

In The Name of God



Sharif University of Technology

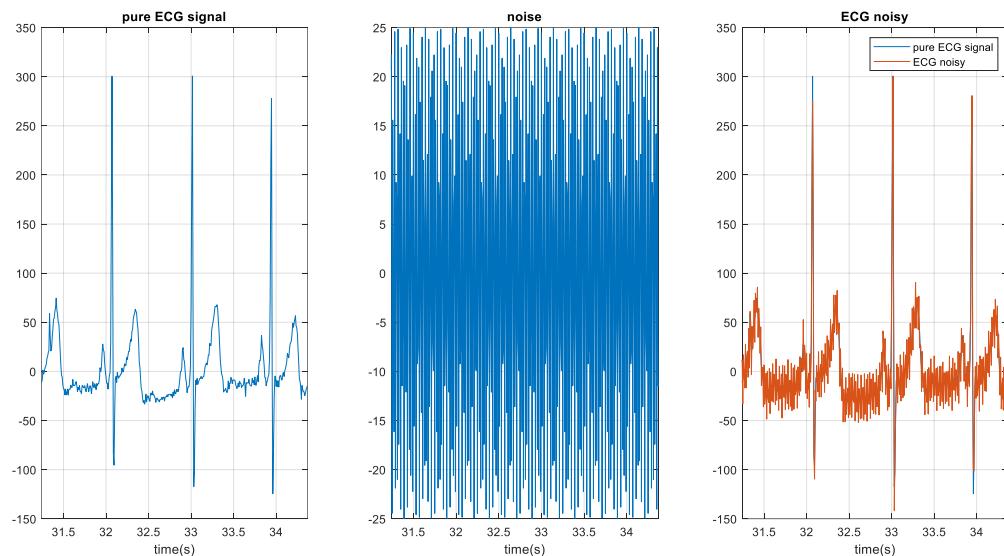
Dr. Shamsolahi

Amirreza Hatamipour

97101507

بخش یک:

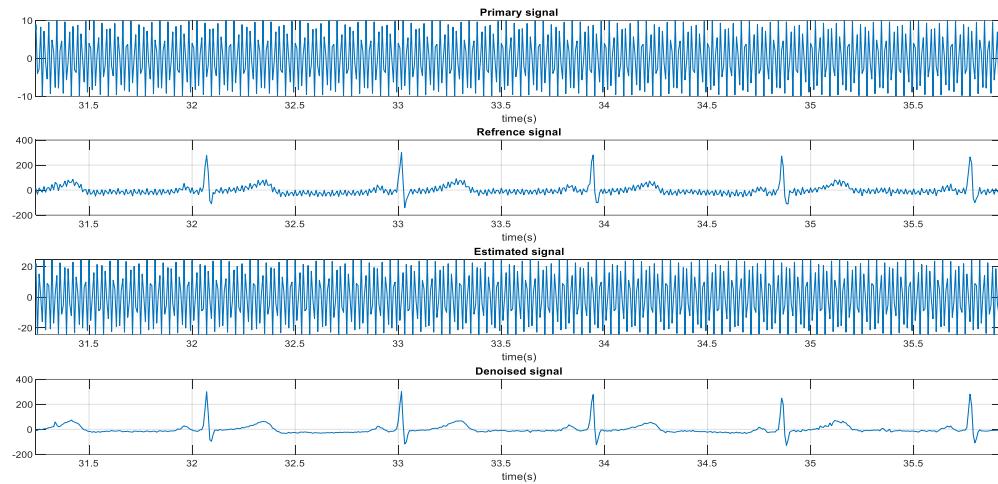
الف) ابتدا سیگنال موردنظر را با فرکانس گفته شده تولید می کنیم و به سیگنال خالص قلب اضافه می کنیم:



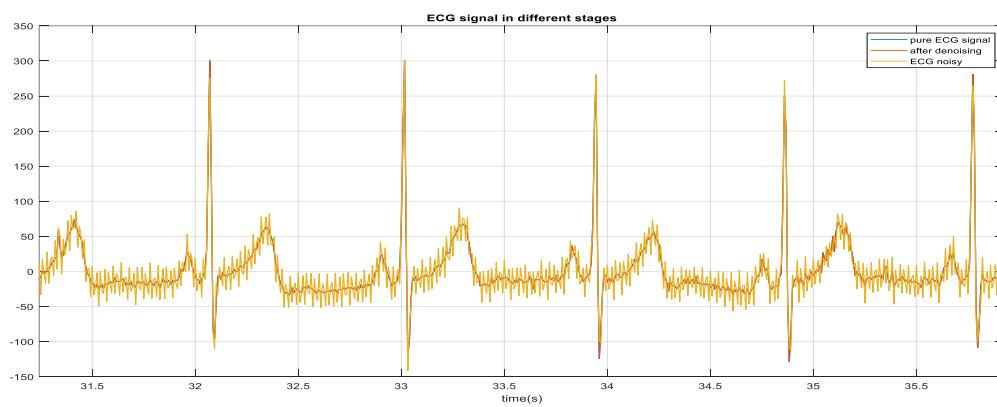
برای حذف نویز برق شهر، از الگوریتم فیلتر وقفی استفاده می کنیم:

```
function [X2 , w] = AdaptiveFilter(X,d,M,w0,mu)
    N = numel(d);
    error = 1;
    w = w0;
    while error > 0.0000001
        w_n = w;
        for i = M+1:N
            X_v = X(1, i: -1 : i - M )';
            delta = (d(i) - w' * X_v) / (X_v' * X_v);
            w = w + 2 * mu * delta * X_v;
        end
        error = sum((w - w_n), 'all');
    end
    X2 = zeros(N,1);
    for i = M+1:N
        X2(i) = w' * X(1, i: -1 : i - M );
    end
end
```

به این صورت که سیگنال قلبی آغشته شده به نویز برق شهر را بعنوان $d[n]$ و $x[n]$ به فیلتر می‌دهیم. سیگنالی با همون فرکانس اما با دامنه متفاوت را بعنوان primary به فیلتر می‌دهیم. طول فیلتر را 200 در نظر می‌گیریم و با توجه به مقدار خطای موجود، به حذف نویز می‌پردازیم.



در دو شکل اول تصویر بالا، دو سیگنال داده شده به فیلتر را مشاهده می‌کنیم. همچنین در شکل سوم، نتیجه حاصل از تخمین نویز مورد نظر ما توسط فیلتر وقی را مشاهده می‌کنیم که اگر سیگنال نویزی را از این نویز تخمین زده شده کم کنیم به سیگنال قلبی حذف نویز شده خواهیم رسید. در شکل زیر می‌توان سه سیگنال نویزی، حذف نویز شده و سیگنال خالص اولیه را مشاهده کرد:



همچنین برای مقایسه نتیجه، از معیار RRMSE استفاده می کنیم:

$$RRMSE = \frac{\sqrt{\sum_{n=1}^{32} \sum_{t=1}^T \left(x_{org}^{(n)}(t) - x_{den}^{(n)}(t) \right)^2}}{\sqrt{\sum_{n=1}^{32} \sum_{t=1}^T \left(x_{org}^{(n)}(t) \right)^2}}$$

که برای این سیگنال بصورت زیر می شود:

```
RRMSE_before =
```

```
0.4361
```

```
RRMSE_after =
```

