, Anonim 2011b).

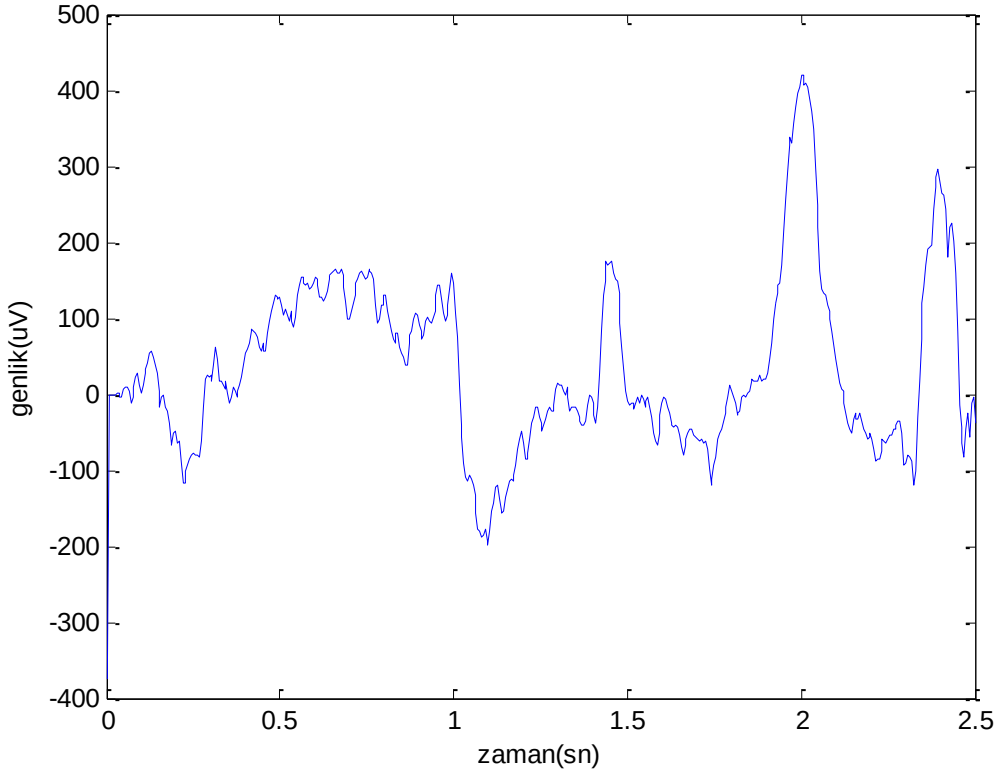


Şekil 2.2. (a) Uluslararası 10-20 Sisteminin soldan görünüşü (b) Uluslararası 10-20 Sisteminin kafanın üzerinden görünüşü (c) Arada bulunan %10 elektrotlarının yerleşim ve adlandırmaları (Malmivuo ve Plonsey 1995)

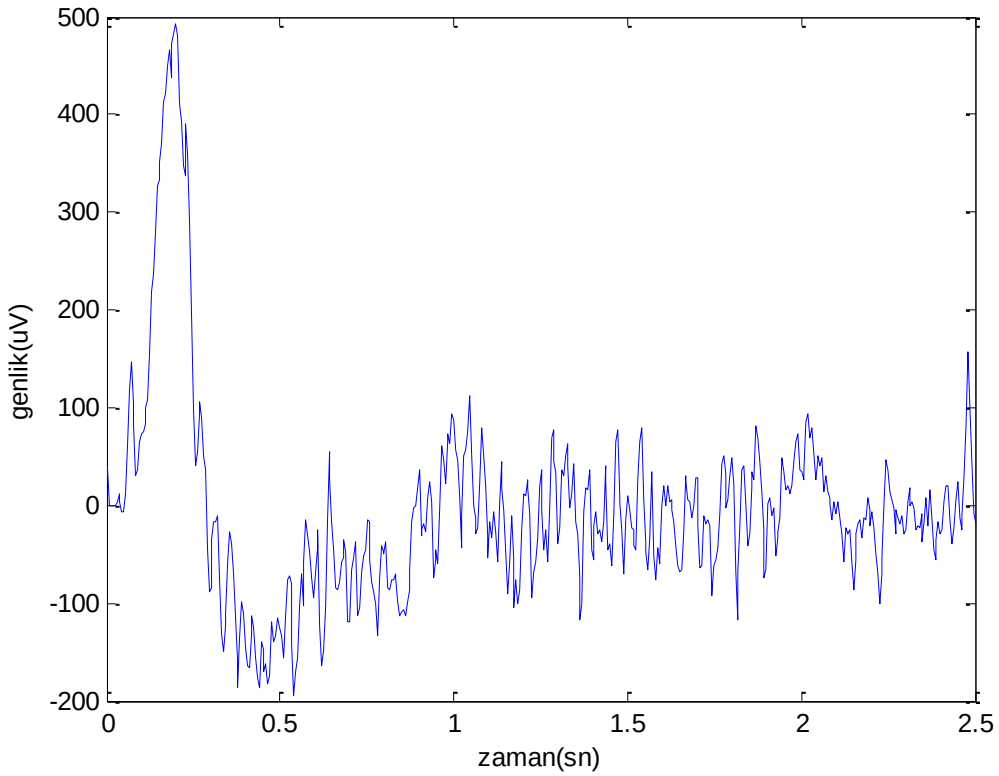
Şekil 2.3'te beyinden EEG ölçümü gösterilmiştir. Ölçümlerle ilgili üç EEG örneği Şekil 2.4, Şekil 2.5 ve Şekil 2.6'da verilmiştir.



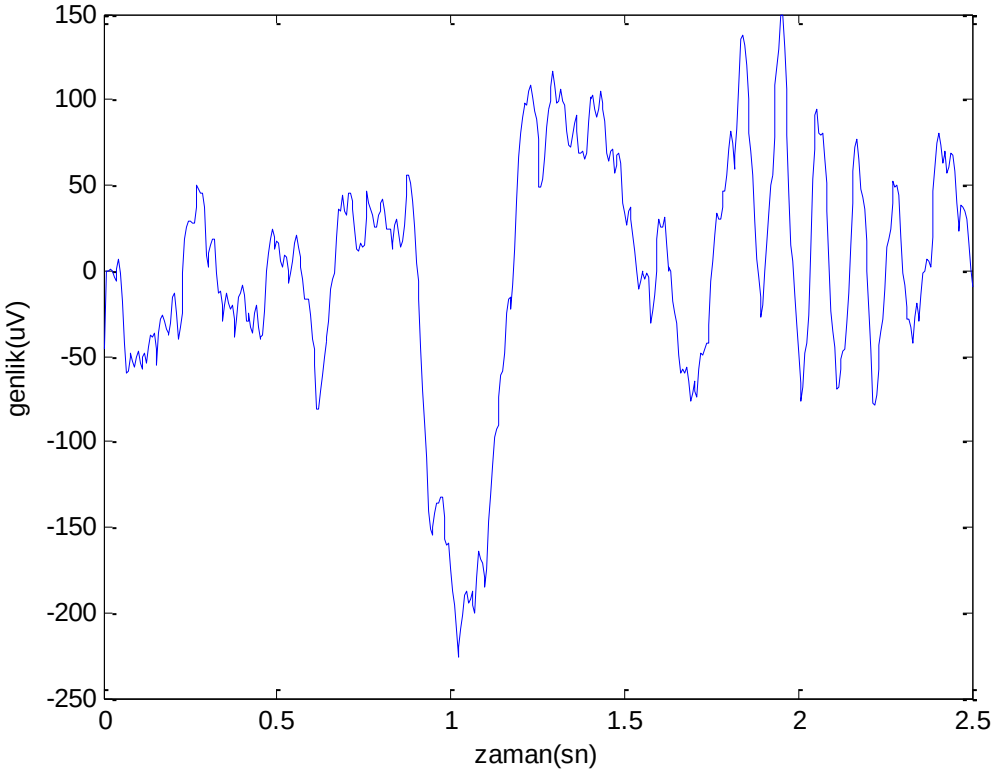
Şekil 2.3. Beyinden EEG Ölçümü (Anonim 2011c)



Şekil 2.4. 11 yaşındaki bir kıza ait EEG kaydının ilk 2,5 sn'si (chb01_01_edfm.mat)



Şekil 2.5. 11 yaşındaki bir erkeğe ait EEG kaydının ilk 2,5 sn'si (chb01_02_edfm.mat)



Şekil 2.6. 14 yaşındaki bir kıza ait EEG kaydının ilk 2,5 sn'si (chb01_03_edfm.mat)

3. ÖZNİTELİK BELİRLEME

3.1. Spektral Entropi

Spektral entropiler, entropi hesaplamalarında olasılıklar olarak, işaretin güç spektrumunun genlik bileşenlerini kullanır. Spektral entropi (SEN) Shannon entropisinin normalize edilmiş biçimidir (Shannon 1948, Fell ve Roshke 1996). Zaman dizilerinin spektral karmaşıklığını niceler. Çeşitli spektral dönüşümler vardır. Bunlardan Fourier dönüşümü, güç spektral yoğunluğunun elde edilebileceği, en iyi bilinen dönüşüm yöntemidir. Güç spektral yoğunluğu, frekansın bir fonksiyonu olarak gücün dağılımını temsil eden bir fonksiyondur. Her frekans için, Fourier dönüşümünden elde edilen güç seviyesi P_f toplanır ve toplam güç $\sum P_f$ hesaplanır. P_f'nin toplam spektral güce göre normalizasyonu bir olasılık yoğunluk fonksiyonunu verir. Her frekansın güç seviyesi toplam güç ile bölünür ($p_f=P_f/P_T$; P_T=toplam güç) ve sonunda $\sum p_f=1$ toplamını verir. Entropi her frekanstaki gücü aynı frekansın logaritması ile çarpıp sonucu -1 ile çarparak hesaplanır ($-p_f$ x $\log(p_f)$). Toplam entropi bütün frekans aralığı üzerinden hesaplanan entropilerin toplamıdır. Böylece spektral entropi

$$\mathrm{SEN} = \sum_{f} p_f \log\left(\frac{1}{p_f}\right) \tag{3.1}$$

ile verilir.

Entropi f frekansındaki olay hakkındaki belirsizliğin bir ölçüsü olarak yorumlanır. Böylece, entropi, sistem karmaşıklığının bir ölçüsü olarak kullanılabilir. Entropi verinin yayılımını ölçer. Geniş, düz olasılık dağılımlı verinin yüksek entropisi vardır. Dar, zirve yapmış dağılımlı verinin düşük entropisi olacaktır. Ayrıca, spektral entropi, Renyi entropileri (REN(α)) denilen entropilerin özel bir durumudur (Kannathal ve ark. 2005).

3.2. Hjorth Parametreleri

Bir zaman dizisi olan $x_1,x_2,\ldots,x_N$ için, Hjorth hareketliliği (HH) ve Hjorth karmaşıklığı (HK) sırasıyla şöyle tanımlanır (Bao ve ark. 2008):

$$\mathrm{HH} = \sqrt{\frac{M2}{TP}} \tag{3.2}$$

ve

$$HK = \sqrt{\frac{M4.TP}{M2.M2}} \tag{3.3}$$

Burada kullanılan ifadeler şu şekildedir:

$$TP = \sum x_i/N \tag{3.4}$$

$$M2 = \sum d_i/N \tag{3.5}$$

$$M4 = \sum (d_i - d_{i-1})^2/N \tag{3.6}$$

$$d_i = x_i - x_{i-1} \tag{3.7}$$

3.3. Tekil Değer Ayrıştırma (TDA) Entropisi

Roberts ve ark. (1998), TDA işleminden yararlanan ölçüme dayalı olan bir entropi tanımlamışlardır. TDA algoritması tarafından üretilen tekil değerler, bir sinyaldeki görünen dinamik bileşenlerin sayısını ya da onun boyutunu gösterebilir.

TDA algoritmasını kullanmak için öncelikle EEG sinyalinin gömülmesi gerçekleştirilmelidir. Bunun gerçekleştirilebilmesinin birçok yolu vardır ve burada Takens gecikmeler yöntemi açıklanacaktır.

Bir $x=(x_1,x_2,x_3 \ldots x_N)$ sinyali alınır. Bu sinyali gömmek için y_n gecikme vektörleri oluşturulur:

$$y(n)=[x(n),x(n+\tau),x(n+2\tau),\ldots,x(n+(d_E-1)\tau)] \tag{3.8}$$

Burada τ zaman gecikmesidir ve d_E gömülme boyutudur. Gömülme uzayı şununla oluşturulur:

$$Y=[y(1),y(2),\ldots,y(N-(d_E-1)\tau)]^T \tag{3.9}$$

Zaman dizisi d_E boyutlu uzayda tanımlanır ve bu uzaydaki y(n) noktalarından oluşur. EEG'yi gömmek için 1 değerinde bir τ ve 20 değerinde bir d_E kullanılır. τ=1 alınması, gecikme vektörleri için her noktanın kullanıldığı ve gömülme işleminde hiç bilgi kaybolmadığı anlamına gelir. d_E'yi seçerken, sinyalin gömülü uzayda tamamen ortaya çıkmasına izin vermeye yetecek kadar yüksek bir boyut seçmeye dikkat edilmelidir.

Gömülmenin boyutuna bağlı olarak, tekil spektrum olarak bilinen bir dizi tekil değerler üretilerek, gömülü matris üzerinde TDA işlemi gerçekleştirilir. Önemli tekil değerlerin sayısı, dinamik bileşenlerin sayısıyla direkt olarak ilgilidir.

Bundan dolayı, sinyalin karmaşıklığı için bir ölçü belirlemek üzere, tekil değerlerin entropisi bulunur. Öncelikle, tekil değerler, her biri bütün tekil değerlerin toplamına bölünerek normalize edilir ve sonra TDA entropisi şu şekilde verilir:

$$\mathrm{Hsvd} = -\sum_{i=1}^{N} \bar{\sigma_i} \log_2 \bar{\sigma_i} \tag{3.10}$$

Burada, N tekil değerlerin sayısıdır ve $\overline{\sigma_1} \ldots \overline{\sigma_N}$ ise $\bar{\sigma_i} = \sigma_i / \sum_j \sigma_j$ ile normalize edilmiş olan tekil değerlerdir (Faul ve ark. 2005).

3.4. Fisher Bilgisi

TDA entropisinin kullanımı, karmaşıklık ölçüsünün, incelenen sinyalin gücü tarafından büyük ölçüde etkilenmesi anlamına gelmektedir. Bundan dolayı, tekil spektrumun şeklindeki değişiklikleri vurgulayan ve bu nedenle kriz içermeyen EEG'den kriz içeren EEG'ye değişimleri göstermesi gereken Fisher bilgisi ile daha uygun bir ölçü verilebilir. Normalize edilmiş tekil spektrum $\overline{\sigma_1} \ldots \bar{\sigma_i}$ için Fisher bilgisi şu şekilde verilir (Faul ve ark. 2005):

$$\mathrm{I} = \sum_{i=1}^{m} \frac{(\overline{\sigma_{i+1}} - \bar{\sigma_i})^2}{\bar{\sigma_i}} \tag{3.11}$$

3.5. Yaklaşık Entropi

Yaklaşık Entropi (YE), bir anlık kalp hızı zaman dizisi HR(i) gibi bir zaman dizisindeki dalgalanmaların tahmin edilebilirliğini ölçen bir düzenlilik istatistiğidir. Sezgisel olarak, bir zaman dizisindeki dalgalanmanın tekrarlayıcı örüntülerinin varlığının, onu, böyle örüntülerin olmadığı bir zaman dizisinden daha tahmin edilebilir kıldığı düşünülebilir.

YE, gözlemlerin benzer örüntülerinin, ilave benzer gözlemler tarafından takip edilmeme olasılığını yansıtır. Birçok tekrarlayıcı örüntü içeren bir zaman dizisi göreceli olarak küçük bir YE'ye sahiptir. Daha az tahmin edilebilir yani daha fazla karmaşık bir işlemin daha yüksek bir YE'si vardır.

N adet anlık kalp hızı ölçümü HR(1),HR(2),…,HR(N)'den meydana gelen bir S_N dizisi verildiğinde, dizinin yaklaşık entropisi YE(S_N,m,r)'yi hesaplamak üzere, iki giriş parametresi olan m ve r için değerler seçilmelidir. m parametresi örüntü uzunluğunu, r parametresi ise benzerlik kriterini tanımlar. S_N içindeki i ölçümünde başlayan kalp hızı ölçümlerinin bir alt dizisi ya da örüntüsü $p_m(i)$ vektörü ile gösterilir. Örüntülerdeki eşleşen ölçüm çiftleri arasındaki uzaklık r'den küçükse, yani $0 \leq k < m$ için $|HR(i+k)-HR(j+k)|<r$ ise $p_m(i)$ ve $p_m(j)$ örüntüleri benzerdir.

S_N içindeki m uzunluğundaki bütün örüntülerin kümesi P_m ele alındığında şu tanımlanabilir:

$$C_{im}(r) = \frac{n_{im}(r)}{N - M + 1} \tag{3.12}$$

Burada, benzerlik kriteri r verildiğinde, $n_{im}(r)$, P_m'deki $p_m(i)$'ye benzeyen örüntülerin sayısıdır. $C_{im}(r)$ büyüklüğü, i aralığında başlayan, aynı uzunluktaki örüntülere benzeyen, m uzunluğundaki örüntülerin oranıdır. P_m'deki her örüntü için $C_{im}(r)$ hesaplanır ve bu $C_{im}(r)$ değerlerinin ortalaması olarak $C_m(r)$ tanımlanır. $C_m(r)$ büyüklüğü, S_N'deki m uzunluğundaki tekrarlayıcı örüntülerin yaygınlığını ifade eder. m uzunluğundaki örüntüler ve r benzerlik kriteri için, S_N'nin yaklaşık entropisi şu şekilde tanımlanır:

$$YE(SN, m, r) = \ln\left[\frac{C_m(r)}{C_{m+1}(r)}\right] \tag{3.13}$$

Bu, m+1 uzunluğundakilerle karşılaştırılan m uzunluğundaki tekrarlayıcı örüntülerin göreceli yaygınlığının doğal logaritmasıdır.

Böylece, bir kalp hızı zaman dizisinde benzer örüntüler bulunursa, YE, her örüntüden sonraki aralığın farklı olmasının, yani örüntülerin benzerliğinin sadece tesadüf olmasının ve tahmin edilebilir değerlerin eksik olmasının logaritmik olasılığını hesaplar. Daha küçük YE değerleri, ölçümlerin benzer örüntülerinin, ilave benzer ölçümler tarafından takip edilmesi için daha büyük bir olasılık belirtir. Eğer zaman dizisi çok düzensizse, benzer örüntülerin meydana gelmesi takip eden ölçümler için tahmin edilebilir olmayacaktır ve YE göreceli olarak büyük olacaktır.

YE'nin önemli zayıflıkları olduğunu da ifade etmek gerekir. Bunlar, dizi uzunluğuna yüksek derecede bağlı ve kendi içindeki tutarlılığının zayıf olmasıdır (Moody 1997).

3.6. Hurst Katsayısı

Hurst katsayısı, bir zaman dizisinin öz benzerliği için boyutsuz bir hesaplayıcıdır. Hurst katsayısını tanımlamak için çeşitli yollar vardır. Harold Hurst'ün kendisi tarafından geliştirilen en eski tanım şu şekildedir:

$$E\left[\frac{R(n)}{S(n)}\right] = Cn^H, n \to \infty \tag{3.14}$$

Eşitliğin sol tarafı, yeniden ölçeklenmiş aralığın beklenen değeri olarak da bilinir. i=1,2,…,n olmak üzere bir X_i zaman dizisi üzerinde R(n) şöyle tanımlanır:

$$R(n)=\max(X_i,i=1,2,\ldots,n)-\min((X_i,i=1,2,\ldots,n) \tag{3.15}$$

S(n) standart sapma ve C bir rastgele sabittir.

N, zaman dizisinin uzunluğu olmak üzere, i=1,2,…,$\frac{N}{5}$ için şu hesaplanır:

$$a_i = E\left[\frac{R(i)}{S(i)}\right] \tag{3.16}$$

Her i boyutu için, eşitliğin sağ tarafı, zaman dizisi i boyutunda yığınlara ayrılarak hesaplanır. Her yığında, o yığın için R(i) ve S(i) hesaplanır. Sonra, bütün zaman dizisi için beklenti değeri $E\left[\frac{R(i)}{S(i)}\right]$ bütün yığınların alt sonuçları üzerinden ortalama alınarak hesaplanır. Bu işlem şu eşitliği verir:

$$E[a_i] = Ci^H \tag{3.17}$$

ve bu nedenle şu elde edilir:

$$\log(E[a_i]) = \log(C) + H\log(i) \tag{3.18}$$

İkiden fazla farklı i kullanarak, yeterli miktarda giriş verisi olduğunda, bu eşitlikler genellikle farklı şekillerde belirlenir ve bir en küçük kareler uyumu kullanarak çözülebilir. Uyumun eğimi, H (Hurst katsayısı) için kestirilen değer olacaktır. Bu değer, sabit ofset ya da bu durum için önemsiz olan, log(C) için kestirilen değerdir. Ne yazık ki, bu prosedür genellikle zayıf yakınsama ve kutup gösterir (Racine 2011).

3.7. Örnek Entropisi

Örnek Entropisi (ÖE), özeşleşmeler hariç olmak üzere, m adet nokta için benzer olan iki dizinin sıradaki diğer noktada benzer kalma olasılığının negatif doğal logaritmasıdır. Böylece, daha düşük bir ÖE değeri, zaman dizisinde daha fazla düzensizlik gösterir.

Bir {x(n)}={x(1),x(2),…,x(N)} zaman dizisinden N adet veri noktası verildiğinde, ÖE'yi tanımlamak için şu adımlar takip edilir (Vrhovec 2009):

1) $1 \leq i \leq N-m+1$ için X(i)=[x(i),x(i+1),…,x(i+m-1)] ile tanımlanan N-m+1 adet X(1),…,X(N-m+1) vektörleri oluşturulur. Bu vektörler, i'nci noktadan başlayarak m adet ardışık sinyal değerini temsil eder.

2) Karşılıklı skaler bileşenleri arasındaki maksimum mutlak fark olarak, X(i) ve X(j) arasındaki uzaklık d=[X(i),X(j)] hesaplanır:

$$d[X(i), X(j)] = \max_{k = 1,2,\dots,m} (|x(i+k) - x(j+k)|) \tag{3.19}$$

3) Verilen bir X(i) için, X(i) ve X(j) arasındaki uzaklık r.SD'ye eşit veya ondan küçük olacak şekilde ve $1 \leq j \leq N-m$, $i \neq j$ olacak şekilde j'lerin sayısı belirlenir ve $B_r^m(i)$ hesaplanır:

$$B_r^m(i) = \frac{1}{N-m-1} \sum_{j=1, j \neq i}^{N-m} \Theta(r.SD - d[X(i), X(j)]) \tag{3.20}$$

Burada Θ, Heaviside fonksiyonu ya da birim basamak fonksiyonudur ($\Theta(z \geq 0)=1$ ve $\Theta(z<0)=0$). SD, x(n) sinyalinin standart sapmasıdır. r bir tolerans penceresidir.

4) B_r^m şu şekilde hesaplanır:

$$B_r^m = \frac{1}{N-m} \sum_{i=1}^{N-m} B_r^m(i) \tag{3.21}$$

5) Boyut m+1'e çıkarılır ve $A_r^m(i)$ hesaplanır:

$$A_r^m(i) = \frac{1}{N-m+1} \sum_{i=1, j \neq i}^{N-m} \Theta(r.SD - d[X(i), X(j)]) \tag{3.22}$$

6) A_r^m şu şekilde hesaplanır:

$$A_r^m = \frac{1}{N-m} \sum_{i=1}^{N-m} A_r^m(i) \tag{3.23}$$

7) ÖE şu şekilde hesaplanır:

$$ÖE(m, r, n) = -\ln \frac{A_r^m}{B_r^m} \tag{3.24}$$

$A_r^m(i)$ ve $B_r^m(i)$, benzer sinyal parçalarının benzerlik ölçüsüdür. A_r^m ve B_r^m ise $A_r^m(i)$ ve $B_r^m(i)$'nin ortalamasıdır.

