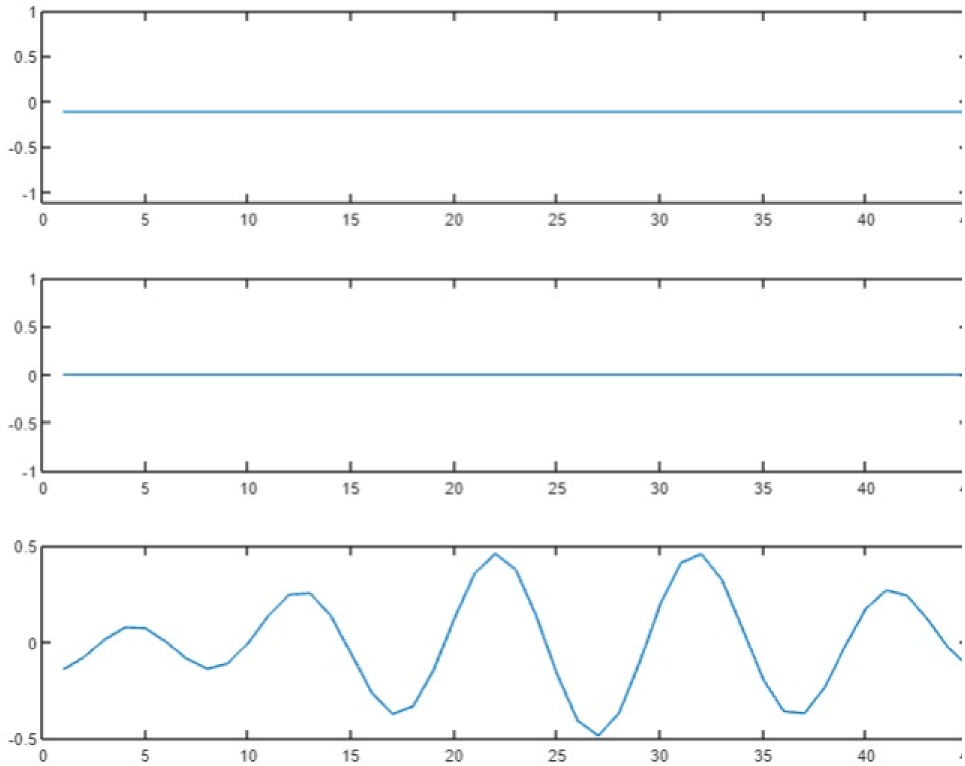


Android App Classification

30.09.2021

18EE35003
Divyansh Singh

Outline

- Using the DFT data of several readings train the SVM to classify input data
- Test the SVM over more input data
- Provide the user with output classes, viz,x-axis motion, y-axis motion, z-axis motion

Improvement

The task is to classify whether the data represents motion along x-axis , y-axis , z-axis or circular motion (on a horizontal surface).

The key observation

During motion along a single axis, the data from other axes should represent a near still waveform. While during circular motion the data from x and y directions should indicate oscillations while the data from the third direction would be more or less still .

Thus , all we need to do is train a SVM to classify whether there is oscillatory motion along one axis or not .

Advantages

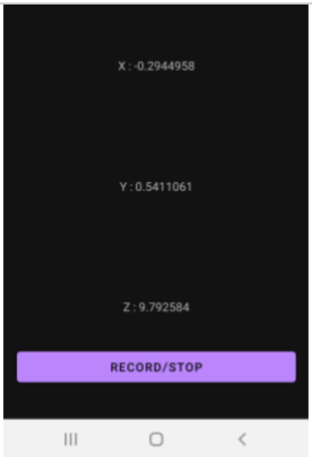
- More data (triple) to train the SVM
- Faster training
- Faster runtime

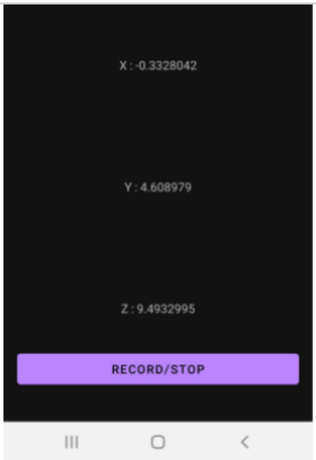
AIM

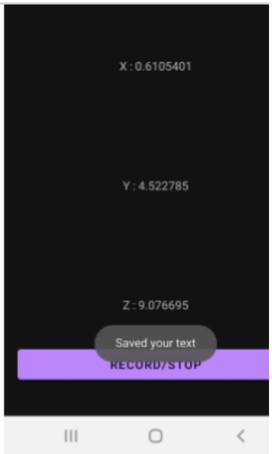
Aim is to classify the type of motion via individual classification along each axis . The idea is to sample certain (here, 40 - found via successive observations) acceleration values and process them to classify the type of motion .

