باسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی برق

درس سیگنال‌ها و سیستم‌ها

گزارش تمرین سری سه

رابط مغز ـ رایانه مبتنی بر سیگنال EEG برای تشخیص حرکت و تصور حرکت

امیرحسین افشارراد

۹۵۱۰۱۰۷۷

بهراد منیــری

۹۵۱۰۹۵۶۴

استاد

دکتر حمید کربلایی آقاجان

بهار ۱۳۹۷

بخش صفرم ـ توضیحات اولیه

۱ – فایل‌های دیتاست به دلیل حجم بالایشان ضمیمهٔ گزارش نشده‌اند. فایل های mat. مربوط به سابجکت‌های مختلف باید همگی در فولدر dataset قرار بگیرند.

۲ ـ برای رسیدن به نتایج بهتر، در بخش پیش‌پردازش از فیلترهایی با مرتبهٔ بالا استفاده شده که به همین دلیل زمان پیش‌پردازش داده‌ها بسیار بالا و نزدیک به یک و نیم ساعت است. در صورت لزوم می‌توانیم داده‌های پیش‌پردازش شده را در اختیارتان قرار دهیم.

۳ـ در تمام مراحل این گزارش از داده‌های سابجکت سوم استفاده شده است ولی پردازش‌ها و نتایج برای تمام سابجکت‌ها ضمیمه شده. در گزارش تنها زمانی به دیگر سابجکت‌ها اشاره شده که نتایج آنها تفاوت معناداری با نتایج سابجکت سوم داشته باشد.

۴ـ به دلیل زمان بالای پردازش پیشنهاد می‌کنیم از اجرای برنامه خودداری کنید. تمام دیتای تولید شده را در صورت لزوم می‌توانیم در اختیارتان قرار دهیم.

۵ـ در تمام بخش‌های این تمرین داده‌های حرکت و تصور حرکت پردازش شده‌اند.

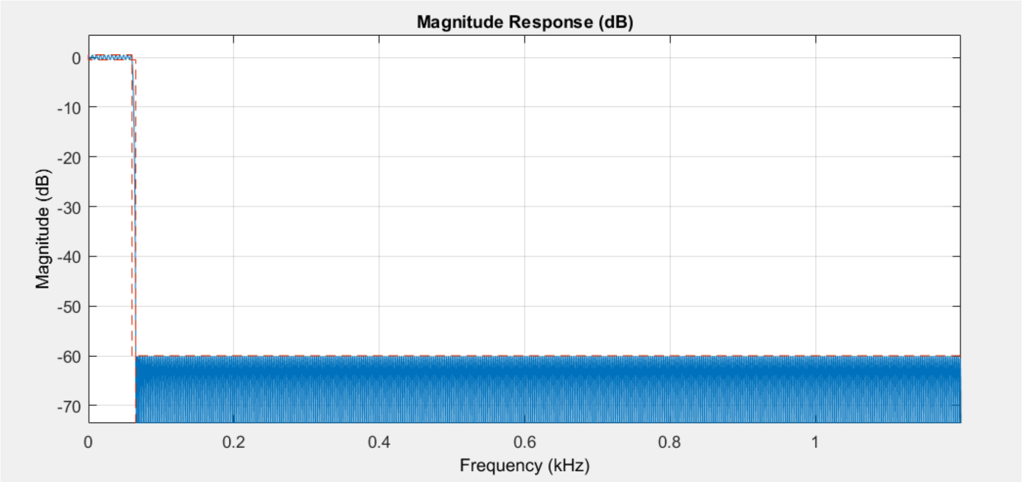
بخش اول – پیش‌پردازش

قبل از شروع پردازش اصلی، در این بخش تلاش می‌کنیم تا حد امکان نویز را کاهش دهیم. می‌دانیم که در فرکانس‌های بالای سیگنال EEG اطلاعات مفید وجود ندارد بنابراین با کمک یک فیلتر پایین‌گذر با فرکانس ۶۰ هرتز، سیگنال را فیلتر می‌کنیم.

