بسم الله الرحمن الرحیم

دانشکده مهندسی برق

درس سیگنال و سیستم ها

استاد کربلایی

پروژه پایانی

صبا نصیری – زهرا سرخئی

98101725-98101052

نمونه برداری_Sampling:

رسم اندازه ی تبدیل فوریه ی گسسته و تابع HalfBandFFTt :

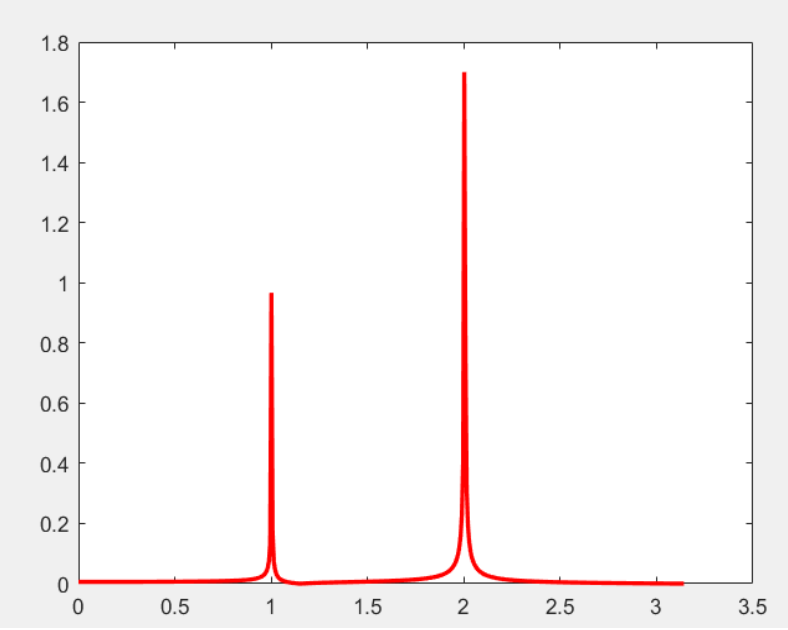
تابع HalfbandFFT1 برای این بخش نوشته شده است که سیگنال ورودی و فرکانس نمونه برداری را دریافت می کند.در این تابع ابتدا از سیگنال ورودی fft گرفته می شود که برای صدق کردن سیگنال در رابطه پارسوال حاصل را بر نصف طول سیگنال تقسیم می کنیم.برای scale کردن محور افقی ،نصف سمپل ها را در نظر می گیریم و بر تعداد کل سمپل ها تقسیم می کنیم با ضرب این عبارت در 2pi*fs محور افقی بر حسب w ،scale می شود که با ضرب آن در Ts ،می توان محور را از 0 تا pi درجه بندی کرد.در نهایت مقادیر به دست آمده رسم می شوند.برای سهولت کار در ادامه ی پروژه تابع دیگری با اسم HalfBandFFT2 نوشته شده است که محور افقی را برحسب فرکانس(هرتز) نمایش می دهد.

سیگنال زیر برای تست توابع این بخش استفاده شده است:

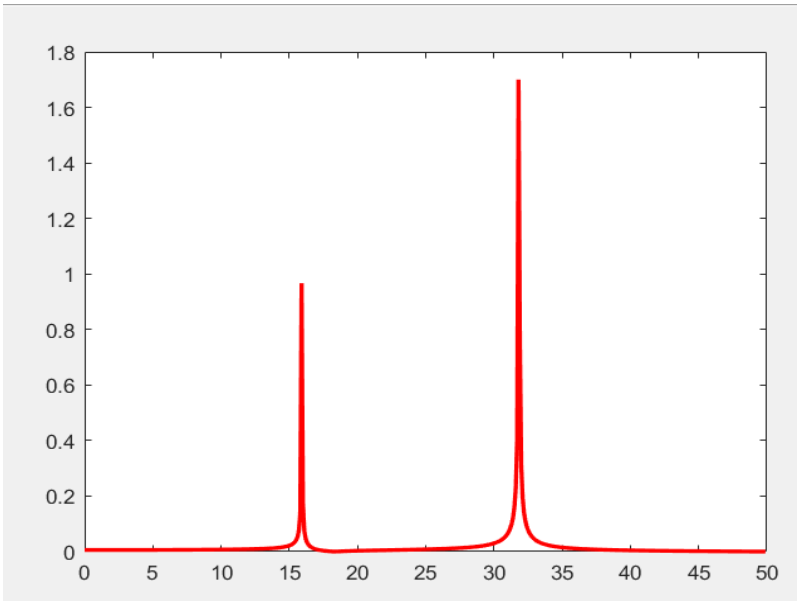
```
x=cos(100*t)+2*cos(200*t);
```

فرکانس نمونه برداری 100 فرض شده است:

خروجی تابع HalfBandFFt1 :

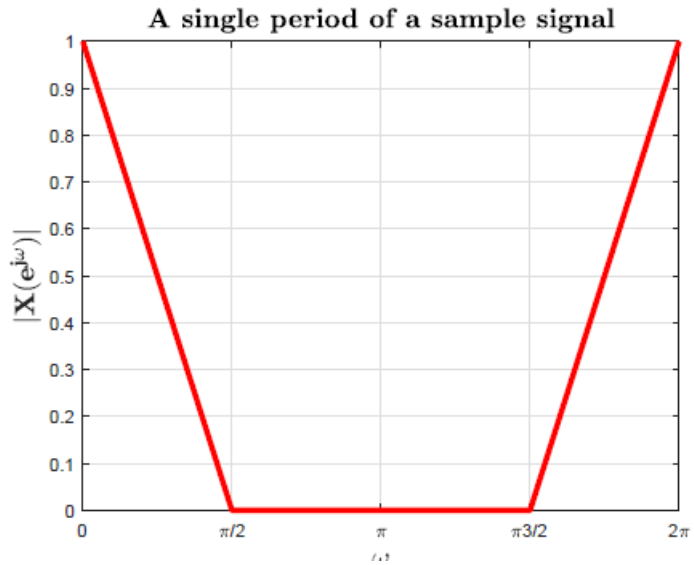


خروجی تابع HalfBandFFt2 :



تعیین حد پایین فرکانس نمونه برداری:

در شکل زیر:



بالاترین محتوای فرکانسی pi/2 است با ضرب این مقدار در فرکانس نمونه برداری ،بیشترین محتوای فرکانسی سیگنال به دست می آید و 2 برابر این مقدار فرکانس نایکوییست است:

Fn=pi/2*Fs*2=pi*Fs

فرکانس نمونه برداری و aliasing :

اگر Fs1=1/6 آن گاه پهنای باند سیگنال به صورت زیر خواهد بود:

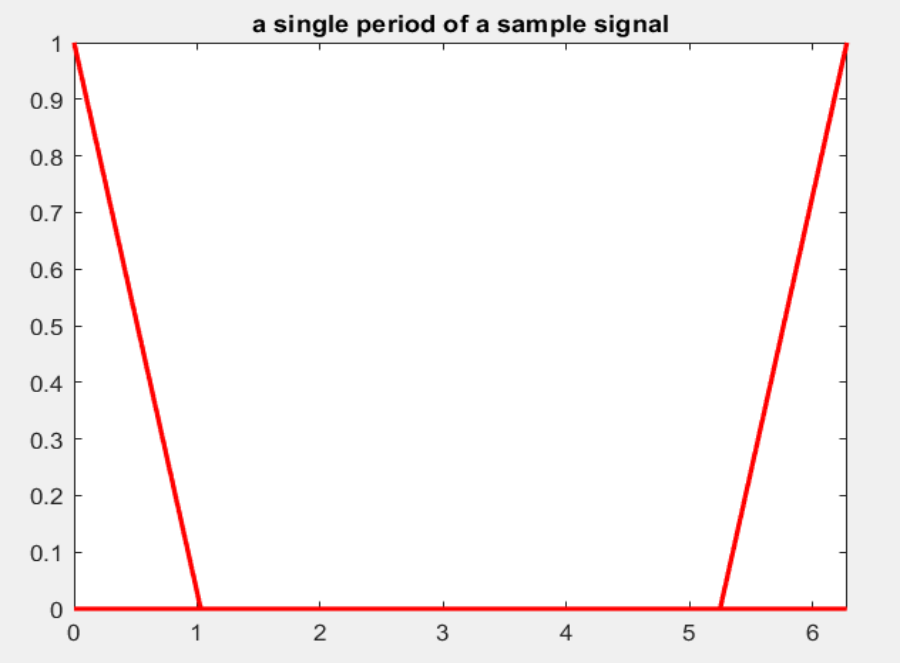
Band width=pi/2*1/6=pi/12

با افزایش فرکانس نمونه برداری از 1/6 به 1/4 احتمال aliasing کاهش می یابد و سیگنال می تواند بازیابی شود.

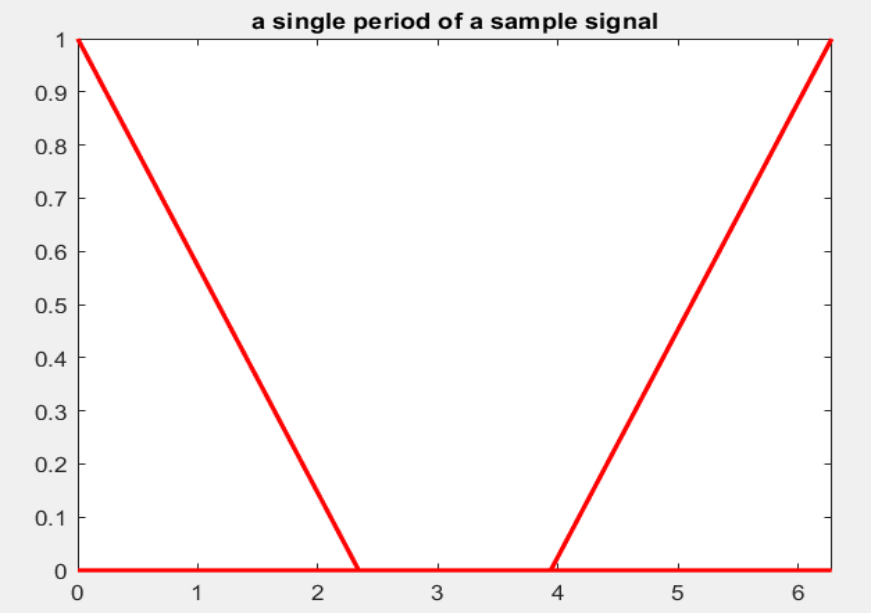
برای رسم این بخش در متلب ،از این نکته استفاده شده است که برای تبدیل محتوای فرکانسی سیگنال گسسته به پیوسته محور افقی در Fs ضرب و برای تبدیل محتوای فرکانسی سیگنال پیوسته به گسسته محور افقی در Ts ضرب می شود.ابتدا محتوای فرکانسی ماکسیمم با توجه به نکته ذکر شده به دست می آید و معادلات خطوط متناظر، نوشته شده و رسم می شوند.

خروجی را در زیر مشاهده می کنید:

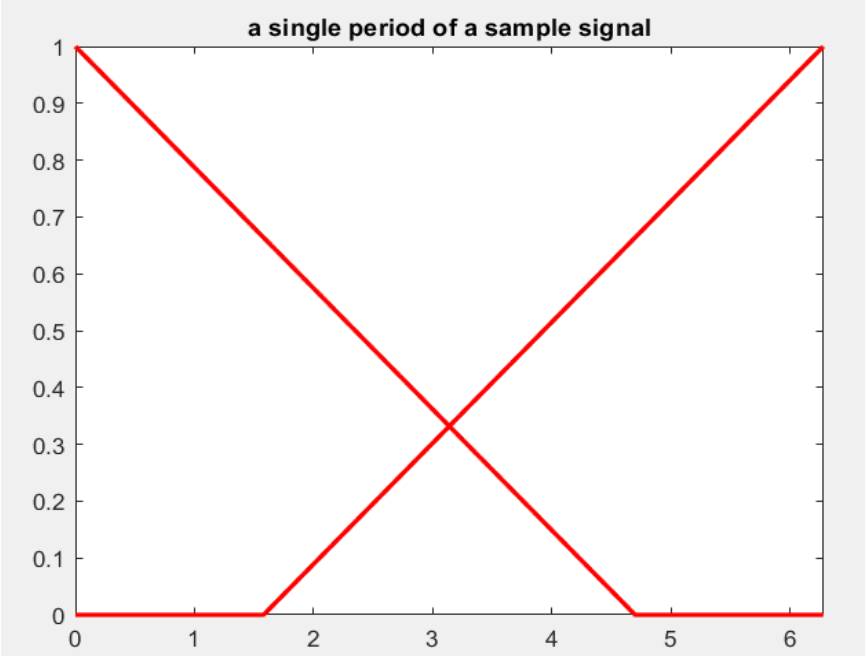
T1=6,T2=4 :



: T1=1/6,T2=1/4



:T1=1/6,T2=1/2



همان طور که مشاهده می کنید،در اخرین مورد دو تناوب سیگنال با هم تداخل کرده و سیگنال اصلی قابل بازیابی نیست.به این پدیده aliasing گفته می شود.برای رفع این مشکل باید شرط نایکوییست رعایت شود.

آشنایی با سیگنال های EEG :

P300,ERP :

به پتانسیل های الکتریکی که در اثر تحریک در مغز به وجود می آیند،ERP می گویند.این تحریک می تواند توسط یک سنسور یا فعالیت روانی ایجاد شود.این پتانسیل ها سیگنال های EEG ای تولید می کنند که با پردازش،اطلاعاتی از آن ها قابل استخراج است.روش های پردازشی می توانند روش های ساده مثل میانگین گیری یا روش های پیچیده ی machine learning باشند.ERP ها بسته به منبع تحریک انواع مختلفی دارند.برای مثال می توانند به صورت بینایی،شنوایی یا از طریق لامسه باشند.p300 یک نوع از انواع ERP است که به صورت بینایی تحریک شده است.این سیگنال ،مهم ترین سیگنال ERP است که در سیگنال EEG به صورت یک پیک مثبت بزرگ در 300 تا 500 میلی ثانیه پس از اعمال تحریک مشاهده می شود که برای اکثر کاربران در 300 میلی ثانیه است که به همین دلیل به آن p300 گفته می شود. ولی محل دقیق آن هم چنان به طور کامل مشخص نیست.می توان آن را در انتهای یک فرایند ادراکی یا هنگام به روز رسانی حافظه پس از ارزیابی اطلاعات جدید در نظر گرفت.یکی از مهم ترین بخش های p300 ،p3a است که در بخش جلوی سر و پیشانی به وجود می آید.بخش دیگر آن که بالاترین پتانسیل در p300 را داراست،p3b است که برای پژوهش های ادراکی در روان شناسی استفاده می شود.معمولا برای استخراج سیگنال های EEG از روش میانگین گیری استفاده می شود که در آن تریال ها،با توجه به این حقیقت که سیگنال های مغزی ترکیبی از فعالیت های گوناگون و نویز و ... هستند، میانگین گرفته می شوند.این سیگنال هنگام تصمیم گیری هم مشاهده می شود.برای آزمایش p300 از 3 پارادایم اصلی استفاده می شود:

1:single stimulus

2:oddball

3:three_stimulus

باندهای فرکانسی مختلف:

5 نوع باند اصلی وجود دارد:

1) دلتا:

این باند محتوی فرکانس های کمتر از 4 هرتز است.در بزرگسالان در جلوی سر و در کودکان در پشت سر ایجاد می شود.هنگامی که کاری به توجه پیوسته نیاز دارد،دلتا می تواند دیده شود.

2) تتا:

این باند محتوی فرکانس های بین 4 تا 7 هرتز است.هنگامی که در حال تسکی هستیم در نواحی از مغز که درگیر تسک نیستند دیده می شود.در جوانان و کودکان بیشتر وجود دارد.بیشتر مواقعی به وجود می آید که فرد سعی دارد یک پاسخ یا فعالیت را سرکوب کند.

3) آلفا:

این باند محتوی فرکانس های بین 8 تا 15 هرتز است.در پشت و دو طرف سر دیده می شود ولی هنگام استراحت در نواحی میانی سر هم می تواند وجود داشته باشد.الفا بیشتر مواقع استراحت و یا مواقعی که چشم ها بسته اند ایجاد می شود.

4) بتا:

این باند محتوی فرکانس های بین 16 تا 31 هرتز است.در 2 طرف سر دیده می شود و به طور متقارن پخش می شود ولی در بخش های جلوی سر مشهودتر است.بتا در مواقع استرس و یا مواقعی که مغز به طور فعال و آگاهانه کار می کند و یا تمرکز وتفکر زیادی نیاز است،دیده می شود.

5) گاما:

این باند محتوی فرکانس های بین 32 تا 100 هرتز است.گاما زمانی که از 2 حس همزمان استفاده می شود ،مثلا شنوایی و بینایی،دیده می شود.هم چنین زمانی که با حافظه ی کوتاه مدت دو ابجکت را match می کنیم ،دو صدا را مطابقت می دهیم و یا دو جسم را لمس می کنیم،وجود دارد.

فرکانس نایکوییست مناسب برای باند های فرکانسی مختلف:

طبق مطالب بالا ، بیشترین فرکانسی که می توان در سیگنال های مغزی مشاهده کرد،100 هرتز است پس فرکانس قطع مناسب برای این سیگنال ها در حدود 100 هرتز و فرکانس نایکوییست 200 هرتز است.

محاسبه ی فرکانس نمونه برداری:

با محاسبه ی تفاضل دو زمان نمونه برداری در سطر اول داده ها و معکوس کردن آن فرکانس نمونه برداری به دست می آید که 256 هرتز است.

فرکانس قطع مناسب:

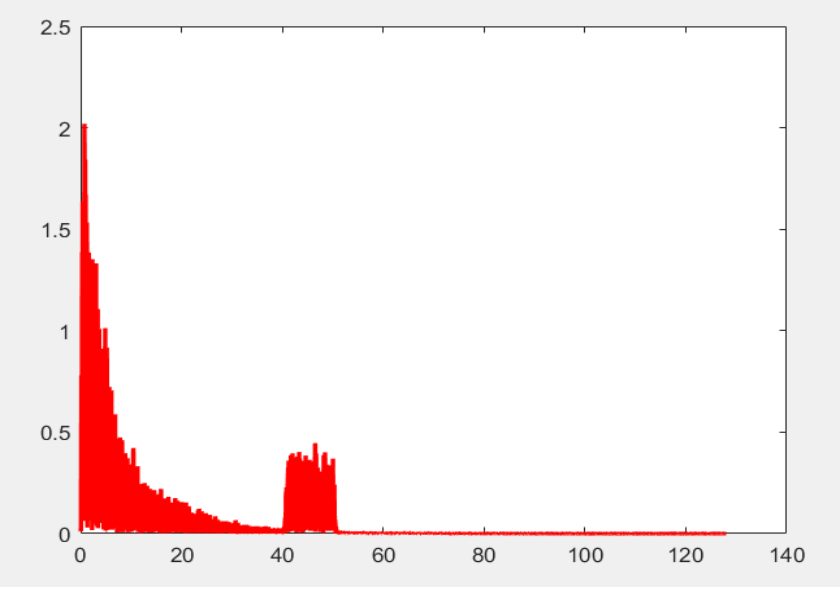
همان طور که بالاتر ذکر شد،فرکانس در حدود 100 هرتز می تواند مناسب باشد.

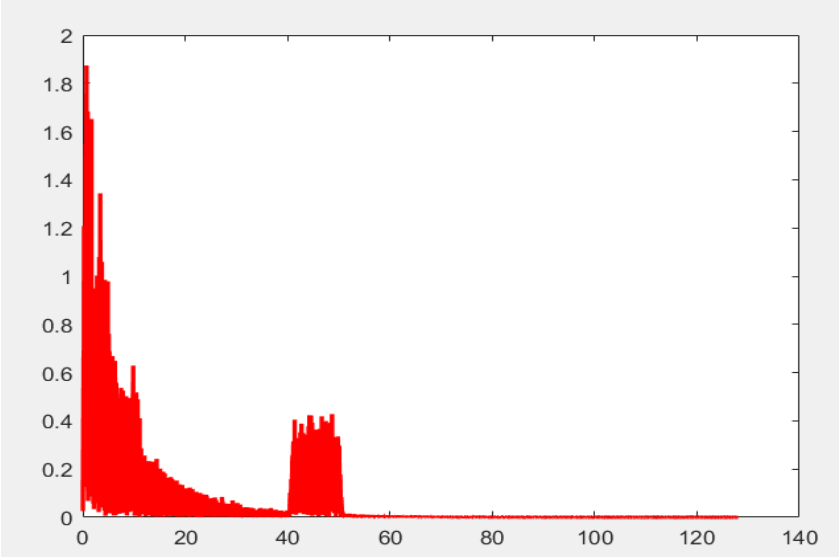
رسم طیف فرکانسی الکترودها:

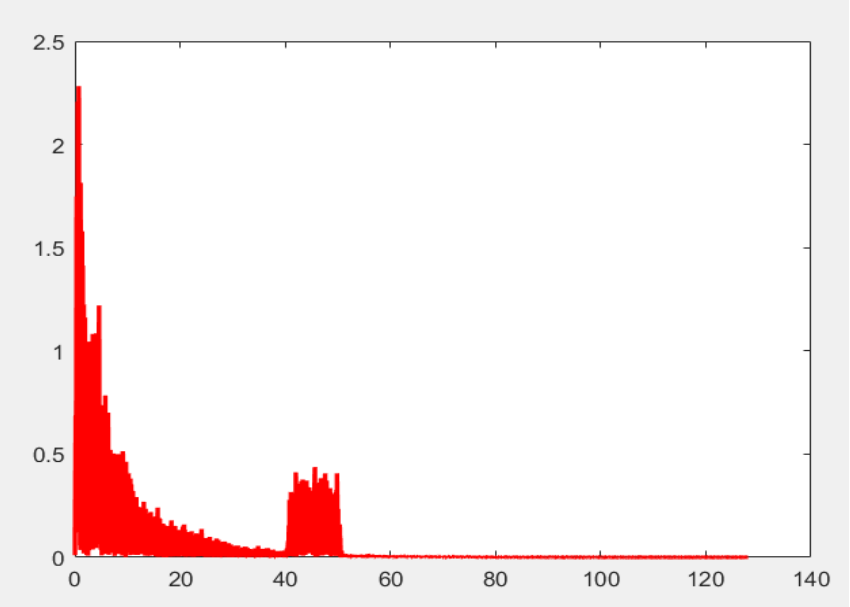
برای رسم طیف ها از تابع HalfbandFFT2 که در بخش اول شرح داده شد،استفاده شده است.

