İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BULANIK (FUZZY) SINIFLAYICILARLA EKG ŞEKİL BOZUKLUKLARININ BELİRLENMESİ

Y. LİSANS TEZİ

Müh. Zümray DOKUR

Tezin Enstitüye Verildiği Tarih: 16 Ocak 1995

Tezin Savunulduğu Tarih : 2 Şubat 1995

Tez Danışmanı : Doç. Dr. Mehmet KORÜREK

Diğer Jüri Üyeleri : Prof. Dr. Uğur ÇİLİNGİROĞLU

Prof. Dr. Cem GÖKNAR

ŞUBAT 1995

ÖNSÖZ

Yapılan yüksek lisans tez çalışmasında, bulanık küme teorisinden hareketle geliştirilen bulanık sınıflayıcılar kullanılarak, insan için hayati önem taşıyan kalp hastalıklarının teşhisinde faydalanılan bir kısım EKG şekil bozuklukları belirlenmiştir.

Bu tez çalışmasında işlenen bulanık mantık kavramıyla tanışmamı sağlayan ve tezin hazırlanmasında yardım ve değerli eleştirilerini esirgemeyen sayın hocam Doç. Dr. Mehmet KORÜREK'e teşekkürü borç bilirim. Araştırmamın tüm adımlarında verdiği yoğun moral desteği, yardım ve eleştirilerinden ötürü arkadaşım Araş. Gör. Tamer ÖLMEZ'e ayrıca teşekkür ederim. Yardımlarını esirgemeyen tüm arkadaşlarıma ve Ayla ÇEVİK'e, sabırla, anlayışla, sevgiyle beni hep desteklemiş olan aileme ve amcam Yaşar ELÇİ'ye tek tek teşekkür ediyorum.

Zümray DOKUR

İstanbul, ŞUBAT 1995

İÇİNDEKİLER

ÖZET V

SUMMARY VI

BÖLÜM 1. GİRİŞ 1

1.1 Çalışmaya Genel Bakış 1

BÖLÜM 2. ELEKTROKARDİYOGRAFİ 5

2.1 Giriş 4
2.2 Biyolojik İşaretler 4
2.3 Aksiyon Potansiyeli 5
2.4 Elektrokardiyogram 10
2.5 Derivasyon Bağlantıları 14
2.5.1 Ekstremite Derivasyonları 14
2.5.2 Göğüs Derivasyonları 17
2.6 QRS Şekil Bozuklukları 19
2.6.1 Normal Vuru 20
2.6.2 Sol Dal Bloğu 21
2.6.3 Erken Karıncık Kasılması 22
2.6.4 Yapay Vuru 23
2.7 MIT-BIH Veri Tabanı 24

BÖLÜM 3. EKG İŞARETİ İÇERİSİNDEN R TEPESİNİN BULUNMASI..28

3.1 Giriş 28
3.2 EKG İşareti Üzerinde Elektriksel Gürültü 29
3.3 QRS İşaretinin Tespit Edilmesi 30
3.4 Sayısal Filtreler 32
3.4.1 Yüksek ve Band Geçiren Filtreler 32
3.4.2 Alçak Geçiren Filtre 38

BÖLÜM 4. EKG İŞARETİNİ SINIFLAMADA KULLANILAN ÖZELLİK VEKTÖRLERİ 40

4.1 Giriş 40
4.2 QRS Kompleksinden Çıkarılan Bilgilerle Özellik Vektörünün Oluşturulması 41
4.3 EKG İşaretinin Frekans Spektrumu Kullanılarak Oluşturulan Özellik Vektörü 43
4.4 QRS İşaretinin Lineer Tahmin Katsayıları (LPCs) Kullanılarak Özellik Vektörünün Oluşturulması.47
4.5 EKG İşareti Direkt Alınarak Özellik Vektörünün Oluşturulması 49

BÖLÜM 5. BULANIK (FUZZY) SINIFLAYICILAR 50

5.1 Giriş 50
5.2 Bulanık Küme Teorisi (Fuzzy Set Theory) 50
5.3 Öz-vektörlerin Öğrenilmesi 53
5.3.1 Bulanık Küme-Ortalar Algoritması 54
5.3.2 Bulanık Kohonen Algoritması 57
5.4 Sınıflama İşlemi 60
5.4.1 Bulanık K-En Yakın Komşu Sınıflayıcısı 60
5.4.2 Bulanık En Yakın Model Sınıflayıcısı 64

BÖLÜM 6. PRATİĞE UYGULAMA: EKG İŞARETİNDEKİ ŞEKİL BOZUKLUKLARININ TESPİTİ 67

6.1 Giriş 67
6.2 EKG İşaretinden Özellik Vektörlerinin Elde Edilmesi 67
6.3 Öz Vektörlerin Bulunması 74
6.4 EKG İşaretlerinin Sınıflandırılması 78

BÖLÜM 7. SONUÇ 84

KAYNAKLAR 87

EKLER 89

ÖZGEÇMİŞ 122

ÖZET

Kalple ilgili oluşabilecek rahatsızlıkları önceden tespit etmek insan sağlığı açısından son derece önemlidir. Oluşabilecek kalp rahatsızlıkları ile ilgili bilgiler EKG kayıtlarından iki tip inceleme ile elde edilir: EKG şekil bozukluğu, ritim bozukluğu. Ancak bu bilgilerin elde edilmesi için uzun EKG (bir günü aşan) kayıtlarına ihtiyaç vardır. EKG kayıtlarında bir program kontrolü olmadan şekil bozukluklarını aramak uzun zaman almaktadır. Tez içerisinde EKG işaretindeki şekil bozukluklarının hızlı ve otomatik olarak tespit edilmesi sağlanmıştır.

Çalışma içerisinde, öncelikle EKG işaretinden QRS kompleksi tespit edilir, daha sonra bu kompleks içinde R tepesinin yeri bulunur. Çalışma içerisinde literatürde kullanılan 4 farklı tip özellik vektörü elde etme yöntemi incelenmiştir: 1) QRS şekil bilgisi kullanılarak özellik vektörünün elde edilmesi, 2) 0-30 Hz arasında frekans bandı kullanılarak özellik vektörünün elde edilmesi, 3) Lineer tahmin katsayıları (Lineer Prediction Coefficients, LPCs) kullanılarak özellik vektörünün elde edilmesi, 4) R tepesi etrafındaki EKG genlikleri kullanılarak özellik vektörlerinin elde edilmesi.

Birinci ve dördüncü tip özellik elde etme yöntemlerinin R tepesinin yerine çok bağlı olduğu gözlenmiştir. Ancak bu iki yöntem ile özellik vektörleri daha hızlı elde edilebilmektedir. Diğer özellik elde etme yöntemleri R tepesinin yerinin yanlış belirlenmesinden etkilenmez; ancak hesaplama yükü diğer yöntemlere göre daha uzundur.

Çalışma içerisinde, EKG kayıtlarını izleyen kişiye daha fazla bilgi verdiği için bulanık sınıflayıcıların kullanılması tercih edilmiştir. İki farklı bulanık sınıflayıcı incelenmiştir: Bulanık K-en yakın komşu sınıflayıcısı ve bulanık en yakın model sınıflayıcısı.

Bulanık sınıflayıcıların öz-vektörlerinin elde edilmesi için iki farklı öğrenme algoritması kullanılmıştır: Bulanık küme-ortalar (fuzzy c-means) algoritması ve bulanık Kohonen kümeleme algoritması.

Öğrenme sırasında eğitim kümesi, üç değişik hastadan alınan EKG kayıtları içerisinden, her sınıftan 15 vektör alınmak suretiyle, dört farklı sınıf için 4*15 vektörden meydana getirilmiştir.

SUMMARY

DETECTION OF ECG SHAPE CHANGES BY USING FUZZY CLASSIFIERS

As the ECG has considerable diagnostic significance, it is important to detect and display shape changes on the ECG recordings fast and automatically. In this study, shape detection was performed by fuzzy classifiers.

A fuzzy classifier searches for the membership of each ECG recording to all of the classes and presents the information held by the membership degrees to the decision of the cardiologist to make diagnosis. The hard classifier searches for the unique class that the ECG recording belongs to.

Two types of classifiers are used in this study: Fuzzy nearest prototype (fuzzy NP) classifier and fuzzy K nearest neighbour classifier (fuzzy K-NN).

All the evaluations in this study are performed on the MIT-BIH Arrhythmia Database. The database consists of 48 half-hour recordings for a total of 24h of ECG data. It provides a standardized means of comparing the basic performances of the algorithms used in this thesis and in other studies.

Before the classification, at the first step of the study, R peaks of the QRS complexes are detected.

After the detection of R peaks, feature vectors are formed by 4 different methods: Linear Prediction Coefficients (LPCs) method, QRS shape information, amplitudes of the significant frequencies on the frequency spectrum, and directly using the amplitudes of the ECG signal.

Feature vectors obtained by the preceding methods constituting the training sets of 4 different classes are used in the learning process. In this process, fuzzy c-means and fuzzy Kohonen clustering algorithms are used to obtain the cluster prototypes. Learning is completed after the formation of the prototypes.

After these processes, fuzzy classification of the feature vectors belonging to the test set is performed. Successful results are obtained for the four different classes by the two fuzzy classifiers.

QRS detection is the first stage of signal processing in which pattern recognition is important. Detection of the QRS is difficult, not only because of the physiological variability of the QRS complexes, but also because of the various types of noise that can be present in the ECG signal. Noise sources include muscle noise, artifacts due to electrode motion, power-line interference, baseline wander, and T waves with high frequency characteristics similar to QRS complexes. In our approach, digital filters reduce the influence of these noise sources, and thereby improve the signal-to-noise ratio.

Software QRS detectors typically include one or more three types of processing steps: Linear digital filtering, nonlinear transformation, and decision rule algorithms. We use all three types, in order to enhance the QRS complexes while suppressing noise artifact, baseline wander and non-QRS portions of the ECG. Linear processes include a band-pass filter (BPF), a high-pass filter (HPF), a low-pass filter (LPF), and a moving window integrator. The nonlinear information that we use is signal amplitude squaring. The linear and nonlinear units all form the preprocessing stage of QRS detection. The output of the preprocessor stage is presented to a decision rule that declares a QRS detection if the preprocessor output exceeds a threshold. The most common rule is a simple amplitude threshold, and in this study, the threshold level is started at some nominal level (based on past history), and then is lowered systematically. The QRS detector is illustrated in fig 1.

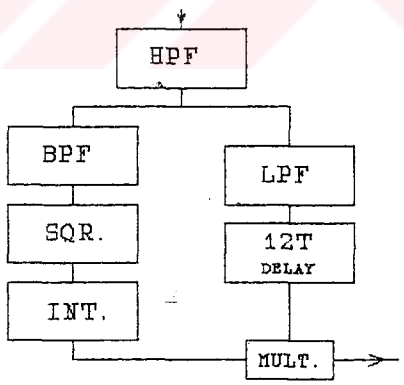


Fig 1: The QRS detector.

In this study, normalized analog filters are used to design the digital filters. A normalized analog filter is a low-pass filter with a cutoff frequency of 1 Hz. By using this filter, other required filters can be designed. Bilinear transformation is used for the transformation from analog filters to digital ones. The bilinear transfer

relation is as follows (T is the sampling period):

$$s \rightarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{+1}}$$

First, in order to restore the baseline, the signal passes through a digital high-pass filter with a cutoff frequency of 2 Hz.

The transfer function for this high-pass filter is:

$$T_H(z) = \frac{Y(z)}{U(z)} = \frac{0.976 - 1.952z^{-1} + 0.976z^{-2}}{1 - 1.951z^{-1} + 0.952z^{-2}}$$

The difference equation is:

$$y(k) = 1.951y(k-1) - 0.952y(k-2) + 0.976u(k) - 1.952u(k-1) + 0.976u(k-2)$$

The output of the high-pass filter is processed in two different branches, and the outputs of these branches are fed into a multiplier. On one of these branches, the output of the high-pass filter is given to a band-pass filter with a center frequency of 17 Hz which is the main frequency of the QRS complex. BPF reduces the influence of muscle noise, 50 Hz interference, and T-wave interference.

The transfer function of the band-pass filter is:

$$T_B(z) = \frac{Y(z)}{U(z)} = \frac{0.0337 - 0.0337z^{-2}}{1 - 1.849z^{-1} + 0.933z^{-2}}$$

The difference equation is:

$$y(k) = 1.849y(k-1) - 0.933y(k-2) + 0.0337u(k) - 0.0337u(k-2)$$

The output of the BPF is processed in the squaring unit. The signal is squared point by point. The equation of this operation is:

$$y(k) = x(k)^2$$

This makes all data points positive and does nonlinear amplification of the output of the band-pass filter emphasizing the higher frequencies (i.e., predominantly the ECG frequencies).

The next process after squaring is moving window integration. The purpose of moving window integration is to obtain wave-form feature information in addition to the slope of the R wave. It is calculated from

$$y(k) = (1/N).[x(k-(N-1)) + x(k-(N-2)) + \ldots\ldots + x(k)]$$

where N is the number of samples in the width of the integration window. The width of the window is determined empirically. For our sample rate of 360 samples/s, the window is 30 sample wide.

On the other branch; the output of the high-pass filter is given to a digital low-pass filter with a cutoff frequency of 36 Hz to reduce the high-frequency noise present in the ECG signal.

This filter is designed directly in the z domain. Its transfer function is:

$$T_L(z) = \frac{(1 - z^{-k})^1}{(1 - z^{-1})^1}$$

k = (sampling frequency) / (cutoff frequency)

In this study k is 10 according to k=360/36=10. 1 is chosen as 2.

$$T_L(z) = \frac{Y(z)}{U(z)} = \frac{1 - 2z^{-10} + z^{-20}}{1 - 2z^{-1} + z^{-2}}$$

The difference equation is:

$$y(k) = 2y(k-1) - y(k-2) + u(k) - 2u(k-10) + u(k-20)$$

The output of the low-pass filter is delayed by the total processing time of the detection algorithm. It is chosen empirically as 12T.

$$y(k) = x(k-12)$$

The outputs of the integration and delay units are multiplied and then presented to the decision rule (amplitude threshold) which declares a QRS detection if the output of the multiplier exceeds a threshold.

After detecting the QRS complex, the maximum amplitude in this complex is taken as the R peak. This process, detection of R peak in the ECG signal is very important.

After the detection of the QRS complex and the R peak, feature extraction process is started.

A common question that arises in connection with clustering and classification concerns the constitution of data space. Features (or characteristics) of data item $x_k \epsilon X$ must be sufficiently representative of the physical

process to enable us to construct realistic clusters or classifiers. By discarding, modifying, enriching and transforming some features of x_k, we need to construct the "right" data space.

In feature selection, we search for internal structure in data items: the goal is to improve their usefulness as constituents of data sets for clustering and/or classification. In short, we hope to identify the "best" data space by looking for structure within its elements.

In this thesis, feature selection is made by using four different methods:

1) Feature vectors are formed by using the shape information of the QRS complex.

The features are:

a) Duration, the width of the QRS complex in milliseconds.

b) Height, the difference between the maximum and the minimum points of the QRS complex.

c) The area above the baseline within the QS duration.

d) | (Baseline) - (Height/2) |

All these parameters highly depend on the position of the R peak in the QRS complex.

2) Amplitudes of the significant frequencies on the frequency spectrum are used to form the feature vectors.

After the detection of the R peak in the QRS complex, a window of 256 samples is formed in the vicinity of the R peak. Frequency analysis is made by using the 0-30 Hz frequency band. Amplitude values are used on the frequency spectrum. The following equation is used for the frequency analysis:

$$x(f) = \sum_{k=0}^{255} x(k)\, e^{\frac{-j2\pi \cdot f \cdot k}{N}}$$

The dimension of the feature vector is 30. Feature vectors obtained by this method are not affected by high frequency noises, and the frequency amplitude values are not dependant on the position of the R peak. Misdetection of the R peak is taken as a delay and does not affect the parameters of the feature vectors.

3) Linear Prediction Coefficients (LPCs) method.

Linear prediction coefficients (LPCs) computed for an average cardiac complex are used for feature vector construction.

The choice of LPCs method was not accidental. LPCs are almost insensitive to high frequency noise. Also, each LPC depends from the whole signal, i.e., it carries some sort of information about the signal, not just a local one as conventional features do. Both facts justify an expectation of much higher robustness when classifying with LPCs.

The position of the R peak does not affect the linear prediction coefficients, and the number of parameters representing the signal is small. In this study 6 coefficients are used as the parameters of the feature vectors .

4) The amplitudes of the ECG signal are used directly for feature vector formation. A window containing only one QRS complex is formed by taking 24 points from the left side of the R peak, and 24 points from the right side of it. In this way a feature vector of 48 parameters is obtained. The parameters of the vector highly depend on the position of the R peak. As no preprocessing is required before feature selection, classification is performed in less time.

Feature vectors obtained by the preceding methods constituting the training sets of four different classes (ECG shape changes) are used in the learning process. There are 4*15 feature vectors in the training set, 15 feature vectors coming from each class.

Class prototypes which will be given as inputs to the classifiers are formed during learning.

In order to obtain the cluster prototypes during the learning process fuzzy c-means and fuzzy Kohonen clustering algorithms (FKCA) are used.

Fuzzy c-means algorithm gives optimal results when the number of the clusters in a class is known a priori. But it is difficult to estimate the number of clusters, especially when one is studying in a multi-dimensional (>2) space.

A second algorithm used in this thesis to obtain the cluster prototypes is the fuzzy Kohonen clustering algorithm. Kohonen clustering networks (KCNs) are used for clustering.

Learning is completed after the formation of the cluster prototypes. In this study 4 prototypes are obtained by the fuzzy c-means algorithm and 8 prototypes (class number*k, k any positive number) by the fuzzy Kohonen clustering algorithm.

After these processes, fuzzy classification of the feature vectors belonging to the test set is performed. Two types of classifiers are used in this study: Fuzzy nearest prototype classifier and fuzzy K nearest neighbour classifier.

The nearest neighbour classifiers require no preprocessing of the labeled sample set prior to their use. The crisp nearest-neighbour classification rule assigns an input sample vector y, which is of unknown classification, to the class of its nearest neighbour. This idea can be extended to the K-nearest neighbours with the vector y being assigned to the class that is represented by a majority amongst the K-nearest neighbours. The sample vector is assigned to the class, for which the sum of distances from the sample to each neighbour in the class is a minimum.

While the fuzzy K-nearest neighbour procedure is also a classification algorithm the form of its results differ from the crisp version. The fuzzy K-nearest neighbour algorithm assigns class membership to a sample vector rather than assigning the vector to a particular class. In addition, the vector's membership values should provide a level of assurance to accompany the resultant classification. For example, if a vector is assigned 0.9 membership in one class and 0.05 membership in two other classes we can be reasonably sure the class of 0.9 membership is the class to which the vector belongs. On the other hand, if a vector is assigned 0.55 membership in class one, 0.44 membership in class two, and 0.01 membership in class three, then we should be hesitant to assign the vector based on these results. However, we can feel confident that it does not belong to class three. In such a case the vector might be examined further to determine its classification, because the vector exhibits a high degree of membership in both classes one and two. Clearly the membership assignments produced by the algorithm can be useful in the classification process.

The basis of the algorithm is to assign membership as a function of the vector's distance from its K-nearest neighbours and those neighbours' memberships in the possible classes. The fuzzy algorithm is similar to the crisp version in the sense that it must also search the labeled sample set for the K-nearest neighbour.

The labeled samples can be assigned class memberships in several ways. First, they can be given complete

membership in their known class and nonmembership in all other classes. Other alternatives are to assign the samples' membership based on distance from their class mean or based on the distance from labeled samples of their own class and those of the other class or classes, and then to use the resulting memberships in the classifier. An alternate scheme for assigning initial memberships based on a learning scheme is also considered in this thesis.

Two different methods are used in order to form the prototypes of the fuzzy K-NN classifier. First, the training set is completely used as the class prototypes and their memberships are assigned by the fuzzy c-means algorithm. Second, class prototypes are formed by using fuzzy Kohonen clustering algorithm and crisp memberships (1, complete membership; 0, nonmembership) are assigned to these vectors.

The second classifier used in this study is the fuzzy nearest prototype classifier. These nearest prototype classifiers bear a marked resemblance to the one-nearest neighbour classifier. Actually, the only difference is that for the nearest prototype classifier the labeled samples are a set of class prototypes, whereas in the nearest neighbour classifier we use a set of labeled samples that are not necessarily prototypical. In this study, fuzzy c-means algorithm is used for the formation of the class prototypes. In the fuzzy NP algorithm, membership in each class is assigned based only on the distance from the prototype of the class.

The fuzzy K-nearest neighbour and fuzzy nearest prototype classifiers developed and investigated in this thesis show useful results. By using these classifiers, we obtain a measure of how "typical" the object is of each class. Memberships obtained by these classifiers do provide a useful level of confidence of classification.

The nearest prototype classifier is the quickest and simplest of the classifiers considered. The reason is that in this algorithm, an unknown sample is compared to one prototype per class as opposed to the K-nearest neighbour algorithms, where an entire set of labeled samples representing each class must be compared before the K nearest are obtained.

The two of the learning algorithms used in this study give successful results for the formation of the prototypes that represent the normal beat, left bundle branch block beat and the paced beat shape changes. As the PVC type shape change is represented by more than one cluster in the feature space, its prototypes can not be formed correctly by the fuzzy c-means algorithm. For this reason, fuzzy nearest prototype classifier which uses the prototypes

formed by the fuzzy c-means algorithm, classifies the PVC by a low success rate. As the fuzzy Kohonen algorithm represents the shape changes by more than one prototype, it gives better results compared to the fuzzy c-means algorithm, especially for the PVC type shape change. Fuzzy K-NN classifier gives better results compared to the fuzzy NP classifier for it uses more than one prototype for each class.

The results presented in this thesis were produced by software implementation of the algorithms. The software was developed using C language on PC.

BÖLÜM 1

GİRİŞ

1.1 Çalışmaya Genel Bakış

Tez içerisinde EKG kayıtlarında hızlı ve otomatik olarak şekil bozuklukları tespit edilmekte ve ekranı izleyen kişiye bulunan şekil bozukluğu tüm sınıflara ait olma bilgisi verilerek gösterilmektedir. Gerçeklenen algoritma ile hastalık teşhisine büyük kolaylık sağlandığı düşünülmektedir. Tez içerisinde; özellik vektörü elde etme yöntemleri, öğrenme yöntemleri ve sınıflayıcılar kullanılarak en iyi sonucu veren yöntemler araştırılmıştır.

Tez içerisinde giriş verisi olarak MIT-BIH veri tabanı kullanılmıştır [1]. Herbir EKG dosyasında 30 dakikalık iki farklı derivasyon bulunmaktadır. Derivasyon içerisindeki veriler, 3 ms'de örneklenmiştir ve kayıt içerisinde herbir veri, işaretiyle birlikte 12 bit ile temsil edilir. Çalışma içerisinde II no'lu (MLII, "modified limb lead II") derivasyonda, dört farklı şekle sahip EKG işaretleri sınıflandırılmıştır.

Şekil bozukluğunu tespit eden algoritmayı temel olarak üç bölümde inceleyebiliriz: R tepesinin tespiti, özellik vektörlerinin elde edilmesi ve sınıflayıcı.

EKG işaretlerinden QRS kompleksinin çıkartılması işlemi işaret üzerindeki gürültüden etkilenir. MIT-BIH veri tabanındaki EKG işaretleri incelendiğinde, işaretler üzerinde gürültünün az olduğu sadece bazı kısa periyotlarda etkili olduğu gözlenmiştir. Ancak çalışma içerisinde genel amaçlı bir QRS tespit algoritması düşünüldüğü için

literatürde kullanılan bir yöntem seçilmiştir [2]. Algoritmada EKG işareti önce 2 Hz'e kilitli yüksek geçiren bir filtreden (YGF) geçirilir. Bu filtrenin çıkışı iki kola ayrılır. Bir koldan YGF çıkışı, band geçiren filtreye (BGF) verilir; diğer koldan ise YGF çıkışı, alçak geçiren filtreye (AGF) verilir. BGF'nin çıkışı kare alma ve daha sonra integral alma işlemlerinden geçirilir. AGF'nin çıkışı geciktirilerek integral alma bloğunun çıkışı ile çarpılır. Sonuçta elde edilen işaret, bir eşik ile karşılaştırılarak QRS deteksiyonu gerçeklenir. Elde edilen işaret üzerinde maksimum genlik değeri, R tepesi olarak belirlenir.

R tepesinin yerinin EKG işareti içerisinde doğru tespit edilmesi son derece önemlidir. R tepesi belirlendikten sonra bu tepe etrafında özellik vektörleri bulunur. Özellik vektörlerinin R tepesinin yerine bağlı olması istenmez. Çalışma içerisinde 4 farklı özellik elde etme yöntemi incelenmiştir: 1) QRS şekil bilgisinden özellik vektörlerinin çıkartılması [3], 2) 0-30 Hz arası frekans bandı kullanılarak özellik vektörlerinin çıkartılması [4], 3) Lineer tahmin katsayıları kullanılarak özellik vektörlerinin çıkartılması [5], 4) R tepesi etrafında EKG işaretinin genlikleri kullanılarak özellik vektörlerinin çıkartılması [6].

Özellik vektörlerinin mümkün olduğu kadar az parametre ile (az bellek elemanı kullanarak) hızlı bir şekilde elde edilmesi ve vektörlerin R tepesinin yerinden etkilenmemesi istenir. Birinci ve dördüncü özellik elde etme yöntemi R tepesinden etkilenir; ancak vektörler hızlı bir şekilde elde edilmektedir. Diğer özellik elde etme yöntemleri R tepesinden etkilenmez; ancak vektörleri elde etmek için geçen işlem zamanı uzamıştır.

Öz-vektörlerin öğrenilmesi işleminden önce, üç farklı kişiden alınan MLII derivasyonlarından dört farklı şekil

(sınıf) için eğitim kümesi oluşturulur. Her sınıf için 15 adet olmak üzere eğitim kümesinde 4*15 adet vektör bulunur. Çalışma içerisinde iki farklı öğrenme algoritması kullanılmıştır: Bulanık küme ortalar [7] ve bulanık Kohonen kümeleme algoritması [8]. Bulanık sınıflayıcılar kullanıldığı için EKG işareti, direkt olarak bir sınıfa dahil edilmez; bunun yerine incelenen işaretin tüm sınıflara yakınlık derecesini belirten bir bilgi (üyelik değeri) sunulur. Klasik sınıflayıcılarda ise giriş vektörü direkt bir sınıfa dahil edilir; giriş vektörünün bu sınıfa yakınlığı hakkında bir bilgi verilmez.

Bulanık küme-ortalar algoritması, küme sayısı önceden bilindiğinde iyi sonuç vermektedir. Ancak kümelenme sayısını önceden tahmin etmek (tam değerini belirlemek) zordur. Bulanık Kohonen algoritmasında ise çıkış düğüm sayısı, kümelenme sayısına eşit veya büyük olarak seçildiğinde iyi sonuç vermektedir. Çalışma içerisinde değişik özellik vektörleri için her iki öğrenme algoritmasının başarısı incelenir. Tez çalışmasında iki farklı bulanık sınıflayıcının başarısı incelenmiştir: Bulanık K-en yakın komşu sınıflayıcısı ve bulanık en yakın model sınıflayıcısı [10]. Bulanık en yakın model sınıflayıcısının öz-vektörlerini bulmak için bulanık küme-ortalar öğrenme algoritması kullanılır. Bulanık K-en yakın komşu sınıflayıcısının öz-vektörleri iki farklı şekilde elde edilir. Birinci yöntemde eğitim kümesinin tamamı öz-vektör olarak düşünülür ve bu vektörlerin üyelikleri bulanık küme ortalar tarafından bulunur. İkinci yöntemde, bulanık Kohonen algoritması kullanılır, ancak öz-vektörlerin üyelik değerleri duru olarak (1 veya 0 olarak belirlenir; ara değer almaz) belirlenir.

Bölüm 2'de, EKG işaretlerinin değişim şekli, oluşumu, derivasyon bağlantıları, incelenen EKG şekil bozuklukları ve çalışmanın üzerinde yapıldığı MIT-BIH veri tabanı anlatılmıştır. Bölüm 3'te QRS kompleksini tespit etmek

için kullanılan filtrelerin tasarımı ve R tepesi tespit algoritması anlatılmıştır. Bölüm 4'te dört farklı özellik vektörü oluşturma yöntemleri incelenmiştir. Bölüm 5'te iki farklı bulanık öğrenme yöntemi ve iki farklı bulanık sınıflayıcı anlatılmıştır. Bölüm 6'da çalışmanın pratik uygulaması verilmiştir.

BÖLÜM 2

ELEKTROKARDİYOGRAFİ

2.1 Giriş

İnsan vücudu üzerinden algılanan ve kalbin elektriksel aktivitesinin sonucu olarak ortaya çıkan belli tipteki biyolojik işaretlere elektrokardiyogram, elektrokardiyografik işaret, EKG işareti veya kısaca EKG adı verilir. EKG işaretinin gösterilmesini veya kaydedilmesini sağlayan cihazlara Elektrokardiyograf ve EKG ile ilgili sistem ve olgulara da genel olarak Elektrokardiyografi denir. İnsan ölümlerinin büyük bir yüzdesini kalp rahatsızlıkları oluşturmaktadır. Bu yüzden kalbin çalışması sırasındaki bozuklukların iyi bir göstergesi olan ve insan vücudu üzerinden, yaralama, deşme yapmadan kolaylıkla elde edilebilen EKG işaretleri, işlenme ve yorumlama açısından büyük önem taşımaktadır. Bu bölümde EKG işaretlerinin değişim şekli ve oluşumu üzerinde kısaca durulduktan sonra, derivasyon bağlantıları, EKG işaretinin aldığı normal değerler, EKG'de görülen ve tez içerisinde tespit edilmeye çalışılan şekil bozuklukları incelenmektedir.

2.2 Biyolojik İşaretler

Biyolojik sistemler yaşam fonksiyonlarını sürdürürken yaşamı için gerekli birçok işaretler üretirler. Canlı doku, hücre, organ ve sistemlerin hayati fonksiyonlarını yerine getirmeleri sırasında oluşturdukları elektro kimyasal işaretlere biyolojik işaretler (biyoelektrik potansiyeller, biyopotansiyeller) adı verilir. Bu

işaretler canlının ve özel olarak insanın iç haberleşme, kontrol ve zihinsel aktivitesinin sonucu olarak ortaya çıkarlar. Biyolojik işaretler genellikle fark işaretleri şeklinde olup küçük genliklidirler. Bu işaretlerin algılanmasında kullanılan dönüştürücüler, kimyasal-elektrik dönüşümünü yaptıklarından özel olarak "elektrodlar" olarak adlandırılırlar.

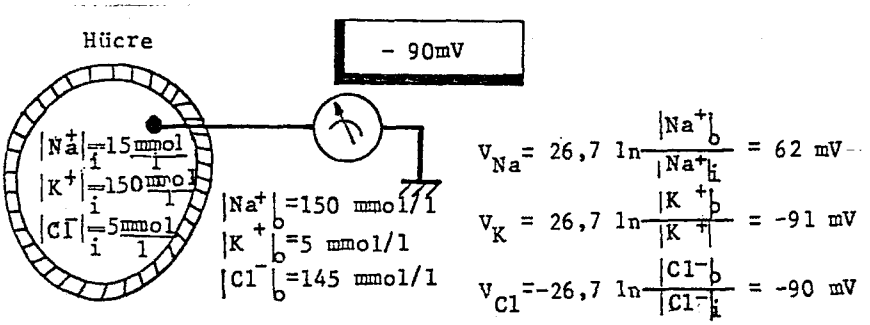
Sinirler yolu ile duyu organlarından ve duyu algılayıcılarından (reseptörlerden) bilgilerin beyine, beyinden emirlerin kaslara (efektör organlara) iletimi, istemli ve istemsiz kasların faaliyetleri, beynin zihinsel aktivitesi ve kalbin elektriksel faaliyetleri sırasında hep bu işaretler oluşur. Bu doku ve organların çalışmalarının normal olup olmadıklarını anlamak için bu işaretler önemli olur. Bu işaretlerden genel olarak deri üzerinden algılanan ve klinik çalışmalar için önemli olanları; beyin, kalp, göz, sinir ve kaslardan alınan işaretler olup kaynaklandıkları yere göre sırası ile EEG (elektroensefalogram), EKG (elektrokardiyogram), ERG (elektroretinogram), ve EMG (elektromiyogram) isimlerini alırlar. Bu işaretlerin temelinde hücrenin elektriksel aktivitesi vardır. Gerçekte ise herhangi bir organa (sisteme) ait biyolojik işaretler, o organı (sistemi) oluşturan hücrelerin elektriksel aktivitelerinin süperpozisyonu (toplamı, bileşkesi) olarak algılanırlar.

