HUMAN IDENTIFICATION FROM ECG SIGNALS VIA SPARSE REPRESENTATION OF LOCAL SEGMENTS

Department of Electrical and Computer Engineering

ECE 573 ADVANCED ENGINEERING DESIGN BY OPTIMIZATION

PROJECT REPORT

Submitted by

Alinstein Jose, (V00918748)

University of Victoria

Date: April 13th, 2019

ABSTRACT

This project implements a method to extract compact and discriminative features from Electrocardiogram (ECG) signals for human identification based on the sparse representation of local segments. Specifically, local segments are extracted from the ECG signals and projected to a small number of basic elements in a dictionary, which is an orthogonal random matrix. Finally, the features are extracted by performing max pooling procedure over all the spares coefficient vectors from the ECG signals. Three optimization methods are used to find the sparse coefficient vector from the ECG signals, and their performances are compared in this report. This method achieves a 96.00% accuracy on a 10 subject dataset.

Keywords: Sparse coding, $\mathbf{l}_1\mathbf{l}_2$ norm, local features.

1. INTRODUCTION

Biometric is an authentication method for comparing data for the person's characteristics such as the face, fingerprint, and gait to that person's biometric "template" in order to determine resemblance. Biometric authentication has been widely used because of many security applications. Recent research found that Electrocardiogram ECG signals can be used to identify individual human because each human being displays certain uniqueness in their ECG signals. In this project, a method to extract compact and discriminative features from the ECG signals based on the sparse signal representation of local segments. In order to extract local segments from the ECG signals, a window of predefined is slid through the ECG signals. Then each of the local segments is normalized to form a local feature of the patient (subject). We adopted sparse coding in to project the local features as a combination of a small number of basic elements in a dictionary, which is an orthogonal random matrix with a predefined size. We have performed sparse coding using various algorithms. To get the final representation of each ECG signal, a procedure called 'max pooling' is performed over all spares coefficient vectors of local segments in the ECG signals. Finally, test subjects ECG signals are classified using 1-Nearest Neighbour classifier (1-NN).

2. THEORY

2.1 Local Segment Extraction

Initially local segments are extracted from the segments which are taken from ECG signals. A predefined number of segments are taken from the patients ECG signal. And each segment is taken with random starting point from the ECG signals. This is to obtain local features from all the part of ECG signals. Then a window of a predefined length of 'd' is slid through the segment to extract local segments. Then we extract local features from each of the local segment. Finally, we normalize all the local features to the l_2 norm, to make it on the same scale.

2.2 Sparse Representation of Local Segments

We represent local feature as a combination of a small number of basic elements in the dictionary. This dictionary is a simple random orthogonal matrix. Since this dictionary is random matrix, it contains almost all the frequency components to represent the ECG signal. The local features are extracted by finding the sparsest vector to represent the local segment using the Dictionary '**D**'. Then the max pooling is performed on these sparse vectors of each local segment we get local features. The dimension of dictionary **D** is $\mathbf{R}^{dxK}$ (K>d). The problem of

sparse representation is to find a linear combination of a small number of bases in the dictionary to represent an ECG signal (local segment) $\mathbf{x} \in \mathbf{R}^d$, i.e.

$$\hat{\boldsymbol{\alpha}} = \arg\min \|\boldsymbol{\alpha}\|_0 \quad (1.1)$$
$$subject\ to\ \boldsymbol{D\alpha} = \boldsymbol{x}$$

Where $\boldsymbol{\alpha} \in \mathbf{R}^K$ is sparse coefficients vector that specifies the projection weight, and $\|\boldsymbol{\alpha}\|_0$ is the l_0 norm, that shows the number of non-zero entities in the vector $\boldsymbol{\alpha}$. This solution gives $\hat{\boldsymbol{\alpha}}$, which is the representation of signal **x** decomposed to a minimum number of bases of the dictionary. Since solving (1.1) is NP hard problem, we approximate this l_1 norm.

$$\hat{\boldsymbol{\alpha}} = \arg\min \|\boldsymbol{\alpha}\|_1 \quad (1.2)$$
$$subject\ to\ \boldsymbol{D\alpha} = \boldsymbol{x}$$

For a set of local segments $\mathbf{X=[x_1,x_2,x_3,\ldots\ldots,x_N]} \in \mathbf{R}^{K \times N}$, the constrained problem can be reformulated unconstrained problem as follows:

$$\min_{\boldsymbol{\alpha_i} \in \boldsymbol{R^K}} \quad \frac{1}{2}(\|\boldsymbol{x_i} - \boldsymbol{D\alpha_i}\|_2^2 + \lambda\|\boldsymbol{\alpha_i}\|_1) \quad (1.3)$$

Where λ is the regularization parameter for sparsity. The optimization problem is not jointly convex but is separately convex with each of the two variables. As there is a large number of datasets, solving this is still an open problem. In this project we minimize this problem with three methods, using CVX, Proximal-Point Algorithm, and Alternative direction method of multipliers (ADMM). CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. The *Alternating Direction method of multipliers* (ADMM) is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. In Proximal-Point Algorithm, each

iteration involves matrix-vector multiplication involving D^T and D followed by a shrinkage/soft-threshold.

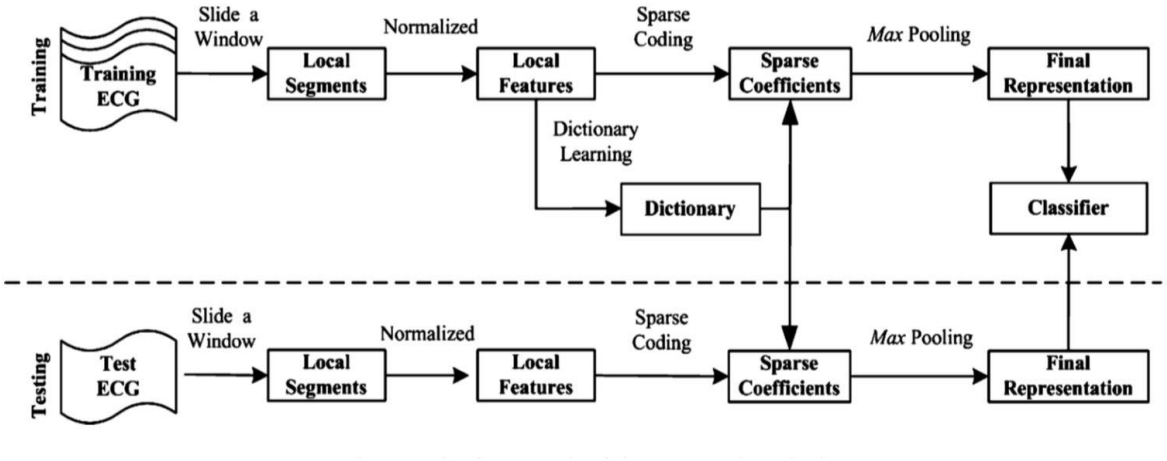

Figure 1: The framework of the proposed method.