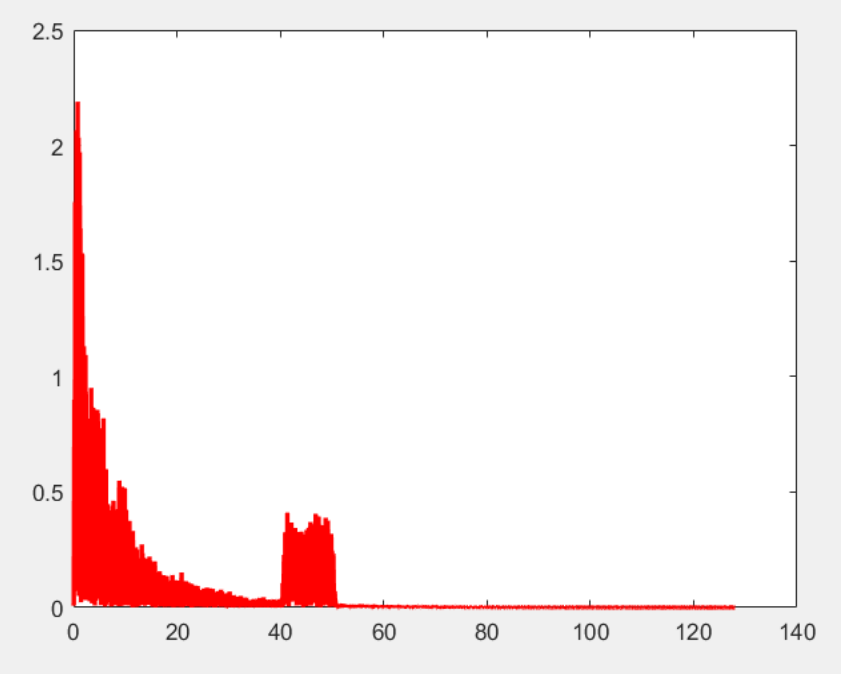
این طیف ها برای هر 8 دیتاست و برای داده های train رسم شده اند و 8 نمودار داده شده برای هر dataset مربوط به 8 الکترود است:

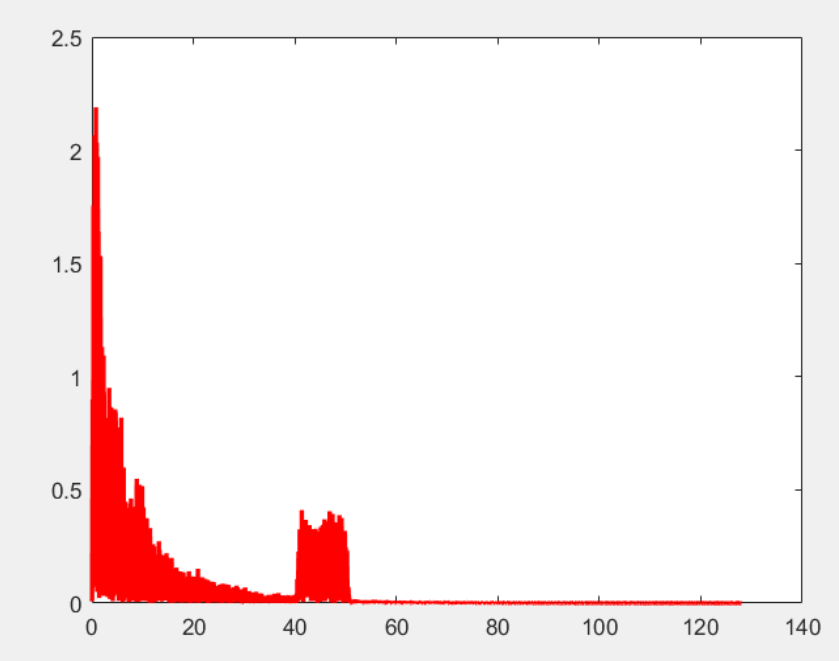
Dataset1:

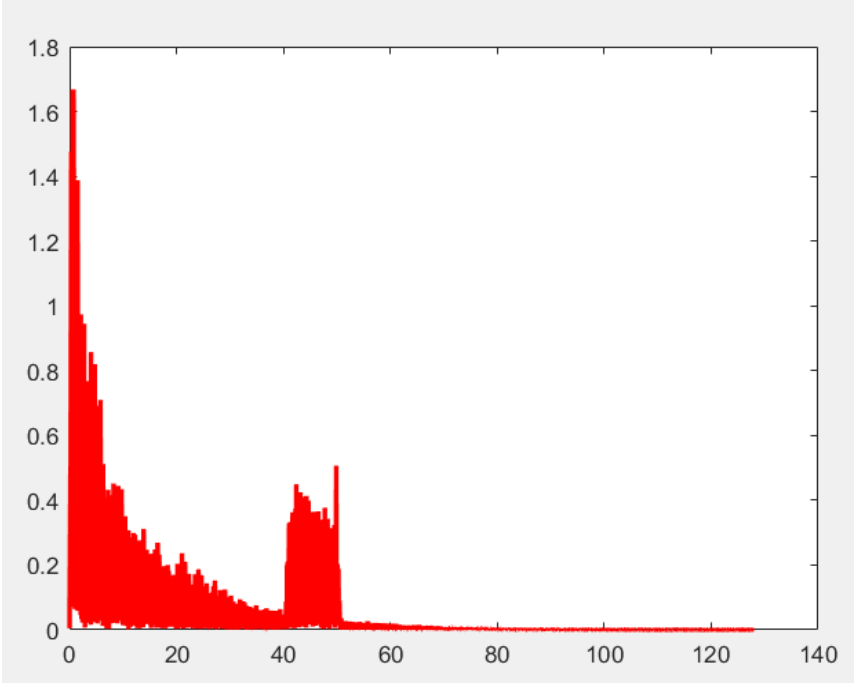


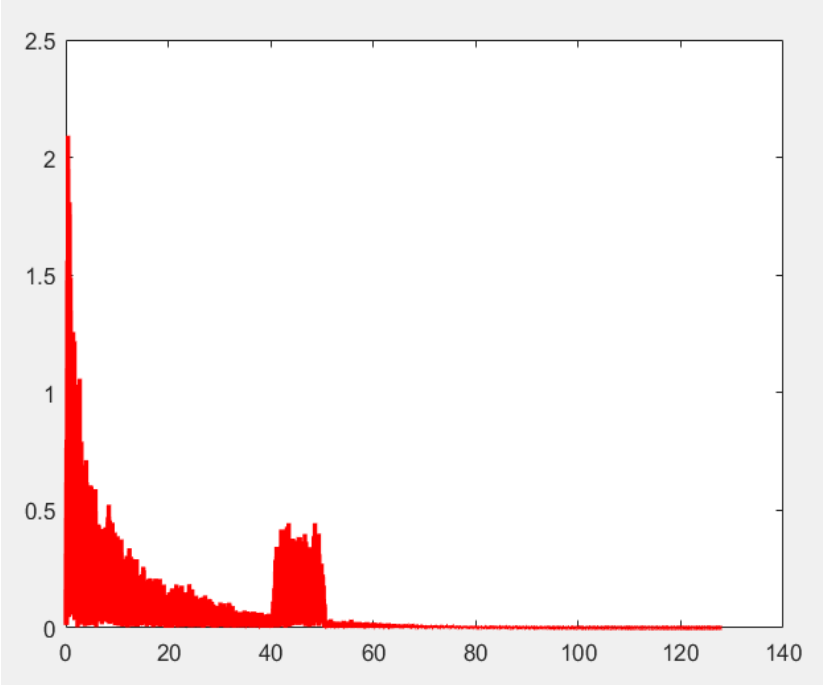

2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
120
140

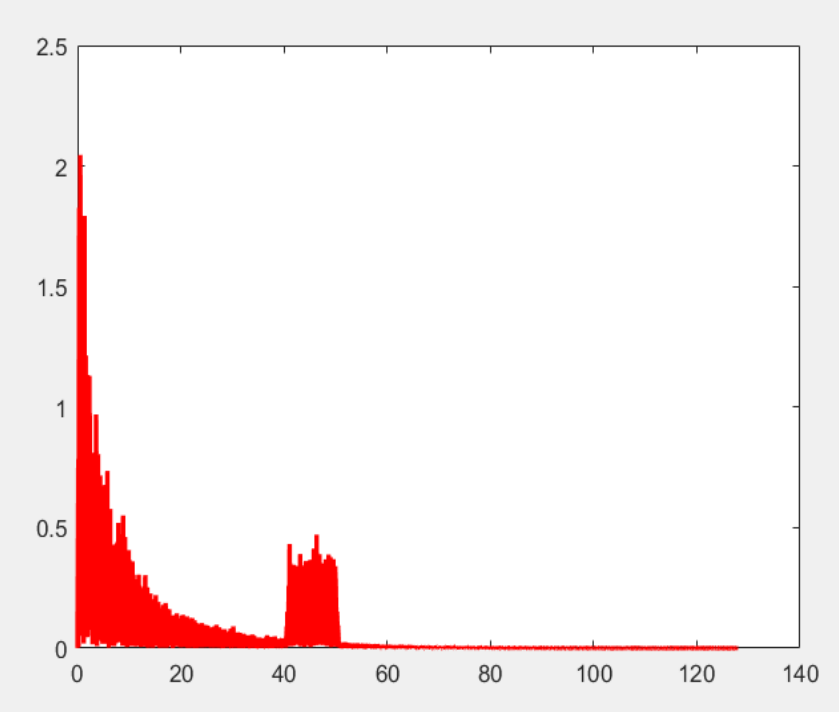

2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140


2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140


2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140

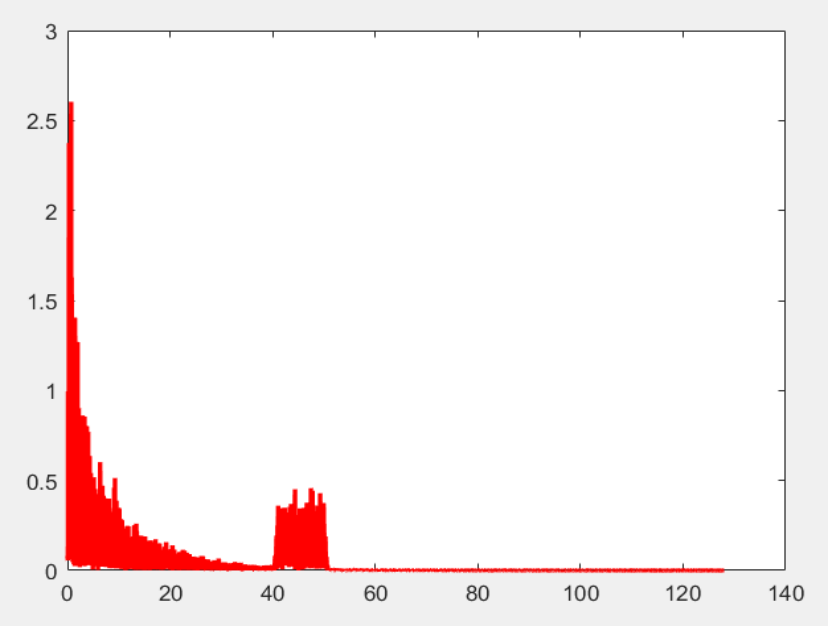

1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
120
140

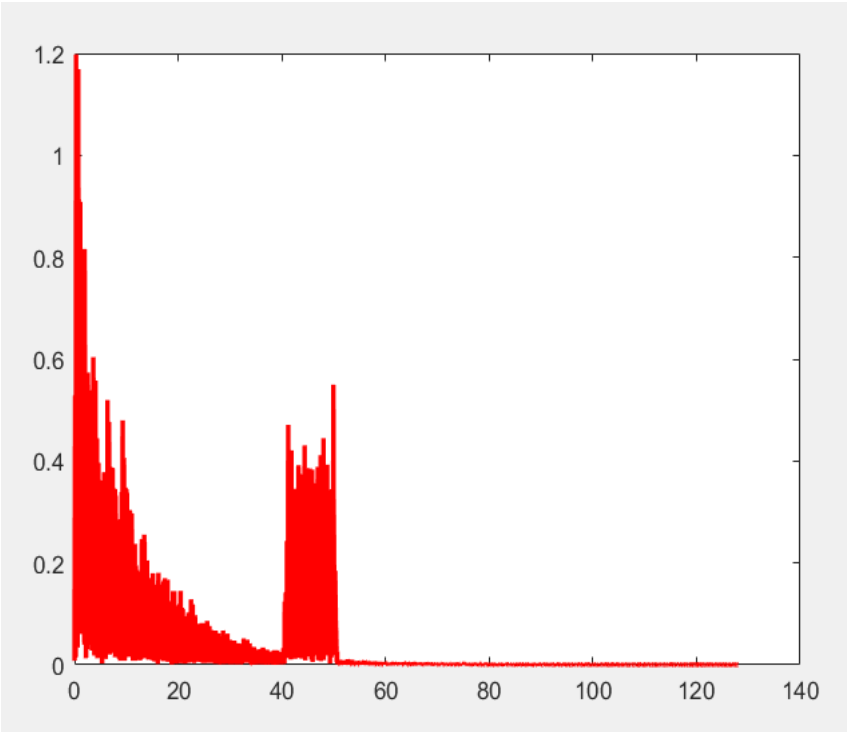

2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140

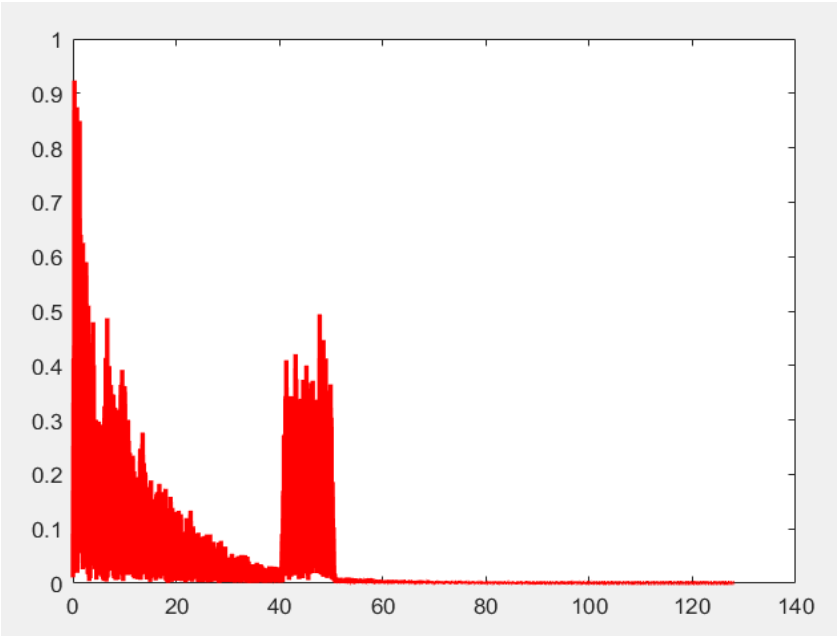


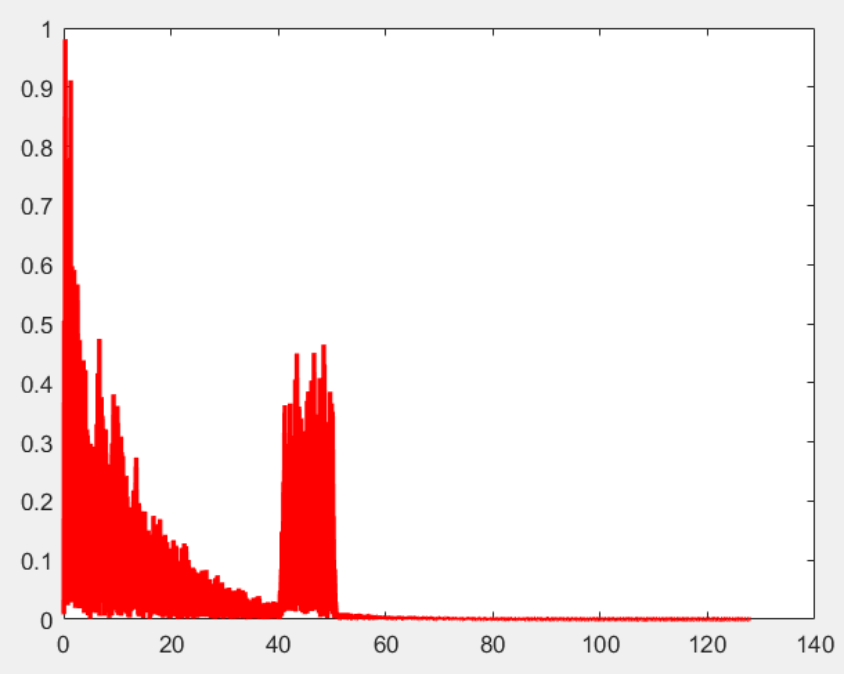
خروجی ها برای دیتاست 2 بسیار مشابه دیتاست 1 است .

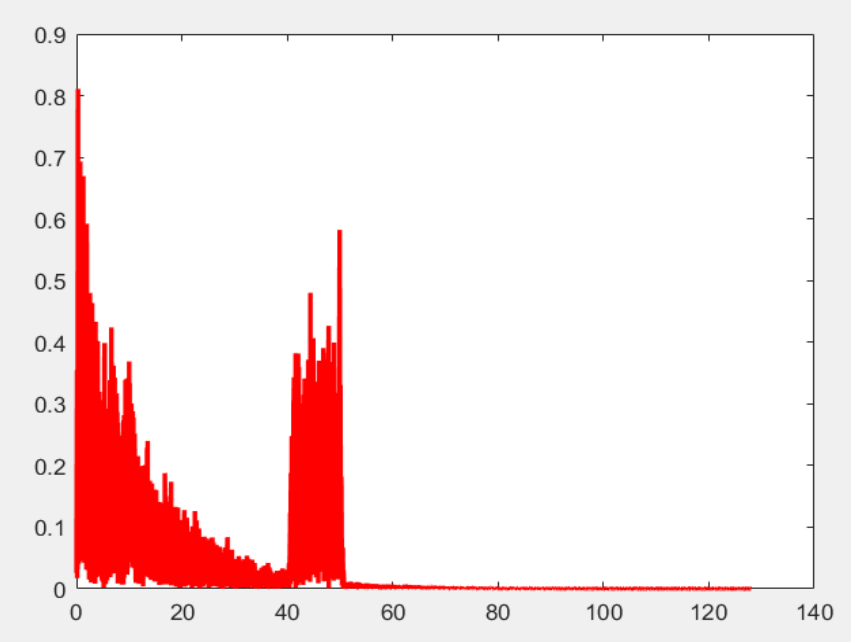
Dataset3:

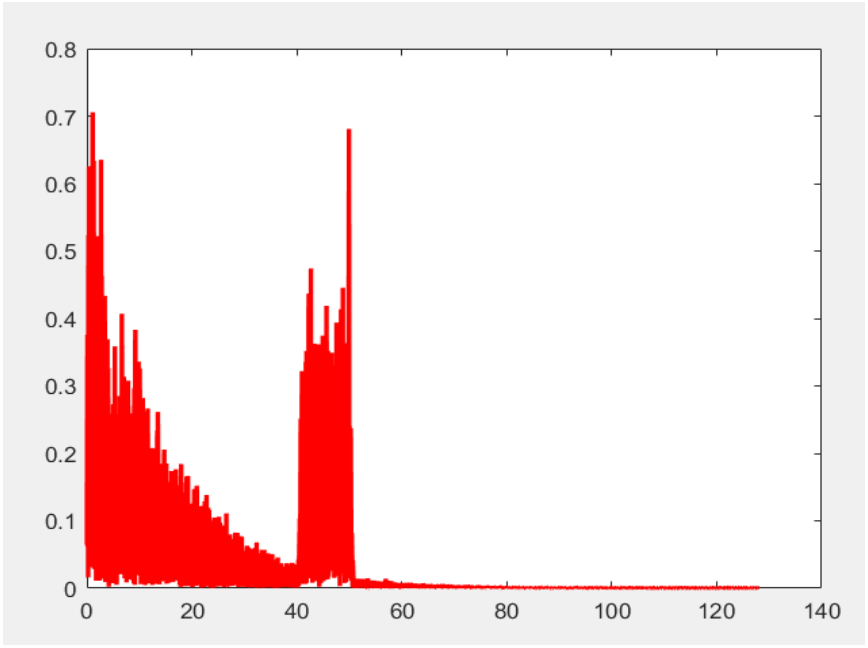


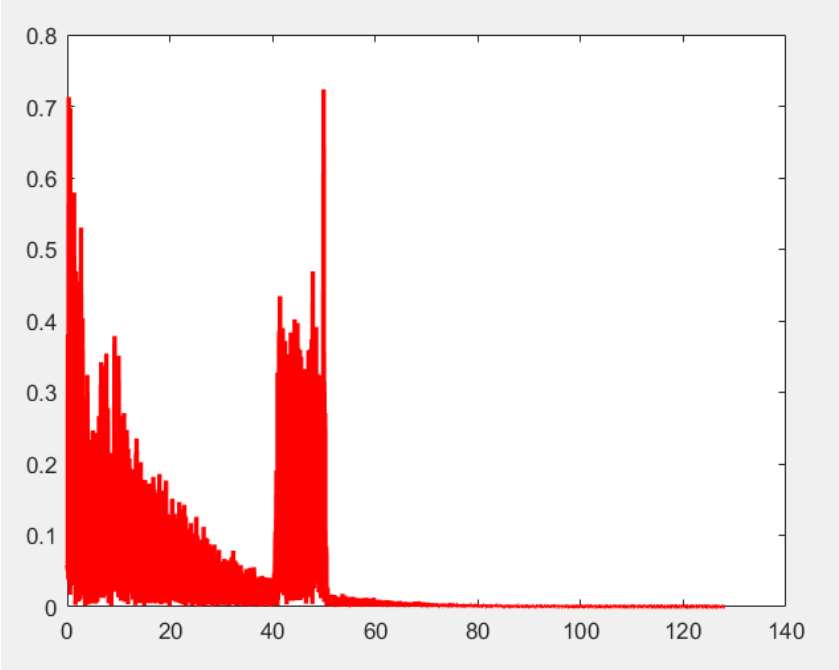

1.2
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
120
140

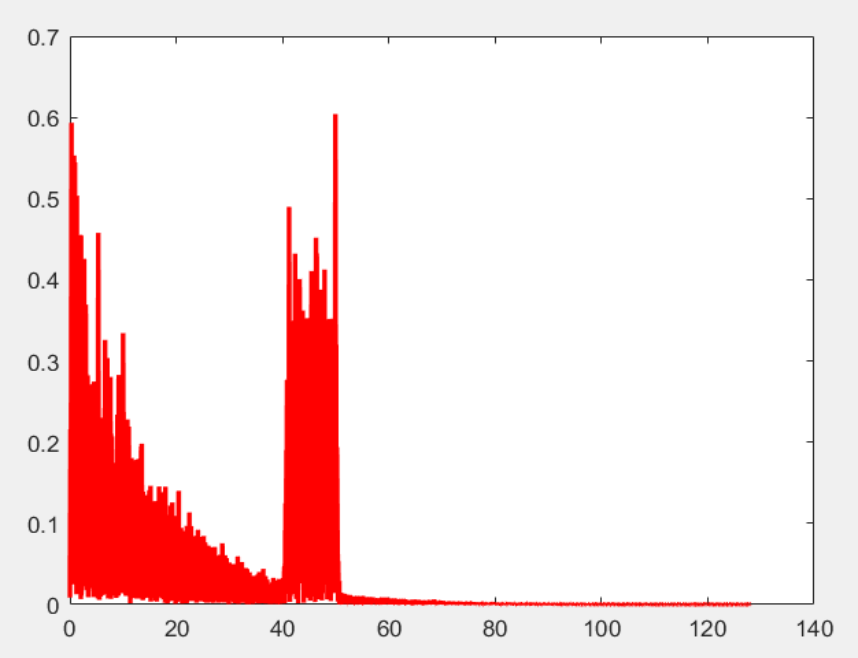

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140

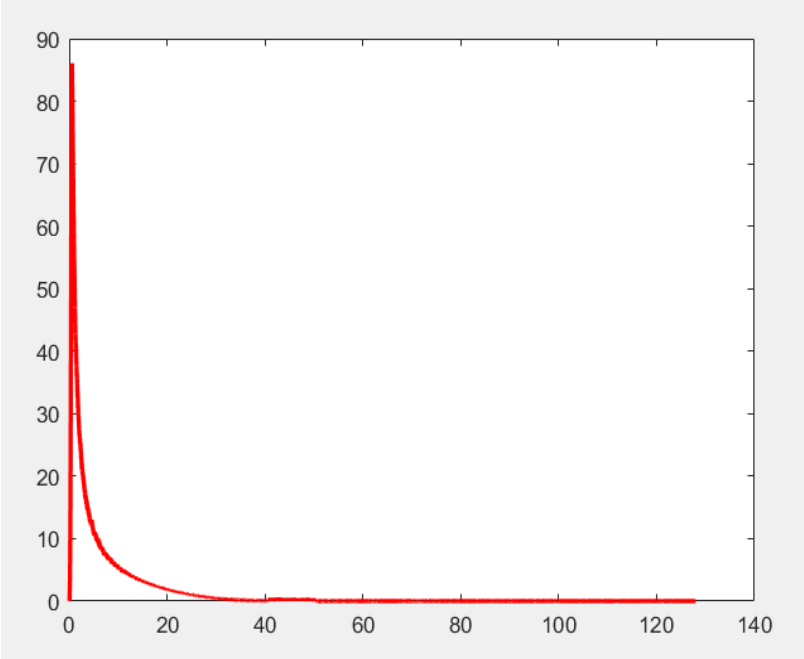

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140

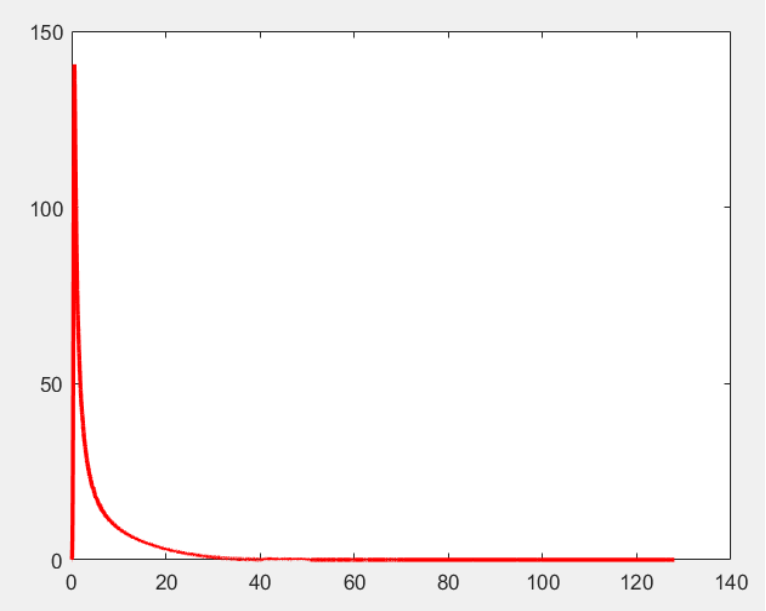

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140

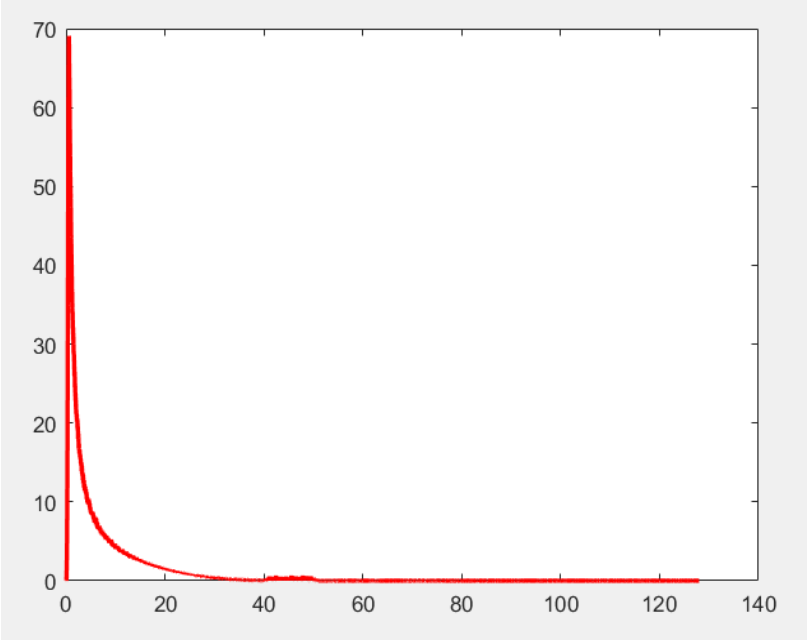


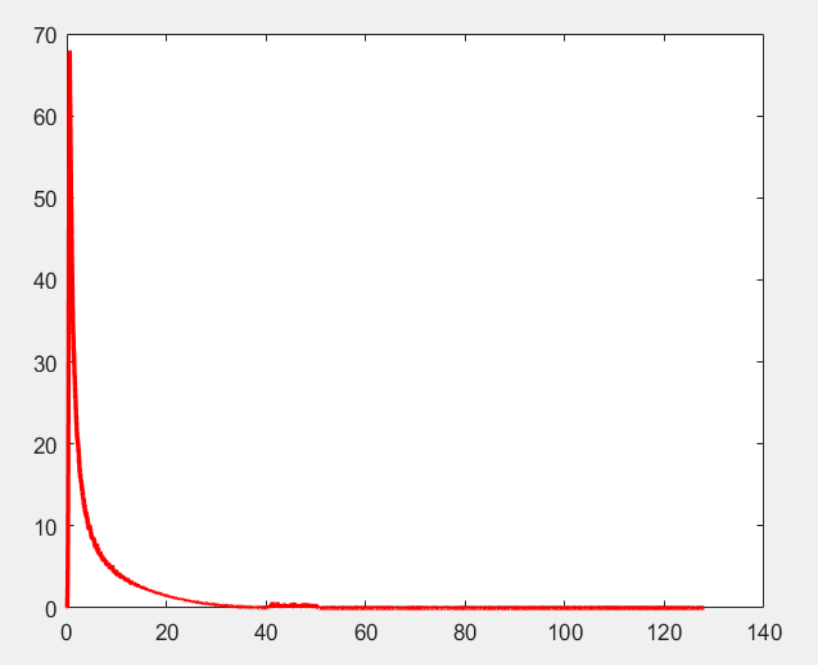
طیف برای dataset5,6,7,8,9 بسیار مشابه است و در زیر به عنوان نمونه برای dataset6 رسم شده است:

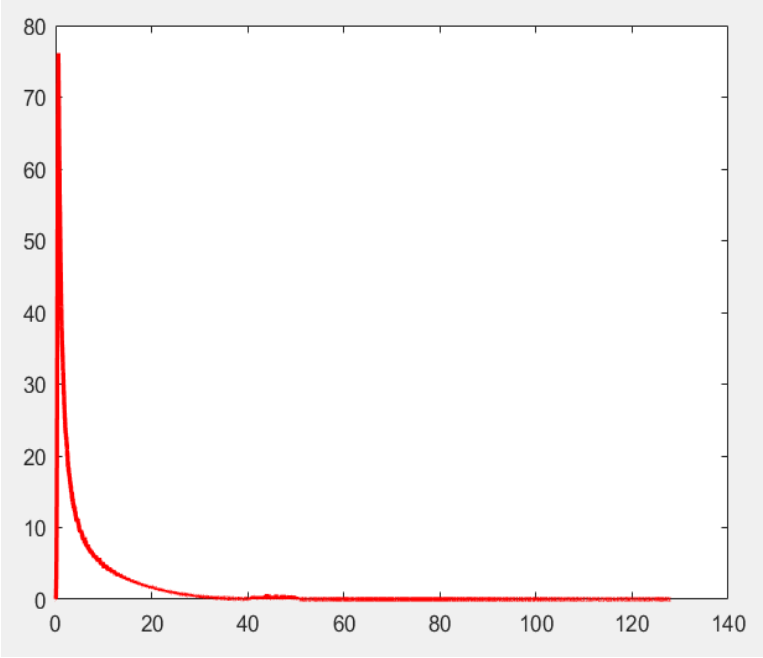
Dataset6:

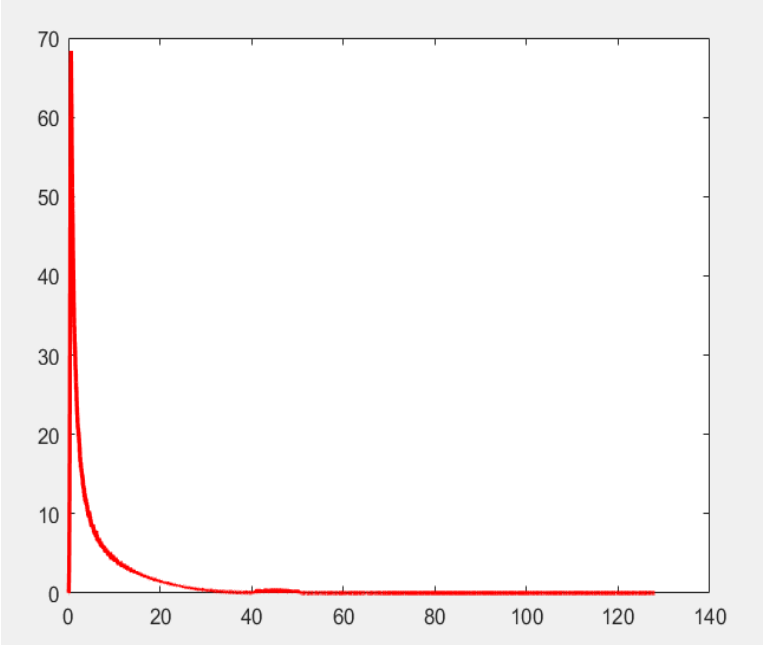


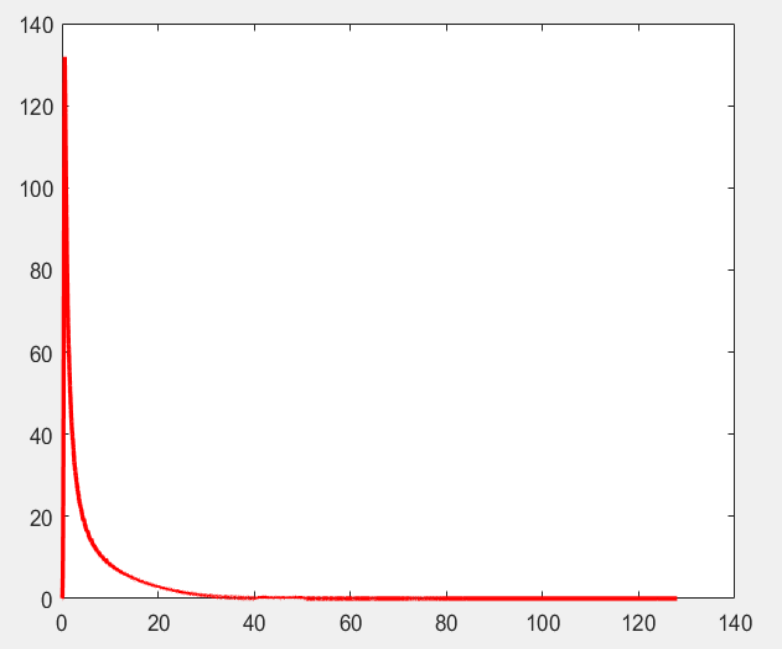

150
100
50
0
0
20
40
60
80
100
120
140

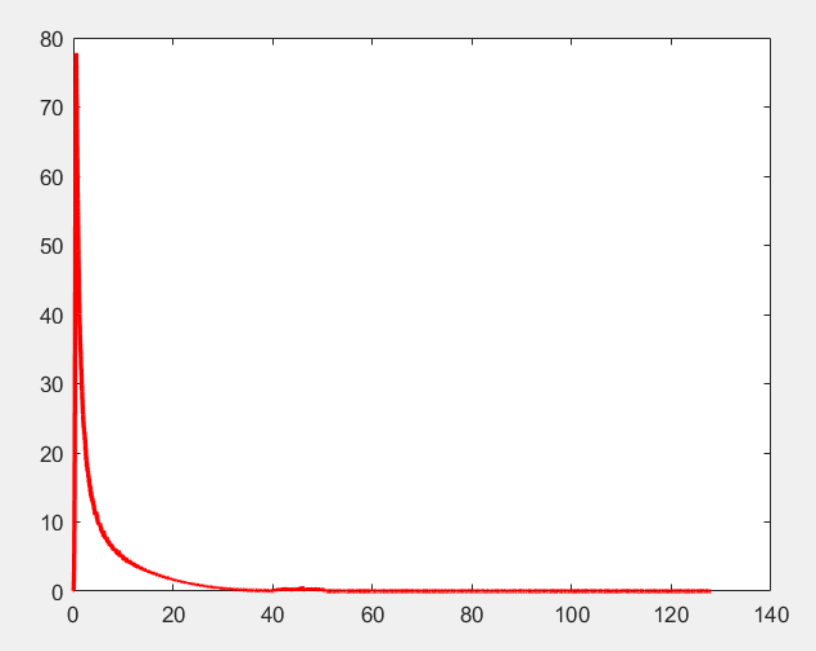

70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140


70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140


80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140


70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140


140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140

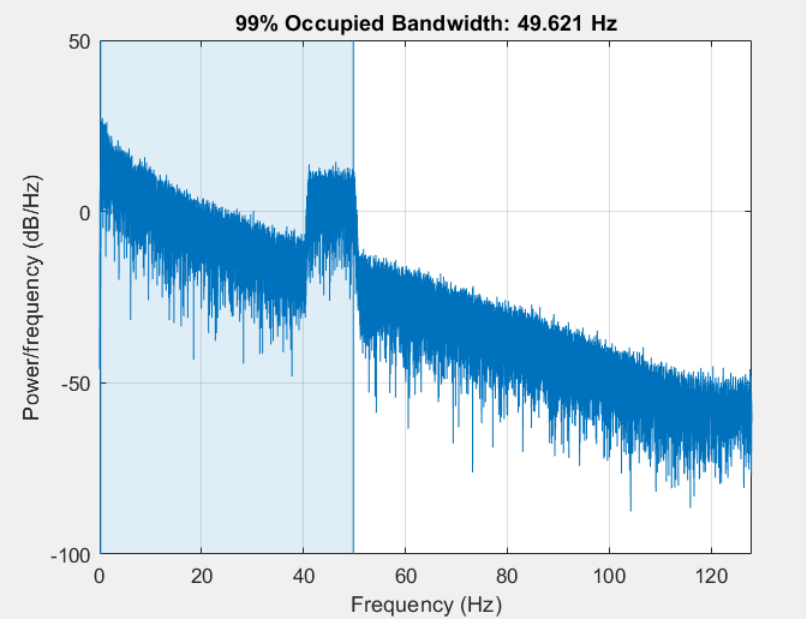


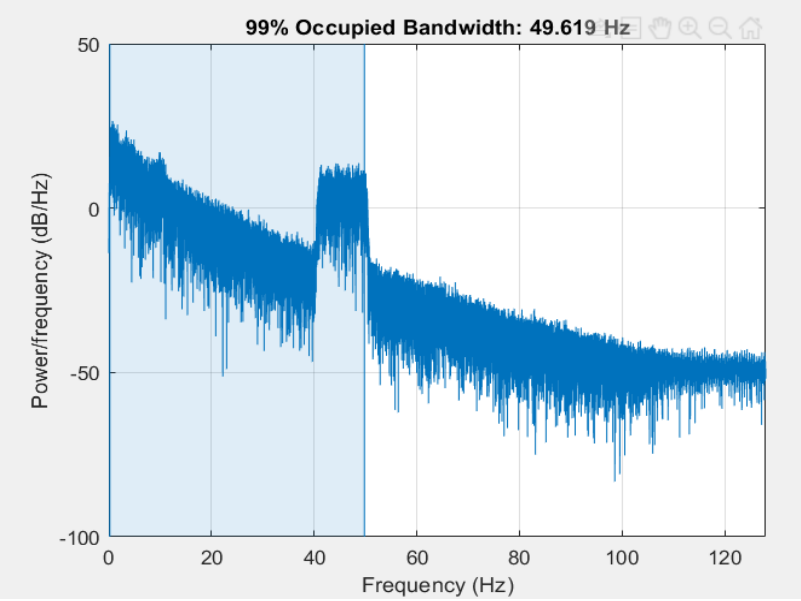
با توجه به نمودارهای بالا اگر فرکانس قطع مقداری بیشتر از حدودا 50 هرتز باشد،مناسب خواهد بود.