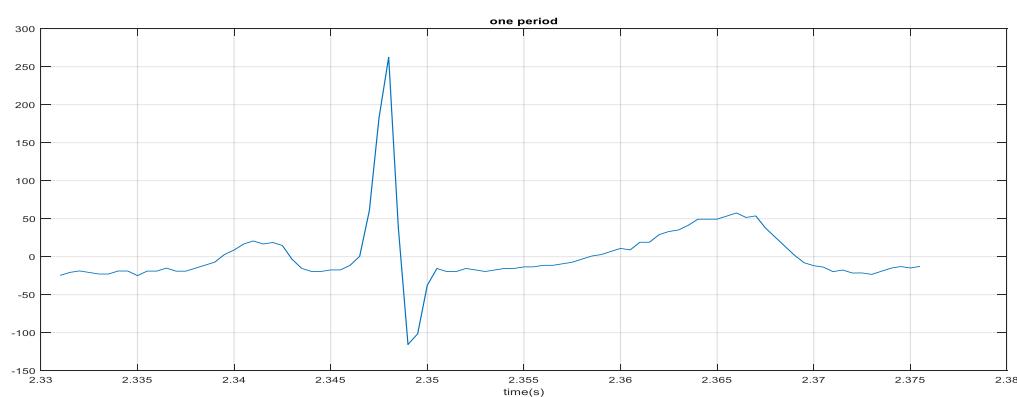
```
0.0553
```

```
RRMSE_improvement =
```

```
0.3808
```

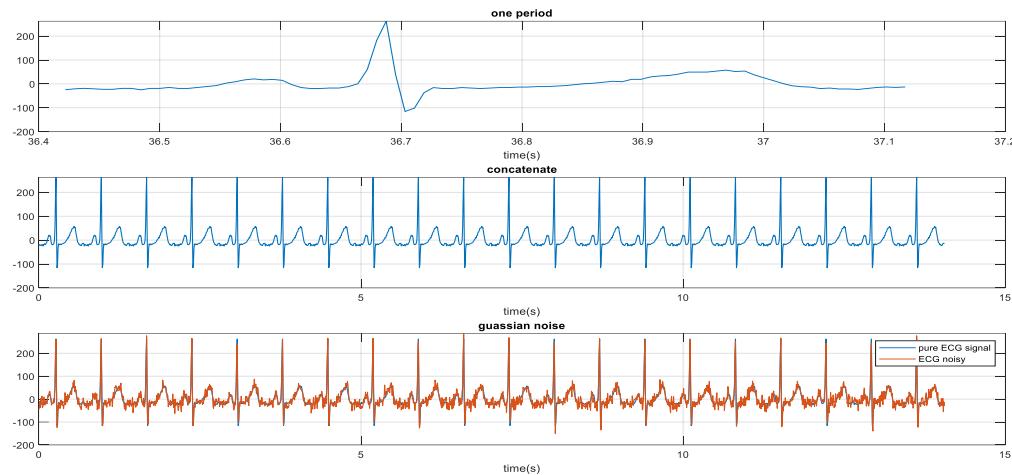
همانطور که مشاهده می کنیم فیلتر ما توانست مقدار خوبی از نویز موجود را حذف کند و نتجه حاصل بسیار نزدیک به سیگنال بدون نویز می باشد.

ب) یک سیکل از سیگنال را بصورت زیر جدا می کنیم:

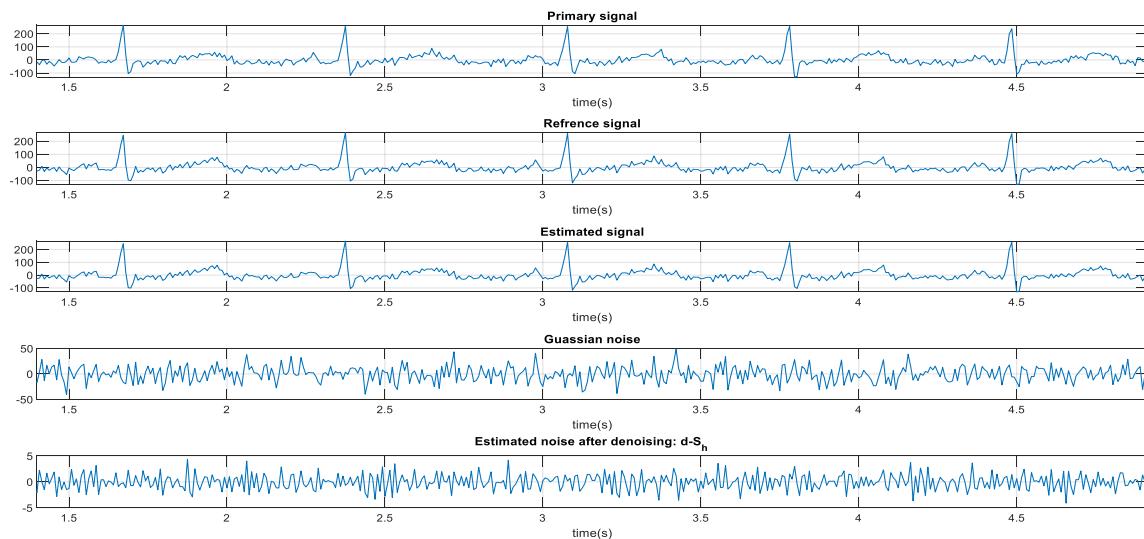


حال بیست قطعه از آن را بهم می چسبونیم و به آن نویز اضافه می کنیم. سیگنال زیر حاصل می

شود:

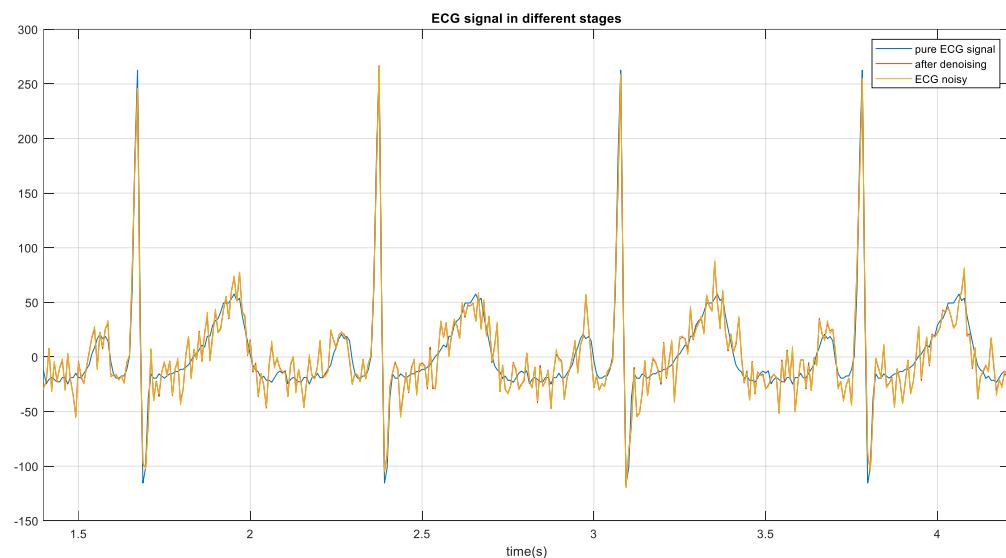


حال از اونجایی که نویز ما سفید می باشد، لذا اگر آنرا شیفت دهیم هیچ همبستگی بین آن و نسخه شیفت یافته آن وجود ندارد. لذا در اینجا یک نسخه شیفت یافته سیگنال نویزی به اندازه یک دوره تناوب را بعنوان سیگنال **primary** در نظر می گیریم.



در اینجا فیلتر وقفی سعی دارد سیگنال قلبی حذف نویز شده را تخمین بزند که در شکل سوم نتیجه این تخمین را مشاهده می کنید. همچنین در دو شکل آخر نتیجه حاصل از نویز اضافه شده به سیگنال و همچنین نویز حاصل از کم کردن تخمین فیلتر وقفی از سیگنال نویزی را مشاهده می کنیم.