برای این کار از فیلتری Equiripple با اطلاعات زیر استفاده می‌کنیم.

```
Fpass = 60;     % Passband Frequency
Fstop = 65;     % Stopband Frequency
Apass = 1;      % Passband Ripple (dB)
Astop = 60;     % Stopband Attenuation (dB)
Fs    = 2400;   % Sampling Frequency
```

شکل ۱، دامنه پاسخ فرکانسی این فیلتر است. دیتای تمام الکترودهای تمام فعالیت‌های برای هر ۲۰ آزمایش انجام‌شده را با این فیلتر، فیلتر می‌کنیم.



شکل ۱

سیگنال حاصل پهنای باندی محدود داشته پس در شرایط قضیهٔ نایکویست صدق می‌کند. بنابر این قضیه، نمونه‌برداری سیگنالمان با فرکانسی برابر نصف بیشینه فرکانس سیگنال، باعث از بین رفتن هیچ اطلاعاتی نمی‌شود و سیگنال اصلی به طور کامل قابل بازیابی است پس می‌توانیم فرکانس نمونه برداری خود را تا ۱۲۰ هرتز کاهش دهیم.

سیگنال را از سیستم زیر می‌گذارنیم

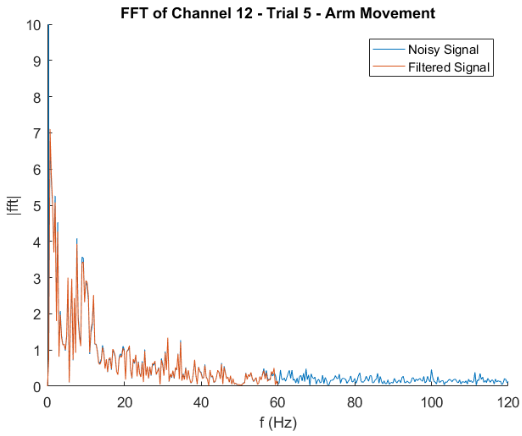
$$y[n] = x[20n]$$

این سیستم از هر بیست نقطهٔ سیگنال اصلی تنها یک نقطه نگه‌داشته و ما را به فرکانس نمونه‌برداری ۱۲۰ هرتز می‌رساند.

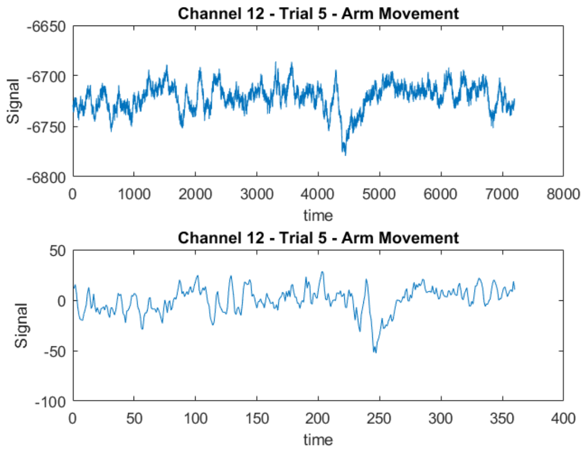
با پایین آوردن فرکانس‌نمونه‌برداری حجم دیتای خود را بسیار کم کردیم درحالی که قضیهٔ نایکوسیت تضمین می‌کند که اطلاعاتی را از دست نداده‌ایم.

در آخر فرکانس زیر نیم هرتز داده‌ها را حذف می‌کنیم زیرا در این فرکانس‌ها نیز اطلاعات مفیدی وجود ندارد.

تمام مراحل بالا را برای داده‌های حرکت و تصور حرکت انجام داده و نتایج را در gstruct به نام Data ذخیره می‌کند.