Structural similarity information can be captured using a method called max pooling function over all the local sparse coefficient vectors of each segment from ECG signals. The max pooling procedure that is consistent with bio-physical evidence in visual coted has been successfully used in image classification to obtain a spare-invariant representation. It is shown that the max pooling procedure obtains a compact and discriminative final representation, although spatial information of local image patches is ignored. The max pooling procedure ignores the temporal order of local segments in an ECG signal and only preserves the strongest response of each basic element in the dictionary.

Suppose M local segments are extracted from an ECG signal an the corresponding sparse coefficient are $\mathbf{A} = [\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \ldots\ldots, \boldsymbol{\alpha_M}] \in \mathbf{R}^{K \times M}$, the max pooling function is defined as:

$$\boldsymbol{\beta} = \varepsilon_{Max}(\boldsymbol{A}) \tag{1.4}$$

Where $\boldsymbol{\beta} \in \mathbf{R}^K$ is the final vector representation of the ECG signal, and ε_{Max} is the max pooling function over the absolute sparse coefficients:

$$\beta_k = max(|\boldsymbol{\alpha_{1k}}|, |\boldsymbol{\alpha_{2k}}|, \ldots\ldots |\boldsymbol{\alpha_{Mk}}|) \quad (1.5)$$

Where β_k is the Kth element of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha_{ik}}$ is the kth element of the ith local segment's sparse coefficient. An example of an ECG signal segment is shown in Figure 2. The Figure 3, Figure 4, Figure 5, and Figure 6 are the local segments of the ECG signal **x** and the corresponding sparse coefficient vector $\boldsymbol{\alpha}$. The Figure 7 is the representation constructed by max pooling over all the sparse coefficient vector of the ECG signal.

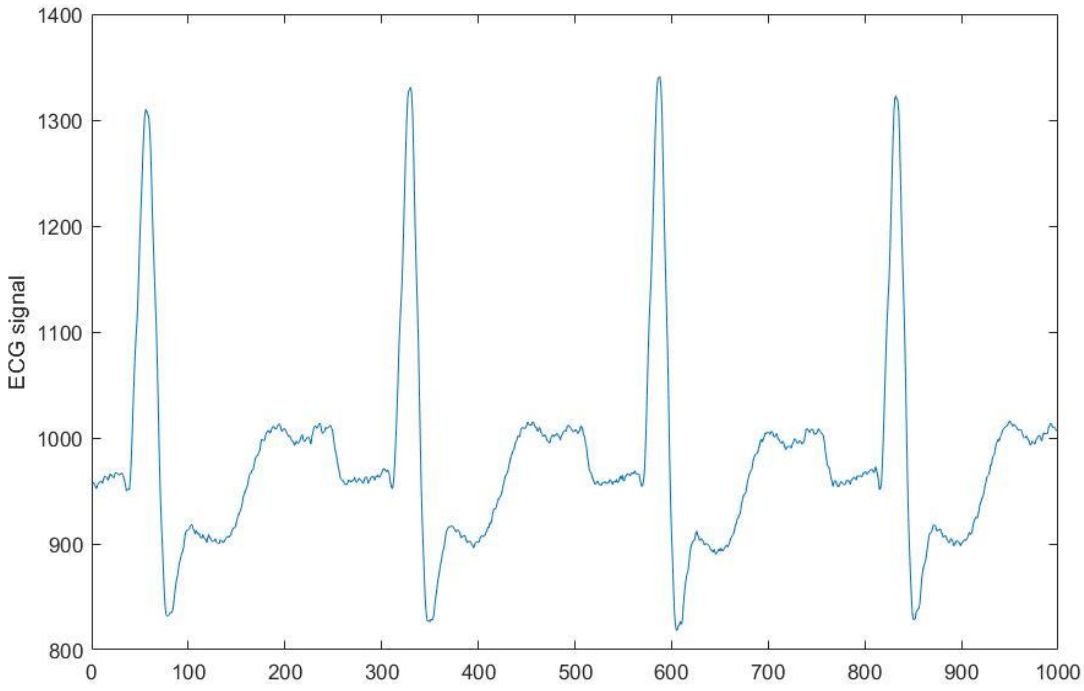

Figure 2: A sample ECG signal segment

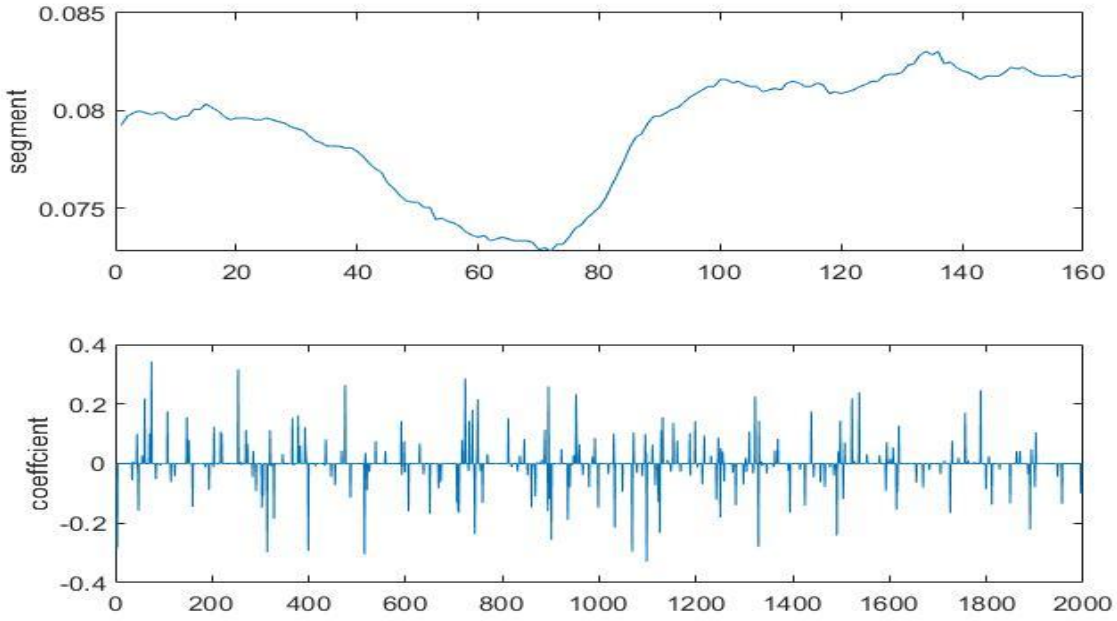

Figure 3: A local segments of the ECG signal and its corresponding sparse coefficient vector.