2.3 Aksiyon Potansiyeli

Hücre, biyoelektrik potansiyel kaynağıdır. Biyoelektrik potansiyel, hücrenin içi ile dışı arasındaki potansiyel farkıdır, ki geniş bir hacim oluşturan hücre dışı ortamı (ekstrasellüler sıvı ortamı) bu potansiyel farkının referans (nötr) noktasını teşkil eder. Yani, sözkonusu olan biyoelektrik potansiyel, hücre içinin dışına göre olan potansiyelidir. Hücre içinde bu potansiyelin

oluşunun nedeni, hücrenin, iyonlara karşı seçici geçirgenliği olan bir zarla (membranla) kaplı (örtülü) olmasıdır. İnsan vücudundaki elektrik akımı, iyonların hareketi (iyonik akım) şeklindedir. Biyoelektrik potansiyel farklarına neden olan da iyonların bölgeler arası (hücre içi ile dışı arası) konsantrasyon farklarıdır. Normal dinlenme durumundaki hücrenin içi ile dışı arasındaki iyonların -ki başlıcaları K^{+1}, Na^{+1} ve Cl^{-1}'dir- konsantrasyonları, hücre uyarıldığında, hücre zarının bu iyonlara olan geçirgenliğinin değişmesi nedeniyle farklılaşır ve bunun sonucu olarak da hücre içi potansiyelinde değişme olur. Hücre içi potansiyelinin hücre uyarıldıktan sonraki bu değişimi "aksiyon potansiyeli" olarak isimlendirilmektedir. İyon konsantrasyonlarının hücre içi ve dışında farklı konsantrasyonlarda, bir denge içinde içinde bulunması pasif ve aktif transport olayları ile açıklanabilmektedir [11].

Hücre içi potansiyelini (membran potansiyeli, V_m) belirleyen başlıca iyonlar Na^{+1}, K^{+1} ve Cl^{-1}'dir. Şekil 2.1'de bu iyonların hücre içi ve dışındaki konsantrasyonlarının ve hücre içi potansiyelinin, dinlenme durumu sözkonusu olduğundaki değerleri gösterilmiştir.



Şekil 2.1: Dinlenme durumundaki hücre.

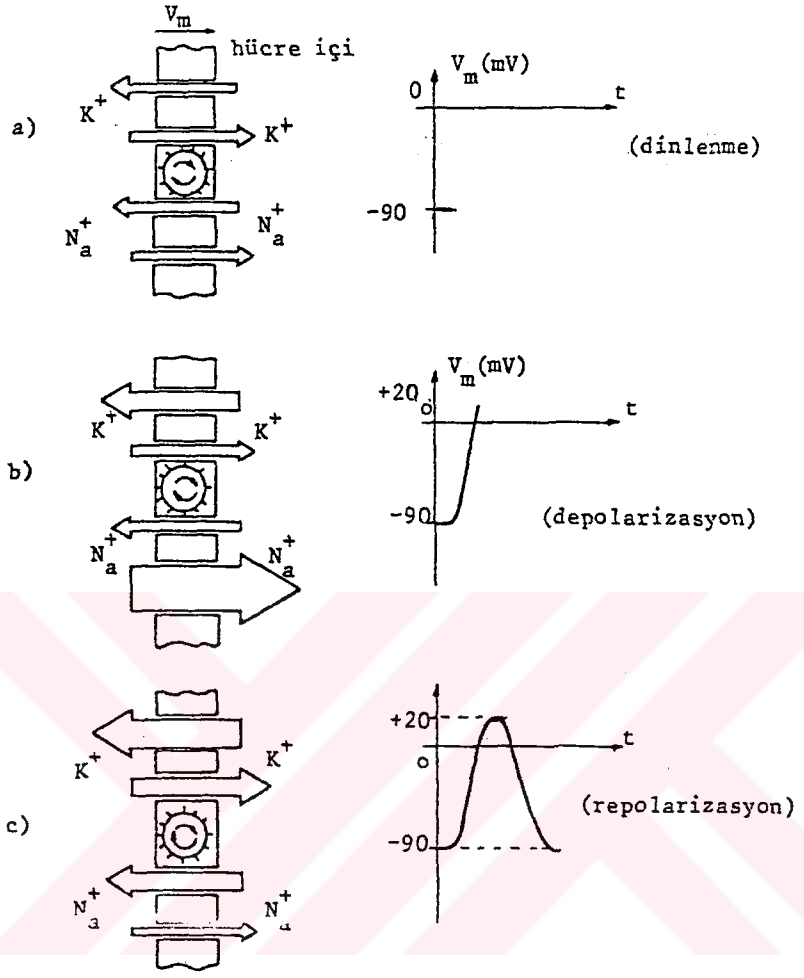
Dinlenme durumundaki hücre içi potansiyeli çeşitli hücreler için farklı değerlerde olmaktadır. Örneğin kalp kası için -95mV, sinüs düğümü için -65mV, iskelet kası

için -70mV, insan alyuvarlarında -10mV olmaktadır. Bu potansiyel nedeniyledir ki hücre dinlenmede (sükunette) iken "polarize olmuştur denir. Bu nedenle hücre içi potansiyelinin dinlenme durumundaki değerine de dinlenme (sükun) potansiyeli veya "polarizasyon gerilimi" adı verilir. Hücre uyarılmadığı durumda her zaman için bu denge durumunda kalmaya çalışır.

Işık, mekanik, magnetik, ısı ve daha çok elektrik (akım veya gerilim) veya kimyasal etkiler ile hücre uyarıldığında hücre zarının Na^+ iyonlarına olan geçirgenliği (μ_{Na}) 500 kat kadar artar ve Na^+ iyonları hızla hücre içine girerek hücre içi potansiyelini pozitif bir değere getirmeye çalışır. Na^+ iyonlarına olan geçirgenliğin bu kadar artmasına uyarılma durumunda Na kapılarının iyice açılması neden olmaktadır. Hücre uyarıldığında K kapıları da Na kapıları kadar olmamakla beraber biraz daha açılır ve K^+ iyonlarının hücre dışına çıkışı biraz artar (Şekil 2.2b).

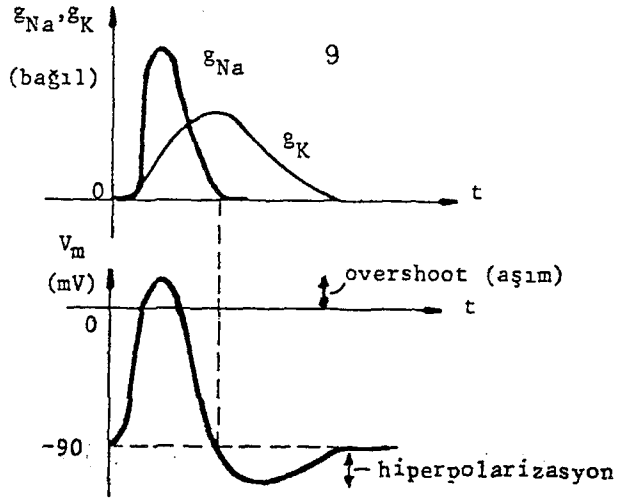
Na kapıları açılarak Na^+ iyonlarının bir çığ gibi içeri girmesinin sonucunda, hücre içi potansiyeli +20mV değerine ulaşır. Hücrenin uyarılması sonucu oluşan bu olaya "depolarizasyon" adı verilir (Şekil 2. 2.b). Depolarizasyon süresi, normal hücrelerde 1-2ms kadardır.

Hücre içi potansiyeli +20mV olunca, gerilime duyarlı Na kapılarının bir kısmı kapanır ve Na^+ iyonlarına karşı hücre zarının geçirgenliği çok çabuk eski durumuna gelir, bu sırada K^+ iyonlarına olan hücre zarı geçirgenliği üst düzeydedir. Hücre içine giren Na^+ kadar K^+ iyonunun hücre dışına çıkması ile hücre içi potansiyeli dinlenme durumundaki değerine gelmiş olur ki bu olaya da hücrenin tekrar polarize olması anlamına gelen "repolarizasyon" adı verilir (Şekil 2.2.c).



Şekil 2.2: Hücre uyarıldığında iyonların geçiş durumu ve hücre içi potansiyelindeki değişim,
a) hücre dinlenmede b) depolarizasyon ve
c) repolarizasyon durumları.

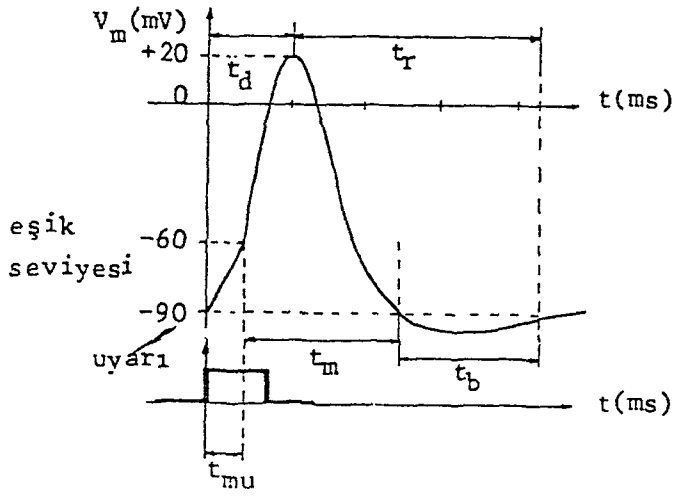
Hücrenin depolarize ve repolarize olması sürelerinde hücre zarının Na^+ ve K^+ iyonlarına olan geçirgenlikleri ile hücre zarı potansiyelinin zamana göre değişimleri Şekil 2.3'te temsili olarak gösterilmiştir.



Şekil 2.3: Depolarizasyon ve repolarizasyon durumlarında Na^+ ve K^+ iletkenliklerinin değişimi ve buna bağlı olarak hücre içi potansiyel değişimi.

Hücre içi potansiyeli polarizasyon değerine yaklaşırken hücrenin K^+ iyonlarına olan geçirgenliği üst düzeydedir ve bu nedenle bir süre daha hücre dışına K^+ taşınması devam eder (g_K'nın azalma eğimi daha düşüktür, Şekil 2.3) ve bu durum hücre içi gerilimini polarizasyon değerinin daha da altına düşürür. Bu durumdaki hücre "hiperpolarize" olmuştur.

Aksiyon potansiyeli olarak bilinen hücre içi potansiyelinin uyarmadan sonra bu şekilde değişimi, uyaran ile birlikte Şekil 2.4'te yeniden çizilmiştir.

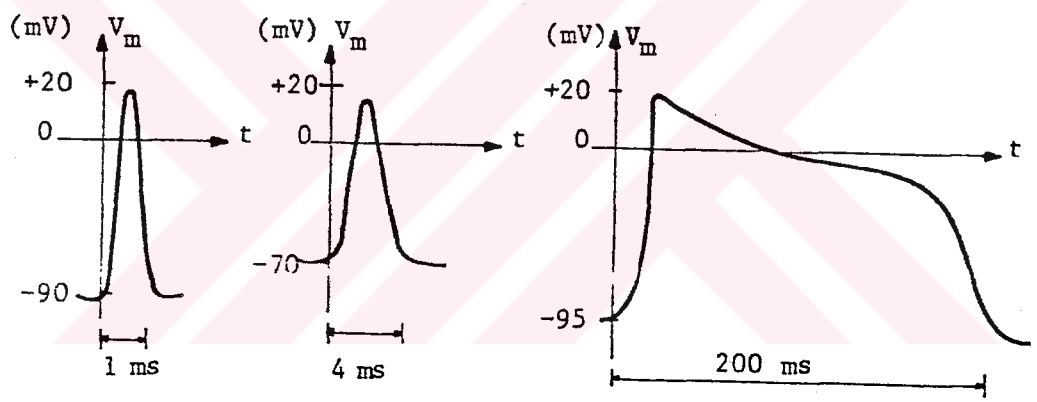


t_d: depolarizasyon süresi
t_r: repolarizasyon süresi
t_m: mutlak bekleme süresi
t_b: bağıl bekleme süresi
t_{mu}: minimum uyarı süresi (verilen genlikteki uyarı için)

Şekil 2.4: Uyarı ve aksiyon potansiyelinin değişimi.

Hücre uyarıldıktan sonra en şiddetli uyaranlara bile cevap veremediği kısa bir süre vardır ki bu süreye "mutlak bekleme süresi (absolute refractory period)" adı verilir (Şekil 2.4). Mutlak bekleme süresinden sonra hücrenin ancak şiddetli uyaranlara cevap verebileceği "bağıl bekleme süresi" vardır.

Aksiyon potansiyelinin değişimi çeşitli hücrelerde farklılıklar gösterir (Şekil 2.5). Sinir ve çizgili kastaki aksiyon potansiyeli, süre ve genlik bakımından az-çok farkeder. Kalp kasında ise şekil biraz değişiktir. Kalp kası potansiyelinde, 0mV'da kaldığı 200ms kadar uzunca süreli bir plato vardır ve daha sonra bu potansiyel hızla dinlenme potansiyeline düşmektedir.



Şekil 2.5: Çeşitli hücrelerde aksiyon potansiyelinin değişimi

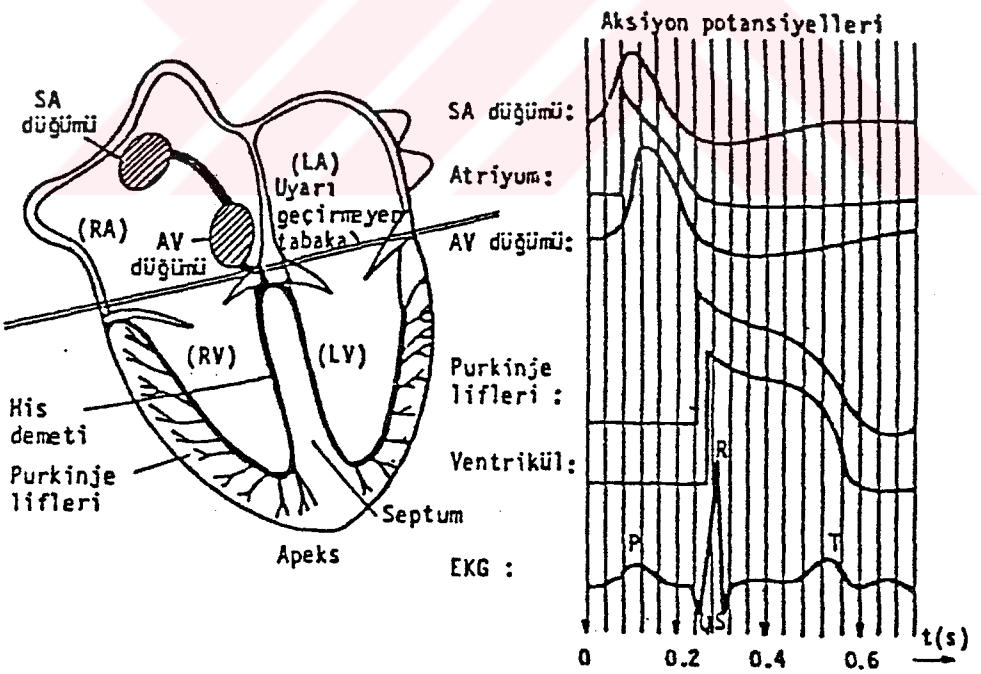
2.4 Elektrokardiyogram

İnsan vücudundaki tüm kaslar biyoelektrik uyarılarla kasılmakta ve bunun yanında kendine has biyolojik işaretler üretmektedir. Uyarılan kasın yakınına konan bir çift elektrod aracılığıyla, kasın kasılması sırasında ortaya çıkan elektriksel işaretlerin izlenmesi mümkündür. Kalp kasları da belli bir ritimde kendi içindeki sinüs düğümü (SA düğümü) tarafından periyodik olarak kendiliğinden

üretilen elektriksel işaretlerle uyarılmakta, bu uyarı sonucunda da kalp periyodik olarak kasılıp gevşeyerek vücudun ihtiyacı olan kanı pompalamaktadır. Kalp dört ayrı pompadan oluşur; bunlar iki primer pompa, atriyumlar (kulakçık) ve iki güç pompası, ventriküllerdir (karıncık) (Şekil 2.6). Dokulardan gelen kirli kan sağ kulakçığa, akciğerlerden gelen temiz kan sol kulakçığa girer. Kulakçıkların kasılmasıyla kan karıncıklara geçer. Karıncıkların kasılmasıyla, kirli kan akciğer atardamarına, temiz kan aorta aktarılır. Kapaklar kapatılır. Bu pompalama işlemi için kalp kaslarının bir düzen içinde kasılması gerekmektedir ki bu, kalbin elektriksel iletim sistemi yardımıyla olur. Kalbin iletim sistemi; normal ritmik uyarıların doğduğu SA düğümü (sino-atrial düğüm), uyarının SA düğümünden AV düğümüne (atriyo-ventriküler düğüm) iletildiği internodal yollar, uyarının atriyumlardan ventriküllere geçerken gecikmeye uğradığı AV düğümü, uyarıyı atriyumlardan ventriküllere ileten AV demeti ve uyarıyı ventriküllerin bütün bölümlerine ileten Purkinje liflerinin sol ve sağ dallarından oluşur [12].

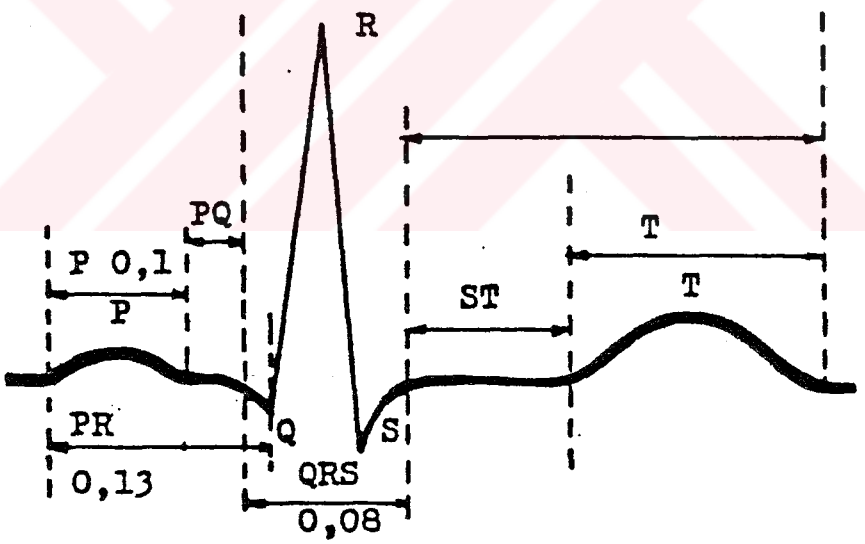
Normal EKG işareti, kalbin dinlenme durumundaki izoelektrik seviyesi (taban seviyesi) üzerinde sıralanan belli başlı P, Q, R, S ve T adları verilen dalgalardan oluşur (Şekil 2.7). S-A düğmünde meydana gelen depolarizasyon dalgası, kulakcıklar içindeki iletim yolu üzerinde 30 cm/s hızla ilerlerken kulakcık kaslarını uyarır ve bunun sonucunda kulakcıklar bir bütün olarak kasılarak kendi içinde kalmış kanın tümünü karıncıklara pompalar. Kulakcıkların kasılması sırasında EKG işaretinde P dalgası adı verilen değişim oluşur. 0,1 s kadar olan P dalgasının genliği, kulakcık kasının aktivitesi hakkında bilgi verir. Depolarizasyon dalgası kulakcıklarla karıncıklar arasındaki AV düğümüne gelince hızı 5 cm/s'ye kadar düşer. Bu sayede, AV düğümü, bir gecikme elemanı gibi görev yaparak aksiyon potansiyel dalgasının karıncıklara, kulakcıklar kasıldıktan 0,1 s sonra ulaşmasına neden olur. Aksiyon potansiyeli

dalgası, his demeti ve purkinje lifleri yardımıyla tüm karıncık kaslarına 3 m/s gibi bir hızla yayılınca karıncık bir bütün olarak kasılır. Karıncıkların kasılması sırasında EKG işaretinde 0,1 s kadar süren ve QRS kompleksi adı verilen değişim gözlenir. Q, R ve S dalgalarından oluşan QRS kompleksinin 0,1 s'den daha uzun sürmesi karıncıkların hemzaman olarak uyarılmadıklarını gösterir. Karıncık kaslarının kulakcık kaslarından daha çok ve güçlü olması nedeniyle QRS kompleksindeki R dalgası, P dalgasından genlik olarak oldukça büyüktür. Bunun yanında R dalgasının genliğinin normalden büyük olması hipertansiyonda kalp büyümesinin bir göstergesi olabilir. S dalgasından sonra karıncık kasları aynı potansiyeldedir ve bu nedenle T dalgasına kadar EKG işareti izoelektrik seviyededir. Bu seviye ST segmenti olarak bilinmektedir.



Şekil 2.6: Kalbin elektriksel iletim sistemi, kalbin çeşitli bölgelerdeki aksiyon potansiyelleri, ve EKG.

Karıncık kaslarının hızlı repolarizasyonu sonucunda T dalgası oluşur. EKG işaretinde dalgalar arasında kalan uzaklıklara "aralık" adı verilir. PR aralığı, his demeti iletim zamanını gösterir. PR aralığının 0,2 s'den uzun olması, his demetinde bir ileti bozukluğu olduğunu belirtir. ST aralığı özel olarak ST segmenti olarak isimlendirilir ve normalde izoelektrik seviyededir. ST segmentinin izoelektrik seviyeden olan sapmaları ve anormal uzunluğu da kalbin normal çalışmayışının göstergesi olarak kullanılmaktadır. İki dalgası arası uzaklığa RR aralığı veya bir kalp periyodu adı verilir. RR aralığı, kalbin dakika vurum hızının bir göstergesidir ve kalp rahatsızlıklarının teşhisinde kullanılan parametrelerden biridir.



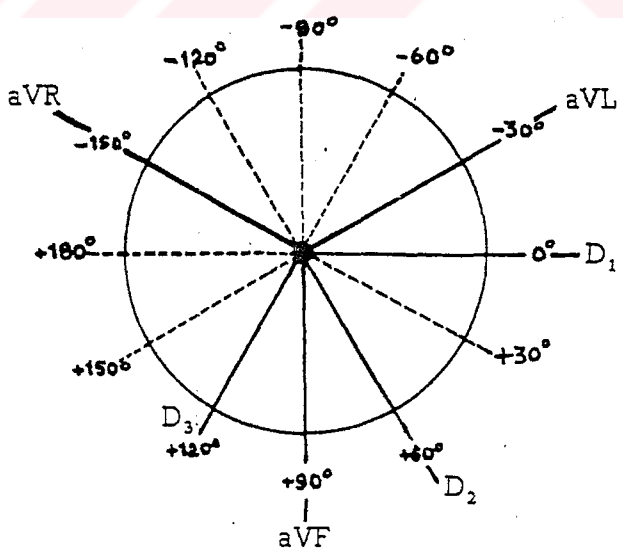
Şekil 2.7: EKG işareti üzerindeki tipik dalga şekilleri ve aralıkları.

2.5 Derivasyon Bağlantıları

2.5.1 Ekstremite Derivasyonları

I, II, III adı verilen üç standart ekstremite derivasyonu bipolar derivasyonlardır. Yani, kayıt aletinin pozitif ucu bir ekstremitede bulunan bir elektroda, negatif ucu da diğer ekstremitede bulunan başka bir elektroda bağlıdır. Kullanılan üç ekstremite sağ kol, sol kol ve sol bacaktır (bunlar için sırasıyla R, L ve F harfleri kullanılır). Standart derivasyonların gerçek bağlantıları şöyledir [13]:

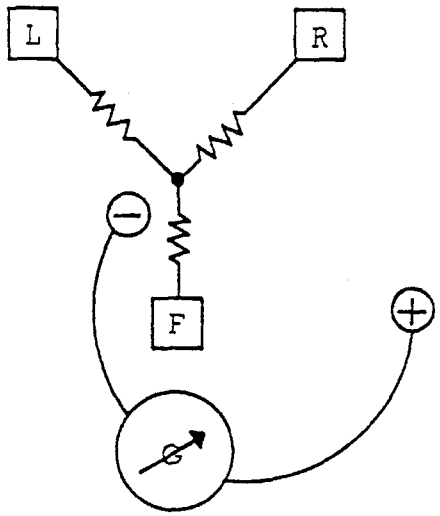
	Pozitif uç	Negatif uç
Standart I (D_I)	L	R
Standart II (D_{II})	F	R
Standart III (D_{III})	F	L



Şekil 2.8: Kalbin elektriksel aktivitesinin vücut yüzeyinden kaydedildiği eksen sistemleri.

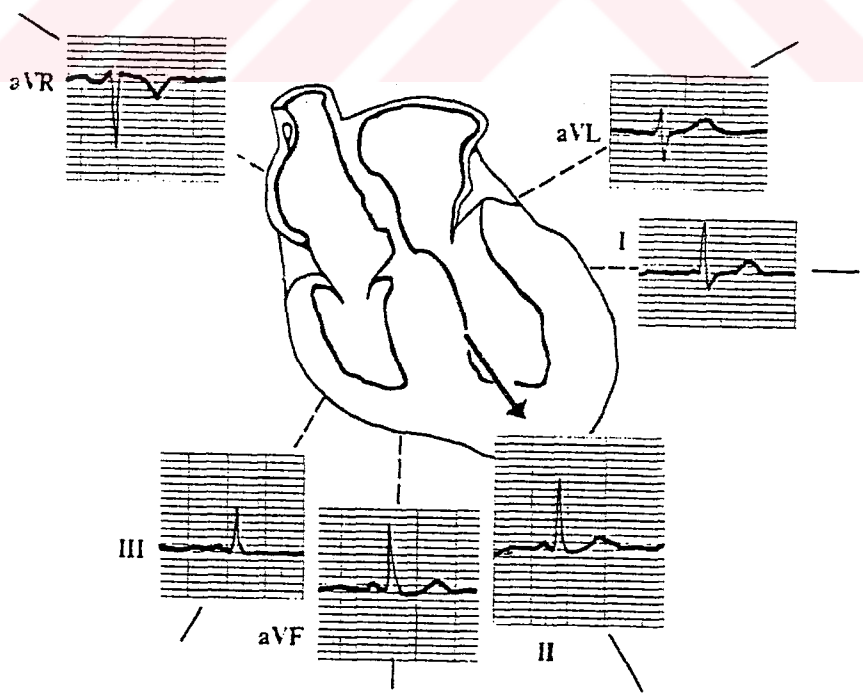
Şekil 2.8'de, ortadaki nokta elektrik güçleri doğuran merkez yani kalptir. Eksenin düz çizgisi pozitif, kesikli çizgisi negatif yönü göstermektedir. D_1, D_2, D_3 eksenleri galvonometrenin (-) ve (+) uçlarının bağlantı noktaları arasından geçirilen çizgilerdir. Bu çizgiler, yani eksenler, frontal düzlemde çizgilerdir. Elektrik akımı doğuran kalp, gövde içinde bir nokta gibi kabul edilebilir. Yukarıdaki teknikle, akımı kayıt için kullanılan eksenler, bir merkezden geçecek şekilde dizilirse yukarıda gösterilen yerleşme düzeni elde edilmiş olur. Burada galvanometrenin iki ucu da kullanıldığı için bu şekilde elde edilen derivasyonlara "bipolar" derivasyonlar denir.

Geriye kalan üç ekstremite derivasyonu ünipolar derivasyonlardır. Bu derivasyonlarda aletin (galvanometre) negatif tarafı kaydedilen potansiyelleri etkilemeyen nötr bir uca yahut sıfır noktasına bağlanır (Şekil 2.9). Nötr bağlantı, üç ekstremiteden gelen derivasyonları, araya konulan dirençler vasıtasıyla kaydedici aletin negatif tarafındaki bir santral uca (Vilson'un santral terminali=V) birleştirmek suretiyle elde edilir. Üç noktadan gelen potansiyellerin birbirlerini yokederek sıfır potansiyel ile sonuçlandığı gösterilebilir. Kaydedici aletin diğer tarafı potansiyellerin kaydedildiği pozitif yahut araştırıcı elektroda bağlanır. 'V' harfi ünipolar bir derivasyonu gösterir. Normal olarak kaydedilen üç ekstremite derivasyonu VR, VL ve VF olarak adlandırılır (buna göre araştırıcı elektrod sırası ile sağ kolda, sol kolda ve sol bacakta yer almaktadır). Modern uygulamada ünipolar ekstremite derivasyonlarından elde edilen genlikler küçük oldukları için %50 oranında kuvvetlendirilirler. Bu suretle ünipolar ekstremite derivasyonlarının adları aVR, aVL, aVF olarak değiştirilmiştir.

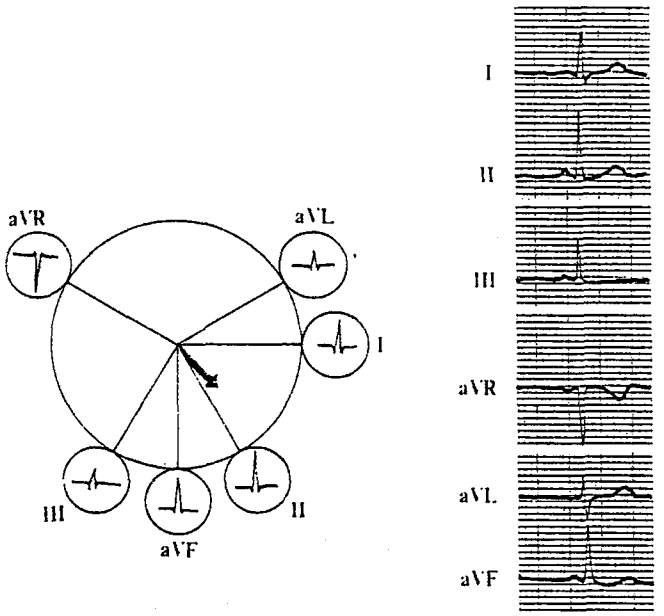


Şekil 2.9: Ünipolar derivasyonların elde edilme şekli.

Ventriküler kompleksin, yukarıda izah edilen altı adet ekstremite derivasyonuna yansıması şekil 2.10.a'da görülmektedir.



Şekil 2.10.a: Ventriküler kompleksin ekstremite derivasyonlarına yansıması.



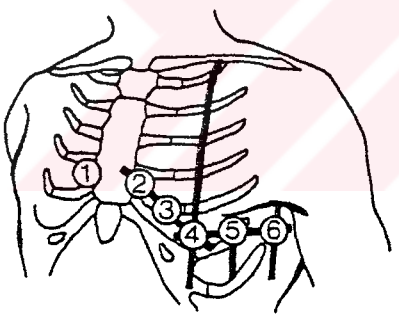
Şekil 2.10.b: Normal (semi-vertikal) elektriksel durum.

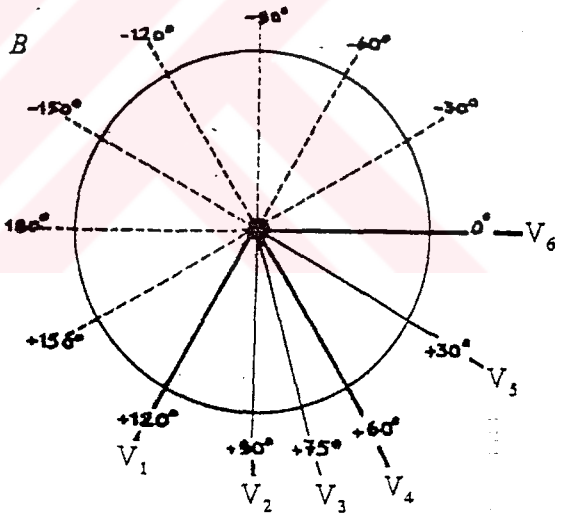
Asıl ventriküler uyarı fazının (depolarizasyon) ortalama yönü (elektriksel eksen) hakim QRS defleksiyonunu tayin eder. Semivertikal (yarı-dik) durumdaki normal bir kalpte bu eksen en çok D_{II} derivasyonunun yönüne uyar (şekil 2.10.a ve şekil 2.10.b'deki oklara dikkat ediniz). Bu sebeple en yüksek R dalgası D_{II} derivasyonunda görülür. aVR'de S (negatif) defleksiyonu hakimdir, çünkü bu derivasyon D_{II} derivasyonunun aksi yönünde kayıt yapar. En küçük ventriküler defleksiyonlar normal olarak elektriksel eksene hemen hemen dik bir açıda yer alan aVL derivasyonunda görülür (şekil 2.10.a ve şekil 2.10.b'ye bakınız) [13].

2.5.2 Göğüs Derivasyonları (Ünipolar Prekordiyal Derivasyonlar)

Uluslararası anlaşmaya göre altı göğüs derivasyonunun yerleştirilme noktaları aşağıdaki şekilde tarif edilmiştir (Şekil 2.11.a).

1. Dördüncü interkostal aralığın sternumun sağ kenarı ile birleştiği nokta,
2. Aynı aralığın sternumun sol kenarı ile birleştiği nokta,
3. İkinci ve dördüncü noktaların tam ortası,
4. Beşinci interkostal aralığın sol medioklaviküler çizgi ile kesiştiği nokta,
5. Dördüncü nokta seviyesinden geçen yatay çizginin ön koltuk çizgisi ile kesiştiği nokta,
6. Dördüncü ve beşinci noktadan geçen çizginin orta koltuk çizgisi ile kesiştiği nokta.