در شکل زیر می توان سه سیگنال نویزی، حذف نویز شده و سیگنال خالص اولیه را مشاهده کرد:

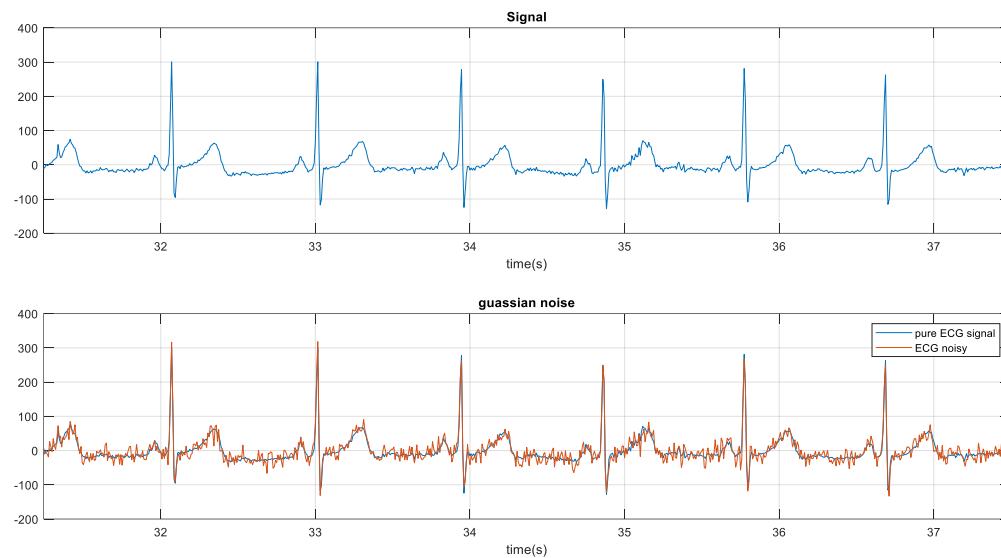


و همچنین معیار های گفته شده در این قسمت برابر است با:

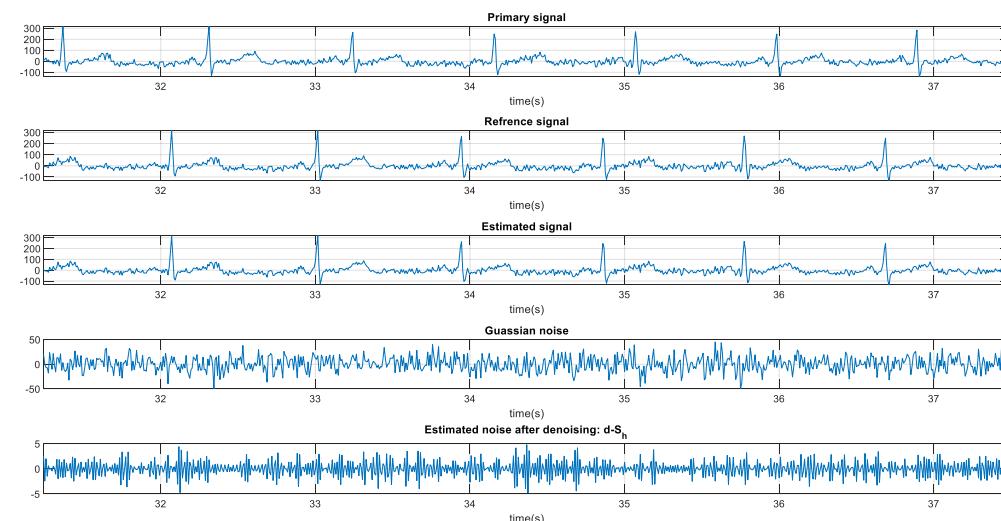
```
RRMSE_before =  
0.3379  
  
RRMSE_after =  
0.3921  
  
RRMSE_improvement =  
-0.0543
```

همانطور که می بینیم در این قسمت فیلتر نتوانست کمکی در حذف نویز بکند.

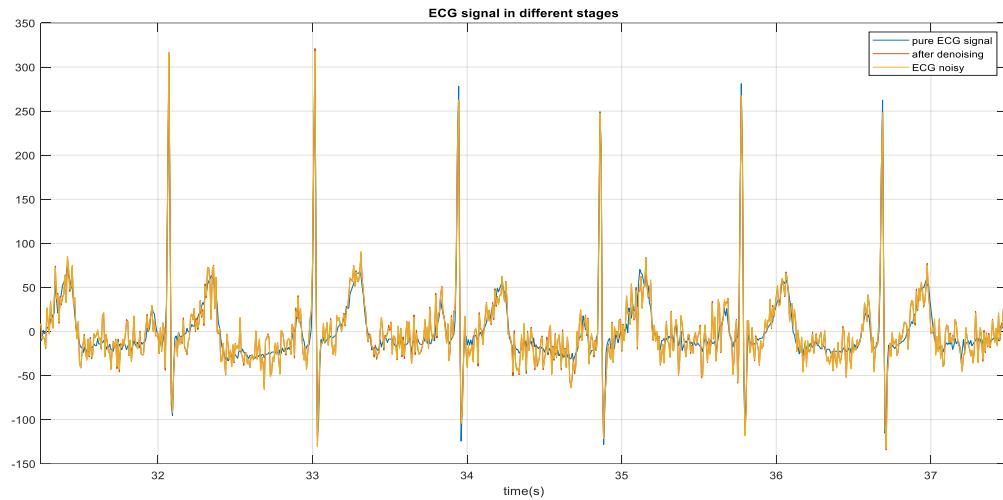
ج) حال سیگنال اصلی را در نظر می گیریم و به آن نویز گاووسی اضافه می کنیم:



مشابه قسمت قبل، سیگنال شیفت یافته را بعنوان ورودی به فیلتر می دهیم و نتایج زیر را می گیریم:



در شکل زیر می توان سه سیگنال نویزی، حذف نویز شده و سیگنال خالص اولیه را مشاهده کرد:



همانطور که می بینیم در این قسمت فیلتر نتوانست کمکی در حذف نویز بکند.

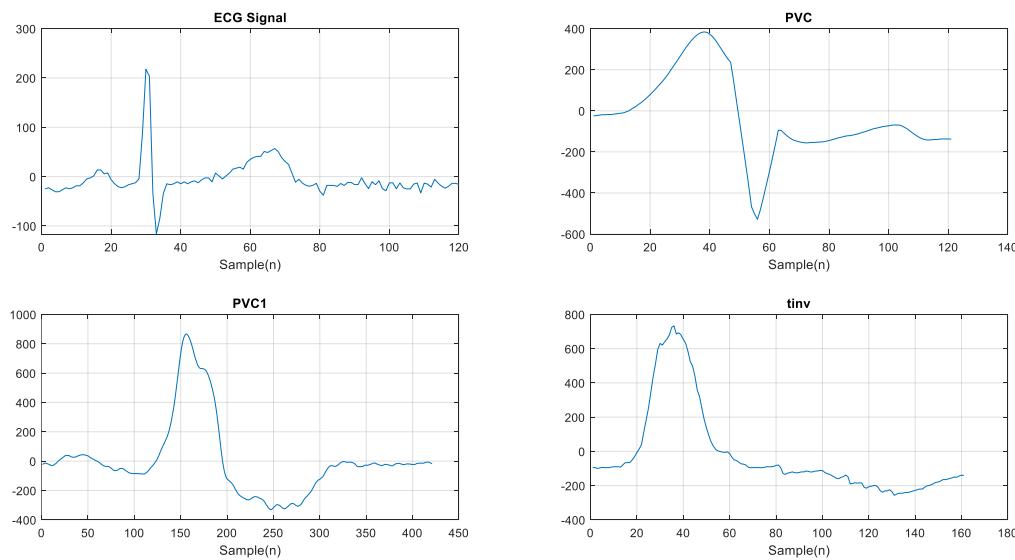
```
RRMSE_before =  
0.3713  
  
RRMSE_after =  
0.3852  
  
RRMSE_improvement =  
-0.0138
```

همانطور که می بینیم در این قسمت فیلتر نتوانست کمکی در حذف نویز بکند.