Getting the data

Project proposal







CODE :

```
package com.example.firstapp;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.View;

import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.nio.charset.StandardCharsets;

public class MainActivity extends AppCompatActivity
implements SensorEventListener  {
   public static final String EXTRA_MESSAGE =
"com.example.myfirstapp.MESSAGE";
   private static final String TAG = "MainActivity";

   private Sensor accelero;
   private SensorManager SM;

   private TextView xtext, ytext, ztext;
   private int record_status;

   String fileName="sensordata.txt";
   String baseDir =
Environment.getExternalStorageDirectory().getAbsolutePath();
   String pathDir = baseDir +
"/Android/data/com.mypackage.firstApp/";
   File file;
   File gpxfile;
   FileWriter writer;
   private  int t;
   private OutputStreamWriter outputWriter;
```

```
        SM = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
//        accelero =
SM.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
       accelero =
SM.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

       SM.registerListener(this, accelero,
SensorManager.SENSOR_DELAY_NORMAL);
       Log.d(TAG, "onCreate: registered accelerometer
listener");
       xtext = (TextView)findViewById(R.id.xtext);
       ytext = (TextView)findViewById(R.id.ytext);
       ztext = (TextView)findViewById(R.id.ztext);
       record_status = 0;
       t = 0;
       file=new
File(MainActivity.this.getFilesDir(),"Kalman");
       if (!file.exists()) {
           file.mkdir();
       }
   }
   @Override
   public void onSensorChanged(SensorEvent event) {
       xtext.setText("X : " + event.values[0]);
       ytext.setText("Y : " + event.values[1]);
       ztext.setText("Z : " + event.values[2]);
       if(record_status == 1) {
           String str = event.values[0] + "," +
event.values[1] + "," + event.values[2] + "," + ++t + "\n";
           try {
               writer.append(str);
               writer.flush();
           } catch (Exception e) {
               e.printStackTrace();
           }
       }


   }

   @Override
   public void onAccuracyChanged(Sensor sensor, int i) {

   }

   public void send_message(View view){
       Intent intent = new Intent(this,
DisplayMessageActivity.class);
       EditText editText = (EditText)
findViewById(R.id.editTextTextPersonName);
       String message = editText.getText().toString();
       intent.putExtra(EXTRA_MESSAGE, message);
       startActivity(intent);
   }

   public void do_record_stop(View view){
       EditText editText = (EditText)
findViewById(R.id.editTextTextPersonName);
       if(record_status%2 == 0 ) {

           editText.setText("Recording");

           try {
               gpxfile=new File(file,"sensordata.txt");
               writer=new FileWriter(gpxfile);

           } catch(Exception e){

               editText.setText(e.getMessage());
               e.printStackTrace();
           }
```

```
text",
                        Toast.LENGTH_SHORT).show();
            } catch (Exception e) {
                editText.setText(e.getMessage());
                e.printStackTrace();
            }
            record_status = 2;
        }
    }
}
```

Processing the data in MATLAB to observe usability of the same

To save the data as csv for analysis in other apps, convert to a matfile for further use

```
d = load('long1.txt');
writematrix(d, 'd.csv')
T=readtable('d.csv');

a=T{:,1:3};
t=T{:,4};
save('testmat.mat','a','t')
```

To run the Kalman filter and save the data (in multiples of 40)

```
load testmat.mat
% Initial Guess
% state
x=randn(9,1);
% Covariance
P=eye(9);
% Process Noise covariance Q
q = 0.0001;
Q= q*eye(9);
% Measurement Noise covariance R
r = 0.01;
R= r*eye(3);
X=[]; Z=[]; Gain=[]; Err=[];
N=length(t);
% Construct H matrix
   H=[zeros(3,6) eye(3)];
for n=1:N-1
   h=t(n+1)-t(n);
   h2=h^2/2;
   % Construct Phi matrix
   phi=[eye(3)     h*eye(3) h2*eye(3)
           zeros(3)     eye(3)
 h*eye(3)
           zeros(3)     zeros(3)
 eye(3)];

   %  Compute the Kalman Gain K
   K=P*H'*inv(H*P*H'+R);

   % Update the states
   z=a(n,:)';
   err=(z-H*x);
   x=x+K*err;
```