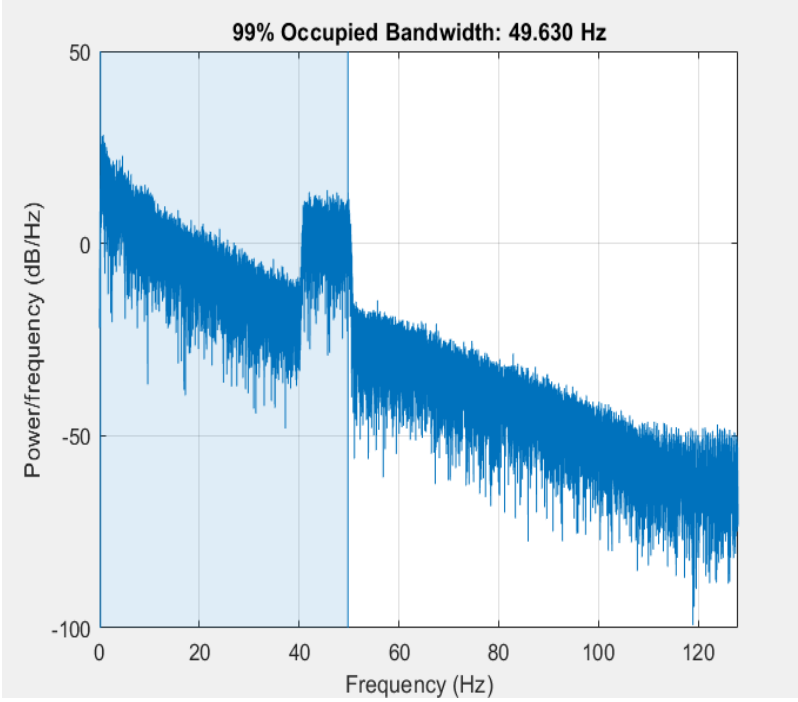
تشخیص فرکانس قطع با توجه انرژی سیگنال:

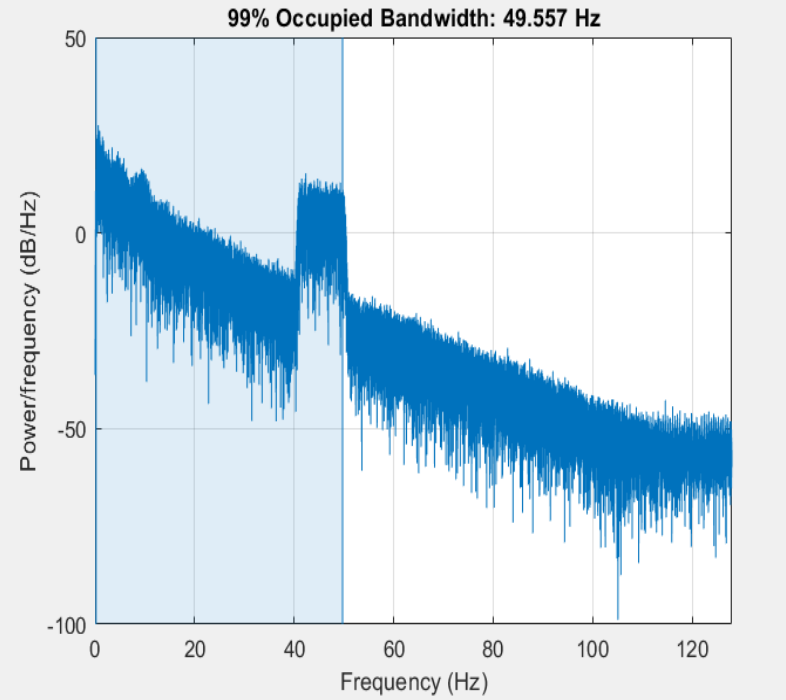
برای این بخش از تابع obw استفاده شده است که سیگنال و فرکانس نمونه برداری را دریافت می کند و فرکانسی که 99 درصد انرژی سیگنال در کمتر از آن موجود است را برمی گرداند.نمودار 8 الکترود train مربوط به dataset1 در زیر موجود است.

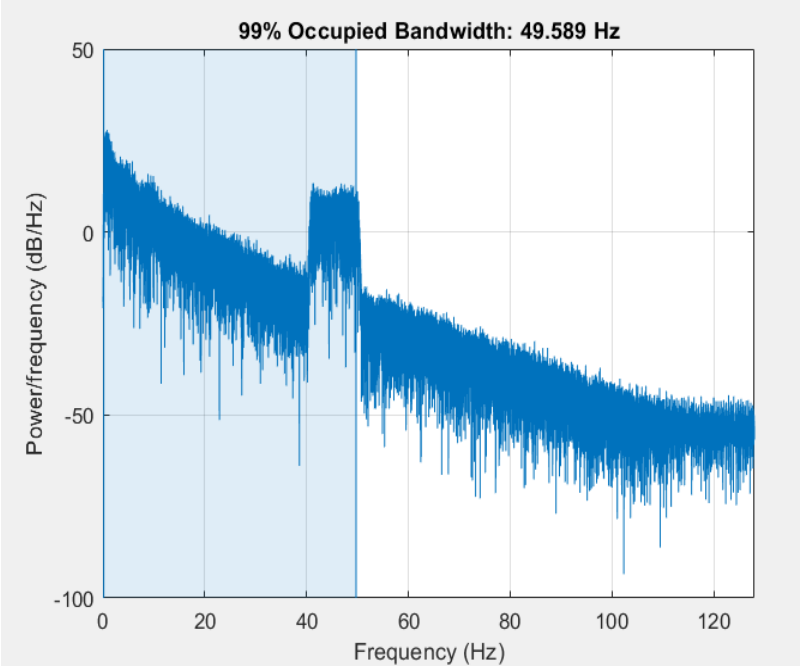
Dataset1:

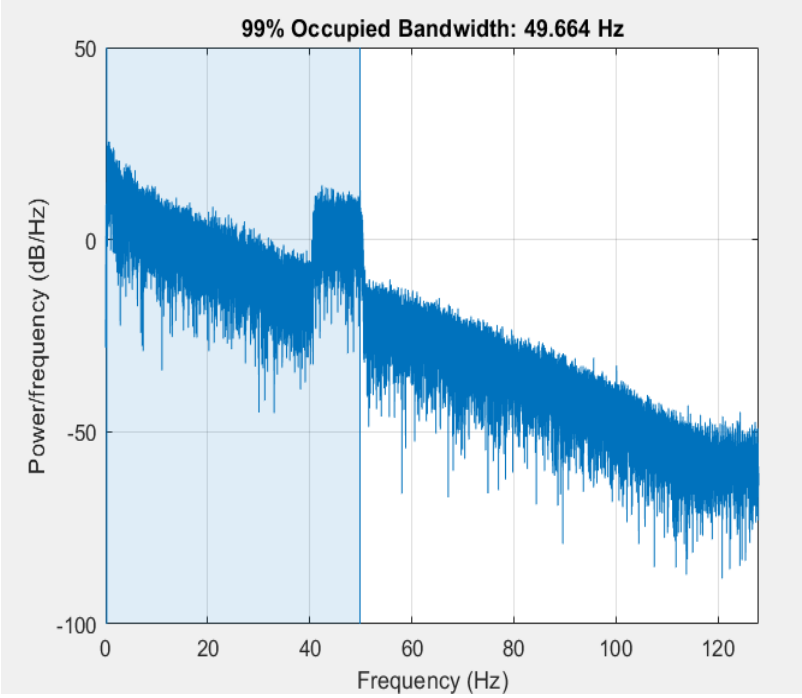


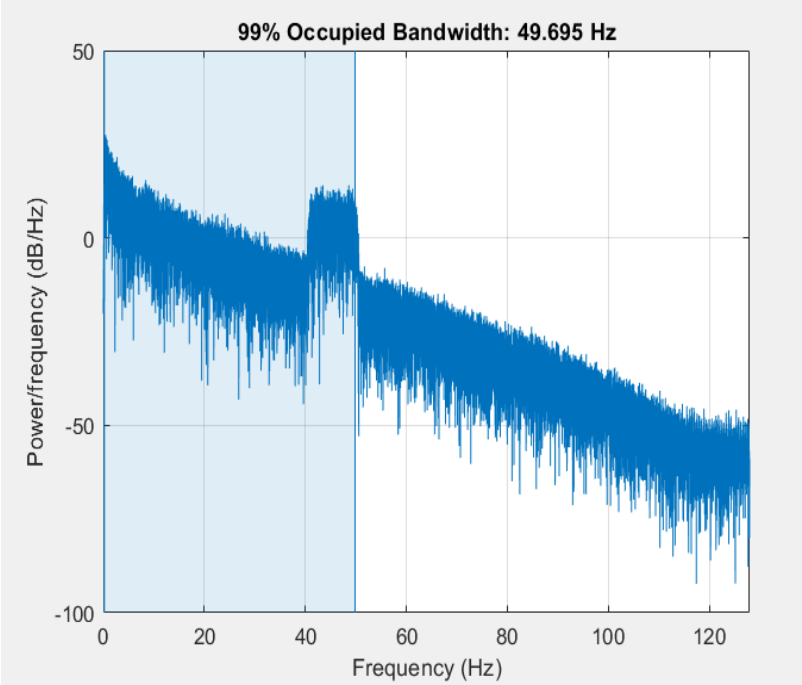

99% Occupied Bandwidth: 49.619 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
50
0
-50
-100
0
20
40
60
80
100
120

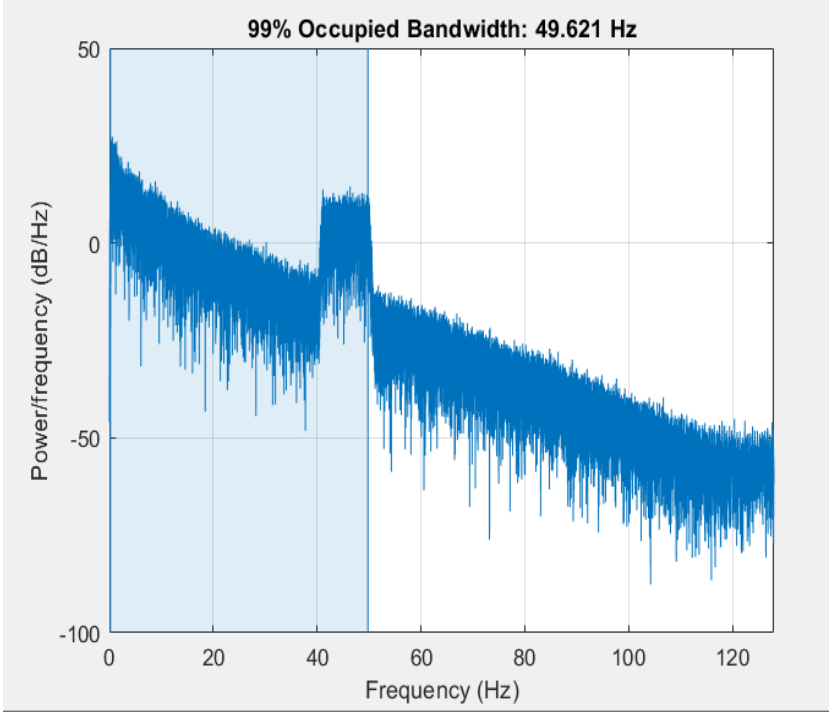

99% Occupied Bandwidth: 49.630 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
50
0
-50
-100
0
20
40
60
80
100
120


99% Occupied Bandwidth: 49.557 Hz
50
0
-50
-100
Power/frequency (dB/Hz)
0
20
40
60
80
100
120
Frequency (Hz)


99% Occupied Bandwidth: 49.589 Hz
50
0
-50
-100
Power/frequency (dB/Hz)
0
20
40
60
80
100
120
Frequency (Hz)


99% Occupied Bandwidth: 49.664 Hz
50
0
-50
-100
Power/frequency (dB/Hz)
0
20
40
60
80
100
120
Frequency (Hz)

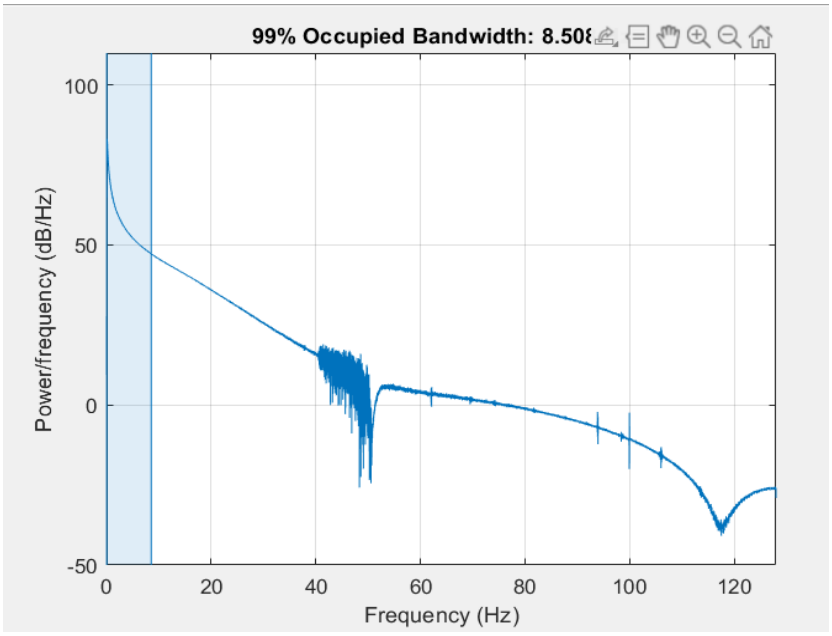

99% Occupied Bandwidth: 49.695 Hz
50
0
-50
-100
Power/frequency (dB/Hz)
0
20
40
60
80
100
120
Frequency (Hz)

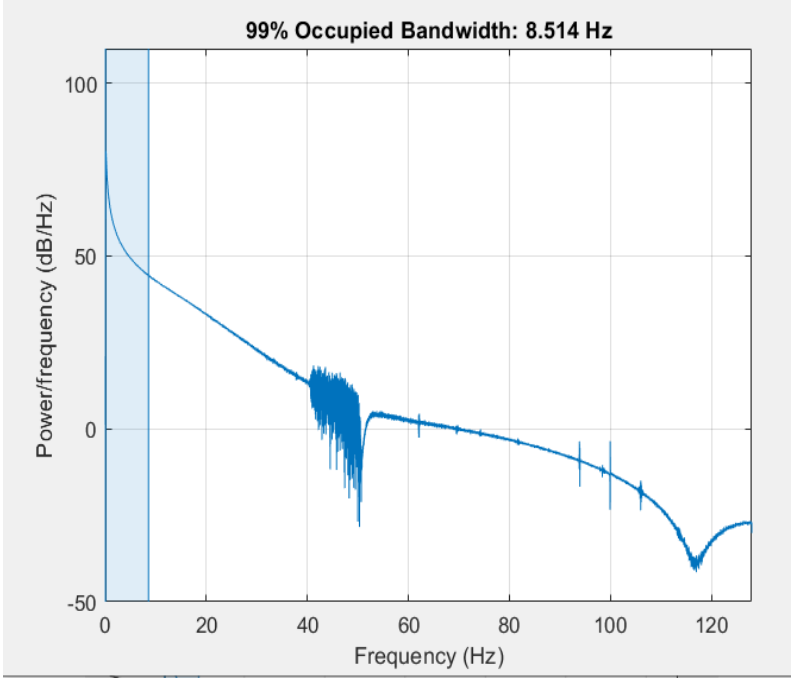


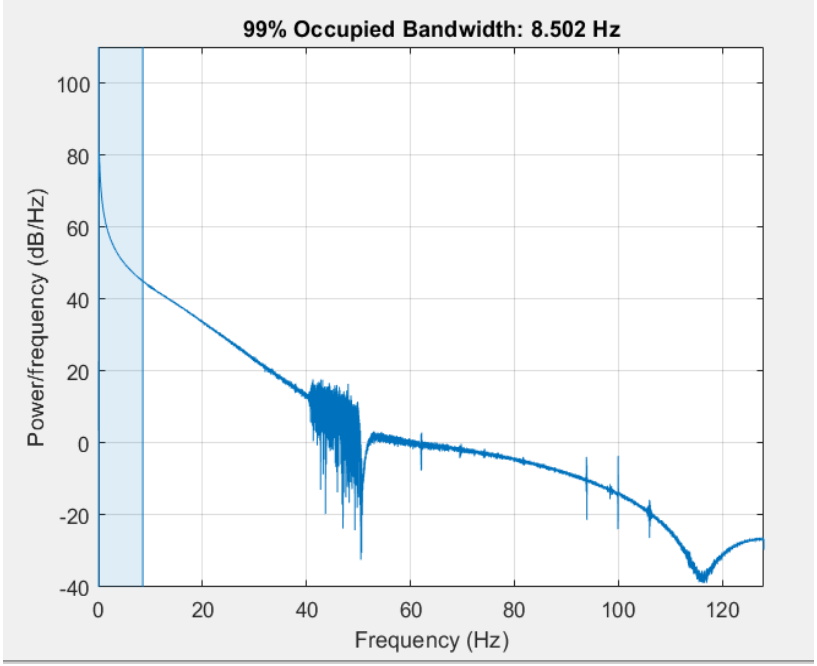
برای dataset2,3 هم نمودارها بسیار مشابه نمودارهای بالا هستند.همان طور که انتظار داشتیم در حدود 50 هرتز می توان سیگنال را قطع کرد.

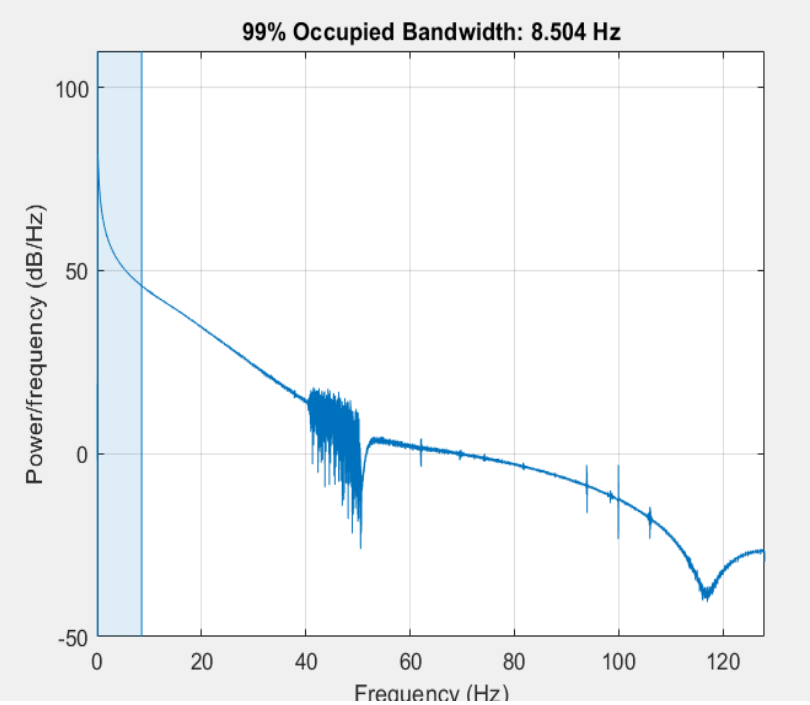
برای dataset5 نمودارها در زیر موجودند.

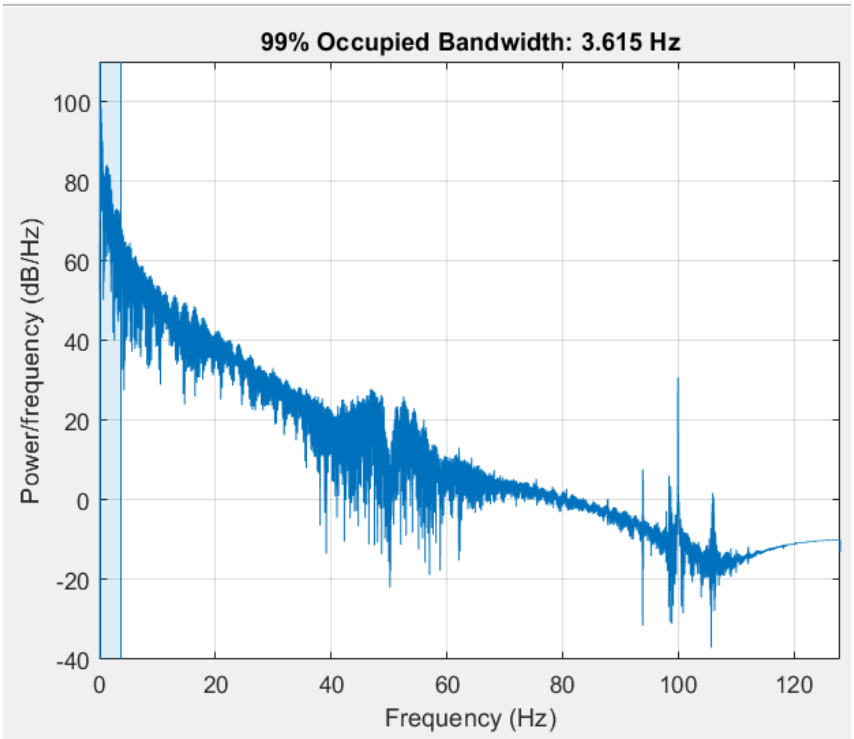
Dataset5:

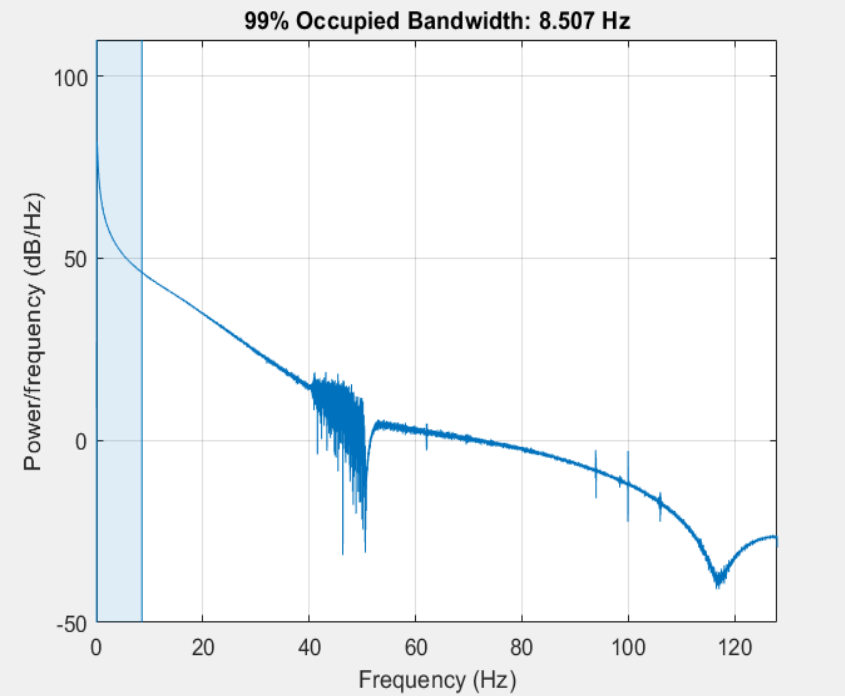


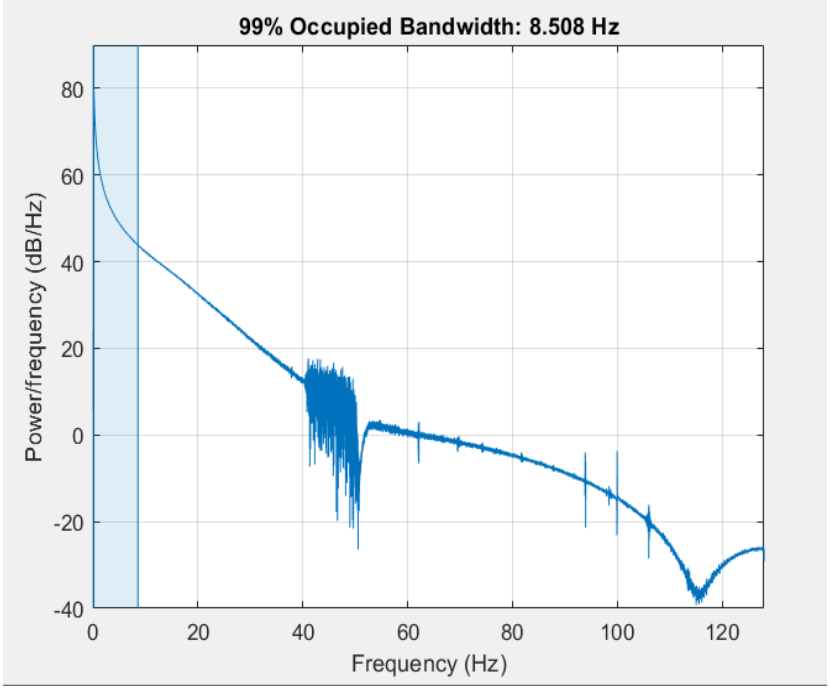

99% Occupied Bandwidth: 8.514 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
100
50
0
-50
0
20
40
60
80
100
120

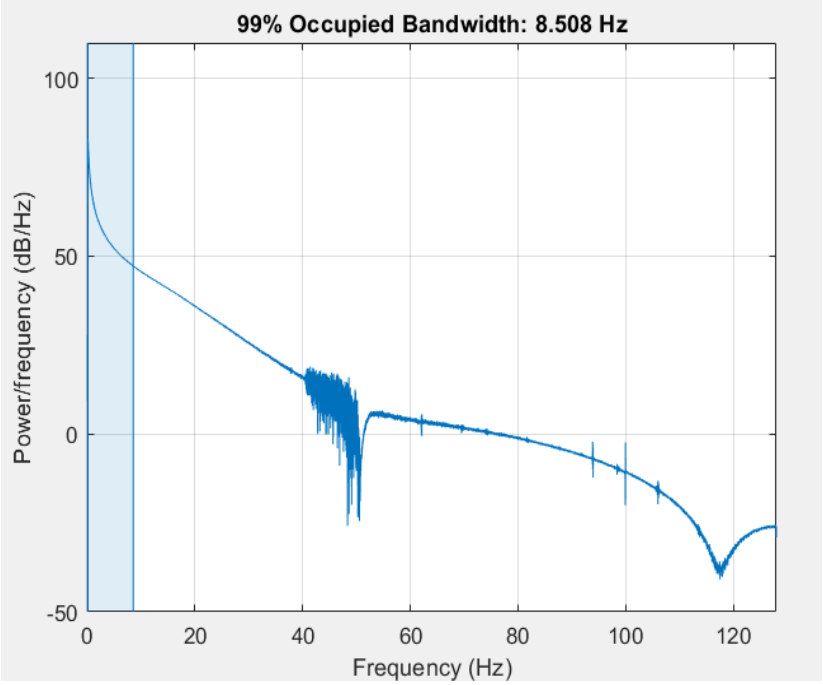

99% Occupied Bandwidth: 8.502 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
100
80
60
40
20
0
-20
-40
0
20
40
60
80
100
120


99% Occupied Bandwidth: 8.504 Hz
Power/frequency (dB/Hz)
100
50
0
-50
0
20
40
60
80
100
120
Frequency (Hz)


99% Occupied Bandwidth: 3.615 Hz
Power/frequency (dB/Hz)
100
80
60
40
20
0
-20
-40
0
20
40
60
80
100
120
Frequency (Hz)


99% Occupied Bandwidth: 8.507 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
100
50
0
-50
0
20
40
60
80
100
120


99% Occupied Bandwidth: 8.508 Hz
Power/frequency (dB/Hz)
Frequency (Hz)
80
60
40
20
0
-20
-40
0
20
40
60
80
100
120

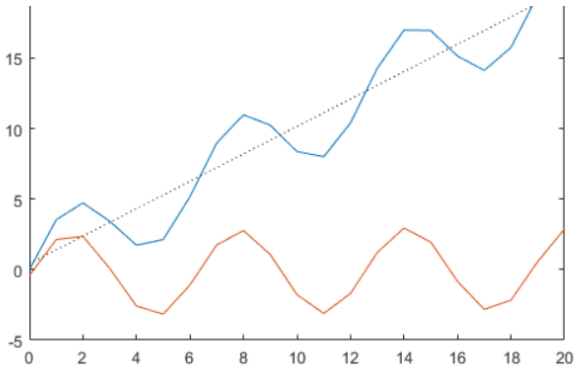


برایdataset6,7,8,9 هم نمودارها بسیار شبیه نمودار های بالاست.به دلیل این که بخش DC سیگنال انرژی زیادی دارد،فرکانس تشخیص داده شده خطا دارد ولی با توجه به نمودارها می توان دید که این فرکانس قطع 50هرتز برای این دیتاست ها هم مناسب است.

فرکانس قطع نهایی که برای فیلتر پایین گذر استفاده شده است 64 هرتز است که هم به 50 نزدیک است و هم در مراحل بعدی برای downsample کردن و کاهش فرکانس،نتایج بهتری به دست می دهد.

تاثیر حذف میانگین بر بخش DC سیگنال:

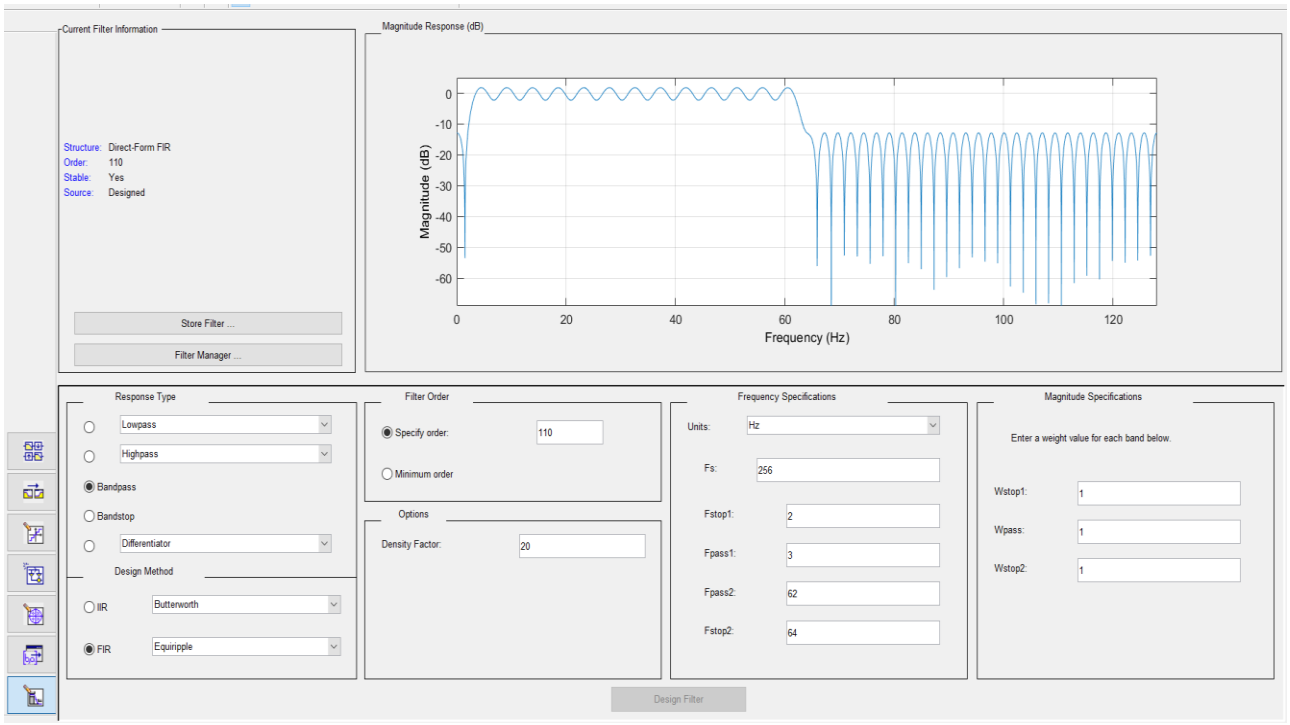
حذف میانگین به طور کامل نمی تواند مقدار DC را حذف کند زیرا ممکن است سیگنال بخشی به صورت زیر داشته باشد که برای پردازش مشکل ایجاد کند :



برای حذف این بخش می توان از تابع detrend استفاده کرد ولی حتی این تابع هم توانایی حذف baseline هایی تناوبی با فرکانس کم را ندارد.برای این منظور از فیلترهای بالاگذر استفاده می کنند.

همچنین اکثر نویز ها دارای میانگین صفر هستند که حذف میانگین تاثیری در حذف آن ها ندارد و برای حذف بهتر بخش DC آن ها بهتر است از فیلتر های بالا گذر استفاده کنیم.

برای ساخت فیلتر از fdatool استفاده شده و تنظیمات به صورت زیر انجام شده است:



برای حذف بخش DC ، فرکانس قطع پایین این فیلتر پایین گذر 3 و فرکانس قطع بالا 64 است.برای این که طول فیلتر از طول سیگنال اصلی و epoch ها هم کمتر باشد،در بخش filter order مقدار 110 برای طول فیلتر در نظر گرفته شده است.

اگر بخواهیم بخشی از سیگنال که در حدود فرکانس های 30 تا 40 هرتز است و نسبت به سیگنال اصلی دامنه ی کمتری دارد را هم حذف کنیم می توانیم فرکانس قطع بالا را 30 یا 40 هرتز در نظر بگیریم.

در ادامه با استفاده از mean و repmat میانگین سیگنال را از آن کم می کنیم.برای اعمال فیلتر ابتدا فیلتر bandpassfilter را لود می کنیم و سپس با استفاده از تابع groupdelay و zerophaesfilter ،فیلتر را بر سیگنال اعمال می کنیم.این توابع در بخش filter شرح داده خواهند شد.توابع نوشته شده برای حذف group delay تنها بر روی داده های نفر اول اعمال شده اند و برای dataset های بعدی از تابع filtfilt که خود groupdelay را حذف می کند استفاده شده است تا دقت افزایش یابد.

کاهش فرکانس نمونه برداری:

با توجه به این که فرکانس قطع بالای فیلتر میان گذر 64 انتخاب شده است،پس فرکانس نایکوییست 128 هرتز است.پس فرکانس نمونه برداری از 256 به 128 می تواند کاهش یابد.پس با dpwn sample با مقدار2 می توان فرکانس نمونه برداری را نصف کرد و به 128 رساند.این کار با تابع downsampler انجام شده است که سیگنال و مقداری که می خواهیم با آن down sample شود را دریافت می کند(n) و سپس در یک حلقه for از هر n نمونه ی سیگنال اصلی ،یکی را ثبت می کند.

تشکیل epoch :

تابع epochconsructor برای این بخش نوشته شده است که سیگنال ورودی،backwardsample و forwardsample و stimulionset و فرکانس نمونه برداری را دریافت می کند.stimulionset همان سطر 10 ماتریس های test,train است که نقاط تحریک را مشخص می کند و فرکانس نمونه برداری هم 128 است چرا که فرکانس پس از downsample کردن با 2 ،نصف می شود.دراین تابع ابتدا با توجه به backwardample و forwardsample و فرکانس نمونه برداری می توان تعداد سمپل هایی که در یک پنجره باید ثبت شوند را محاسبه کرد. سپس با استفاده از stimulionset اندیس هایی که دارای تحریک هستند به دست می آید.برای حذف تحریک های تکراری از این روش استفاده می کنیم که تنها اندیس هایی را به عنوان تحریک ثبت می کنیم که اختلاف اندیس شان با تحریک قبلی بیش از 1 باشد.در این حالت اگر دو تحریک تنها یک اندیس اختلاف داشته باشند،یک تحریک یکسان در نظر گرفته می شوند.در نهایت در یک حلقه یfor ماتریس epoch تشکیل می شود که بعد اول آن شماره ی الکترود،بعد دوم آن شماره ی تریال و بعد سوم آن داده های هر تریال است که یا توجه به مقادیری که بالاتر ذکر شد،تعداد داده ی مناسب در هر تریال ثبت می شود.

چرا باید قبل از کاهش فرکانس نمونه برداری سیگنال را فیلتر کنیم؟

فیلتر کردن باعث می شود پهنای باند سیگنال کاهش یابد و در نتیجه فرکانس نایکوییست کاهش یابد و بتوانیم فرکانس نمونه برداری را مقادیر کمتری اتخاذ کنیم.اگر ابتدا فیلتر نکنیم محتوای فرکانسی در فرکانس های بالاتری باقی خواهد ماند و در اثر کاهش فرکانس نمونه برداری aliasing رخ می دهد.این فیلتر در واقع می تواند مانند فیلتر antialiasing عمل کند.

تحلیل ترتیب زمانی filter و epoch :

اگر ابتدا epoch کنیم و سپس فیلتر کنیم به دلیل افزایش لبه های سیگنال و این که عملکرد فیلتر درلبه ها مطلوب نیست،تاثیر فیلتر مناسب نخواهد بود و سیگنال هایی که در لبه ها وجود دارند به درستی فیلتر نخواهند شد به همین دلیل بهتر است ابتدا فیلتر و سپس epoch کنیم.

عدم ثبت داده مدت زمانی پس از شروع داده گیری:

با شروع آزمایش نیاز به مدت زمان کوتاهی داریم تا نورون ها فعالیت خود را آغاز کنند و پاسخ گذرا را نباید ثبت کنیم در این صورت داده ها دچار خطا خواهند بود و نویز سیگنال افزایش می یابد.

خوشه بندی بر مبنای همبستگی

اثبات اول :

ابتدا برای اثبات نیاز داریم که بین انتگرال یک سیگنال وضرب داخلی یک تناظر ایجاد کنیم:

خواص ضرب داخلی به شرح زیر است:

1. $v.v \geqslant 0$
2. $v.v = 0 \rightarrow v = 0$
3. $(u.v).z = u.z + v.z$
4. $au.v = a(u.v)$

همچنین در مورد انتگرال دو عدد نیز می دانیم.

1. $\int_{-\infty}^{\infty} X(t)^2 dt \geq 0$
2. $\int_{-\infty}^{\infty} X(t)^2 dt = 0 \leftrightarrow X(t) = 0$
3. $\int_{-\infty}^{\infty} (X(t) + Y(t))Z(t)dt = \int_{-\infty}^{\infty} X(t)Z(t)dt + \int_{-\infty}^{\infty} Y(t)Z(t)dt$
4. $\int_{-\infty}^{\infty} aX(t)Y(t)dt = a\int_{-\infty}^{\infty} X(t)Y(t)dt$

پس می توان ضرب داخلی را این گونه بازگویی کرد:

$$\int_{-\infty}^{\infty} X(t)Y(t)dt = X(t).Y(t)$$

همچنین از خواص ضرب داخلی می دانیم:

$$||u||^2 = u.u$$

پس می توان گفت:

$$||X(t)^2|| = \int_{-\infty}^{\infty} X(t)^2 dt \quad same\ as \quad Y(t)$$

$$\left|\int_{-\infty}^{\infty} X(t)Y(t)dt\right|^2 \leq ||X(t)^2||\,||Y(t)^2|| = \int_{-\infty}^{\infty} X(t)^2 dt \int_{-\infty}^{\infty} Y(t)^2 dt$$

$$\frac{\left|\int_{-\infty}^{\infty} X(t)Y(t)dt\right|^2}{\int_{-\infty}^{\infty} X(t)^2 dt \int_{-\infty}^{\infty} Y(t)^2 dt} \leq 1 \Rightarrow -1 \leq \frac{\left|\int_{-\infty}^{\infty} X(t)Y(t)dt\right|}{\sqrt{\int_{-\infty}^{\infty} X(t)^2 dt \int_{-\infty}^{\infty} Y(t)^2 dt}} \leq 1$$

اثبات دوم :

برای اثبات دوم ابتدا طرف اول را اثبات می کنیم : که در صورت X(t)=aY(t) رابطه rxy=1 برقرار است.

بر اساس جایگذاری:

$$\left|\frac{\int_{-\infty}^{\infty}\alpha X(t)^2}{\sqrt{\int_{-\infty}^{\infty}\alpha^2 X(t)^2\int_{-\infty}^{\infty}X(t)^2}}\right| = \left|\frac{\alpha\int_{-\infty}^{\infty}X(t)^2}{\alpha\sqrt{\int_{-\infty}^{\infty}X(t)^2\int_{-\infty}^{\infty}X(t)^2}}\right| = \left|\frac{\int_{-\infty}^{\infty}X(t)^2}{\left|\int_{-\infty}^{\infty}X(t)^2\right|}\right| = 1$$

نامساوی کوشی شوارتز بدین ترتیب می باشد:

$$u.v \leqslant \|u\|\,\|v\|$$

اگر u و v را بدین صورت تعریف کنیم:

$$u = \frac{u.v}{\|v^2\|}v + w$$

تنها در صورتی در رابطه کوشی شوارتز روابط مساوی برقرار می شود که w برابر صفر باشد. در نتیجه u ضریبی v است.

با توجه به روابطی که بین انتگرال و ضرب داخلی دیدیم برای قرار معادل کوشی شوارتز نیز باید Y ضریبی از X باشد.

چرا همبستگی معیار مناسبی برای خوشه بندی است؟

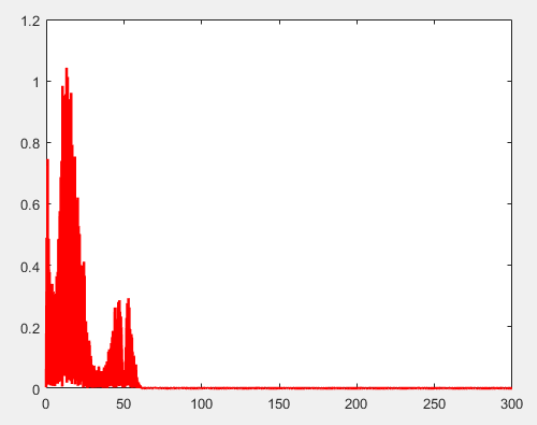
از آنجایی که متوجه شدیم زمانی که اندازه rxy حداکثر و برابر یک هست دو متغیر X و Y رابطه خطی با یکدیگر دارند ودارای بیشترین وابستگی بهم هستند و به همین ترتیب هر چه رابطه بین دو متغیر خطی تر و وابسته تر باشند این ضریب به یک نزدیک تر است بنابراین این معیار می تواند به خوبی تابع بودن خطی یک سیگنال تحت یک سیگنال دیگر را به نمایش بگذارد.

دلیل دیگر آن است که همبستگی در اصل برحسب کواریانس دو عدد است و هر چه کواریانس ترکیبی دو سیگنال کمتر باشد دو سیگنال شباهت بیشتری دارند.

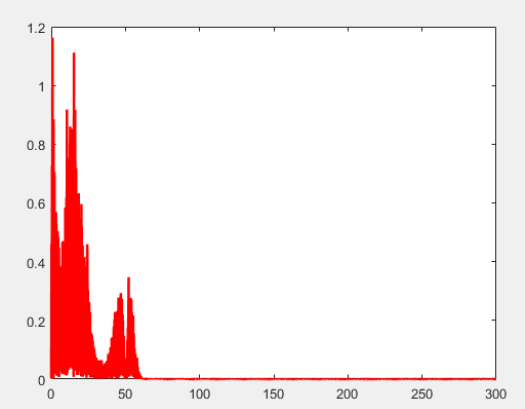
در خوشه بندی کانال های مختلف EEG نیز متوجه می شویم اگر دو سیگنال تنها در دامنه متفاوت باشند اطلاعاتی مشابه هم دارند و بهم وابسته هستند.