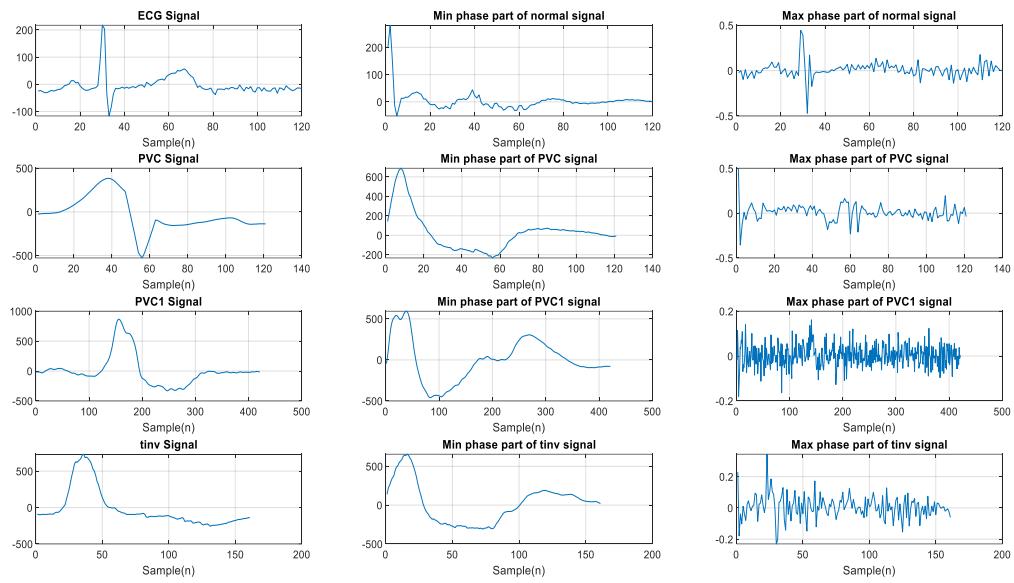
از آنجایی که سیگنال دیگر متناوب یکسان بصورت قسمت قبل نمی باشد، نتیجه حاصل بدتر از حالت قبل می شود.

بخش دو:

الف) سه سیگنال بیمار و یک سیگنال سالم را در شکل زیر رسم می کنیم:



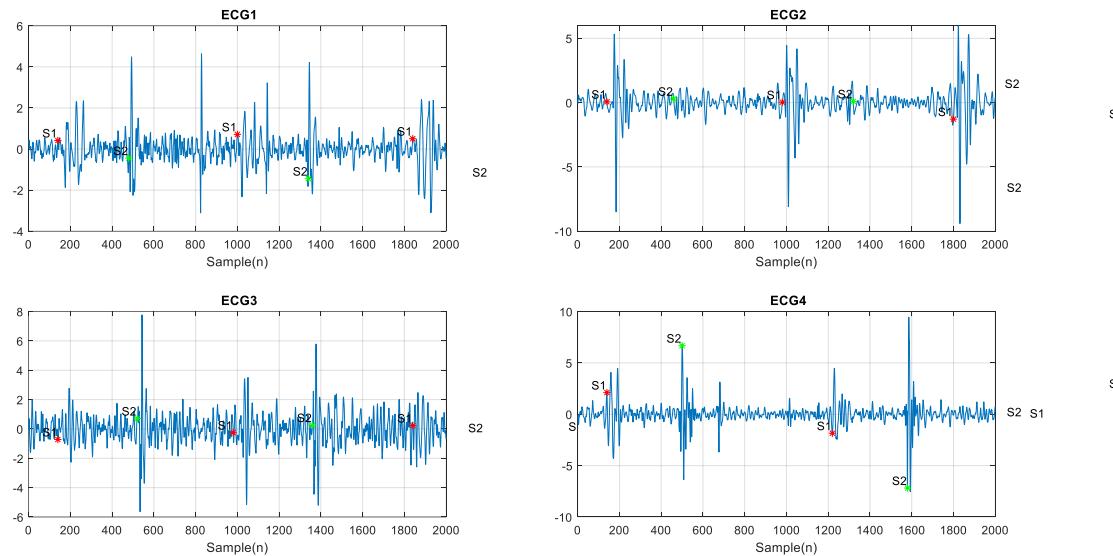
حال برای جدا کردن قسمت مینیمم فاز از ماکسیمم فاز، ابتدا با استفاده از دستور `rceps` قسمت مینیمم فاز رو در حوزه زمان بدست می آوریم. سپس با استفاده از دستور `cceps` کپستروم مختلط سیگنال اصلی و سیگنال مینیمم فاز را محاسبه کرده، سپس از هم در حوزه کپستروم کم می کنیم و در نهایت از آن کپستروم معکوس می گیریم که در حوزه زمان به سیگنال ماکسیمم فاز می رسیم:



همانطور که در شکل بالا مشاهده می کنیم، با توجه به قسمت مینیمم فاز می توانیم یک دسته بندی بین سیگنال های سالم و مریض انجام دهیم. همونطور معلوم است سیگنال های مریض دارای قله هایی با تاخیر بیشتر نسبت به سیگنال سالم می باشد و همچنین با سرعت کمتری افت می کنند. از روی اطلاعات در بازه های میانی سیگنال هم می توان بین خود سیگنال های مریض هم دسته بندی انجام داد.

بخش سوم:

الف) بازه هایی از سیگنال مورد نظر را رسم می کنیم و از روی annotation های داده شده مکان آنها را در 20 ضرب کرده و نشان می دهیم:



S1 و صدای دوم قلب (S2، صدای قلب دیاستولیک) اجزای طبیعی چرخه قلبی هستند. درست پس از شروع سیستول رخ می دهد و عمدتاً به دلیل بسته شدن میترال است.

ب) در این قسمت برای ذخیره ویژگی ها و لیبل ها کد را بصورت زیر تکمیل می کنیم:

```
[for PCGi = 1:length(PCGCellArray)

    PCG_audio = PCGCellArray{PCGi};

    S1_locations = annotationsArray{PCGi,1};
    S2_locations = annotationsArray{PCGi,2};

    [PCG_Features, featuresFs] = getSpringerPCGFeatures(PCG_audio, Fs);

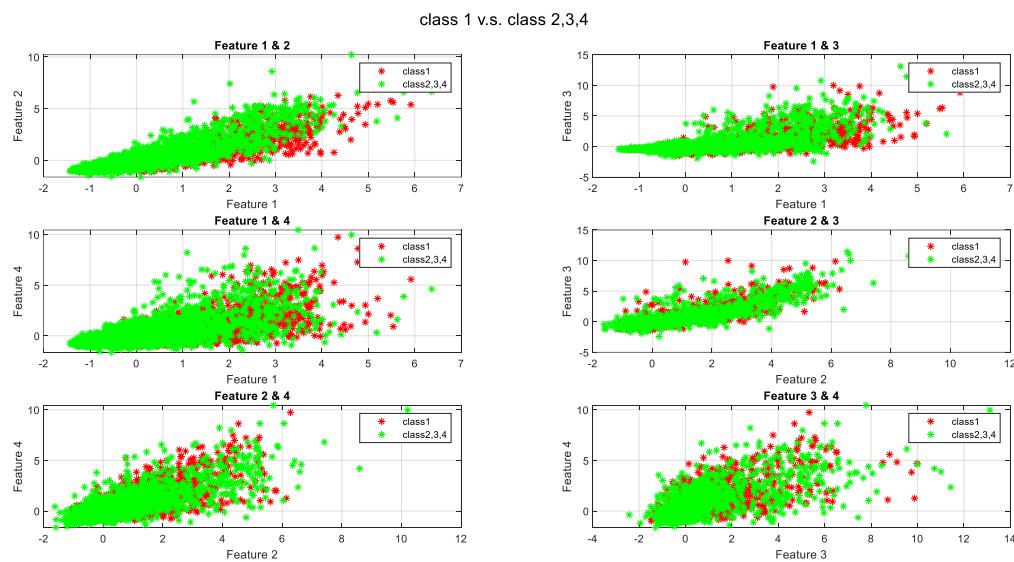
    PCG_states = labelPCGStates(PCG_Features(:,1),S1_locations, S2_locations, feature
    % your code here:
    PCG_Features_all = [PCG_Features_all ; PCG_Features];
    PCG_states_all = [PCG_states_all ; PCG_states];
end

save('PCG_Features_all.mat','PCG_Features_all')
save('PCG_states_all.mat','PCG_states_all')
```

ج) برای هر کلاس بطور جداگانه ابتدا با استفاده از معیار فیشر تک بعدی، دو ویژگی برتر را انتخاب می کنیم، سپس با استفاده از مراکز دو کلاس جدید ایجاد شده، مرز طبقه بند که عمود منصف مراکز می باشد برای هر کدام از کلاس ها رسم می کنیم.