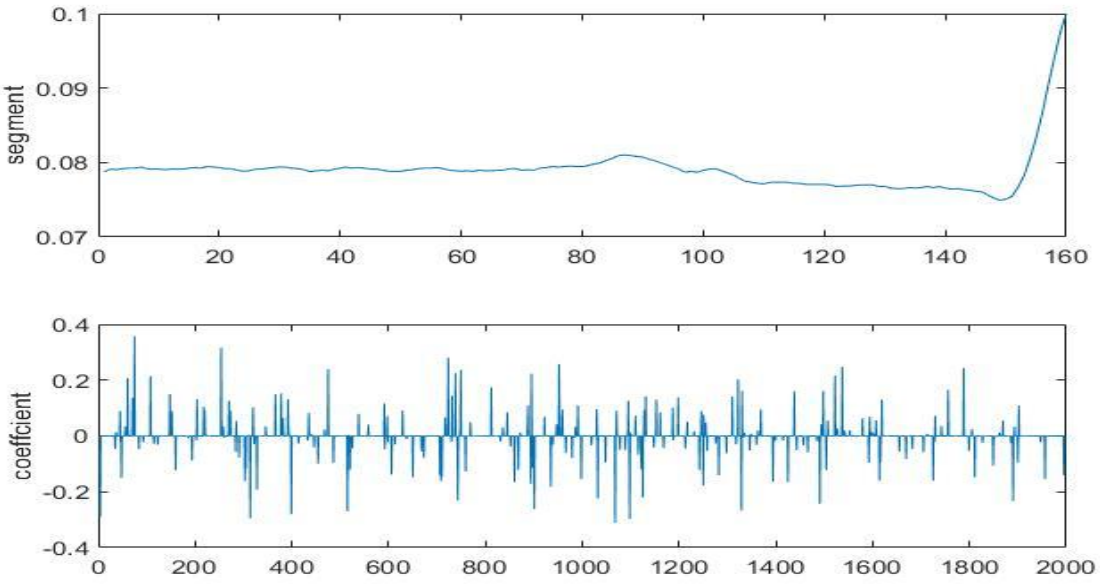

Figure 4: A local segments of the ECG signal and its corresponding sparse coefficient vector.

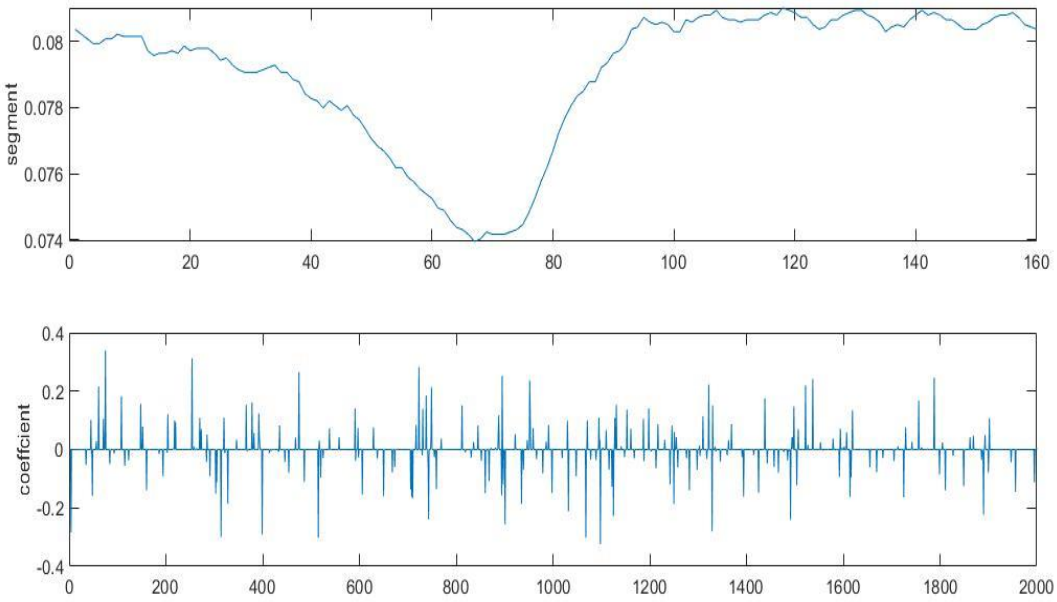

Figure 5: A local segments of the ECG signal and its corresponding sparse coefficient vector.

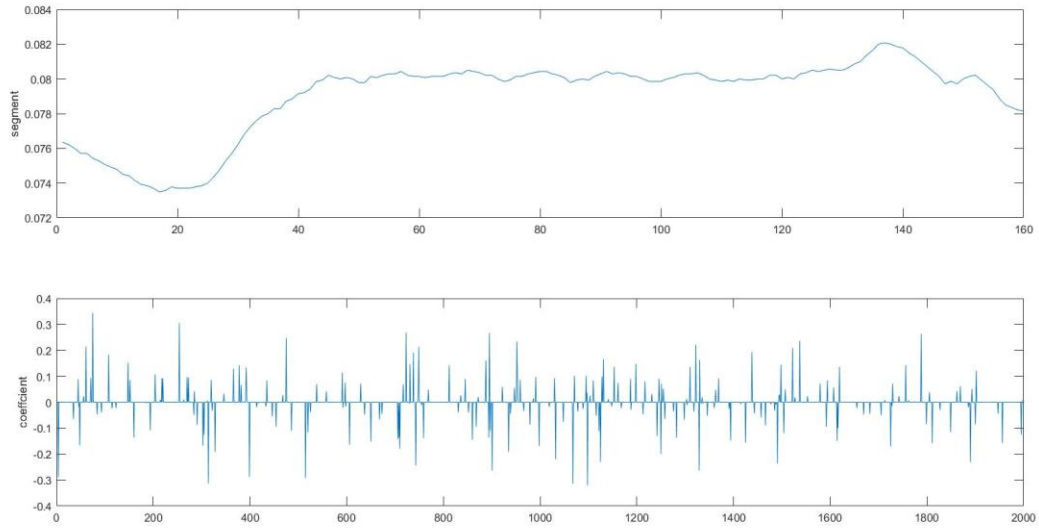

Figure 6: A local segments of the ECG signal and its corresponding sparse coefficient vector.

