Program outline:

Main function: GAclassifier

```
function GAclassifier
%   This function is used to perform classification for
%   terahertz signals  based on geometric algebra theory.

%   Southwest University
%   Shengling Zhou
%   May,2016

%data collecting
load Xtime.mat      %The  horizontal ordinate of reference temporal signal set
load Rsignal.mat    %:reference temporal signal set
load Ssignal.mat    %sample temporal signal set
Ssignal={Mel Tar Lac Glu};


for n=1:4   % four substances

    % get vector set from temporal signal set
    [Vectorset, Frquencyset]=getVectorsetfromSignal(Xtime,Rsignal,Ssignal{n});
    VectorSets{n}=Vectorset;
    FrequencySets{n}=Frquencyset;

    %get vectors that could be used to form 2-blade for a given substance
    % and the modular of their wedge product
    Bladeset{n}=prepareforGAclassifier(Vectorset);
end

% save vector set and frequency set into file GAvectorset.mat
save GAvectorset.mat VectorSets FrequencySets

% store the modulus of the wedge product
Modulus={Bladeset{1}{3} Bladeset{2}{3} Bladeset{3}{3} Bladeset{4}{3}};

% store four pair of vectors that could be used to form 2-blades for four
% different substances
VectorPair{1}=[Bladeset{1}{1},Bladeset{1}{2}];
VectorPair{2}=[Bladeset{2}{1},Bladeset{2}{2}];
VectorPair{3}=[Bladeset{3}{1},Bladeset{3}{2}];
VectorPair{4}=[Bladeset{4}{1},Bladeset{4}{2}];

% predict by GA classifier and save the  predicted results of the
% GAclassifier to AccbyGA.mat
predictbyGA(VectorPair, Modulus)
```

```
% load the  predicted results of the GA  classifier
load AccbyGA.mat
A=[cell2mat(ACC_Mel);cell2mat(ACC_Tar);cell2mat(ACC_Lac);cell2mat(ACC_Glu)];
A=[A;mean(A)]
end % end of the function GAclassifier
```

subfunction: getVectorsetfromSignal

```
function [Vectorset Frquencyset]=getVectorsetfromSignal(Xtime,Rsignal,Ssignal)
%   This function is used to get the transform function set  and
%   map  m-dimensional complex valued vectors to 2m-dimensional
%   real-valued vectors.

% input arguments:
%      Xtime:    time delay set;
%      Rsignal:     reference signal set
%      Ssignal:      sample signal set
% output arguments:
%      Vectorset:   the transformed vector set;
%       Frquencyset:  frequency set

dbstop if error
[m,n]=size(Ssignal{1});
for j=1:5   % five different thickness: 1.0 mm 1.5 mm 2.0 mm 2.5 mm 3.0 mm
    X=Xtime;
    R=Rsignal;
    S=Ssignal{j};
    for i=1:n
        [v,f]=getVector(X(:,i),R(:,i),S(:,i));
        Vector(:,i)=v;
        Frequency(:,i)=f;
    end
    Vectorset{j}=Vector;
    Frquencyset{j}=Frequency;
end

end  % end of function getVectorsetfromSignal
```

subfunction: getVector

```
function [v,f]=getVector(xt,r,s)
%   This function is used to get the transform function for a given
%   terahertz temporal signal and map  m-dimensional complex
%   transform function to 2m-dimensional real-valued vector.

% input arguments:
%      xt:    time delay;
%       r:     reference signal
%       s:      sample signal
% output arguments:
%      v:   the transformed vector;
%       f:  frequency

c=3e8; %The speed of light,
Fs=1.0e12*length(xt)/(max(xt)-min(xt));% Sampling frequency
L=length(xt);
freq=(0:L-1)*Fs/L+1e-10;
start=max(find(freq<2e11));
term=min(find(freq>1.6e12));  %frequecy range 0.2-1.6 THz
% term=start+104;
```

```
w=2*pi*freq;
Er=fft(r);
Es=fft(s);
Trans=Es./Er;
Trans=Trans(start:term);
freq=freq(start:term);
Magnitude=abs(Trans);
LM=log(Magnitude);
phase=-unwrap(angle(Trans));
p=mapminmax(phase',0,1);
v=[p,LM']';
f=freq;
end % end of function getVector
```

subfunction: prepareforGAclassifier

```
function Blade=prepareforGAclassifier(Vectorset)
% This function is used to get vectors that could be
%  used to form 2-blades and the modular of their vedge product

% input arguments:
%       Vectorset:    the vector set corresponding to all data samples;
% output arguments:
%       Blade:    pair of vector,v1 and v2, and the modulus of their wedge product ;

% randomly select 200 vector
index=randperm(400);
r=index(1:200);

trainData1=Vectorset{2}(:,r); % randomly select 200 vector of 1.5 mm thickness
trainData2=Vectorset{3}(:,r); % randomly select 200 vector of 2.0 mm thickness

% calculated mean vector
v1=mean(trainData1,2);
v2=mean(trainData2,2);

% calculated the modulus of the wedge product of v1 and v2
modulus=getModulus(v1,v2);

Blade={v1 v2 modulus};
end % end of the function prepareforGAclassifier
```

subfunction: getModulus

```
function Modulus=getModulus(v1,v2)
% This function is used to get the
% modulus of wedge product of v1 and v2

% input arguments:
%       v1,v2:    vectors are used to form the 2-blade;
% output arguments:
%      modulus:   the modulus of wedge product of v1 and v2;

m=length(v1);
MB=[];
for i=1:m-1
    for j=i+1:m
        MB=[MB,((v1(i)*v2(j)-v1(j)*v2(i)))^2];
    end
end
```

```
Modulus=sqrt(sum(MB));
end  % end of the function getModulus
```