کلاس یک:

داده های دو کلاس بر حسب تمام حالات ویژگی های ممکن بصورت زیر می باشد:

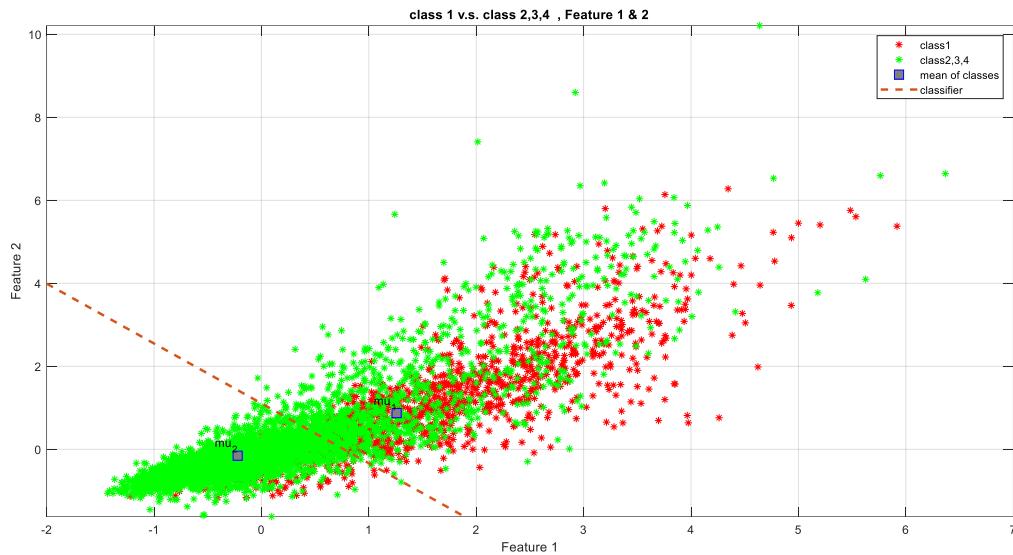


معیار فیشر برای هر ویژگی بصورت زیر می شود:

$J =$

0.7745 0.3207 0.1694 0.1958

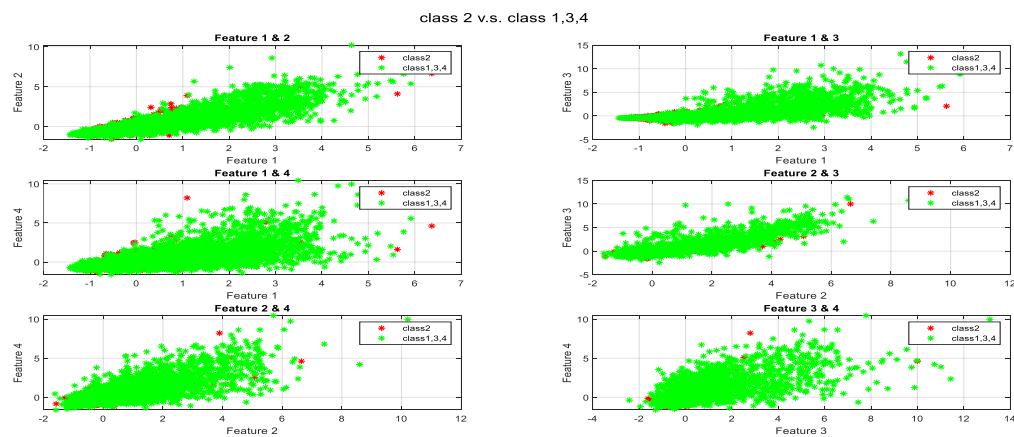
که همانطور که مشاهده می کنیم برای دو ویژگی اول مقادیر بیشتری می باشد. حال مرز طبقه بند ما بصورت زیر می شود:



همانطوری که مشاهده می کنیم مرز طبقه بند تشخیص داده شده دارای خطای زیادی می باشد چون اولاً دو ویژگی برای طبقه بندی کم می باشد و ثانیاً طبقه بند ما یک طبقه بند ساده می باشد.

کلاس دو:

داده های دو کلاس بر حسب تمام حالات ویژگی های ممکن بصورت زیر می باشد:

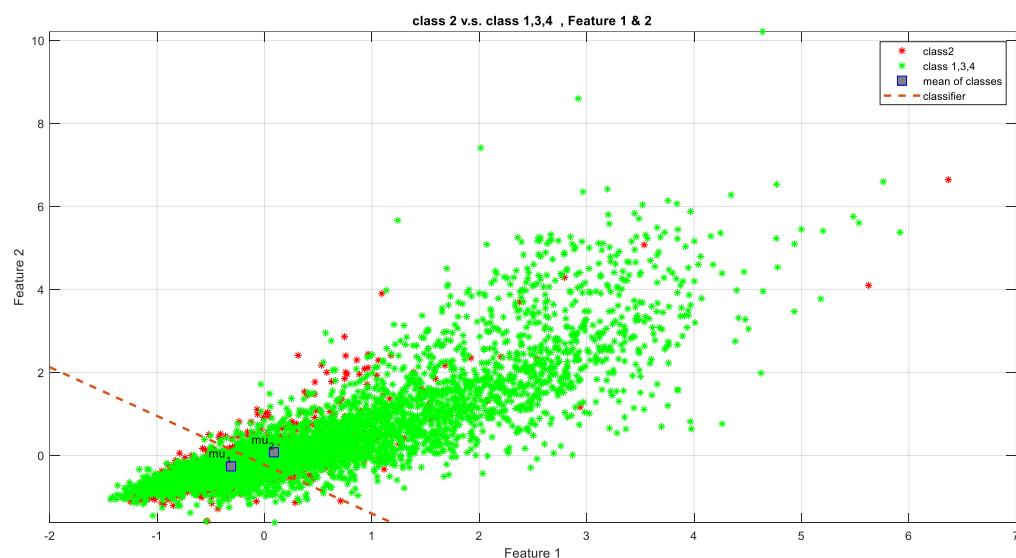


معیار فیشر برای هر ویژگی بصورت زیر می شود:

$J =$

0.0736 0.0524 0.0477 0.0359

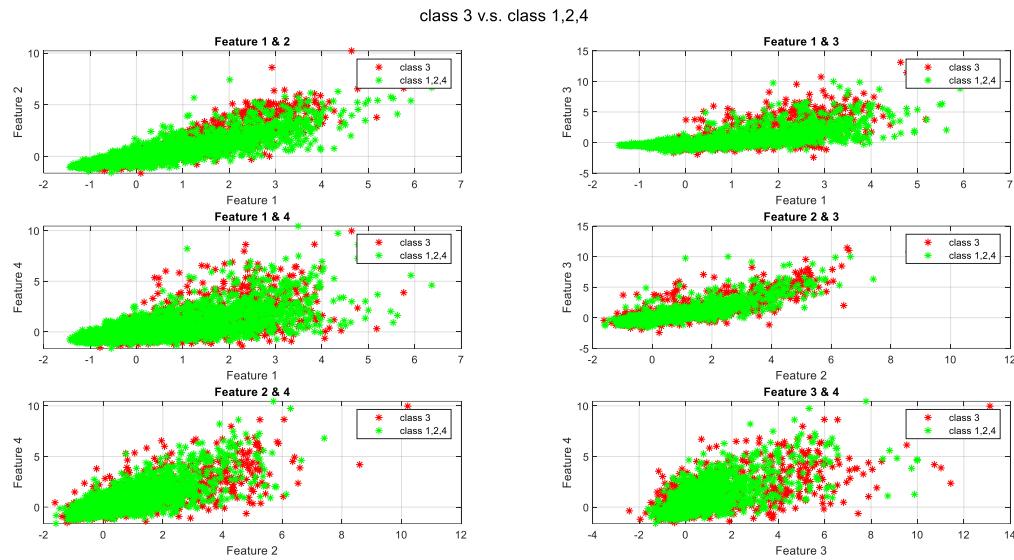
که همانطور که مشاهده می کنیم برای دو ویژگی اول مقادیر بیشتری می باشد گرچه تمام ویژگی های مقدار کمی دارند که با توجه به شکل بالا این مقادیر قابل حدس بود. حال مرز طبقه بند ما بصورت زیر می شود:



همانطوری که مشاهده می کنیم مرز طبقه بند تشخیص داده شده دارای خطای زیادی می باشد چون اولاً دو ویژگی برای طبقه بندی کم می باشد و ثانیاً طبقه بند ما یک طبقه بند ساده می باشد.