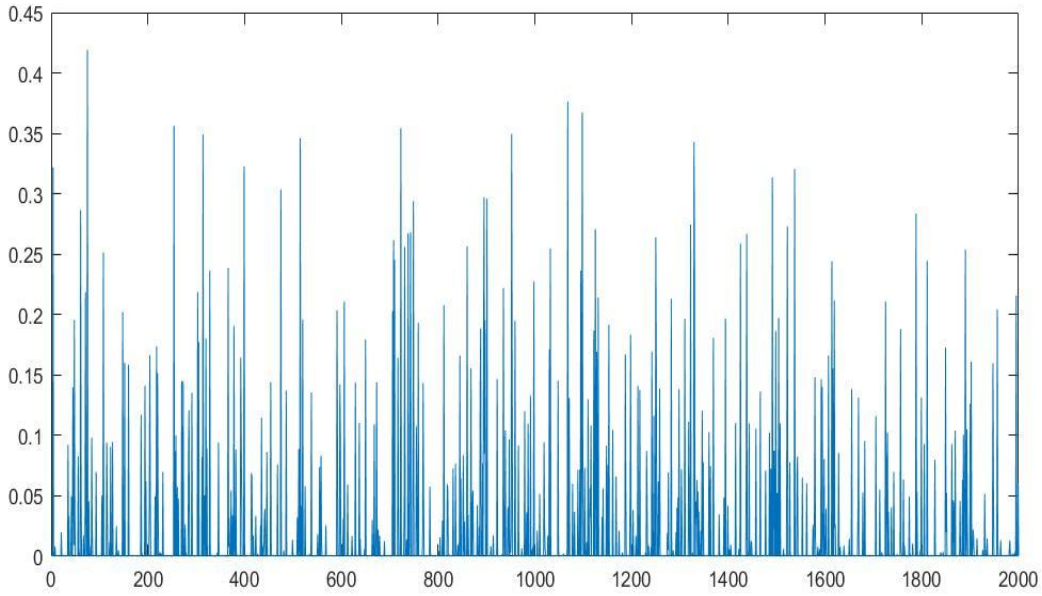

Figure 7: Final vector representation of the ECG signal

The final features $\boldsymbol{\beta}$ are inputted to a simple classifier, that is the 1 Nearest Neighbour (1-NN) or classification. Let $\boldsymbol{\beta^{tr}}$ are training data sets and $\boldsymbol{\beta^{te}}$ are testing data sets of ECG signal generated before. The test data is determined as the subject C whose training sample has minimal distance with the test data i.e., $\mathbf{C}^* = \arg\min_i \text{Dist}(\boldsymbol{\beta^{tr}}, \boldsymbol{\beta^{te}})$, where dist(.,.) is the Euclidean distance measure of two feature vectors.

3. IMPLEMENTATION

3.1 Local Segment Extraction

Data set required from this project is collected from the MIT-BIH database, which is widely used for ECG analysis. The MIT-BIH database contains 48 ECG records, sampled at a frequency of 360 Hz. For the easier assess of data set, a file is created, which contains all the names of the data set. From the collection of records, we randomly choose 'Nopatient' number of records(patients) for the experiment. Then from each record 'Nseg' number of segment of length of 1000 samples are taken at random starting positions. 'Nseg' contains the segments for both testing and training data. The testing and training segments are stored in different variables. Then local segments created from each segment. To create local segments a window

of predefined length (d) is slid through the segments. In this experiment windows are slid with a standard gap (gp), this is done to reduce the amount of computation power, as a consequence if the gap is large the efficiency decrease. In this experiment, we use an orthogonal random matrix with dimension dxK**,** where d is the length of local segments and K is the length of sparse coefficient vector**.** The value of lamda in formula 1.3 is initialized as 0.25/sqrt(d).

3.2 CVX

Three ways are used to minimize the function (formula 1.3). First, a method using CVX, which is a Matlab-based modeling system for convex optimization. CVX is used to minimize function in (1.3) for each local segments. This record is collected as a reference to compare with other algorithms performed here. Figure 8 is the sparse coefficient vector of a random local segment from the ECG signal obtained using CVX.

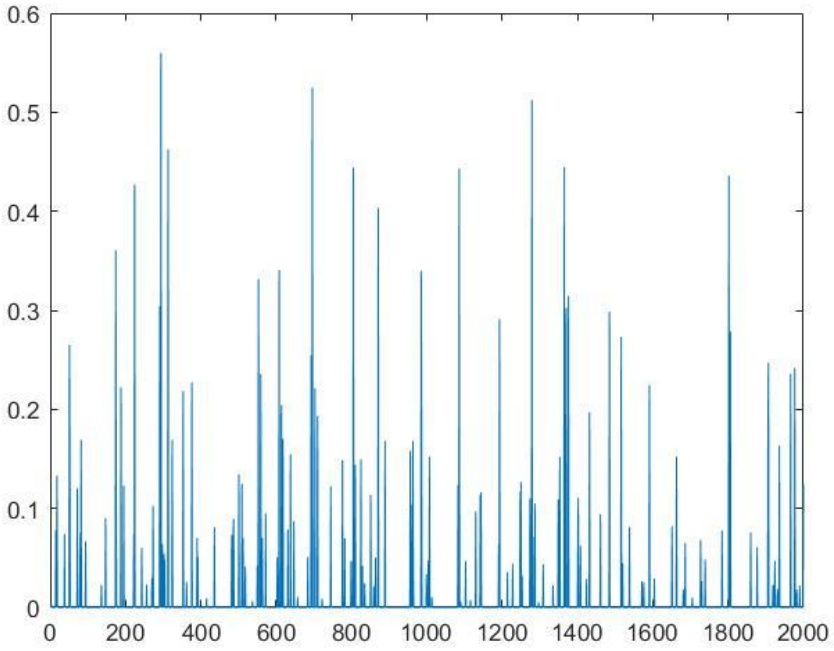

Figure 8: Sparse coefficient vector generated from the local segment using CVX toolbox.