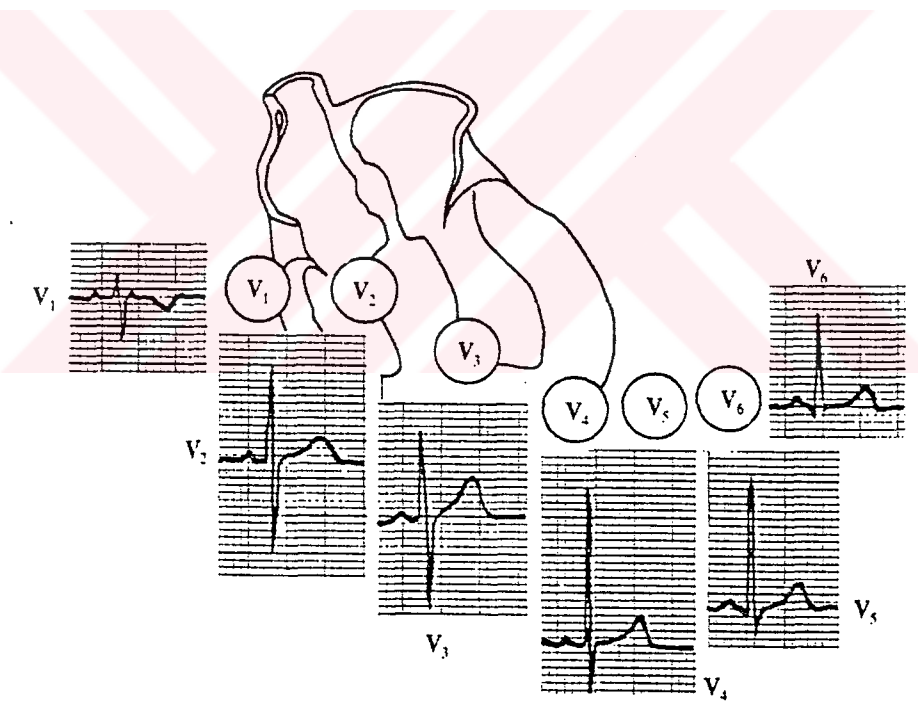
(a) (b)

Şekil 2.11: a) Göğüs derivasyonları için standart noktalar.

b) Göğüs derivasyonlarının geometrik yerleşimi.

Yukarda, şekil 2.9'da tarif edilmiş olan ünipolar elektrokardiyografi tekniği ile araştırıcı elektrod bu resimde gösterildiği şekilde gögüs duvarına yerleştirilirse "ünipolar prekordiyal" derivasyonlar elde edilmiş olur. Standart V_1, V_2, V_3, V_4, V_5, V_6 derivasyonları olarak bilinen bu derivasyonlar yatay (horizontal) planda yer alan derivasyonlardır. Bu şekilde elde edilen prekordiyal derivasyonların geometrik yerleşimleri şekil 2.11.b'de gösterilmiştir. Yatay düzlemde düz çizgi eksenlerin (+) tarafını, kesikli çizgi ise negatif tarafını gösteriyor.

Gögüs üzerindeki bu noktaların altındaki kalple münasebeti ve bu derivasyonlardan elde edilen tipik normal EKG örnekleri şekil 2.12'da gösterilmiştir.



Şekil 2.12: Gögüs derivasyonlarının kalp ile münasebeti.

2.6 QRS Şekil Bozuklukları

Bu bölümde, tez içerisinde tespit edilmeye çalışılan biri normal olmak üzere dört farklı EKG şekil bozukluğu

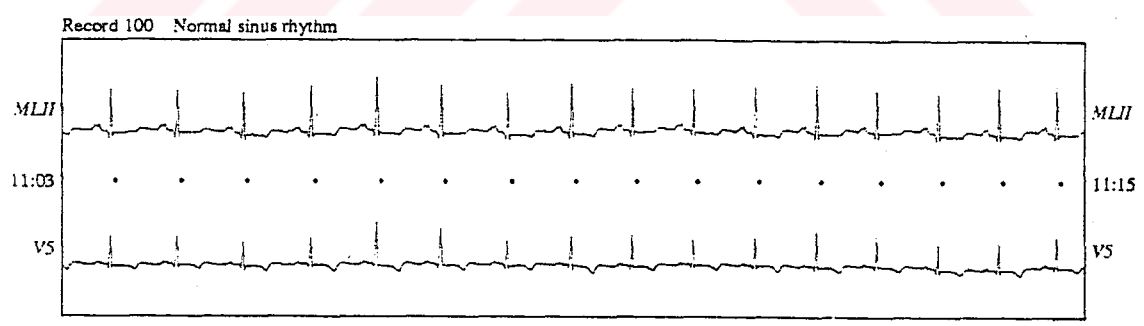
anlatılacaktır [14]. Bunlar N, normal vuru; L, sol dal bloğu (left bundle branch block beat); PVC, erken karıncık kasılması (premature ventricular contraction); P, yapay vurusu (paced beat)'dur.

2.6.1 Normal Vuru (N)

Normal vuru şekil 2.13'te görülen dalga şekline sahiptir. Normalde P dalgasının süresi, 0.10 saniyedir. 0.10-1.12 s arasında ise geniş P dalgasından bahsedilebilir.

PR süresi, P dalgasının başından QRS kompleksinin başlangıcına kadar olan süredir. 0.12-0.20 s arasındadır; ortalama 0.16 saniye bulunur. PR süresi genellikle kalp hızının artması ile kısalır, yaş ile uzar.

QRS süresi erişkinlerde 0.06-0.10 s arasındadır. Prekordiyal derivasyonlarda (V_1-V_6) QRS süresi genellikle 0.01-0.02 saniye daha uzundur.



Şekil 2.13: Normal EKG vurusu.

P, PR, QRS süreleri normal olarak bipolar ve ekstremite derivasyonlarında ölçülür.

QT süresi, QRS kompleksinin başlangıcından T dalgasının sonuna kadar olan aralıktır. En belirgin olan herhangi bir derivasyondan ölçülebilir. QT süresi kalp hızına bağlı olarak değişiklik gösterir. Bu bakımdan hıza

göre gerçek değerinin hesaplanması gerekir. Bu süre normal erişkinlerde 0.35-0.44 s arasındadır.

T dalgasının normal süresi, 0.10-0.25 s arasındadır. Ancak QT süresinin içine girmiş olduğu için klinik elektrokardiyografide T dalgasının süresi ayrıca ölçülmez.

QRS süresinin uzaması, intraventriküler iletimin gecikmiş olduğu anlamına gelir. En çok dal bloklarında görülür.

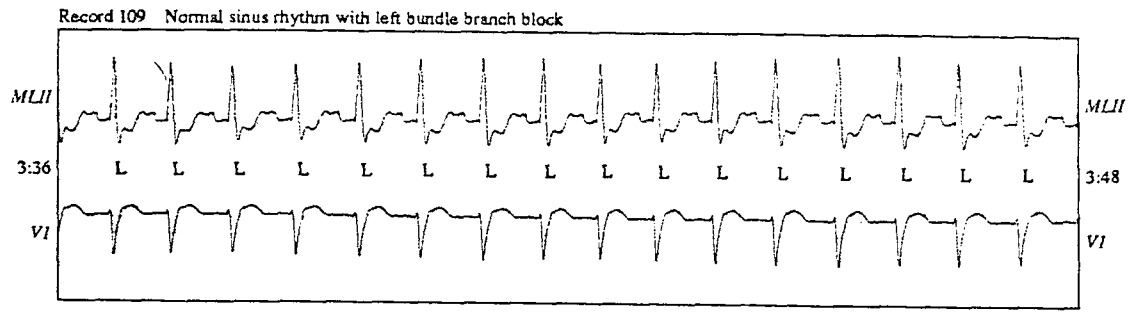
2.6.2 Sol Dal Bloğu (Left Bundle Branch Block, L)

Sol dal bloğunda sol ventrikülün aktivasyonu gecikir. Sol ventrikül aktive olmaya başladığı zamanda sağ ventrikülün bu potansiyellere karşı gelen potansiyelleri artık kalmamıştır. Bu yüzden karşılıksız kalan, dengelenemeyen sol ventrikül potansiyelleri büyük sol ventrikül vektörleri oluştururlar. Bundan dolayı sol dal bloğunda ortaya çıkan QRS vektörlerinin genliği yüksek olabilir.

QRS süresi 0.12 saniye veya daha uzundur. Ventrikül aktivasyonunun gecikmesi, QRS'nin orta ve son kısımlarında olur. Ventrikül aktivasyonunun orta ve son kısımlarında oluşan vektörler genellikle sola arkaya ve yukarıya yönelmiş olduğu için V_5, V_6 derivasyonlarında geniş, yukarı yönelmiş, çentikli R dalgası bulunur (Şekil 2.12).

Komplikasyonsuz sol dal blokunda QRS dalgasının aksi yönde ST-T dalgası oluşur.

Klinikte sol dal bloğunun önemi, hasta hikayesi ve fizik muayene bulguları ile tespit edilir. Ancak, istatistik olarak sol dal bloğu vakalarının büyük çoğunluğunda (%90-95) önemli bir kalp hastalığı vardır.



Şekil 2.14: EKG işaretinde gözlenen bir sol dal bloğu.

2.6.3 Erken Karıncık Kasılması (Premature Ventricular Contraction, PVC)

Erken karıncık kasılması birçok kardiyak ritminde ortaya çıkan bir şekil bozukluğudur. AV düğümü altındaki bölgeden kaynaklanan elektriksel işaret, beklenenden daha önce oluşursa bu PVC olarak kendini gösterir.

Karıncıklar bir PVC oluşturduğunda, kulakçıklar depolarize olamayabilirler. Depolarizasyonun gerçekleşmemesi durumunda P dalgası oluşmaz. Kulakçık depolarizasyonu meydana gelince, karıncık depolarizasyonunun etkisiyle P dalgası QRS kompleksi içerisinde kaybolur.

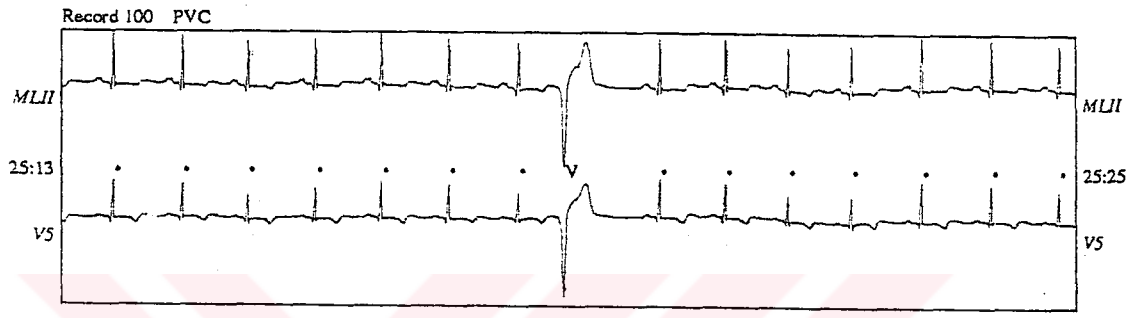
QRS kompleksi geniş ve anormal bir görünümdedir. QRS süresi 0.12 saniyeden uzundur ve ters yöne bir sapma gösterebilir.

PVC'yi hemen takip eden T dalgası, genellikle PVC'ye ait QRS kompleksine zıt yöndedir. Bu nedenle ST segmentinde anormallikler görülebilmektedir.

PVC'yi genellikle kompanze edici bir aralık takip eder. Bu aralık sayesinde esas ritim, PVC olmamış gibi kendi normal ritminde seyreder. PVC'den önceki ve sonraki komplekslerin R tepeleri arasındaki süre, esas ritmin R-R

aralığının en az iki katıdır. Bu durum, kompanze edici aralık sayesinde gerçekleşir.

Dakikada 10'un üzerinde PVC vurusu gelmesi hayati tehlike arz eder. Şekil 2.15'te PVC içeren bir EKG kaydı görülmektedir.



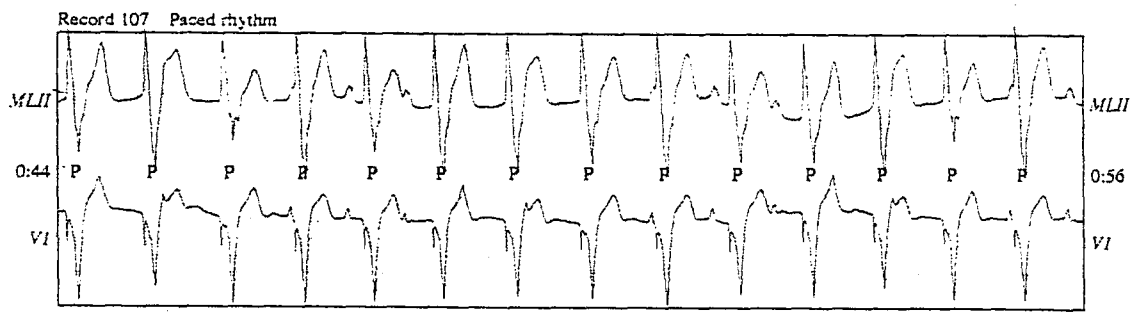
Şekil 2.15: Erken karıncık kasılması (PVC) içeren EKG kaydı.

2.6.4 Yapay vuru (Paced beat, P)

Kalp, kendi üretip oluşturduğu elektrik akımını yine kendi içindeki yollardan kasına dağıtarak, kasılmasını sağlar. Kalbin doğal ileti sisteminde belirli aksaklıklar olması durumunda, kalp kasının kasılması için muhtaç olduğu elektriği yapay olarak sağlayan "pacemaker" adı verilen sistemler kullanılmaktadır. Pacemaker, sağ karıncık, sağ kulakçık ve her ikisinden ardışıl olmak üzere yapay vuruyu kalbe vermektedir.

Yapay vuruda P dalgası nadiren görülmektedir. PR aralığı ölçülemez. QRS kompleksi, yapay vurudan sonra görülür. Genellikle 0.12 saniyeden uzundur. Eğer varsa, P-P aralığı değişik değerler alır. R-R aralığı, kalp vuru hızı, yapay vuru aralığı pacemaker ayarına göre farklılıklar

gösterir.



Şekil 2.16: Yapay vuru kaydı.

2.7 MIT-BIH Veri Tabanı (Massachusetts Institute of Technology-Beth Israel Hospital Data Base)

MIT-BIH veri tabanında bulunan EKG kayıtları, Beth-Israel Hospital Aritmi Labaratuarı tarafından 1975-1979 yılları arasında alınmış olan 4000'i aşkın uzun-dönem Holter kayıtlarından oluşturulmuş bir kümedir [1]. Bu veri tabanında 48 adet EKG kaydı bulunmaktadır. Bunların 23'ü, Holter kayıtları kümesinden rastgele seçilmiştir. 25 kayıt ise aynı kümeden seçilen (Holter kayıtlarından rastgele seçilmekle iyi bir şekilde temsil edilmesi mümkün olamayacak) nadir görülen fakat klinik açıdan önem arz eden kayıtlardır. Bu 48 kaydın herbiri 30 dakika uzunluktadır.

EKG kayıtları yaşları 32-89 arasında değişen 25 erkek ve 23-89 yaşları arasındaki 22 kadından alınmıştır.

Hastaların tümüne iki derivasyon uygulanmıştır. Birçok kayıtta bulunan ve bigisayarla görüntülenirken ekranın üstünde görülen işaret II nolu derivasyondur (MIT-BIH veri tabanında notasyon olarak MLII kullanılmıştır). Alttaki işaret ise çoğunlukla V_1, zaman zaman V_2 veya V_5, kayıtların birinde de V_4 derivasyonudur. Normal QRS kompleksleri genellikle üstteki işarette belirgindir.

İşaret 360 Hz ile örneklenmiş olup ±5 mV bölgesinde 11-bit rezolüsyona sahiptir. Örnek değerleri 0-2047 kapalı aralığında değerler alırlar ve 1024 değeri 0 V'a karşılık gelmektedir.

Beşinci bölümde gerçeklenen algoritmalarda, EKG kayıtları hakkında bilgi veren bir tablo kullanılır. Kayıt adreslerini ifade eden bu tablolar, MIT-BIH veritabanı ile birlikte satın alınmış bir komut tarafından oluşturulur. Aşağıda tabloları oluşturmakta kullanılan RDANN komutunun çalışması izah edilmiştir:

rdann -r kayıt -a uzantı [seçenek] >çıkış

Komut içerisinde kayıt, EKG giriş veri dosyasını; uzantı, incelenen kayıtla ilgili giriş bilgi dosyasını; çıkış ise sonucun yazıldığı dosya ismini ifade eder. Seçenek alanından 3 farklı imkan temin edilmektedir. -f seçeneği ile incelenen kayıt içerisinde zaman başlangıcı; -t seçeneği ile zaman olarak kayıt sonu; -p seçeneği ile bu zaman aralığı içerisinde araştırılacak sınıf kodu (incelenen şekil bozukluğu) belirtilir. Bu komut ile sonuçlar tablo şeklinde çıkış alanında saklanır. Şekil 2.17'de RDANN komutu ile oluşturulmuş çıkış dosyası görülmektedir.

```
0:00.130      47    +    0    0    0      (P
0:00.263      95    P    0    0    0
0:00.755     272    P    0    0    0
0:01.580     569    P    0    0    0
0:02.455     884    P    0    0    0
0:03.350    1206    P    0    0    0
0:04.219    1519    P    0    0    0
```

Şekil 2.17: RDANN komut çıktısı.

Şekil 2.17'deki ilk sütun, dakika:saniye şeklinde aranılan şekil bozukluğunun kayıt içerisindeki zaman olarak yerini; ikinci sütun, aranılan şekil bozukluğunun R tepesinin tüm kayıt içerisinde kaçıncı veri olduğunu; üçüncü sütun ise şekil bozukluğunun etiketini ifade eder. Çalışma içerisinde sadece 2 ve 3 sütunları kullanılmıştır.

Tüm kayıt içerisinden aranılan şekil bozukluğuna direkt olarak ulaşabilmek için tablonun ikinci sütunu kullanılır. Her bir kayıt içerisinde iki derivasyona ait EKG verisi bulunur. Bir derivasyon verisi için 12 bit kullanılmıştır. Kayıt içerisinde incelenen zamana ait 2 farklı derivasyonun genliği için 3 bayt (2*12 bit) kullanılmıştır. Aşağıda örnek olarak anlaşılması için veri içerisinde iki farklı derivasyonun genliklerinin nasıl ifade edildiği gösterilmiştir:

bayt	1	2	3	4	5	6
	3 E	4 F	7 A	8 2	2 4	3 5
zaman	t=0.000			t=0.003		t=0.006

1. derivasyon → F3E	1. derivasyon → 482
2. derivasyon → 47A	2. derivasyon → 235

Yukarıda görülen genlik değerleri, kayıt dosyası içerisinde "hexadecimal" olarak ifade edilmiştir. Aşağıda bu "hexadecimal" değerin ondalık olarak anlamları verilmiştir. Her bir ikili (bit), 0.005 mV'a karşılık gelmektedir.

(0H — 400H)	→	(-5.120 V — 0 V)
(401H — 800H)	→	(+0.005 V — +5.120 V)
(801H — FFFH)	→	(-15.360 V — -5.125 V)

Bu şekilde incelenen derivasyona ait veri kayıttan okunur ve QRS deteksiyon bloğuna giriş olarak verilir.

BÖLÜM 3

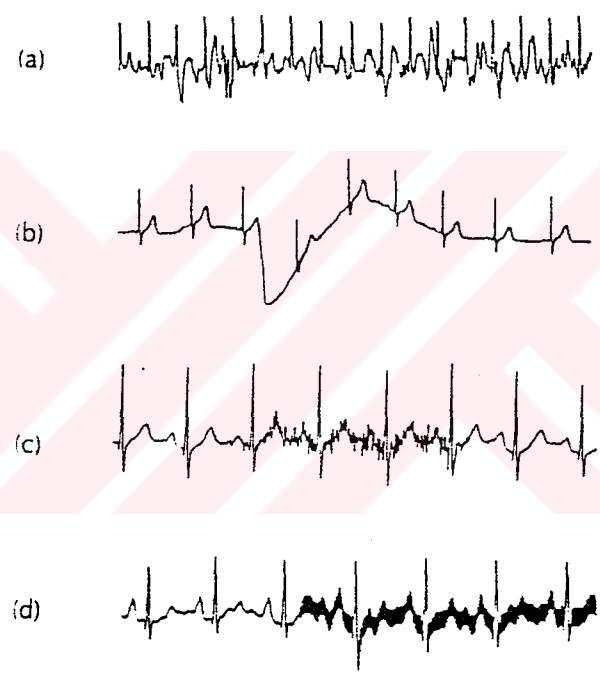
EKG İŞARETİ İÇERİSİNDEN R TEPESİNİN BULUNMASI

3.1 Giriş

QRS kompleksinin fark edilmesi, bu işaretin sadece fizyolojik değişiminden değil aynı zamanda gürültü tiplerinin değişik olmasından ötürü zordur. Bozucu etki olarak kas hareketinden, elektrod kaymasından, şebeke frekansı karışmasından ve taban hattı değişiminden kaynaklanan gürültü, örnek olarak verilebilir. Çalışma içerisinde kullanılan sayısal filtreler bu etkileri ortadan kaldırır. Tez içerisinde kullanılan QRS kompleksini farketme algoritması üç temel işlemden meydana gelir: Sayısal filtreler, lineer olmayan dönüşüm ve karar verme algoritması. Literatürde bu konu ile ilgili uygulamada sadece eğim bilgisi kullanılarak probleme cevap arandığı gözlenir. Ancak türev alma işlemi ile arzu edilmeyen yüksek frekanslı gürültünün genlik değeri de büyütülmüş olur. Yüksek genlikli, uzun süreli anormal QRS kompleksleri, eğimleri düşük olduğu için EKG içerisinde tespit edilemezler. Daha iyi performans elde edilmesi için tez içerisinde sadece eğim bilgisi değil; genlik, genişlik ve QRS enerjisi gibi farklı bilgiler de kullanılmıştır. Bu konunun üzerinde durulmasının sebebi, R tepesinin doğru olarak tespit edilmesinin son derece önemli olmasındandır. Özellik vektörleri R tepesi etrafında yapılan inceleme sonunda elde edilir. R tepesinin yeri doğru olarak tespit edilemezse yanlış özellik vektörleri çıkartılır. Bu durumda sınıflayıcılar hatalı karar verir.

3.2 EKG İşareti Üzerinde Elektriksel Gürültü

Teşhis için kullanılan EKG işaretlerinin gürültüden büyük ölçüde arıtılması gerekmektedir. EKG işaretinin büyüklüğü genelde 10 mV civarındadır. Bu büyüklükteki işaretler biyolojik ve harici işaretlerden kolayca etkilenirler. Şekil 3.1'de çeşitli gürültülerden etkilenmiş EKG kayıtları görülmektedir.



Şekil 3.1: Çeşitli gürültülerden etkilenmiş EKG kayıtları.

İnsan derisi, fizyolojik ve harici etkilerden dolayı elektriksel potansiyel kaynağıdır. Deride genelde hareketten dolayı DC 25 mV dolayında potansiyel oluşur. Bu potansiyel, yüksek geçiren bir filtre ile kolayca yok edilebilir. Eğer bu potansiyel, işaretin QRS bölümünde ortaya çıkarsa, QRS tespitini büyük ölçüde etkiler.

İnsan vücudu kas gürültüsü üretir. Bu gürültünün büyüklüğü EKG işareti civarındadır. Fakat frekansı daha

yüksektir. Kas gürültüsünün büyüklüğü çeşitli kasların hareketlerini azaltarak düşürülebilir.

Şehir şebeke geriliminin vücut yüzeyinde elektriksel indüklenmesi EKG işaretinde en önemli problemdir. Çevredeki elektrik hatları, elektrikli aletler, transformatörler, vb.'den ve manyetik yolla insan üzerinde gürültü oluşur.

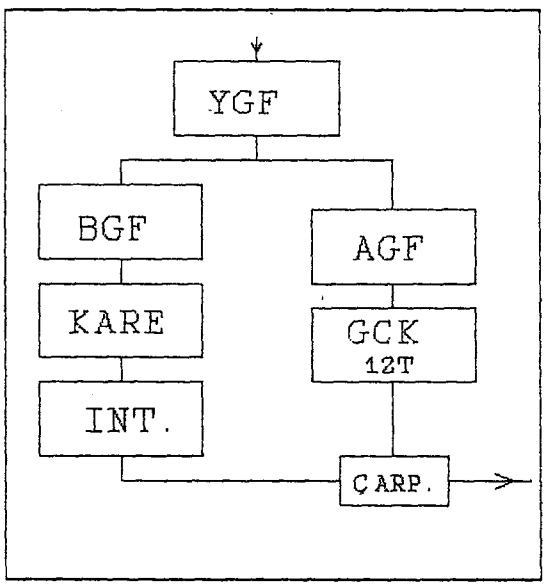
İnsan vücudu, elektrodlar ve EKG cihazı, bir döngü teşkil eder. Harici elektromanyetik alanlar bu döngü vasıtasıyla vücuda bir akım indükler. Bu akım EKG elektrodlarından ve yükselteçlerden geçerek EKG işaretinin üzerinde oluşur. Manyetik olarak indüklenen akım, döngü ile sınırlanan alan ile orantılıdır.

Diğer harici gürültü kaynağı, radyo frekansı (RF) yayıcılarıdır. Çeşitli elektriksel ve tıbbi cihazlar, örneğin elektrik motorları, elektrikli ameliyat cihazları dolaylı olarak girişim yayarlar. Bazı cihazlar vücut üzerinde birkaç yüz milivolt civarında 1 MHz'lik işaret oluşacak kadar işaret yayarlar.

3.3 QRS İşaretinin Tespit Edilmesi

Şekil 3.2'de QRS kompleksini tespit etmede kullanılan işlemler, bloklar şeklinde gösterilmiştir [2]. Bölüm 3.3'te bu bloklar içersinde kullanılan sayısal filtrelerin elde edilişi anlatılmaktadır.

QRS kompleksinin tespit edilmesinde taban-hattı değişimi önemli rol oynar. 2 Hz'e kilitli bir yüksek geçiren filtre (YGF) kullanılarak taban hattı değişimi EKG işareti üzerinden kaldırılır. Bu sayısal filtrenin çıkışı iki alt kola ayrılır.



Şekil 3.2: QRS detektörü blok diyagramı.

Şekil 3.2'deki 17 Hz'e kilitli band geçiren filtre (BGF), QRS işaretinin enerjisini artırmak için kullanılır. Band geçiren filtrenin çıkışı bir kare alma bloğundan geçirilir. Böylece incelenen işaretin gücü daha artırılır ve oluşan ikinci tepenin de pozitif olması sağlanır. Bu işlemin fark denklemi:

$$y(k) = x(k)^2 \qquad (3.1)$$

şeklindedir. Bu işlemden sonraki işlem hareketli pencere integrasyonudur. İntegral alıcı blok ile oluşan iki tepe birleştirilir ve pozitif tek bir işaret haline getirilir. Bu integrasyonun fark denklemi ise aşağıdaki gibidir:

$$y(k) = (1/N).[x(k-(N-1)) + x(k-(N-2)) + \ldots\ldots + x(k)] \qquad (3.2)$$

Bu denklemde N, integratörün genişliğinin içine sığan örnekleme sayısıdır.

Yüksek geçiren filtrenin diğer çıkışı ise 36 Hz'e kilitli alçak geçiren filtreye (AGF) verilir. Böylece

işaret üzerindeki yüksek frekanslı gürültü ortadan kaldırılır. Bu bloğun çıkışına 12T'lik bir gecikme yerleştirilir. Geciktirme işleminin amacı, filtre ve diğer işlemlerde ortaya çıkan gecikmeleri kompanze etmektir. Program için gereken fark denklemi:

$$y(n) = x(n-12) \tag{3.3}$$

şeklindedir. Daha sonra gecikme bloğunun çıkışı ile QRS işaretinin enerjisini yükselten bloğun çıkışı, bir çarpıcı devresi ile çarpılır. Böylece EKG işareti içerisinde gürültü olsa bile QRS'nin yeri yukarıda anlatılan işlemlerle tespit edilir.

3.4 Sayısal Filtreler

EKG içerisinden gürültüyü temizleyerek QRS işaretini tespit etmek için AGF, YGF ve BGF sayısal filtreleri tasarlanmıştır. Alt bölümler içerisinde bu filtrelerin elde edilişi anlatılmıştır.

3.4.1 Yüksek ve Band Geçiren Filtreler

Bu tez çalışmasında sayısal filtreleri tasarlamak için normalize analog filtrelerden yola çıkılmıştır. Normal analog filtre, kesim frekansı 1 Hz olan alçak geçiren bir filtredir. İkinci adım, bu normalize filtreden istenilen filtrenin elde edilmesidir. Bu işlem standard bazı transfer fonksiyonlar ile gerçeklenir. Aşağıda YGF'ye geçişle ilgili transfer fonksiyonu verilmiştir:

$$s \rightarrow \frac{w}{s} \tag{3.4}$$

w= Alçak geçiren filtrenin analog kesim frekansı.

Üçüncü adım, analog filtrelerden sayısal filtrelere geçiştir. Tez çalışmasında, geçiş yöntemi olarak bilineer transfer kullanılmıştır. Bu dönüşümün en önemli özelliği basitliği, beğenilmeyen tarafı ise kullanılan bağıntının lineer olmayışıdır. Sayısal filtreleri hesaplamak için kullanılan bilineer transfer bağıntısı aşağıda gösterilmiştir:

$$s \rightarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \qquad (3.5)$$

Aşağıda sayısal filtreler ile analog filtrelerin frekansları arasındaki bağıntının nasıl elde edildiği anlatılmıştır.

$$s \rightarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

$$z \rightarrow \frac{\frac{2}{T}+s}{\frac{2}{T}-s}$$

$$z = e^{jW_dT}$$

$$s = jW_A \quad \Rightarrow \quad jW_A = \frac{2}{T}\frac{1-e^{-jW_dT}}{1+e^{-jW_dT}}$$

$$jW_A = \frac{2}{T}\frac{e^{jW_d\frac{T}{2}}-e^{-jW_d\frac{T}{2}}}{e^{jW_d\frac{T}{2}}+e^{-jW_d\frac{T}{2}}}$$

$$jW_A = \frac{2}{T}j\tan\left(\frac{W_dT}{2}\right) \qquad (3.6)$$

W_d= Sayısal filtrenin açısal frekansı
W_A= Analog filtrenin açısal frekansı

Bu alt bölüm içinde önce yüksek geçiren filtre daha sonra band geçiren filtrenin tasarımı anlatılmıştır:

$$W_A = \frac{2}{T}\tan\left(\frac{W_d T}{2}\right) \qquad T = \frac{1}{360}$$

$$= 2 \cdot 360 \cdot \tan\left(\frac{2\pi \cdot 2}{2 \cdot 360}\right)$$

$$W_A = 12.256$$

Alçak geçiren filtrenin (2. dereceden Butterworth tipi) transfer fonksiyonu :

$$T_A(s) = \frac{1}{s^2 + 1.4s + 1} \tag{3.7}$$

şeklindedir. Yüksek geçiren filtrenin transfer fonksiyonu:

$$T_Y(s) = \frac{1}{\left(\frac{W_A}{s}\right)^2 + 1.4\frac{W_A}{s} + 1} \tag{3.8}$$

$$T_Y(s) = \frac{s^2}{157.929 + 17.593s + s^2}$$

şeklindedir. Yukarıdaki ifadeye bilineer dönüşüm uygulanarak s domeninden z domenine geçilir.

$$T(z) = \frac{720^2 \frac{(1-z^{-1})^2}{(1+z^{-1})^2}}{157.929 + 17.593\left(720\frac{1-z^{-1}}{1+z^{-1}}\right) + 720^2\frac{(1-z^{-1})^2}{(1+z^{-1})^2}}$$

$$T(z) = \frac{720^2 (1-z^{-1})^2}{157.929 (1+z^{-1})^2 + 17.593 \cdot 720 (1-z^{-2}) + 720^2 (1-z^{-1})^2}$$

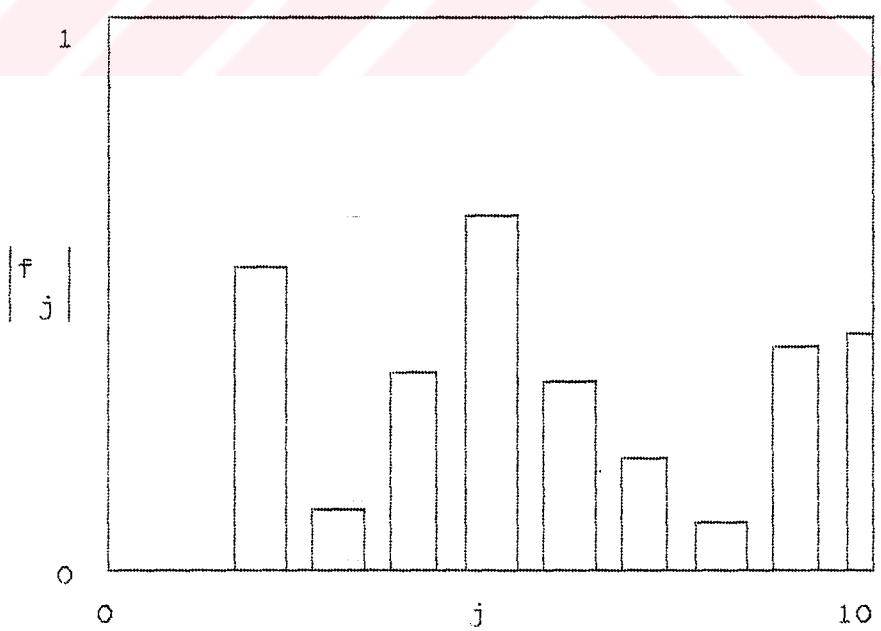
$$T(z)=\frac{720^2(1-2z^{-1}+z^{-2})}{157.929(1+2z^{-1}+z^{-2})+17.593\cdot 720(1-z^{-2})+720^2(1-2z^{-1}+z^{-2})}$$

$$\frac{Y(z)}{U(z)} = \frac{0.976-1.952z^{-1}+0.976z^{-2}}{1-1.951z^{-1}+0.952z^{-2}}$$

$$Y(z) = 1.951z^{-1}Y(z)-0.952z^{-2}Y(z)+0.976U(z)$$
$$-1.952z^{-1}U(z)+0.976z^{-2}U(z)$$

$$y(k) = 1.951y(k-1)-0.952y(k-2)+0.976u(k)$$
$$-1.952u(k-1)+0.976u(k-2) \qquad (3.9)$$

Şekilde yüksek geçiren filtrenin frekans cevabı verilmiştir.