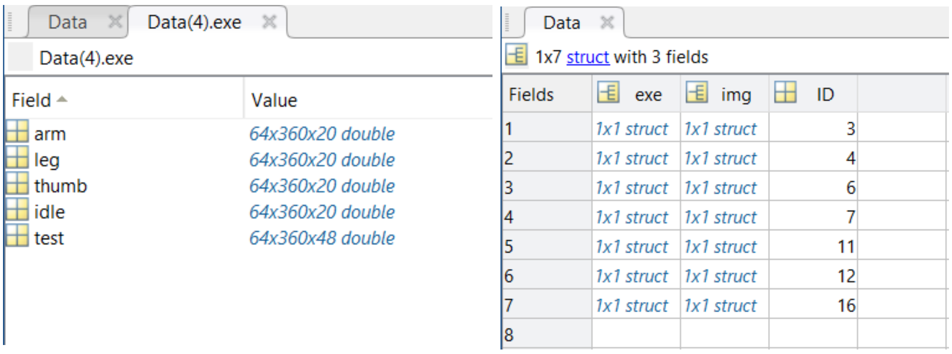
شکل ۲



شکل ۳

به عنوان نمونه، شکل یک و دو نمونهٔ یک سیگنال خاص قبل و بعد از پیش‌پردازش را در حوزهٔ زمان و فرکانس نمایش می‌دهند.

شکل ۴ نمایش‌دهندهٔ داده ساختارِ مورد بحث است.


Data | Data(4).exe

Data(4).exe

Field	Value
arm	64x360x20 double
leg	64x360x20 double
thumb	64x360x20 double
idle	64x360x20 double
test	64x360x48 double

Data

1x7 struct with 3 fields

Fields	exe	img	ID
1	1x1 struct	1x1 struct	3
2	1x1 struct	1x1 struct	4
3	1x1 struct	1x1 struct	6
4	1x1 struct	1x1 struct	7
5	1x1 struct	1x1 struct	11
6	1x1 struct	1x1 struct	12
7	1x1 struct	1x1 struct	16
8			

شکل ٤

بخش دوم – استخراج ویژگی‌ها

در این بخش تلاش می‌کنیم ویژگی‌هایی از سیگنال EEG که احتمال‌می‌دهیم برای طبقه‌بندی دیتا مفید باشد را به دست می‌آوریم. ما برای این بخش از تمام ویژگی‌های ذکر شده در صورت تمرین و چندین ویژگی دیگر که به نظرمان مناسب بود استفاده کردیم. استفاده از ویژگی های حوزهٔ زمان سیگنال در پژوهش‌های این حوزه، به دلیل نویز زیاد سیگنال‌های EEG بسیار نامرسوم است اما ما در این بخش تنها به استخراج ویژگی می‌پردازیم و تصمیم خود برای استفاده یا عدم استفاده از این ویژگی ها در Classifier خود را به بخش بعد موکول می‌کنیم.

تمام پردازش‌ها را بر روی داده‌های حرکت و تصور حرکت انجام داده و نتایج را در struct‌ی به نام FData ذخیره می‌کنیم.

۱. واریانس داده‌ها

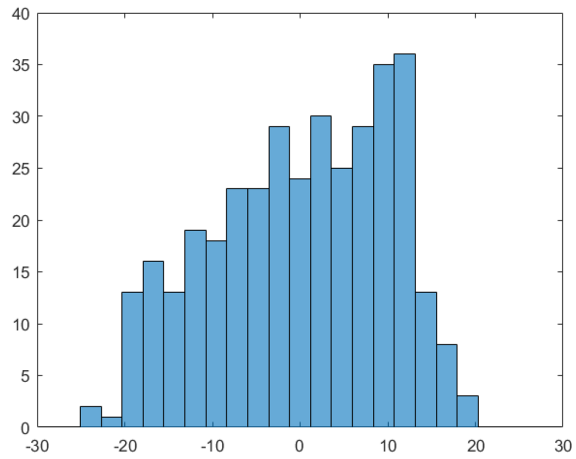
در هر Trial، از ۶۴ الکترود داده گرفته‌ایم. واریانس داده‌های هر الکترود را به عنوان ویژگی محاسبه می‌کنیم یعنی ۶۴ ویژگی از این جنس داریم.