3.8. Fraktal Boyut (FB)

Fraktal; matematikte, çoğunlukla kendine benzeme özelliği gösteren karmaşık geometrik şekillerin ortak adıdır. Fraktallar, klasik, yani Öklitsel geometrideki kare, daire, küre gibi basit şekillerden çok farklıdır. Bunlar, doğadaki, Öklitsel geometri aracılığıyla tanımlanamayacak pek çok uzamsal açıdan düzensiz olguyu ve düzensiz biçimi tanımlama yeteneğine sahiptir. Fraktal terimi parçalanmış ya da kırılmış anlamına gelen Latince "fractus" sözcüğünden türetilmiştir. İlk olarak 1975'te Polonya asıllı matematikçi Beneoit B.

Mandelbrot tarafından ortaya atılan fraktal kavramı, yalnızca matematik değil fiziksel kimya, fizyoloji ve akışkanlar mekaniği gibi değişik alanlar üzerinde önemli etkiler yaratan yeni bir geometri sisteminin doğmasına yol açmıştır.

Tüm fraktallar kendine benzer ya da en azından tümüyle kendine benzer olmamakla birlikte, çoğu bu özelliği taşır. Kendine benzer bir cisimde cismi oluşturan parçalar ya da bileşenler cismin bütününe benzer. Düzensiz ayrıntılar ya da desenler giderek küçülen ölçeklerde yinelenir ve tümüyle soyut nesnelerde sonsuza değin sürebilir; öyle ki, her parçanın her bir parçası büyütüldüğünde, gene cismin bütününe benzer. Bu fraktal olgusu, kar tanesi ve ağaç kabuğunda kolayca gözlenebilir. Bu tip tüm doğal fraktallar ile matematiksel olarak kendine benzer olan bazıları, stokastik, yani rastgeledir; bu nedenle ancak istatistiksel olarak ölçeklenirler.

Fraktalların bir başka önemli özelliği de, fraktal boyut olarak adlandırılan bir matematiksel parametredir. Bu, cisim ne kadar büyütülürse büyütülsün ya da bakış açısı ne kadar değiştirilirse değiştirilsin, hep aynı kalan fraktalların bir özelliğidir. Öklitsel boyutun tersine fraktal boyut, genellikle tam sayı olmayan bir sayıyla, yani bir kesir ile ifade edilir. Fraktal boyut, bir fraktal eğri yardımıyla anlaşılabilir.

Fraktal boyut kavramı, ölçeklemeye ve boyuta geleneksel olmayan bir şekilde bakmaya dayalıdır. Geometrinin geleneksel kavramlarına göre, şekiller, tahmin edilebilir bir şekilde, içinde yer aldıkları uzay hakkındaki sezgisel ve tanıdık fikirlere göre ölçeklenir. Örneğin, bir çizgiyi önce belli uzunluktaki bir ölçme çubuğu ile ölçüp, sonra o çubuğun üçte biri uzunluğunda bir ölçme çubuğu ile ölçersek, ikinci çubukla yapılan ölçüm birinci çubukla yapılan ölçüme göre üç kat daha büyük bir uzunluğu verir. Bu durum iki boyutlulukta da geçerlidir. Bir küpün içine kenar uzunluğu bu küpün üçte biri olan küplerden dokuz tane sığar. Bu tür ölçekleme ilişkileri, matematiksel olarak, şu genel ölçekleme kuralı ile ifade edilir:

$$N \propto \in^{-B} \tag{3.25}$$

Burada kullanılan ifadeler şu şekildedir:

N: Yeni çubukların sayısı

Є: Ölçekleme faktörü

B: Fraktal boyut

Bu ölçekleme kuralı, geometri ve boyut hakkındaki geleneksel kuralların tipik bir örneğidir. Doğrular için, yukarıdaki örnekteki gibi, Є=1/3 olduğunda N=3 olduğu durumda B=1 olur, Є=1/3 olduğunda N=9 olduğu durumda B=2 olur.

Aynı kural daha az sezgisel olarak fraktal geometri için de geçerlidir. Ayrıntılarına girmek için, başlangıçta 1 birim uzunluğunda olacak şekilde ölçülen bir fraktal doğru, eskisinin üçte biri ile ölçeklenen yeni bir çubuk kullanılarak yeniden ölçüldüğünde, beklenildiği gibi 3 değil 4 kat ölçeklenmiş çubuk uzunluğunda olabilir. Bu durumda Є=1/3 olduğunda N=4 olur ve B'nin değeri (3.25) denklemini yeniden düzenleyerek şu şekilde bulunabilir:

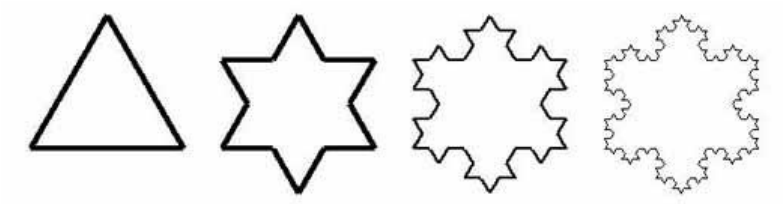
$$\log_{\epsilon} N = -B = \frac{\log N}{\log \epsilon} \qquad (3.26)$$

Yani, Є=1/3 olduğunda N=4 olarak tanımlanan bir fraktal için, B=1,2619 olur. Bu tamsayı olmayan bir boyuttur ve fraktalın içinde bulunduğu uzayınkine eşit olmayan bir boyuta sahiptir. Bu örnekte kullanılan ölçekleme, Koch eğrisi ve kar tanesininkiyle aynı ölçeklemedir (Anonim 2012b).

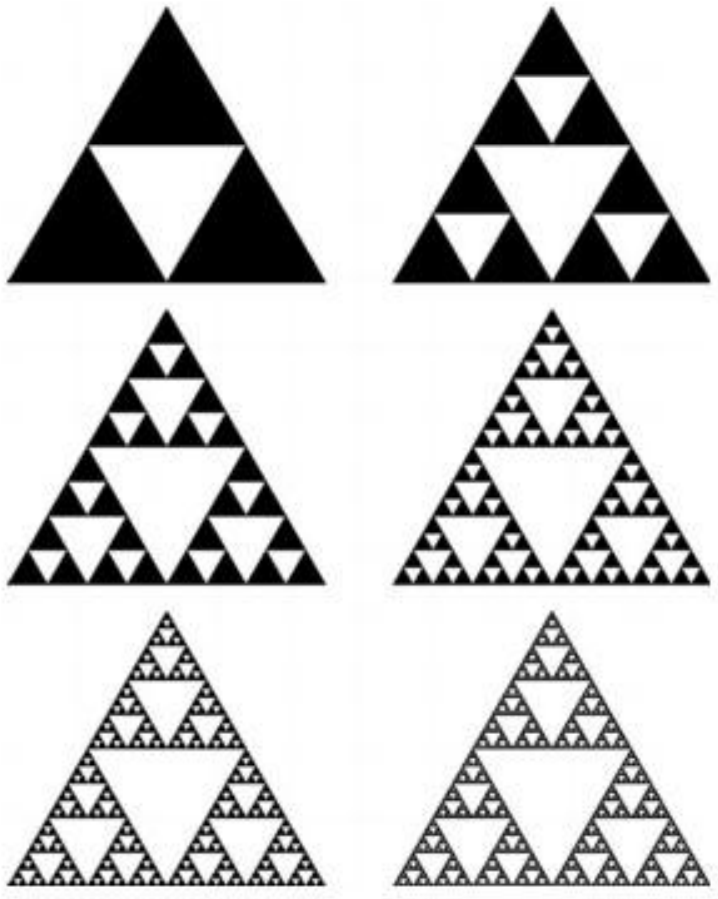
Oluşturulmasının her aşamasında bu tip bir eğrinin çevre uzunluğu 4/3 oranında büyür. Fraktal boyut (B) 4'e eşit olabilmesi için alınması gereken kuvvetini gösterir; yani 3^B =4 olduğundan fraktal eğriyi niteleyen boyut B=log4/log3 ya da kabaca 1,26'dır. Fraktal boyut, Öklitsel olmayan belirli bir biçimin karmaşıklığını ve şekil nüanslarını açığa çıkarır (Anonim 2012c).

Şekil 3.1'de yaklaşık 1,2619 değerinde bir fraktal boyuta sahip Koch kar tanesinin ilk dört tekrarı gösterilmiştir. Şekil 3.2'de ise kendisinin üç adet yarı boyutundaki kopyasından oluşan, bu nedenle bir ile iki arasında bir fraktal boyuta sahip olan, Sierpinski'nin contası olarak bilinen üçgen gösterilmiştir. Bu şeklin kenar uzunluğu iki katına çıkınca, orijinal şeklin üç kopyasından oluşan bir şekil meydana gelmektedir. Halbuki geleneksel geometri kavramlarına göre, bir boyutlu uzayda bir şeklin kenar uzunluğunu iki katına çıkarırsak, 2^1'den orijinal şeklin iki kopyasını elde etmiş oluruz ve iki boyutlu uzayda bir şeklin kenar uzunluğunu iki katına çıkarırsak, 2^2'den orijinal şeklin dört kopyasını elde etmiş oluruz. O halde Sierpinski'nin contası olarak bilinen üçgen için boyut, bir ile iki arasında bir fraktal boyut olmalıdır. 2^B=3 eşitliğini sağlayan B sayısı, bu fraktal boyutun değerini verir ve B=$\log_2 3$=1,5849 olarak bulunur.

Kendine benzerlik ve tamsayı olmayan boyutlu kavramlarıyla birlikte fraktal geometri, istatistiksel mekanikte, özellikle görünürde rastgele özelliklerden oluşan fiziksel sistemlerin incelenmesinde giderek daha yaygın olarak kullanılmaya başlanmıştır. Örneğin, gökada kümelerinin evrendeki dağılımının saptanmasında ve akışkan burgaçlanmalarına ilişkin problemlerin çözülmesinde fraktal benzetimlerden yararlanılmaktadır. Fraktal geometri bilgisayar grafiklerinde de yararlı olmaktadır. Fraktal algoritma ise, engebeli dağlık araziler ya da ağaçların karışık dal sistemleri gibi karmaşık, çok düzensiz doğal cisimlerin gerçektekine benzer görüntülerinin oluşturulabilmesini olanaklı kılmıştır (Anonim 2012b).



Şekil 3.1. Koch kar tanesinin ilk dört tekrarı (Haas 2012)



Şekil 3.2. Sierpinski'nin contası olarak bilinen üçgen (Banchoff 1990)

Biyomedikal işaretlerin analizinde fraktal boyutu hesaplamak üzere, Esteller ve ark. (2001), Higuchi algoritması, Katz algoritması ve Petrosian algoritmasını kullanıp karşılaştırmıştır. Benzer bir diğer çalışmada ise Polychronaki ve ark. (2010), Katz algoritması, Higuchi algoritması ve en yakın k-komşu algoritmasını değerlendirip kıyaslamıştır. Goh ve ark. (2005)'in çalışmasında da Katz Algoritması, Petrosian Algoritması ve Sevcik Algoritması incelenerek karşılaştırılmıştır.

3.8.1. Petrosian fraktal boyutu

Petrosian fraktal boyutu şu şekilde ifade edilir:

$$PFB = \frac{\log_{10} N}{\log_{10} N + \log_{10}\left(\frac{N}{N+0.4N_{\delta}}\right)} \tag{3.27}$$

Burada N dizinin uzunluğu, N_{δ} ise işaret türevindeki işaret değişimlerinin sayısıdır (Petrosian 1995). Petrosian fraktal boyutu her sınıf içinde yüksek derecede yoğunlaşır ve her bir sınıfın verileri arasında hiç örtüşme olmaz. Bu nedenle, Petrosian fraktal boyutu kullanılarak, bütün sınıflar açıkça birbirinden ayırt edilebilir (Bao ve ark. 2008).

3.8.2. Katz fraktal boyutu

$x_i < x_{i+1}$ ve i=1,2,…,N (N: nokta sayısı) olmak üzere, dalga şekilleri, $\overrightarrow{p_i} = (x_i, y_i)$ noktalarının toplamları olarak görüntülenebilir ve bunlar yalnızca x yönünde ileri doğru ilerleyen düzlemsel eğrilerin özel durumlarıdır. Katz (1988), fraktal boyut hesaplama yöntemini bu tür eğrilerin uzunluk ölçümüne dayandırmıştır. Fraktal boyutu hesaplamak için uzayın ayrıklaştırılması gerektiğini dikkate alarak, Mandelbrot'un orijinal makalesini (Mandelbrot 1982) bir ölçü birimi tanımıyla birleştirmiştir. Mandelbrot (1982)'a göre düzlemsel bir eğrinin fraktal boyutu

$$FB = \log(L) / \log(d) \tag{3.28}$$

ile verilir. Burada L eğrinin toplam uzunluğu, d ise onun çapıdır. Dalga şekilleri için L toplam uzunluğu ardışık noktalar arasındaki uzaklıkların toplamıdır.

$$L = \sum_{i=1}^{N} \|\overrightarrow{p_{i+1}} - \overrightarrow{p_i}\| \tag{3.29}$$

Burada $\|.\|$ Öklit uzaklığıdır. d çapı, dalga şeklinin başlangıç noktasıyla herhangi bir noktası arasındaki en uzak mesafe olarak düşünülebilir.

$$d = \max\|\overrightarrow{p_i} - \overrightarrow{p_1}\| \tag{3.30}$$

Katz (1988)'a göre, (3.28) eşitliği ardışık noktalar arasındaki ortalama uzaklık olan bir ölçü birimi olarak dalga şeklinin ortalama adımı $\underline{a}$'dan faydalanılarak düzeltilmelidir. $\underline{a}$'yı kullanarak (3.28) eşitliği şu hale dönüşür:

$$FB = \frac{\log\left(\frac{L}{\underline{a}}\right)}{\log\left(\frac{d}{\underline{a}}\right)} \tag{3.31}$$

n'yi eğrideki adımların sayısı (nokta sayısı N'nin bir eksiği) olarak tanımlarsak $n = L/\underline{a}$ olur. n'yi (3.31) eşitliğinde yerine koyarsak, Katz yöntemine göre fraktal boyut şu şekilde ifade edilir (Polychronaki ve ark. 2010):

$$FB = \frac{\log(n)}{\log(n) + \log\left(\frac{d}{L}\right)} \tag{3.42}$$

3.8.3. Sevcik fraktal boyutu

Sevcik algoritması, bir N değerleri kümesinden FB'yi ve 0 ile t_{max} arasındaki zaman aralığındaki bir dalga biçiminden örneklenen y_i'yi hesaplar. Dalga biçimi, kendisini bir birim kareye dönüştüren bir çift doğrusal dönüşüme tabi tutulur. Karenin normalize edilmiş apsisi olan x_i^* ve karenin normalize edilmiş ordinatı olan y_i^* şu şekilde ifade edilirler:

$$x_i^* = \frac{x_i}{x_{max}} \tag{3.36}$$

$$y_i^* = \frac{y_i - y_{min}}{y_{max} - y_{min}} \tag{3.37}$$

Burada x_{max} maksimum x_i, y_{min} minimum y_i, y_{max} maksimum y_i'dir. Dalga biçiminin FB'si şu şekilde bulunur:

$$FB_{Sevcik} = 1 + \frac{\ln(L) + \ln(2)}{\ln(2N')} \tag{3.38}$$

Burada L, birim karedeki zaman dizisinin uzunluğudur ve N'=N-1 dir (Sevcik 1998, Goh ve ark. 2005).

3.9. Dalgacık Analizi

Birçok veri sinyali, önemli sayılabilecek durağansızlıklar veya eğim, ansızın değişim, kırılma ve olayların başlangıç ve bitişleri gibi geçici özellikler içerebilir. Bu beklenmedik özellikler ve durağansızlıklar, özellikle EEG gibi veri sinyalinin en önemli kısımları olabilmektedir (Coşkun ve İstanbullu 2012).

Dalgacık dönüşümü Fourier dönüşümünün durağan olmayan sinyallerdeki eksiklerini gidermek için geliştirilmiş farklı bir dönüşüm yöntemidir. Bu analiz yöntemi gürültüye karşı daha az hassasiyet göstermekte ve durağan olamayan sinyallere rahatlıkla uygulanabilmektedir (Erdoğmuş ve Pekçakar 2009).

Dalgacık dönüşümü, Fourier dönüşümünden farklı olarak düşük frekanslar için geniş, yüksek frekanslar için dar olacak şekilde değişen pencere boyutlarına sahiptir. Böylece, bütün frekans aralıklarında en iyi zaman-frekans çözünürlüğü sağlanabilmektedir. Dalgacık dönüşümü farklı frekanslarda durağan olmayan güce sahip zaman serisi sinyallerinin analizinde kullanılabilir. Bunun yanı sıra, Fourier dönüşümünün aksine sadece zaman-frekans bölgesini değil, aynı zamanda zaman-ölçek bölgesini kullanır (Coşkun ve İstanbullu 2012).

Aşağıdaki koşulları sağlayan kompleks değerli bir ψ fonksiyonunu ele alalım:

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty \tag{3.39}$$

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \tag{3.40}$$

Burada Ψ, ψ'nin Fourier dönüşümüdür. (3.39) eşitliği ψ fonksiyonunun sonlu enerjisini gösterir. (3.40) eşitliği kabul edilebilirlik koşuludur ve eğer Ψ(ω) düzgünse Ψ(0)=0 olduğunu gösterir. ψ fonksiyonu ana dalgacıktır (Akansu ve Haddad 1992, Lee ve Yamamoto 1994).

3.9.1. Sürekli dalgacık dönüşümü

ψ fonksiyonu (3.39) ve (3.40) koşullarını sağlarsa, bir s(t) gerçek sinyalinin ψ(t) dalgacık fonksiyonuna göre dalgacık dönüşümü şu şekilde ifade edilir:

$$S(b,a) = \int_{-\infty}^{\infty} \psi'\left(\frac{t-b}{a}\right) s(t)dt \tag{3.41}$$

Burada ψ', ψ'nin karmaşık eşleniğidir ve açık (b,a) yarı düzlemi (b∈**R**, a>0) üzerinde tanımlanmıştır. b parametresi zaman kaymasıdır ve a parametresi analiz eden dalgacığın ölçeğidir.

a ile yeniden ölçeklemek ve b ile kaydırmak suretiyle $\psi_{a,b}(t)$ şu şekilde tanımlanabilir:

$$\psi_{a,b}(t) = a^{-1/2}\psi\left(\frac{t-b}{a}\right) \tag{3.42}$$

(3.41) eşitliği, bir skaler ya da s(t) gerçek sinyali ile $\psi_{a,b}(t)$ fonksiyonunun iç çarpımı olarak yazılabilir:

$$S(b,a) = \int_{-\infty}^{\infty} \psi'_{a,b}(t)s(t)dt \tag{3.43}$$

ψ(t) fonksiyonu (3.40) eşitliğindeki kabul edilebilirlik koşulunu sağladığında, s(t) orijinal sinyali şu ters formül ile S(b,a) dalgacık dönüşümünden elde edilebilir (Lee ve Yamamoto 1994):

$$s(t) = \frac{1}{c_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(b,a)\,\psi_{a,b}(t)\,\frac{dadb}{a^2} \tag{3.44}$$

3.9.2. Ayrık dalgacık dönüşümü

Ayrık etki alanında, ölçek ve kaydırma parametreleri $a = a_0^m$ ve $b = nb_0$ olarak ayrıklaştırılır ve analiz eden dalgacıklar şu şekilde ayrıklaştırılır:

$$\psi_{m,n}(t) = a_0^{-m/2}\psi\left(\frac{t - nb_0}{a_0^m}\right) \tag{3.45}$$

Burada m ve n tamsayı değerleridir. Ayrık dalgacık dönüşümü ve onun ters dönüşümü şu şekilde ifade edilir:

$$S_{m,n} = \int_{-\infty}^{\infty} \psi'_{m,n}(t)s(t)dt \tag{3.46}$$

$$s(t) = k_\psi \sum_m \sum_n S_{m,n}\psi_{m,n}(t) \tag{3.47}$$

Burada k_ψ normalizasyon için sabit bir değerdir.

$\psi_{m,n}(t)$ fonksiyonu, ölçek-zaman düzlemi üzerindeki örnekleme noktalarını verir. b ekseni olan zaman yönünde doğrusal örnekleme, fakat a ekseni olan ölçek yönünde logaritmik örnekleme yapılır.

En yaygın durum a_0'ın şu şekilde seçildiği durumdur:

$$a_0 = 2^{1/v} \tag{3.48}$$

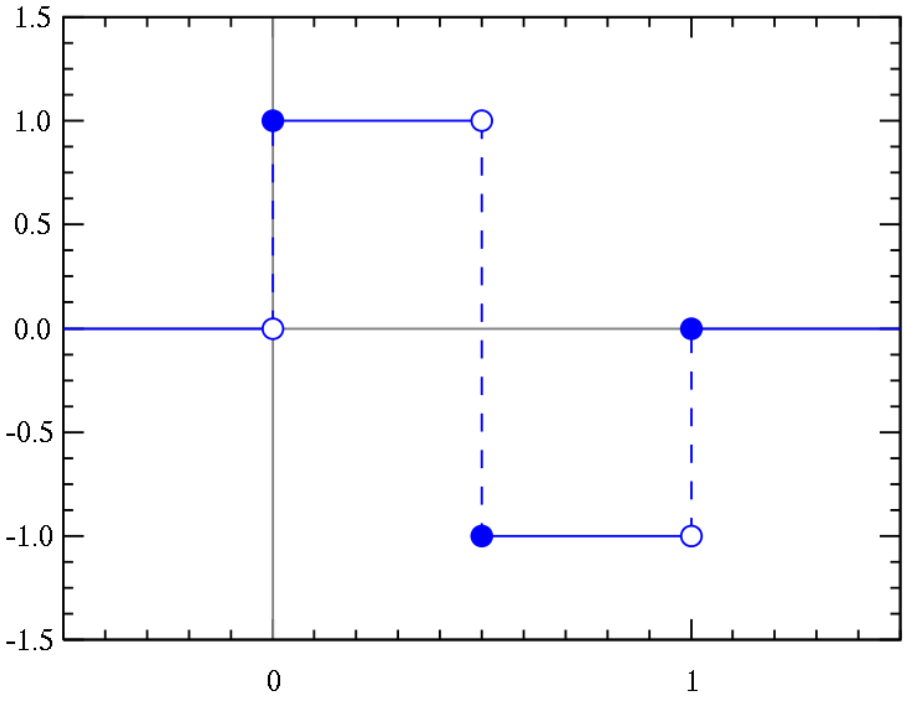
Burada v bir tamsayı değeridir ve $\psi_{m,n}(t)$'nin bu v parçaları "ses" denilen bir grup olarak işleme sokulur. v tamsayısı, her oktav için seslerin sayısıdır ve müzik bağlamında iyi akortlu bir ölçek tanımlar. Bu, geleneksel Fourier analizindeki bir dar bant filtreleri kümesinin kullanımına benzer (Lee ve Yamamoto 1994).

3.9.3. Haar Dalgacığı

Dalgacık analizi işlemi analiz edici dalgacık ya da ana dalgacık denilen bir dalgacık prototip fonksiyonu seçerek gerçekleştirilir. Zaman analizi prototip dalgacığın kısaltılmış, yüksek frekanslı bir versiyonu ile gerçekleştirilirken; frekans analizi aynı dalgacığın genişletilmiş, düşük frekanslı bir versiyonu ile gerçekleştirilir. Orijinal işaret ya da fonksiyon

dalgacık katsayılarının doğrusal bir kombinasyonundaki katsayılar kullanılarak bir dalgacık açılımı ile ifade edilebilir. Veriye uyarlanmış en iyi dalgacıklar seçilirse ya da dalgacıklar bir eşik değerinin altına kırpılırsa, veri seyrek olarak temsil edilir. Bu seyrek kodlama, dalgacıkları, veri sıkıştırma alanında mükemmel bir araç yapar.