Şekil 3.4: Yüksek geçiren filtrenin frekans cevabı.

Aşağıda 17 Hz band geçiren filtrenin elde edilişi verilmiştir:

$$W_{BÜ} = \frac{2}{T}\tan\left(\frac{\Omega_{BÜ}\cdot T}{2}\right)$$

$$W_{BA} = \frac{2}{T}\tan\left(\frac{\Omega_{BA}\cdot T}{2}\right) \qquad (3.10)$$

$\Omega_{BÜ}$= Band geçiren filtrenin sayısal üst kesim frekansı
$W_{BÜ}$= Band geçiren filtrenin analog üst kesim frekansı
Ω_{BA}= Band geçiren filtrenin sayısal alt kesim frekansı
W_{BA}= Band geçiren filtrenin analog alt kesim frekansı

$f_{BÜ}$= 19 Hz, f_{BA}= 15 Hz seçilirse analog karşılıkları aşağıdaki şekilde elde edilir.

$$W_{BA} = 720\cdot\tan\left(\frac{2\pi\cdot 15}{2\cdot 360}\right) = 94.789$$

$$W_{BÜ} = 720\cdot\tan\left(\frac{2\pi\cdot 19}{2\cdot 360}\right) = 120.486$$

Birinci dereceden Butterworth tipi AGF transfer fonksiyonu:

$$T_A(s) = \frac{1}{s+1} \qquad (3.11)$$

şeklindedir. Aşağıda band geçiren filtreye geçişle ilgili transfer fonksiyonu verilmiştir:

$$s \rightarrow \frac{s^2 + W_{BA}\cdot W_{BÜ}}{s\,(W_{BÜ} - W_{BA})} \qquad (3.12)$$

Band geçiren filtrenin transfer fonksiyonu:

$$T_B(s) = \frac{1}{\frac{s^2 + W_{BA} \cdot W_{BÜ}}{s\,(W_{BÜ} - W_{BA})} + 1} = \frac{s\,(W_{BÜ} - W_{BA})}{s^2 + s\,(W_{BÜ} - W_{BA}) + W_{BÜ} \cdot W_{BA}}$$

$$T_B(s) = \frac{s \cdot 25.697}{s^2 + 25.697\,s + 11420 \cdot 747} \qquad (3.13)$$

şeklindedir. Bu ifadeye bilineer dönüşüm uygulanarak s domeninden z domenine geçilir.

$$s \rightarrow 720 \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

$$T_B(z) = \frac{25.697 \cdot 720 \cdot \frac{1 - z^{-1}}{1 + z^{-1}}}{720^2 \cdot \frac{(1 - z^{-1})^2}{(1 + z^{-1})^2} + 25.697 \cdot 720 \cdot \left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) + 11420 \cdot 747}$$

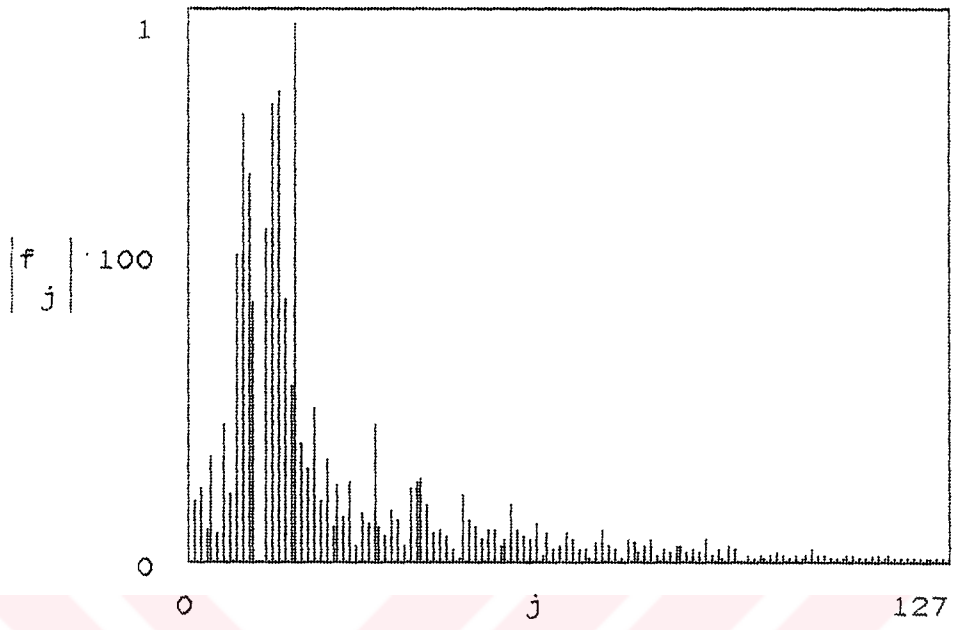
$$T_B(z) = \frac{Y(z)}{U(z)} = \frac{0.0337 - 0.0337\,z^{-2}}{1 - 1.849\,z^{-1} + 0.933\,z^{-2}}$$

$$Y(z) = 1.849\,z^{-1}Y(z) - 0.933\,z^{-2}Y(z) + 0.0337\,U(z) - 0.0337\,z^{-2}U(z)$$

$$y(k) = 1.849\,y(k-1) - 0.933\,y(k-2) + 0.0337\,u(k) - 0.0337\,u(k-2)$$

(3.14)

Şekilde band geçiren filtrenin frekans cevabı verilmiştir.



Şekil 3.5: Band geçiren filtrenin frekans cevabı.

3.4.2 Alçak Geçiren Filtre

Aşağıdaki sayısal alçak geçiren filtre direkt olarak z domeninde tasarlanmıştır:

$$T_A(z) = \frac{(1 - z^{-k})^1}{(1 - z^{-1})^1} \qquad (3.15)$$

$$k = \frac{\text{Örnekleme Frekansı}}{\text{Kesim Frekansı}}$$

Alçak geçiren filtrenin frekansını 36 Hz olarak seçersek, örnekleme frekansı 360 Hz olduğu için k, 10 olarak bulunur. Çalışma içerisinde 1, 2 olarak seçilmiştir.

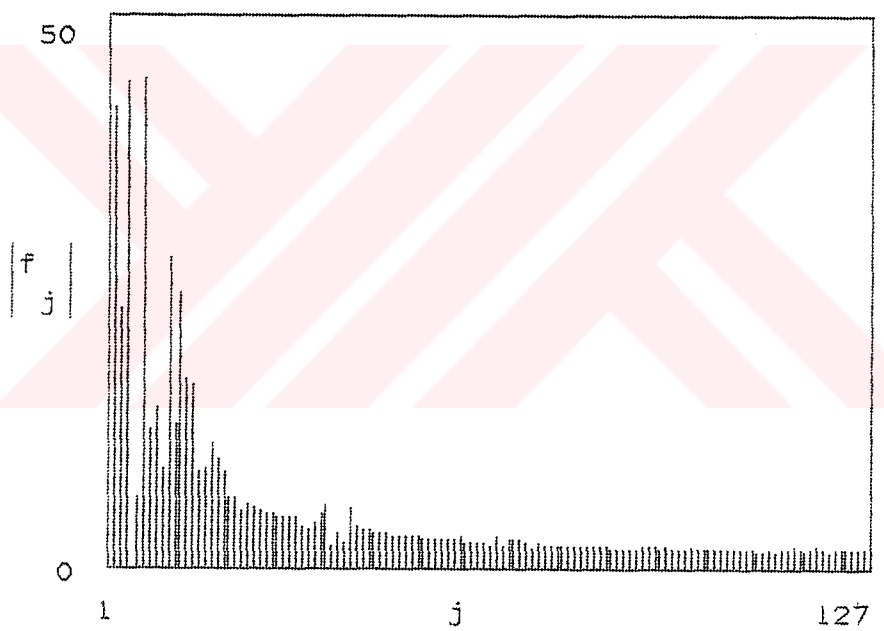
$$T_A(z) = \frac{(1 - z^{-10})^2}{(1 - z^{-1})^2} = \frac{1 - 2z^{-10} + z^{-20}}{1 - 2z^{-1} + z^{-2}}$$

$$\frac{Y(z)}{U(z)} = \frac{1 - 2z^{-10} + z^{-20}}{1 - 2z^{-1} + z^{-2}}$$

$$Y(z) = 2z^{-1}Y(z) - z^{-2}Y(z) + U(z) - 2z^{-10}U(z) + z^{-20}U(z)$$

$$y(k) = 2y(k-1) - y(k-2) + u(k) - 2u(k-10) + u(k-20) \qquad (3.16)$$

Şekilde alçak geçiren filtrenin frekans cevabı gösterilmiştir.



Şekil 3.6: Alçak geçiren filtrenin frekans cevabı.

BÖLÜM 4

EKG İŞARETLERİNİ SINIFLAMADA KULLANILAN ÖZELLİK VEKTÖRLERİ

4.1 Giriş

EKG işaretlerinin şekli, temel olarak bilinmesine karşın, alındığı kişiye, alınma şekline ve diğer bozucu etkilere çok bağlıdır. Özellik vektörlerinin EKG işareti içerisinden R tepesinin yerinin doğru olarak bulunmasına az ya da hiç bağlı olmaması istenir. Aksi taktirde, öğrenme işlemi tamamlandıktan sonra bulunan öz vektörlere hiç benzemeyen bir giriş vektörü geldiğinde sınıflayıcılar hata yapmaya başlar. Sınıflayıcıların kullandığı özellik vektörlerinin basit olması, dolayısıyla hızlı bir şekilde sınıflama yapabilmesi istenir. Aynı zamanda bir sınıfı diğerinden ayıran özellik vektörlerinde kullanılan parametre sayısı dolayısıyla özellik vektörlerinin boyutunun küçük olması istenir. Özellik vektörünün boyutu büyük olduğu taktirde, elde edilen öz vektörlerin boyutu da büyük olacak ve bu vektörlerin saklanması için daha fazla bellek elemanı kullanılacaktır.

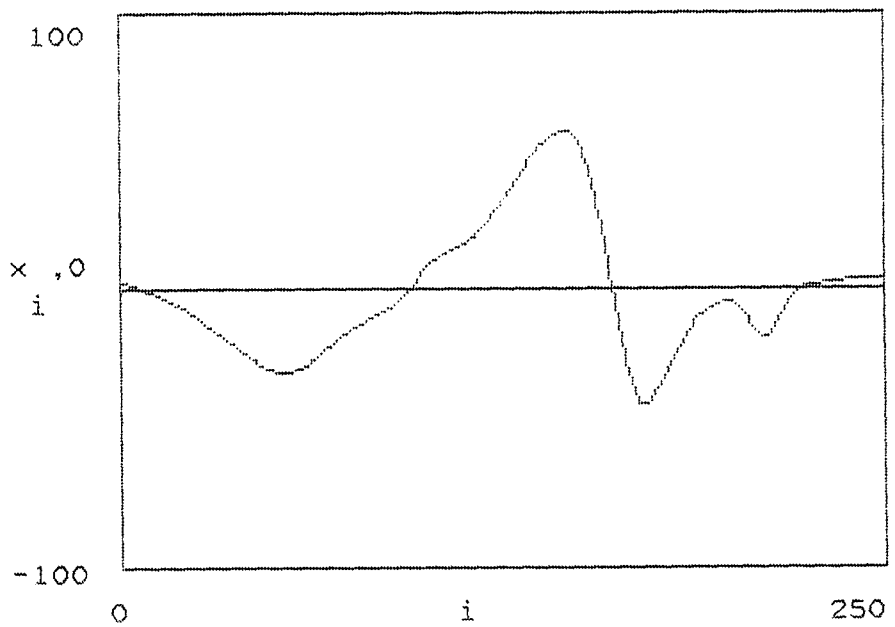
Sonuç olarak gerçeklenen sınıflayıcıdan az bellek elemanı kullanması, hızlı sınıflama yapabilmesi ve genelleme özelliğinin iyi olması beklenir. Aşağıdaki bölümlerde literatürde EKG işaretlerini sınıflama amacı ile kullanılan 4 farklı özellik elde etme yöntemi incelenmiştir: QRS şekil bilgisi, frekans spektrumu, lineer tahmin katsayıları ve direkt yöntem.

4.2 QRS Kompleksinden Çıkarılan Bilgilerle Özellik Vektörünün Oluşturulması

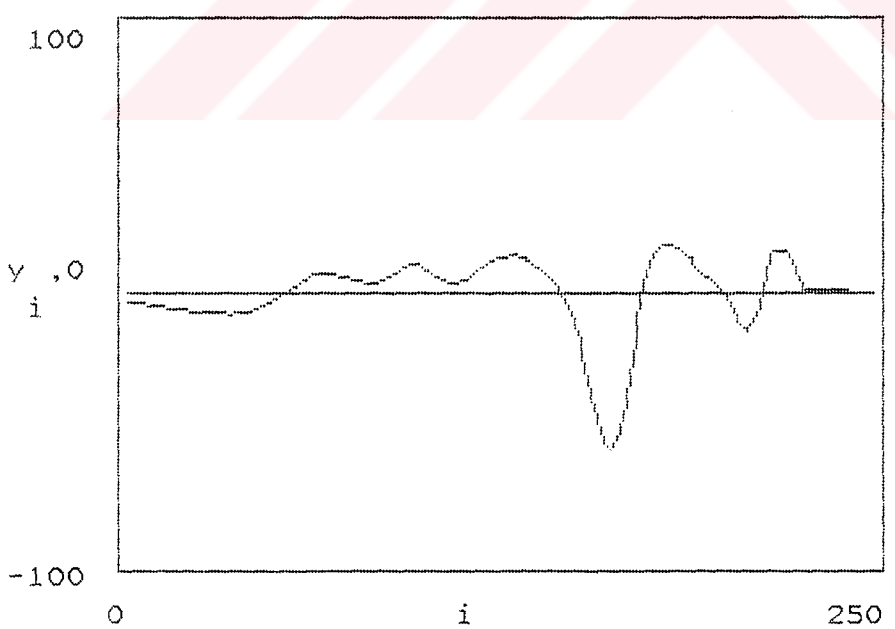
Özellik vektörünün parametrelerini elde etmek için Q ve S noktalarının EKG işareti içerisindeki yerinin ve taban-hattı değerinin bilinmesi gerekir. Q ve S noktalarının bulunması için QRS kompleksinin eğim bilgisi kullanılır. Eğim işlemi EKG işareti üzerindeki gürültüden etkilendiği için işaret önce bir alçak geçiren filtreden (AGF) geçirilir. Bu işlem için ayrıca bir alçak geçiren filtre tasarlanmaz. R noktasının yerinin bulunmasında tasarlanmış 36 Hz'e kilitli alçak geçiren filtre çıkışı (Şekil 3.3) kullanılır. Eğim işlemi için aşağıdaki ardışıl denklem kullanılmaktadır:

$$\text{Eğim} = y(k) = (1/8T).[-x(k-2) - 2x(k-1) + 2x(k+1) + x(k+2)] \quad (4.1)$$

Denklemdeki x(k) değişkeni, 36 Hz'e kilitli AGF'nin çıkışı olarak alınmıştır. Şekil 4.1.a'da orijinal EKG işareti, 4.1.b'de ise bu işaretin eğim alındıktan sonraki durumu gösterilmiştir. Orijinal EKG işareti üzerinde R tepesinden sola doğru eğimi incelenir ve ilk karşılaşılan sıfır değeri EKG içerisinde Q noktasının yeri olarak kabul edilir. Aynı şekilde EKG işareti üzerinde R tepesinden sağa doğru eğim bilgileri incelenir. İlk karşılaşılan sıfır değeri, EKG içerisinde S noktasının yeri olarak kabul edilir. Taban-hattının bulunması için EKG işareti üzerinde R tepesinden sağa doğru eğimin sıfır olduğu ikinci noktanın EKG üzerindeki genlik bilgisi kullanılır. İlk 5 periyot için elde edilen genlik değerlerinin ortalaması alınarak taban-hattının değeri elde edilir. Q ve S noktalarının yerinin ve taban-hattının değerinin bulunmasında oldukça basit işlemler kullanılmıştır. Dolayısıyla özellik vektörlerinin elde edilmesi işlemi için çok fazla süre geçmez. Ancak işaret üzerindeki gürültü, alınış şekli ve alındığı kişiye çok bağlı oluşu bu parametrelerin geniş bir aralıkta değişmesine sebep olur.



Şekil 4.1.a: Orjinal EKG işareti



Şekil 4.1.b: EKG işaretinin Eğimi

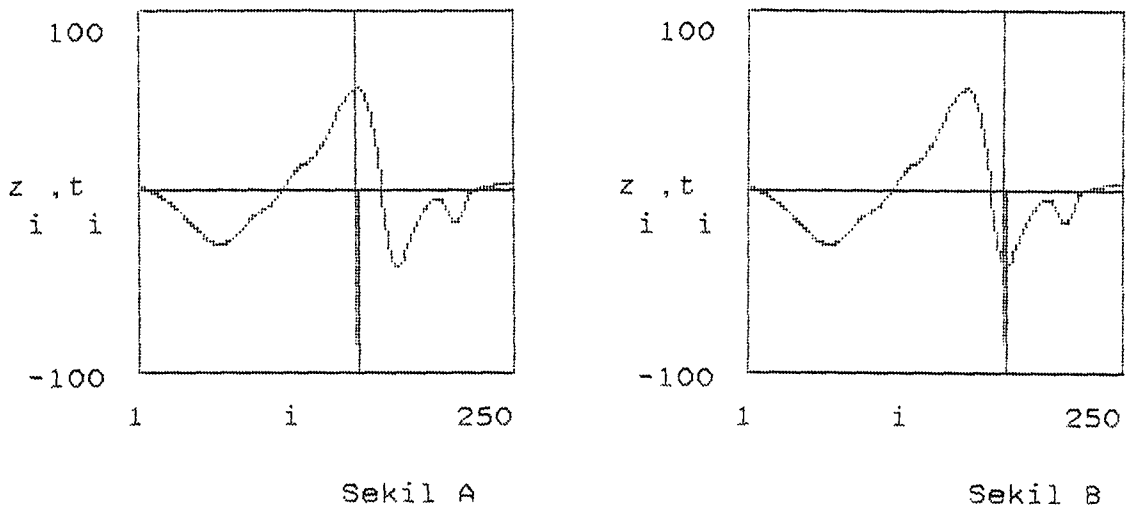
Q, S ve taban-hattı bilgileri kullanılarak elde edilen özellik vektörünün 4 elemanı aşağıda tanımlanmıştır:

a) QS süresi
b) Yükseklik = | max(QR,RS) |
c) QS zaman aralığı içinde taban-hattı üzerindeki alan
d) | (Taban-hattı) - (Yükseklik/2) | [3]

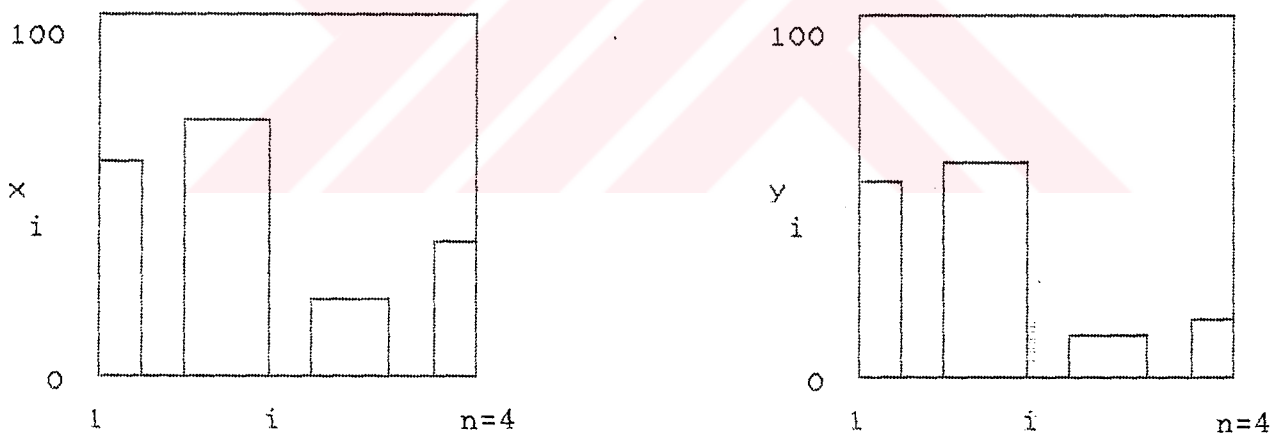
Yukarıda tanımlı tüm parametreler R tepesinin QRS kompleksi içindeki yerine son derece bağlıdır. Eğim bilgilerinden hareketle Q ve S noktalarının yerleri her zaman doğru olarak tespit edilemez. R tepesinin yerinin doğru olarak tespit edilememesi durumunu incelemek için şekil 4.2'de EKG işaretine benzeyen bir örnek işaret gösterilmiştir. Şekil 4.3'te ise R tepesinin yerinin doğru ve yanlış tespit edilmesi durumunda oluşan iki farklı özellik vektörü gösterilmiştir.

4.3 EKG İşaretinin Frekans Spektrumu Kullanılarak Oluşturulan Özellik Vektörü

QRS kompleksi içerisinde R tepesinin yeri tespit edildikten sonra R tepesi etrafında sola doğru 160, sağa doğru 96 örnek alınarak bir pencere oluşturulur. Böylece pencere içerisinde sadece 1 QRS kompleksi olması temin edilmiştir. Bu pencere içerisindeki işaretten 0-30 Hz aralığındaki frekans spektrumunun sadece genlik bilgileri temin edilir, faz bilgileri kullanılmaz. Elde edilen özellik vektörünün boyutu 30'dur. İşaret üzerindeki 50 Hz şebeke gürültüsü spektrumda gözükmez. Gürültü genellikle yüksek frekanslı bileşenlere sahip olduğu için özellik vektörleri bu iki tip gürültüden etkilenmez. Frekans genlik bilgileri, R tepesinin yerinin doğru tespitinden etkilenmez. R tepesinin yerinin yanlış belirlenmesi bir gecikme olarak kabul edilirse aşağıdaki sonucu elde ederiz.



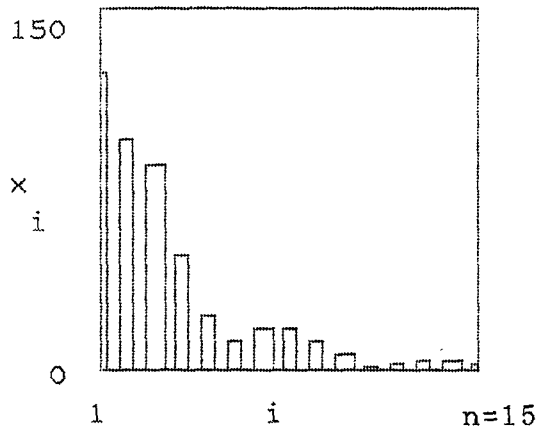
Şekil 4.2: R tepesinin belirlenmesi

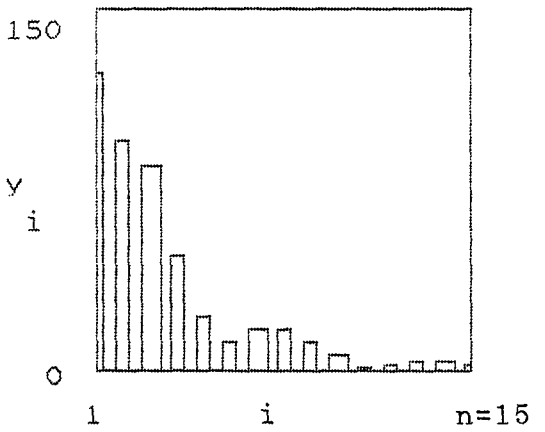


x i
59
71
21.408
37.362

y i
54
59
11.567
16.175

Şekil 4.3: QRS şekil bilgisi için özellik vektörleri





x i
122.935
95.975
85.17
48.037
22.574
12.052
16.968
17.179
12.329
6.426
0.729
2.774
3.663
3.727
2.01

Şekil 4.4.a

y i
122.935
95.975
85.17
48.037
22.574
12.052
16.968
17.179
12.329
6.426
0.729
2.774
3.663
3.727
2.01

Şekil 4.4.b

4.4 QRS İşaretinin Lineer Tahmin Katsayıları (LPC) Kullanılarak Özellik Vektörünün Oluşturulması

Yöntemin incelenmesinin sebebi, bulunan katsayıların, giriş işareti üzerindeki gürültüden etkilenmemesi ve daha az parametre ile işaretin temsil edilebilmesidir. R tepesinin yerinin yanlış seçilmesi EKG işaretinin gecikmesi olarak yorumlanmıştır. Bölüm içinde elde edilen katsayılar gecikmeden etkilenmez. Bu yöntem ile hem gecikmeden hem de gürültüden etkilenmeyen, aynı zamanda frekans spektrumundan elde edilen katsayılardan daha az sayıda katsayı kullanılarak özellik vektörü oluşturulur.

Aşağıda lineer tahmin katsayılarının (LPC) elde edilişi ile ilgili denklemler verilmiştir [15]:

$$e^{(0)}(k) = y(k)$$

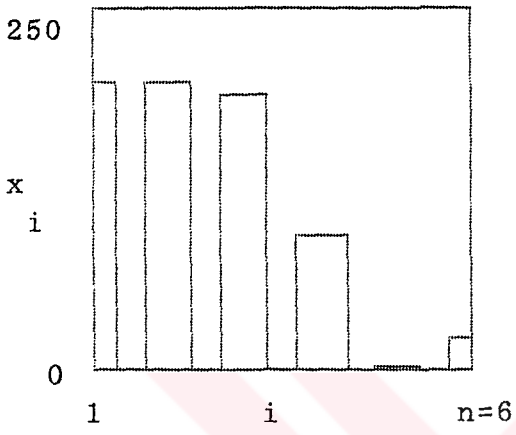
$$\tilde{e}^{(0)}(k) = y(k) \qquad (4.4)$$

$$e^{(N+1)}(k) = e^{(N)}(k) - K^{(N+1)}\,\tilde{e}^{(N)}(k-1)$$

$$\tilde{e}^{(N+1)}(k) = \tilde{e}^{(N)}(k-1) - K^{(N+1)}\,\tilde{e}^{(N)}(k)$$

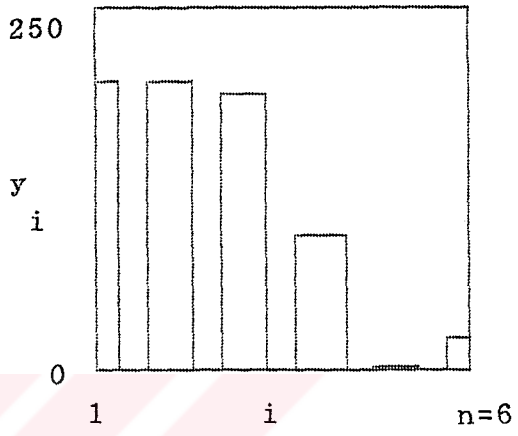
$$K^{(N+1)} = \frac{\frac{1}{N}\sum_{k=1}^{N} e^{(N)}(k)\cdot\tilde{e}^{(N)}(k-1)}{\frac{1}{N}\sum_{k=1}^{N}\left(\tilde{e}^{(N)}(k-1)\right)^2}$$

6 lineer tahmini kaysayısı kullanılarak öz vektörler oluşturulur. Şekil 4.5.a'da R tepesinin doğru olarak belirlenmesi durumunda LPC katsayıları grafik olarak gösterilmiştir. Şekil 4.5.b'de ise R tepesinin doğru olarak belirlenememesi durumunda LPC katsayıları grafik olarak gösterilmiştir.



x i
199.443
198.583
189.982
93.99
2.801
21.525

Şekil 4.5.a

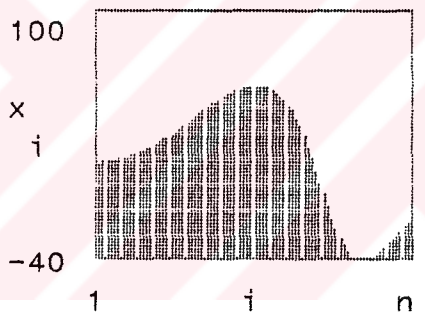


y i
199.443
198.583
189.982
93.99
2.801
21.525

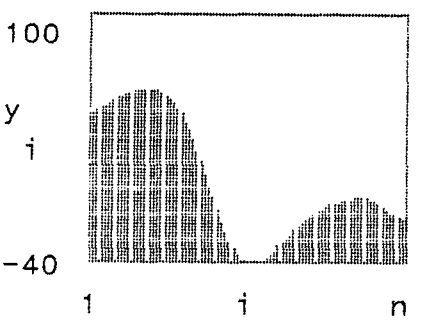
Şekil 4.5.b

4.5 EKG İşareti Direkt Alınarak Özellik Vektörünün Oluşturulması

R tepesi etrafında sola doğru 24, sağa doğru 24 örnek alınarak sadece 1 QRS kompleksi içeren bir pencere oluşturulur. Böylece 48 örnekten meydana gelen bir özellik vektörü elde edilir. Vektörün elemanları R tepesinin yerinin doğru olarak seçilmesine çok bağlıdır; ancak özellik vektörünün oluşumunda bir ön işlem kullanılmadığı için sınıflama işlemi oldukça kısa sürede gerçeklenir. Şekil 4.5.a'da R tepesinin doğru olarak belirlenmesi durumunda elde edilen özellik vektörü; şekil 4.5.b'de R tepesinin doğru olarak belirlenmemesi durumunda elde edilen özellik vektörü gösterilmiştir.



Şekil 4.5.a: R tepesinin yerinin doğru belirlenmesi durumunda elde edilen özellik vektörü.



Şekil 4.5.b: R tepesinin yerinin yanlış belirlenmesi durumunda elde edilen özellik vektörü.

BÖLÜM 5

BULANIK (FUZZY) SINIFLAYICILAR

5.1 Giriş

Yapılan tez çalışmasında, EKG işareti içerisindeki şekil bozukluklarını tespit etmek üzere "bulanık sınıflayıcılar" kullanılmıştır. Bulanık bir sınıflayıcı incelenen her bir EKG'nin bütün sınıflara olabilecek üyeliğinin tespitine çalışır. Klasik sınıflayıcılar ise incelenen EKG'nin ait olduğu tek bir sınıfı belirleme yoluna gider. Buna karşın, bulanık sınıflayıcı, üyelik dereceleriyle tutulan bilgiyi, şekil bozukluklarından giderek yapacağı hastalık teşhisleri için kardiyolojistin kararına bırakır.

Bölüm içerisinde, öncelikle bulanık küme teorisi incelenecek, duru küme teorisiyle arasındaki ilişkiler irdelenecektir. İlerki kısımlarda, sınıflayıcılara giriş olarak verilecek öz-vektörleri elde etmek üzere kullanılan iki farklı öğrenme algoritması izah edilmiştir. Bu algoritmalar: Bulanık küme-ortalar ve bulanık Kohonen kümeleme algoritmalarıdır. Son bölümde ise bulanık en yakın model (fuzzy nearest prototype) ve bulanık K-en yakın komşu (fuzzy k nearest neighbour) sınıflayıcıları anlatılmıştır.