با استفاده از تابع HalfBandFFT2 که در بخش های قبل توضیح داده شده طیف فرکانسی به صورت زیر می باشد:

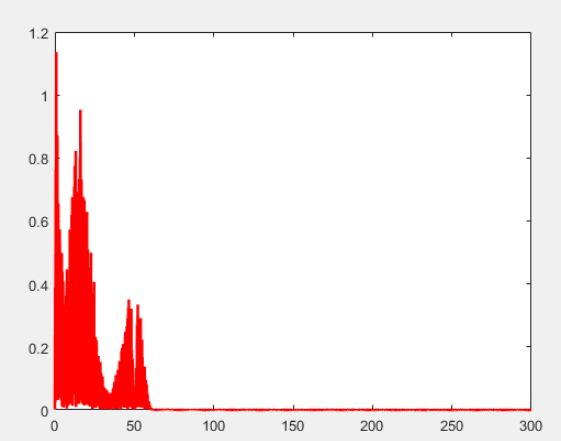
طیف فرکانسی سه داده اول رسم شده است:



کانال اول



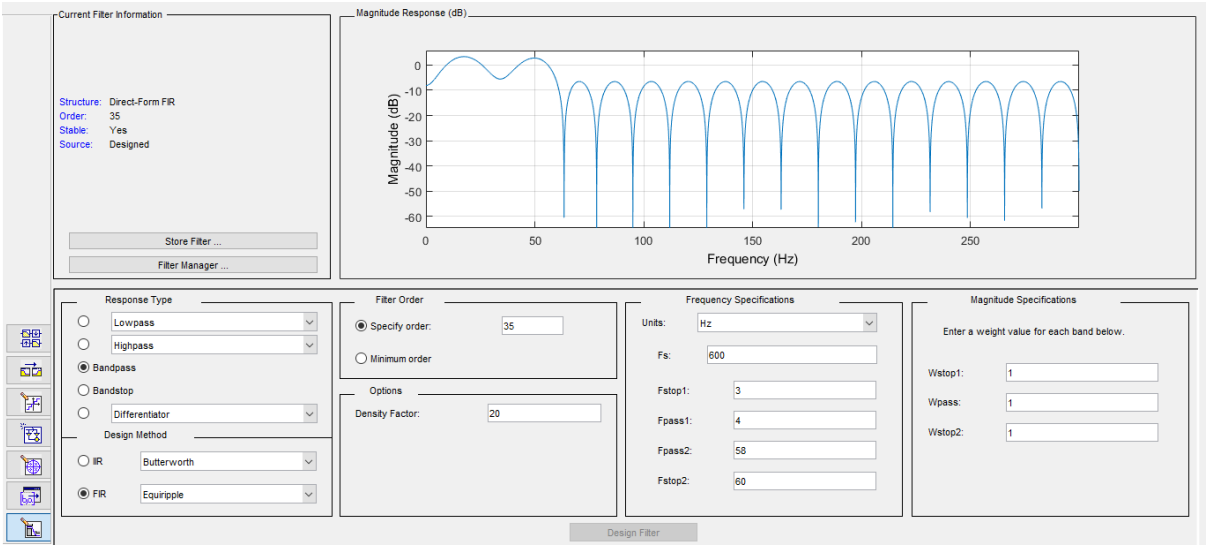
کانال دوم



کانال سوم

ابتدا با استفاده از دستور reshape داده Epoch شده را به یک ماتریس دو بعدی تبدیل می کنیم که در هر سطر داده های یک کانال در قرار دارد. همچنین برای کاهش اثر DC میانگین سیگنال را از آن کم می کنیم.

با مشاهده طیف فرکانسی تصمیم میگیریم از فرکانس 60 به بعد را حذف می کنیم. این کار را استفاده از امکانات fdatool و با تنظیمات زیر انجام می دهیم :



توجه داریم فرکانس نمونه برداری را برابر 600 قرار دهیم. همچنین فرکانس های خیلی پایین را حذف می کنیم تا اثر DC کم بشود. همجنین به عنوان filter order عدد 35 را قرار می دهیم تا طبق بخش های قبل طول فیلتر از طول داده کمتر شود.

حال فیلتر ذکر شده را با فرمت mat. ذخیره کرده وضرایب آن استفاده می کنیم. با استفاده از تابع filtfilt و ضرایب گفته شده سیگنال مربوطه به هر کانال را فیلتر می کنیم تا group delay نیز حذف شود. البته در صورتی که بخواهیم دقیق تر فیلتر کنیم می توانیم از فرکانس 40 به بعد فیلتر کنیم.

البته توجه داریم از آنجایی این داده ها پس از Epoching هستند فیلتر شده اند و فیتر دوباره ما تاثیر خاصی بر داده ها نداره.

از آنجایی که فرکانس قطع هم اکنون برابر 60 Hz است و فرکانس نایکوئیست برابر 120 می شود. بنابراین حداقل فرکانسی که می توانیم به آن کاهش دهیم برابر با 120 می باشد. بنابراین ضریب Downsapling برابر است با: $\frac{600}{120} = 5$

حال با تابع آماده متلب این down sampling را انجام می دهیم.

به منظور محاسبه همبستگی بین سیگنال های کانال ها این توابع را تعریف کرده ایم:

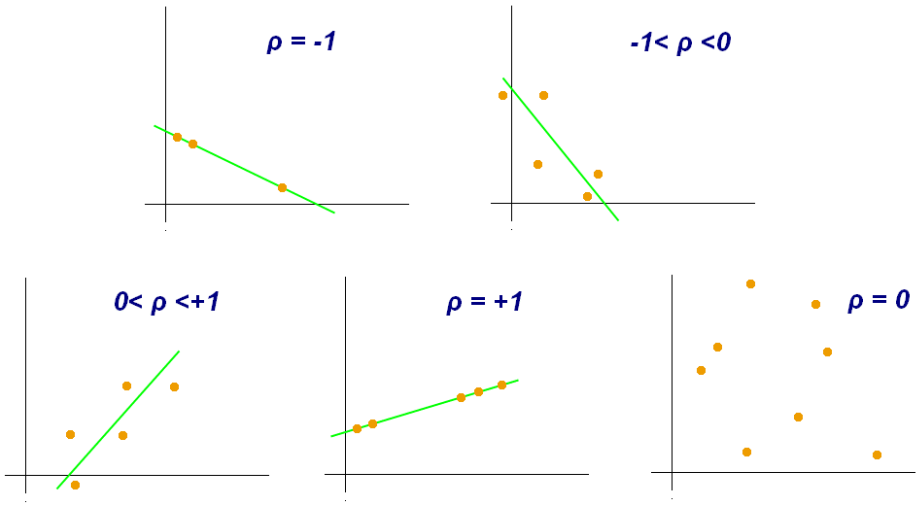
```
1.function corr=correlation(X,Y)
```

این تابع با گرفتن دو بردار مربوط به دو سیگنال با استفاده از فرمول داده در دستور کار همبستگی بین دو عدد را محاسبه می کند. برای محاسبه انتگرال از دستور sum استفاده می شود.

```
2.function rxy=correlationMatrix(data,numOfChanell)
```

این تابع با دریافت ماتریس حاصل از 63کانال و با استفاده از تابع correlation یک ماتریس 63 در 63 بر میگرداند که درایه (i,j) آن بیانگر همبستگی بین دو کانال i ام و j ام می باشد. حال خروجی این تابع را rxy می نامیم.

می دانیم در صورتی که ضریب همبستگی برابر یک باشد بیشترین وابستگی بین دو کانال وجود دارد بنابراین ماتریس فاصله (distance) را به صورت 1-abs(rxy) تعریف می کنیم.



سه راه را برای خوشه بندی برحسب فاصله که پیش تر تعریف کرده ایم می آوریم:

1. راه یک: الگوریتم K means

Kداده را به عنوان مرکز خوشه انتخاب می کنیم، سپس فواصل بقیه داده ها با مرکز خوشه ها را تعیین میکنیم و داده هایی که به مرکز هر خوشه نزدیکتر هستند را در آن خوشه قرار میدهیم. میانگین هر خوشه را به عنوان مرکز جدید خوشه انتخاب میکنیم این مراحل را تا زمانی ادامه میدهیم که خوشه ها بدون تغییر باقی بمانند. در ادامه مراحل الگوریتم K-means بیان شده است.

ورودی: خصوصیات n داده و k تعداد دسته ها

خروجی :k دسته که داده های هر دسته از نظر شباهت به هم نزدیک و از دسته های دیگر دورند

- داده را به عنوان مرکز خوشه انتخاب می کنیم.
- مرحله سوم تا پنجم را تا رسیدن به عدم تغییر در خوشه ها تکرار می کنیم .
- فواصل بقیه داده ها با مرکز خوشه ها را تعیین می کنیم.

- داده هایی که به مرکز هر خوشه نزدیکترند در آن خوشه قرار می گیرند.
- میانگین هر خوشه را به عنوان مرکز جدید خوشه در نظر می گیریم.

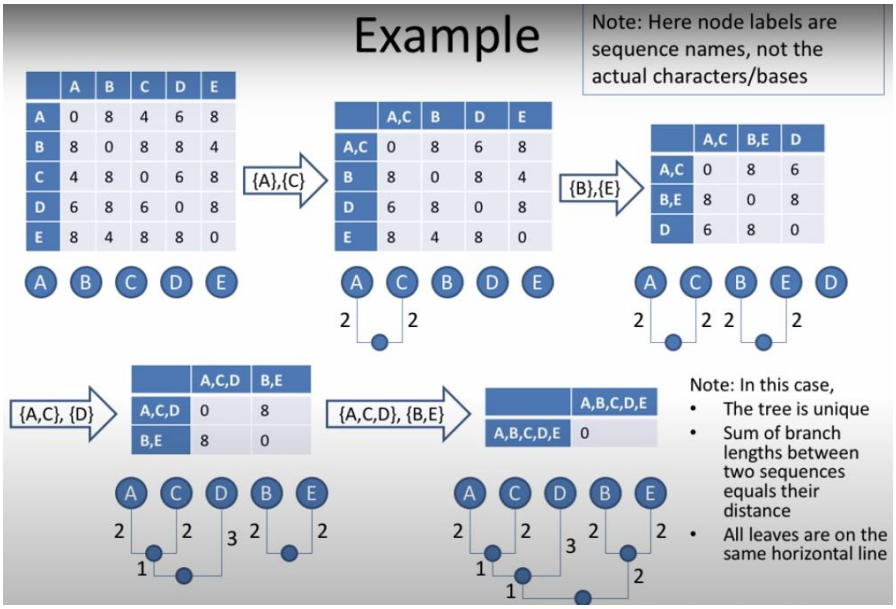
به طور معمول، مرکزخوشه های اولیه به صورت تصادفی از میان نمونه های اولیه گزینش می شوند. به همین دلیل، مرکز خوشه های اولیه در دو خوشه بندی مستقلk-means می توانند متفاوت باشند. این موضوع موجب می شود که خوشه های به جا مانده از دو اجرای مختلف K-means با هم متفاوت باشند. بنابراین همواره به بهینه ی سراسری نمی رسد اما ممکن است به بهینه ی محلی برسد.

2. UPGMA

این الگوریتم بدین صورت عمل می کند:

ابتدا از روی ماتریس فاصله کمترین فاصله را تعیین می کنیم و این دو کانال را با هم ادغام می کنیم سپس برای فواصل جدید این کانال ادغام شده ، میانگین بدون وزن از فواصل دو خوشه ای که ادغام شده اند را به عنوان فاصله جدید قرار می دهیم و ماتریس distance جدید را می سازیم. این کار تا مرحله مورد نظر ما ادامه پیدا می کند و در نهایت همه کانال ها می توانند در یک خوشه قرار بگیرند.

یک مثال از آن به صورت زیر می باشد:



3. WPGMA

این الگوریتم بسیار شبیه به الگوریتم قبلی است با این تفاوت که به جای میانگین بدون وزن بر اساس تعداد هر دو خوشه ای که با هم ادغام می شوند میانگین وزن دار گرفته می شود و بقیه مراحل یکسان است.

برای فاصله دو خوشه نیز به جز همبستگی از روابط روبرو می توان استفاده کرد:

Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
Maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the Covariance matrix

که بهترین آنها در این همان همبستگی است.

به منظور خوشه بندی تابعی به صورت زیر تعریف شده است:

```
function [Group Dmatrix]=UPGMA(numOfStage,Dmatrix)
```

این تابع با دریافت ماتریس فاصله (Dmatrix) و همچنین تعداد مراحل خوشه بندی (numOfStage) بر اساس الگوریتم دوم بیان شده خوشه بندی را انجام می دهد.

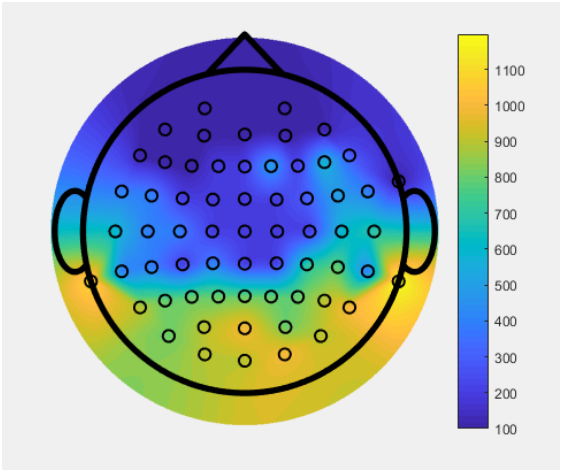
ابتدا درایه های روی قطر اصلی را با استفاده از دستورFind و یک حلقه for برابر دو قرار می دهیم. سپس داده های ماتریس را با استفاده از دستور Sort به صورت زیر مرتب می کنیم.

```
sorted=sort(Dmatrix(:))
```

حال درایه اول Sorted را در ماتریس فاصله می یابیم. در مختصات با عدد کوچکتر میانگین فواصل دو مختصات را در سطر و ستون آن قرار می دهیم و سطر وستون مختصات بزرگتر را برابر NaN قرار می دهیم تا در محاسبات بعدی بی تاثیر باشد. همچنین ماتریس همانی در ابتدا ساخته ایم که 63 در 63 می باشد و نام آن Group می باشد. در صورتی که دو کانال i و j (i<j) با هم ادغام شوند درایه (i,j) یک شده و درایه قطری J برابر NaN می شود. (با استفاده از این ماتریس در ادامه کانال هایی که در یک خوشه هستند را متوجه می شویم)

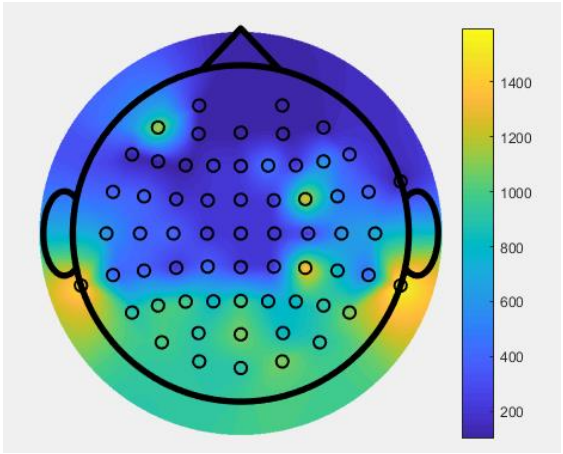
حال دوباره در ماتریس فاصله با استفاده از دستور Diag درایه های روی قطر را صفر می کنیم. همچنین با استفاده از سایز اصلی ماتریس فاصله که همان تعداد کانال و همچنین تعداد NaN ماتریس group که بیانگر تعداد کانال ادغام شده است مرحله طی شده بدست می آید. این کار تا زمانی که به تعداد مرحله لازم برسیم ادامه پیدا می کند.

توجه به کارکرد قسمت های مختلف مغز می تواند معیار خوبی برای خوشه بندی باشد. برای مثال الکترود های روبروی بینی برای تشخیص پلک چشم هستند و باید در یک خوشه بندی قرار بگیرند و یا الکترود های زیر گوش بهم مرتبط هستند. همچنین الکترود های پس سری نیز به هم وابسته بر این اساس اگر خوشه بندی را تا مرحله ای ادامه دهیم که به 12 خوشه برسیم می تواند نتیجه نسبتا مطلوبی بگیریم:

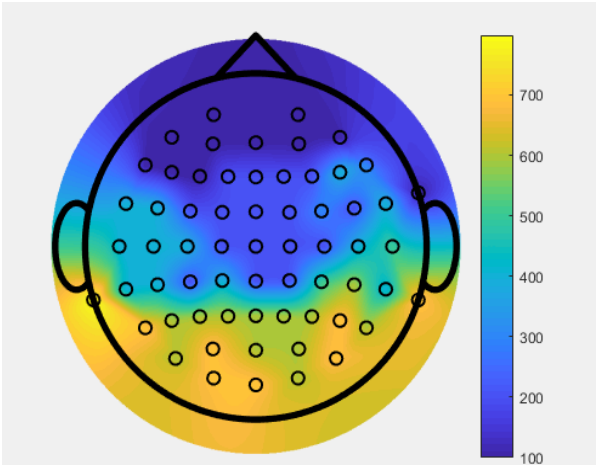


همچنین دو خوشه بندی دیگر را نیز ذکر می کنیم:

16 خوشه:



8 خوشه:



با استفاده از شهودی که داریم به این نتیجه می رسیم که الکترود های قرینه سمت راست و چپ عملکرد مشابهی دارند. همچنین الکترود های پس سری و همچنین الکترود های جلوی سر به هم وابسته هستند. همچنین هر چه الکترود ها فاصله فیزیکی کمتری روی مغز داشته باشند عملکرد وابسته تری نسبت به هم دارند.

با توجه به خوشه بندی نیز الکترود هایی که نزدیک به هم هستند و یا نسبت به سمت چپ و راست قرینه هستند در یک خوشه قرار می گیرند.

معیار های دیگری که برای توقف خوشه بندی می توانیم استفاده کنیم عبارت اند از:

1. مینمم فاصله برای دو خوشه:

می توانیم با توجه به این کمترین فاصله بین سیگنال دو کانال تا چه اندازه باشد خوشه بندی را ادامه دهیم.

2. تعداد مراحل طی شده

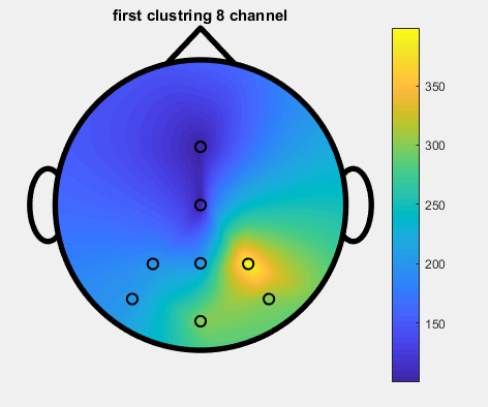
که از این راه در تابع نوشته شده استفاده شده است و بر اساس عملکرد مغز بهترین تعداد مرحله در نظر گرفته شده است.

دیتاست 8 الکترودی:

با توجه به این موضوع که ما صرفا اسامی این 8 الکترود را می دانیم و نمی دانیم هر اسم متعلق به کدام الکترود است باید با استفاده از خوشه بندی و این موضوع که کدام الکترود ها در یک خوشه قرار می گیرند این اسامی حدودی مشخص شوند. برای این کار از train 4 دیتاست اول استفاده شده است. بدین صورت که که یک بار خوشه بندی با 7 خوشه انجام می دهیم و دو الکترودی که بیشترین وابستگی را بهم دارند در ابتدا در یک خوشه قرار می گیرند و با مقایسه با دیتا 64 الکترودی و ترتیب قرار گرفتن الکترود های مربوطه الکترود متناظر را پیدا می کنیم. (اسامی آن 8 الکترود در 64 کانال برابر است با 8 ، 17 ، 59، 27، 25، 58، 60 و 61). به همین ترتیب ادامه می دهیم تا وابستگی خوشه نسبت به هم را از سه دیتاست حدس بزنیم. با توجه به این دیتا ست ها متوجه می شویم الکترود 1 و 2 تا مرحله نسبتا زیای از هر دیتاست با هم در یک گروه تنها می باشند این اتفاق برای الکترود 8 و17 نیز می افتد همچنین ابتدا الکترود های 4 و 5 در یک گروه قرار می گیرند و بقیه الکترود ها به مرور به گروه آن ها افزوده می شوند که چنین چیزی برای الکترود 58 و 25 می افتد. همچنین الکترود 6 و 7 نیز با در مراحل بعدی با هم ادغام شده و سپس به گروه 4 و5 افزوده می شوند که مشابه الکترود 60 و 61 است. در انتها نیز دو الکترود 3 و 8 به طور جدا به گروه نهایی اضافه می شوند که شبیه به به الکترود های 27 و 59 می باشد. سپس با توجه به دیتاست های بعدی

و همچنین خطا وآزمایش نمودار خوشه بندی در مغز این گروه های دوتایی به طور تقریبی بهم نسبت می دهیم . در نهایت الکترود ها را بدین ترتیب میگریم:

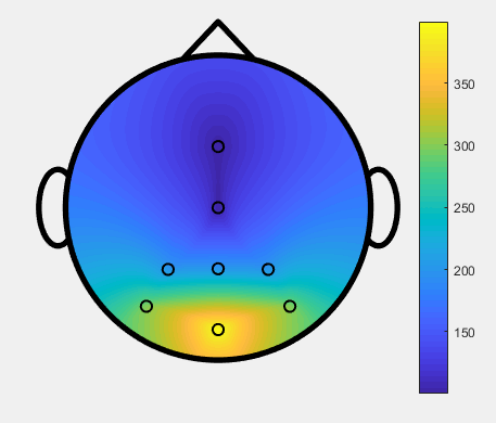
```
ch_list = {'FZ','CZ','PZ','PO7','P3','PO8','OZ','P4'};
```



همچنین با چند آزمایش و توجه به نمودار خوشه ها این دسته بندی نیز مناسب می باشد:

```
ch_list = {'FZ','CZ','P3','PZ','P4','PO7','PO8','OZ'};
```

تصویر آن نیز بدین صورت می باشد:



طراحی فیلتر:

خروجی های حاصل از ورودی های ذکر شده به 2 نوع فیلتر با فاز خطی و وغیر خطی:

فاز خطی:

$$H(w) = A(w)\ e^{j\phi(w)},\ \phi(w) = -\pi/3\ Sign(w)$$

$$x(t) = \cos(w_0 t) + \cos(2w_0 t) =$$

$$\frac{1}{2}\left(e^{jw_0 t} + e^{-jw_0 t}\right) + \frac{1}{2}\left(e^{2jw_0 t} + e^{-2jw_0 t}\right) \Rightarrow$$

$$X(w) = \pi\left(\delta(w-w_0) + \delta(w+w_0)\right) +$$

$$\pi\left(\delta(w-2w_0) + \delta(w+2w_0)\right)$$

$$Y(w) = X(w)H(w) = \pi A(w)\ e^{-j\pi/3\ Sign(w)}\left(\delta(w-w_0) + \delta(w+w_0)\right) +$$

$$\pi A(w)\ e^{-j\pi/3 Sign(w)}\left(\delta(w-2w_0) + \delta(w+2w_0)\right) \Rightarrow$$

$$y(t) = \frac{1}{2}A(w)\left(e^{j(w_0 t - \pi/3)} + e^{j(-w_0 t + \pi/3)}\right) +$$

$$\frac{1}{2}A(w)\left(e^{j(2w_0 t - \pi/3)} + e^{j(-2w_0 t + \pi/3)}\right) =$$

$$A(w)\cos(w_0 t - \pi/3) + A(w)\cos(2w_0 t - \pi/3)$$

فاز غیرخطی:

$$H(w) = A(w)\, e^{jw\phi(w)}, \quad \phi(w) = -\pi/3\, w$$

$$x(t) = \cos(w_0 t) + \cos(2w_0 t) =$$

$$\frac{1}{2}\left(e^{jw_0 t} + e^{-jw_0 t}\right) + \frac{1}{2}\left(e^{2jw_0 t} + e^{-2jw_0 t}\right) \Rightarrow$$

$$X(w) = \pi\left(\delta(w-w_0) + \delta(w+w_0)\right) +$$

$$\pi\left(\delta(w-2w_0) + \delta(w+2w_0)\right)$$

$$Y(w) = X(w)H(w) = \pi A(w) e^{-j\pi/3 w}\left(\delta(w-w_0) + \delta(w+w_0)\right) +$$

$$\pi A(w) e^{-j\pi/3 w}\left(\delta(w-2w_0) + \delta(w+2w_0)\right) \Rightarrow$$

$$y(t) = \frac{1}{2} A(w)\left(e^{jw_0(t-\pi/3)} + e^{-jw_0(t-\pi/3)}\right) +$$

$$\frac{1}{2} A(w)\left(e^{2jw_0(t-\pi/3)} + e^{-2jw_0(t-\pi/3)}\right) =$$

$$A(w)\cos\left(w_0(t-\pi/3)\right) + A(w)\cos\left(2w_0(t-\pi/3)\right)$$

طبق نتایج بالا اگر فاز غیر خطی باشد،دو سیگنال با مقادیر متفاوتی شیفت پیدا می کنند و فرم کلی سیگنال از بین می رود .ولی در حالت دوم هر دو سیگنال با مقدار برابر شیفت پیدا می کنند و سیگنال شکل و ویژگی های خود را حفظ می کند تنها مقداری در زمان جابجا می شود.پس فاز خطی مطلوب ما خواهد بود.