Bu tezde gerçekleştirilen dalgacık analizlerinde Haar dalgacıkları kullanılmıştır. Şekil 3.3'te Haar dalgacığı gösterilmiştir



Şekil 3.3. Haar dalgacığı (Anonim 2010)

Haar dalgacığının ana dalgacık fonksiyonu ψ(t) şu şekilde ifade edilir (Anonim 2012d):

$$\psi(t)=\begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \\ 0 & \text{diğer durumlarda} \end{cases} \tag{3.49}$$

Haar dalgacığının ölçekleme fonksiyonu ϕ(t) ise şu şekilde ifade edilir (Anonim 2012d):

$$\phi(t)=\begin{cases} 1 & 0 \le t < 1 \\ 0 & \text{diğer durumlarda} \end{cases} \tag{3.50}$$

Şekil 3.4, Şekil 3.5, Şekil 3.6 ve Şekil 3.7'deki grafikler bir Haar dalgacığı ile üst üste getirilmiş bir işareti göstermektedir.

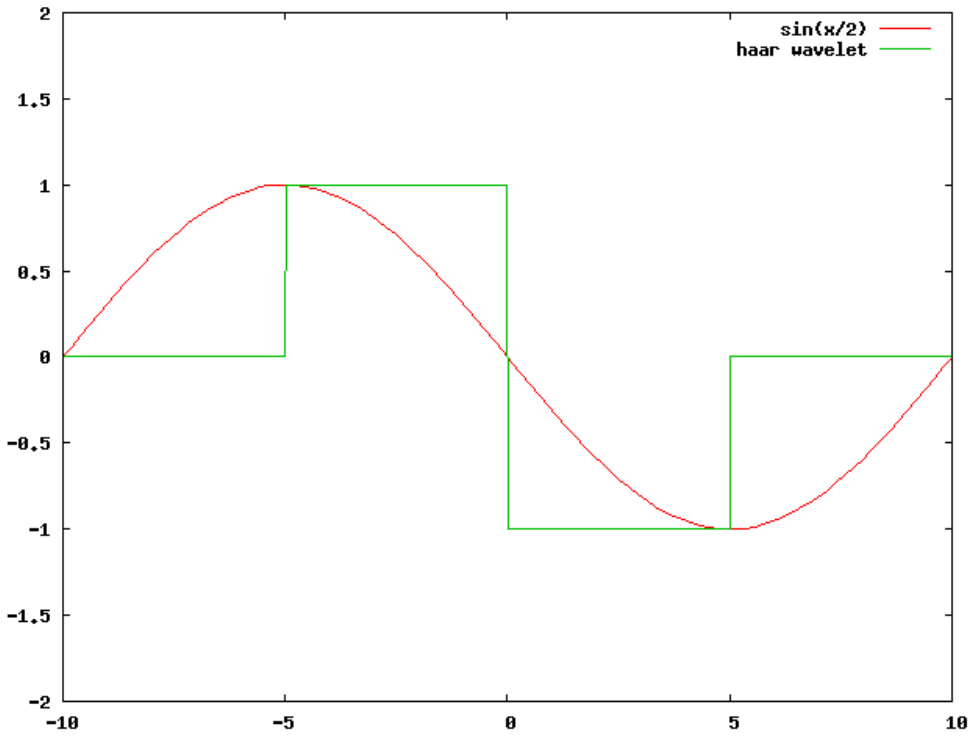
Şekil 3.4 hedef frekansın yarısı kadar bir sinyali gösterir. Dalga şekli simetrik olmadığı halde, ikinci yarısı Haar dalgacığının negatif kısmıyla çarpılacak ve pozitife çevrilecektir. Sonuçta, integralin küçük fakat önemli bir pozitif değeri olacaktır.

Şekil 3.5 hedef frekanstaki bir işareti gösterir. Haar dönüşümüyle integral maksimum olacaktır. Haar dalgacığının ikinci yarısı ile işaret değiştirme sonucu sinüs dalgasının negatif kısmı pozitife çevrilerek alan en büyük hale getirilir.

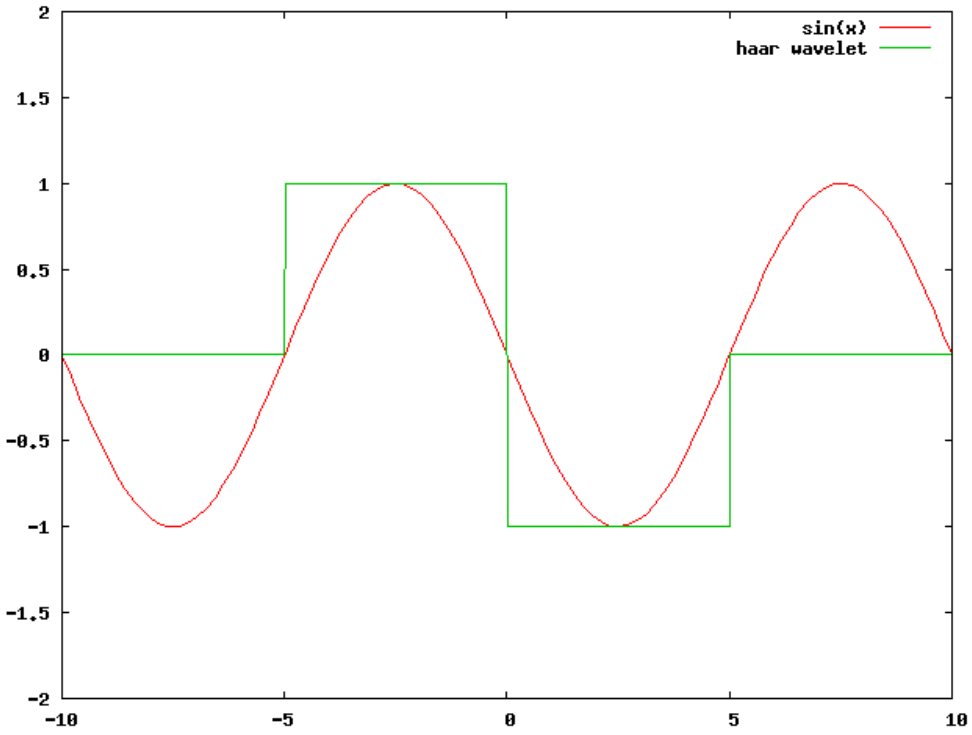
Şekil 3.6 hedef frekansın iki katındaki bir işareti gösterir. Bu işaret, Haar dalgacığının her yarısında tam bir döngü yaparak integrali sıfır yapar. Bütün daha yüksek harmoniklerde de aynı durum görülebilir.

Şekil 3.7'de işaret Haar dalgacık penceresinin ilerisinde başlamak üzere geciktirilmiştir. Burada sonuç çok düşük değerli olacaktır.

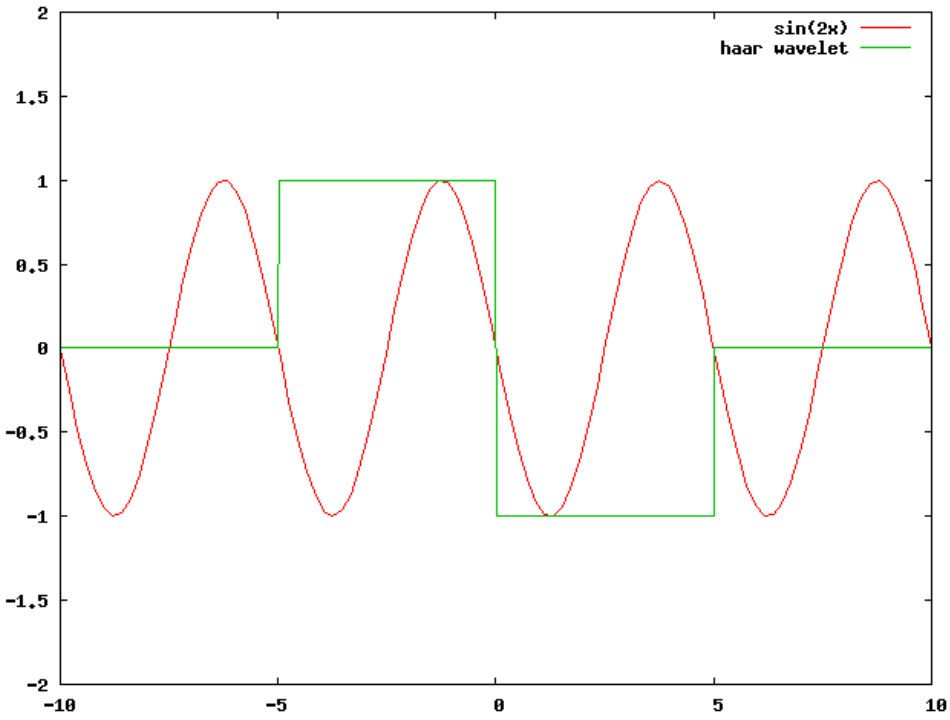
Şekil 3.4, Şekil 3.5, Şekil 3.6 ve Şekil 3.7 göstermektedir ki, bir dalgacık hedef frekanstaki veya onun altındaki bütün sinyalleri kabul eder, hedef frekanstan daha yüksek frekanstaki sinyalleri reddeder (Anonim 2010).



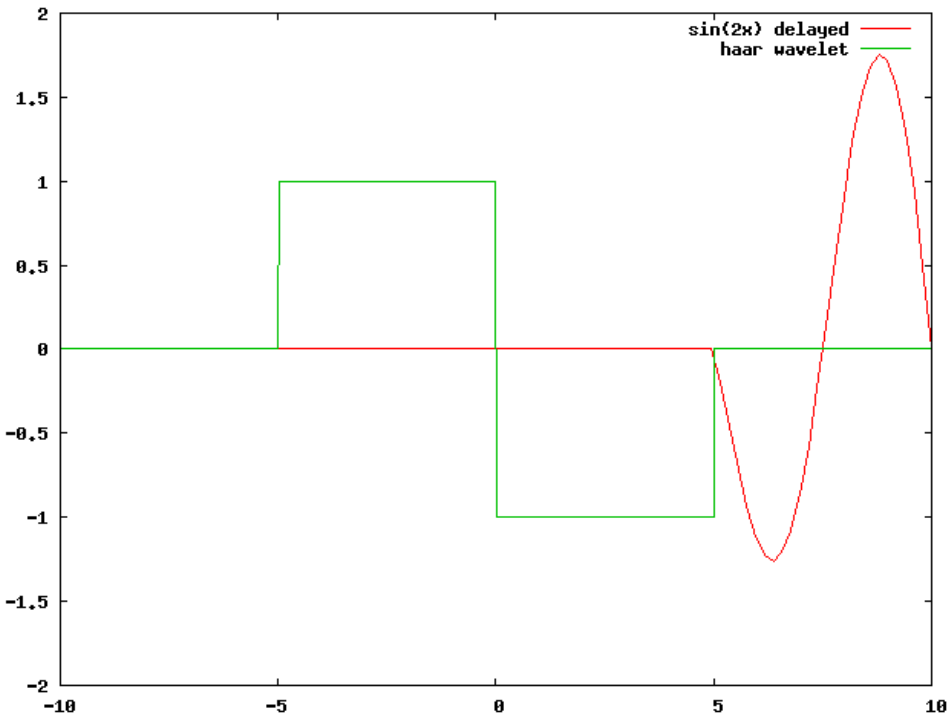
Şekil 3.4. Hedef frekansın yarısı kadar bir sinyal ile Haar dalgacığının üst üste gelmesi (Anonim 2010)



Şekil 3.5. Hedef frekanstaki bir sinyal ile Haar dalgacığının üst üste gelmesi (Anonim 2010)



Şekil 3.6. Hedef frekansın iki katı kadar bir sinyal ile Haar dalgacığının üst üste gelmesi (Anonim 2010)



Şekil 3.7. Haar dalgacığının ilerisinde başlamak üzere geciktirilmiş bir işaret (Anonim 2010)

4. YAPAY SİNİR AĞLARI (YSA)

Yapay sinir ağları, insan beyninin özelliklerini kullanarak, öğrenme yoluyla yeni bilgiler üretebilen ve keşfedebilen, önceki verilmiş örnekleri kullanarak yeni durumlara göre cevaplar üretebilen, kendi kendine karar verebilme yeteneği olan bilgisayar sistemleridir.

Günümüzdeki sinir ağı teorisinin babası olarak bilinen Donald Hebb, beynin çalışma yapısı hakkında yaptığı çalışmalarda beyin sinir hücrelerini temel olarak almıştır. İki hücre arasındaki korelasyonu inceleyerek bunu sinir ağı teorisinin temeli yapmıştır. Günümüzde beynin çalışma yapısı hakkındaki görüşler teorilerden öteye gidemediği için iki hücre arasındaki korelasyon, sinir ağı teorisinde ele alınacak tek faktör olamaz. Fakat Hebb'in ortaya attığı bu temele dayanarak bugün yüzlerce teori üretilmiş ve çok büyük oranda başarı oranına sahip birçok yapay sinir ağı modeli uygulamaya konmuştur (Fausett 1994, Haykin 1994).

4.1. Yapay Sinir Ağlarının Genel Özellikleri

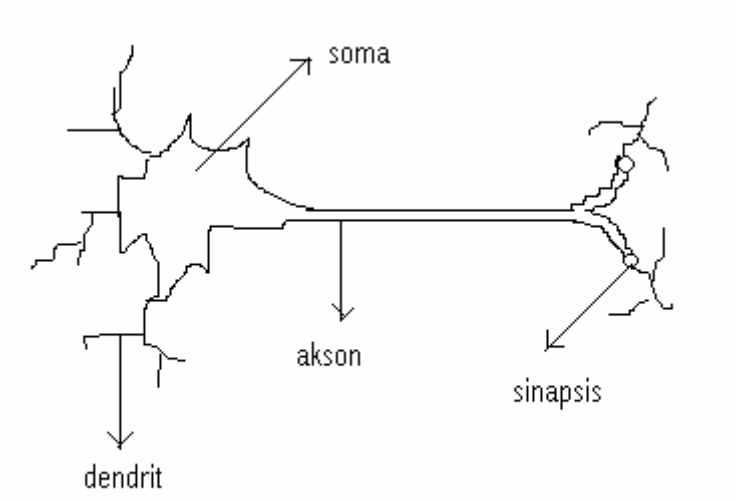
Yapay sinir ağlarının genel özellikleri şu şekilde sıralanabilir (Bozer 2012):

- Yapay sinir ağları, makine öğrenmesi gerçekleştirirler.
- Programları çalışma stili bilinen programlama yöntemlerine benzememektedir.
- Doğrusal olmayan özelliğe sahiptirler.
- Yapay sinir ağlarında bilgi ağın bağlantılarının değerleri ile ölçülmekte ve bağlantılarda saklanmaktadır.
- Yapay sinir ağları, örnekleri kullanarak öğrenirler.
- Yapay sinir ağları, görülmemiş örnekler hakkında bilgi üretebilirler.
- Kendi kendini organize etme ve öğrenebilme yetenekleri vardır.
- Eksik bilgi ile çalışabilmektedirler.
- Hata toleransına sahiptirler.
- Belirsiz ve tam olmayan bilgileri işleyebilmektedirler.
- YSA'lar ani bozulma göstermezler. Yapay sinir ağlarının hata toleransına sahip olmaları dereceli bozulma gösterebilmelerini sağlar.
- Dağıtık bilgiye sahiptirler. Ağın bilgisi bağlantılara yayılmış haldedir.
- Sadece sayısal bilgiler ile çalışabilmektedirler.
- YSA'lar normal yollarla çözülmesi zor olan sınıflandırma ve tahminleme gibi problemleri çözmek için tasarlanmışlardır.

Yapay sinir ağlarının amacı insan beyninin çalışma yapısını modellemektir. Yapay sinir ağları şu işleri yapabilir: Öğrenme, ilişkilendirme, sınıflandırma, genelleme, tahmin, özellik belirleme, eniyileme. Yapay sinir ağları bu işleri gerçekleştirirken dışarıdan bir etki ve yardım olmadan elindeki bilgileri kullanarak yeni durumlara yeni sonuçlar üretir. Yapay sinir ağları öğrenme aşamasında kendisine sunulan bilgileri değerlendirerek daha sonra karşısına çıkan yeni durumlarda yeni ve doğru kararlar verebilir (Kakıcı 2009a).

4.2. Gerçek Sinir Hücresi

İnsan beyni, her biri bir işlemci gibi çalışan milyonlarca sinir hücresinden oluşur. Beynin işlevlerinin yerine getirilmesi, bu hücreler arasındaki etkileşim ve paralel çalışabilme özelliği ile olur. Şekil 4.1'de gerçek bir sinir hücresinin yapısı gösterilmiştir.



Şekil 4.1. Biyolojik sinir hücre yapısı (Fausset 1994)

Dendrit: Görevi diğer sinir hücrelerinden iletilen sinyalleri, sinir hücresinin çekirdeğine iletmektedir.

Soma: Dendritler yoluyla iletilen tüm sinyalleri alıp toplayan merkezdir. Biyolojik olarak hücre çekirdeği (nükleus) olarak da bilinen yapıdır. Çekirdek gelen toplam sinyali diğer sinir hücrelerine göndermek üzere, bilgiyi aksona iletir.

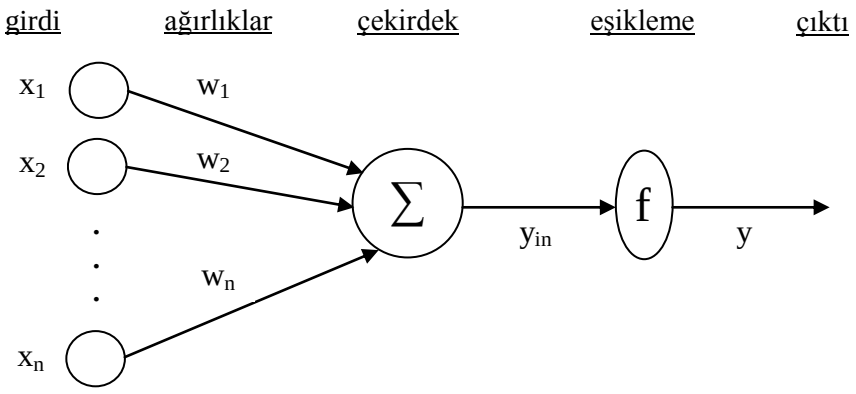
Akson: Hücre çekirdeğinden aldığı toplam bilgiyi sinapsise iletmekle görevlidir.

Sinapsis: Aksondan gelen toplam bilgiyi ön işlemden geçirdikten sonra diğer sinir hücrelerinin dendritlerine iletmekle görevlidir. Sinapsisin ön işlem ile gerçekleştirdiği görev çok önem taşımaktadır. Bu ön işlem gelen toplam sinyalin, belli bir eşik değerine göre değiştirilmesinden ibarettir. Böylece toplam sinyal olduğu gibi değil, belli bir aralığa

indirgenerek diğer sinir hücrelerine iletilmiş olur. Bu açıdan, her gelen toplam sinyal ile dendrite iletilen sinyal arasında bir korelasyon (ilişki) oluşturulur. Buradan yola çıkılarak "öğrenme" işleminin sinapsislerde gerçekleştiği fikri ortaya atılmış ve bu hipotez, günümüz yapay sinir ağı dünyası için teori haline dönüşmüştür. Yapay sinir ağı modelleri üzerinde "öğrenme" bu teoriye dayanılarak, sinapsisler ve dendritler arasında yer alan ağırlık katsayılarının güncellenmesi olarak algılanmaktadır.

4.3. Yapay Sinir Hücresi

Yapay sinir hücreleri gerçek sinir hücrelerinin simülasyonu ile elde edilir. Şekil 4.2'deki yapay sinir hücresinin dendritleri x_n ve her bir dendritin ağırlık katsayısı (önemlilik derecesi) w_n ile gösterilmiştir. x_n girdi işaretlerini, w_n bu işaretlerin ağırlık katsayılarını ifade eder. Çekirdek ise tüm girdi işaretlerinin ağırlıklı toplamlarını elde eder. Tüm bu toplam işaret y_{in} ile gösterilmiş ve eşikleme fonksiyonuna girdi olarak sinapsise yönlendirilmiştir. Sinapsis üzerindeki eşikleme fonksiyonundan çıkan sonuç işareti y ile gösterilmiş olup diğer hücreye beslenmek üzere yönlendirilmiştir.



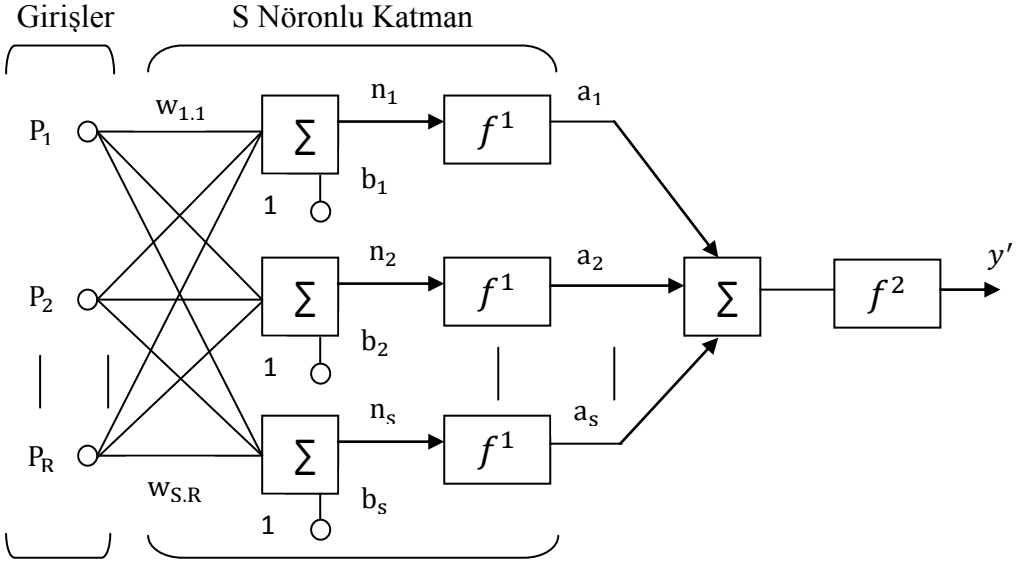
Şekil 4.2. Yapay sinir hücre yapısı

Yapay sinir hücresinin görevi kısaca; x_n girdi örüntüsüne karşılık y çıktı işaretini oluşturmak ve bu işareti diğer hücrelere iletmektir. Her x_n ile y arasındaki korelasyonu temsil eden w_n ağırlıkları, her yeni girdi örüntüsü ve çıktı işaretine göre tekrar ayarlanır. Bu ayarlama süreci öğrenme olarak adlandırılır. Öğrenmenin tamamlandığının belirtilebilmesi için; girdi örüntüleri, w_n ağırlıklarındaki değişim durağan olana dek sistemi beslemektedir. Durağanlık sağlandığı zaman hücre öğrenmesini tamamlamıştır.

Yapay sinir ağları; görevi yukarıdaki biçimde belirtilen yapay sinir hücrelerinin birleşiminden oluşan katmanlı yapının tümü olarak nitelendirilir. Böylece "m" adet yapay sinir hücresinin katmanlı yapısıyla yapay sinir ağı modeli kurulmuş olmaktadır (Cinsdikici 2012).