$$\sigma^2_{Channel} = \sum_n (X_{channel}[n] - mean[\, X_{Channel}])^2$$

۲. هیستوگرام داده‌ها

باید تمام هیستوگرام‌ها را با Bin Edge های یکسان رسم کنیم بنابراین ابتدا میانگین و واریانس سیگنال‌های تمام کانال ها در تمام Trial های تمام فعالیت‌ها به دست می‌آوریم و هیستوگرام را با ۲۵ Bin در بازهٔ $Mean \pm 2\, STD$ رسم می‌کنیم.

هیستوگرام مربوط به حرکت دست، کانال ۱۱ و Trial ۱۲ در شکل زیر آمده است.



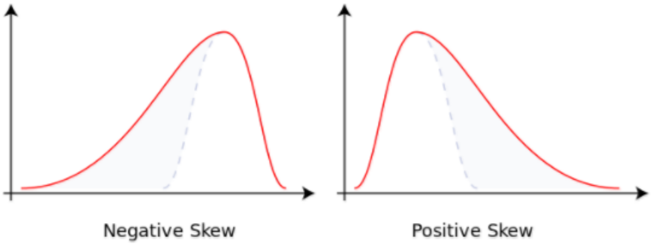
شکل ۵

مقدار هیستوگرام داده‌های هر کانال در هر Bin را یک ویژگی می‌گیریم پس در کل ۶۴*۲۵ ویژگی داریم.

۳. Skewness سیگنال‌های هر کانال

Skewness یا چولیدگی یک ویژگی آماری است که نشان‌گر عدم تقارن تابع چگالی احتمال یک متغیر تصادفی است. این ویژگی برای متغییر نرمال صفر است و به تعریف می‌شود $Skew = \frac{\mu_3}{\sigma^3}$.

که در آن μ_3 گشتاور مرکزی مرتبهٔ سوم سیگنال یعنی $\mu_3 = \sum_n (X_{channel}[n] - mean[X_{Channel}])^3$ است.



شکل ٦

مقدار Skewness هر کانال را یک ویژگی در نظر می‌گیریم یعنی ۶۴ ویژگی جدید از این جنس داریم.

۴. Form Factor

فرم فاکتور یک سیگنال برابر مقدار RMS آن سیگنال تقسیم بر میانگین قدرمطلق آن سیگنال است

$$k_f = \frac{\text{RMS}}{\text{ARV}} = \frac{\sqrt{\frac{1}{T}\int_{t_0}^{t_0+T}[x(t)]^2\,dt}}{\frac{1}{T}\int_{t_0}^{t_0+T}|x(t)|\,dt} = \frac{\sqrt{T\int_{t_0}^{t_0+T}[x(t)]^2\,dt}}{\int_{t_0}^{t_0+T}|x(t)|\,dt}$$

مقدار Form Factor هر کانال را یک ویژگی در نظر می‌گیریم یعنی ۶۴ ویژگی جدید از این جنس داریم.

۵. فرکانس مد

فرکانسی که بیشترین محتوا را دارد، به بیان دیگر، ماکزیمم نمودار حاصل از رسم تبدیل فوریه گسسته سیگنال. همچنین برای محاسبه‌ی این فرکانس بر حسب هرتز، mapping مورد نیاز را نیز انجام داده‌ایم.

۶. فرکانس میانگین

با استفاده از تابع meanfreq محاسبه شده است.

۷. فرکانس میانه

با استفاده از تابع medfreq محاسبه شده است.