3.3 Proximal Point Algorithm

This one of the famous algorithm to minimize l_1l_2 minimization problem. Soft-shrinkage is applied to minimize this function in this algorithm. The original problem is formulated here to apply proximal point algorithm.

$$\boldsymbol{\alpha}_{k+1} = arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{D\alpha_k}\|_2^2 + \lambda\|\boldsymbol{\alpha_k}\|_1\right)$$

$$\boldsymbol{\alpha}_{k+1} = arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\left(\left\| \boldsymbol{x} - (\boldsymbol{\alpha_k} - \frac{\mathbf{1}}{\boldsymbol{L}} * \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k}) \right\|_2^2 + \lambda \|\boldsymbol{\alpha_k}\|_1 \right)$$

Here L=1, As $L = \lambda_{max}$(A*A')=1, A=I;

$$\boldsymbol{\alpha}_{k+1} = arg\min_{\boldsymbol{\alpha}} \frac{1}{2}(\|\boldsymbol{x} - (\boldsymbol{\alpha_k} - \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k})\|_2^2 + \lambda \|\boldsymbol{\alpha_k}\|_1)$$

$\mathbf{b_k} = (\boldsymbol{\alpha_k} - \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k}))$

So, by soft-shrinkage:

$$\boldsymbol{\alpha}_{k+1} = S_\lambda((\boldsymbol{\alpha_k} - \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k})))$$

$$\boldsymbol{\alpha}_{k+1} = sign((\boldsymbol{\alpha_k} - \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k}))).\max(|\boldsymbol{\alpha_k} - \boldsymbol{D}(\boldsymbol{x} - \boldsymbol{\alpha_k})| - \lambda, 0) \quad (1.6)$$

The Figure 9 shown below in sparse coefficient vector of Local segment optimized using formula 1.6. To make both sparse vector generated by CVX and proximal point algorithm similar method various values of threshold (λ) and iteration is tried for formula 1.6.

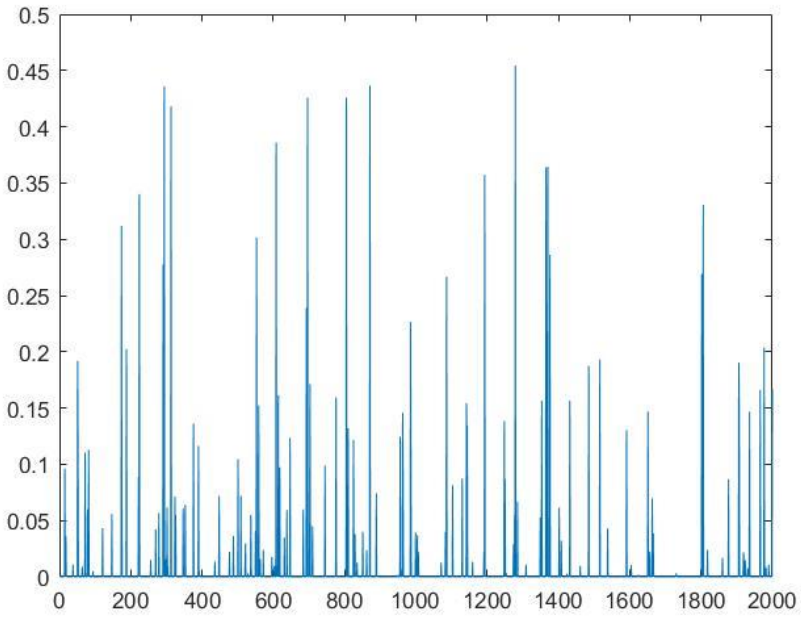

Figure 9: Sparse coefficient vector generated from local segment using proximal point algorithm.

3.4 Alternate Direction Method of Multipliers

ADMM method has become increasingly important because of its ability to deal with large scale convex problems. Since this project contains a huge number of data set, ADMM is introduced here to minimize function (Equation 1.3). To minimize the function in (Equation 1.3) the following steps are calculated:

Step a) Initialize: (λ):lamd=0.25/sqrt(d),y(1)=0;lamda(1)=0 and alpha=lamd;

Step b) Start loop and calculate the following

$$\boldsymbol{x_{k+1}} = (\boldsymbol{D'D} + alpha * \boldsymbol{I})^{-1} * (\boldsymbol{A'b} + alpha * \boldsymbol{y_k} - \boldsymbol{lamda_k})$$

$$\boldsymbol{y_{k+1}} = S_{\frac{lamd}{alpha}}(\boldsymbol{x_{k+1}} + \boldsymbol{lamda_k}/alpha)$$

$$\boldsymbol{lamda_{k+1}} = \boldsymbol{lamda_k} + alpha(\boldsymbol{x_{k+1}} - \boldsymbol{y_{k+!}})$$

Step c) Break the loop when primal residual and dual residual are less than the predefined value.

Else go back step a

The Primal residual: $r_k = ||\mathbf{x_{k+1}}\text{-}\mathbf{y_{k+1}}||_2$

Dual residual : $d_k = ||alpha*(\mathbf{y_{k+1}}\text{-}\mathbf{y_k})||_2$

Both alpha and lamd is assigned with the value 0.25/sqrt(d), because this value gives better efficient classification.

The following two figures show the primal residual (Figure 10) and dual residual (Figure 11) of sample local segment from the ECG. The Figure 11 depicts that the amplitude of the dual residual reaches 10^{-03} by 50 iterations, but it took more than 100 iteration to reach 10^{-03} for primal residual. However, comparing to proximal point algorithm, which took 1000 iteration to reach the optimal solution. This method has drastically reduced the number of iterations.

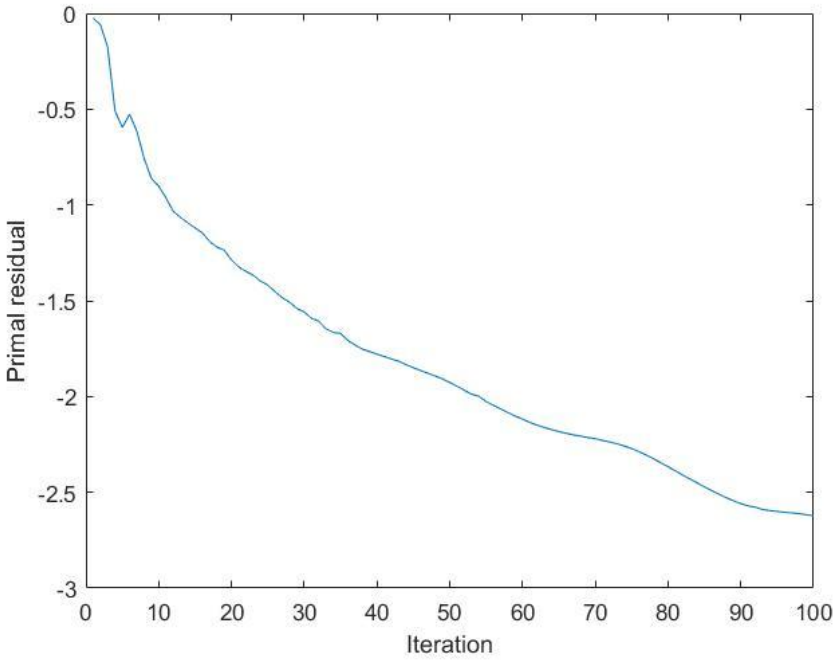

Figure 10: Primal residual of the local segment from ECG (in log)