4.4. Çok Girişli S Nöronlu Hücre Modeli

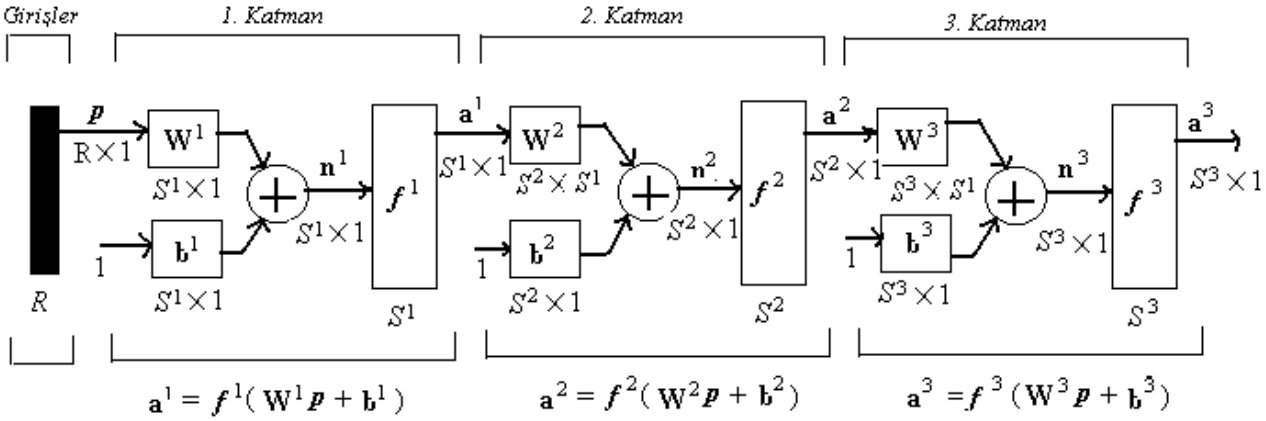
Birçok giriş için genellikle bir nöron yeterli olmayabilir. Paralel işlem yapan birden fazla nörona ihtiyaç duyulduğunda katman kavramı devreye girmektedir. S tane nöronun tek bir katmanı Şekil 4.3'te gösterilmiştir. Burada R girişin her biri bir nörona bağlıdır (Çetin 2012).



Şekil 4.3. Çok girişli S nöronlu hücre modeli

4.5. Çok Katmanlı Yapay Sinir Ağları

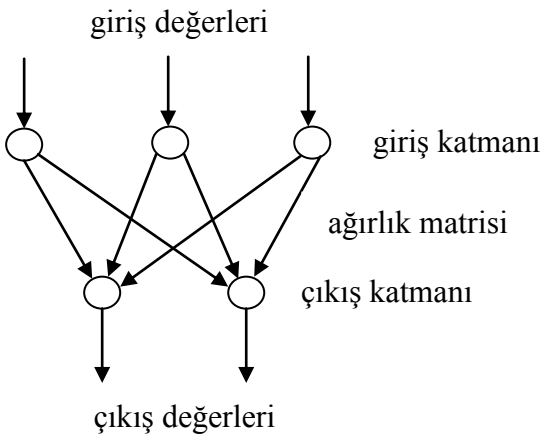
Çok katmanlı yapay sinir ağları, nöronları çıkışa direkt olarak bağlı olmayan saklı katmanları kullanarak lineer olmayan sınıflandırma problemlerini çözebilir. İlave saklı katmanlar, geometrik olarak, ağın ayırt etme kapasitesini artıran ilave hiperdüzlemler olarak yorumlanabilir (Schmidt 2000). Şekil 4.4'te üç katmandan oluşan bir yapay sinir ağının genel yapısı gösterilmiştir.



Şekil 4.4. Üç katmanlı bir yapay sinir ağının genel yapısı (Çetin 2012)

4.6. Perceptron Yapay Sinir Ağı Modeli

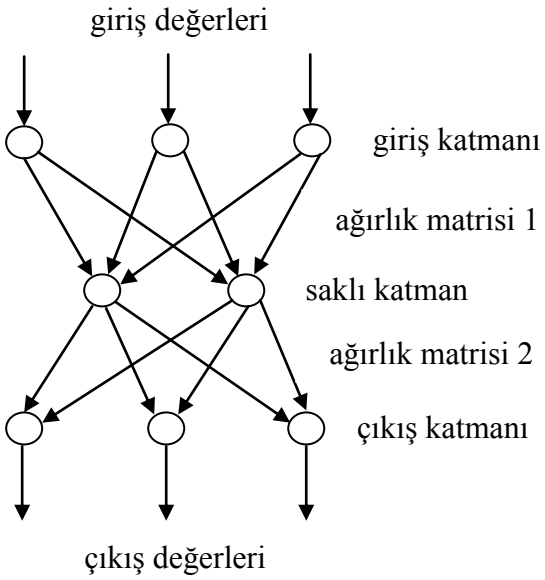
Şekil 4.5'te gösterilen ve yapısı ileri beslemeli, öğrenme metodu öğretmenli öğrenme, öğrenme algortiması Hebb algoritması olan perceptron YSA modeli "And", "Or" gibi doğrusal problemler için uygundur. Öğretmenli öğrenme sırasında ağa verilen giriş değerleri için çıktı değerleri de verilir. Ağ, verilen girdiler için istenen çıkışları oluşturabilmek için kendi ağırlıklarını günceller. Ağın çıktıları ile beklenen çıktılar arasındaki hata hesaplanarak ağın yeni ağırlıkları bu hata payına göre düzenlenir. Hata payı hesaplanırken ağın bütün çıktıları ile beklenen çıktıları arasındaki fark hesaplanır ve bu farka göre her nörona düşen hata payı bulunur. Daha sonra her nöron kendine gelen ağırlıkları günceller (Kakıcı 2009b).



Şekil 4.5. Perceptron yapay sinir ağı modeli

4.7. Çok Katmanlı Perceptron Yapay Sinir Ağı Modeli

Şekil 4.6'da gösterilen ve yapısı ileri beslemeli, öğrenme metodu öğretmenli öğrenme, öğrenme algoritması "delta öğrenme kuralı / geri yayılma" algoritması olan çok katmanlı perceptron YSA modeli, doğrusal olmayan problemlerin çözümüne de olanak sağlamaktadır (Bozer 2012).

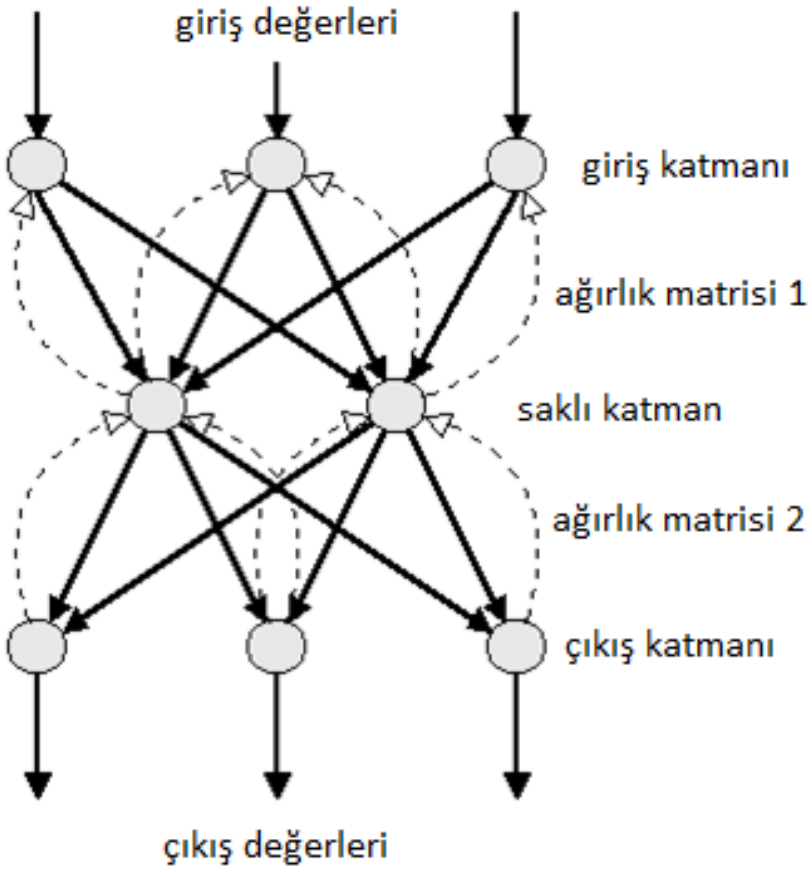


Şekil 4.6. Çok katmanlı perceptron yapay sinir ağı modeli

4.8. Geriye Yayılım Ağı

Bir tek katmanlı YSA'da birçok kısıtlama vardır ve çok sınırlı işler yapabilir. Minsky ve Papert (1969), iki katmanlı ileri beslemeli bir YSA'nın birçok kısıtlamanın üstesinden gelebileceğini göstermişlerdir. Ama girişten saklı katmana ağırlıkların nasıl ayarlanacağı problemine bir çözüm sunmamışlardır. Bu probleme çözüm Rumelhart ve ark. (1986) tarafından sunulmuştur. Bu çözümün ana fikri, saklı katmanın birimleri için hataların, çıkış katmanının birimlerinin hatalarının geriye yayılması ile belirlenmesidir. Bu yönteme geriye yayılma öğrenme kuralı denir. Geriye yayılma, çok katmanlı YSA'ları eğitmenin sistematik bir yöntemidir.

Şekil 4.7'de çok katmanlı perceptron modelinin geriye yayılım algoritması ile yapılandırılması gösterilmiştir.



Şekil 4.7. Çok katmanlı perceptron modelinin geriye yayılım algoritması ile yapılandırılması

4.9. Yapay Sinir Ağlarının Kullanım Sahaları

Yapay sinir ağları pek çok sektörde değişik uygulama alanları bulmuştur. Bunlardan bazıları şunlardır (Altıntaş 2011):

Uzay: Uçuş benzetimleri, otomatik pilot uygulamaları, bileşenlerin hata denetimleri vs.

Otomotiv: Otomatik yol izleme, rehber, garanti aktivite analizi, yol koşullarına göre sürüş analizi vs.

Bankacılık: Kredi uygulamaları geliştirilmesi, müşteri analizi ve kredi başvuru değerlendirilmesi, bütçe yatırım tahminleri vs.

Savunma: Silah yönlendirme, hedef seçme, radar, sensör sonar sistemleri, işaret işleme, görüntü işleme vs.

Elektronik: Kod sırası öngörüsü, çip bozulma analizi, doğrusal olmayan modelleme vs.

Eğlence: Animasyonlar, özel efektler, pazarlama öngörüsü vs.

Finans: Kıymet biçme, pazar performans analizi, bütçe kestirimi, hedef belirleme vs.

Sigortacılık: Ürün eniyilemesi, uygulama politikası geliştirme vs.

Üretim: Üretim işlem kontrolü, ürün tasarımı, makine yıpranmalarının tespiti, dayanıklılık analizi, kalite kontrolü, iş çizelgeleri hazırlanması vs.

Sağlık: Göğüs kanseri erken teşhis ve tedavisi, EEG, EKG, MR (manyetik rezonans), kalite artırımı, ilaç etkileri analizi, kan analizi sınıflandırma, kalp krizi erken teşhis ve tedavisi vs.

Petrokimya: Arama, verim analizi vs.

Robotik: Yörünge kontrol, forklift robotları, görsel sistemler, uzaktan kumandalı sistemler, en iyi rota belirleme vs.

Dil: Sözcük tanıma, yazı ve konuşma çevrimi, dil tercüme vs.

Telekomünikasyon: Görüntü ve veri karşılaştırma, filtreleme, eko ve gürültü sönümlendirilmesi, ses ve görüntü işleme, trafik yoğunluğunun kontrolü ve anahtarlama vs.

Güvenlik: Parmak izi tanıma, kredi kartı hileleri saptama, retina tarama, yüz eşleştirme vs.

5. DESTEK VEKTÖR MAKİNELERİ (DVM)

5.1. Doğrusal Olarak Ayrılabilir İkili Sınıflandırma

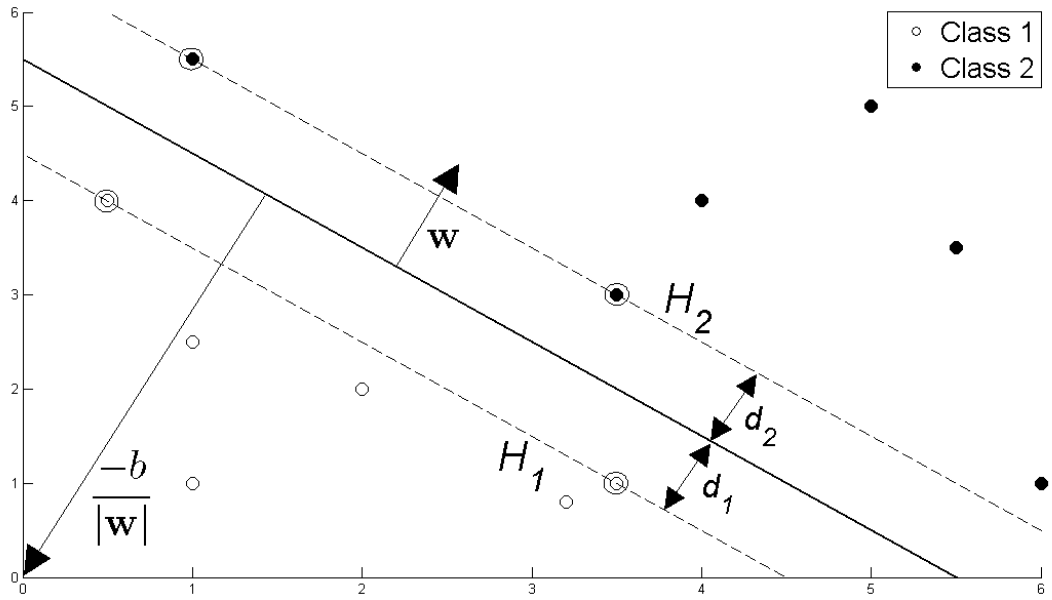
Her x_i girişinin B adet özniteliği olduğu, yani B boyutlu olduğu ve y_i=-1 ya da +1 olmak üzere iki sınıftan birine ait olduğu, yani eğitim verisinin i=1…L, $y_i \in \{-1,+1\}$, $x \in R^B$ olmak üzere $\{x_i,y_i\}$ şeklinde olduğu L adet eğitim noktasının olduğu varsayılsın.

Burada verinin doğrusal olarak ayrılabilir olduğu varsayılsın. Yani B=2 olduğunda x_1'e karşı x_2'nin grafiği üzerinde iki sınıfı ayıran bir doğru çizilebilir. B>2 olduğu durum için de $x_1,x_2 \ldots x_B$'nin grafikleri üzerinde bir hiperdüzlem çizilebilir.

Bu hiperdüzlem w.x+b=0 olarak tanımlanabilir.

Burada w hiperdüzlemin normalidir, $\frac{b}{\|w\|}$ ise hiperdüzlemden orjine olan dik uzaklıktır.

Destek vektörleri ayırıcı hiperdüzleme en yakın olan örneklerdir ve destek vektör makinelerinin amacı bu hiperdüzlemi, her iki sınıfın en yakın üyelerinden mümkün olduğu kadar uzak olacak şekilde yönlendirmektir. Doğrusal olarak ayrılabilir iki sınıf boyunca bir hiperdüzlem örneği Şekil 5.1'de gösterilmiştir.



Şekil 5.1. Doğrusal olarak ayrılabilir iki sınıf boyunca hiperdüzlem (Fletcher 2009)

Şekil 5.1'e göre, bir DVM'yi uygulamak demek, eğitim verisi şu şekilde tanımlanacak şekilde w ve b değişkenlerini seçmek demektir:

$$x_i . w + b \geq +1 \qquad y_i = +1 \text{ için} \tag{5.1}$$

$$x_i . w + b \leq -1 \qquad y_i = -1 \text{ için} \tag{5.2}$$

Bu ifadeler şu şekilde birleştirilebilir:

$$y_i(x_i . w + b) - 1 \geq 0 \quad \forall_i \tag{5.3}$$

Ayırıcı hiperdüzleme en yakın olarak uzanan noktalar, yani Şekil 5.1'deki destek vektörleri ele alınırsa, bu noktaların üzerinde uzandıkları H_1 veH_2 düzlemleri şu şekilde tanımlanır:

$$x_i . w + b = +1 \qquad H_1 \text{ için} \tag{5.4}$$

$$x_i . w + b = -1 \qquad H_2 \text{ için} \tag{5.5}$$

Şekil 5.1'e göre, d_1, H_1'den hiperdüzleme olan uzaklık, d_2 ise H_2'den hiperdüzleme olan uzaklık olarak tanımlanır. Hiperdüzlemin H_1 ve H_2'den eşit uzaklıkta olması d_1=d_2 olması anlamına gelir ve bu büyüklük DVM'nin marjini olarak bilinir. Hiperdüzlemi destek vektörlerinden mümkün olduğu kadar uzak yönlendirmek için marjinin maksimize edilmesi gerekir.

Basit vektör geometrisine göre, marjin $\frac{1}{\|w\|}$'ya eşittir ve bunu (5.3) eşitsizliğinin sınırlaması altında maksimize etmek şunu bulmaya eşittir:

$$\min\|w\| \quad \text{öyle ki} \quad y_i(x_i . w + b) - 1 \geq 0 \quad \forall_i$$

$\|w\|$'yu minimize etmek $\frac{1}{2}\|w\|^2$'yi minimize etmek demektir ve bu terimin kullanımı daha sonra kuadratik programlama (KP) eniyilemeyi gerçekleştirmeyi mümkün kılar. Bu nedenle şunu bulmak gerekir:

$$\min \frac{1}{2}\|w\|^2 \quad \text{öyle ki} \quad y_i(x_i . w + b) - 1 \geq 0 \quad \forall_i \tag{5.6}$$

Bu minimizasyondaki sınırlandırmaları karşılamak için onlara, $\alpha_i \geq 0 \ \forall_i$ olmak üzere, α Lagrange çarpanları tayin edilmelidir:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \alpha[y_i(x_i.w + b) - 1 \ \forall_i] \tag{5.7}$$

$$\equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i[y_i(x_i.w + b) - 1] \tag{5.8}$$

$$\equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i y_i (x_i.w + b) + \sum_{i=1}^{L} \alpha_i \tag{5.9}$$

$\alpha_i \geq 0 \ \forall_i$ olmak üzere (5.9) denklemini minimize eden w ve b ile (5.9) denklemini maksimize eden α'yı bulmak hedeflenmektedir. Bu, L_P'nin w ve b'ye göre türevi alınıp türevlerin sıfıra eşitlenmesi ile yapılabilir:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \tag{5.10}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{5.11}$$

(5.10) ve (5.11) denklemleri (5.9) denkleminde yerine koyulursa, α'ya bağlı olmak üzere maksimize edilmesi gereken şu yeni formülasyon elde edilir:

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i.x_j \quad \text{öyle ki} \quad \alpha_i \geq 0 \ \forall_i, \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{5.12}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i H_{ij} \alpha_j \ \rightarrow \ \text{burada } H_{ij} \equiv y_i y_j x_i.x_j \text{ biçimindedir.} \tag{5.13}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha \quad \text{öyle ki} \quad \alpha_i \geq 0 \ \forall_i, \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{5.14}$$

Bu yeni L_D formülasyonuna ikili biçim ya da birincil L_P denir. Şunu da belirtmek gerekir ki, ikili biçim sadece hesaplanacak her x_i giriş vektörünün nokta çarpımını gerektirir.

L_P'yi minimize etmekten L_D'yi maksimize etmeye geçince şunu bulmak gerekir:

$$\max_{\alpha}\left[\sum_{i=1}^{L}\alpha_i-\frac{1}{2}\alpha^T H\alpha\right] \quad \text{öyle ki} \quad \alpha_i \geq 0 \ \forall_i \text{ ve } \sum_{i=1}^{L}\alpha_i y_i = 0 \qquad (5.15)$$

Bu bir konveks kuadratik eniyileme problemidir ve α'yı ve (5.10) denkleminden w'yu verecek olan bir KP çözümleyici çalıştırılır. Geriye b'yi hesaplamak kalır.

(5.11) denklemini sağlayan ve bir x_s destek vektörü olan herhangi bir nokta şu şekilde olacaktır:

$$y_s(x_s.w+b)=1$$

(5.10) denkleminde yerine koyulursa şu elde edilir:

$$y_s\left(\sum_{m\in S}\alpha_m y_m x_m.x_s+b\right)=1$$

Burada S, destek vektörlerinin indisler kümesini gösterir. S, $\alpha_i > 0$ olduğu i indislerini bularak belirlenir. (5.1) ve (5.2) denklemlerinden, y_s ile çarparak ve sonra y_s^2=1'i kullanarak şunlar elde edilir:

$$y_s^2\left(\sum_{m\in S}\alpha_m y_m x_m.x_s+b\right)=y_s$$

$$b=y_s-\sum_{m\in S}\alpha_m y_m x_m.x_s$$

Rastgele bir x_s destek vektörü kullanmak yerine, S kümesindeki bütün destek vektörlerinin ortalamasını almak daha iyidir.

$$b=\frac{1}{N_s}\sum_{s\in S}\left(y_s-\sum_{m\in S}\alpha_m y_m x_m.x_s\right) \qquad (5.16)$$

Artık, ayırıcı hiperdüzlemin en iyi yönlendirmesini tanımlayan w ve b değişkenleri ve dolayısıyla destek vektör makinesi elde edilmiştir.

Doğrusal olarak ayrılabilir bir ikili sınıflandırma problemini çözmek üzere bir destek vektör makinesi kullanmak için şunlar yapılır:

- $H_{ij} = y_i y_j x_i . x_j$ olmak üzere H oluşturulur.
- $\alpha_i \geq 0 \ \forall_i$ ve $\sum_{i=1}^{L} \alpha_i y_i = 0$ sınırlandırması altında $\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha$ maksimize edilecek şekilde α bulunur. Bu, bir KP çözümleyici kullanılarak yapılır.
- $w = \sum_{i=1}^{L} \alpha_i y_i x_i$ hesaplanır.
- $\alpha_i > 0$ olacak şekilde indisler bulunarak S destek vektörler kümesi belirlenir.
- $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m x_m . x_s)$ hesaplanır.
- $y' = \text{sgn}(w . x' + b)$ hesaplanarak her yeni x' noktası sınıflandırılır (Fletcher 2009).

5.2. Tamamen Doğrusal Olarak Ayrılabilir Olmayan Veri için İkili Sınıflandırma

Tamamen doğrusal olarak ayrılabilir olmayan veri ile işlem yapmak üzere DVM metodolojisini genişletmek için, (5.1) ve (5.2) sınırlandırmaları, yanlış sınıflandırılan noktalara izin vermek için hafifçe serbest bırakılır. Bu, i=1,…L olmak üzere bir pozitif arttıran yapay değişken ξ_i kullanılarak yapılır:

$$x_i . w + b \geq +1 - \xi_i \qquad y_i = +1 \text{ için} \tag{5.17}$$

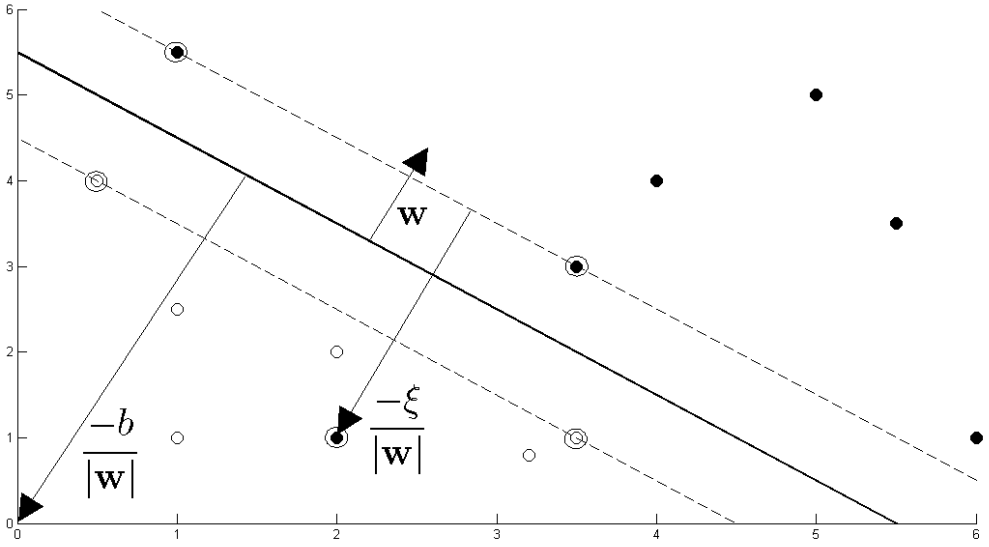
$$x_i . w + b \leq -1 + \xi_i \qquad y_i = -1 \text{ için} \tag{5.18}$$

$$\xi_i \geq 0 \ \forall_i \tag{5.19}$$

Bunlar şu şekilde birleştirilebilir:

$$y_i (x_i . w + b) - 1 + \xi_i \geq 0 \ \rightarrow \ \text{burada } \xi_i \geq 0 \ \forall_i \text{ biçimindedir.} \tag{5.20}$$

Doğrusal olarak ayrılabilir olmayan iki sınıf boyunca bir hiperdüzlem örneği Şekil 5.2'de gösterilmiştir.