subfunction: predictbyGA

```
function predictbyGA(VectorPair, Modulus)
% This function is used to create GA classifier and determine labels for all vector sets
% input arguments:
%       VectorPair:  pairs of vectors,  v1 and v2;
%       Modulus:    the modulus set of the wedge product of v1 and v2

load('GAvectorset.mat') % saved in the function GAclassifier

% vector set for substance melamine with five different thickness
Mel=VectorSets{1};
label=1;
for i=1: length(Mel)

 % compute the parameter lambda and classify based on the minimum lambda
    [P_Tag,A_Tag,Acc,lambda]=ClassifybyGA(Mel{i},label,VectorPair,Modulus);
    P_Mel{i}=P_Tag;           % Prediction tag
    A_Mel{i}=A_Tag;           % actual tag
    ACC_Mel{i}=Acc;           % accurancy
    D_Mel{i}=lambda;           % lambda of vector Mel to every plane
end

% vector set for substance tartaric acid with different thickness
Tar=VectorSets{2};
label=2;
for i=1: length(Tar)

     % compute the parameter lambda and classify based on the minimum lambda
    [P_Tag,A_Tag,Acc,lambda]=ClassifybyGA(Tar{i},label,VectorPair,Modulus);
    P_Tar{i}=P_Tag;       % prediction tag
    A_Tar{i}=A_Tag;       % actual tag
    ACC_Tar{i}=Acc;       % accurancy
    D_Tar{i}=lambda;               % lambda of vector Tar to every plane
end

% vector set for substance lactose with different thickness
Lac=VectorSets{3};
label=3;
for i=1: length(Lac)

     % compute the parameter lambda and classify based on the minimum lambda
    [P_Tag,A_Tag,Acc,lambda]=ClassifybyGA(Lac{i},label,VectorPair,Modulus);
    P_Lac{i}=P_Tag;    % prediction tag
    A_Lac{i}=A_Tag;   % actual tag
    ACC_Lac{i}=Acc;   % accurancy
    D_Lac{i}=lambda;           % lambda of vector Lac to every plane
end

% vector set for substance glucose with different thickness
Glu=VectorSets{4};
label=4;
for i=1: length(Glu)
      % compute the parameter lambda and classify based on the minimum lambda
    [P_Tag,A_Tag,Acc,lambda]=ClassifybyGA(Glu{i},label,VectorPair,Modulus);
    P_Glu{i}=P_Tag;   % prediction tag
    A_Glu{i}=A_Tag;   % actual tag
    ACC_Glu{i}=Acc;   % accurancy
    D_Glu{i}=lambda;     % lambda of vector Glu to every plane
```

```
end

% save the predict accuracy
eval(['save AccbyGA.mat P_Mel  A_Mel  ACC_Mel  D_Mel  P_Tar  A_Tar  ACC_Tar  D_Tar  P_Lac  A_Lac  ACC_Lac
D_Lac  P_Glu  A_Glu ACC_Glu D_Glu'])
end % end of the function predictbyGA
```

subfunction: classifybyGA

```
function  [P_Tag,A_Tag,Acc,Lambda]=ClassifybyGA(vectors,label,VectorPair,Modulus)
%  This function is used to  compute the lambda for a given
%  substance vector to every plane, and choose the substance
%  corresponding to plane with the lest value of lambda as its label.

% input arguments:
%       vectors:    vectors to be forecasted;
%       label:         actual tag of vectors;
%       VectorPair: four pairs of vectors, v1 and v2;
%       modulus:   four modulus of wedge product of v1 and v2;
% output arguments:
%       P_Tag:      prediction tag of vectors
%       A_Tag:      actual tag of vectors
%       Acc:          prediction accuracy
%       Lambda:   Lambda that be used to estimate vector deviaton from plane

dbstop if error
for i=1:length(Modulus)
    v1=VectorPair{i}(:,1);
    v2=VectorPair{i}(:,2);
    %compute parameters Lambda for a given vectors
    Lambda(i,:)=getLambda(v1,v2,vectors,Modulus{i});
end
L=size(Lambda,2);
A_Tag=label*ones(1,L);
[Lambda_min,P_Tag]=min(Lambda);
right=sum(P_Tag==A_Tag);
Acc=right/L;
end % end of the function ClassifybyGA
```

subfunction: getLambda

```
function Lambda=getLambda(v1,v2,vectors,modulus)
% This function is used to compute parameter Lambda
% input arguments:
%       v1, v2:    input vectors that could be used  to form the plane for a given substance.
%       vx:          vector set obtained  from unkown substance with unkown thickness;
%       modulus:   the modulus of wedge product of v1 and v2;
% output arguments:
%       D:            Lambda that be used to estimate vector deviaton from plane

Lambda=[];
[m,n]=size(vectors);
for i=1:n
    vx=vectors(:,i);
    Mvx=sqrt(sum(vx.*vx));  % compute the modulus of vx
    m=length(v1);
    P=[];
    p=[];
    for i=1:m
        for j=1:m
            if j~=i
```

```
                p=[p,-vx(j)*(v1(i)*v2(j)-v1(j)*v2(i))];
            else
            end
        end
        P=[P,sum(p)];
        p=[];
    end
    P=P/modulus;
    MP=sqrt(sum(P.*P)); % computer the modulus of vector P
    d=MP/Mvx;
    Lambda=[Lambda,1-sqrt(d^2)];
end
end %  end of function getLambda
```

Published with MATLAB? R2013a