5.2 Bulanık Küme Teorisi (Fuzzy Set Theory)

Duru küme teorisinin bir genellemesi olan bulanık kümeler, günlük hayatımızdaki belirsizliği ifade etmede kullanılan yeni bir yol olarak karşımıza çıkarlar. Çeşitli

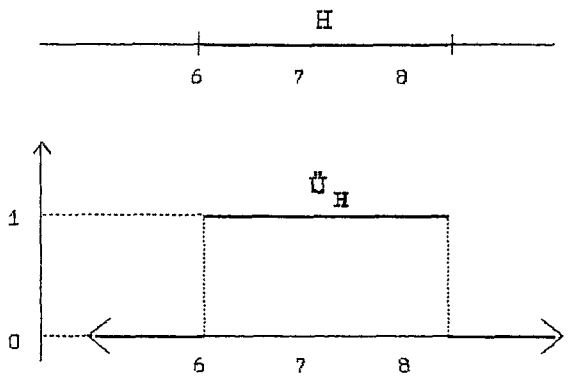
örüntü tanıma problemlerinin formulasyonu ve çözümünde, veri yapılarının bulanık ifadeleri kullanılmaktadır. Duru kümeler, üyelik için, kesin özellikler gösteren elemanlardan oluşurlar. Duru kümelerin tanımına göre, söz konusu evrenin bireyleri iki grupta toplanır: Üyeler (o kümeye kesinlikle ait olanlar) ve üye olmayanlar (o kümeye kesinlikle ait olmayanlar). Üyeler ve üye olmayanlar arasında keskin ve belirli bir ayırım mevcuttur.

Duru kümelerin karakteristik fonksiyonları, evrensel kümenin elemanlarına 0 veya 1 değerlerini karşı düşürmek suretiyle, bu elemanların söz konusu duru kümeye ait olup olmadıklarını belirler. Bu fonksiyonları genelleştirip, elemanlara atanan değerlerin belli bir aralıkta kalması ve bu değerlerin, elemanın söz konusu kümeye aitlik derecesini belirtmesi sağlanabilir. 6'dan 8'e kadar olan sayıların kümesi H, duru bir kümedir ve $H=\{ r \in R \mid 6 \leq r \leq 8 \}$ şeklinde yazılır. Buna eşdeğer olarak H, üyelik fonksiyonu $ü_H : R \rightarrow \{0,1\}$ ile şu şekilde ifade edilir:

$$ü_H(r) = \begin{cases} 1; & 6 \leq r \leq 8 \\ 0; & \text{aksi taktirde} \end{cases}$$

H duru kümesi ve $ü_H$'nın grafiği şekil 5.1'de görülmektedir.

Her r, reel sayısı H içindedir veya değildir. $ü_H$, tüm $r \in R$ reel sayılarını iki noktaya {0,1} karşı düşürdüğünden, duru kümeler; iki-değerli mantık, var veya yok, açık veya kapalı, siyah veya beyaz, 1 veya 0'a karşılık gelirler. Mantıkta, $ü_H$ üyelik fonksiyonunun aldığı değerler; "r, H içinde mi?" sorusuna cevap veren "doğruluk değerleri" dir. Cevap, sadece ve sadece $ü_H(r)=1$ iken "evet" tir, aksi taktirde, "hayır" cevabı verilir.



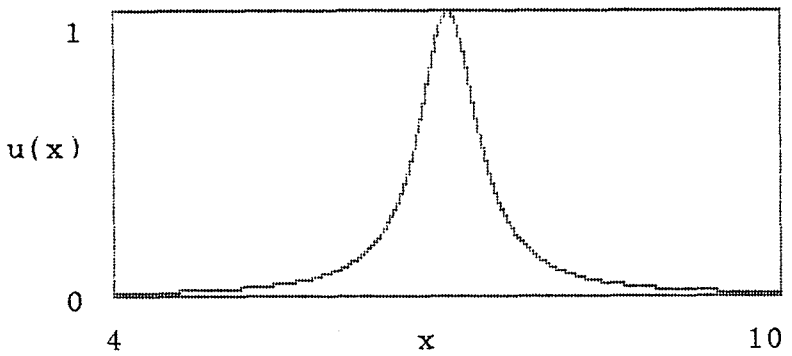
Şekil 5.1 Duru H kümesine ait üyelik fonksiyonu

Bir bulanık kümeyi matematiksel olarak ifade etmek için, sözkonusu evrende yer alan tüm bireylere, onun bulanık kümeye ait olma derecesini belirten bir değer atanır. Bu derece, o bireyin bulanık küme tarafından temsil edilen kavrama benzerlik veya uyumluluk derecesini göstermektedir. Bu, kümeye ait olma dereceleri, genellikle [0,1] kapalı aralığında yer alan bir reel sayıyla ifade edilir. Tümüyle ait olma durumu 1 ve tümüyle ait olmama durumu ise 0 değeri ile gösterileceğinden duru kümeleri, daha genel olan bulanık kümelerin bir özel durumu olarak görebiliriz. Bulanık mantıkta, doğruluk da dahil olmak üzere herşeyin bir derecesi vardır.

F kümesi, 7'ye yakın reel sayılar kümesi olsun. "7'ye yakın" ifadesi bulanık olduğundan F için tek bir $ü_F$ üyelik fonksiyonu yoktur ($ü_F$: X → [0,1]). $ü_F$'nin nasıl olacağı, F için arzulanan özelliklere ve uygulanacağı alana bağlıdır. F için uygun gözüken özellikler: (i) normallik ($ü_F(7)=1$); (ii) monotonluk (x, 7'ye yaklaştıkça $ü_F(x)$ 1'e yaklaşır ve tersi); (iii) simetri (7'nin sağında ve solunda, 7'ye eşit uzaklıktaki sayılar eşit üyeliğe sahiptir). Bu sezgisel yaklaşımla, 7'ye yakın olan reel sayılar bulanık kümesinin üyelik fonksiyonu şu şekilde tanımlayalım:

$$\ddot{u}_F(x) = \frac{1}{1 + 10\,(x-7)^2}$$

Bu fonksiyonun grafiği aşağıdaki şekildedir:



Şekil 5.2: 7'ye yakın reel sayıların oluşturduğu bulanık kümenin üyelik fonksiyonu.

$\ddot{u}_F(6.8) = 0.714$ bulunur. Bulunan 0.714 değeri 6.8'in 7'ye oldukça yakın bir değer olduğu anlamındadır.

Örneğin 7'ye "çok" yakın reel sayılar bulanık kümesinin üyelik fonksiyonu:

$$\ddot{u}_F(x) = \frac{1}{(1 + 10\,(x-7)^2)^2}$$

şeklinde alınabilir.

5.3 Öz-vektörlerin Öğrenilmesi

Küme analizi, bir veri noktaları grubunun çeşitli sayıda alt gruplara bölmelenmesine dayanır. Bu alt gruplardaki (kümelerdeki) elemanlar birbirleriyle belirli bir derecede yakınlık veya benzerlik göstermelidirler. Bildiğimiz klasik (hard) kümelemede, her veri noktası

(özellik vektörü), kümeler arasındaki sınırların iyi tanımlanmış olduğu varsayımı altında, sadece ve sadece bir kümeye 1 üyelik derecesiyle atanır. Bu model, alt gruplar arasındaki sınırların bulanık olduğu ve elemanın belirli bir kümeye yakınlığının daha hassas bir şekilde istendiği durumlardaki gerçek veri tanımını yansıtamaz. Bu nedenle günlük hayatta karşılaşılan çeşitli problemlere, bulanık bir ortamda yapılan kararlarla çözümler aranmaktadır.

Verinin alt gruplara "optimal bölmelenmesi" tanımı için kriter aşağıdaki üç şartın sağlanmasına dayanır:

1) Oluşan kümeler arasındaki ayırımın açık olması,

2) Kümelerin minimal hacime sahip olması,

3) Küme merkezi civarında maksimal sayıda veri noktasının toplanmış olması.

Böylelikle, ortam bulanık olsa bile, sınıflamadaki amaç, iyi tanımlı alt grupların üretimi ve böylece veri kümesinin "keskin" bölmelenmesidir.

5.3.1 Bulanık Küme-Ortalar Algoritması

Öğrenme sürecinde, eğitim kümesinden alınan özellik vektörlerine bulanık küme-ortalar algoritması uygulanarak sınıfların öz-vektörleri elde edilir.

Bulanık küme-ortalar algoritması, aşağıda verilmiş olan J_q, ağırlıklandırılmış en küçük kareler amaç fonksiyonunun minimizasyonuna dayanır:

$$J_q(U,v) = \sum_{j=1}^{n} \sum_{i=1}^{c} (u_{ij})^q . d^2(x_j, v_i); \quad 2 \le c < n, \quad 1 \le q < \infty \qquad (5.1)$$

$v=(v_1, v_2, \ldots, v_c) \in R^{cp}$, $v_i \in R^p$; c adet öz-vektör (model,prototip) kümesi.

$x=(x_1, x_2, \ldots, x_n) \subset R^p$; özellik uzayı R^p'de sınırlı bir veri (özellik vektörleri) kümesi.

$d^2(x_j,v_i)=\|x_j-v_i\|^2$ ve $\|\ .\ \|$ R^p üzerinde herhangi bir iç çarpım metriği (x_j ve v_i arasındaki mesafe).

Burada x_j, eğitim kümesinden gelen j-inci p boyutlu özellik vektörü; u_{ij}, [0,1] kapalı aralığında değer alan (u_{ij}: X → [0,1]) x_j'nin i-inci kümeye üyelik derecesi; v_i, i-inci bulanık kümenin (u_i) merkezi veya bulanık kümenin öz-vektörü (prototip); U, tüm üyelik değerlerinin matrisi; q, 1'den büyük ($q \in [1,\infty)$) herhangi bir reel sayı; n, veri noktaları (özellik vektörleri) sayısı; c, küme sayısıdır. q parametresi u_{ij} için bir ağırlıklandırma üssüdür ve oluşan kümelerin "bulanıklığını" kontrol etmektedir.

u_{ij} üyelik derecesi $d^2(x_j,v_i)$ mesafe ölçümünün tanımına bağlıdır. Burada:

$$d^2(x_j,v_i)=\|x_j-v_i\|^2=(x_j-v_i)^T(x_j-v_i) \qquad (5.2)$$

şeklinde Euclidean mesafesi olarak alınmıştır.

Aşağıda bulanık küme-ortalar algoritması görülmektedir:

1) Bulanık küme-ortalar algoritmasında herbir kümeye ait örneklerin ortalamaları alınarak kümelerin v_i (öz-vektör) ilk değer koşullamaları yapılır.

$$v_i = \frac{\sum_{l=1}^{n} x_{il}}{n}, \qquad i = 1,2,..,c \tag{5.3}$$

c, sınıf sayısı.

2) $\hat{u}_{ij} = u_{ij}$

3) Özellik vektörlerinin bütün kümelerdeki u_{ij} üyelik dereceleri hesaplanır.

$$u_{ij} = \frac{\dfrac{1}{d^2(x_j, v_i)}^{\frac{1}{(q-1)}}}{\sum_{k=1}^{c} \dfrac{1}{d^2(x_j, v_k)}^{\frac{1}{(q-1)}}} \tag{5.4}$$

4) Yeni v_i merkezleri hesaplanır.

$$v_i = \frac{\sum_{j=1}^{n} (u_{ij})^q . x_j}{\sum_{j=1}^{n} (u_{ij})^q} \tag{5.5}$$

5) Eğer $\max_{ij} [|\hat{u}_{ij} - u_{ij}|] < \epsilon$ ise öğrenme işlemi durdurulur; aksi taktirde adım 2'ye gidilir.

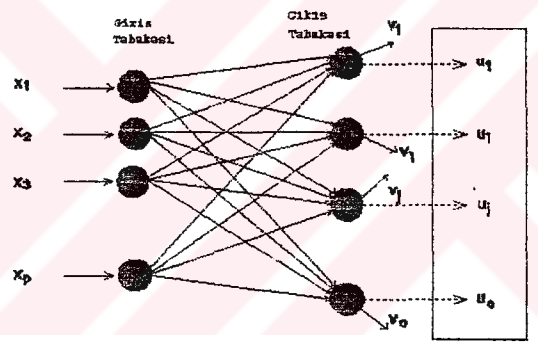
Burada ϵ, 0-1 arasında değer alan ve algoritmayı sonlandırmada kullanılan bir parametredir.

Elde edilen sonuç v_i değerleri, sınıflayıcılara giriş olarak verilecek öz-vektörlerdir.

5.3.2 Bulanık Kohonen Kümeleme Algoritması (Fuzzy Kohonen Clustering Algorithm, FKCA)

Öğrenme sürecinde, öz-vektörleri elde etmede kullanılan bir diğer algoritma, bulanık Kohonen kümeleme algoritmasıdır. Bu algoritma, eğitim kümesinde bulunan sınıflardaki kümelenme sayısından ve bu kümelenmelerin öbeklendiği yerlerden etkilenmeden sınıfları en iyi temsil eden öz-vektörleri (prototipleri) tespit etmeye çalışır.

Kohonen kümeleme ağı (KCA), şekil 5.3'te görülmektedir.



Şekil 5.3: Kohonen kümeleme ağı.

Kohonen kümeleme ağında, giriş tabakası direkt olarak çıkış tabakasına bağlıdır. Çıkış tabakasında, herbir düğüme bir ağırlık vektörü bağlanmıştır. $v=(v_1, v_2, \ldots, v_c)$ öz-vektörleri, ağın (bilinmeyen) küme merkezleri vektörüdür (ağırlıkları veya prototipleri). $v_i \in R^p$, $1 \le i \le c$ (c, küme sayısı). Ağa x giriş vektörü girdiğinde, herbir v_i'yle x arasındaki mesafe hesaplanır. Çıkış düğümleri yarışır ve minimum mesafeli v_i düğümü "kazanan" olarak belirlenir. Kohonen yönteminin bulanıklaştırılması, bulanık küme-ortalar algoritmasının Kohonen kümeleme ağına entegrasyonu sonucu gerçekleşir. Kohonen kümeleme ağındaki öğrenme oranlarının $\{\alpha_{ik,t}\}$ yerini, bulanık üyelik değerleri $\{u_{ik,t}\}$

almaktadır. Bu üyelik değerleri, bulanık küme-ortalar algoritmasında J_m'i minimize eden $u_{ik,t}$ değerleridir.

$$u_{jk,t} = \frac{\sum_{i=1}^{c} D_{ik,t}^{-2/(m-1)}}{D_{jk,t}^{-2/(m-1)}} \qquad (5.6)$$

$D_{ik,t}$, t-inci itersyondaki D_{ik} mesafesidir.

Kohonen güncelleştirilmesinde kullanılan öğrenme oranı aşağıdaki şekilde tanımlanır.

$$\alpha_{ik,t}=(u_{ik,t})^{m(t)}=(\beta/D_{ik,t})^{(2m(t)/(m(t)-1))} \qquad (5.7)$$

$$m(t)=(m(0)-t\Delta m); \qquad \Delta m=(m(0)-1)/t_{max}$$

$\alpha_{ik,t}$, t-inci iterasyonda k-ıncı özellik vektörünün i-inci sınıfa olan üyeliğinin değişimi (öğrenme oranı); $u_{ik,t}$, t-inci iterasyonda k-ıncı özellik vektörünün i-inci sınıfa olan üyeliği; m(0), 1'den büyük pozitif bir sabit (m_0=1); t_{max}, iterasyon limiti; β, pozitif bir sabittir.

$$D_{ik,t}=\| x_k - v_{i,t} \|^2$$

şeklinde, t iterasyonundaki D_{ik} mesafesidir.

Bulanık Kohonen kümeleme algoritmasına ait adımlar aşağıda verilmiştir:

1) Etiketlenmemiş veri kümesi $X=\{x_1, x_2, \ldots, x_n\}$ verilmek üzere, c (küme sayısı), $\|.\|$ (herhangi bir iç çarpım metriği) ve $1<\epsilon<0$ (iterasyonu durdurma sabiti) belirlenir.
2) Öz-vektörlere rastgele başlangıç değerleri atanır ve iterasyon sayısı t_{max} belirlenir. $m_0>1$ seçilir.

$$v_0=(v_{1,0},\ v_{2,0},\ldots,v_{c,0})\ \in R^{cp}$$

3) $t=1,2,\ldots,t_{max}$ için

a. Tüm (c.n) adet $\{a_{ik,t}\}$ öğrenme oranları hesaplanır, (5.7).

b. Tüm (c) adet $\{V_{i,t}\}$ öz-vektör (ağırlık vektörleri) değerleri güncelleştirilir.

$$v_{i,t}=v_{i,t-1}+(\Sigma_k a_{ik,t}(x_k-v_{i,t-1})/\Sigma a_{ik,t}) \qquad (5.8)$$

c. $E_t=\|v_t-v_{t-1}\|^2=\Sigma_i\|v_{i,t}-v_{i,t-1}\|^2$ hesaplanır.

d. Eğer $E_t\leq\epsilon$ ise öğrenme işlemi durdurulur; aksi taktirde adım 3'e gidilir.

Burada ϵ, 0-1 arasında değer alan iterasyonu durdurma katsayısıdır.

Yukarıdaki algoritma ile öz-vektörler elde edilerek öğrenme işlemi tamamlanır.

Klasik Kohonen öğrenme algoritmasında (5.7) no'lu denklemdeki "kazanan", x_k'ya en yakın olan $v_{i,t}$'dir. Limit durumda (m(t)=1) güncelleştirme kuralı, klasik küme-ortalar'daki hale döner, yani "kazanan hepsini alır". m(t)'nin büyük değerleri için (m(0) civarında), tüm c adet düğüm, küçük öğrenme oranlarıyla güncelleştirilir. $m(t)\rightarrow 1$ için güncelleştirmelerin büyük bir kısmı "kazanan" düğüme uygulanır. Diğer bir deyişle, öğrenme oranlarının dağılımı t'nin bir fonksiyonudur ve her x_k için $m(t)\rightarrow^{+}1$ iken kazanan düğümde daha şiddetlenir. FKCA'da her x_k için güncelleştirmeler tüm c adet düğüme uygulanır. FKCA'daki her adım, bulanık küme-ortalar'daki bir iterasyona karşı düşmektedir.

5.4 Sınıflama İşlemi

5.4.1 Bulanık K-En Yakın Komşu Sınıflayıcısı (Fuzzy K-Nearest Neighbour, Fuzzy K-NN Classifier)

Nesnelerin sınıflandırılması, örüntü tanıma ve yapay zeka, istatistik, görüntü analizi, tıp, vs. gibi pekçok alanda karşımıza çıkan önemli bir araştırma ve uygulama konusudur. Bir örüntü tanıma problemi olarak -sınıflama- konusunda çeşitli teknikler geliştirilmiştir. Çalışma uzayı hakkında ne kadar ön-bilgiye sahip olunursa, sınıflama algoritmasının gerçek durumu yansıtacak şekilde gerçekleştirilebilmesi o kadar mümkün olur. Fakat, birçok örüntü tanıma probleminde, giriş vektörünün sınıflandırılması, gerçek olasılık dağılımını temsil edemeyen ve herbir sınıfta az sayıda örneğin bulunduğu verilere dayanmaktadır. Tam bir ön-bilginin mevcut olmadığı bu tür durumlarda, birçok algoritma örnekler arasındaki mesafe ve benzerlik bilgisini kullanarak sınıflama yoluna gider. K-en yakın komşu (K-NN) algoritması da sınıflamada sıkça kullanılan bir algoritmadır. Bu karar kuralı, parametrik olmayan basit bir yöntemle, sınıflandırılacak giriş vektörüne, K yakınındaki (örneğin, Euclidean anlamında) komşularının sınıf etiketlerine dayanarak bir sınıf etiketi atar.

Bu karar kuralının tercih edilmesinin sebebi, getirdiği hesap basitliği ve örnek sayısının az olduğu birçok problemde şaşırtıcı iyi sonuçlar vermesidir.

K-NN sınıflayıcılarda, giriş vektörüne sınıf etiketleri atanırken herbir örnek vektörüne eşit önem verilmesi problem yaratmaktadır. Örnek kümelerinin örtüştüğü yerlerde bu sakıncalı bir durumdur. Kümeleri gerçekten temsil eden vektörlerle, bu kümelerin tipik vektörü olmayan vektörlere aynı büyüklükte ağırlıklar verilmektedir. Bir diğer problem ise; bir giriş vektörü

bir sınıfa atandığında, bu vektörün bu sınıfa üyeliğinin gücü hakkında herhangi bir bilginin olmamasıdır. Karşılaşılan bu iki problem neticesinde, bulanık küme teorisi K-NN kuralına dahil edilmiştir.

Giriş vektörüne bulanık sınıf üyeliğinin atanması, bulanık etiketlere dayanan bulanık kararlar verilmesine neden olur. Burada da bulanık küme metodları, klasik K-NN karar kuralıyla birleştirilmiştir. Örnek kümelerin bulanık sınıf üyelikleri kullanılarak "bulanık K-NN" algoritması geliştirilmiş ve böylece bulanık sınıflama kuralı çıkarılmıştır. Klasik K-NN sınıflayıcıya nazaran bulanık K-NN sınıflayıcı ile daha küçük hata oranları elde edilmiştir. Üyelik değerlerinin, sınıflamada bir güven ölçümü yerine geçtiği de gözardı edilmemesi gereken önemli bir husustur.

Bulanık K-en yakın komşu algoritmasının sonucu, klasik K-NN versiyonundan farklıdır. Klasik K-NN sınıflayıcı kuralı, ait olduğu sınıfı bilinmeyen y giriş örnek vektörünü en yakın komşularının sınıfına atar. Bulanık K-NN algoritması ise örnek vektörü belirli bir sınıfa dahil etmek yerine, bu vektöre sınıf üyeliği atar. Avantajı, algoritma tarafından keyfi atamaların yapılmayışıdır. Buna ek olarak, vektörün üyelik değerleri, sınıflama sonucuna eşlik eden bir güven seviyesi oluştururlar. Örneğin, bir vektör 0.9 üyelikle bir sınıfa, 0.05 üyelikle de farklı iki sınıfa üye ise, haklı olarak 0.9 üyelikli sınıfı, bu vektörün ait olduğu sınıf olarak tayin edebiliriz. Diğer taraftan, bir vektör birinci sınıfta 0.55 üyeliğe, ikinci bir sınıfta 0.44 ve üçüncü bir sınıfta da 0.01 üyeliğe sahip ise bu vektörün ait olduğu sınıfı belirlerken bir tereddüte düşebiliriz. Vektörün üçüncü sınıfa ait olmadığı açıkça görülmektedir. Böylesi bir durumda vektör; birinci ve ikinci sınıflarda büyük bir üyelik değeri gösterdiğinden, sınıflamayı gerçekleştirebilmek amacıyla daha sonra tekrar

incelenebilir. Algoritma tarafından üretilen üyelik atamaları, sınıflama işleminde faydalı olacaktır.

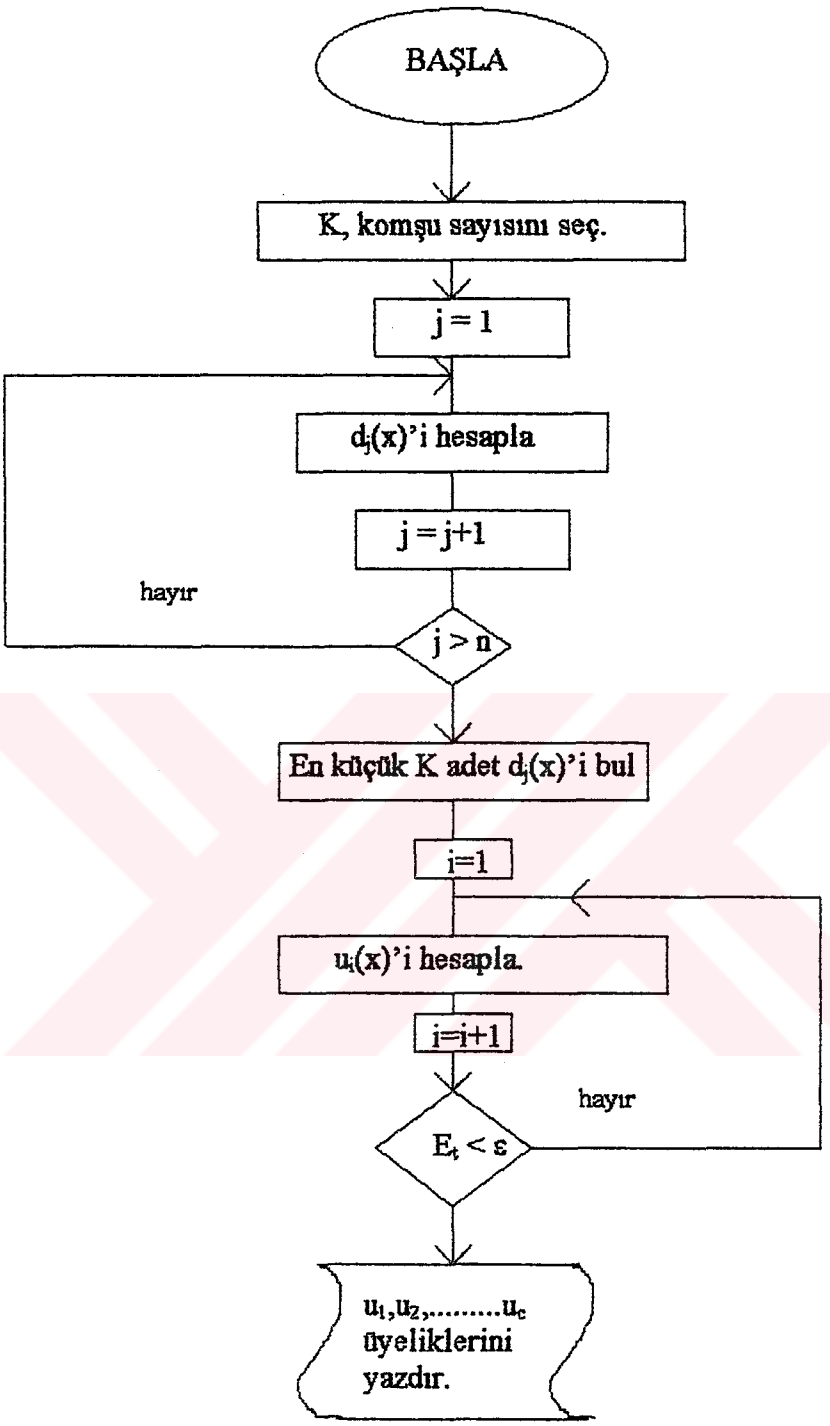
Algoritmanın esası; vektörün K-en yakın komşularına olan uzaklıklarının ve bu komşuların olası sınıflardaki üyeliklerinin fonksiyonu olarak bu giriş vektörüne üyelik atanmasıdır. Klasik K-NN'de olduğu gibi, bulanık K-NN'de de K-en yakın komşular için etiketlenmiş örnek küme araştırılmıştır. Şekil 5.4'te bulanık K-en yakın komşu sınıflayıcısına ait algoritma görülmektedir.

$W=\{x_1, x_2, \ldots\ldots, x_n\}$, n adet etiketlenmiş örnekler (eğitim kümesinden gelen örnekler) kümesi; x_j, eğitim kümesinden gelen komşu özellik vektörü; u_{ij}, j-inci komşu vektörünün i-inci sınıfa olan üyeliği; K, komşuluk sayısı; m, pozitif bir ağırlık katsayısı; x, test kümesinden gelen özellik vektörü; $u_i(x)$, x vektörüne atanmak üzere hesaplanacak üyelik değeri (x'in i-inci sınıfa olan üyeliğinin derecesi) olmak üzere bulanık K-NN algoritması aşağıdaki şekildedir:

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij} \left(1 / [d_j^2(x)]^{1/(m-1)}\right)}{\sum_{j=1}^{K} \left(1 / [d_j^2(x)]^{1/(m-1)}\right)} \qquad (5.9)$$

$$d_j^2(x) = \|x-x_j\|^2 = (x-x_j)^T.(x-x_j)$$

(5.9) denkleminden de görüldüğü gibi x'e atanan üyelikler, en yakın komşuların sınıf üyeliklerine ve bu komşuların x'den olan uzaklıklarının tersine bağlıdır. Ters mesafe kullanılmasının sebebi; komşu vektör sınıflandırılacak vektöre yakınken üyeliğini daha fazla ağırlıklandırabilmek, uzak iken de üyeliğini daha az ağırlıklandırmaktır.



Şekil 5.4: Bulanık K-en yakın komşu Algoritması.

Eğitim kümesinden gelen etiketlenmiş örneklere sınıf üyeliği çeşitli şekillerde atanabilir. Öncelikle, bu vektörlere kendi bilinen sınıflarında tam üyelik ve diğer tüm sınıflarda 0 üyelik verilebilir. Diğer alternatifler; örneklerin kendi sınıf ortalamalarından olan uzaklıklarından faydalanarak ya da kendi sınıflarındaki ve diğer sınıflardaki etiketlenmiş örneklerden olan uzaklıklardan faydalanarak örneklere üyeliğin atanmasıdır. Bulunan bu üyelikler daha sonra sınıflayıcıda kullanılmaktadır.

m değişkeni, herbir komşunun üyelik değerine katkısı hesaplanırken, mesafenin hangi şiddette ağırlıklandırılacağını tayin eder. m=2 alınırsa, herbir komşu noktanın katkısı, sınıflandırılmakta olan noktaya olan uzaklığın resiproku ile ağırlıklandırılır. m arttıkça, komşular daha muntazam ağırlıklandırılırlar ve sınıflandırılan noktaya olan bağıl uzaklıkları daha az bir etkiye sahip olur. m, 1'e yaklaştıkça, yakın komşular uzakta olanlara nazaran çok daha büyük oranda ağırlıklandırılırlar. Bu da, sınıflandırılmakta olan noktanın üyelik değerine katkıda bulunan nokta sayısının azalmasına yol açar. Yapılan tez çalışmasında m=2 olarak alınmıştır. Fakat m'in geniş bir bölge üzerinde aldığı değerlerle yaklaşık eşit sınıflama başarımları elde edilmiştir.

5.4.2 Bulanık En Yakın Model Sınıflayıcısı (Fuzzy Nearest Prototype, Fuzzy NP Classifier)

En yakın model sınıflayıcıları, 1-en yakın komşu (1-NN) sınıflayıcısına büyük bir benzerlik gösterir. Aradaki tek fark; en yakın model sınıflayıcıda, etiketli örneklerin bir sınıf modelleri (prototip, öz-vektör) kümesi olması, buna karşın en yakın komşu sınıflayıcıda, model (prototip) teşkil etmesi gerekmeyen etiketlenmiş örnekler kümesinin kullanılmasıdır. Etiketlenmiş örnek kümesinin sınıf

ortalamaları alınarak öz-vektörler (sınıf modelleri) oluşturulmuştur. Bulanık en yakın model algoritması aşağıdaki şekildedir:

BAŞLA
    Sınıfı bilinmeyen x özellik vektörünü giriş olarak ver.
    i=1 başlangıç değerine ayarla.
    DEVAM ET (herbir öz-vektörden x'e olan mesafe
    hesaplanıncaya dek)
        v_i'den x'e olan mesafeyi hesapla.
        i'yi artır.
    SONLANDIR (DEVAM ET döngüsü sonu)
    i=1 başlangıç değerine ayarla.
    DEVAM ET (x'e tüm sınıflarda üyelik atanıncaya dek)
        $u_i(x)$'i (5.10) formülünü kullanarak hesapla.
        i'yi artır.
    SONLANDIR (DEVAM ET döngüsü sonu)
SONLANDIR

$$u_i(x) = \frac{1/\|x-v_i\|^{2/(m-1)}}{\sum_{j=1}^{c} (1/\|x-v_j\|^{2/(m-1)})} \qquad (5.10)$$

Burada, x, test kümesinden gelen özellik vektörü; $u_i(x)$, x vektörüne atanmak üzere hesaplanacak üyelik değeri (x'in i-inci sınıfa olan üyeliğinin derecesi); v_i, i-inci sınıfın öz-vektörü (modeli); m, +2 olarak seçilmiş ağırlık katsayısı; c, sınıf sayısıdır.

En yakın model sınıflayıcıda, herbir sınıftaki üyelikler hesaplanırken, sınıfın öz-vektöründen olan mesafe bilgisi kullanılmakta, en yakın komşu sınıflayıcılarında olduğu gibi bu mesafeler öz-vektörlerin üyelikleriyle ayrıca ağırlıklandırılmaktadırlar. Öz-vektörlerin temsil ettikleri sınıfta tam üyeliğe sahip olmalarının neticesinde, doğal olarak bu durum ortaya çıkmaktadır.

En yakın model sınıflayıcısı, hızlı ve basit bir sınıflayıcıdır. Çünkü sınıflandırılan örnek, herbir sınıftan gelen sadece bir öz-vektörle karşılaştırılmaktadır. K-en yakın komşu algoritmasında ise, sınıflandırılacak özellik vektörü, K en yakınlar elde edilmeden önce, herbir sınıfı temsil eden etiketlenmiş örnekler kümesinin tümüyle karşılaştırılmaktadır.