Şekil 5.2. Doğrusal olarak ayrılabilir olmayan iki sınıf boyunca hiperdüzlem (Fletcher 2009)

Bu yumuşak marjin DVM'sinde, marjin sınırının yanlış tarafı üzerindeki veri noktalarının, ondan olan uzaklıkla artan bir cezası vardır. Yanlış sınıflandırmaların sayısı azaltılmak istendiğine göre, (5.6) hedef fonksiyonunu uyarlamak için mantıklı bir yol şunu bulmaktır:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i \quad \text{öyle ki} \quad y_i(x_i.w + b) - 1 + \xi_i \geq 0 \ \ \forall_i \qquad (5.21)$$

Burada C parametresi, arttıran yapay değişken cezası ve marjinin boyutu arasındaki uzlaşmayı kontrol eder. Lagrange ile yeniden formüle edilerek, daha önce yapılana benzer şekilde, w, b ve ξ_i'ye göre minimize etmek ve α'ya göre maksimize etmek gerekir:

$$L_P \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i[y_i(x_i.w + b) - 1 + \xi_i] - \sum_{i=1}^{L}\mu_i\xi_i \qquad (5.22)$$

w, b ve ξ_i'ye göre türevler alınarak sıfıra eşitlenir:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \tag{5.23}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{5.24}$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i \tag{5.25}$$

Bunlar yerine koyulursa, L_D (5.14) denklemindekiyle aynı biçimde olur. Ancak, $\mu_i \geq 0 \ \forall_i$ olmak üzere (5.25) denklemi $\alpha \leq C$ olduğunu gösterir. Bu nedenle şunu bulmak gerekir:

$$\max_{\alpha} \left[\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \right] \quad \text{öyle ki} \quad 0 \leq \alpha_i \leq C \ \forall_i \quad \text{ve} \quad \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{5.24}$$

Daha sonra, (5.6) denklemindeki gibi b hesaplanır, ancak bu durumda, b'yi hesaplamak için kullanılan destek vektörleri kümesi, $0 < \alpha_i \leq C$ olmak üzere i indisleri bulunarak belirlenir.

Tamamen doğrusal olarak ayrılabilir olmayan veri için bir ikili sınıflandırmayı çözmek üzere bir DVM kullanmak için şunlar yapılır:

- $H_{ij} = y_i y_j x_i . x_j$ olmak üzere H oluşturulur.
- C parametresi için uygun bir değer seçilerek yanlış sınıflandırmalara ne kadar önem verileceği belirlenir.
- $0 \leq \alpha_i \leq C \ \forall_i$ ve $\sum_{i=1}^{L} \alpha_i y_i = 0$ sınırlandırması altında $\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha$ maksimize edilecek şekilde α bulunur. Bu, bir KP çözümleyici kullanılarak yapılır.
- $w = \sum_{i=1}^{L} \alpha_i y_i x_i$ hesaplanır.
- $0 < \alpha_i \leq C$ olacak şekilde indisler bulunarak S destek vektörler kümesi belirlenir.
- $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m x_m . x_s)$ hesaplanır.
- $y' = \text{sgn}(w . x' + b)$ hesaplanarak her yeni x' noktası sınıflandırılır (Fletcher 2009).

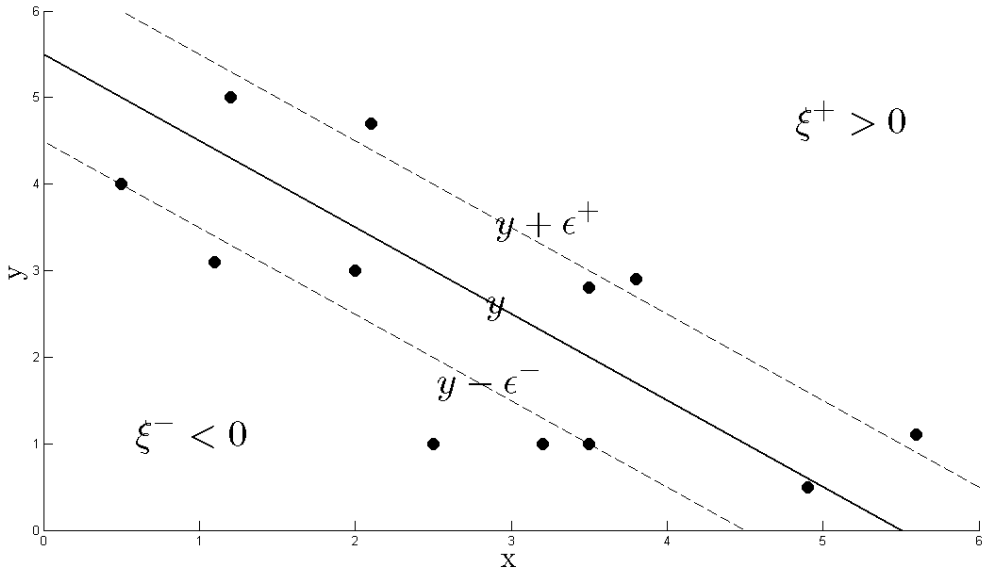
5.3. Regresyon için Destek Vektör Makineleri

Görülmemiş yeni x' değişkenlerini $y' = \pm 1$ olmak üzere iki kategoriden birine sınıflandırmaya çalışmak yerine, şimdi, eğitim verisi şu şekilde olacak biçimde, y' için gerçek değerli bir çıkış öngörülecektir:

$$\{x_i, y_i\} \rightarrow \text{burada } i = 1 \ldots L, y_i \in R, x \in R^D \text{ biçimindedir.}$$

$$y_i = w.x_i + b \tag{5.25}$$

Şekil 5.3'te Є-duyarsız tüp ile regresyon örneği gösterilmiştir.



Şekil 5.3. Є-duyarsız tüp ile regresyon (Fletcher 2009)

Regresyon DVM'si öncekinden daha karmaşık bir ceza fonksiyonu kullanır. Eğer tahmin edilen değer y_i, gerçek değer t_i'den, Є uzaklığından daha az bir uzaklıkta ise, yani $|t_i - y_i| < Є$ ise, ceza tayin edilmez. Şekil 5.3'e göre $y_i \pm Є \ \forall_i$ ile sınırlanan bölgeye Є-duyarsız tüp denir. Ceza fonksiyonundaki diğer bir değişiklik ise, tüpün dışındaki çıkış değişkenlerine, $\xi^+ > 0, \ \xi^- > 0 \ \forall_i$ olmak üzere, tüpün üstünde (ξ^+) veya altında (ξ^-) olmalarına bağlı olarak iki arttıran yapay değişken cezasından birinin verilmesidir:

$$t_i \leq y_i + Є + \xi^+ \tag{5.26}$$

$$t_i \geq y_i - Є - \xi^- \tag{5.27}$$

DVM regresyonu için hata fonksiyonu şu şekilde yazılabilir:

$$C\sum_{i=1}^{L}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\|w\|^2 \tag{5.28}$$

Bu, $\xi^+ \geq 0$, $\xi^- \geq 0\ \forall_i$ ve (5.26) ile (5.27) sınırlandırmaları altında minimize edilmelidir. Bunu yapmak için, $\alpha_i^+ \geq 0$, $\alpha_i^- \geq 0$, $\mu_i^+ \geq 0$, $\mu_i^- \geq 0\ \forall_i$ Lagrange çarpanları kullanılır:

$$\begin{aligned} L_P = C\sum_{i=1}^{L}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\|w\|^2 \\ - \sum_{i=1}^{L}(\mu_i^+\xi_i^+ + \mu_i^-\xi_i^-) \\ - \sum_{i=1}^{L}\alpha_i^+(\epsilon + \xi_i^+ + y_i - t_i) - \sum_{i=1}^{L}\alpha_i^-(\epsilon + \xi_i^- - y_i + t_i) \end{aligned} \tag{5.29}$$

y_i'nin yerine koyarak, w, b, ξ^+ ve ξ^- ye göre türev alınıp sıfıra eşitlenir:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)x_i \tag{5.30}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \tag{5.31}$$

$$\frac{\partial L_P}{\partial \xi_i^+} = 0 \Rightarrow C = \alpha_i^+ + \mu_i^+ \tag{5.32}$$

$$\frac{\partial L_P}{\partial \xi_i^-} = 0 \Rightarrow C = \alpha_i^- + \mu_i^- \tag{5.33}$$

(5.30) ve (5.31)'i yerine koyarak, $\alpha_i^+ \geq 0$, $\alpha_i^- \geq 0\ \forall_i$ olmak üzere, α_i^+ ve α_i^- ye göre L_D maksimize edilir.

$$L_D = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i - \epsilon\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)\left(\alpha_j^+ - \alpha_j^-\right)x_i.x_j \tag{5.34}$$

μ_i^+ ve μ_i^- nin (5.30) ve (5.25) ile birlikte kullanılması, $\alpha_i^+ \leq C$ ve $\alpha_i^- \leq C$ olması anlamına gelir. Bu nedenle şunu bulmak gerekir:

$$\max_{\alpha^+, \alpha^-} \left[\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) t_i - \epsilon \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) - \frac{1}{2} \sum_{i,j} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) x_i . x_j \right] \quad (5.35)$$

öyle ki $0 \leq \alpha_i^+ \leq C$, $0 \leq \alpha_i^- \leq C$ ve $\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) = 0 \ \forall_i$

(5.30)'u (5.25)'te yerine koyarak, yeni y' tahminleri şu kullanılarak bulunabilir:

$$y' = \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) x_i . x' + b \quad (5.36)$$

x_s destek vektörlerinin kümesi S, $0 < \alpha < C$ ve $\xi_i^+ = 0$ veya $\xi_i^- = 0$ olmak üzere, i indisleri bulunarak oluşturulabilir. Bu, şunu verir:

$$b = t_s - \epsilon - \sum_{m \epsilon = S}^{L} (\alpha_m^+ - \alpha_m^-) x_m . x_s \quad (5.37)$$

S kümesindeki bütün i indisleri üzerinden ortalama almak daha iyidir:

$$b = \frac{1}{N_s} \sum_{s \in S} \left[t_s - \epsilon - \sum_{m \epsilon = S}^{L} (\alpha_m^+ - \alpha_m^-) x_m . x_s \right] \quad (5.38)$$

Bir regresyon problemini çözmek üzere bir DVM kullanmak için şunlar yapılır:

- C ve ϵ parametreleri için uygun değerler seçilerek, yanlış sınıflandırmalara ne kadar önem verileceği ve duyarsız kayıp bölgenin ne kadar geniş olacağı belirlenir.
- $0 \leq \alpha_i^+ \leq C$, $0 \leq \alpha_i^- \leq C$ ve $\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) = 0 \ \forall_i$ sınırlandırması altında, $\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) t_i - \epsilon \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) - \frac{1}{2} \sum_{i,j} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) x_i . x_j$ maksimize olacak şekilde α^+ve α^- bulunur. Bu, bir KP çözümleyici kullanılarak yapılır.
- $w = \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) x_i$ hesaplanır.

- $0 < \alpha \leq C$ ve $\xi_i = 0$ olacak şekilde i indisleri bulunarak S destek vektörler kümesi belirlenir.
- $b = \frac{1}{N_s}\sum_{s\in S}[t_i - \epsilon - \sum_{m=1}^{L}(\alpha_i^+ - \alpha_i^-)x_i \,.\, x_m]$ hesaplanır.
- $y' = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\, x_i.x' + b$ hesaplanarak her yeni x' noktası belirlenir (Fletcher 2009).

5.4. Doğrusal Olmayan Destek Vektör Makineleri

Doğrusal olarak ayrılabilir veriye DVM'yi uygulamak için, giriş değişkenlerinin nokta çarpımından bir H matrisi oluşturarak başlanmıştı:

$$H_{ij} = y_i y_j k(x_i, x_j) = x_i.x_j = x_i^T x_j \tag{5.39}$$

$k(x_i, x_j)$, çekirdek fonksiyonları denilen fonksiyon ailesinin bir örneğidir. $k(x_i, x_j) = x_i^T x_j$ doğrusal çekirdek olarak bilinir. Çekirdek fonksiyonlar kümesi, hepsi iki vektörün iç çarpımlarını hesaplamaya dayalı olacak şekilde, (5.40)'ın değişik biçimlerinden oluşur. Bu, fonksiyonlar, potansiyel olarak doğrusal olmayan bir öznitelik dönüştürme fonksiyonu $x \rightarrow \phi(x)$ ile daha yüksek boyutlu bir uzaya değiştirilebilirse, sadece öznitelik uzayındaki dönüştürülen girişlerin iç çarpımlarının, ϕ'nin hesaplanmasına gerek olmadan belirlenmesi gerektiği anlamına gelir.

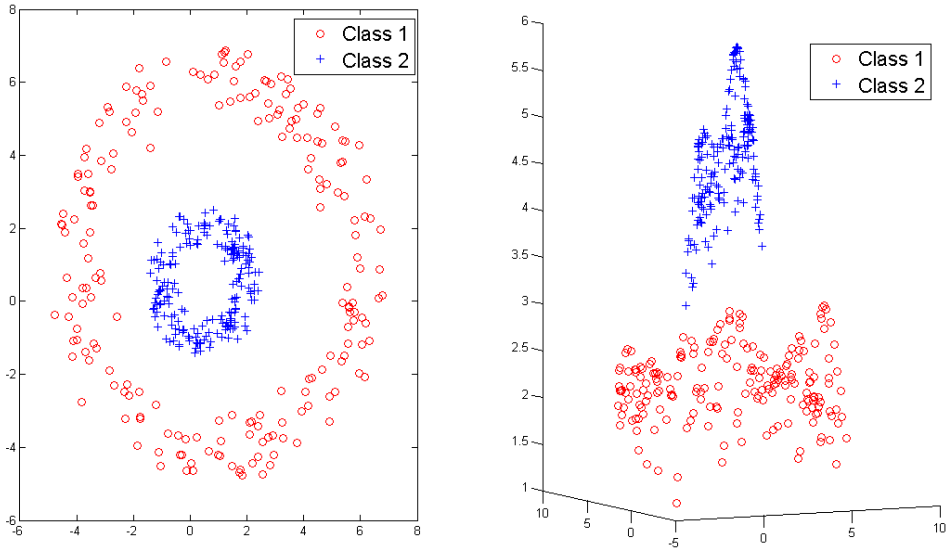
Bu çekirdek hilesinin yararlı olmasının sebebi, uygun bir $x \rightarrow \phi(x)$ dönüştürmesi verildiğinde daha yüksek boyutlu bir öznitelik uzayında olabilecek olan x girişleri uzayında doğrusal olarak ayrılabilir/gerileyebilir olmayan birçok sınıflandırma/regresyon probleminin olmasıdır.

Şekil 5.4'te radyal taban çekirdeği kullanılarak yeniden dönüştürülmüş iki sınıfa ayrılmış veri örneği gösterilmiştir.

Şekil 5.4'e göre, çekirdek şöyle tanımlanırsa;

$$k(x_i, x_j) = e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}$$

Şekil 5.4'ün sol tarafındaki iki boyutlu veri uzayı x'teki doğrusal olarak ayrılabilir olmayan veri kümesi, radyal tabanlı çekirdek olarak bilinen doğrusal olmayan çekirdek fonksiyonu ile üstü kapalı olarak tanımlanan, Şekil 5.4'ün sağ tarafındaki doğrusal olmayan öznitelik uzayında ayrılabilirdir.



Şekil 5.4. Radyal taban çekirdeği kullanılarak yeniden dönüştürülmüş iki sınıfa ayrılmış veri örneği (Fletcher 2009)

Sınıflandırma ve regresyon için diğer popüler çekirdekler polinomsal çekirdek:

$$k(x_i, x_j) = (x_i . x_j + a)^b$$

ve sigmoidal çekirdektir:

$$k(x_i, x_j) = \tanh(ax_i . x_j - b)$$

Burada a ve b çekirdeğin davranışını tanımlayan parametrelerdir.

Ayarlara, dizilere ve hatta müziğe göre davrananlar da dahil olmak üzere birçok çekirdek fonksiyonu vardır. Bu alanla ilgili bu kısa özetin kapsamı dışında da, bir fonksiyonun çekirdek fonksiyonu olarak kabul edilebilmesi için koşullar vardır.

Doğrusal olarak ayrılabilir olmayan veri üzerinde bir sınıflandırma veya regresyon problemini çözmek üzere, ilk olarak, doğrusal olarak ayrılabilir olmayan veriyi doğrusal olarak ayrılabilir olduğu bir öznitelik uzayına dönüştürmesi beklenebilen bir çekirdek ve ilgili parametreler seçilmelidir. Bu, örneğin deneme yanılma yolu ile, deneysel olarak elde edilmelidir. Başlangıç olarak hassas çekirdekler radyal taban, polinomsal ve sigmoidal çekirdeklerdir.

Bu nedenle, ilk adım, çekirdeği seçme ile $x \rightarrow \phi(x)$ dönüştürmesidir.

Daha sonra sınıflandırma için şunlar yapılır:

- $H_{ij} = y_i y_j \phi(x_i).\phi(x_j)$ olmak üzere H oluşturulur.
- C parametresi için uygun bir değer seçilerek yanlış sınıflandırmalara ne kadar önem verileceği belirlenir.
- $0 \leq \alpha_i \leq C \ \forall_i$ ve $\sum_{i=1}^{L} \alpha_i y_i = 0$ sınırlandırması altında $\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha$ maksimize edilecek şekilde α bulunur. Bu, bir KP çözümleyici kullanılarak yapılır.
- $w = \sum_{i=1}^{L} \alpha_i y_i \phi(x_i)$ hesaplanır.
- $0 < \alpha \leq C$ olacak şekilde indisler bulunarak S destek vektörler kümesi belirlenir.
- $b = \frac{1}{N_s} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m \phi(x_m).\phi(x_s) \right)$ hesaplanır.
- $y' = \text{sgn}(w.\phi(x') + b)$ hesaplanarak her yeni x' noktası sınıflandırılır.

Regresyon için de şunlar yapılır:

- C ve Є parametreleri için uygun değerler seçilerek, yanlış sınıflandırmalara ne kadar önem verileceği ve duyarsız kayıp bölgenin ne kadar geniş olacağı belirlenir.
- $0 \leq \alpha_i^+ \leq C$, $0 \leq \alpha_i^- \leq C$ ve $\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) = 0 \ \forall_i$ sınırlandırması altında, $\sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) t_i - Є \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) - \frac{1}{2} \sum_{i,j} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) \phi(x_i).\phi(x_j)$ maksimize olacak şekilde α^+ ve α^- bulunur.
- $w = \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) \phi(x_i)$ hesaplanır.
- $0 < \alpha \leq C$ ve $\xi_i = 0$ olacak şekilde i indisleri bulunarak S destek vektörler kümesi belirlenir.
- $b = \frac{1}{N_s} \sum_{s \in S} [t_i - Є - \sum_{m=1}^{L} (\alpha_i^+ - \alpha_i^-) \phi(x_i).\phi(x_m)]$ hesaplanır.
- $y' = \sum_{i=1}^{L} (\alpha_i^+ - \alpha_i^-) \phi(x_i).\phi(x') + b$ hesaplanarak her yeni x' noktası belirlenir (Fletcher 2009).

6. EN YAKIN K-KOMŞU ALGORİTMASI

Örüntü tanımada, en yakın k-komşu algoritması, öznitelik uzayındaki en yakın eğitim örneklerine dayalı olan nesneleri sınıflandırmak için bir yöntemdir. En yakın k-komşu algoritması, fonksiyonun sadece yerel olarak tahmin edildiği ve bütün hesaplamaların sınıflandırmaya kadar ertelendiği bir örnek tabanlı öğrenme ya da ağır öğrenmedir. En yakın k-komşu algoritması, tüm makine öğrenme algoritmalarının en kolayları arasındadır. k bir pozitif tamsayı ve tipik olarak küçük olmak üzere, bir nesne, en yakın k-komşuları arasında en yaygın olan sınıfa atanarak komşularının çoğunluk oyu ile sınıflandırılır. k=1 ise nesne basitçe en yakın komşusunun sınıfına atanır.

Aynı yöntem, nesnenin, en yakın k-komşularının değerlerinin ortalaması olması için, özellik değerini atayarak, regresyon için kullanılabilir. Daha yakın komşular, ortalamaya, daha uzak komşulardan daha fazla katkıda bulunacak şekilde, komşuların katkılarını eklemek kullanışlı olabilir. Yaygın bir ekleme yöntemi, her komşuya, 1/d'lik bir ağırlık vermektir. Burada d komşuya olan uzaklıktır. Bu yöntem doğrusal interpolasyonun bir genelleştirilmesidir.

Komşular, doğru sınıflandırmanın ya da regresyon durumunda özelliğin değerinin bilindiği bir nesneler kümesinden alınır. Bu, hiçbir belli eğitim aşaması gerekmediği halde, algoritma için eğitim kümesi olarak düşünülebilir. En yakın k-komşu algoritması, verinin yerel yapısına duyarlıdır.

En yakın komşu kuralları aslında üstü kapalı bir şekilde karar verme sınırını hesaplar. Karar verme sınırını açık bir şekilde hesaplamak da mümkündür ve bu, hesapsal karmaşıklık sınır karmaşıklığının bir fonksiyonu olacak şekilde, etkin bir biçimde yapılabilir (Anonim 2012e).

Eğitim örnekleri, her biri bir sınıf etiketine sahip olan, çok boyutlu bir öznitelik uzayındaki vektörlerdir. Algoritmanın eğitim aşaması, sadece öznitelik vektörlerini ve eğitim örneklerinin sınıf etiketlerini depolamaktan oluşur.

Sınıflandırma aşamasında, k kullanıcı tarafından belirlenen bir sabit olmak üzere, bir etiketlenmemiş vektör (bir sorgulama ya da test noktası), bu sorgulama noktasına en yakın k eğitim örnekleri arasında en sık olan etiket atanarak sınıflandırılır.

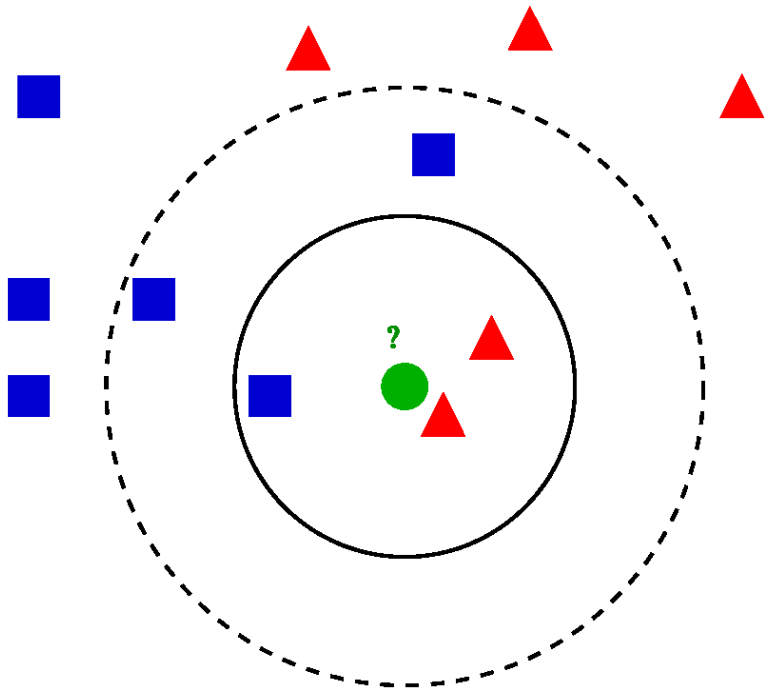
Genellikle Öklit uzaklığı uzaklık ölçütü olarak kullanılır. Ama bu sadece sürekli değişkenlere uygulanabilir. Metin sınıflandırma gibi durumlarda, örtüşme ölçütü ya da Hamming uzaklığı gibi diğer bir ölçüt kullanılabilir. Çoğunlukla, eğer uzaklık ölçütü, büyük

marjin en yakın komşu veya komşuluk bileşenler analizi gibi özel algoritmalar ile öğrenilirse, en yakın k-komşu algoritmasının sınıflandırma doğruluğu önemli ölçüde artırılabilir.