بررسی این گزاره که group delay میزان تاخیر سیگنال پس از فیلتر را مشخص می کند:

برای سیستم اول که دارای فاز خطی نیست نمی توان از رابطه ی gd داده شده استفاده کرد هم چنین همان طور که محاسبه شد دو بخش x(t) مقدار یکسانی تاخیر پیدا نمی کنند و نمی توان تاخیر سیگنال را مشخص نمود.

برای سیستم دوم:

$$gd(w) = \frac{-d\phi(w)}{dw} = -\frac{d}{dw}(-\pi/3\, w) = \pi/3$$

$$x(t-\pi/3) = \cos(w_0(t-\pi/3)) + \cos(2w_0(t-\pi/3))$$

همان طور که مشاهده می کنید مقدار gd برابر pi/3 است و سیگنال نیز دقیقا به همین مقدار شیفت خورده است در نتیجه صحت این عبارت که تاخیر گروه برای هرمحتوای فرکانس مشخص می کند که پس از عبور از فیلترت چه مقدار شیفت خواهد یافت،تایید می شود.

اثبات رابطه:

$$\mathrm{gd}(\omega) = \mathbf{Re}\left\{ \frac{j\frac{d}{d\omega}H(\omega)}{H(\omega)} \right\}$$

$$H(\omega) = A(\omega)\, e^{j\phi(\omega)}$$

$$\ln(H(\omega)) = \ln(A(\omega)) + j\phi(\omega) \longrightarrow$$

$$\frac{d\ln(H(\omega))}{d\omega} = \frac{H'(\omega)}{H(\omega)} = \frac{A'(\omega)}{A(\omega)} + j\phi'(\omega) \longrightarrow$$

$$\frac{jH'(\omega)}{H(\omega)} = \frac{jA'(\omega)}{A(\omega)} - \phi'(\omega) \longrightarrow$$

$$\mathrm{Real}\left(\frac{j\frac{dH(\omega)}{d\omega}}{H(\omega)} \right) = \frac{-d\phi(\omega)}{d\omega}$$

تابع group delay :

این تابع سیگنال فیلتر و N را دریافت می کند و برای تشکیل H(w) از h ، dft با N نقطه می گیرد. برای تشکیل jH'(w) از این نکته استفاده می کنیم که تبدیل فوریه nh(n) برابر این مقدار است.با تقسیم این 2 عبارت و real گرفتن از آن،خروجی مطلوب به دست می آید.

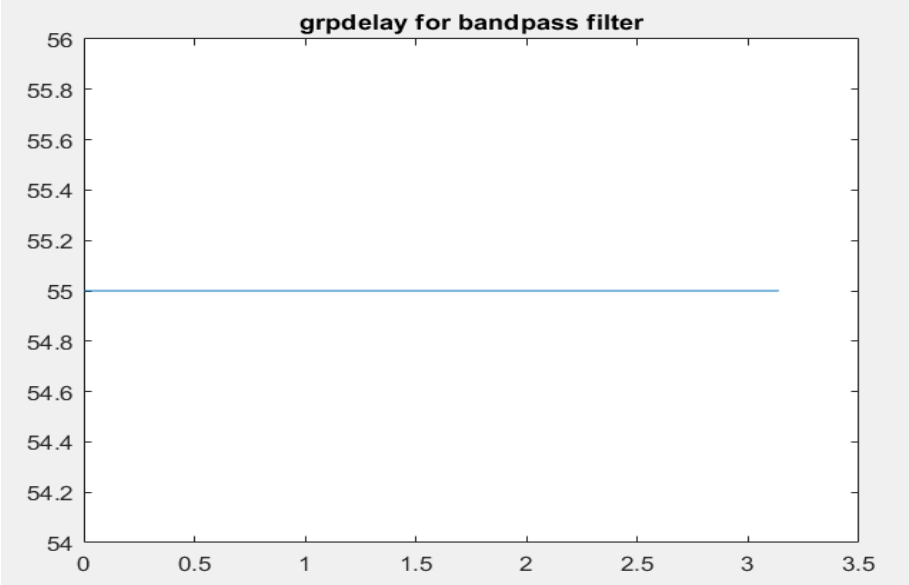
عملکرد DFT با N نقطه به این صورت است که ابتدا سیگنال را با تناوب N متناوب می کند و ضرایب سری فوریه ی این سیگنال را تحت عنوان DFT بر می گرداند.اگر N از طول سیگنال کمتر باشد، بخش های مختلف سیگنال پس از تناوبی شدن،تداخل پیدا می کنند و سیگنال از دست می رود ولی هر چه N بیشتر باشد

aliasing کمتر می شود و سیگنال با دقت بالاتری بازیابی می شود.پس افزایش N باعث افزایش دقت محاسبات می شود.

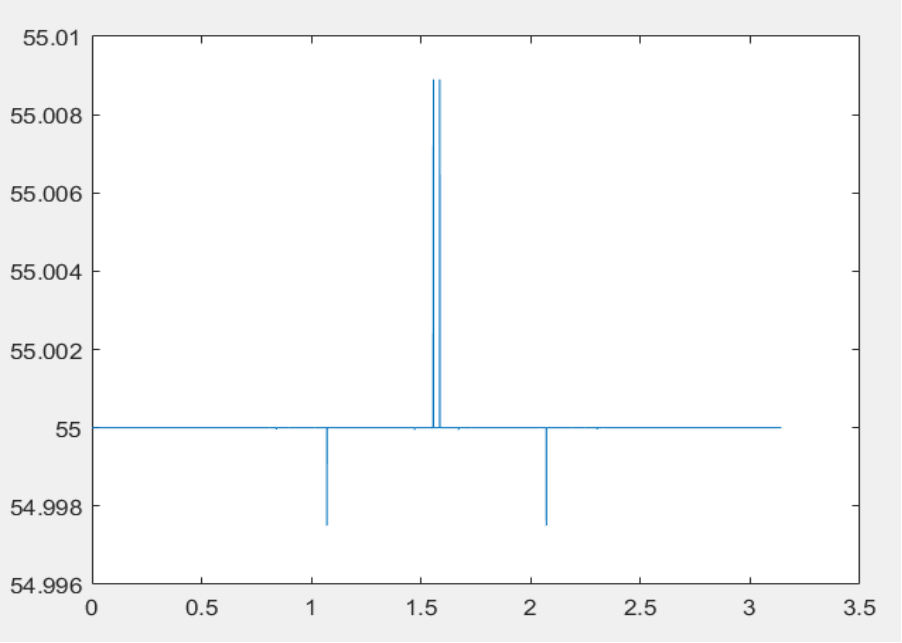
دو فیلتر در طول پروژه استفاده شده است که مقدار gd آن همراه با خروجی تابع اصلی متلب در زیر موجود است.هم چنین نتایج برای فیلتری که در اختیارمان قرار دادید هم رسم شده است.

Bandpass filter:

خروجی grpdelay:



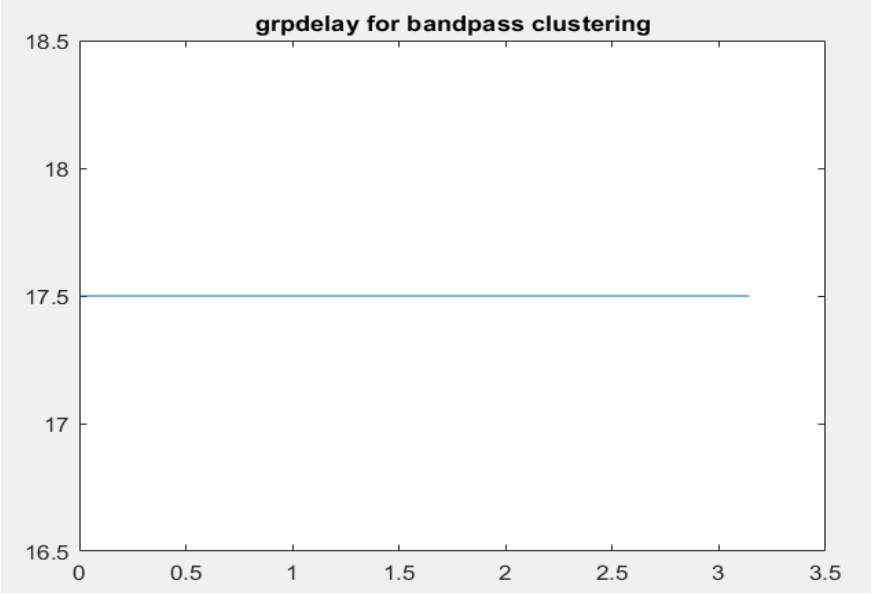
خروجی groupdelay:



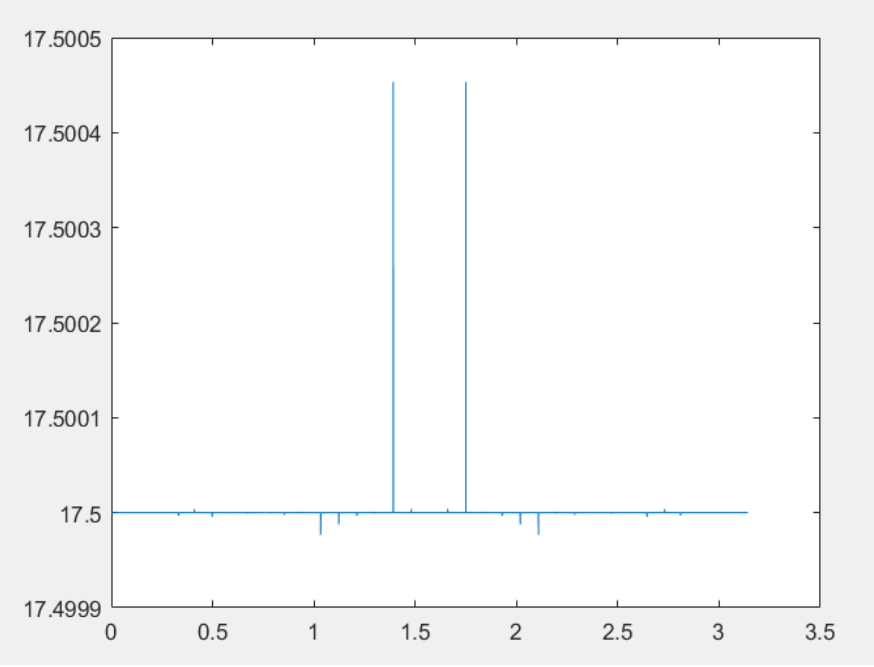
N=1000000 فرض شده است.

bandPass_clustring filter:

grpdelay خروجی:



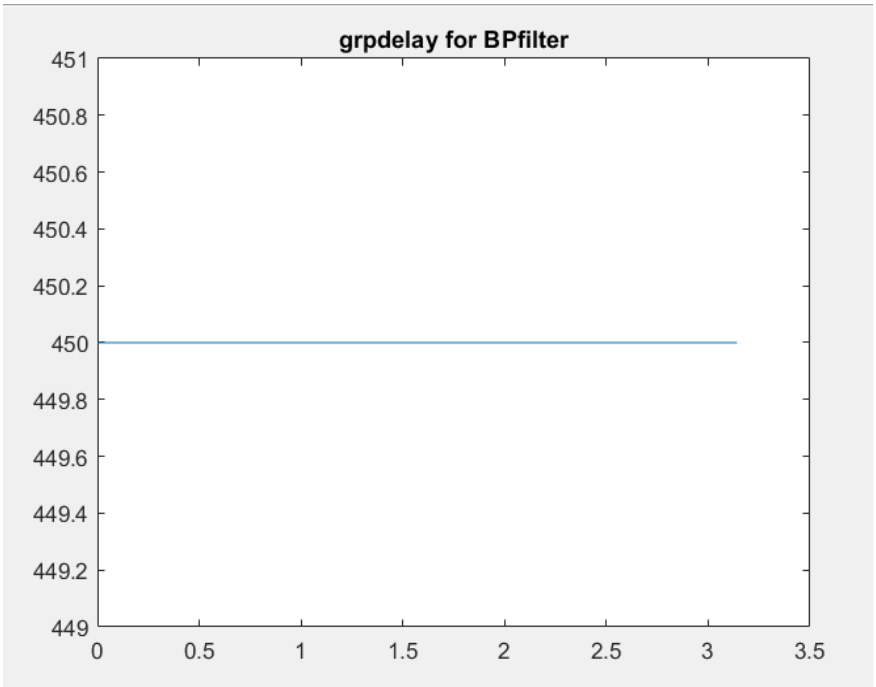
groupdelay خروجی:



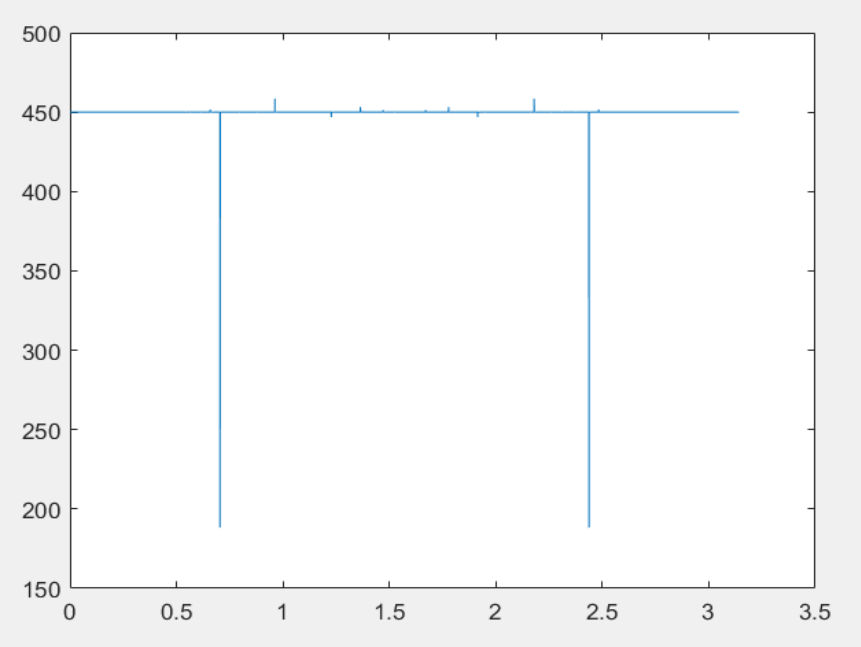
N=1000000 فرض شده است.

BP filter:

grpdelay خروجی:



groupdelay خروجی:



N=1000000 فرض شده است.

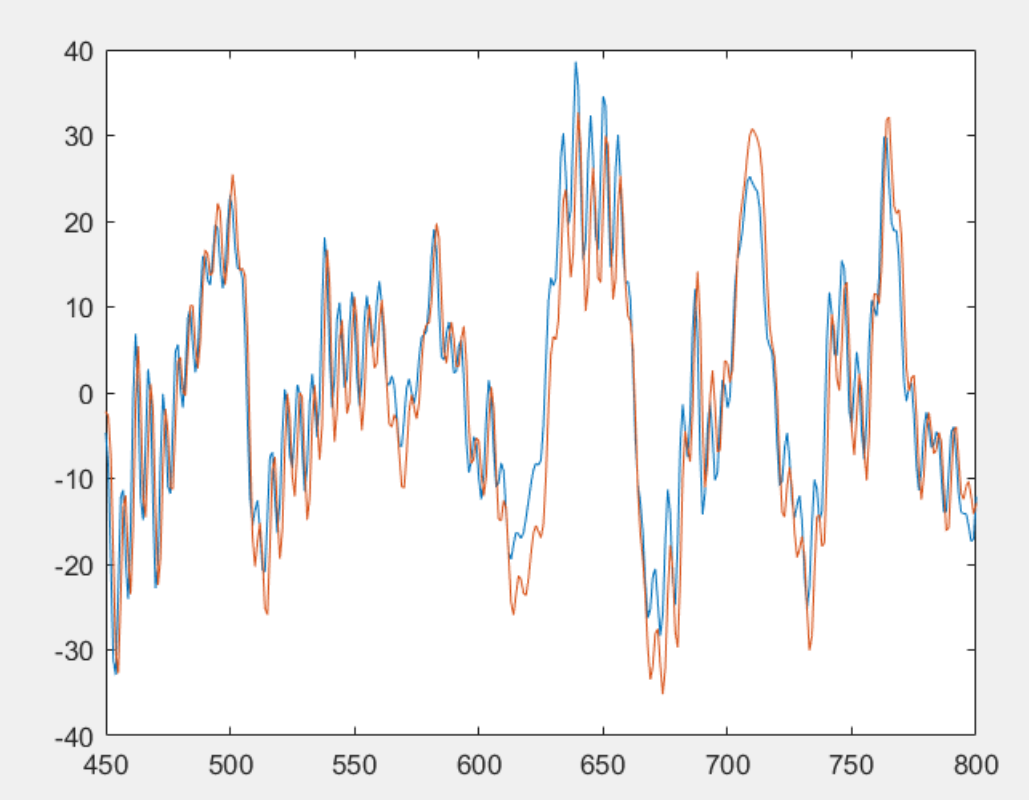
همان طور که مشاهده می کنید در تمامی موارد خروجی تابع groupdelay با متلب یکسان است مگر در تعداد بسیار کمی داده که اختلاف بسیار کمی با مقدار اصلی دارند.

تابع zphasefilter :

این تابع سیگنال فیلتر،سیگنالی که باید فیلتر شود،group delay و فرکانس نمونه برداری را دریافت می کند و بسته به مقدار gd ،سیگنال را تناوبی می کند به گونه ای که پس از شیفت داده های انتهای سیگنال از بین نروند و قابل بازیابی باشند.(خود سیگنال را به انتهایش اضافه می کند.).سپس با استفاده از تابع filter ، سیگنال را فیلتر می کند.از آن جایی که مقدار شیفت مشخص است،سیگنال اصلی را می توان از روی سیگنال فیلترشده که دارای تاخیر است،به دست آورد.

نمودار زیر بخشی از دیتای فیلتر شده ی الکترود 1 در dataset1 است که یک بار با تابع zphasefilter فیلتر شده است و یک بار با تابع filtfilt که خود،تاخیر فاز را حذف می کند.

همان طور که مشاهده می کنید نتایج به جز در نقاط معدودی که دامنه ها اختلاف ناچیزی دارند،با هم یکسان هستند.



آیا فیلتر علی و حقیقی وجود دارد که فاز آن در تمامی فرکانس ها صفر باشد؟

برای این که تبدیل فوریه یک سیگنال حقیقی باشد ، فرم زمانی آن باید حقیقی و زوج باشد. از آن جایی که زوج بودن با علیت متناقض است در نتیجه چنین فیلتری وجود ندارد.