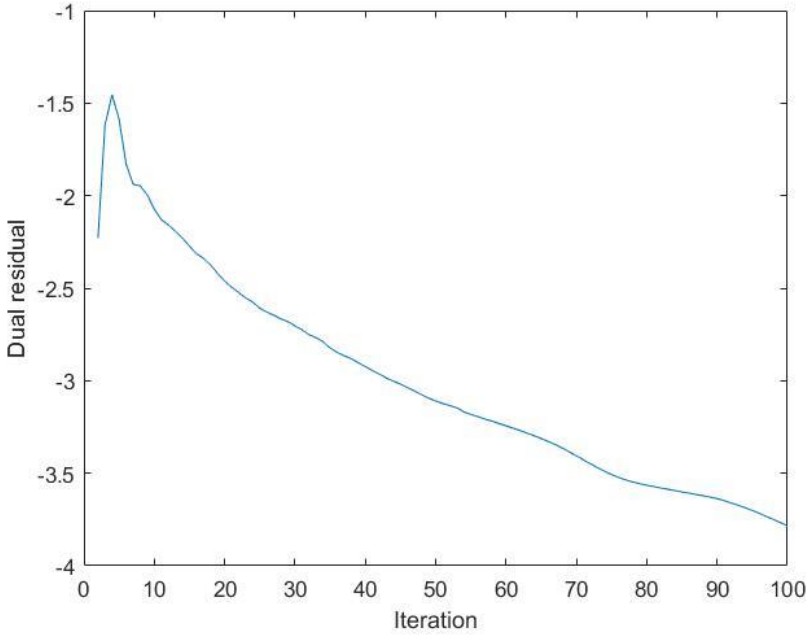

Figure 10: Dual residual of the local segment from ECG (in log)

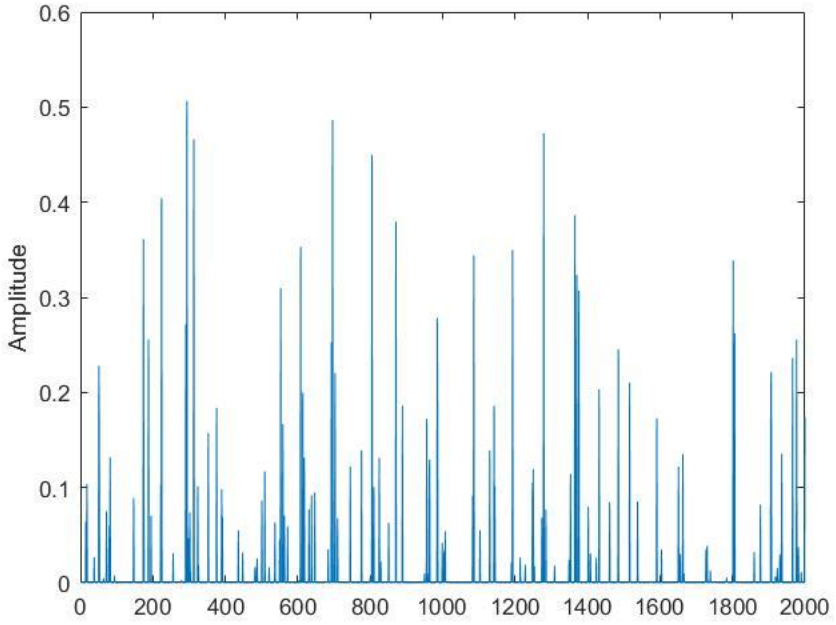

Figure 11: Sparse coefficient vector generated from the local segment using ADMM.

3.5 Max pooling and Classification

After generating the final sparse vector of the local segment using each segment, max pooling is performed. It is performed according to Equation 1.5. The final vector generated after max pooling of testing set is compared with all the training vectors. This function finds two similar vectors by computing the shortest Euclidian distance between the vectors. Then their efficiency is also computed and it is output is returned.