Temel "çoğunluk oylaması" sınıflandırmasının bir dezavantajı, daha sık örnekli sınıflar, komşular sayılarının fazlalığına bağlı olarak hesaplandığında, en yakın k-komşularda ortaya çıkma eğiliminde oldukları için, yeni vektörün tahmininde daha üstün olma eğiliminde olmalarıdır. Bu problemin üstesinden gelmenin bir yolu, test noktasından onun her en yakın k-komşusuna olan uzaklığını dikkate alarak sınıflandırmayı ağırlıklandırmaktır.

En yakın k-komşu algoritması, değişken bant genişliğinin özel bir durumu olarak, birörnek çekirdekli çekirdek yoğunluk balon kestiricisidir.

Şekil 6.1'de gösterilen en yakın k-komşu sınıflandırma örneğinde, daire ile temsil edilen test örneği ya karelerle temsil edilen birinci sınıfa ya da üçgenlerle temsil edilen ikinci sınıfa sınıflandırılmalıdır. Eğer k=3 olursa ikinci sınıfa sınıflandırılır, çünkü iç çemberin içinde 2 adet üçgen ve sadece 1 adet kare vardır. Eğer k=5 olursa birinci sınıfa sınıflandırılır, çünkü dış çemberin içinde 3 kareye karşılık 2 üçgen vardır (Anonim 2012e).



Şekil 6.1. En yakın k-komşu sınıflandırma örneği (Anonim 2012e)

6.1. Parametre Seçimi

En iyi k seçimi veriye bağlıdır. Büyük k değerleri sınıflandırma üzerindeki gürültü etkisini azaltır fakat sınıflar arasındaki sınırların arasındaki uzaklığı azaltır. İyi bir k seçimi çapraz sağlama gibi çeşitli buluşsal tekniklerle yapılabilir. Sınıfın, k=1 olduğu durumdaki en yakın eğitim örneğinin sınıfı olarak tahmin edildiği özel duruma en yakın komşu algoritması denir.

En yakın k-komşu algoritmasının doğruluğu, gürültünün ve yersiz özniteliklerin varlığı ile veya eğer öznitelik ölçekleri özniteliklerin önem dereceleri ile tutarlı değilse, ciddi şekilde azalabilir. Sınıflandırmayı iyileştirmek için özniteliklerin seçimi ve ölçeklendirilmesi konusunda bugüne kadar yapılan çalışmalarda çok çaba harcanmıştır. Özellikle popüler olan bir yaklaşım, öznitelik ölçeklendirilmesinin eniyilenmesi için evrimsel algoritmaların kullanımıdır. Diğer bir popüler yaklaşım ise, eğitim verisinin eğitim sınıfları ile karşılıklı bilgisi ile öznitelikleri ölçeklendirmektir.

İkili yani iki sınıflı sınıflandırma problemlerinde, bağlı oylardan kaçınıldığı için, k'yı tek sayı olarak seçmek yararlı olur. Burada deneysel olarak en iyi k'yı seçmenin popüler bir yolu da önyükleme yöntemidir (Anonim 2012e).

6.2. Özellikler

Algoritmanın denenmemiş sürümünü, test örneklerinden olan uzaklıkları hesaplayarak, bütün yüklenmiş vektörlere uygulamak kolaydır, fakat özellikle eğitim kümesinin boyutu büyüdükçe daha yoğun hesaplamalar içerir. Yıllardır birçok en yakın komşu araştırma algoritması önerilmiştir. Bunlar genellikle, yapılan uzaklık hesaplamalarının sayısını azaltmaya çalışmışlardır. Uygun bir en yakın komşu araştırma algoritmasının kullanımı, en yakın k-komşu yöntemini, büyük veri kümeleri için bile hesaplamalar açısından kolaylaştırır.

En yakın komşu algoritmasının tutarlılık açısından bazı güçlü sonuçları vardır. Veri miktarı sonsuza yaklaştıkça, algoritmanın, Bayes hata oranının iki katından daha fazla olmayan bir hata oranı vermesi garantidir. Bayes hata oranı, verinin dağılımı verildiğinde minimum elde edilebilir hata oranıdır. k'nın, veri noktalarının sayısının bir fonksiyonu olarak arttığı bazı k değerleri için, en yakın k-komşunun Bayes hata oranına yaklaşması garantidir. Yakınlık grafikleri kullanarak, en yakın k-komşu yöntemleri için bazı iyileştirmelerin yapılması mümkündür (Anonim 2012e).

6.3. Sürekli Değişkenlerin Hesaplanması

En yakın k-komşu algoritması aynı zamanda, sürekli değişkenlerin hesaplanmasında kullanmak üzere uyarlanabilir. Bu tür uygulamalardan biri, en yakın çok değişkenli k-komşuların bir ters uzaklık ağırlıklı ortalamasını kullanır. Bu algoritma şu şekilde işler:

1. Örneklenmiş olan noktalardan hedef noktaya olan Öklit ya da Mahalanobis uzaklığı hesaplanır.
2. Hesaplanan uzaklıklar dikkate alınarak örnekler sıralanır.
3. Çapraz sağlama tekniği ile yapılan hata kareleri karekökünün ortalamasına (HKKO) dayalı buluşsal olarak en iyi k-komşu seçilir.
4. En yakın çok değişkenli k-komşular ile bir ters uzaklık ağırlıklı ortalaması hesaplanır.

Çoğu veri kümesi için en iyi k, 10 ya da daha fazladır. Bu, en yakın 1-komşuluktan daha iyi sonuçlar üretir. Her bir en yakın k-noktalar sınıfının ya da regresyon problemlerindeki değerin çarpıldığı ağırlıkların, bu nokta ile sınıfın tahmin edileceği nokta arasındaki uzaklığın tersi ile orantılı olduğu bir ağırlıklı en yakın k-komşuluğun kullanımı sonuçları önemli ölçüde iyileştirir (Dasarathy 1991, Anonim 2012e).

7. BENZETİM

Bu tezde açık kaynak kodlu programlar kullanılarak öznitelik elde etme ve sınıflandırma işlemleri gerçekleştirilmiştir. Yapılan benzetimlerde, pyeeg (Python + EEG/MEG) python program paketi (Bao ve ark. 2011), mlpy (Machine Learning Python) python program paketi (Albanese ve ark. 2012) ve sklearn (scikit-learn) python program paketi (Pedregosa ve ark. 2011) kullanılmıştır (Ek 1, Ek 2, Ek 3).

Elde edilen sınıflandırma doğruluk oranları Çizelge 7.1, Çizelge 7.2, Çizelge 7.3, Çizelge 7.4, Çizelge 7.5 ve Çizelge 7.6'da gösterilmiştir.

Çizelge 1. mlpy python program paketi ile zaman düzleminde elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri		
	YSA	DVM	En yakın 1-komşu
Spektral Entropi	80,00	81,00	79,00
Hjorth Hareketliliği	21,00	21,00	21,00
Hjorth Karmaşıklığı	21,00	80,00	80,00
TDA Entropisi	21,00	21,00	21,00
Fisher Bilgisi	80,00	80,00	23,00
Yaklaşık Entropi	22,00	21,00	80,00
Hurst Katsayısı	80,00	80,00	79,00
Örnek Entropisi	73,00	79,00	70,00
Petrosian Fraktal Boyutu	21,00	21,00	21,00
Katz Fraktal Boyutu	80,00	80,00	80,00
Sevcik Fraktal Boyutu	21,00	21,00	80,00
Hjorth Fraktal Boyutu	80,00	80,00	21,00

Çizelge 2. mlpy python program paketi ile alt yan bant dalgacık katsayılarından elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri		
	YSA	DVM	En yakın 1-komşu
Spektral Entropi	99,13	98,10	100,00
Hjorth Hareketliliği	81,20	83,38	84,26
Hjorth Karmaşıklığı	100,00	100,00	100,00
TDA Entropisi	99,85	100,00	91,40
Fisher Bilgisi	99,85	100,00	91,84
Yaklaşık Entropi	100,00	100,00	99,13
Hurst Katsayısı	96,50	95,92	100,00
Örnek Entropisi	100,00	100,00	99,27
Petrosian Fraktal Boyutu	87,17	85,13	80,47
Katz Fraktal Boyutu	80,32	80,32	80,47
Sevcik Fraktal Boyutu	99,85	99,85	98,40
Hjorth Fraktal Boyutu	100,00	100,00	100,00

Çizelge 3. mlpy python program paketi ile üst yan bant dalgacık katsayılarından elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri		
	YSA	DVM	En yakın 1-komşu
Spektral Entropi	100,00	100,00	100,00
Hjorth Hareketliliği	100,00	100,00	100,00
Hjorth Karmaşıklığı	100,00	100,00	100,00
TDA Entropisi	100,00	100,00	100,00
Fisher Bilgisi	100,00	100,00	100,00
Yaklaşık Entropi	100,00	100,00	100,00
Hurst Katsayısı	100,00	100,00	100,00
Örnek Entropisi	100,00	100,00	100,00
Petrosian Fraktal Boyutu	100,00	100,00	100,00
Katz Fraktal Boyutu	100,00	100,00	100,00
Sevcik Fraktal Boyutu	100,00	100,00	100,00
Hjorth Fraktal Boyutu	100,00	100,00	100,00

Çizelge 4. sklearn python program paketi ile zaman düzleminde elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri					
	DVM	DVR	En yakın 1-komşu	En yakın 2-komşu	En yakın 3-komşu	Radyal en yakın komşu
Spektral Entropi	79,88	73,94	100,00	86,00	87,00	81,00
Hjorth Hareketliliği	79,88	73,91	100,00	87,00	88,00	80,00
Hjorth Karmaşıklığı	99,85	90,00	100,00	86,00	86,00	100,00
TDA Entropisi	79,88	74,14	100,00	85,00	85,00	81,00
Fisher Bilgisi	79,88	73,93	100,00	84,00	85,00	80,00
Yaklaşık Entropi	79,88	73,96	100,00	85,00	87,00	81,00
Hurst Katsayısı	79,88	73,95	100,00	84,00	85,00	82,00
Örnek Entropisi	80,32	74,64	100,00	85,00	86,00	84,00
Petrosian Fraktal Boyutu	79,88	73,92	100,00	87,00	86,00	80,00
Katz Fraktal Boyutu	79,88	74,24	100,00	85,00	85,00	81,00
Sevcik Fraktal Boyutu	79,88	74,24	100,00	85,00	85,00	81,00
Hjorth Fraktal Boyutu	79,88	74,09	100,00	87,00	86,00	82,00

Çizelge 5. sklearn python program paketi ile alt yan bant dalgacık katsayılarından elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri					
	DVM	DVR	En yakın 1-komşu	En yakın 2-komşu	En yakın 3-komşu	Radyal en yakın komşu
Spektral Entropi	79,88	73,92	100,00	85,00	87,00	81,00
Hjorth Hareketliliği	79,88	73,91	100,00	85,00	85,00	80,00
Hjorth Karmaşıklığı	99,85	89,92	100,00	86,00	85,00	100,00
TDA Entropisi	79,88	74,09	100,00	84,00	85,00	81,00
Fisher Bilgisi	79,88	73,92	100,00	84,00	85,00	80,00
Yaklaşık Entropi	79,88	73,92	100,00	86,00	85,00	81,00
Hurst Katsayısı	79,88	73,94	100,00	84,00	84,00	81,00
Örnek Entropisi	80,32	75,13	100,00	85,00	84,00	95,00
Petrosian Fraktal Boyutu	79,88	73,92	100,00	87,00	85,00	80,00
Katz Fraktal Boyutu	79,88	74,24	100,00	85,00	85,00	81,00
Sevcik Fraktal Boyutu	79,88	74,22	100,00	84,00	86,00	81,00
Hjorth Fraktal Boyutu	79,88	74,07	100,00	86,00	86,00	82,00

Çizelge 6. sklearn python program paketi ile üst yan bant dalgacık katsayılarından elde edilen özniteliklere göre sınıflandırma sonuçları (%)

Öznitelik belirleme yöntemleri	Sınıflandırma yöntemleri					
	DVM	DVR	En yakın 1-komşu	En yakın 2-komşu	En yakın 3-komşu	Radyal en yakın komşu
Spektral Entropi	79,88	73,92	100,00	86,00	84,00	80,00
Hjorth Hareketliliği	79,88	73,91	100,00	86,00	86,00	80,00
Hjorth Karmaşıklığı	87,03	85,01	100,00	84,00	86,00	100,00
TDA Entropisi	79,88	74,31	100,00	86,00	87,00	81,00
Fisher Bilgisi	79,88	73,91	100,00	84,00	85,00	80,00
Yaklaşık Entropi	79,88	73,95	100,00	85,00	87,00	81,00
Hurst Katsayısı	79,88	73,93	100,00	85,00	86,00	80,00
Örnek Entropisi	80,32	74,53	100,00	85,00	86,00	82,00
Petrosian Fraktal Boyutu	79,88	73,95	100,00	87,00	89,00	80,00
Katz Fraktal Boyutu	79,88	74,26	100,00	85,00	86,00	81,00
Sevcik Fraktal Boyutu	79,88	74,41	100,00	87,00	85,00	81,00
Hjorth Fraktal Boyutu	79,88	74,08	100,00	86,00	87,00	81,00

8. SONUÇLAR

EEG işaretlerine ait spektral entropi, Hjorth parametreleri, tekil değer ayrıştırma entropisi, Fisher bilgisi, yaklaşık entropi, Hurst katsayısı, örnek entropisi, Petrosian fraktal boyutu, Katz fraktal boyutu, Sevcik fraktal boyutu ve Hjorth fraktal boyutu hesaplanarak, yapay sinir ağları, destek vektör makineleri ve en yakın k-komşu algoritması ile sınıflandırılmıştır. Aynı işlemler EEG işaretlerinin dalgacık katsayıları için de tekrar edilmiştir. Böylelikle her bir sınıflayıcı ve parametre için en iyi durumlar elde edilmeye çalışılmıştır.

mlpy python program paketi kullanılarak YSA için en iyi sonuç, alt yan bant dalgacık katsayılarından elde edilen Hjorth karmaşıklığı, yaklaşık entropi, örnek entropisi, Hjorth fraktal boyutu ve üst yan bant dalgacık katsayılarından elde edilen bütün öznitelikler kullanılarak elde edilmiştir.

mlpy python program paketi kullanılarak DVM için en iyi sonuç, alt yan bant dalgacık katsayılarından elde edilen Hjorth karmaşıklığı, TDA entropisi, Fisher bilgisi, yaklaşık entropi, örnek entropisi, Hjorth fraktal boyutu ve üst yan bant dalgacık katsayılarından elde edilen bütün öznitelikler kullanılarak elde edilmiştir.

mlpy python program paketi kullanılarak en yakın 1-komşu için en iyi sonuç, alt yan bant dalgacık katsayılarından elde edilen spektral entropi, Hjorth karmaşıklığı, Hurst katsayısı, Hjorth fraktal boyutu ve üst yan bant dalgacık katsayılarından elde edilen bütün öznitelikler kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak DVM için en iyi sonuç, zaman düzleminde elde edilen Hjorth karmaşıklığı ve alt yan bant dalgacık katsayılarından elde edilen Hjorth karmaşıklığı kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak DVR (Destek Vektör Regresyonu) için en iyi sonuç, zaman düzleminde elde edilen Hjorth karmaşıklığı kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak en yakın 1-komşu için en iyi sonuç, zaman düzleminde elde edilen bütün öznitelikler; alt yan bant dalgacık katsayılarından elde edilen bütün öznitelikler ve üst yan bant dalgacık katsayılarından elde edilen bütün öznitelikler kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak en yakın 2-komşu için en iyi sonuç, zaman düzleminde elde edilen Hjorth hareketliliği, Petrosian fraktal boyutu, Hjorth fraktal boyutu; alt yan bant dalgacık katsayılarından elde edilen Petrosian fraktal boyutu ve üst yan bant

dalgacık katsayılarından elde edilen Petrosian fraktal boyutu, Sevcik fraktal boyutu kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak en yakın 3-komşu için en iyi sonuç üst yan bant dalgacık katsayılarından elde edilen Petrosian fraktal boyutu kullanılarak elde edilmiştir.

sklearn python program paketi kullanılarak radyal en yakın komşu için en iyi sonuç, zaman düzleminde elde edilen Hjorth karmaşıklığı, alt yan bant dalgacık katsayılarından elde edilen Hjorth karmaşıklığı ve üst yan bant dalgacık katsayılarından elde edilen Hjorth karmaşıklığı kullanılarak elde edilmiştir.

9. KAYNAKLAR

Akansu A N, Haddad R A (1992). Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets, Academic Press.

Albanese D, Visintainer R, Merler S, Riccadonna S, Jurman G, Furlanello C (2012). mlpy: Machine Learning Python, 4p. http://arxiv.org/pdf/1202.6548.pdf (erişim tarihi, 15.05.2012).

Altıntaş E (2011). Yapay Sinir Ağları (Artificial Neural Networks), http://www.yapay-zeka.org/modules/wiwimod/index.php?page=ANN (erişim tarihi, 03.04.2012).

Amon C (2012). Polysomnography, Medscape Reference (Drugs, Disesases and Procedures), http://emedicine.medscape.com/article/1188764-overview (erişim tarihi, 02.05.2012).

Anonim (2010). Wavelet Functions, HSCTechnicalWiki, http://wiki.hsc.com/wiki/Main/DiscreteWaveletTransform (erişim tarihi, 02.06.2012)

Anonim (2011a). Elektroensefalografi, Wikipedia, http://tr.wikipedia.org/wiki/Elektroensefalografi (erişim tarihi, 24.04.2012).

Anonim (2011b). CHB-MIT Scalp EEG Database, Physiobank Archive Index, http://physionet.org/physiobank/database/chbmit/ (erişim tarihi, 01.05.2012).

Anonim (2011c). Clinical Cognitive Neuroscience and Neuropsychology Laboratory, Brigham Young University, https://cogneuro.byu.edu/Pages/FAQ.aspx (erişim tarihi, 23.05.2012)

Anonim (2012a). European Data Format, Wikipedia, http://en.wikipedia.org/wiki/European_Data_Format (erişim tarihi, 02.05.2012).

Anonim (2012b). Fractal Dimension, Wikipedia, http://en.wikipedia.org/wiki/Fractal_dimension (erişim tarihi, 07.05.2012).

Anonim (2012c). Fraktallar, Fraktal Nedir, http://matlab.s5.com/fraktal.htm (erişim tarihi, 04.03.2012).

Anonim (2012d). Haar Wavelet, Wikipedia, http://en.wikipedia.org/wiki/Haar_wavelet (erişim tarihi, 03.06.2012).

Anonim (2012e). k-Nearest Neighbor Algorithm, Wikipedia, http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm (erişim tarihi, 22.04.2012).

Banchoff T F (1990). Dimension, On the Shoulders of Giants: New Approaches to Numeracy, Chapter 2, Ed:Steen LA, National Academy Press, Washington, D.C.

Bao F S, Lie D Y, Zhang Y (2008). A New Approach to Automated Epileptic Diagnosis Using EEG and Probabilistic Neural Network, ICTAI '08 Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence - Volume 2:1-5.

Bao F S, Liu X, Zhang C (2011). PyEEG: AnOpen Source PythonModule for EEG/MEG Feature Extraction, Computational Intelligence and Neuroscience, Volume 2011, Article ID 406391, 7 p.

Boashash B, Boubchir L, Azemi G (2011). Time-Frequency Signal and Image Processing of Non-stationary Signals with Application to the Classification of Newborn EEG Abnormalities, IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 14-17 Dec. 2011, pp:120-129.

Bozer N (2012). Yapay Sinir Ağlarında Sınıflandırma, http://www.docstoc.com/docs/113940547/YAPAY-SINIR-AGLARINDA-SINIFLANDIRMA (erişim tarihi 22.04.2012).

Chaovalitwongse W A, Pottenger R S, Wang S, Fan Y J, Iasemidis L D (2011). Pattern-and Network-Based Classification Techniques for Multichannel Medical Data Signals to Improve Brain Diaognosis, IEEE Transactions on Systems, Man, and Cybernetics—Part a: Systems and Humans, vol. 41(5):977-988.

Cinsdikici M (2012). Yapay Sinir Ağları, http://ube.ege.edu.tr/~cinsdiki/UBI521/Chapter-1/cinsdikici-neural-net-giris.pdf (erişim tarihi, 15.04.2012).

Cortes C, Vapnik V (1995). Support Vector Networks, Machine Learning 20:1-25.

Coşkun M, İstanbullu A (2012). EEG İsaretlerinin FFT ve Dalgacık Dönüşümü ile Analizi, http://ab.org.tr/ab12/bildiri/91.pdf (erişim tarihi, 30.05.2012).

Çetin M (2012). İleri Beslemeli Yapay Sinir Ağlarında Backpropagation (Geriye Yayılım) Algoritmasının Sezgisel Yaklaşımı, http://ab.org.tr/ab06/sunum/8.ppt (erişim tarihi 09.04.2012).

Çınar E, Şahin F (2010). A Study of Recent Classification Algorithms and a Novel Approach for EEG Data Classification, IEEE International Conference on Systems Man and Cybernetics (SMC), 10-13 Oct. 2010, pp:3366-3372.

Dasarathy B V (1991). Nearest Neighbor (NN) Norms-NN Pattern Classification Techniques, Los Alamitos, CA: IEEE Computer Society Press.

Erdoğmuş P, Pekçakar A (2009). Dalgacık Dönüşümü ile EKG Sinyallerinin Özellik Çıkarımı ve Yapay Sinir Ağları ile Sınıflandırılması, 5. Uluslararası İleri Teknolojiler Sempozyumu (IATS'09), 13-15 Mayıs 2009, Karabük, Türkiye, s:1-3.

Esteller R, Vachtsevans G, Echauz J, Litt B (2001). A Comparison of Waveform Fractal Dimension Algorithms, in; IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications 48(2):177-183.

Faul S, Connolly, S, Marnane W, Lightbody G (2005). Chaos Theory Analysis Of The Newborn EEG - İs İt Worth The Wait? IEEE International Workshop on Intelligent Signal Processing, 1 Sept. 2005, pp: 381 – 386.

Fausett L (1994). Fundamentals of Neural Networks, Architectures Algorithms and Applications, Prentice-Hall, Inc., New Jersey.

Fell J, Roschke J (1996). Discrimination of Sleep Stages: A Comparison between Spectral and Nonlinear EEG Mesures, Electroencephalogr. Clin. Neurophsyiol. 98:401-410.

Fletcher T (2009). Support Vector Machines Explained, http://www.tristanfletcher.co.uk/SVM%20Explained.pdf (erişim tarihi, 18.05.2012).

Forney E M, Anderson C W (2011). Classification of EEG During Imagined Mental Tasks by Forecasting with Elman Recurrent Neural Networks, The International Joint Conference on Neural Networks, July 31 2011-Aug. 5 2011, pp:2749 – 2755.

Fukuda O, Tsuji T, Kaneko M (1995). Pattern Classification of EEG Signals Using a Log-Linearized Gaussian Mixture Neural Network, Proceedings IEEE International Conference on Neural Networks, vol.5, pp:2479 – 2484.

Geetha G, Geethalakshmi S N (2011). Detecting Epileptic Seizures Using Electroencephalogram: A New and Optimized Method for Seizure Classification using Hybrid Extreme Learning Machine, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05978923 (erişim tarihi,10.07.1012)

Goh C, Hamadicharef B, Henderson G T, Ifeachor E C (2005). Comparison of Fractal Dimension Algorithms for the Computation of EEG Biomarkers for Dementia, in; Prceedings, 2nd International Conference on Computational Intelligence in Medicine and Healthcare, pp:464-471.