BÖLÜM 6

PRATİĞE UYGULAMA: EKG İŞARETİNDEKİ ŞEKİL BOZUKLUKLARININ TESPİTİ

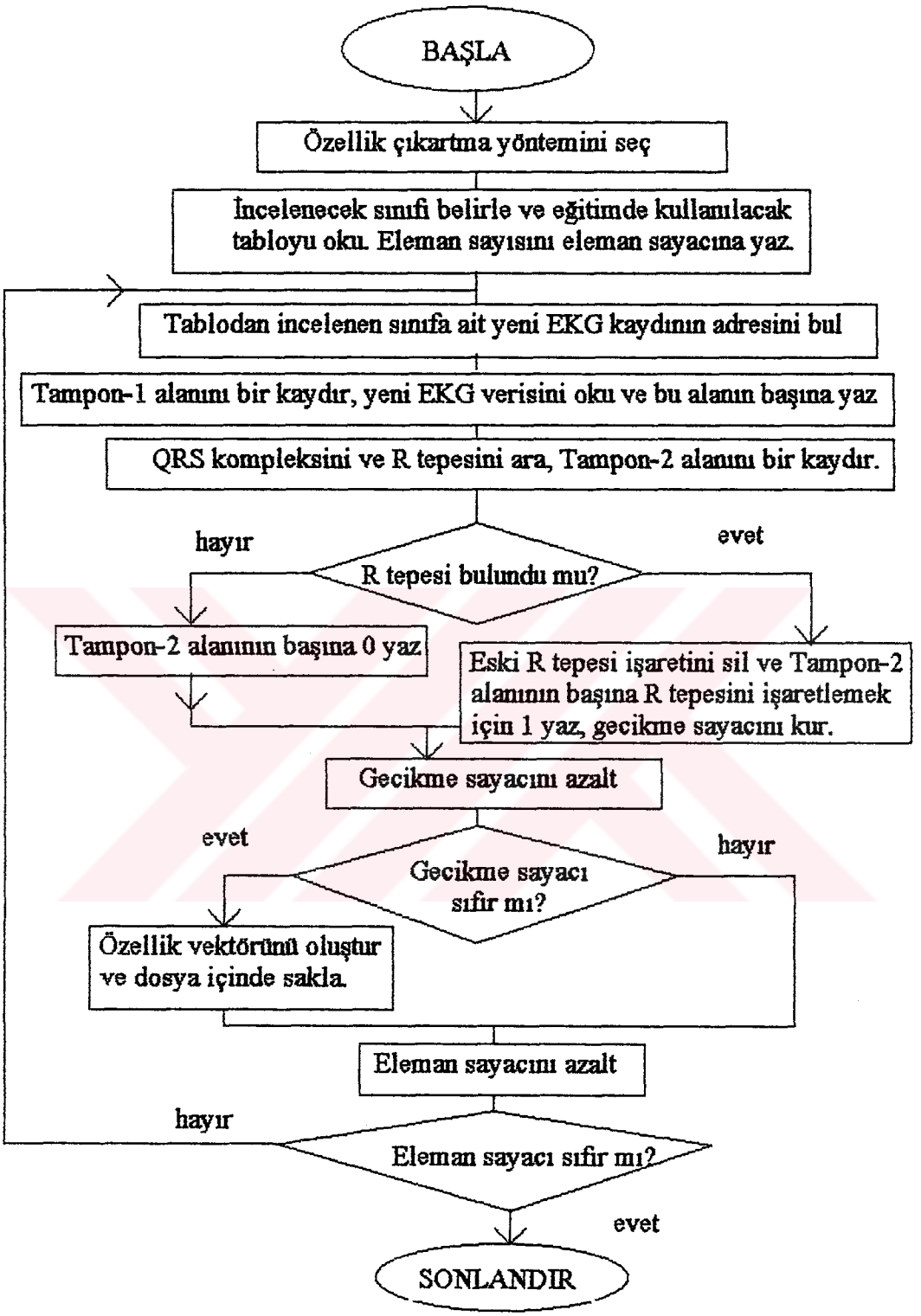
6.1 Giriş

Bu bölüm içerisinde 4 farklı EKG işaretinin sınıflandırılması konusu incelenmiştir. Literatürde kullanılan 4 farklı özellik elde etme yönteminin sınıflamadaki başarıları araştırılmıştır. Çalışma içerisinde önce bulanık sınıflayıcıların eğitilmesi üzerinde durulmuş, daha sonra iki farklı bulanık sınıflayıcı kullanılarak EKG işaretlerinin sınıflandırılması gerçeklenmiştir.

6.2 EKG İşaretinden Özellik Vektörlerinin Elde Edilmesi

EKG işaretinin şekli kullanılan derivasyona göre büyük farklılıklar göstermektedir. Dolayısıyla sınıflama işleminde sadece incelenen derivasyonun öz vektörleri kullanılmalıdır. Çalışma içerisinde MLII derivasyonu için öz vektörler elde edilmiştir. Diğer derivasyonlar için öz vektörler benzer şekilde elde edilir.

İncelenen derivasyon içerisindeki şekil bozukluklarını bulanık sınıflayıcılara öğretmek için önce eğitim kümesi oluşturulur. Sınıflayıcıların öz vektörleri bu küme ile bulunduktan sonra test kümesi kullanılarak sınıflayıcıların başarıları incelenir. P, N, L ve V olarak etiketlenen şekil bozukluklarından 15'er örnek alınarak eğitim kümesi oluşturulur. Şekil 6.1'deki algoritma kullanılarak her bir sınıf için özellik vektörleri elde edilir.



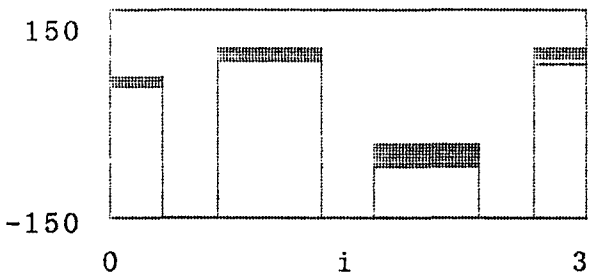
Şekil 6.1: Özellik vektörlerinin elde edilmesi.

Programın başında özellik vektörlerinin saklanacağı dosya ismi, seçilen özellik çıkartma yöntemi ve incelenen sınıf ile bu sınıf için eleman sayısı sorulur. Program çalıştırılmadan önce incelenen sınıfa ait bilgileri içeren bir dosya oluşturulur. Bölüm 2'de kullanılan tablonun yapısı ve EKG kayıtlarının okunuş şekli anlatılmıştır. Algoritma içerisinde bölüm 3'te anlatılan R tepesini bulma yöntemi kullanılır. R tepesi bulunduktan sonra bölüm 4 içerisinde anlatılan yöntemlerden biri kullanılarak (programın başlangıcında kullanılacak yöntem seçilir) özellik vektörleri oluşturulup bir dosya içerisine yazılır.

Tüm özellik çıkartma işlemlerinde R tepesi etrafındaki (solunda ve sağındaki) EKG verileri kullanılır. Bu nedenle R tepesi tespit edilse bile işlem yapılmaz; EKG verileri bir bellek alanında R tepesinin yer bilgisi ile birlikte geciktirilir. R tepesi etrafında (solunda ve sağında) yeteri kadar EKG veri bilgisi oluştuktan sonra özellik çıkartma işlemleri gerçeklenir.

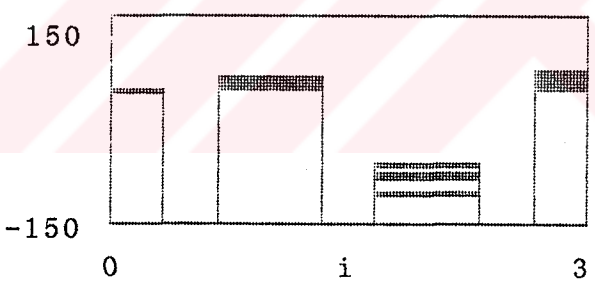
İncelenen sınıftan eleman sayısı kadar yukarıda anlatılan işlemler tekrarlanır.

Şekil 6.2.a'da P; 6.2.b'de N; 6.2.c'de L ve 6.2.d'de V tipi şekil bozuklukları için QRS şekil bilgisinden elde edilmiş özellik vektörlerinin histogramı gösterilmiştir. Şekil 6.3.a'da P; 6.3.b'de N; 6.3.c'de L ve 6.3.d'de V tipi şekil bozuklukları için LPC katsayılarından oluşturulmuş özellik vektörlerinin histogramı gösterilmiştir. Şekil 6.4.a'da P; 6.4.b'de N; 6.4.c'de L ve 6.4.d'de V tipi şekil bozuklukları için frekans spektrumu kullanılarak oluşturulmuş özellik vektörlerinin histogramı gösterilmiştir. Şekil 6.5.a'da P; 6.5.b'de N; 6.5.c'de L ve 6.5.d'de V tipi şekil bozuklukları için EKG işaretinin R tepesi etrafındaki genlik değerleri kullanılarak oluşturulmuş özellik vektörlerinin histogramı gösterilmiştir.

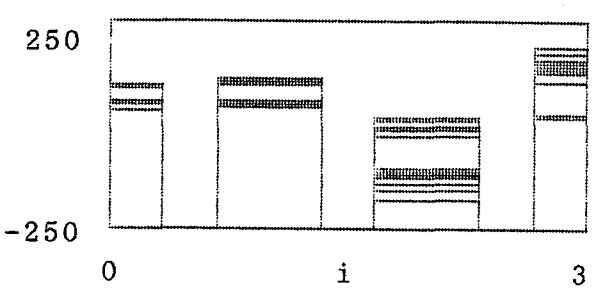


Şekil 6.2.a: P tipi şekil bozukluğu için eğitim kümesi.

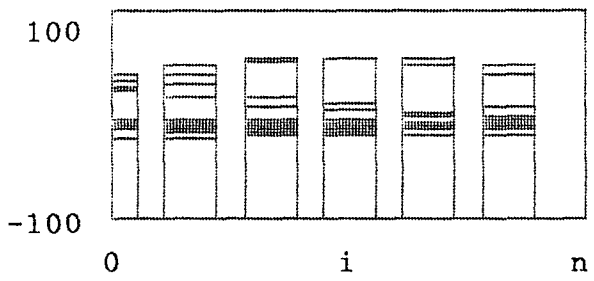
Şekil 6.2.b: N tipi şekil bozukluğu için eğitim kümesi.



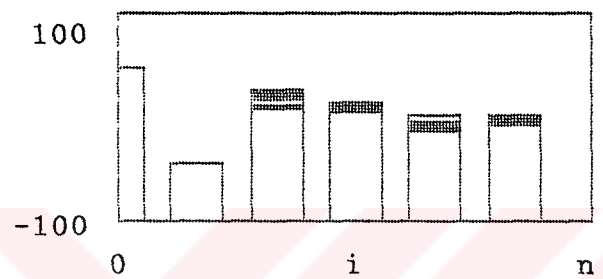
Şekil 6.2.c: L tipi şekil bozukluğu için eğitim kümesi.



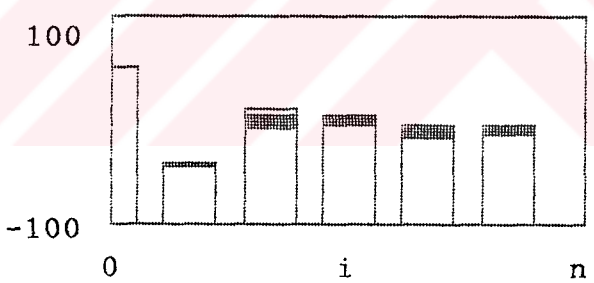
Şekil 6.2.d: V tipi şekil bozukluğu için eğitim kümesi.



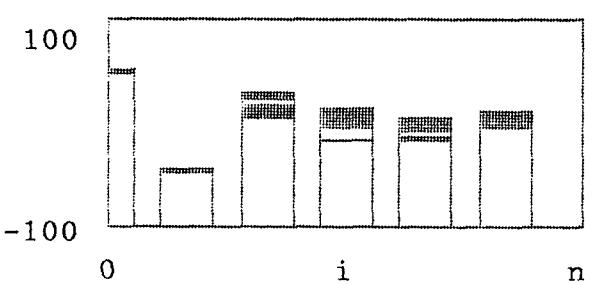
Şekil 6.3.a: P tipi şekil bozukluğu için eğitim kümesi.



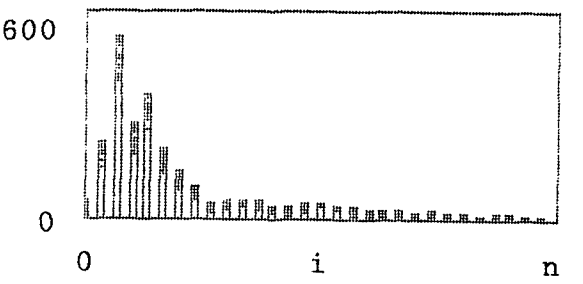
Şekil 6.3.b: N tipi şekil bozukluğu için eğitim kümesi.



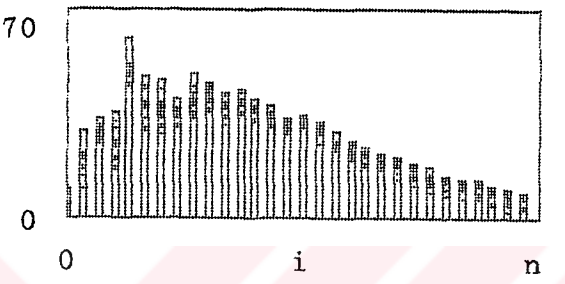
Şekil 6.3.c: L tipi şekil bozukluğu için eğitim kümesi.



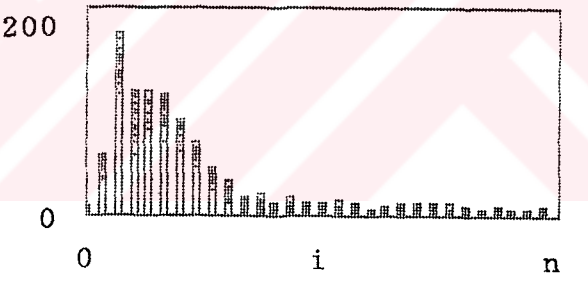
Şekil 6.3.d: V tipi şekil bozukluğu için eğitim kümesi.



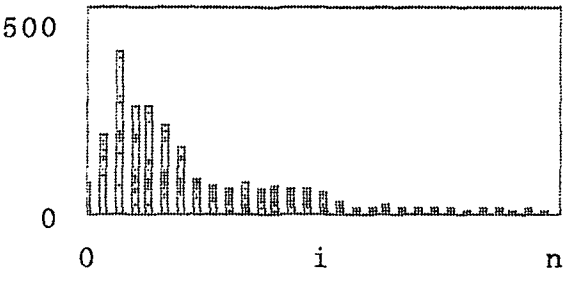
Şekil 6.4.a: P tipi şekil bozukluğu için eğitim kümesi.



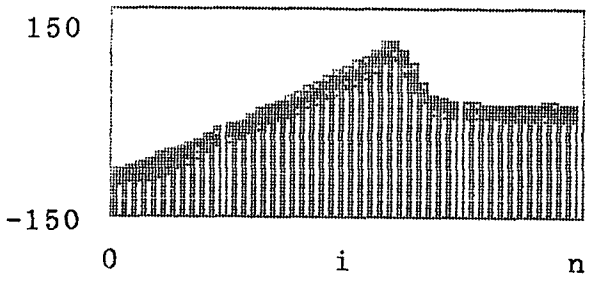
Şekil 6.4.b: N tipi şekil bozukluğu için eğitim kümesi.



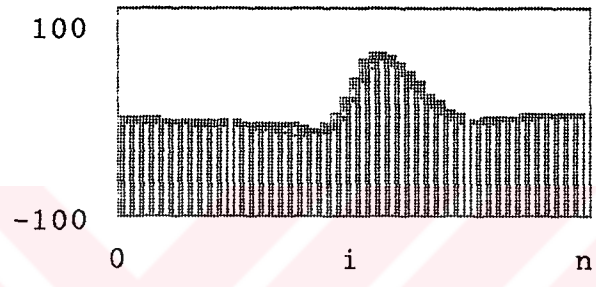
Şekil 6.4.c: L tipi şekil bozukluğu için eğitim kümesi.



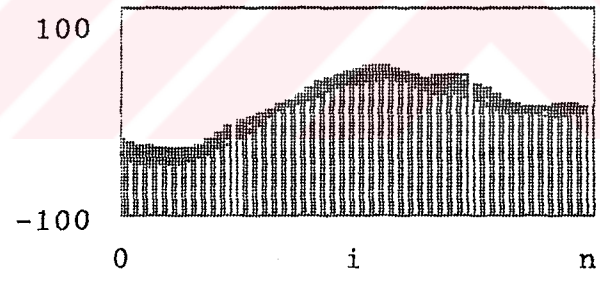
Şekil 6.4.d: V tipi şekil bozukluğu için eğitim kümesi.



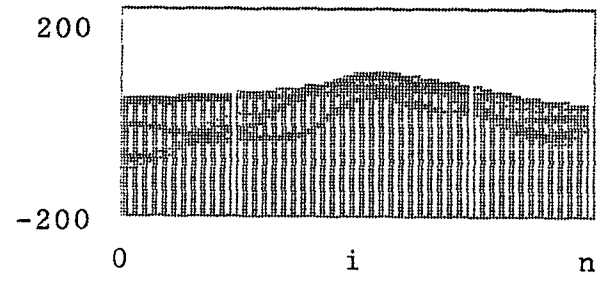
Şekil 6.5.a: P tipi şekil bozukluğu için eğitim kümesi.



Şekil 6.5.b: N tipi şekil bozukluğu için eğitim kümesi.



Şekil 6.5.c: L tipi şekil bozukluğu için eğitim kümesi.



Şekil 6.5.d: V tipi şekil bozukluğu için eğitim kümesi.

6.3 Öz Vektörlerin Bulunması

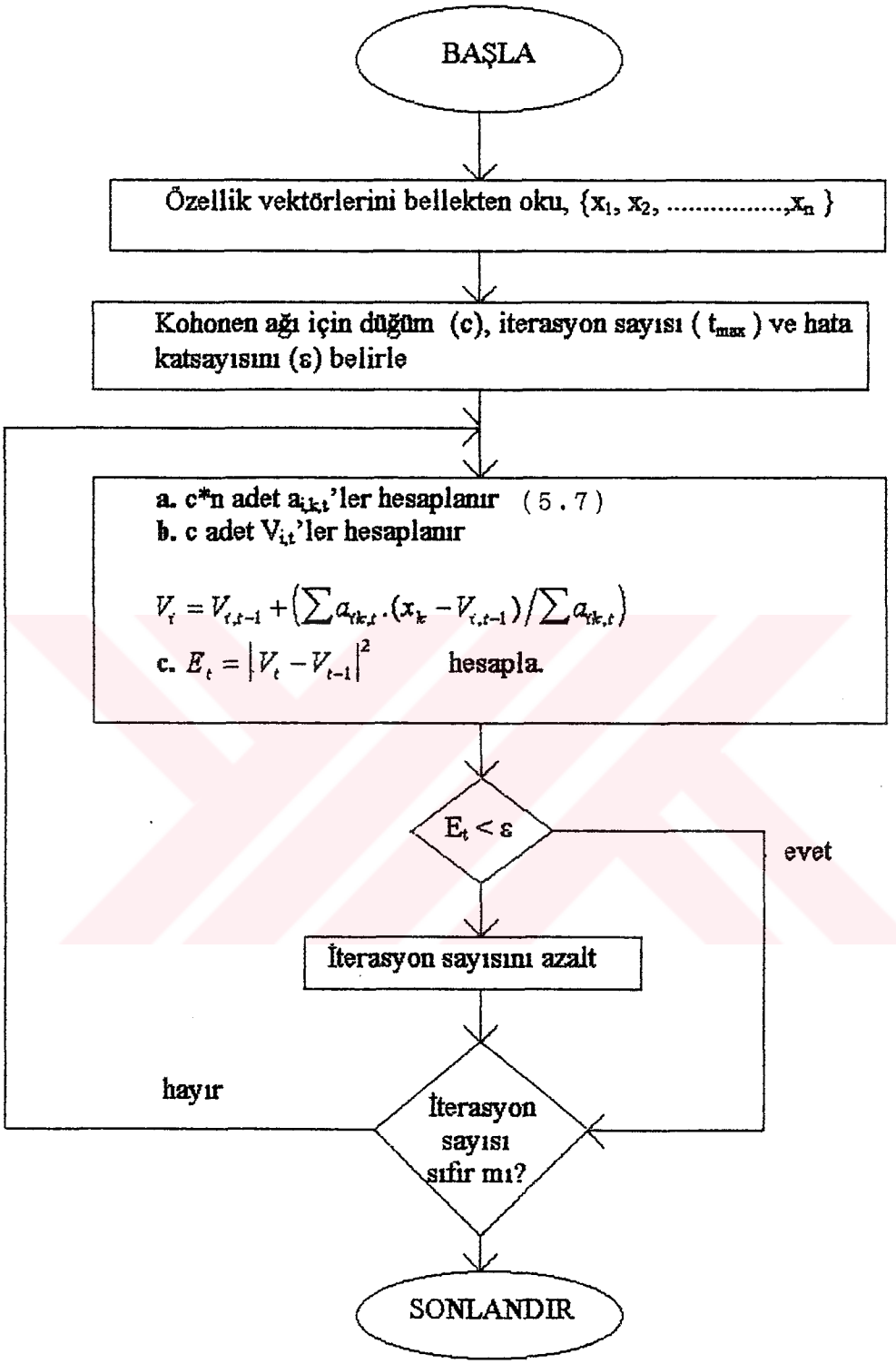
MLII derivasyonunda karşılaşılan 4 farklı EKG işaretinin herbiri için iki küme oluşturulmuştur: Eğitim kümesi ve Test kümesi. Bölüm 6.2'de bulunan özellik vektörleri kullanılarak 4 farklı EKG işareti için öz-vektörler bulunmuştur. Bölüm 5 içerisinde iki farklı öğrenme yöntemi incelenmiştir.

Şekil 6.6.a'da öz-vektörlerin bulanık Kohonen algoritması kullanılarak elde edilmesi gösterilmiştir.

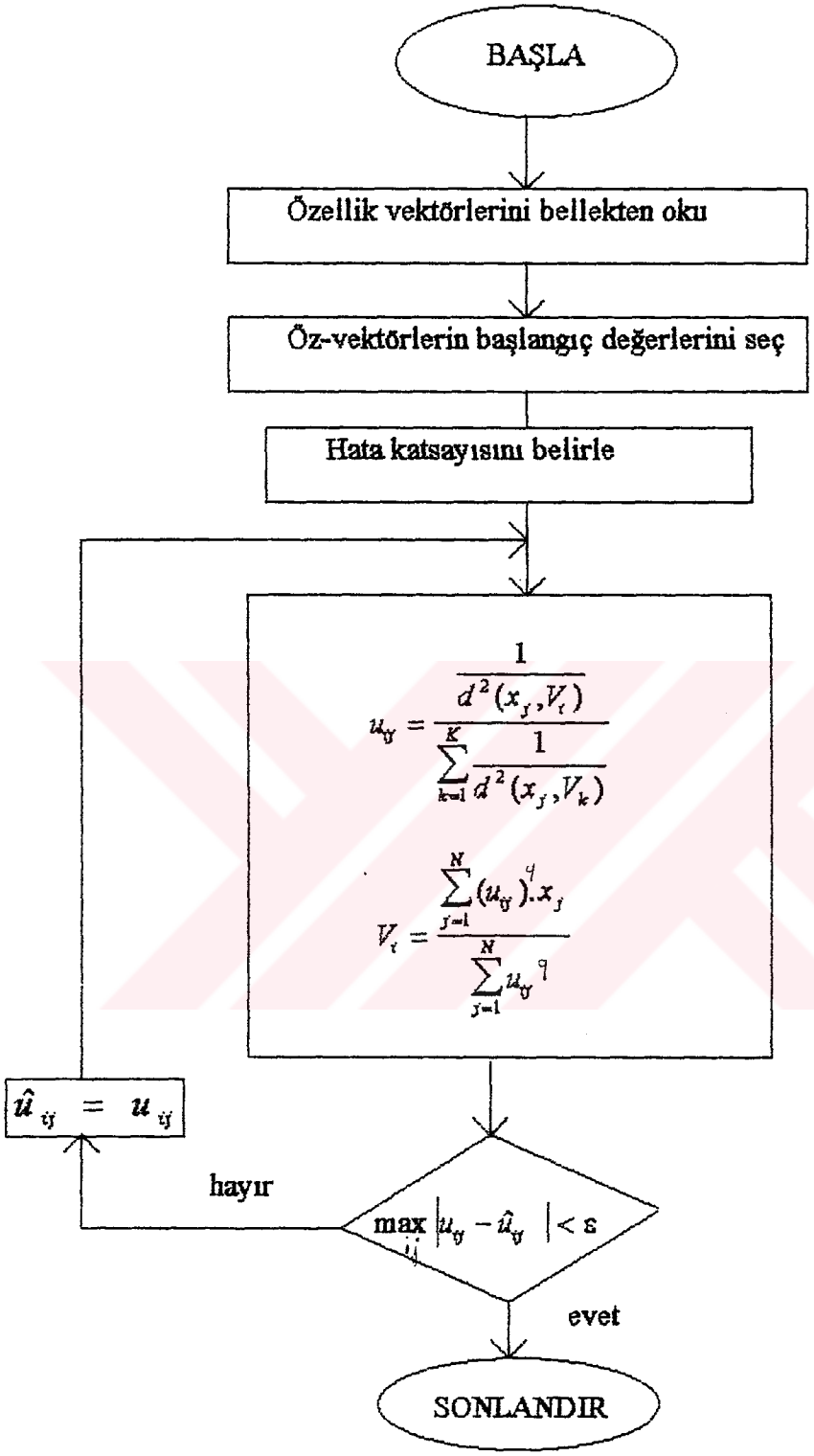
Bölüm 6.2'de 4 farklı sınıf için 4 farklı özellik çıkartma yöntemi kullanılarak 16 değişik sonuç, farklı isimlerle diskte saklanır. Bulanık Kohonen öğrenme algoritması her sonucu farklı isimli bir dosya ile çağırır. Programın ikinci adımında düğüm, iterasyon sayısı ve hata katsayısı belirlenir. Kohonen düğümlerine başlangıç değeri verilerek bölüm 4'te anlatıldığı şekilde algoritma çalıştırılır. Tez içerisinde yakınsama hızını artırmak için düğümlerin başlangıç değerleri, öğrenme kümesindeki vektörlerden temin edilir.

Şekil 6.6.b'de öz-vektörlerin bulanık küme-ortalar algoritması kullanılarak elde edilmesi gösterilmiştir. Her sınıfın öz-vektörlerinin başlangıç değeri, incelenen sınıfın özellik vektörlerinin ortalama değerleri alınarak elde edilir.

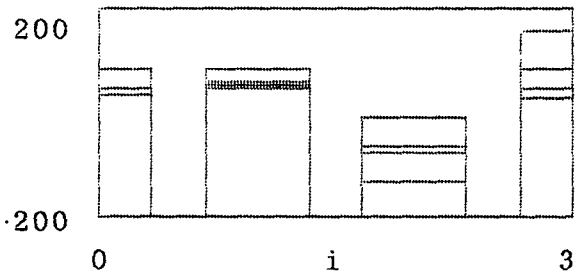
Şekil 6.7.a'da 4 sınıf için bulanık küme-ortalar algoritması kullanılarak özellik vektörlerinden (QRS şeklinden) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.7.b'de 4 sınıf (L, V, N ve P) için bulanık küme-ortalar algoritması kullanılarak özellik vektörlerinden (frekans spektrumundan) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.7.c'de 4



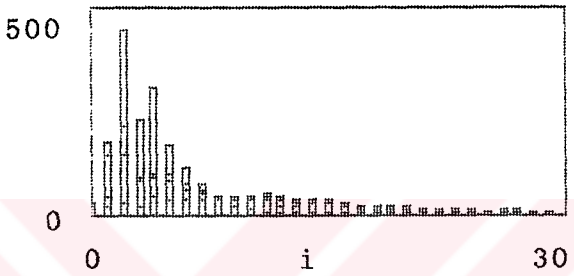
Şekil 6.6.a: Bulanık Kohonen Öğrenme Algoritması.



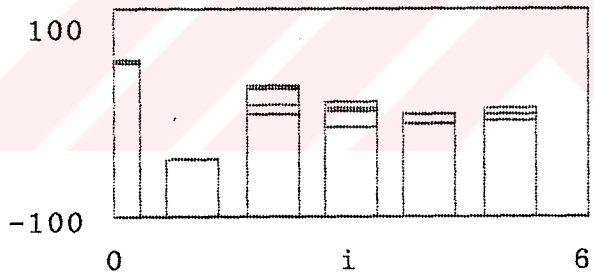
Şekil 6.6.b: Bulanık Küme-Ortalar Algoritması.



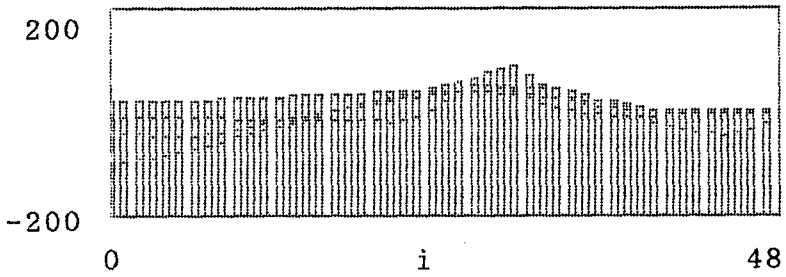
Şekil 6.7.a: QRS şeklinden elde edilen öz-vektör kümesi.



Şekil 6.7.b: Frekans spektrumundan elde edilen öz-vektör kümesi.



Şekil 6.7.c: LPC'den elde edilen öz-vektör kümesi.



Şekil 6.7.d: Direkt EKG işaretinden elde edilen öz-vektör kümesi.

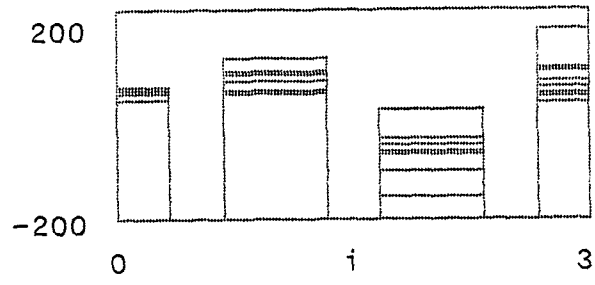
sınıf için bulanık küme-ortalar algoritması kullanılarak özellik vektörlerinden (LPC parametrelerinden) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.7.d'de 4 sınıf için bulanık küme-ortalar algoritması kullanılarak özellik vektörlerinden (direkt EKG işaretinden) elde edilen öz-vektörlerin histogramı gösterilmiştir.

Şekil 6.8.a'da 4 sınıf için bulanık Kohonen algoritması kullanılarak özellik vektörlerinden (QRS şeklinden) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.8.b'de 4 sınıf için bulanık Kohonen algoritması kullanılarak özellik vektörlerinden (frekans spektrumundan) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.8.c'de 4 sınıf için bulanık Kohonen algoritması kullanılarak özellik vektörlerinden (LPC parametrelerinden) elde edilen öz-vektörlerin histogramı gösterilmiştir. Şekil 6.8.d'de 4 sınıf için bulanık Kohonen algoritması kullanılarak özellik vektörlerinden (direkt EKG işaretinden) elde edilen öz-vektörlerin histogramı gösterilmiştir.

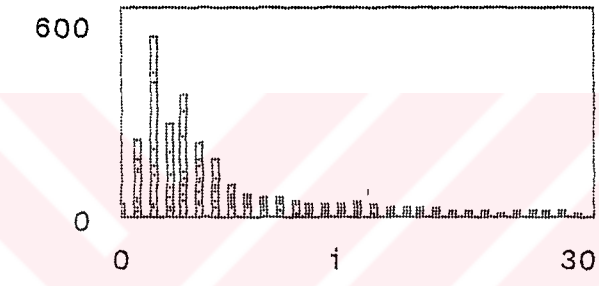
6.4 EKG İşaretlerinin Sınıflandırılması

Tez içerisinde iki bulanık sınıflayıcı kullanılmıştır: Bulanık en yakın model ve bulanık K-en yakın komşu sınıflayıcıları. Bulanık en yakın model sınıflayıcısında öz-vektörler, bulanık küme-ortalar algoritması kullanılarak elde edilir. Çalışma içerisinde kullanılan bulanık küme-ortalar algoritmasında sınıf sayısı kadar öz-vektör seçilmiştir.