کلاس سه:

داده های دو کلاس بر حسب تمام حالات ویژگی های ممکن بصورت زیر می باشد:

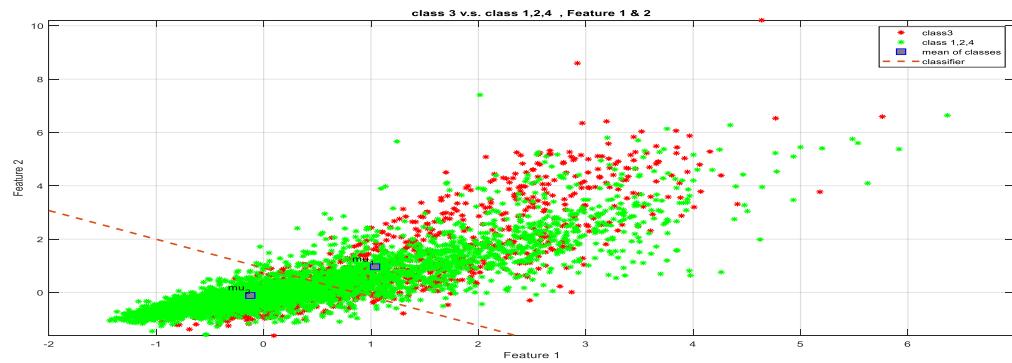


معیار فیشر برای هر ویژگی بصورت زیر می شود:

$J =$

0.5075 0.2733 0.1554 0.1931

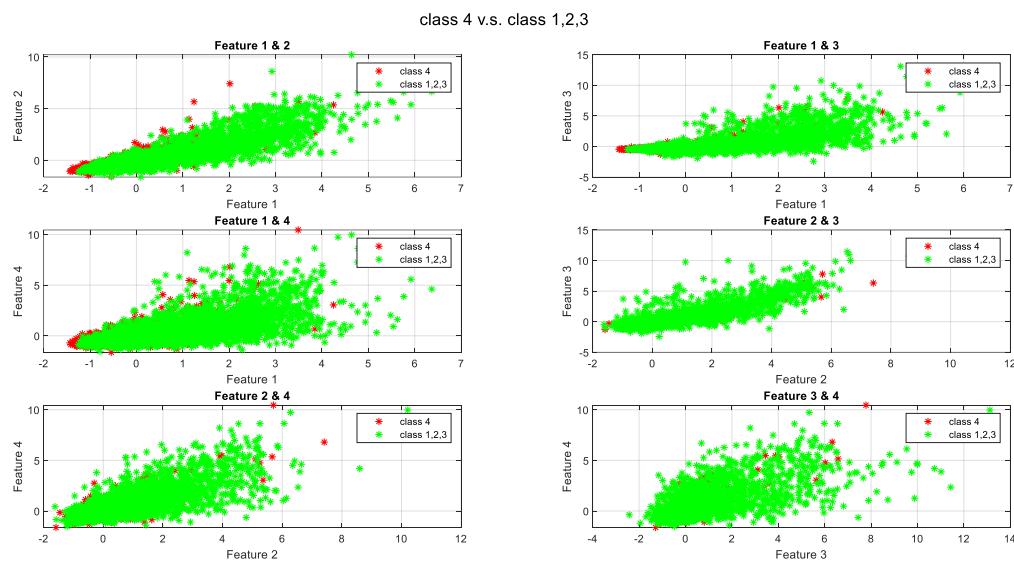
که همانطور که مشاهده می کنیم برای دو ویژگی اول مقادیر بیشتری می. حال مرز طبقه بند ما بصورت زیر می شود:



همانطوری که مشاهده می کنیم مرز طبقه بند تشخیص داده شده دارای خطای زیادی می باشد
چون اولاً دو ویژگی برای طبقه بندی کم می باشد و ثانیاً طبقه بند ما یک طبقه بند ساده می
باشد

کلاس چهارم:

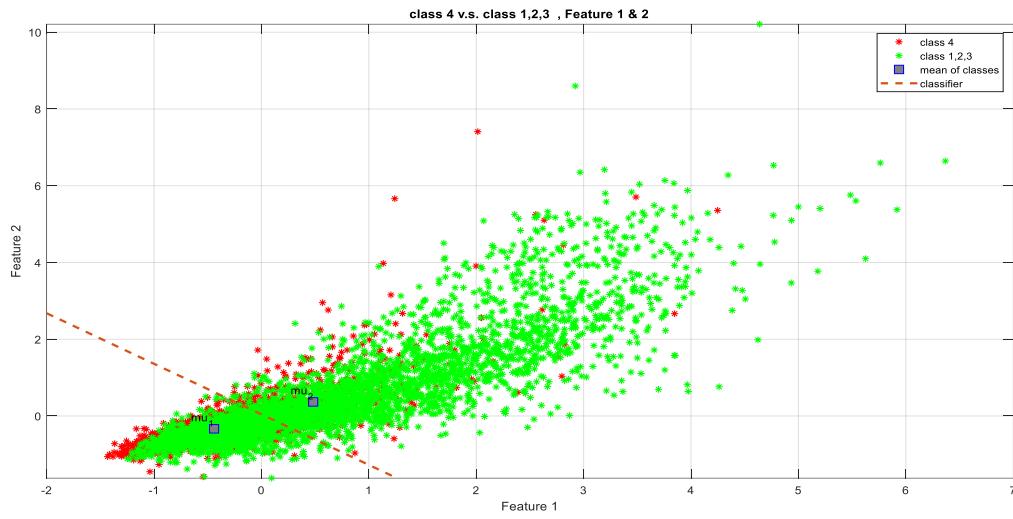
داده های دو کلاس بر حسب تمام حالات ویژگی های ممکن بصورت زیر می باشد:



داده های دو کلاس بر حسب تمام حالات ویژگی های ممکن بصورت زیر می باشد:

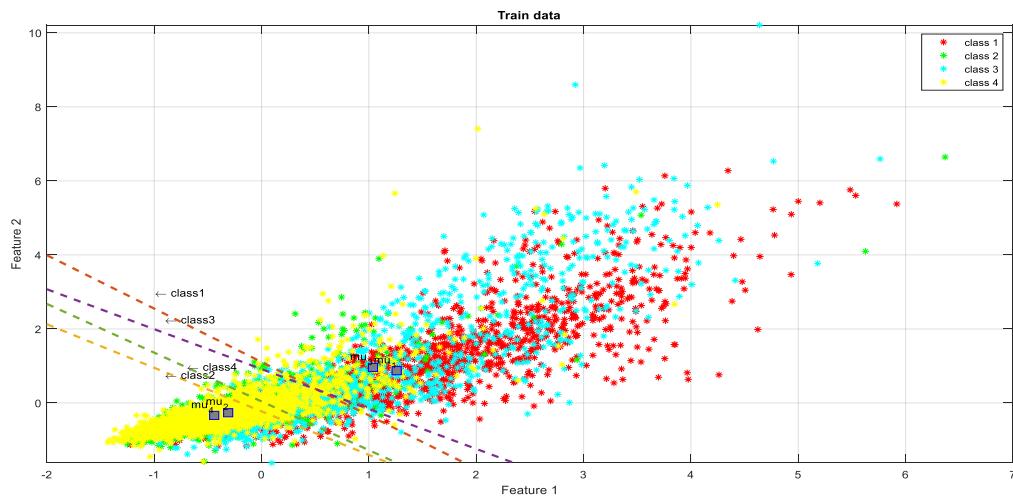
$$\text{J} = \begin{matrix} 0.2620 & 0.1354 & 0.0777 & 0.0924 \end{matrix}$$

که همانطور که مشاهده می کنیم برای دو ویژگی اول مقادیر بیشتری می باشد گرچه تمام ویژگی های مقدار کمی دارند که با توجه به شکل بالا این مقادیر قابل حدس بود. حال مرز طبقه بند ما بصورت زیر می شود:



همانطوری که مشاهده می کنیم مرز طبقه بند تشخیص داده شده دارای خطای زیادی می باشد چون اولاً دو ویژگی برای طبقه بندی کم می باشد و ثانیاً طبقه بند ما یک طبقه بند ساده می باشد.