Goldberger A L, Amaral L A N, Glass L, Hausdorff J M, Ivanov P C, Mark R G, Mietus J E, Moody G B, Peng C K, Stanley H E (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals, Circulation; 101(23):E215-20.

Grajski K A, Breiman L, Prisco G V, Freeman W J (1986). Classification of EEG Spatial Patterns with a Tree-Structured Methodology: CART, IEEE Transactions on Biomedical Engineering, Vol.: BME-33 (12):1076 – 1086.

Haas G M (2012). BFOIT Introduction to Computer Programming, http://www.bfoit.org/itp/Recursion.html (erişim tarihi, 15.03.2012).

Han M, Sun L (2010). EEG Signal Classification for Epilepsy Diagnosis based on AR Model and RVM, International Conference on Intelligent Control and Information Processing (ICICIP), 13-15 Aug. 2010, pp:134 – 139.

Haykin S (1994). Neural Networks:A Comprehensive Foundation, Prentice Hall.

Isaksson A, Wennberg A, Zetterberg L H (1981). Computer Analysis of EEG Signals with Parametric Models, Proceedings of th IEEE 69(4):451-461.

Kakıcı A (2009a). Yapay Sinir Ağlarına Giriş, http://www.ahmetkakici.com/yazilim/yapay-sinir-aglarina-giris (erişim tarihi, 06.03.2012).

Kakıcı A (2009b). Yapay Sinir Ağlarının Sınıflandırılması, http://www.ahmetkakici.com/yapay-sinir-aglari/yapay-sinir-aglarinin-siniflandirilmasi (erişim tarihi, 20.04.2012).

Kannathal N, Choo M L, Acharya U R, Sadavisan P K (2005). Entropies for Detection of Epilepsy in EEG, Computer Methods and Programs in Biomedicine 80:187-194.

Katz M J (1988). Fractals and the Analysis of Waveforms, Comput. Biol. Med. 18:145-156.

Kemp B, Olivan J (2003). European Data Format 'Plus' (EDF+), an EDF alike Standard Format for the Exchange of Physiological Data, Clinical Neurophysiology 114: 1755–1761.

Khorshidtalab A, Salami M J E (2011). EEG Signal Classification for Real-Time Brain-Computer Interface Applications: A Review, 4th International Conference on Mechatronics (ICOM), 17-19 May 2011, Kuala Lumpur, Malaysia, pp:1-7.

Ko K E,Sim K B (2011). An EEG Signals Classification System Using Optimized Adaptive Neuro-Fuzzy Inference Model Based on Harmony Search Algorithm, 11th International Conference on Control, Automation and Systems (ICCAS), pp:1457 – 1461.

Lee D T L, Yamamoto A (1994). Wavelet Analysis: Theory and Applications, Hewlett-Packard Journal, December 1994, pp:44-59.

Li ve ark. (2009). A Prior Neurophysiologic Knowledge Free Tensor-Based Scheme for Single Trial EEG Classification, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 17(2):107-15.

Liao X, Yao D, Wu D, Li C (2007). Combining Spatial Filters for the Classification of Single-Trial EEG in a Finger Movement Task, IEEE Transactions on Biomedical Engineering, vol. 54(5):821-831.

Lu H, Eng H L,Guan C, Plataniotis K N,Venetsanopoulos A N (2010). Regularized Common Spatial Pattern with Aggregation for EEG Classification in Small-Sample Setting, Transactions on Biomedical Engineering 57(12):1-10.

Malmivuo J, Plonsey R (1995). Bioelectromagnetism, Principles and Applications of Bioelectric and Biomagnetic Field, New York Oxford Unıversıty Press, 471 p.

Mandelbrot B B (1982). The Fractal Geometry of Nature (New York: Freeman).

Minsky M, Papert S (1969). Perceptrons. MIT Press, Cambridge, MA.

Moody G B (2012). Approximate Entropy (ApEn), http://physionet.org/physiotools/ApEn (erişim tarihi, 05.06.2012).

Murugavel A S M, Ramakrishnan S, Balasamy K, Gopalagrishnan T (2011). Lyapunov Features based EEG Signal Classification by Multi-Class SVM, 2011 World Congress on Information and Communication Technologies, pp:197-201.

Naderi M A, Nasab H M (2010). Analysis and Classification of EEG Signals using Spectral Analysis and Recurrent Neural Networks, Proceedings of the 17th Iranian Conference of Biomedical Engineering, 3-4 November 2010, pp:1-4.

Oh C, Kim M S, Lee J J (2006). EEG Signal Classification Based on PCA and NN, . International Joint Conference SICE-ICASE, 18-21 Oct. 2006., pp:1848-1851.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E, (2011). "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research,12:2825-2830.

Petrosian A (1995). Kolmogorov Complexity of Finite Sequence and Recognation of Different Preictal EEG Patterns, in; Proc. of the Eight IEEE Symposium on Computer-Based Medical Systems, pp:1-5.

Polychronaki G E, Ktonas P Y, Gatzonis S, Siatouni A, Asvestas P A, Tsekou H, Sakas D, Nikita K S (2010). Comparison of Fractal Dimension Estimation Algorithms for Epileptic Seizure Unset Detection, Journal of Neural Engineering 7:1-18.

Racine R (2011). Estimating the Hurst Exponent, Bachelor Thesis, MOSAIC Group, Prof. Ivo F. Sbalzarini, ETH Zurich, http://www.mosaic.ethz.ch/research/docs/Racine2011.pdf (erişim, 10.06.1012).

Roberts S J, Penny W, Rezek I (1998). Temporal and Spatial Complexity Measures for Eeg Based Brain-Computer Interfacing, Medical & Biological Engineering & Computing, 37(1):93–99.

Rumelhart D E, Hinton G E, Williams R J (1986). Parallel Distributed Processing, Vol.1, MIT Press, Cambridge, MA.

Schmidt A (2000). Multilayer Neural Network, http://www.teco.uni-karlsruhe.de/~albrecht/neuro/html/node18.html (erişim tarihi, 26.03.2012).

Sevcik C (1998). A Procedure to Estimate the Fractal Dimension of Waveforms, Complexity International , 5:1-19.

Seymen H O (2007). Elektroensefalogram (EEG), http://www.ctf.edu.tr/ctffizyo/oktayseymen/dersler/EEG2007W.pdf (erişim tarihi, 02.05.2012).

Shannon C E (1948). A Mathematical Theory of Communication, Bell Syst.Tech.J.27:379-423.

Skinner B T, Nguyen H T, Liu D K (2007). Classification of EEG Signals Using a Genetic-Based Machine Learning Classifier, Proceedings of the 29th Annual International Conference of the IEEE EMBS, August 23-26, 2007, Cité Internationale, Lyon, France, pp:3120-3123.

Stastny J, Sovka P, Stancak A (2001). EEG Signal Classification, Engineering in Medicine and Biology Society, Proceedings of the 23rd Annual International Conference of the IEEE, vol.2, pp: 2020 – 2023.

Vrhovec J (2009). Evaluating the Progress of the Labour with Sample Entropy Calculated from the Uterine EMG Activity, Electrotechnical Review 76(4): 165-170.

Vrocher III D, Lowell M J (2005). Electroencephalography (EEG), http://www.emedicinehealth.com/electroencephalography_eeg/page10_em.htm (erişim tarihi, 02.05.2012).

Xiao D, Mu Z, Hu J (2009). A Linear Discrimination Method used in Motor Imagery EEG Classification, ICNC '09. Fifth International Conference on Natural Computation, 14-16 Aug. 2009, Vol.2: 94 – 98.

EKLER

EK 1

pyeeg Python Dosyası

```
"""Copyleft 2010 Forrest Sheng Bao http://fsbao.net
PyEEG, a Python module to extract EEG features, v 0.02_r2
Project homepage: http://pyeeg.org
**Data structure**
PyEEG only uses standard Python and numpy data structures,
so you need to import numpy before using it.
For numpy, please visit http://numpy.scipy.org
**Naming convention**
I follow "Style Guide for Python Code" to code my program
http://www.python.org/dev/peps/pep-0008/
Constants: UPPER_CASE_WITH_UNDERSCORES, e.g., SAMPLING_RATE,
LENGTH_SIGNAL.
Function names: lower_case_with_underscores, e.g., spectrum_entropy.
Variables (global and local): CapitalizedWords or CapWords, e.g., Power.
If a variable name consists of one letter, I may use lower case, e.g., x, y.
Functions listed alphabetically
-------------------------------------------------
"""
from numpy.fft import fft
from numpy import zeros, floor, log10, log, mean, array, sqrt, vstack, cumsum, \
                                        ones, log2, std
from numpy.linalg import svd, lstsq
import time
######################### Functions contributed by Xin Liu ##################
def hurst(X):
        """ Compute the Hurst exponent of X. If the output H=0.5,the behavior
        of the time-series is similar to random walk. If H<0.5, the time-series
        cover less "distance" than a random walk, vice verse.
```

```
        Parameters
                ----------
                X
                        list
                        a time series
                Returns
                -------
                H
        float
                Hurst exponent
                Examples
                --------
                >>> import pyeeg
                >>> from numpy.random import randn
                >>> a = randn(4096)
                >>> pyeeg.hurst(a)
                >>> 0.5057444

"""
N = len(X)
T = array([float(i) for i in xrange(1,N+1)])
Y = cumsum(X)
Ave_T = Y/T
S_T = zeros((N))
R_T = zeros((N))
for i in xrange(N):
        S_T[i] = std(X[:i+1])
        X_T = Y - T * Ave_T[i]
        R_T[i] = max(X_T[:i + 1]) - min(X_T[:i + 1])
        R_S = R_T / S_T
        R_S = log(R_S)
        n = log(T).reshape(N, 1)
        H = lstsq(n[1:], R_S[1:])[0]
        return H[0]
########################## Begin function definitions ######################
def embed_seq(X,Tau,D):
```

"""Build a set of embedding sequences from given time series X with lag Tau
and embedding dimension DE. Let X = [x(1), x(2), ... , x(N)], then for each
i such that 1 < i < N - (D - 1) * Tau, we build an embedding sequence,
Y(i) = [x(i), x(i + Tau), ... , x(i + (D - 1) * Tau)]. All embedding
sequence are placed in a matrix Y.
Parameters

X
	list
	a time series
Tau
	integer
	the lag or delay when building embedding sequence
D
	integer
	the embedding dimension
Returns

Y
	2-D list
	embedding matrix built
Examples

>>> import pyeeg
>>> a=range(0,9)
>>> pyeeg.embed_seq(a,1,4)
array([[0., 1., 2., 3.],
    [1., 2., 3., 4.],
    [2., 3., 4., 5.],
    [3., 4., 5., 6.],
    [4., 5., 6., 7.],
    [5., 6., 7., 8.]])
>>> pyeeg.embed_seq(a,2,3)
array([[0., 2., 4.],
    [1., 3., 5.],
    [2., 4., 6.],

```
        [ 3.,  5.,  7.],
        [ 4.,  6.,  8.]])
    >>> pyeeg.embed_seq(a,4,1)
    array([[ 0.],
        [ 1.],
        [ 2.],
        [ 3.],
        [ 4.],
        [ 5.],
        [ 6.],
        [ 7.],
        [ 8.]])

    """
    N =len(X)
    if D * Tau > N:
            print "Cannot build such a matrix, because D * Tau > N"
            exit()
    if Tau<1:
            print "Tau has to be at least 1"
            exit()
    Y=zeros((N - (D - 1) * Tau, D))
    for i in xrange(0, N - (D - 1) * Tau):
            for j in xrange(0, D):
                    Y[i][j] = X[i + j * Tau]
    return Y


def in_range(Template, Scroll, Distance):
    """Determines whether one vector is the the range of another vector.
    The two vectors should have equal length.
    Parameters
    ----------
    Template
            list
            The template vector, one of two vectors being compared
```

```
Scroll
        list
        The scroll vector, one of the two vectors being compared
D
        float
        Two vectors match if their distance is less than D
Bit
Notes
-------
The distance between two vectors can be defined as Euclidean distance
according to some publications.
The two vector should of equal length
"""
for i in range(0,  len(Template)):
                if abs(Template[i] - Scroll[i]) > Distance:
                    return False
return True
""" Desperate code, but do not delete
def bit_in_range(Index):
        if abs(Scroll[Index] - Template[Bit]) <=  Distance :
                print "Bit=", Bit, "Scroll[Index]", Scroll[Index], "Template[Bit]",\
                 Template[Bit], "abs(Scroll[Index] - Template[Bit])",\
                 abs(Scroll[Index] - Template[Bit])
                return Index + 1 # move

Match_No_Tail = range(0, len(Scroll) - 1) # except the last one
#       print Match_No_Tail
# first compare Template[:-2] and Scroll[:-2]
for Bit in xrange(0, len(Template) - 1): # every bit of Template is in range of Scroll
        Match_No_Tail = filter(bit_in_range, Match_No_Tail)
        print Match_No_Tail
# second and last, check whether Template[-1] is in range of Scroll and

#       Scroll[-1] in range of Template
# 2.1 Check whether Template[-1] is in the range of Scroll
Bit = - 1
```

```
        Match_All =  filter(bit_in_range, Match_No_Tail)
        # 2.2 Check whether Scroll[-1] is in the range of Template
        # I just write a  loop for this.
        for i in Match_All:
                if abs(Scroll[-1] - Template[i] ) <= Distance:
                        Match_All.remove(i)
        return len(Match_All), len(Match_No_Tail)
        """
def bin_power(X,Band,Fs):
        """Compute power in each frequency bin specified by Band from FFT result of
        X. By default, X is a real signal.
        Note
        -----
        A real signal can be synthesized, thus not real.
        Parameters
        -----------
        Band
                list
                boundary frequencies (in Hz) of bins. They can be unequal bins, e.g.
                [0.5,4,7,12,30] which are delta, theta, alpha and beta respectively.
                You can also use range() function of Python to generate equal bins and
                pass the generated list to this function.
                Each element of Band is a physical frequency and shall not exceed the
                Nyquist frequency, i.e., half of sampling frequency.
        X
                list
                a 1-D real time series.
        Fs
                integer
                the sampling rate in physical frequency
        Returns
        ------
        Power
                list
                spectral power in each frequency bin.
```

```
        Power_ratio
                list
                spectral power in each frequency bin normalized by total power in ALL
                frequency bins.
        """
        C = fft(X)
        C = abs(C)
        Power =zeros(len(Band)-1);
        for Freq_Index in xrange(0,len(Band)-1):
                Freq = float(Band[Freq_Index])
                        ## Xin Liu
                Next_Freq = float(Band[Freq_Index+1])
                Power[Freq_Index] = sum(C[floor(Freq/Fs*len(X)):floor(Next_Freq/Fs*len(X))])
        Power_Ratio = Power/sum(Power)
        return Power, Power_Ratio
def first_order_diff(X):
        """ Compute the first order difference of a time series.
                For a time series X = [x(1), x(2), ... , x(N)], its    first order
                difference is:
                Y = [x(2) - x(1) , x(3) - x(2), ..., x(N) - x(N-1)]
        """
        D=[]
        for i in xrange(1,len(X)):
                D.append(X[i]-X[i-1])
        return D
def pfd(X, D=None):
        """Compute Petrosian Fractal Dimension of a time series from either two
        cases below:
                1. X, the time series of type list (default)
                2. D, the first order differential sequence of X (if D is provided,
                  recommended to speed up)
        In case 1, D is computed by first_order_diff(X) function of pyeeg
        To speed up, it is recommended to compute D before calling this function
        because D may also be used by other functions whereas computing it here
        again will slow down.
        """
```

```
        if D is None:
                                                                                ##
Xin Liu
                D = first_order_diff(X)
        N_delta= 0; #number of sign changes in derivative of the signal
        for i in xrange(1,len(D)):
                if D[i]*D[i-1]<0:
                        N_delta += 1
        n = len(X)
        return log10(n)/(log10(n)+log10(n/n+0.4*N_delta))
def hfd(X, Kmax):
        """ Compute Hjorth Fractal Dimension of a time series X, kmax
         is an HFD parameter
        """
        L = [];
        x = []
        N = len(X)
        for k in xrange(1,Kmax):
                Lk = []
                for m in xrange(0,k):
                        Lmk = 0
                        for i in xrange(1,int(floor((N-m)/k))):
                        Lmk += abs(X[m+i*k] - X[m+i*k-k])
                        Lmk = Lmk*(N - 1)/floor((N - m) / float(k)) / k
                        Lk.append(Lmk)
                L.append(log(mean(Lk)))
                x.append([log(float(1) / k), 1])
        (p, r1, r2, s)=lstsq(x, L)
        return p[0]
def hjorth(X, D = None):
        """ Compute Hjorth mobility and complexity of a time series from either two
        cases below:
                1. X, the time series of type list (default)
                2. D, a first order differential sequence of X (if D is provided,
                  recommended to speed up)
        In case 1, D is computed by first_order_diff(X) function of pyeeg
```

```
    Notes
    -----
    To speed up, it is recommended to compute D before calling this function
    because D may also be used by other functions whereas computing it here
    again will slow down.
    Parameters
    ----------
    X
        list
        a time series
    D
        list
        first order differential sequence of a time series
    Returns
    ------
    As indicated in return line
    Hjorth mobility and complexity
    """
    if D is None:
        D = first_order_diff(X)
    D.insert(0, X[0]) # pad the first difference
    D = array(D)
    n = len(X)
    M2 = float(sum(D ** 2)) / n
    TP = sum(array(X) ** 2)
    M4 = 0;
    for i in xrange(1, len(D)):
        M4 += (D[i] - D[i - 1]) ** 2
    M4 = M4 / n
    return sqrt(M2 / TP), sqrt(float(M4) * TP / M2 / M2)     # Hjorth Mobility and Complexity
def spectral_entropy(X, Band, Fs, Power_Ratio = None):
    """Compute spectral entropy of a time series from either two cases below:
    1. X, the time series (default)
    2. Power_Ratio, a list of normalized signal power in a set of frequency
    bins defined in Band (if Power_Ratio is provided, recommended to speed up)
    In case 1, Power_Ratio is computed by bin_power() function.
```

```
        Notes
        -----
        To speed up, it is recommended to compute Power_Ratio before calling this
        function because it may also be used by other functions whereas computing
        it here again will slow down.
        Parameters
        ----------
        Band
                list
                boundary frequencies (in Hz) of bins. They can be unequal bins, e.g.
                [0.5,4,7,12,30] which are delta, theta, alpha and beta respectively.
                You can also use range() function of Python to generate equal bins and
                pass the generated list to this function.
                Each element of Band is a physical frequency and shall not exceed the
                Nyquist frequency, i.e., half of sampling frequency.
        X
                list
                a 1-D real time series.
        Fs
                integer
                the sampling rate in physical frequency
        Returns
        -------
        As indicated in return line
        See Also
        --------
        bin_power: pyeeg function that computes spectral power in frequency bins
        """
        if Power_Ratio is None:
                Power, Power_Ratio = bin_power(X, Band, Fs)
        Spectral_Entropy = 0
        for i in xrange(0, len(Power_Ratio) - 1):
                Spectral_Entropy += Power_Ratio[i] * log(Power_Ratio[i])
        Spectral_Entropy /= log(len(Power_Ratio))       # to save time, minus one is omitted
        return -1 * Spectral_Entropy
def svd_entropy(X, Tau, DE, W = None):
```

```
    """Compute SVD Entropy from either two cases below:
    1. a time series X, with lag tau and embedding dimension dE (default)
    2. a list, W, of normalized singular values of a matrix (if W is provided,
    recommend to speed up.)
    If W is None, the function will do as follows to prepare singular spectrum:
            First, computer an embedding matrix from X, Tau and DE using pyeeg
            function embed_seq():
                        M = embed_seq(X, Tau, DE)


            Second, use scipy.linalg function svd to decompose the embedding matrix
            M and obtain a list of singular values:
                                W = svd(M, compute_uv=0)
            At last, normalize W:
                                W /= sum(W)
    Notes
    -------------
    To speed up, it is recommended to compute W before calling this function
    because W may also be used by other functions whereas computing        it here
    again will slow down.
    """
    if W is None:
            Y = embed_seq(X, Tau, DE)
            W = svd(Y, compute_uv = 0)
            W /= sum(W) # normalize singular values
    return -1*sum(W * log(W))
def fisher_info(X, Tau, DE, W = None):
    """ Compute Fisher information of a time series from either two cases below:
    1. X, a time series, with lag Tau and embedding dimension DE (default)
    2. W, a list of normalized singular values, i.e., singular spectrum (if W is
      provided, recommended to speed up.)
    If W is None, the function will do as follows to prepare singular spectrum:
            First, computer an embedding matrix from X, Tau and DE using pyeeg
            function embed_seq():
                    M = embed_seq(X, Tau, DE)
            Second, use scipy.linalg function svd to decompose the embedding matrix
```

M and obtain a list of singular values:

W = svd(M, compute_uv=0)

At last, normalize W:

W /= sum(W)

Parameters

X

list

a time series. X will be used to build embedding matrix and compute singular values if W or M is not provided.

Tau

integer

the lag or delay when building a embedding sequence. Tau will be used to build embedding matrix and compute singular values if W or M is not provided.

DE

integer

the embedding dimension to build an embedding matrix from a given series. DE will be used to build embedding matrix and compute singular values if W or M is not provided.

W

list or array

the set of singular values, i.e., the singular spectrum

Returns

FI

integer

Fisher information

Notes

To speed up, it is recommended to compute W before calling this function because W may also be used by other functions whereas computing it here again will slow down.