3.6 MATLAB CODE

MAIN FUNCTION

```
clc;
clear all;
close all;

%Nopatient:              Number of patients
%Nseg:                   Total number of training and testing
segments for patients
%ntr:                    Number of train segments for patients
%nte:                    Number of test segments for patients
%K:                      Size of dictionary is d by K
%st1,st2,st3:            Initial random states.
%d:                      Length of the window used to create a
local segment
% gp: number of samples a local segment differs from the next
local segment.


Nopatient=3;
ntr=10;
nte=10;
d=160;
K=2000;
st1=9;
st2=17;
st3=30;
d=160;
gap=100;

% File ind_mit2019 stores the name of all patients files
Nseg = ntr + nte;
Dtr = [];
Dte = [];
load ind_mit2019
InG = ind_mit2019;
L = length(InG);
rand('state',st1)
Lr = randperm(L);
Lr = Lr(1:Nopatient);
store=[];

%Required data patients are extracted from the file names from
the index
for i = 1:Nopatient
```

```
    ni = Lr(i);
    fname = sprintf('a%.0f.mat',InG(ni));
    si = load(fname);
    ai = struct2cell(si);
    di = cell2mat(ai);
    di=di(1:500000);
    store(:,i)= di;
end


%Following loop takes Nseg number of segments (for both %testing
and training) for all the patients
for j=1:Nopatient

        Li = length(store(:,j));
        rand('state',st2+j-1)
        r = randperm(Li);
        for i=1:Nseg
            rq=r(i);
            segtemp(i,:)=store(rq:rq+999,j)';
        end
        Trsegments(j,:,:)=segtemp(1:ntr,:);
        Tesegments(j,:,:)=segtemp(ntr+1:Nseg,:);
end

%p stores numbers of windows applied to each segment
p = 1 + floor((1000 - d)/gap);
Ktr = size(Trsegments,2);

%This loop applies a window to training segments to obtain local
segments
for k=1:Nopatient
    for j=1:ntr
        for i=1:p
        temp=Trsegments(k,j,(i-1)*gap+1:(i-1)*gap+d);
        temp=reshape(temp,d,1);
        temp=temp/norm(temp);
        TrainLseg(k,(j-1)*p+i,:)=temp;
        TrainLseg4d(k,j,i,:)=temp;
        end
    end
end


%This loop applies a window to testing segments to obtain local
segments
```

```
for k=1:Nopatient
    for j=1:nte
        for i=1:p
        temp=Tesegments(k,j,(i-1)*gap+1:(i-1)*gap+d);
        temp=reshape(temp,d,1);
        temp=temp/norm(temp);
        TestLseg(k,(j-1)*p+i,:)=temp;
        TestLseg4d(k,j,i,:)=temp;
        end
    end
end


 TestLseg4d=permute(TestLseg4d,[4 3 2 1]);
 TrainLseg4d=permute(TrainLseg4d,[4 3 2 1]);

%create Random dictionary with dimension d and K
randn('state',st3)
D=randn(d,K);
D=orth(D')';

%lamda is chosen as 0.25/sqrt(d)
lamda=0.25/sqrt(d);

Btr = [];

tic
%This block calculates the sparse vector for each local segment
using cvx
%for training set
Btr = [];
for i = 1:Nopatient
    for j = 1:ntr
        t = (i-1)*ntr*p+(j-1)*p;
        Atr = [];
        for k = 1:p
            xk = Xtr(:,t+k);
            cvx_begin quiet
            variable a(K,1)
            minimize(0.5*norm(D*a-xk)+lam*norm(a,1))
            cvx_end
            Atr = [Atr a];
            current_state_tr = [i j k]

        end
        b = max(abs(Atr)')';
        Btr = [Btr b];
```

```
    end
end

%Following calculates the absolute value of each sparse vector
for k=1:Nopatient
    for j=1:ntr
        for i=1:p
        mspar(:,i,j,k)=abs(spar(:,i,j,k));
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:ntr
        beta(:,j,k)=max(mspar(:,:,j,k),[],2);
    end
end


Bte = [];
%This block calculates the sparse vector for each local segment
using CVX
%for testing set
tic
for k=1:Nopatient
    for j = 1:nte%Nseg
        Ate = [];
        for i=1:p
            cvx_begin quiet
            variable alp(K,1)
            minimize (  (0.5*norm(TestLseg4d(:,i,j,k)-
D*alp)+lamda*norm(alp,1)) )
            cvx_end
            spartes(:,i,j,k)=alp;
            Ate = [Ate alp];
            current_state_tr = [i j k];
        end
          b = max(abs(Ate)')';
          Bte = [Bte b];
    end
end
%end

%Following calculates the absolute value of each sparse vector
toc
```

```
for k=1:Nopatient
    for j=1:nte
        for i=1:size(TestLseg4d,2)
        mspartes(:,i,j,k)=abs(spartes(:,i,j,k));
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:nte
        betates(:,j,k)=max(mspartes(:,:,j,k),[],2);
    end
end
effciency_of_using_cvx_is = [ checkeffciency(Btr,Bte,ntr,nte)]



%This block calculates the sparse vector for each local segment
using
%proximal method for the training set
Btr = [];

for k=1:Nopatient
    for j = 1:ntr

        Atr = [];
        for i=1:p
        theta2=zeros(K,1);
        for kq=1:1000
              z1=(theta2+D'*(TrainLseg4d(:,i,j,k)-D*theta2));
             theta2=sign(z1).*max((abs(z1)-lamda/0.9),0);
        end
        alp=theta2;
        spar(:,i,j,k)=alp;
        current_state_tr = [i j k]
        Atr = [Atr alp];
        end
        b = max(abs(Atr)')';
        Btr = [Btr b];
      end
end


%Following calculates the absolute value of each sparse vector
for k=1:Nopatient
```

```
    for j=1:ntr
        for i=1:p
        mspar(:,i,j,k)=abs(spar(:,i,j,k));
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:ntr
        beta(:,j,k)=max(mspar(:,:,j,k),[],2);
    end
end


Bte = [];
%This block calculates the sparse vector for each local segment
using
%proximal method for testing set

for k=1:Nopatient
    for j = 1:nte%Nseg
        Ate = [];
        for i=1:p

            theta2=zeros(K,1);
            for kq=1:1000
                  z1=(theta2+D'*(TestLseg4d(:,i,j,k)-D*theta2));
                theta2=sign(z1).*max((abs(z1)-lamda/.9),0);
            end
            alp=theta2;
            spartes(:,i,j,k)=alp;
            Ate = [Ate alp];
            current_state_te = [i j k]
        end
          b = max(abs(Ate)')';
          Bte = [Bte b];
    end
end


%Following calculates the absolute value of each sparse vector
for k=1:Nopatient
    for j=1:nte
        for i=1:size(TestLseg4d,2)
        mspartes(:,i,j,k)=abs(spartes(:,i,j,k));
```

```
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:nte
        betates(:,j,k)=max(mspartes(:,:,j,k),[],2);
    end
end
effciency_of_using_proximal_method_is = [
checkeffciency(Btr,Bte,ntr,nte)]




%This block calculates the sparse vector for each local segment
using
%ADMM for training set
Btr = [];
r4=0;
d4=0;
alppha=lamda;
Inv=(D'*D+alppha*eye(K));
v=inv(Inv);
for k=1:Nopatient
    for j = 1:ntr
        Atr = [];
        for i=1:p

        xk1(:,1)=zeros(K,1);lamk(:,1)=zeros(K,1);
        yk1(:,1)=zeros(K,1);

        v1=D'*TrainLseg4d(:,i,j,k);
        for f=1 :60
            xk1(:,f+1)=v*(v1+alppha*yk1(:,f)-lamk(:,f));
            kk1=xk1(:,f+1)+lamk(:,f)/alppha;
            yk1(:,f+1)=sign(kk1).*max((abs(kk1)-lamda),0);
            lamk(:,f+1)=lamk(:,f)+alppha*(xk1(:,f+1)-
yk1(:,f+1));
            r4(f)=norm(xk1(:,f+1)-yk1(:,f+1),2);
            d4(f)=norm(alppha*(yk1(:,f+1)-yk1(:,f)),2);
            if(r4(f)<1e-2)
                if (d4(f)<1e-2)
                    break;
```

```
                end
            end
        end

        alp=xk1(:,f);
        spar(:,i,j,k)=alp;
        current_state_tr = [i j k f]
        Atr = [Atr alp];
        end
        b = max(abs(Atr)')';
        Btr = [Btr b];

    end
end


%Following calculates the absolute value of each sparse vector
for k=1:Nopatient
    for j=1:ntr
        for i=1:p
        mspar(:,i,j,k)=abs(spar(:,i,j,k));
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:ntr
        beta(:,j,k)=max(mspar(:,:,j,k),[],2);
    end
end

r4=0;
d4=0;
Bte = [];
%This block calculates the sparse vector for each local segment
using
%ADMM for testing set
for k=1:Nopatient
    for j = 1:nte%Nseg
        Ate = [];
        for i=1:p

        xk1(:,1)=zeros(K,1);lamk(:,1)=zeros(K,1);
        yk1(:,1)=zeros(K,1);
```

```
        v1=D'*TestLseg4d(:,i,j,k);
        for f=1 :35
               xk1(:,f+1)=v*(v1+alppha*yk1(:,f)-lamk(:,f));
            kk1=xk1(:,f+1)+lamk(:,f)/alppha;
            yk1(:,f+1)=sign(kk1).*max((abs(kk1)-lamda),0);
            lamk(:,f+1)=lamk(:,f)+alppha*(xk1(:,f+1)-
yk1(:,f+1));
            r4(f)=norm(xk1(:,f+1)-yk1(:,f+1),2);
            d4(f)=norm(alppha*(yk1(:,f+1)-yk1(:,f)),2);
            if(r4(f)<1e-3)
                if (d4(f)<1e-3)
                    break;
                end
            end
        end

        alp=xk1(:,f);
        spartes(:,i,j,k)=alp;
        Ate = [Ate alp];
        current_state_te = [i j k]
        end
      b = max(abs(Ate)')';
      Bte = [Bte b];
    end
end
%end

%Following calculates absolute value of each sparse vector

for k=1:Nopatient
    for j=1:nte
        for i=1:size(TestLseg4d,2)
        mspartes(:,i,j,k)=abs(spartes(:,i,j,k));
        end
    end
end

%This block calculates the max pooling, which takes the largest
%coefficient from all columns
for k=1:Nopatient
    for j=1:nte
        betates(:,j,k)=max(mspartes(:,:,j,k),[],2);
    end
end


effciency_of_using_ADMM_is = [ checkeffciency(Btr,Bte,ntr,nte)]
```