حال از آنجایی که در هر چهار حالت دو ویژگی اول بعنوان ویژگی برتر انتخاب شده اند، می توانیم مرز های طبقه بندی را به همراه برچسب داده ها در یک شکل بصورت زیر رسم کنیم:

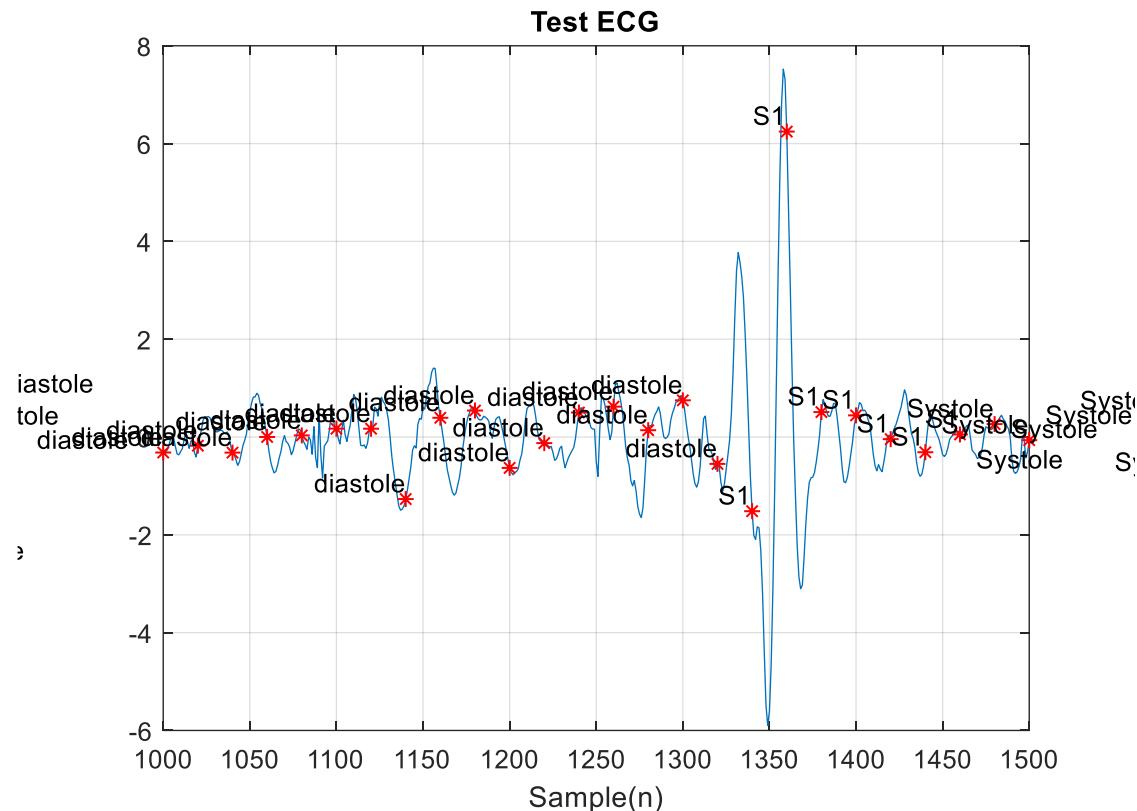


همانطوری که از مرز کلاس ها و همچنین داده ها مشخص است تفاوت چندانی بین کلاس دو با چهار و همچنین یک با سه وجود ندارد.

د) ابتدا با توجه به قطعه کد موجود، ویژگی ها و همچنین برچسب داده های تست را استخراج می کنیم. حال با استفاده از مرز های بدست آمده می توانیم به طبقه بندی سیگنال تست بپردازیم. همانطور که در بخش قبل گفتیم، نمی دسته بندی دقیقی بین دو کلاس یک با سه و همچنین دو با چهار قائل شد. لذا بعد از تعیین نقاط که در کدام طرف مرز کدام طبقه بند قرار می گیرند، با توجه به اینکه به مرکز کلاس نزدیک تر است طبقه بندی را انجام می دهیم و نتیجه بصورت زیر می شود:

		Confusion Matrix				
		1	2	3	4	
Output Class	1	19 1.1%	15 0.9%	42 2.5%	40 2.4%	16.4% 83.6%
	2	17 1.0%	191 11.2%	18 1.1%	553 32.5%	24.5% 75.5%
3	3	104 6.1%	7 0.4%	39 2.3%	28 1.6%	21.9% 78.1%
	4	63 3.7%	132 7.8%	46 2.7%	386 22.7%	61.6% 38.4%
		9.4% 90.6%	55.4% 44.6%	26.9% 73.1%	38.3% 61.7%	37.4% 62.6%
		↖	↗	↘	↗	↖
		Target Class				

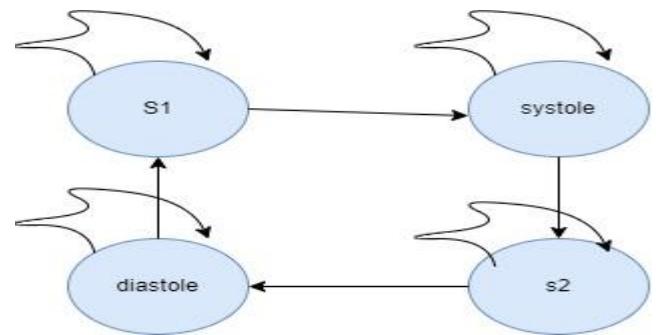
همانطور که انتظار داشتیم، misclassification بین دو کلاس گفته شده بسیار زیاد می باشد.
حال برچسب های تخمین زده شده را به فرکانس اصلی 1000 هرتز بر می گردانیم و بصورت زیر
قسمتی از آن را نمایش می دهیم:



بخش چهارم:

الف) ابتدا ویژگی ها و لیبل های ذخیره شده در بخش قبل را لود می کینم. حال برای مدل مارکوف، چهار state معادل چهار کلاسمون در نظر می گیریم. برای تخمین ماتریس transition، از برچسب داده های ترین استفاده می کنیم. به این ترتیب که برای کل داده های آموزش، تعداد تغییر استیت را می شماریم و در درایه مربوط می گذاریم. ماتریسی مشابه زیر حاصل می شود:

Transition_Matrix				
1	2	3	4	
1356	226	0	0	
0	2115	226	0	
0	0	904	226	
226	0	0	5295	



که با توجه به اینکه انتقال ما از استیتی ایی به استیت دیگر بصورت ترتیبی رو به جلو می باشد، لذا برخی از درایه های آن صفر می شود. با نرمال کردن هر سطر به ماتریس زیر خواهیم رسید:

Transition_Matrix				
1	2	3	4	
0.8571	0.1429	0	0	
0	0.9035	0.0965	0	
0	0	0.8000	0.2000	
0.0409	0	0	0.9591	

بطور مثال احتمال اینکه در استیت یک باشیم و در همان استیت بموئیم 0.85 است. یا احتمال اینکه در استیت یک باشیم و به استیت دو ببریم 0.142 است. یا از آنجایی که امکان ندارد از مرحله یک به یک باره به مرحله سه یا چهار ببریم این احتمال برابر صفر است.