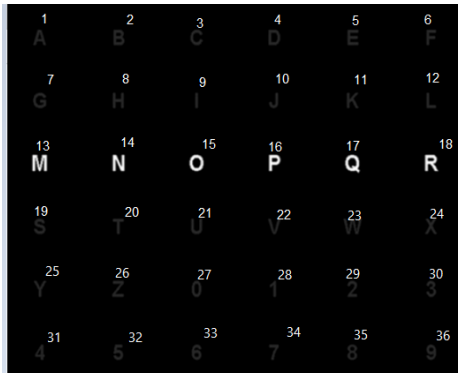
شناسایی کلمات

برای تشخیص نوع پارادایم توجه داریم که برای RC 6 ستون و 6 سطر داریم بنابراین اعداد بین بین 1 تا 12 باید باشند و برای SC نیز 36 پارامتر داریم. بنابراین اگر در سطر ده داده بزرگتر از 12 داشته باشیم سیستم ما پارادایم هست. بدین ترتیب دو دیتاست اول پارادایم SC و بقیه دیتاست ها پارادایم RC دارند. البته در ادامه نیز با توجه به تابع IndexExtraction در صورتی تعداد تارگت ها 75 عدد (15 × 5حرف) باشد پارادایم SC و در در صورتی که تعداد تارگت ها 150 عدد (15 × 10سطر و ستون) باشد پارادایم مربوطه RC می باشد.

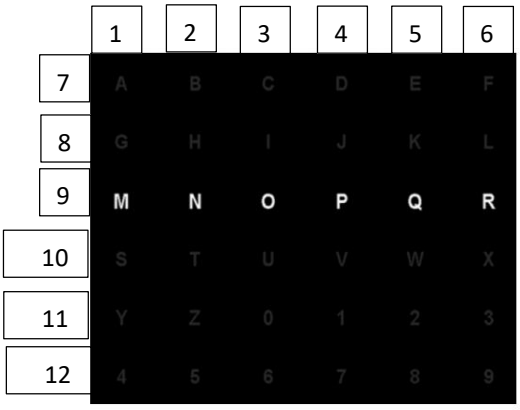
برای پیدا کردن کلمه *LUKAS*

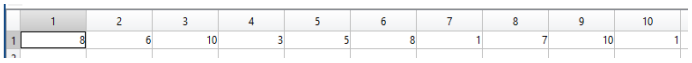
در صورتی که در تابع IndexExtraction که بعد تر توضیح می دهیم تمامی تارگت ها را دریافت کرده وسپس برحسب 15 بار تکرار هر تحریک حروف را جدا کنیم به نتیجه زیر برای پارادایم *SC* می رسیم. پس شماره گذاری با توجه به کلمه *LUKAS* به صورت زیر بوده است:

	1	2	3	4	5
1	12	21	11	1	19
2					



در داده های RC ترتیب ابتدا آمدن سطر یا ستون حرف متفاوت بوده است اما کماکان شماره گذاری سطر و ستون یکسان است: (دیتاست ششم)





برای تعریف تابع IndexExtraction ابتدا چند تابع دیگر را معرفی می کنیم:

```
function [n out]=WhichLetterTarget(txt)
```

این تابع با دریافت داده train ابتدا سطر 11 و 10 را جدا می کند. سپس با پیمایش روی سطر 10 هر جا که سطر 11 صفر باشد داده سطر 10 را NaN می کند. حال برای هر داده سطر ده تعداد تکرار آن بدست می آید و در صورتی که بیش از 4 بار تکرار شده باشد اندیس زمانی آن را در متغیر n ذخیر می کند و همچنین تا 4 تحریک را در سطر ده NaN می کند. و در نهایت اندیس های زمانی و همچنین مقادیر تحریک ها در آن زمان ها را برمی گرداند.

```
function [n out]=WhichLetterNanTarget(txt)
```

این تابع بسیار شبیه به تابع قبل هست وتنها تفاوت آن در این است که در ابتدا هر درایه سطر ده که سطر یازده در آن یک بوده است را NaN می کند و ادامه عملکرد مانند تابع پیشین است.

```
function [n out]=WhichLetterTest(txt)
```

این تابع با دریافت داده Test کار می کند و عملکرد آن نیز بسیار مشابه دو تابع پیشین است با این تفاوت که دیگر target وجود ندارد و در همان ابتدا هیچ داده ای از سطر ده حذف نمی شود.

```
function [subject] =IndexExtraction(txt)
```

حال این تابع با استفاده از سه تابع پیشین اندیس های زمانی target و nan target داده train و همچنین اندیس زمانی همه تحریک های داده test را بدست می آورد و در استراکت subject با نام های targettime ، nanTargettime و testtime ذخیره می کند.

پیاده سازی الگوریتم یادگیری ماشین:

ابتدا مقداری در مورد دو الگوریتم SVM و LDA توضیح می دهیم:

SVM:

الگوریتم SVM یا ماشین بردار پشتیبان (Support vector machine) یکی از الگوریتم ها در حوزه دسته بندی داده ها است.

به مجموعه ای از نقاط در فضای n بعدی داده ها، بردار پشتیبان گفته می شود که مرز بندی دسته ها را نشان داده و دسته بندی و مرزبندی آنها را انجام می دهد و با جابجایی یکی از این دو مورد ممکن است تغییر کند. SVM یا ماشین بردار پشتیبان، با معیار قرار دادن بردار های پشتیبان بهترین دسته بندی و تفکیک بین داده ها را انجام می دهد همچنین در SVM مبنای یادگیری ماشین و ساخت مدل، داده های قرار گرفته شده در بردارهای پشتیبان می باشد. هدف الگوریتم SVM یافتن بهترین مرز در بین داده ها بوده و بیشترین فاصله ممکن از تمام دسته ها را در نظر می گیرد و به سایر نقاط داده ها حساس نمی باشد.

LDA:

تحلیل یا «آنالیز تشخیصی خطی» (Linear Discriminant Analysis – LDA) یک روش آماری برای کاهش ابعاد یک مسئله و تشخیص دسته‌ها بوسیله بیشینه‌سازی نسبت «پراکندگی بین گروه‌ها» (Scatters between groups) به «درون گروه‌ها» (Scatters within groups) است. رویکرد آنالیز تشخیصی خطی در واقع مشابه و وام گرفته از روشی است که «رونالد فیشر» (Ronald Fisher) برای تعیین میزان افتراق بین گروه‌ها به کار برد و مبنایی برای تحلیل واریانس گردید. به همین دلیل گاهی به این تحلیل، «آنالیز افتراقی خطی» نیز می‌گویند.

حال روند کلی تشخیص کلمه را بیان می کنیم:

ابتدا با استفاده از تابع IndexExtraction بردار های target و nan target داده ی train تشخیص داده می شود.

حال تابع detecting داده تولید شده را دریافت می کند. ابتدا با بدست آوردن target و nan target و Sort کردن اندیس زمانی آن ها در ماتریسResult تحریک ها را مرتب کرده و سپس با استفاده از دستور ismember در دو بخش target و nan target ، تشخیص target بودن می دهیم و آن درایه Result را یک می کنیم. بدین ترتیب support vector ساخته می شود.

حال تابع دیگری به نام epochTodata استفاده شده است که با دریافت نام فایل Epoch شده train feacher می سازد. این تابع بدین ترتیب عمل می کند در سطر اول تریال اول همه کانال ها در سطر دوم تریال دوم هم کانال ها را قرار می دهد و به همین ترتیب پیش می رود. حال خروجی این تابع و بردار پشتیبانی که

ساخته ایم را یکی از دو تابع fitcsvm و یا fitcdiscr می دهیم. این دو تابع بدین صورت عمل می کند که با دریافت یک داده نمونه و بردار پشتیبان آن ، یک مدل جهت پیش بینی می سازد و در خروجی برمیگرداند. حالا مدل گرفته شده از آن ها به همراه دیتایی که قرار به پیش بینی آن است (کماکان این داده را نیز با استفاده از تابع epochTodata به ماتریس دو بعدی تبدیل می کنیم.) به تابع predict می دهیم. این تابع با دریافت مدل و همچنین دیتا مورد نظر ، بردار پشتیبان این داده را پیش بینی می کند.

حال مقادیر مربوط به هر اندیس زمانی testرا مرتب کرده و در کنار هم قرار می دهیم و با توجه به بردار پشتیبان بدست آمده target ها را برمیداریم و هر حرفی که 15 بار تکرار شده است به عنوان حرف دریافت شده شناسایی کرده. حال برای متناظر کردن حروف با یک String از تابع زیر استفاده شده است:

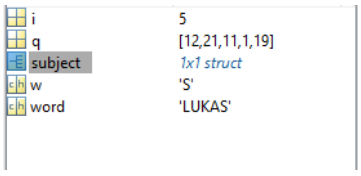
```
function [string]= detectLetter(txt , matrix)
```

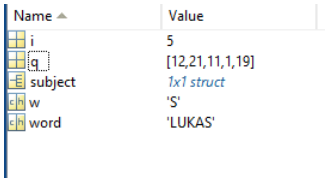
این تابع با دریافت نوع پارادایم و همچنین مقدار مورد نظر طبق الگوریتم بخش قبل حرف را برمیگرداند.

انتظار داریم از آنجایی مدل با همان train ساخته شده است در صورت درخواست خود train نتیجه تابع predict نیز به طور کامل درست باشد. در مدل سازی با استفاده از تابع fitcsvm این اتفاق می افتد ولی در استفاده از تابع fitcdiscr بعضی از حروف تشخیص داده نمی شود که با استفاده از اصلاح تابع هزینه می توان به نتیجه درست رسید. تابع هزینه برای این تابع را بدین صورت تعریف می کنیم:

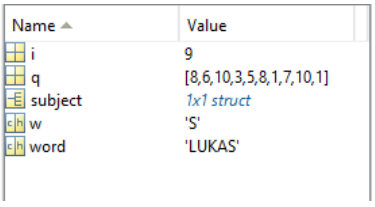
```
cost=[0 0.5;10 0];
```

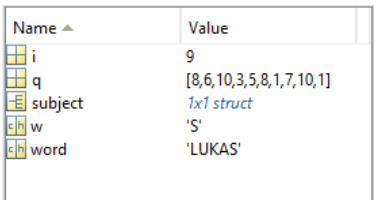
نتیجه با دو الگوریتم برای دیتاست اول:





نتیجه با دو الگوریتم برای دیتاست ششم:





دلیل این امر دو موضوع می تواند باشد :

Imbalanced classification:

این مشکل زمانی پیش می آید که در بردار پشتیبان تعداد یک داده نسبت به داده دیگر بسیار بیشتر است. در موقعیت فعلی نیز تعداد non target ها نسبت به target بسیار بیشتر است بنابراین داده train دارای تعادل نیست و پیش بینی غلط می شود.

راه حل های پیشنهادی:

1. کاهش فرکانس نمونه برداری در برخی مواقع
2. افزایش فرکانس نمونه برداری در برخی مواقع
3. خوشه بندی برحسب over sampling
4. تنظیم دقیق تر تابع هزینه برای دو مدل گفته شده
5. انجام MSMOTE (Modified synthetic minority oversampling technique)

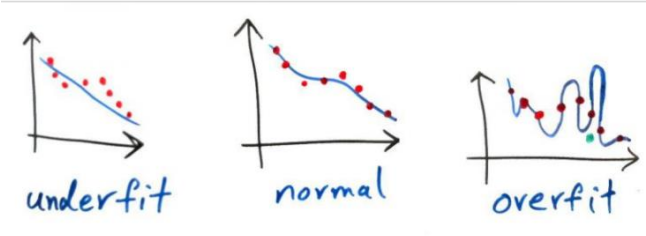
Overfitinng و Underfitting:

اجازه بدهید با یک مثال شروع کنیم. فرض کنید شما برای یک امتحانِ آخرِ ترم در حال درس خواندن هستید. استاد هم به شما ۱۰۰ عدد نمونه سوال داده است تا با استفاده از آن‌ها بتوانید خود را برای امتحان آماده کنید. اگر شما طوری مطالعه کنید که فقط این ۱۰۰ نمونه سوال را کامل بلد باشید و هر سوالِ دیگری که کمی از این ۱۰۰ سوال فاصله داشته باشد، اشتباه جواب دهید، یعنی ذهنِ شما بر روی سوالاتِ آموزشی که استاد برای یادگیری داده است Overfit یا بیش‌برازش شده است. حال اگر تمامی سوالات را به صورت مفهومی بلد باشید ولی هیچ کدام از سوالات را به صورتِ دقیق بلد نباشید، حتی اگر دقیقاً همان سوال‌ها هم در جلسه امتحان به شما داده شود، باز هم نمی‌توانید به درستی و با دقت پاسخ آن ها را بدهید، البته شاید بتوانید یک پاسخ نصفه و نیمه از سوالات بنویسید. اینجا ذهن شما Underfit شده است. این در حالی است که سوالات دیگری که نزدیک به این سوالات هستند را هم شاید بتوانید نصفه و نیمه پاسخ دهید (ولی دقیق نمی‌توانید).

در دنیای الگوریتم‌ها Overfit شدن به معنای این است که الگوریتم فقط داده‌هایی که در مجموعه آموزشی (train set) یاد گرفته است را می‌تواند به درستی پیش‌بینی کند ولی اگر داده‌ای کمی از مجموعه‌ی آموزشی فاصله داشته باشد، الگوریتمی که Overfit شده باشد، نمی‌تواند به درستی پاسخی برای این داده‌های جدید پیدا کند و آن‌ها را با اشتباهِ زیادی طبقه‌بندی می کند.

Underfit شدن نیز زمانی رخ می دهد که الگوریتم یک مدلِ خیلی کلی از مجموعه آموزشی به دست می‌آورد. یعنی حتی اگر خودِ داده‌های مجموعه‌ی آموزشی را نیز به این الگوریتم بدهیم، این الگوریتم خطایی قابل توجه خواهد داشت.

فرض کنید نقطه‌ها در شکل زیر نمونه سوالاتی هستند که استاد برای آمادگی در امتحان همراه با پاسخِ آن‌ها به ما داده است. سوال در محور افقی داده می‌شود و پاسخ در محور عمودی است. به این معنی که به شما X داده می‌شود و شما باید از روی عددِ این X، عددِ Y (عدد روی محور عمودی) را تشخیص دهید. مثلاً اگر مختصاتِ عدد نخست سمت چپدر تصویر زیر [۱ ,۶] باشد، به این معنی است که اگر عدد ۱ را به این الگوریتم بدهیم، الگوریتم عدد ۶ را برگرداند. پس با این حساب اگر به الگوریتم عدد ۱,۱ را دادیم، این الگوریتم (که یادگیری را قبلاً از روی داده‌ها انجام داده است) احتمالاً باید عددی نزدیک به ۶ را برگرداند. این ها در واقع همان مجموعه آموزشی ما هستند:



خط آبیِ موجود، در واقع یادگیریِ مدل طبقه‌بندی است (به صورت دقیق‌تر در این‌جا رگرسیون داریم). همان‌طور که می‌بینید در سمت چپ، خطی که الگوریتمِ طبقه‌بندیْ یادگرفته است از تمامی داده‌ها به مقدار قابل توجهی فاصله دارد. یعنی در این شکل (سمت چپ) underfitting رخ داده است. این در حالی است که در شکل سمت راست، overfitting رخ داده. توجه کنید که در شکلِ سمت راست، اگر یک نقطه جدید (مثلا یک سوال جدید در امتحان) داده شود (نقطه سبز رنگ داده شده) الگوریتم خطای بسیار زیادی دارد. یعنی مقدار Yی که برمیگرداند بسیار با مقدار واقعی فاصله دارد – چون الگوریتم خیلی نتوانسته است که یادگیری را عمومی سازی کند و نسبت به مقادیر جدید خطای بالایی نشان می‌دهد. شکل وسط نیز یک خط معقول و درست برای یک طبقه بند را نشان می دهد که overfit یا underfit نشده است.

این دو خطای معروف در حوزه طبقه‌بندی هستند که الگوریتم های مختلف طبقه بندی باید از آن ها اجتناب کنیم.

راه حل:

1. یکی از بهترین راه حل های آن تنظیم تابع هزینه است.

تابع هزینه:

بیشتر الگوریتم‌ها در «یادگیری ماشین» (Machine learning)، برمبنای کمینه یا بیشینه‌سازی «تابع هدف» (Objective Function)، عمل می‌کنند. گروهی از توابع هدف که قرار است کمینه شوند به نام «توابع زیان» (Loss Function) معروفند. البته به این توابع در مباحث هوش مصنوعی گاهی «توابع هزینه» (Cost Function) نیز می‌گویند.

تابع زیان، معیاری برای سنجش مناسب بودن مدل از نظر قابلیت و توانایی در پیشگویی مقدارهای جدید است. یکی از روش‌های معمول برای پیدا کردن کمینه تابع زیان، استفاده از مشتق و الگوریتم «گرادیان کاهشی» (Gradient Descent) است. زیرا در بیشتر مسائل مربوط به هوش مصنوعی، توابع زیان به صورت «محدب» (Convex) هستند. متاسفانه نمی‌توان برای همه نوع داده‌ای از یک تابع زیان استفاده کرد. انتخاب تابع زیان مناسب به عوامل متعددی نظیر وجود نقاط یا داده‌های پرت، نوع الگوریتم یادگیری ماشین، هزینه زمانی اجرای الگوریتم و سادگی محاسبه مشتق و ... بستگی دارد.

به طور کلی می‌توان توابع زیان در زمینه یادگیری ماشین را به دو گروه عمده تقسیم کرد::

توابع زیان مربوط به الگوریتم‌های دسته‌بندی (Classification)

توابع زیان مربوط به رگرسیون (Regression)

روش های بدست آمدن آن بدین ترتیب:

1- تابع زیان میانگین مربعات (Means Square Error)
2- میانگین قدرمطلق خطا (Mean Absolute Error)
3- تابع زیان لگاریتم کسینوس هذلولوی (Log-Cosh)
4- تابع زیان چندکی (Quantile Loss)

می توانیم در موقعیت فعلی با چنین سینتکسی cost مورد نظر خود در مدل ساختن را به دو تابع fitcsvm و fitcdiscr اعمال کنیم:

```
m1=fitcdiscr(data,result,'Cost',cost);
```

در fitcsvm و fitcdiscr متغیری چیزی وجود دارد به نام cost که یک ماتریس ۲ در ۲ است.

درایه اول : هزینه تشخیص درست target

درایه دوم : هزینه اینکه یه target رو non target تشخیص بدهد

درایه سوم : هزینه اینکه non target رو non target تشخیص بدهد

و درایه چهارم : هزینه اینکه یه non target رو target تشخیص بدهد

در مشکل imbalanced classification ، باید هزینه درایه دوم که تشخیص target به عنوان nan target است را بیشتر کنیم و در مشکل overfittng باید هزینه درایه چهارم را بیشتر کنیم.

حال تغییرات تابع انتهایی جهت تشخیص دقیق کلمه و کاهش imbalance classification را معرفی می کنیم:

```
function [q]=detecting(a,txt,txt2,nameOfway,file,th,num,cost)
```

ورودی اول همان خروجی تابع IndexExtraction است و جهت داشتن target و nan target های داده ی train و همچینین مقادیر تحریک ها در داده test دریافت می شود.

ورودی دوم اسم فایل داده ای که قرار بر این است که کلمه تشخیص داده شود و ورودی سوم اسم فایلی است که train feature از روی آن ساخته می شود.

ورودی چهارم اسم روشی است که مدل قرار است بر مبنای آن تشخیص داده شود (lda و یا svm)

ورودی پنجم نشان دهنده است که پردازش روی داده train است و یا test.

ورودی ششم نشان می دهد اگر تا جه حد یک حرف تکرار شود مبنی بر این است حرف واقعا دیده شده است.

ورودی هفتم مبنی بر این است که ستون nan target دو در نظر گرفته شده است و یا صفر. این کار به علت آن انجام می شود که با عدد2 مدل سازی دقیق تری انجام می شود!

می دانیم مشکل اصلی که وجود دارد این است که تعداد target ها کم می باشد. بنابراین همان ستونی که مربوط به target است را 5 بار تکرار می کنیم و این کار باعث می شود داده ای با تعادل بیشتر داشته باشیم.

همچنین متلب تابع آماده ای به نام tabulate وجود دارد که توزیع داده های train feature بیان می کنیم.

خروجی دیتاست دوم:

```
  Value    Count   Percent
      1      825    23.91%
      2     2625    76.09%

word =

    'LUKAS'
```

برای خروجی RC ، متاسفانه خروجی کاملا درست و به ترتیب نگرفتیم ☹

ولی بدین صورت برای دیتاست سوم و ششم بدین صورت است:

دیتاست سوم:

	1	2	3	4	5	6	7	8	9	10
1	10	11	7	6	12	8	9	4	5	1
2										

دیتاست ششم:

	1	2	3	4	5	6	7	8	9	10
1	3	9	5	11	4	7	10	8	2	6
2										

Value	Count	Percent
1	1650	68.75%
2	750	31.25%

دیتاست نهم:

1x12 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8	10	1	7	5	2	9	6	3	4	7	10	
2													

```
  Value    Count   Percent
      1     1650     68.75%
      2      750     31.25%

word =

    'AR'
```

راه های دیگری است که موجب بهتر شدن کار ممکن است بشود:

Under Sampling:

از این روش استفاده شد و تعدادی از nan target ها حذف اما متاسفانه نتیجه عکس دارد.

Over Sampling:

از این روش استفاده شده است و برای داده SC به طور مناسبی پاسخ گرفته ایم ولی برای داده RC کافی نیست.

Generate Data:

این کار با استفاده از الگوریتم smot و تابع SMOT پیاده سازی شده است ولی اثر چندانی بر داده ها نداشته است.

همچنین دو تابع detectRC(q) و detectSC(q) بر اساسا الگوریتم SC و RC (برای این پارادایم هر دو عدد که پشت سر هم هستند عدد بزرگتر بیانگر سطر و عدد کوچکتر بیانگر ستون است) و با کمک detectLetter رشته کلمه را بر میگرداند.