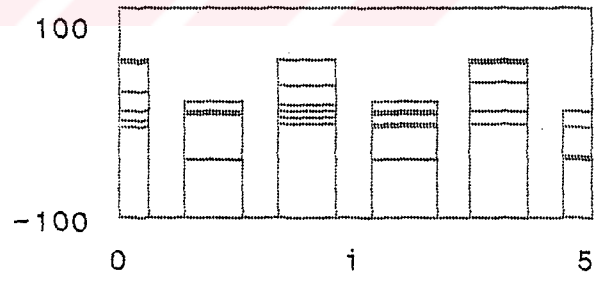
Şekil 6.9.a'da en yakın model sınıflayıcısı blok olarak gösterilmiştir. Öğrenme bloğu sadece öğrenme işleminde kullanılır; sınıflama işeminde bu blok kullanılmaz. Şekil 6.9.b'de K-en yakın komşu sınıflayıcısı blok olarak gösterilmiştir.



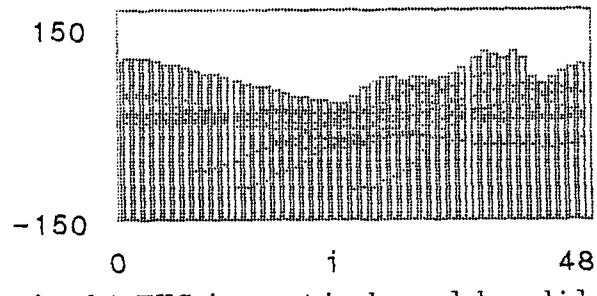
Şekil 6.8.a: QRS şeklinden elde edilen öz-vektör kümesi.



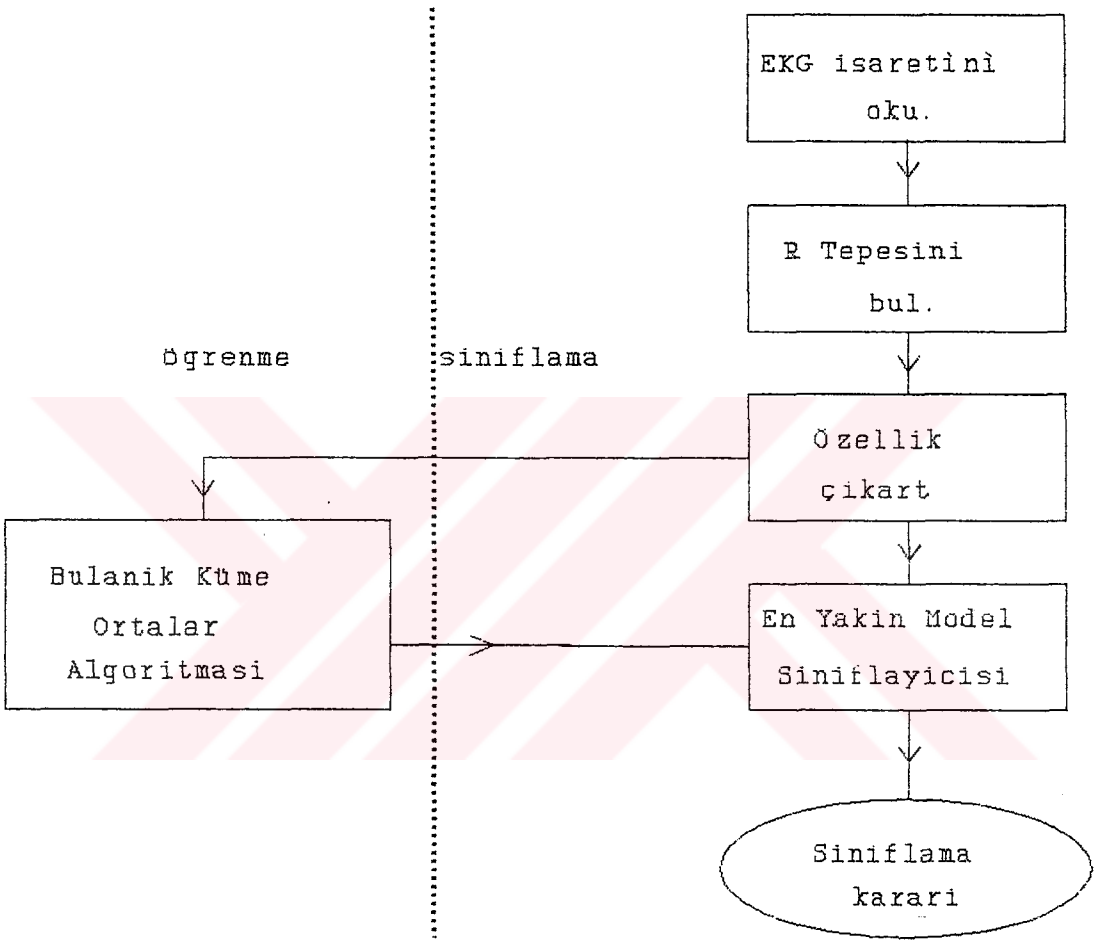
Şekil 6.8.b: Frekans spektrumundan elde edilen öz-vektör kümesi



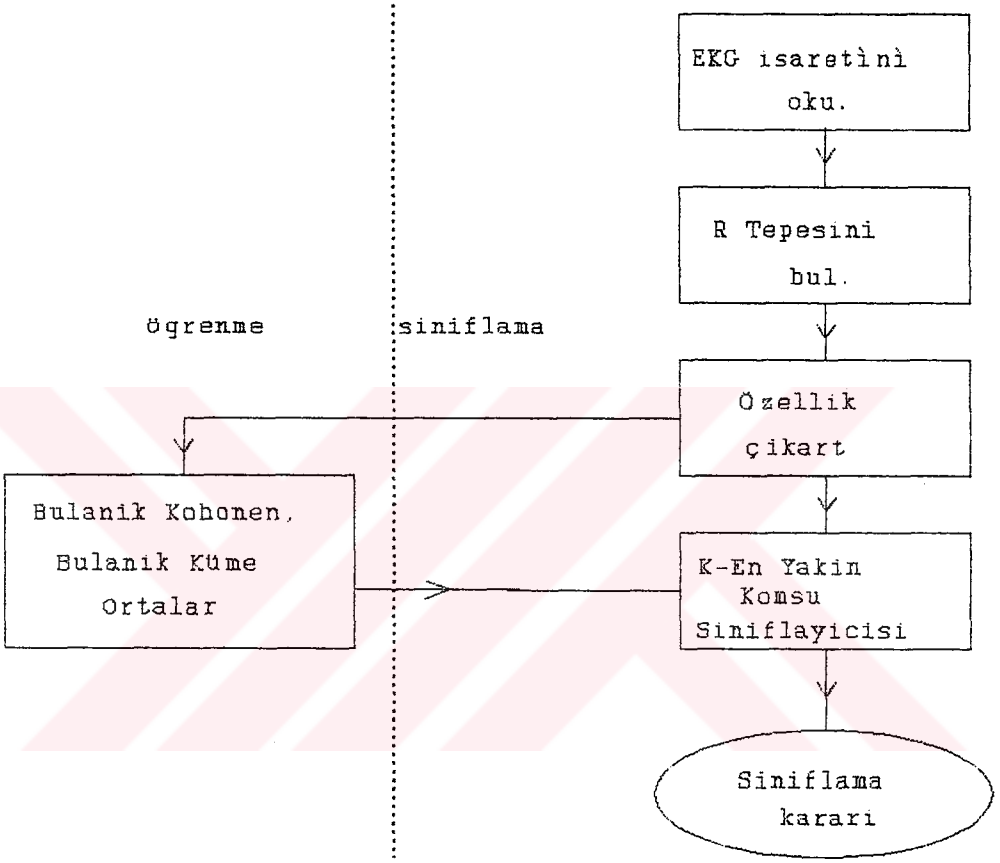
Şekil 6.8.c: LPCs'den elde edilen öz-vektör kümesi.



Şekil 6.8.d: Direkt EKG işaretinden elde edilen öz-vektör kümesi.



Şekil 6.9.a: En yakın model sınıflayıcının blok gösterilimi.

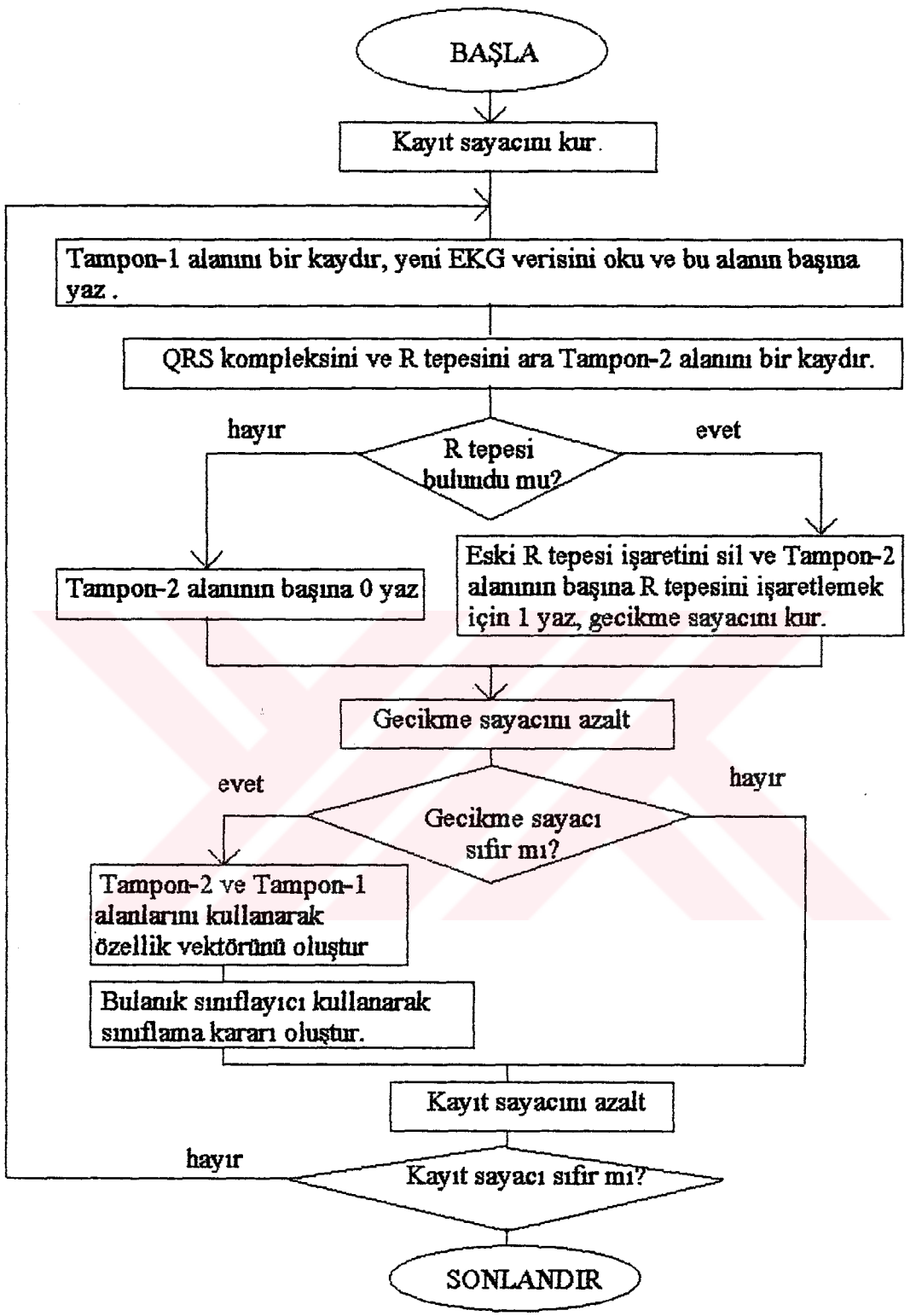


Şekil 6.9.b: K-en yakın komşu sınıflayıcısının blok gösterilimi.

Bulanık K-en yakın komşu sınıflayıcısında öz-vektörler, iki değişik yöntem kullanılarak elde edilir: Bulanık Kohonen algoritması ve eğitim kümesini kullanmak.

Şekil 6.10'da tüm kayıt içerisindeki EKG işaretlerinin sınıflandırılmasında kullanılan algoritma gösterilmiştir.

Kayıt sayıcısı içerisinde, incelenen kayıt (*.dat) dosyasındaki QRS komplekslerinin sayısı tutulur. Kayıttan okunan EKG verileri bir tampon (tampon-1) alana yazılır. Veriler diskten okunmadan önce bu alan bir kaydırılır. Tampo-1 alanı içerisinde bölüm 3'te anlatılan algoritmalar kullanılarak QRS kompleksi ve R tepesi aranır. R tepesi olarak düşünülen o anki EKG işareti ayrı bir tampon (tampon-2) alanda işaretlenir. İşaretleme işlemi yapılmadan önce alan bir kaydırılır. İşaret üzerinde yeni gelen değerler incelenir ve eski gözlem ile karşılaştırılır. Eğer eski karar yanlış olarak düşünülüp R tepesinin yeri yeniden işaretlenmek istenirse, tampon-2 alanındaki eski R tepesi işareti silinir, tampon-2 alanının başına R tepesi işareti konur ve gecikme sayacı kurulur. Gecikme işleminin yapılmasının iki sebebi vardır: 1) R tepesinin yeri yeni gelen işaretlerle yeniden ayarlanabilir. Bu nedenle biraz gecikme yapılarak kararın sabit kalması beklenir. 2) Özellik vektörlerinin oluşturulabilmesi için R tepesinin etrafı kullanılır. Bir miktar gecikme yapılarak R tepesinin etrafında inceleme yapabilme olanağı elde edilir.



Şekil 6.10: Sınıflandırma Algoritması

BÖLÜM 7

SONUÇ

Tez içerisinde, MIT-BIH veritabanı kullanılarak 4 farklı EKG şeklinin (sınıf) ayırt edilmesi için algoritmalar gerçekleştirilmiştir. Sınıflayıcıların tasarımından önce literatürde kullanılan özellik vektörleri incelenerek, en iyi sonuç veren özellik çıkartma yöntemi araştırılmıştır. Daha sonra iki farklı öğrenme yöntemi kullanılarak özellik vektörlerinden öz-vektörler elde edilmiştir. Bu öz-vektörlerden iki farklı bulanık sınıflayıcı gerçeklenmiştir.

QRS şekil bilgisinden özellik elde etme yönteminin diğer yöntemlere göre başarısız olduğu tespit edilmiştir. Bu yöntemin bazı derivasyonlarda başarılı olduğu ve bazı derivasyonlarda ise tamamen başarısız olduğu gözlenmiştir.

Frekans spektrumundan özellik çıkartma yöntemi en iyi sonuç veren iki yöntemden biridir. Daha iyi sonuç verebilmesi için analiz için kullanılan frekans bandının (dolayısıyla özellik vektörü boyutunun) büyütülmesi gerektiği tespit edilmiştir. Bu sonuç, beklenen bir sonuçtur. EKG işaretinde eğer büyük bir değişim varsa sınıflama sonucu iyidir. İşaret üzerinde küçük değişimler varsa sınıflama işleminde hatalar oluşur. Çünkü işaret üzerindeki küçük değişimler yüksek frekans bölgesinde değişime sebebiyet verir. Bu değişim, inceleme için baktığımız 0-30 Hz aralığının dışına etki eder. Ancak inceleme için alınan boyut artırıldığında, hesaplama zamanının uzadığı gözlenmiştir.

Çalışma içerisinde LPC yönteminin daha iyi sonuç

vermesi beklenmiştir. Ancak işaret üzerindeki küçük değişimlerin 6 adet katsayı ile temsil edilemeyeceği gözlenmiştir. Bu nedenle sınıflama başarısının düşük olduğu görülmüştür.

EKG işaretinin R tepesi etrafındaki genlikleri ile oluşturulan özellik vektörleriyle iyi sınıflama sonuçları elde edilmiştir. Bu yöntem R tepesinin EKG kaydı içerisindeki yerine son derece bağımlıdır. Ancak yöntemin, işaret üzerinde R tepesi etrafındaki ufak değişimlere duyarlı olduğu gözlenmiştir. R tepesi etrafında seçilen aralık büyük tutulursa, işaret üzerindeki alçak frekans bileşenli değişimlere de duyarlı olacaktır. Fakat bu durum, giriş vektörünün daha da büyümesine ve dolayısıyla sınıflama için daha fazla bellek alanı kullanılmasına sebebiyet verir.

İncelenen her iki öğrenme algoritması, normal QRS, sol dal bloğu ve yapay vuru şekil bozukluklarını temsil eden öz-vektörleri başarıyla bulmaktadır. Ancak erken karıncık kasılması (PVC tipi şekil bozukluğu), özellik uzayında birden fazla kümelenme gösterdiğinden, bulanık küme-ortalar algoritması tarafından öz-vektörü doğru olarak tespit edilemez. Bu nedenle, öz-vektörlerini bulanık küme-ortalar ile oluşturan bulanık en yakın model sınıflayıcısı, PVC tipi şekil bozukluğunu düşük bir başarı oranı ile belirleyebilmiştir. Öğrenme sırasında kullanılan Kohonen algoritmasının, şekil bozukluklarını temsil ederken birden fazla öz-vektör kullandığı için, özellikle PVC tipi şekil bozukluğunda, bulanık küme-ortalar algoritmasına nazaran daha avantajlı olduğu gözlenmiştir. Herbir sınıf için birden fazla öz-vektör kullanan bulanık K-NN sınıflayıcısı, en yakın model sınıflayıcısına nazaran daha iyi sonuç vermiştir.

Sınıflayıcılar içerisinde bulanık küme ortalar algoritması kullanılarak öz-vektörleri elde edilen K-en

yakın komşu sınıflayıcısının daha başarılı sonuçlar verdiği gözlenmiştir.

KAYNAKLAR

[1] MIT-BIH Arrhythmia Database. Available from Beth Israel Hospital, Biomedical Engineering Divison Room KB-26, 330 Brookline Ave., Boston, MA 02215.

[2] JIAPU, P., WILLIS, J.T., A Real-Time QRS Detection Algorithm, IEEE Tran. on Biomed. Eng., Vol. BME-32, pp. 230-236, March (1985).

[3] LEWIS, J.T.JR., Automated Cardiac Dysrhythmia Analysis, Proc. of the IEEE, Vol. 67, No. 9, pp. 1322-1337, Sept. (1979).

[4] KALITA, S., BELINA, J., Effective ECG Classification Using Single Layer Neural Network with Data Pre-Processing, Proc. of the 15th Ann. Int. Conf. IEEE Eng. in Med. and Bio. Soc., pp. 734-735, Oct. 28-33, (1993).

[5] MARQUES, J.P. De SA, A New ECG Classifier Based on Linear Prediction Techniques, Computers and Biomedical Research 19, pp. 213-223, (1986).

[6] SAKK, E., BELINA, J., A Method for Decreasing Neural Network Training Time as Applied to ECG Classification", 0-7803-0902-2/92 IEEE (1992).

[7] GATH, I., GEVA, B., Unsupervized Optimal Fuzzy Clustering, IEEE Tran. on Patt. Anal. and Mach. Intell., Vol. 11, No. 7, pp. 773-781, (1989).

[8] BEZDEK, J.C., Computing With Uncertainty, IEEE Comm. Magazine, pp. 24-36, Sept. (1992).

[9] KELLER, J.M., A Fuzzy K-Nearest Neighbor Algorithm, IEEE Tran. on Sys. Man and Cyb., Vol. SMC-15, No. 4, pp. 580-585, (1985).

[10] BEZDEK, J.C., Pattern Recognition With Fuzzy Objective Function Algorithms, Plenum Press, NEW YORK and LONDON, (1987).

[11] KORÜREK, M., Fizyoloji Ders Notları, İ.T.Ü., (1992).

[12] YAZGAN, E., Biyolojik İşaretlerin İşlenmesi ve Değerlendirilmesi, DPT projesi, İ.T.Ü., Haziran (1993).

[13] BÜYÜKÖZTÜRK, K., Pratisyen Hekimler İçin Elektrokardiografi, İstanbul Tıp Fakültesi İç Hast. Kürsüsü Kardioloji Bölümü.

[14] ATWOOD, S., STANTON, C., Introduction to Basic Dysrhythmias, The C.V. Mosby Company, (1990).

[15] SOPHOCLES, J.O., Optimal Signal Processing: An Introduction, Macmillan Publishing Company, Newyork, (1988).