ب) حال باز هم برای هر کلاس با استفاده از دستور `mnrfit` ضرایب مربوط به مدلمان را بدست می آوریم که این ضرایب بصورت زیر می شوند:

The screenshot shows the MATLAB Variables browser titled "Variables - B_3". It contains four variables: B_1, B_2, B_3, and B_4, each represented as a 5x1 double matrix. The data for each matrix is as follows:

	B_1	B_2	B_3	B_4
1	1 -2.2625	1 -1.3875	1 -2.4003	1 -0.1616
2	2.3093	-0.6329	0.7708	-2.4277
3	-1.1744	0.4231	0.0214	1.1467
4	7.1338e-04	-0.6458	-0.0152	-0.3444
5	0.0490	0.0243	0.0434	-0.0432

بطوری که این ضرایب برای دسته بندی کلاس ها بصورت `one vs all` می باشند.

حال با استفاده از دستور `mnrvall` و ضرایب بدست آمده برای هر کدام از مدل ها و همچنین ویژگی های داده های تست ما، احتمال رخداد برای هر استیت را محاسبه می کنیم.

The screenshot shows the MATLAB Workspace browser titled "Workspace". It contains one variable: pihat_4, represented as a 1700x2 double matrix. The data for the first 11 rows is as follows:

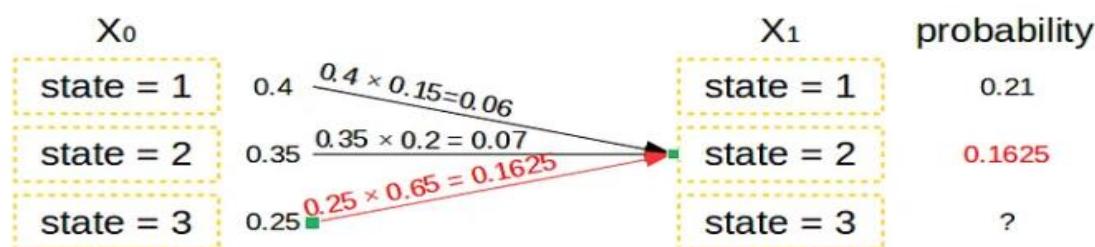
	1	2
1	0.6167	0.3833
2	0.4366	0.5634
3	0.7711	0.2289
4	0.7334	0.2666
5	0.7299	0.2701
6	0.4933	0.5067
7	0.0713	0.9287
8	0.0056	0.9944
9	6.7504e-04	0.9993
10	0.0047	0.9953
11	0.0544	0.9456

بطور مثل ماتریس بالا بیان می کند که احتمال رخداد اینکه مشاهده یک ما در کلاس چهار باشد برابر 0.6167 می باشد. لذا ستون اول این چهار ماتریس تولید شده را می توانیم بعنوان احتمال رخداد مشاهدات ما در نظر بگیریم و ماتریس B را بسازیم.

حال به پیاده سازی الگوریتم viterbi می پردازیم. برای این کار بصورت بازگشتی عمل می کنیم. ابتدا با توجه به احتمال مساوی استیت های اولیه، T_0 را درست می کنیم که برابر مقدار ماتریس B در سطر اول می شود.

حال برای زمان های بعدی شکل زیر را در نظر میگیریم:

Step 1



به این صورت که برای هر استیت، مقادیر احتمالی از چهار استیت قبل را محاسبه می کنیم و سپس از بین آنها ماکس را بعنوان احتمال وقوع آن استیت انتخاب می کنیم:

$T_{-1}(i-1,1) * \text{Transition_Matrix}(1,j) * pihat(i,j)$

به نتیجه ای بی مشابه زیر می رسیم:

	1	2	3	4
1	0.0628	0.2405	0.0540	0.6167
2	0.0062	0.0461	0.0032	0.2582
3	3.3563e-04	0.0118	2.0466e-04	0.1909
4	2.9378e-04	0.0029	5.7278e-05	0.1343
5	2.2090e-04	7.2068e-04	1.3195e-05	0.0940
6	3.4477e-04	1.3116e-04	4.6359e-06	0.0445
7	7.2564e-04	7.9905e-06	2.1191e-06	0.0030
8	5.0874e-04	3.3132e-06	9.9710e-07	1.6291e-05
9	4.2546e-04	1.7026e-06	5.8082e-07	1.0547e-08
10	3.3180e-04	4.0197e-06	2.5517e-07	5.4404e-10
11	1.6634e-04	6.0449e-06	7.7557e-08	2.7751e-09

حال کافیست برای هر زمان، ماکس عددی که اتفاق می‌وقتند را بعنوان استیت پیشنهادی انتخاب کنیم. در شکل زیر دو حالت تخمین زده شده و واقعی را برای چند نمونه مشاهده می‌کنید:

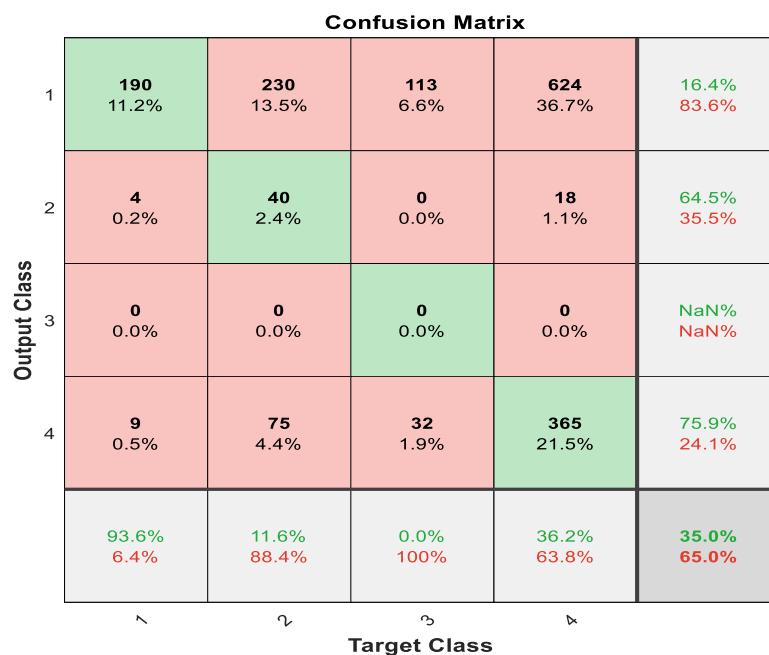
The figure shows the MATLAB workspace with two variables:

- est_state**: A 1700x1 double array with values ranging from 1 to 4.
- PCG_states_test**: A 1700x1 double array with values ranging from 1 to 4.

	1	2
1	4	
2	4	
3	4	
4	4	
5	4	
6	4	
7	4	
8	1	
9	1	
10	1	
11	1	
12	1	
13	1	
14	1	
15	2	
16	2	
17	2	
18	2	
19	4	
20	4	
21	4	

	1	2
1	4	
2	4	
3	4	
4	4	
5	4	
6	4	
7	1	
8	1	
9	1	
10	1	
11	1	
12	1	
13	1	
14	2	
15	2	
16	2	
17	2	
18	2	
19	2	
20	2	
21	2	

و اگر ماتریس آشتفتگی را برای آن رسم کنیم به نتیجه ای مشابه شکل زیر می‌رسیم:



نتیجه: با توجه به اینکه در مارکوف توالی استیت ها و ترتیب شون در نظر گرفته می شود لذا انتظار می رود که در این حالت به نتیجه بهتری برسیم. اما برای تشکیل ماتریس transition نیاز به داده های زیادی داریم که شاید در همه تسك ها در اختیار نداشته باشیم. همچنین از آنجایی که ویژگی هایی که داشتیم جدایی پذیر خطی نبودند، لذا در هر دو حالت به نتیجه مطلوبی نرسیدیم.

باتشکر ☺