Function to classify the test set and calculate efficiency

```
%This function classify each test local segment and returns the
%efficiency of classification.
function efficency=checkeffciency(Btr,Bte,ntr,nte)
Ltrain=size(Btr,2);
Ltest=size(Bte,2);
error=zeros(Ltest,1);
result=zeros(Ltest,1);

%This loop find the shortest euclidean distance for each Bte
column
%vector to Btr
for i=1:Ltest

   for j=1:Ltrain
      error(j)=norm(Btr(:,j)-Bte(:,i));
   end

   [M,I] = min(error);
   intex(i)=I;
end

range=1:ntr;
range=ntr:ntr:Ltrain;

j=1;
for i=1:Ltest

    if j==1
         if intex(i)<=range(j)
        result(i)=1;
         end
    end
    if j~=1
           if intex(i)<=range(j) & intex(i)>=range(j-1)
            result(i)=1;
           end
    end


    if range(j)==i*(ntr/nte)
       j=j+1;
```

```
    end
end

efficency=sum(result)/size(result,1)
end
```

4. Results and Evaluations

Table 1: Compares three methods, for 10 patients, 15 training and 15 testing segments with a gap of 5.

Method	Efficiency of Classification	Time to compute
CVX	94.67%	56 hours
ADMM	96.00%	1 hour 10 minutes
Proximal point Algorithm with(0.25/sqrt(d))*1.11	95.33%	1 hour 46 minutes

Table 2: Compares the ADMM method for different gap, for 5 patients, 10 training and 5 testing segments.

Gap	Efficiency of Classification
5	96
30	78
50	60
100	48

Table 3: Compares efficiency of various threshold value for proximal point algorithm

Threshold:(Lamda=(0.25/sqrt(d)))	Efficiency of Classification
Lamda*1.67	92.67%
Lamda*1.25	94.67%
Lamda*1.11	95.33%
Lamda*1	94.00%

Table4: Compares efficiency of various cut-off values of primal and dual residual for 10 patients,15 training and testing set.

Minimum primal residual	Minimum Dual residual	Efficiency of Classification
10^{-1}	10^{-1}	90.67%
10^{-2}	10^{-2}	93.33%
10^{-3}	10^{-3}	96.00%
10^{-4}	10^{-4}	94.67%

5. Discussion

Table 1 is the comparison of the efficiency of three methods such as CVX, ADMM, and Proximal point algorithm. From Table 2 shows how efficiency changes with respect to the gap, where a gap is a number of samples a local segment differs from the next local segment. It is evident that as we increase gap classification efficiency is decreasing. The high value of gaps is chosen because of time limitation and long time required to compute small gaps. From the Table 3, it can be inferred that classification efficiency is best when the threshold value of proximal point algorithm is equal to (0.25/sqrt(d))*1.11, where d is the length of each local segment.
Table 4 is the comparison of the efficiency of classification for 4 different cut-off values of primal and dual residual. From the table we could conclude that the classification efficiency is highest when the primal residual and dual residual are made to $10^{-3.}$

CVX is very powerful tool to minimize the objective function by formulating the function as l_1l_2 form; however, the time required minimizing every function of local segments for 10 patients is extremely long. In this project to train 10 patients, it has taken 56 hours. The efficiency in classifying the testing set is above 90%. These results are satisfiable because a number of segments used to train classifier are small. So, if are able to increase the number of segments to train classifier we can increase the classifier efficiency.

ADMM is able to minimize the objective function with much less time than CVX and has classification efficiency like CVX. For most cases in ADMM, the dual residual reaches 10^{-03} by 50 iterations, and primal residual reaches reach 10^{-03} by 100 iterations. ADMM method yields satisfactory classification efficiency for this experiment.

Proximal point algorithm method is a simple method to implement and minimize the objective functions, and time to compute is slightly more than to ADMM and less than CVX. However, a number of iteration required to minimize function is near 1000. This method is successful in classifying the patients from the ECG signal with satisfactory efficiency.

6. Conclusion

This project has focused on dealing with human identification from ECG signal using a sparse representation of local segments from ECG signals. Project is successfully completed by classifying human sample ECG signal with an efficiency of 96.00% for 10 subjects with 15 testing segments and 15 training segments. The objective function is minimized with three methods and their characteristics are obtained and discussed.

7. References

[1] J. Wang, M. She, S. Nahavandi and A. Kouzani, "Human Identification From ECG Signals Via Sparse Representation of Local Segments," in IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 937-940, Oct. 2013.*
doi: 10.1109/LSP.2013.2267593
[2] W.-S. Lu, Course notes of Advanced Mathematical optimizations.
[3] Y.Wang,F.Agrafioti,D.Hatzinakos,andK.N.Plataniotis,"Analysis of human electrocardiogram for biometric recognition," EURASIP J. Adv. Signal Process., vol. 2008, no. 19, 2008.
[4] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov,R.G.Mark,J.E.Mietus,G.B.Moody,C.-K.Peng,andH.E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of anewresearchresourceforcomplexphysiologicsignals,"Circulation, vol. 101, no. 23, pp. 215–220, 2000.