EKLER

```
/*              EK A. PROGRAM CIKTILARI                          */

/*          Ozellik Vektorlerinin Elde Edilmesi                  */


#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>

#define K       5           /* komsuluk sayisi       */
#define MM      2           /* K-NN icin m sabiti    */
#define M       300         /* toplam data uzunlugu  */
#define N       250         /* pencere uzunlugu      */
#define S       4           /* Sınıf sayısı          */
#define Q       2           /* q sabiti              */
#define T       8           /* iterasyon sayisi      */

FILE      *dos1,*dos2,*dos3;
int       i,y,n,m,r,k1,k2,j,t,s,sa,sb,ya,fa,h;
int       iter,PS;
float     tut,usst;
float     ust,taban;
float     pay,paya,ab,ac;
char      cev;
int       k,n1,n2,k1,k2,k3,k4,p,pp,uu;
int       esomik,esomak,esamik,esamak,yesa;
float     uye1[S+9][N+3];
float     fil1[45],fil2[45],fil3[45],fil4[45],fil5[45],fil6[45];
float     fil7[6],fil8[40],fil9[10];
float     vek1[201],vek2[201];
float     m1,m2,m3,m4,m5,m6,m7,m8,bali;
char      ch,ch2,ch3,s1[20],s2[20];
int       sayac,ii;
long      gi,kk,yy,ll,tt;
double    rr,ff;

void            grafik();
void            filtre1();
void            vektor();
void            DFT();
void            latis();
void            goster();
```

```
void            Okuma();
void            Yazdir();
void            EKG();
void            Soru();


main    ()
{

printf("\n EKG Kayitini Yazin=");
gets(s1);
dos1=fopen(s1,"rb");

printf("\n Kullanilacak Tablo Dosyasi=");
gets(s1);
dos2=fopen(s1,"r");

printf("\n Cikis Dosyasinin Ismini Yazin=");
gets(s2);
dos3=fopen(s2,"w");

printf("\n Ozellik Vektorunun Boyutu=");
gets(s1);
PS=atoi(s1);

printf("\n Buyuk Harfle Incelenen Sekil bozuklugunu Yazin=");
ch2=getchar();

p=0;l1=0;pp=0;m7=0;

grafik();

for(m=1;m<=400;m++) putpixel(10,m,10);
for(m=1;m<=260;m++) putpixel(m,200,10);

outtextxy(15,5,"EKG Isareti");outtextxy(0,203,"0");

randomize();

bali=0;gi=0;
tt=0;
while((!feof (dos2)) && (!kbhit()) )
      {
tt=tt+1;
p=p+1;

fscanf(dos2,"%*s%ld%s%d%*d%*d",&kk,&ch3,&m);
m1=kk;m1=m1/3;m1=m1+0.5;kk=m1;kk=kk*9;


if((tt<4) || (ch2==ch3))
```

```
{

if(tt>=4)
{
kk=kk-3*100;
k=fseek (dos1,kk,SEEK_SET);
if(k!=0) printf("\n HATA");
}
      for(ii=1;ii<=350;ii++)
      {
      ll=ll+1;
      kk=kk+1;
      Okuma();
      m2=m3;

            for(m=250;m>=2;m--)
               {
               uye1[2][m]=uye1[2][m-1];
               uye1[3][m]=uye1[3][m-1];
               uye1[9][m]=uye1[9][m-1];
               }


/* R Tepesinin Tespiti                              */
            filtre1();
/*******************************************/

            uye1[2][1]=m2*100;
            uye1[9][1]=fil7[1]/10;


if ((ll>400) && (uye1[3][160]>0))
   {
     for(h=0;h<=250;h++)
            {
            n2=uye1[1][h];
            putpixel(10+h,(200-n2),0);
            n2=uye1[2][h];
            putpixel(10+h,(200-n2),12);
            uye1[1][h]=n2;
            putpixel(10+h,200,10);
            }

      for(m=50;m<=250;m++)      putpixel(10+178,m,1);
      if (PS==4)      vektor();
      if (PS==50)     DFT();
      if (PS==40)     latis();
      if (PS==100)    EKG();
      goster();

if(tt>=4)
```

```
      {
      sayac=sayac+1;
      Soru();
      if(m1==1)
            {
            for(m=1;m<=PS;m++)
               {
               fprintf(dos3,"%f ",vek1[m]);
               } fprintf(dos3,"\n");
            }
      }
   }
   if(kbhit()) goto cik1;
   }

}
cik1:      i=0;
      }

getch();
closegraph();
fclose(dos1);
fclose(dos2);
fclose(dos3);
}

/***************   ALT PROGRAMLAR  ********************/

void Soru()
{
char ch1;

      for(m=0;m<=300;m++){for(h=450;h<=460;h++)putpixel(m,h,0);}
      itoa(sayac,s2,10);outtextxy(0,450,s2);
      outtextxy(20,450,". Vektor Kaydedilsin mi? (e/h)=");
tekrar:
      ch1=getch();
      switch (ch1) {

      case 'e': m1=1;break;
      case 'h': m1=0;sayac=sayac-1;break;
                }
}

void EKG()
{
for(m=-50;m<=50;m++)      vek1[m+51]=uye1[2][m+178];
}

void Yazdir()
{
```

```
      outtextxy(265,200,"t");

      for(m=0;m<=60;m++){for(h=400;h<=410;h++) putpixel(m,h,0);}
      ltoa(kk,s1,10);
      outtextxy(0,400,s1);

      outtextxy(15,100,"1.0 mV");outtextxy(15,300,"-1.0 mV");
}
void Okuma()
{

      k1=0;k1=getc(dos1);
      k2=0;k2=getc(dos1);
      k3=0;
      k3=k2<<8;k3=k3 & 0x0F00;
      k1=k1 | k3;
      k4=0;k4=getc(dos1);
      k3=0;
      k3=k2<<4;k3=k3 & 0x0F00;
      k4=k4 | k3;

    if((k1>1024) && (k1<2048)) {m3=k1;m3=m3-1024;m3=m3*0.005;}
    if((k1>0) && (k1<1024))    {m3=k1;m3=m3*0.005;m3=m3-5.120;}
    if((k1>2048) && (k1<4095))

      {m3=k1;m3=m3-2048;m3=m3*0.005;m3=m3-15.360;}

    if((k4>1024) && (k4<2048)) {m2=k4;m2=m2-1024;m2=m2*0.005;}
    if((k4>0) && (k4<1024))    {m2=k4;m2=m2*0.005;m2=m2-5.120;}
    if((k4>2048) && (k4<4095))

      {m2=k4;m2=m2-2048;m2=m2*0.005;m2=m2-15.360;}

}


void filtre1()
{
int  mm,mn;
float     m5,m9;

pp=pp+1;

/******************  YGF **********************/

      for(m=40;m>=2;m--)  fil5[m]=fil5[m-1];
      fil5[1]=m2+1.951*fil5[2]-0.952*fil5[3];
      m9=0.976*fil5[1]-1.952*fil5[2]+0.976*fil5[3];
      m2=m9/2;
```

```
/******************  BGF  *********************/

     for(m=45;m>=2;m--)  fil1[m]=fil1[m-1];fil1[1]=m2;
     for(m=45;m>=2;m--)  fil2[m]=fil2[m-1];
     fil2[1]=1.849*fil2[2]-0.933*fil2[3]
          +0.0013*fil1[1]-0.0013*fil1[3];
     m1=1000*fil2[1];


/******************  AGF  *********************/

     for(m=40;m>=2;m--)  fil4[m]=fil4[m-1];
     fil4[1]=2*fil4[2]-fil4[3]+fil1[1]-2*fil1[11]+fil1[21];


/*****************  EGIM **********************/

     for(m=6;m>=2;m--)   fil7[m]=fil7[m-1];
     fil7[1]=5*(1*fil4[1]+2*fil4[2]-2*fil4[4]-1*fil4[5]);


/***************** Kare Alici ******************/

     m5=m1*m1;


/*************  Integral Alici  ***************/

     for(m=30;m>=2;m--)  fil3[m]=fil3[m-1];fil3[1]=m5;
     m1=0;
     for(m=1;m<=30;m++)  m1=m1+fil3[m];
     m5=m1/30;

/***************** Geciktirme ******************/

     m1=fil4[13]/1;

/***************** Carpma islemi ***************/

     m5=m5*m1;

         /*****************************/

uye1[3][1]=0;
uu=uu+1;
if(uu>200)      {uu=1;m7=0;}
if(m5>m7) {
          uye1[3][uu]=0;
          m7=m5;uu=1;uye1[3][1]=10;
          }
}
```

```
void vektor()
{
int  o,o1,p,u,l,somik,somak,samik,samak;
int  w1,w2;
float     somin,somax,samin,samax;

l=0;
randomize();

if(uye1[2][178]<0)
        {
        for(m=0;m<=250;m++)    uye1[9][m]=-uye1[9][m];
        }
somin=10000;
for(m=178;m>=100;m--)
   {
   if(uye1[9][m]<somin)  {somin=uye1[9][m];somik=m;}
   }

samin=0;
for(m=178;m<=220;m++)
   {
   if(uye1[9][m]>samin)  {samin=uye1[9][m];samik=m;}
   }

   o=0;o1=0;
   for(m=somik;m>=0;m--)
      {
      m1=-uye1[9][m];
      if((o==0) && (m1<0.4*(-somin)))    {w1=m;o=1;}
      if((o1==0) && (o==1) && (m1<=0))
          {
          if((somik-m)<20)     {o1=1;w1=m;}
          }
      }
   somik=w1+10;

ck1:
   o=0;o1=0;
   for(m=samik;m<=250;m++)
      {
      m1=uye1[9][m];
      if((o1==0) && (uye1[9][m]<0.4*samin)) {w2=m;o1=1;}
      if((o1==1) && (o==0) && (m1<=0))
          {
          if((m-samik)<20)     {o=1;w2=m;}

          }

      }
```

```
ck2:
       samik=w2+10;

gi=gi+1;o=0;
       for(m=samik+1;m<=samik+50;m++)
if((gi<=4) && (o==0) && (uye1[9][m]>=0))
      {
      o=1;bali=bali*(gi-1);bali=bali+uye1[2][m+7];
      bali=bali/gi;o1=m;
      }

for(m=100;m<=210;m++)     {putpixel(10+esamik,m,0);}
for(m=100;m<=210;m++)     {putpixel(10+samik,m,1);}
esamik=samik;
for(m=100;m<=210;m++)     {putpixel(10+esomik,m,0);}
for(m=100;m<=210;m++)     {putpixel(10+somik,m,1);}
esomik=somik;
for(m=100;m<=210;m++)     putpixel(10,m,10);

vek1[1]=samik-somik;

if(uye1[2][178]>0) {m1=samin;if(somin<m1) m1=somin;}
if(uye1[2][178]<0) {m1=samin;if(somin>m1) m1=somin;}
vek1[2]=uye1[2][178]-m1;

if(uye1[2][178]>0)  m1=m1+(vek1[2]/2);
if(uye1[2][178]<0)  m1=m1-(vek1[2]/2);
vek1[3]=10*(bali-m1);

m1=0;
for(m=somik;m<=samik;m++)
    {
    if((uye1[2][178]>0) && (uye1[2][m]>bali)) m1=m1+uye1[2][m];
    if((uye1[2][178]<0) && (uye1[2][m]<bali)) m1=m1+uye1[2][m];
    }
vek1[4]=m1/10;

for(m=5;m<=18;m++)  vek1[m]=0;

}


void goster()

{

int  o,p,u,l,o1,kk1;
float     mm;

mm=320/PS;kk1=mm;
```

```
l=0;
for(o=1;o<=PS;o++)
   {

outtextxy(315,0,"Ozellik Vektorunun Histogrami");

for(m=1;m<=350;m++) putpixel(310,m+50,3);
for(m=1;m<=340;m++) putpixel(300+m,240,3);

   o1=o;

   for(p=1;p<=kk1;p++)
      {
      l=l+1;

       if(vek2[o]>0)
     {
     for(u=1;u<=vek2[o];u++)    putpixel(310+l,240-u,0);
     }
      if(vek2[o]<0)
     {
     for(u=1;u<=abs(vek2[o]);u++) putpixel(310+l,240+u,0);
     }

     if(o1==16)      o1=19;

      if(vek1[o]>0)
     {
     for(u=1;u<=vek1[o];u++)    putpixel(310+l,240-u,o1);
     }
      if(vek1[o]<0)
     {
     for(u=1;u<=abs(vek1[o]);u++)  putpixel(310+l,240+u,o1);
     }

      }
   vek2[o]=vek1[o];
   }

}

void DFT()
{
int  o,p,o1,p1;
float     h1,h2,h3,h4;

h4=0.0001;
for(o=0;o<250;o++)
   {
   o1=o-125;
   h1=h2=0;
```

```
    for(p=0;p<250;p++)
        {
        p1=p-125;
        h1=h1+uye1[2][p+1]*cos(2*M_PI*o*p/250);
        h2=h2-uye1[2][p+1]*sin(2*M_PI*o*p/250);
        }
    h3=(h1*h1+h2*h2);
    uye1[10][o+1]=(pow(h3,0.5))/1;

    uye1[2][251]=uye1[10][1]/250;

    }

for(m=1;m<=PS;m++)  vek1[m]=0.1*uye1[10][m];

}

void latis()
{
float      m1,m2,m3,m4,m5;
int  j,l;

for(l=1;l<=250;l++)
    {
    uye1[4][l]=uye1[2][l];
    uye1[7][l]=uye1[2][l];
    }

for(l=1;l<=PS;l++)
    {
    m3=m4=0;
    for(j=2;j<=250;j++)
        {
        m3=m3+uye1[7][j]*uye1[4][j-1];
        m4=m4+uye1[4][j-1]*uye1[4][j-1];
        }m4=m4+uye1[4][250]*uye1[4][250];m3=m3/250;m4=m4/250;
    m5=m3/m4;
    for(j=2;j<=250;j++)
      {
      uye1[5][j]=uye1[7][j]-m5*uye1[4][j-1];
      uye1[6][j]=uye1[4][j-1]-m5*uye1[7][j];
      }
    for(j=1;j<=250;j++)
      {
      uye1[7][j]=uye1[5][j];
      uye1[4][j]=uye1[6][j];
      }
    vek1[l]=m5*200;
    }

}
```

```
void grafik()

 {   int i,j;
     int gdriver=DETECT,gmode,errorcode;
     initgraph(&gdriver,&gmode,"");
     errorcode=graphresult();
if(errorcode != grOk)
  {printf("Graphics error: %s\n",grapherrormsg(errorcode));
   printf(" Bir tusa basınız ");
   getch();
   exit(1);
  }
}
```

```
/*          OZ VEKTORLERIN ELDE EDILMESI          */


#include<stdio.h>
#include<math.h>
#include<string.h>
#include<dos.h>
#include<alloc.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>


#define    K    4          /* komsuluk sayisi 5   */
#define    S    4          /* sınıf sayısı        */
#define    Q    2          /* q sabiti            */
#define    T   50          /* iterasyon sayisi    */

FILE       *dos1,*dos2,*dos3;
int        i,y,n,m,r,k,k1,k2,j,t,s,sa,sb,ya,fa,h;
int        iter;
float      mt;
float      mesafe[S+2][S+1][15*S+1],veri,payda;
float      kesir[S+2][S*15+1],u[S+1][S+1][S*15+1],ust;
float      pay,paya,ac;
float      xa[1][1][1],ort[S*2+1][100+2];;

char       cev;
int        n1,n2,kk,O,oo;
char       s1[10];
float      diz[S*15+1][100+2],uye1[S*2+1][S*15+1];
float      buf[3][K+2], m1,m2;

void            Egitim_kume_oku();
void            Test_kume_oku();
void            orta();
void            Fuzzy_c_mean();
void            Fuzzy_Kohonen_og();
void            Fuzzy_En_Yakin_Si();
void            Fuzzy_K_Komsu_Si();
void            Uyelik_bul();

float      far  *p1,*p3,*p4;
int        PS;

main     ()
{

printf("\n Giris Vektorunun Boyutu=");
gets(s1);
PS=atoi(s1);
```

```
if((p1=farcalloc(100*(PS+5),sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p4=farcalloc(100*(PS+5),sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p3=farcalloc(700,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}


printf("\n Cikis Dosyasinin Ismini Yaz=");
gets(s1);
dos3=fopen(s1,"w");

printf("\n Oz Vektorlerin Sayisini Yazin=");
gets(s1);
O=atoi(s1);

printf("\n Bulanik C_Mean Ogrenme icin   1");
printf("\n Egitim Kumesini Direk Alma    2");
printf("\n Bulanik Kohonen Ogrenme icin  3");
printf("\n ---------------------------  =");
gets(s1);
oo=atoi(s1);

ust=pow((Q-1),-1);
n=15;
/************ Bulanik En Yakin Siniflayici ****************/

if (oo==1)
     {
     Egitim_kume_oku();
     orta();
     Fuzzy_c_mean();
     Test_kume_oku();
     n=n*S;
     Fuzzy_En_Yakin_Si();
     }


/* Bulanik Kume-ortalar ile egitilen K. Komsu Siniflayici */

if (oo==2)
     {
     Egitim_kume_oku();
     orta();
     Fuzzy_c_mean();

     Test_kume_oku();
     n=S*n;
     for(m=1;m<=n;m++)
          for(j=1;j<=PS;j++)
               diz[m][j]=*(p4+(m-1)*101+j-1);
     Uyelik_bul();

     Fuzzy_K_Komsu_Si();
     }
```

```
/** Bulanik Kohonen ile egitilen K. Komsu Siniflayici */

if (oo==3)
     {
     Egitim_kume_oku();
     orta();
     Test_kume_oku();
     n=S*n;

     Fuzzy_Kohonen_og();

     Fuzzy_K_Komsu_Si();
     }
/*************************************************************/

son:
     farfree(p1);
     farfree(p4);
     farfree(p3);

     printf("\ Program Sonlandi");
     getch();
     return 0;

     fclose(dos3);

}


/*****************    ALT PROGRAMLAR      *****************/

void Fuzzy_Kohonen_og()
{
float     m1,m2,m3;

m1=n/S;
n2=0;n1=1;
for(i=1;i<=n;i++)
   {
   n2=n2+1;
   if(n2>m1) {n1=n1+1;n2=1;}
   for(j=1;j<=PS;j++)
      {
      diz[i][j]=*(p1+(n1-1)*16*(PS+1)+(n2-1)*(PS+1)+j-1);
      }
   }

m1=n/S;n2=m1;k=1;
for(i=1;i<=S;i++)
   {
   for(j=1;j<=2;j++)
      {
      n1=(rand() %n2)+1;
      for(m=1;m<=PS;m++)
      ort[k][m]=*(p1+(i-1)*16*(PS+1)+(n1-1)*(PS+1)+m-1);
```

```
        uye1[i][k]=1;
        k=k+1;
        }
    }


for (iter=1; iter<=T; iter++)
{

m1=T;
mt=2-iter/m1;
ust=2*mt;

for(k=1;k<=n;k++)
    {
    for(i=1;i<=2*S;i++)
         {
         m1=0.01;
         for(j=1;j<=PS;j++)
          {
          m2=(diz[k][j]-ort[i][j]);
          m1=m1+(m2*m2);
          }
         m3=0.3/m1;
         m1=pow (m3, ust);
         *(p3+(i-1)*61+k-1)=m1;
         }
    }

for(j=1;j<=PS;j++)
{
 for(i=1;i<=2*S;i++)
    {
    m2=0.00001;mt=0.00001;
    for(k=1;k<=n;k++)
      {
      m1=(diz[k][j]-ort[i][j])* *(p3+(i-1)*61+k-1);
      m2=m2+m1;
      mt=mt+*(p3+(i-1)*61+k-1);
      }
    ort[i][j]=ort[i][j]+(m2/mt);
    }
}

}
yyy:i=0;

for(i=1;i<=2*S;i++)
    {
    for(j=1;j<=PS;j++)
      {
      diz[i][j]=ort[i][j];
      }
    }
}
```

```
void Uyelik_bul()
{
m1=n/S;
for(i=1;i<=S;i++)
    {
    n2=1;n1=0;
    for(j=1;j<=n;j++)
        {
        n1=n1+1;if(n1>m1) {n2=n2+1;n1=1;}
        uye1[i][j]=u[i][n2][n1];
        }
    }
}

void Fuzzy_K_Komsu_Si()
{

k1=O/S;
k=1;
for (h=1;h<=S;h++)
      {
      for(m=1;m<=k1;m++)
         {
         for (j=1;j<=PS;j++)
                {
                fprintf(dos3,"%f ",diz[k][j]);
                }
         for(j=1;j<=S;j++)
                {
                fprintf(dos3,"%f ",uye1[j][k]);
                }
         fprintf(dos3,"\n");
         k=k+1;
         }
      }

for(m=1;m<=n;m++)
  {
  for(i=1;i<=K;i++) buf[1][i]=10000000;
      for(h=1;h<=O;h++)
      {
      m1=0.0001;
      for(j=1;j<=PS;j++)
         {
         m1=m1+(*(p4+(m-1)*101+j-1)-diz[h][j])
            *(*(p4+(m-1)*101+j-1)-diz[h][j]);
         }
      m2=0;
      for(i=1;i<=K;i++)
         {
if( (m1<buf[1][i]) && (m2==0) )
                    {
                     m2=1;
                     for(j=K;j>=i;j--)
                      {
```

```
                    buf[1][j+1]=buf[1][j];
                    buf[2][j+1]=buf[2][j];
                    }
                   buf[1][i]=m1;buf[2][i]=h;
                    }
          }
atla:       n1=0;
      }
 getch();

  m1=0.001;
  for(i=1;i<=K;i++)  m1=m1+(1/buf[1][i]);

  for(h=1;h<=S;h++)
     {
     m2=0.001;
     for(i=1;i<=K;i++)
     {
     n1=buf[2][i];
     m2=m2+(uye1[h][n1]/buf[1][i]);
     }
     printf("\n %d. vektorun %d. sinifa uyelik=%f",m,h,m2/m1);
     }
  }
}

void Fuzzy_En_Yakin_Si()
{
for (h=1;h<=S;h++)
     {
     for (j=1;j<=PS;j++)
          {
          fprintf(dos3,"%f ",ort[h][j]);
          }fprintf(dos3,"\n");
     }

for (h=1; h<=S; h++)
  { for (i=1; i<=n; i++)
    { mesafe[S+1][h][i]=0.000001;
      for (j=1; j<=PS; j++)
      mesafe[S+1][h][i]=mesafe[S+1][h][i]
                +pow((*(p4+(i-1)*101+j-1)-ort[h][j]),2);
    }
  }

for (j=1; j<=n; j++)
  {
  kesir[S+1][j]=0.000001;
  for (m=1; m<=S; m++)
  kesir[S+1][j]=kesir[S+1][j]
              +pow( (pow(mesafe[S+1][m][j],-1)), ust);

  }
```

```
for (m=1; m<=S; m++)
  {
  for (j=1; j<=n; j++)
       {
       u[S+1][m][j]=0.0;
       ac=pow ( (pow( (mesafe[S+1][m][j]), -1 )), ust);
       u[S+1][m][j]=ac * pow( (kesir[S+1][j]), -1 );
       kk=10*u[S+1][m][j];
       printf("%d ",kk);
       }
  printf("\n");getch();
  }
}


void Test_kume_oku()
{
n2=1;
for(n1=1;n1<=S;n1++)
   {
   for(i=1;i<=n;i++)
     {
     for(j=1;j<=PS;j++)
        {

*(p4+(n2-1)*101+j-1)=*(p1+(n1-1)*16*(PS+1)+(i-1)*(PS+1)+j-1);
        }
     n2=n2+1;
     }
   }
}


void Fuzzy_c_mean()
{


for (iter=1; iter<=T; iter++)
{
  for (m=1; m<=S; m++)
    {
    for (h=1; h<=S; h++)
      {
      for (i=1; i<=n; i++)
        {
        mesafe[m][h][i]=0.000001;
        for (j=1; j<=PS; j++)
        mesafe[m][h][i]=mesafe[m][h][i]

+pow((*(p1+(h-1)*16*(PS+1)+(i-1)*(PS+1)+j-1)-ort[m][j]),2);
        }
      }
    }
```

```
/* j vektörünün sentroidlere uzaklıkları toplamı  */

for (h=1; h<=S; h++)
   {

   for (j=1; j<=n; j++)
     {
     kesir[h][j]=0.0;
     for (m=1; m<=S; m++)
     kesir[h][j]=kesir[h][j]
          +pow( (pow(mesafe[m][h][j],-1)), ust);
     }
   }

for (h=1; h<=S; h++)
  {
  for (m=1; m<=S; m++)
    {
    for (j=1; j<=n; j++)
      {
      ac=pow ( (pow( (mesafe[h][m][j]), -1 )), ust);
      u[h][m][j]=ac * pow( (kesir[m][j]), -1 );
      }
    }
  }



/*   Vi'nın hesabı        */

for (m=1; m<=S; m++)
  {
  for (i=1; i<=PS; i++)
    {
    pay=0.0;  paya=0.0;
    for (h=1; h<=S; h++)
      {
      for (j=1; j<=n; j++)
        {
        pay =pay+ pow( u[m][h][j], Q);
        paya =paya+

pow(u[m][h][j],Q)**(p1+(h-1)*16*(PS+1)+(j-1)*(PS+1)+i-1);
        }
      }
    ort[m][i] = paya*pow(pay,-1);
    }
  }

}
}
```

```
void orta()
{

for (i=1; i<=S; i++)
{ for (m=1; m<=PS; m++)
  {
     ort[i][m]=0;
     for(j=1;j<=n;j++)
     ort[i][m]=ort[i][m]+
            *(p1+(i-1)*16*(PS+1)+(j-1)*(PS+1)+m-1);
     ort[i][m]=ort[i][m]/n;
   }
}

}


void Egitim_kume_oku()
{
float     far *p2;

if((p2=farcalloc(PS*15+10,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}

for(n1=1;n1<=S;n1++)
  {
  printf("\n %d. Sinifin Dosya Ismi=",n1);
  gets(s1);
  dos1=fopen(s1,"rt");i=0;
          for(j=1;j<=n;j++)
             {
              for(m=1;m<=PS;m++)
                 {
                 fscanf(dos1,"%f ",&veri);
                 i=i+1;*(p2+i-1)=veri;
                 }
              }
  for (i=1; i<=n; i++) for (j=1; j<=PS; j++)

*(p1+(n1-1)*16*(PS+1)+(i-1)*(PS+1)+(j-1))=*(p2+PS*(i-1)+j-1);

  fclose(dos1);
  }

farfree(p2);
}
```

```
/*          EKG ISARETLERININ SINIFLANDIRILMASI          */


#include<dos.h>
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<alloc.h>
#include<process.h>
#include<graphics.h>
#include<conio.h>


#define   K     3       /* komsuluk sayisi       */
#define   MM    2       /* K-NN icin m sabiti    */
#define   M     300     /* toplam data uzunluğu  */
#define   N     250     /* Pencere Uzunlugu      */
#define   S     4       /* sınıf sayısı          */
#define   Q     2       /* q sabiti              */
#define   T     8       /* iterasyon sayisi      */


int       esomik,esamik,uu,O,PS;
float     ust,m2,m3,m7,bali;
long      ll;


void      vektor              (float *p4,float *p6);
void      DFT                 (float *p4,float *p6);
void      latis               (float *p4,float *p6);
void      goster              (float *p4,float *p5);
void      Okuma               (FILE *dos1);
void      Yazdir              ();
void      EKG                 (float *p4,float *p6);
void      Fuzzy_En_Yakin_Si   (float *p1,float *p2);
void      Fuzzy_K_Komsu_Si    (float *p1,float *p2,float *p3);
void      Uyelik              (int k1,int k2,int k3,int k4);
void      filtre1             (float *p6,float *f1,float *f2,
                               float *f3,float *f4,float *f5
                               ,float *f7);

main      ()
{

FILE      *dos1,*dos2,*dos3;
char      s1[20],s2[20];
int       k,n2,i,y,n,m,r,j,t,s,h,hh,p;
int       k1,k2,k3,k4;
int       BS,ii;
float     m1;
long      tt;
```

```
float      far  *p1,*p2,*p3,*p4,*p5,*p6;
float      far  *f1,*f2,*f3,*f4,*f5,*f7;


if((p1=farcalloc(300,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p2=farcalloc(5000,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p3=farcalloc(800,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p4=farcalloc(80,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p5=farcalloc(80,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((p6=farcalloc(5000,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}

if((f1=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((f2=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((f3=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((f4=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((f5=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}
if((f7=farcalloc(50,sizeof(float)))==NULL)
     {printf("\n Not Enough Memory");exit(1);}

ust=pow((Q-1),-1);
p=0;l1=0;m7=0;

printf("\n EKG Kayitini Yazin=");
gets(s1);
dos1=fopen(s1,"rb");

printf("\nQRS Sekil Bilgisinden Olusturulan Vektor  Icin   4");
printf("\nFrekans Spekturumundan Olusturulan Vektor Icin  30");
printf("\nAR katsayilarindan Olusturulan Vektor     Icin   6");
printf("\nEKG Isaretinden  Olusturulan Vektor       Icin  48");
printf("\nOzellik Vektorunun Tipini Yazin                 =  ");
gets(s2);
PS=atoi(s2);

printf("\n Oz Vektorlerin Yazili Oldugu Dosya Ismi=");
gets(s2);
dos3=fopen(s2,"r");


printf("\n Bulanik En Yakin Model Siniflayicisi Icin  1");
printf("\n Bulanik K Komsu Siniflayicisi Icin         2");
gets(s2);
```

```
BS=atoi(s2);

if(BS==1)
      {
      for(m=1;m<=S;m++)
         {
         for(i=1;i<=PS;i++)
            {
            fscanf(dos3,"%f ",&m1);
            *(p2+(m-1)*81+(i-1))=m1;
            }
         }
      }


if(BS==2)
      {
      printf("\n Oz Vektorlerin Sayisini Yazin=");
      gets(s1);
      O=atoi(s1);
      k1=O/S;

      k=1;
      for(m=1;m<=S;m++)
         {
         for(j=1;j<=k1;j++)
            {
            for(i=1;i<=PS;i++)
              {
              fscanf(dos3,"%f ",&m1);
              *(p2+(k-1)*81+(i-1))=m1;
              }
            for(i=1;i<=S;i++)
              {
              fscanf(dos3,"%f ",&m1);
              *(p3+(i-1)*61+(k-1))=m1;
              }
            k=k+1;
            }
         }
      }


grafik();


for(m=1;m<=400;m++) putpixel(10,m,10);
for(m=1;m<=260;m++) putpixel(m,200,10);
outtextxy(15,5,"EKG Isareti");outtextxy(0,203,"0");


bali=0;
tt=0;
```

```
   while((!feof (dos1)) && (!kbhit()) )
     {

     tt=tt+1;
     p=p+1;


  for(ii=1;ii<=350;ii++)
     {
     ll=ll+1;
     Okuma(dos1);
     m2=m3;
          for(m=250;m>=2;m--)
             {
             *(p6+252+m-1)=*(p6+252+m-2);
             *(p6+504+m-1)=*(p6+504+m-2);
             *(p6+2268+m-1)=*(p6+2268+m-2);
             }


/*********** R Tepesinin Tespiti ***************/
          filtre1(p6,f1,f2,f3,f4,f5,f7);

/*************************************************/

          *(p6+252)=m2*100;
          *(p6+2268)=*(f7)/10;

if ((ll>400) && (*(p6+504+159)>0))
     {
     for(h=1;h<=250;h++)
          {
          n2=*(p6+h-1);
          putpixel(10+h,(200-n2),0);
          n2=*(p6+252+h-1);
          putpixel(10+h,(200-n2),12);
          *(p6+h-1)=n2;
          putpixel(10+h,200,10);
          }
     for(m=50;m<=250;m++)      putpixel(10+178,m,1);
     if   (PS==4 )             vektor(p4,p6);
     if   (PS==30)             DFT(p4,p6);
     if   (PS==6 )             latis(p4,p6);
     if   (PS==48)             EKG(p4,p6);
     goster(p4,p5);

if(tt>=4)
      {
      for(m=1;m<=PS;m++) *(p1+m-1)=*(p4+m-1);
      if(BS==1) Fuzzy_En_Yakin_Si(p1,p2);
      if(BS==2) Fuzzy_K_Komsu_Si(p1,p2,p3);
      k1=10*(*(p1+0));k2=10*(*(p1+1));
      k3=10*(*(p1+2));k4=10*(*(p1+3));
      Uyelik(k1,k2,k3,k4);
      }
```

```
       }
      if(kbhit())     goto cik1;
      }
cik1:    i=0;
      }
son:
farfree(p1);farfree(p2);farfree(p3);
farfree(p4);farfree(p5);farfree(p6);
farfree(f1);farfree(f2);farfree(f3);
farfree(f4);farfree(f5);farfree(f7);

getch();
closegraph();
fclose(dos1);
fclose(dos2);
fclose(dos3);
}


/***************  ALT PROGRAMLAR  ******************/

void EKG(float *p4,float *p6)
{

int  m;
    for(m=-24;m<=24;m++)  *(p4+m+24)=*(p6+252+(m+177));
}


void Yazdir()
{

int  m,h;
char s1[20];
       outtextxy(265,200,"t");
       for(m=0;m<=60;m++){for(h=400;h<=410;h++)
                 putpixel(m,h,0);}
       ltoa(ll,s1,10);
       outtextxy(0,400,s1);
       outtextxy(15,100,"1.0 mV");outtextxy(15,300,"-1.0 mV");
}


void Uyelik(int k1,int k2,int k3,int k4)
{

int m,h;
char s1[20];
for(m=0;m<=200;m++){for(h=450;h<=460;h++) putpixel(m,h,0);}
itoa(k1,s1,10);outtextxy(10,450,"N=");outtextxy(30,450,s1);
itoa(k2,s1,10);outtextxy(60,450,"P=");outtextxy(80,450,s1);
itoa(k3,s1,10);outtextxy(110,450,"L=");outtextxy(130,450,s1);
itoa(k4,s1,10);outtextxy(160,450,"V=");outtextxy(180,450,s1);

}
```

```
void Okuma(FILE *dos1)
{

int  k1,k2,k3,k4;

     k1=0;k1=getc(dos1);
     k2=0;k2=getc(dos1);
     k3=0;
     k3=k2<<8;k3=k3 & 0x0F00;
     k1=k1 ¦ k3;
     k4=0;k4=getc(dos1);
     k3=0;
     k3=k2<<4;k3=k3 & 0x0F00;
     k4=k4 ¦ k3;

   if((k1>1024) && (k1<2048)) {m3=k1;m3=m3-1024;m3=m3*0.005;}
   if((k1>0) && (k1<1024))    {m3=k1;m3=m3*0.005;m3=m3-5.120;}
   if((k1>2048) && (k1<4095))

          {m3=k1;m3=m3-2048;m3=m3*0.005;m3=m3-15.360;}


   if((k4>1024) && (k4<2048)) {m2=k4;m2=m2-1024;m2=m2*0.005;}
   if((k4>0) && (k4<1024))    {m2=k4;m2=m2*0.005;m2=m2-5.120;}
   if((k4>2048) && (k4<4095))

          {m2=k4;m2=m2-2048;m2=m2*0.005;m2=m2-15.360;}

}


void filtre1(float *p6,float *f1,float *f2,float *f3
             ,float *f4,float *f5,float *f7)
{

int       mm,mn,m;
float     m1,m5,m9;


/*********************  YGF  ***************************/
     for(m=40;m>=2;m--)  *(f5+m-1)=*(f5+m-2);
     *(f5)=m2+1.951**(f5+1)-0.952**(f5+2);
     m9=0.976**(f5)-1.952**(f5+1)+0.976**(f5+2);
     m2=m9/2;



/*********************  BGF  ***************************/
     for(m=45;m>=2;m--)  *(f1+m-1)=*(f1+m-2);*(f1)=m2;
     for(m=45;m>=2;m--)  *(f2+m-1)=*(f2+m-2);
     *(f2)=1.849**(f2+1)-0.933**(f2+2)
          +0.0013**(f1)-0.0013**(f1+2);
     m1=1000**(f2);
```

```
/*********************  AGF  ************************/
     for(m=40;m>=2;m--)  *(f4+m-1)=*(f4+m-2);
     *(f4)=2**(f4+1)-*(f4+2)+*(f1)-2**(f1+10)+*(f1+20);


/*********************  Egim ************************/
     for(m=6;m>=2;m--)   *(f7+m-1)=*(f7+m-2);
     *(f7)=5*(1**(f4)+2**(f4+1)-2**(f4+3)-1**(f4+4));



/***************** Kare Alma Islemi *****************/
     m5=m1*m1;


/************** Integral Alma Islemi  ***************/
     for(m=30;m>=2;m--)  *(f3+m-1)=*(f3+m-2);*(f3)=m5;
     m1=0;
     for(m=1;m<=30;m++)  m1=m1+*(f3+m-1);
     m5=m1/30;


/****************** Geciktirme **********************/
     m1=*(f4+12)/1;


/*************** Carpma Islemi **********************/
     m5=m5*m1;

/************************************************/

*(p6+504)=0;
uu=uu+1;
if(uu>200)      {uu=1;m7=0;}
if(m5>m7) {
          *(p6+504+uu-1)=0;
          m7=m5;uu=1;*(p6+504)=10;
          }
}


void vektor(float *p4,float *p6)
{

int       m,o,o1,p,u,l,somik,somak,samik,samak;
int       w1,w2;
float     m1,somin,somax,samin,samax;

l=0;
randomize();

if(*(p6+252+177)<0)
     {
     for(m=0;m<=250;m++) *(p6+2268+m-1)=-*(p6+2268+m-1);
     }
```

```
somin=10000;
for(m=178;m>=100;m--)
   {
   if(*(p6+2268+m-1)<somin) {somin=*(p6+2268+m-1);somik=m;}
   }

samin=0;
for(m=178;m<=220;m++)
   {
   if(*(p6+2268+m-1)>samin) {samin=*(p6+2268+m-1);samik=m;}
   }

   o=0;o1=0;
   for(m=somik;m>=0;m--)
      {
      m1=-*(p6+2268+m-1);
      if((o==0) && (m1<0.4*(-somin)))    {w1=m;o=1;}
      if((o1==0) && (o==1) && (m1<=0))
          {
          if((somik-m)<20)     {o1=1;w1=m;}
          }
      }
   somik=w1+10;

ck1:
   o=0;o1=0;
   for(m=samik;m<=250;m++)
      {
      m1=*(p6+2268+m-1);
      if((o1==0) && (*(p6+2268+m-1)<0.4*samin)){w2=m;o1=1;}
      if((o1==1) && (o==0) && (m1<=0))
          {
          if((m-samik)<20)     {o=1;w2=m;}
          }
      }
ck2:
      samik=w2+10;

o=0;
      for(m=samik+1;m<=samik+50;m++)
if((ll<=4) && (o==0) && (*(p6+2268+m-1)>=0))
     {
     o=1;bali=bali*(ll-1);bali=bali+*(p6+252+m+6);
     bali=bali/ll;o1=m;
     }

for(m=100;m<=210;m++)     {putpixel(10+esamik,m,0);}
for(m=100;m<=210;m++)     {putpixel(10+samik,m,1);}

esamik=samik;

for(m=100;m<=210;m++)     {putpixel(10+esomik,m,0);}
for(m=100;m<=210;m++)     {putpixel(10+somik,m,1);}

esomik=somik;
```

```
for(m=100;m<=210;m++)     putpixel(10,m,10);

*(p4)=samik-somik;

if(*(p6+252+177)>0) {m1=samin;if(somin<m1) m1=somin;}
if(*(p6+252+177)<0) {m1=samin;if(somin>m1) m1=somin;}
*(p4+1)=*(p6+252+177)-m1;

if(*(p6+252+177)>0) m1=m1+(*(p4+1)/2);
if(*(p6+252+177)<0) m1=m1-(*(p4+1)/2);
*(p4+2)=bali-m1;

m1=0;
for(m=somik;m<=samik;m++)
      {
      if((*(p6+252+177)>0) && (*(p6+252+m-1)>bali))
                              m1=m1+*(p6+252+m-1);
      if((*(p6+252+177)<0) && (*(p6+252+m-1)<bali))
                              m1=m1+*(p6+252+m-1);
      }
*(p4+3)=m1/10;

for(m=5;m<=18;m++)   *(p4+m-1)=0;

}


void goster(float *p4,float *p5)
{

int  m,o,p,u,l,o1,kk1;
float     mm;

mm=320/PS;kk1=mm;

l=0;
for(o=1;o<=PS;o++)
 {

outtextxy(315,0,"Ozellik Vektorunun Histogrami");

for(m=1;m<=350;m++) putpixel(310,m+50,3);
for(m=1;m<=340;m++) putpixel(300+m,240,3);

   o1=o;


for(p=1;p<=kk1;p++)
   {
   l=l+1;

   if(*(p5+o-1)>0)
     {
     for(u=1;u<=*(p5+o-1);u++)  putpixel(310+l,240-u,0);
     }
```

```
    if(*(p5+o-1)<0)
      {
      for(u=1;u<=abs(*(p5+o-1));u++) putpixel(310+l,240+u,0);
      }

    if(o1==16)  o1=19;

    if(*(p4+o-1)>0)
      {
      for(u=1;u<=*(p4+o-1);u++)    putpixel(310+l,240-u,o1);
      }
    if(*(p4+o-1)<0)
      {
      for(u=1;u<=abs(*(p4+o-1));u++)putpixel(310+l,240+u,o1);
      }

    }
    *(p5+o-1)=*(p4+o-1);
  }

}


void DFT(float *p4,float *p6)
{

int         m,o,p,o1,p1;
float       h1,h2,h3,h4;

h4=0.0001;
for(o=0;o<250;o++)
    {
    o1=o-125;
    h1=h2=0;
    for(p=0;p<250;p++)
        {
        p1=p-125;
        h1=h1+*(p6+252+p)*cos(2*M_PI*o*p/250);
        h2=h2-*(p6+252+p)*sin(2*M_PI*o*p/250);
        }
    h3=(h1*h1+h2*h2);
    *(p6+2520+o)=(pow(h3,0.5))/1;

    *(p6+252+250)=*(p6+2520)/250;

    }
for(m=1;m<=PS;m++)   *(p4+m-1)=0.1**(p6+2520+m-1);
}


void latis(float *p4,float *p6)
{

float       m1,m2,m3,m4,m5;
int         j,l;
```

```
for(l=1;l<=250;l++)
    {
    *(p6+1008+l-1)=*(p6+252+l-1);
    *(p6+1764+l-1)=*(p6+252+l-1);
    }

for(l=1;l<=PS;l++)
   {
   m3=m4=0;
   for(j=2;j<=250;j++)
     {
     m3=m3+*(p6+1764+j-1)**(p6+1008+j-2);
     m4=m4+*(p6+1008+j-2)**(p6+1008+j-2);
     }m4=m4+*(p6+1008+249)**(p6+1008+249);m3=m3/250;m4=m4/250;
   m5=m3/m4;
   for(j=2;j<=250;j++)
      {
      *(p6+1260+j-1)=*(p6+1764+j-1)-m5**(p6+1008+j-2);
      *(p6+1512+j-1)=*(p6+1008+j-2)-m5**(p6+1764+j-1);
      }
   for(j=1;j<=250;j++)
      {
      *(p6+1764+j-1)=*(p6+1260+j-1);
      *(p6+1008+j-1)=*(p6+1512+j-1);
      }
   *(p4+l-1)=m5*100;
   }
}


void Fuzzy_K_Komsu_Si(float *p1,float *p2,float *p3)
{

float      m1,m2,buf[4][K+2];
int        n1,i,h,j;


for(i=1;i<=K;i++)    buf[1][i]=1000000000;

    for(h=1;h<=O;h++)
      {
      m1=0.0001;

      for(j=1;j<=PS;j++)
         {
         m1=m1+(*(p1+j-1)-*(p2+(h-1)*81+(j-1)))*
               (*(p1+j-1)-*(p2+(h-1)*81+(j-1)));
         }

      m2=0;
      for(i=1;i<=K;i++)
         {
         if( (m1<buf[1][i]) && (m2==0) )
                   {
                    m2=1;
```

```
                    for(j=K;j>=i;j--)
                     {
                     buf[1][j+1]=buf[1][j];
                     buf[2][j+1]=buf[2][j];
                     }
                    buf[1][i]=m1;buf[2][i]=h;
                   }
        }

     }

  m1=0.001;
  for(i=1;i<=K;i++)  m1=m1+(1/buf[1][i]);

  for(h=1;h<=S;h++)
     {
     m2=0.001;
     for(i=1;i<=K;i++)
        {
        n1=buf[2][i];
        m2=m2+( *(p3+(h-1)*61+(n1-1)) /buf[1][i]);
        }

     *(p1+h-1)=m2/m1;
     }

}


void Fuzzy_En_Yakin_Si(float *p1,float *p2)
{

float     d1[S+1],m1;
int       m,h,j,k;

for (h=1; h<=S; h++)
  {
  d1[h]=0.0000001;
  for (j=1; j<=PS; j++)
  d1[h]=d1[h]+pow((*(p1+j-1)-*(p2+(h-1)*81+(j-1)) ),2);
  }

  m1=0.000001;
  for (m=1; m<=S; m++)
               m1=m1+pow( (pow(d1[m],-1)), ust);

for (m=1; m<=S; m++)
  {
  *(p1+m-1)=pow( pow( d1[m], -1 ),ust) * pow(m1, -1);
  }

}
```

ÖZGEÇMİŞ

Zümray Dokur, 1970 yılında Muş'ta doğdu. Orta öğrenimini 1988 yılında Elazığ Anadolu Lisesi'nde birincilikle bitirdi. Lisans öğrenimini 1992 yılında İ.T.Ü. Elektronik ve Haberleşme Mühendisliğinde tamamladı. Aynı yıl İ.T.Ü. Elektronik Anabilim Dalına Araştırma Görevlisi olarak girmiştir ve halen bu bölümde çalışmaktadır.