See Also

---

embed_seq : embed a time series into a matrix

```
        """
        if W is None:
                M = embed_seq(X, Tau, DE)
                W = svd(M, compute_uv = 0)
                W /= sum(W)
        FI = 0
        for i in xrange(0, len(W) - 1):    # from 1 to M
                FI += ((W[i +1] - W[i]) ** 2) / (W[i])
        return FI
def ap_entropy(X, M, R):
        """Computer approximate entropy (ApEN) of series X, specified by M and R.
        Suppose given time series is X = [x(1), x(2), ... , x(N)]. We first build
        embedding matrix Em, of dimension (N-M+1)-by-M, such that the i-th row of Em
        is x(i),x(i+1), ... , x(i+M-1). Hence, the embedding lag and dimension are
        1 and M-1 respectively. Such a matrix can be built by calling pyeeg function
        as Em = embed_seq(X, 1, M). Then we build matrix Emp, whose only
        difference with Em is that the length of each embedding sequence is M + 1
        Denote the i-th and j-th row of Em as Em[i] and Em[j]. Their k-th elments
        are       Em[i][k] and Em[j][k] respectively. The distance between Em[i] and Em[j]
        is defined as 1) the maximum difference of their corresponding scalar
        components, thus, max(Em[i]-Em[j]), or 2) Euclidean distance. We say two 1-D
        vectors Em[i] and Em[j] *match* in *tolerance* R, if the distance between them
        is no greater than R, thus, max(Em[i]-Em[j]) <= R. Mostly, the value of R is
        defined as 20% - 30% of standard deviation of X.
        Pick Em[i] as a template, for all j such that 0 < j < N - M + 1, we can
        check whether Em[j] matches with Em[i]. Denote the number of Em[j],
        which is in the range of Em[i], as k[i], which is the i-th element of the
        vector k. The probability that a random row in Em matches Em[i] is
        \simga_1^{N-M+1} k[i] / (N - M + 1), thus sum(k)/ (N - M + 1),
        denoted as Cm[i].
        We repeat the same process on Emp and obtained Cmp[i], but here 0<i<N-M
        since the length of each sequence in Emp is M + 1.
        The probability that any two embedding sequences in Em match is then
        sum(Cm)/ (N - M +1 ). We define Phi_m = sum(log(Cm)) / (N - M + 1) and
        Phi_mp = sum(log(Cmp)) / (N - M ).
        And the ApEn is defined as Phi_m - Phi_mp.
```

```
        Notes
        -----
        #. Please be aware that self-match is also counted in ApEn.
        #. This function now runs very slow. We are still trying to speed it up.
        References
        ----------
        Costa M, Goldberger AL, Peng CK, Multiscale entropy analysis of biolgical
        signals, Physical Review E, 71:021906, 2005
        See also
        --------
        samp_entropy: sample entropy of a time series
        Notes
        -----
        Extremely slow implementation. Do NOT use if your dataset is not small.
        """
        N = len(X)
        Em = embed_seq(X, 1, M)
        Emp = embed_seq(X, 1, M + 1) #          try to only build Emp to save time
        Cm, Cmp = zeros(N - M + 1), zeros(N - M)
        # in case there is 0 after counting. Log(0) is undefined.
        for i in xrange(0, N - M):
#               print i
                for j in xrange(i, N - M): # start from i, self-match counts in ApEn
#                       if max(abs(Em[i]-Em[j])) <= R:# compare N-M scalars in each subseq v
0.01b_r1
                        if in_range(Em[i], Em[j], R):
                                Cm[i] += 1

                        ### Xin Liu
                                Cm[j] += 1
                                if abs(Emp[i][-1] - Emp[j][-1]) <= R: # check last one
                                        Cmp[i] += 1
                                        Cmp[j] += 1
                if in_range(Em[i], Em[N-M], R):
                        Cm[i] += 1
                        Cm[N-M] += 1
```

```
                # try to count Cm[j] and Cmp[j] as well here
#               if max(abs(Em[N-M]-Em[N-M])) <= R: # index from 0, so N-M+1 is N-M  v
0.01b_r1
#       if in_range(Em[i], Em[N - M], R):  # for Cm, there is one more iteration than Cmp
#                       Cm[N - M] += 1 # cross-matches on Cm[N - M]
        Cm[N - M] += 1 # Cm[N - M] self-matches
#       import code;code.interact(local=locals())
        Cm /= (N - M +1 )
        Cmp /= ( N - M )
#       import code;code.interact(local=locals())
        Phi_m, Phi_mp = sum(log(Cm)),  sum(log(Cmp))
        Ap_En = (Phi_m - Phi_mp) / (N - M)
        return Ap_En
def samp_entropy(X, M, R):
        """Computer sample entropy (SampEn) of series X, specified by M and R.
        SampEn is very close to ApEn.
        Suppose given time series is X = [x(1), x(2), ... , x(N)]. We first build
        embedding matrix Em, of dimension (N-M+1)-by-M, such that the i-th row of Em
        is x(i),x(i+1), ... , x(i+M-1). Hence, the embedding lag and dimension are
        1 and M-1 respectively. Such a matrix can be built by calling pyeeg function
        as Em = embed_seq(X, 1, M). Then we build matrix Emp, whose only
        difference with Em is that the length of each embedding sequence is M + 1
        Denote the i-th and j-th row of Em as Em[i] and Em[j]. Their k-th elments
        are       Em[i][k] and Em[j][k] respectively. The distance between Em[i] and Em[j]
        is defined as 1) the maximum difference of their corresponding scalar
        components, thus, max(Em[i]-Em[j]), or 2) Euclidean distance. We say two 1-D
        vectors Em[i] and Em[j] *match* in *tolerance* R, if the distance between them
        is no greater than R, thus, max(Em[i]-Em[j]) <= R. Mostly, the value of R is
        defined as 20% - 30% of standard deviation of X.
        Pick Em[i] as a template, for all j such that 0 < j < N - M , we can
        check whether Em[j] matches with Em[i]. Denote the number of Em[j],
        which is in the range of Em[i], as k[i], which is the i-th element of the
        vector k.
        We repeat the same process on Emp and obtained Cmp[i], 0 < i < N - M.
        The SampEn is defined as log(sum(Cm)/sum(Cmp))
        References
```

```
        ----------
        Costa M, Goldberger AL, Peng C-K, Multiscale entropy analysis of biolgical
        signals, Physical Review E, 71:021906, 2005
        See also
        --------
        ap_entropy: approximate entropy of a time series
        Notes
        -----
        Extremely slow computation. Do NOT use if your dataset is not small and you
        are not patient enough.
        """
        N = len(X)
        Em = embed_seq(X, 1, M)
        Emp = embed_seq(X, 1, M + 1)
        Cm, Cmp = zeros(N - M - 1) + 1e-100, zeros(N - M - 1) + 1e-100
        # in case there is 0 after counting. Log(0) is undefined.
        for i in xrange(0, N - M):
                for j in xrange(i + 1, N - M): # no self-match
#                       if max(abs(Em[i]-Em[j])) <= R:  # v 0.01_b_r1
                        if in_range(Em[i], Em[j], R):
                                Cm[i] += 1
#                       if max(abs(Emp[i] - Emp[j])) <= R: # v 0.01_b_r1
                                if abs(Emp[i][-1] - Emp[j][-1]) <= R: # check last one
                                        Cmp[i] += 1
        Samp_En = log(sum(Cm)/sum(Cmp))
        return Samp_En
def dfa(X, Ave = None, L = None):
        """Compute Detrended Fluctuation Analysis from a time series X and length of
        boxes L.
        The first step to compute DFA is to integrate the signal. Let original seres
        be X= [x(1), x(2), ..., x(N)].
        The integrated signal Y = [y(1), y(2), ..., y(N)] is otained as follows
        y(k) = \sum_{i=1}^{k}{x(i)-Ave} where Ave is the mean of X.
        The second step is to partition/slice/segment the integrated sequence Y into
        boxes. At least two boxes are needed for computing DFA. Box sizes are
        specified by the L argument of this function. By default, it is from 1/5 of
```

signal length to one (x-5)-th of the signal length, where x is the nearest power of 2 from the length of the signal, i.e., 1/16, 1/32, 1/64, 1/128, ...
In each box, a linear least square fitting is employed on data in the box. Denote the series on fitted line as Yn. Its k-th elements, yn(k), corresponds to y(k).
For fitting in each box, there is a residue, the sum of squares of all offsets, difference between actual points and points on fitted line.
F(n) denotes the square root of average total residue in all boxes when box length is n, thus
Total_Residue = \sum_{k=1}^{N}{(y(k)-yn(k))}
F(n) = \sqrt(Total_Residue/N)
The computing to F(n) is carried out for every box length n. Therefore, a relationship between n and F(n) can be obtained. In general, F(n) increases when n increases.
Finally, the relationship between F(n) and n is analyzed. A least square fitting is performed between log(F(n)) and log(n). The slope of the fitting line is the DFA value, denoted as Alpha. To white noise, Alpha should be 0.5. Higher level of signal complexity is related to higher Alpha.
Parameters

X:
    1-D Python list or numpy array
    a time series
Ave:
    integer, optional
    The average value of the time series
L:
    1-D Python list of integers
    A list of box size, integers in ascending order
Returns

Alpha:
    integer
    the result of DFA analysis, thus the slope of fitting line of log(F(n))
    vs. log(n). where n is the
Examples

```
--------
>>> import pyeeg
>>> from numpy.random import randn
>>> print pyeeg.dfa(randn(4096))
0.490035110345
Reference
---------
Peng C-K, Havlin S, Stanley HE, Goldberger AL. Quantification of scaling
exponents and   crossover phenomena in nonstationary heartbeat time series.
_Chaos_ 1995;5:82-87
Notes
-----
This value depends on the box sizes very much. When the input is a white
noise, this value should be 0.5. But, some choices on box sizes can lead to
the value lower or higher than 0.5, e.g. 0.38 or 0.58.
Based on many test, I set the box sizes from 1/5 of         signal length to one
(x-5)-th of the signal length, where x is the nearest power of 2 from the
length of the signal, i.e., 1/16, 1/32, 1/64, 1/128, ...
You may generate a list of box sizes and pass in such a list as a parameter.
"""
X = array(X)
if Ave is None:
        Ave = mean(X)
Y = cumsum(X)
Y -= Ave
if L is None:
        L = floor(len(X)*1/(2**array(range(4,int(log2(len(X)))-4))))
F = zeros(len(L)) # F(n) of different given box length n
for i in xrange(0,len(L)):
        n = int(L[i])                                   # for each box length L[i]
        if n==0:
                print "time series is too short while the box length is too big"
                print "abort"
                exit()
        for j in xrange(0,len(X),n): # for each box
                if j+n < len(X):
```

```
                    c = range(j,j+n)
                    c = vstack([c, ones(n)]).T # coordinates of time in the box
                    y = Y[j:j+n]                    # the value of data in the box
                    F[i] += lstsq(c,y)[1]       # add residue in this box
        F[i] /= ((len(X)/n)*n)
F = sqrt(F)
Alpha = lstsq(vstack([log(L), ones(len(L))]).T,log(F))[0][0]
return Alpha
```

EK 2

Sevcik ve Katz Fraktal Boyutlarini Hesaplayan Python Programları

```
def katz(wave):
        n = len(wave);
        x=arange(n)
        y = wave;
        d = sqrt((x-x[0])**2+(y-y[0])**2);
        d = max(d);
        x = ones(n-1);
        y = wave[1:n]-wave[0:(n-1)];
        L = sum(sqrt(x**2+y**2));
        D = log10(n)/(log10(d/L)+log10(n));
        return D
def sevcik(wave):
    n = len(wave);
    x = arange(n);
    y = wave;
    span = (max(y)-min(y));
    if span < 1e-6:
            D = 1;
        return;
    y = (y-max(y))/span;
    x = (1/(n-1))*ones(n-1);
    y = y[1:n]-y[0:(n-1)];
    L = sum(sqrt(x**2+y**2));
    D = 1+log(L)/log(2*(n-1));
        return D
```

EK 3

Öznitelik ve Sınıflandırma için Python Programları

```
from pylab import *
from scipy import *
from scipy.io import *
from glob import *
from pywt import *
from os import *
from pyeeg import *
import mlpy
from sklearn import *

def eeg_oznitelik():
        vektor_hurst=zeros([3,22,686]);vektorthurst=zeros([22,686]);vektorw1hurst=zeros([22,686]);
vektorw2hurst=zeros([22,686]);
        vektor_pfd=zeros([3,22,686]);vektortpfd=zeros([22,686]);vektorw1pfd=zeros([22,686]);vekto
rw2pfd=zeros([22,686]);
        vektor_hfd=zeros([3,22,686]);vektorthfd=zeros([22,686]);vektorw1hfd=zeros([22,686]);vekto
rw2hfd=zeros([22,686]);
        vektor_spectral_entropy=zeros([3,22,686]);vektortspectral_entropy=zeros([22,686]);vektorw1
spectral_entropy=zeros([22,686]);vektorw2spectral_entropy=zeros([22,686]);
        vektor_svd_entropy=zeros([3,22,686]);vektortsvd_entropy=zeros([22,686]);vektorw1svd_entr
opy=zeros([22,686]);vektorw2svd_entropy=zeros([22,686]);
        vektor_hjorth_m=zeros([3,22,686]);vektorthjorth_m=zeros([22,686]);vektorw1hjorth_m=zero
s([22,686]);vektorw2hjorth_m=zeros([22,686]);
        vektor_hjorth_c=zeros([3,22,686]);vektorthjorth_c=zeros([22,686]);vektorw1hjorth_c=zeros([
22,686]);vektorw2hjorth_c=zeros([22,686]);
        vektor_fisher_info=zeros([3,22,686]);vektortfisher_info=zeros([22,686]);vektorw1fisher_info
=zeros([22,686]);vektorw2fisher_info=zeros([22,686]);
        vektor_ap_entropy=zeros([3,22,686]);vektortap_entropy=zeros([22,686]);vektorw1ap_entropy
=zeros([22,686]);vektorw2ap_entropy=zeros([22,686]);
```

```
vektor_samp_entropy=zeros([3,22,686]);vektortsamp_entropy=zeros([22,686]);vektorw1samp
_entropy=zeros([22,686]);vektorw2samp_entropy=zeros([22,686]);
vektor_sevcik=zeros([3,22,686]);vektortsevcik=zeros([22,686]);vektorw1sevcik=zeros([22,68
6]);vektorw2sevcik=zeros([22,686]);
vektor_katz=zeros([3,22,686]);vektortkatz=zeros([22,686]);vektorw1katz=zeros([22,686]);vek
torw2katz=zeros([22,686]);
chdir("/eeg/");
dosya=glob("chb*.mat");

for i in range(686):
    mat1=loadmat(dosyas[i]);mat3=mat1['val'];
    mat4=mat3[:,:];s1=mat4.shape;s2=s1[0];s3=s1[1]
    mat5=zeros([s2,s3])
    mat2=zeros([s2,s3])
    for ii in range(22):
        mat5[ii][:]=mat4[ii][:]-mean(mat4[ii][:])
        mat2[ii][:]=mat5[ii][:]/max(abs(mat5[ii][:]))

    for j in range(22):
        w1,w2=dwt(mat2[j,:],'haar')
        eeg_hurst=hurst(mat2[j,:]);eeg_hurstw1=hurst(w1);eeg_hurstw2=hurst(w2);

vektorthurst[j][i]=eeg_hurst;vektorw1hurst[j][i]=eeg_hurstw1;vektorw2hurst[j][i]=eeg_hurstw
2;
        eeg_pfd=pfd(mat2[j,:]);eeg_pfdw1=pfd(w1);eeg_pfdw2=pfd(w2);

vektortpfd[j][i]=eeg_pfd;vektorw1pfd[j][i]=eeg_pfdw1;vektorw2pfd[j][i]=eeg_pfdw2;
        eeg_hfd=hfd(mat2[j,:],5);eeg_hfdw1=hfd(w1,5);eeg_hfdw2=hfd(w2,5);

vektorthfd[j][i]=eeg_hfd;vektorw1hfd[j][i]=eeg_hfdw1;vektorw2hfd[j][i]=eeg_hfdw2;

eeg_hjorth_m,eeg_hjorth_c=hjorth(mat2[j,:]);eeg_hjorthw1_m,eeg_hjorthw1_c=hjorth(w1);ee
g_hjorthw2_m,eeg_hjorthw2_c=hjorth(w2);

vektorthjorth_m[j][i]=eeg_hjorth_m;vektorw1hjorth_m[j][i]=eeg_hjorthw1_m;vektorw2hjorth
_m[j][i]=eeg_hjorthw2_m;
```

vektorthjorth_c[j][i]=eeg_hjorth_c;vektorw1hjorth_c[j][i]=eeg_hjorthw1_c;vektorw2hjorth_c[j][i]=eeg_hjorthw2_c;

eeg_spectral_entropy=spectral_entropy(mat2[j,:],[0.5,4,7,12,30],256);eeg_spectral_entropyw1=spectral_entropy(w1,[0.5,4,7,12,30],256);eeg_spectral_entropyw2=spectral_entropy(w2,[0.5,4,7,12,30],256);

vektortspectral_entropy[j][i]=eeg_spectral_entropy;vektorw1spectral_entropy[j][i]=eeg_spectral_entropyw1;vektorw2spectral_entropy[j][i]=eeg_spectral_entropyw2;

eeg_svd_entropy=svd_entropy(mat2[j,:],1,4);eeg_svd_entropyw1=svd_entropy(w1,1,4);eeg_svd_entropyw2=svd_entropy(w2,1,4);

vektortsvd_entropy[j][i]=eeg_svd_entropy;vektorw1svd_entropy[j][i]=eeg_svd_entropyw1;vektorw2svd_entropy[j][i]=eeg_svd_entropyw2;

eeg_fisher_info=fisher_info(mat2[j,:],1,4);eeg_fisher_infow1=fisher_info(w1,1,4);eeg_fisher_infow2=fisher_info(w2,1,4);

vektortfisher_info[j][i]=eeg_fisher_info;vektorw1fisher_info[j][i]=eeg_fisher_infow1;vektorw2fisher_info[j][i]=eeg_fisher_infow2;

eeg_ap_entropy=ap_entropy(mat2[j,:],2,0.15);eeg_ap_entropyw1=ap_entropy(w1,2,0.15);eeg_ap_entropyw2=ap_entropy(w2,2,0.15);

vektortap_entropy[j][i]=eeg_ap_entropy;vektorw1ap_entropy[j][i]=eeg_ap_entropyw1;vektorw2ap_entropy[j][i]=eeg_ap_entropyw2;

eeg_samp_entropy=samp_entropy(mat2[j,:],2,0.15);eeg_samp_entropyw1=samp_entropy(w1,2,0.15);eeg_samp_entropyw2=samp_entropy(w2,2,0.15);

vektortsamp_entropy[j][i]=eeg_samp_entropy;vektorw1samp_entropy[j][i]=eeg_samp_entropyw1;vektorw2samp_entropy[j][i]=eeg_samp_entropyw2;

eeg_sevcik=sevcik(mat2[j,:]);eeg_sevcikw1=sevcik(w1);eeg_sevcikw2=sevcik(w2);

```
        vektortsevcik[j][i]=eeg_sevcik;vektorw1sevcik[j][i]=eeg_sevcikw1;vektorw2sevcik[j][i]=eeg_
sevcikw2;
                        eeg_katz=katz(mat2[j,:]);eeg_katzw1=katz(w1);eeg_katzw2=katz(w2);

        vektortkatz[j][i]=eeg_katz;vektorw1katz[j][i]=eeg_katzw1;vektorw2katz[j][i]=eeg_katzw2;
        vektor_hurst[0][:][:]=vektorthurst;vektor_hurst[1][:][:]=vektorw1hurst;vektor_hurst[2][:][:]=v
ektorw2hurst;
        vektor_pfd[0][:][:]=vektortpfd;vektor_pfd[1][:][:]=vektorw1pfd;vektor_pfd[2][:][:]=vektorw2
pfd;
        vektor_hfd[0][:][:]=vektorthfd;vektor_hfd[1][:][:]=vektorw1hfd;vektor_hfd[2][:][:]=vektorw2
hfd;
        vektor_hjorth_m[0][:][:]=vektorthjorth_m;vektor_hjorth_m[1][:][:]=vektorw1hjorth_m;vektor
_hjorth_m[2][:][:]=vektorw2hjorth_m;
        vektor_hjorth_c[0][:][:]=vektorthjorth_c;vektor_hjorth_c[1][:][:]=vektorw1hjorth_c;vektor_hj
orth_c[2][:][:]=vektorw2hjorth_c;
        vektor_spectral_entropy[0][:][:]=vektortspectral_entropy;vektor_spectral_entropy[1][:][:]=vek
torw1spectral_entropy;vektor_spectral_entropy[2][:][:]=vektorw2spectral_entropy;
        vektor_svd_entropy[0][:][:]=vektortsvd_entropy;vektor_svd_entropy[1][:][:]=vektorw1svd_en
tropy;vektor_svd_entropy[2][:][:]=vektorw2svd_entropy;
        vektor_fisher_info[0][:][:]=vektortfisher_info;vektor_fisher_info[1][:][:]=vektorw1fisher_info
;vektor_fisher_info[2][:][:]=vektorw2fisher_info;
        vektor_ap_entropy[0][:][:]=vektortap_entropy;vektor_ap_entropy[1][:][:]=vektorw1ap_entrop
y;vektor_ap_entropy[2][:][:]=vektorw2ap_entropy;
        vektor_samp_entropy[0][:][:]=vektortsamp_entropy;vektor_samp_entropy[1][:][:]=vektorw1s
amp_entropy;vektor_samp_entropy[2][:][:]=vektorw2samp_entropy;
        vektor_sevcik[0][:][:]=vektortsevcik;vektor_sevcik[1][:][:]=vektorw1sevcik;vektor_sevcik[2][
:][:]=vektorw2sevcik;
        vektor_katz[0][:][:]=vektortkatz;vektor_katz[1][:][:]=vektorw1katz;vektor_katz[2][:][:]=vekto
rw2katz;

def eeg_sinifla():
        mat1=loadmat(“dosya.mat”);
        mat2=loadmat("dosya_t.mat")
        xx=mat1['vektor_hurst'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_hurst=ee
g_siniflama_uc(xx,yy)
```

```
xx=mat1['vektor_pfd'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_pfd=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_hfd'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_hfd=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_hjorth_m'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_hjorth_m=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_hjorth_c'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_hjorth_c=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_spectral_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_spectral_entropy=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_svd_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_svd_entropy=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_fisher_info'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_fisher_info=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_ap_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_ap_entropy=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_samp_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_samp_entropy=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_sevcik'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_sevcik=eeg_siniflama_uc(xx,yy)
xx=mat1['vektor_katz'];yy=transpose(array(mat2['targetvector'])[0]);s_mlpy_vektor_katz=eeg_siniflama_uc(xx,yy)

xx=mat1['vektor_hurst'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_hurst=eeg_siniflama_sl(xx,yy)
xx=mat1['vektor_pfd'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_pfd=eeg_siniflama_sl(xx,yy)
xx=mat1['vektor_hfd'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_hfd=eeg_siniflama_sl(xx,yy)
xx=mat1['vektor_hjorth_m'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_hjorth_m=eeg_siniflama_sl(xx,yy)
xx=mat1['vektor_hjorth_c'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_hjorth_c=eeg_siniflama_sl(xx,yy)
xx=mat1['vektor_spectral_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_spectral_entropy=eeg_siniflama_sl(xx,yy)
```

```
        xx=mat1['vektor_svd_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_svd_
entropy=eeg_siniflama_sl(xx,yy)
        xx=mat1['vektor_fisher_info'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_fisher
_info=eeg_siniflama_sl(xx,yy)
        xx=mat1['vektor_ap_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_ap_en
tropy=eeg_siniflama_sl(xx,yy)
        xx=mat1['vektor_samp_entropy'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_sa
mp_entropy=eeg_siniflama_sl(xx,yy)
        xx=mat1['vektor_sevcik'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_sevcik=ee
g_siniflama_sl(xx,yy)
        xx=mat1['vektor_katz'];yy=transpose(array(mat2['targetvector'])[0]);s_sl_vektor_katz=eeg_sin
iflama_sl(xx,yy)

def eeg_siniflama_uc(xx_data,yy_data):
        x0=transpose(xx_data[0][:][:]);x1=transpose(xx_data[1][:][:]);x2=transpose(xx_data[2][:][:]);
        eeg_t=eeg_siniflama_yontem(x0,yy_data);eeg_w1=eeg_siniflama_yontem(x1,yy_data);eeg_w
2=eeg_siniflama_yontem(x2,yy_data)
        sinif=[eeg_t,eeg_w1,eeg_w2]
        return sinif

def eeg_siniflama_sl(xx_data,yy_data):
        x0=transpose(xx_data[0][:][:]);x1=transpose(xx_data[1][:][:]);x2=transpose(xx_data[2][:][:]);
        eeg_t=eeg_siniflama_sl_yontem(x0,yy_data);eeg_w1=eeg_siniflama_sl_yontem(x1,yy_data);
eeg_w2=eeg_siniflama_sl_yontem(x2,yy_data)
        sinif=[eeg_t,eeg_w1,eeg_w2]
        return sinif

def eeg_ysa_siniflama(xx_data,yy_data):
        x0=transpose(xx_data[0][:][:]);x1=transpose(xx_data[1][:][:]);x2=transpose(xx_data[2][:][:]);
        eeg_t=eeg_ysa_siniflama_yontem(x0,yy_data);eeg_w1=eeg_ysa_siniflama_yontem(x1,yy_dat
a);eeg_w2=eeg_ysa_siniflama_yontem(x2,yy_data)
        sinif=[eeg_t,eeg_w1,eeg_w2]
        return sinif
```

```
def eeg_ysa_siniflama_yontem(xx_input,yy_target):
        ysas=ysa_sinifla(xx_input,yy_target);
        ysas1=100-100*sum(abs(yy_target-ysa))/686;
        return ysas1

def ysa_sinifla(xx_input,yy_target):
        input = xx_input;
        target = yy_target;
        net=nl.net.newff([input.min(),input.max()],[5, 1])
        err=nl.net.train(input, target, show=15)
        return nl.net.sim(input)
```

ÖZGEÇMİŞ

1977 yılında İstanbul'da doğdu. İlk, orta ve lise öğrenimini Yalova'da tamamladı. 2002 yılında İstanbul Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümünden mezun oldu. 2007 yılında Namık Kemal Üniversitesi Bilgi İşlem Daire Başkanlığında göreve başladı. 2009 yılı güz döneminde Namık Kemal Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalında yüksek lisans öğrenimine başladı. Halen aynı anabilim dalında yüksek lisans öğrencisi olarak devam etmekte ve Namık Kemal Üniversitesi Bilgi İşlem Daire Başkanlığındaki görevini sürdürmektedir.