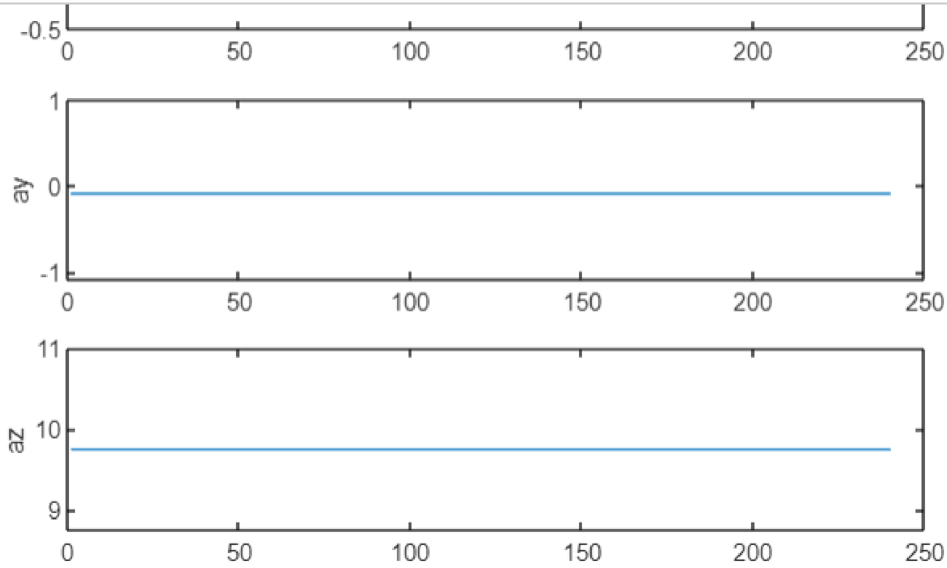
```
    % Project Ahead
    x=phi*x;
    P=phi*P*phi'+Q;
end
ae=[X(:,7:9);a(end,:)];
subplot(311)
plot([a(:,1) ae(:,1)])
ylabel("a_x")
legend("Sampled Data", "Filtered
Data", Location="best")
grid on
title("Q = " + q + "  R = " + r)
subplot(312)
plot([a(:,2) ae(:,2)])
ylabel("a_y")
legend("Sampled Data", "Filtered
Data", Location="best")
grid on
subplot(313)
plot([a(:,3)-9.8 ae(:,3)-9.8])
plot([a(:,3) ae(:,3)])
ylabel("a_z")
legend("Sampled Data", "Filtered
Data", Location="best")
grid on
figure()
plot([ae(:,1) ae(:,2) ae(:,3)-9.8])
plot([ae(:,1) ae(:,2) ae(:,3)])
legend()
shg
ae = ae(1:40*floor(length(ae)/40), :);
save('otput.mat','ae')
x = ae(1:40, :);
save('unopoint.mat', 'x')
```

THE KEY POINT IS TO OBSERVE THE DIFFERENCE FIRST (WHETHER THE DATA IS THEORETICALLY SUFFICIENT TO CLASSIFY OR NOT)

I observed such distinctions produced by :

- Taking fourier transform of accelerometer data along each axis
- Derive the magnitudes of fourier coefficients using log operation
- Threshold the Fourier Space data using magnitude as criteria
- Taking the Inverse fourier transform

For x-axis motion :



CODE :

```
load otput.mat
thresh = 1;
x = ae(:, 1);
y = fft(x);
mag = real(log(y));
%y(mag ~= max(mag(1:length(mag)))) =
0;
y(mag < thresh) = 0;
ne = ifft(y);
subplot(311)
plot(ne)
ylabel('ax')
x = ae(:, 2);
y = fft(x);
mag = real(log(y));
%y(mag ~= max(mag(1:length(mag)))) =
0;
y(mag < thresh) = 0;
ne = ifft(y);
subplot(312)
plot(ne)
ylabel('ay')
x = ae(:, 3);
y = fft(x);
mag = real(log(y));
%y(mag ~= max(mag(1:length(mag)))) =
0;
y(mag < thresh) = 0;
ne = ifft(y);
subplot(313)
plot(ne)
ylabel('az')
%save('p.mat', 'ne')
```

KEY INFERENCE :

- The Fourier space data is sufficient for proper classification
- Only the magnitude of Fourier space data can be used for complete classification

Now the data is sorted in lengths of 40

```
is_z = -1;
t = ae(:, 1);
z = reshape(t, [40
floor(length(t)/40)]);
z = transpose(z);
%z(1, :)
%z(2, :)
T = table(z);
a = rowfun(@fft, T);
outa = 1:height(a);
outa = outa*0 + is_x;
a = a{:, :};
a = log(a);
a = real(a);
t = ae(:, 2);
z = reshape(t, [40
floor(length(t)/40)]);
z = transpose(z);
T = table(z);
b = rowfun(@fft, T);
outb = 1:height(b);
outb = outb*0 + is_y;
b = b{:, :};
b = log(b);
b = real(b);
t = ae(:, 3);
z = reshape(t, [40
floor(length(t)/40)]);
z = transpose(z);
T = table(z);
c = rowfun(@fft, T);
outc = 1:height(c);
outc = outc*0 + is_z;
c = c{:,:};
c = log(c);
c = real(c);
out = transpose([outa outb outc]);
input_data = [a ; b ; c];
save('p.mat', 'input_data', 'out')
```

Next the SVM is designed and solved using quadprog

```
%% Design SVM
H = zeros(n,n);
for i=1:n
   for j=i:n
       H(i,j) =
y(i)*y(j)*x(:,i)'*x(:,j);
       H(j,i) = H(i,j);
   end
end
f = -ones(n,1);
Aeq=y;
beq=0;
lb=zeros(n,1);
ub=C*ones(n,1);
Alg{1}='trust-region-reflective';
Alg{2}='interior-point-convex';
options=optimset('Algorithm',Alg{2},...
   'Display','off',...
   'MaxIter',20);
alpha=quadprog(H,f,[],[],Aeq,beq,lb,ub,
[],options)';
AlmostZero=(abs(alpha)
<max(abs(alpha))/1e5);
alpha(AlmostZero)=0;
```

Project proposal