۸. ضرایب تبدیل گسسته سینوسی داده‌های هر کانال (DST)

با استفاده از تابع dst و بر اساس فرمول زیر محاسبه شده می‌شود:

$$y(k) = \sum_{n=1}^{N} x(n)\sin\left(\pi \frac{kn}{N+1}\right), \quad k = 1, \ldots, N.$$

۹. ضرایب تبدیل گسسته کسینوسی داده‌های هر کانال (DCT)

با استفاده از تابع dct و بر اساس فرمول زیر محاسبه شده می‌شود:

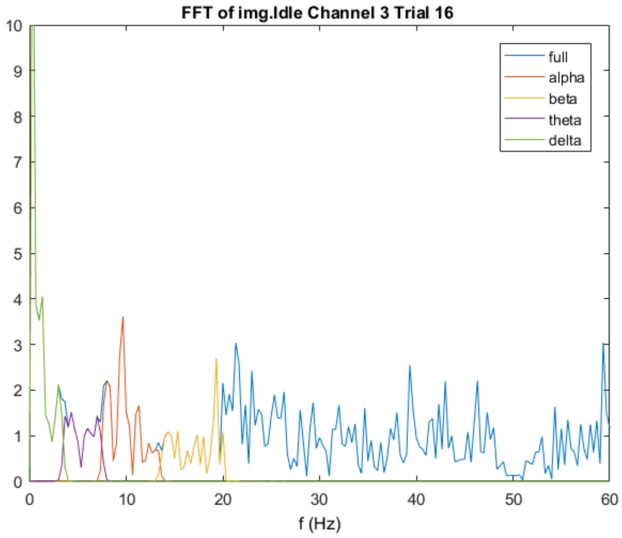
$$y(k) = w(k)\sum_{n=1}^{N} x(n)\cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right), \quad k = 1, 2, \ldots, N,$$

where

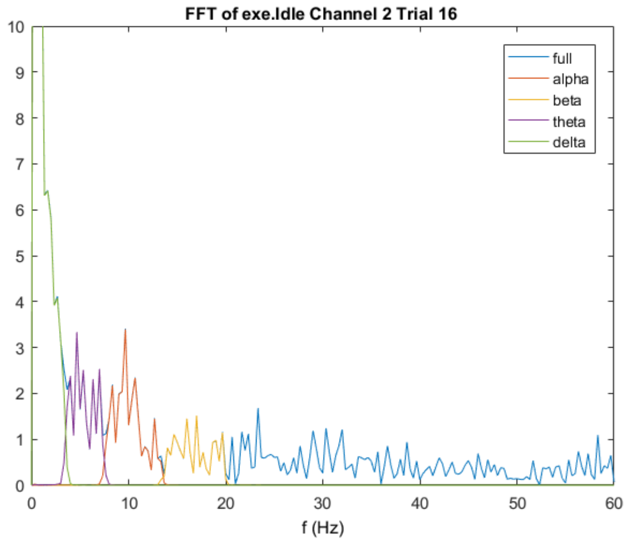
$$w(k) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 1, \\ \sqrt{\frac{2}{N}}, & 2 \le k \le N, \end{cases}$$

۱۰. انرژی سیگنال در ۴ باند فرکانسی

انرژی موجود در ۴ باند فرکانسی متعارف را (با تعاریف ارائه شده در صورت تمرین) محاسبه کردیم. برای این کار، ابتدا سیگنال‌ها را فیلتر کردیم و باندهای فرکانسی را به تفکیک به دست آوردیم، سپس انرژی موجود در هر کدام از سیگنال‌های فیلترشده را محاسبه کردیم. همچنین، برای اطمینان از صحت عمل جداسازی باندهای فرکانسی، به صورت تصادفی، نمونه‌هایی از سیگنال‌های اصلی و فیلترشده را در حوزه فرکانس رسم کرده‌ایم.



شکل ۷

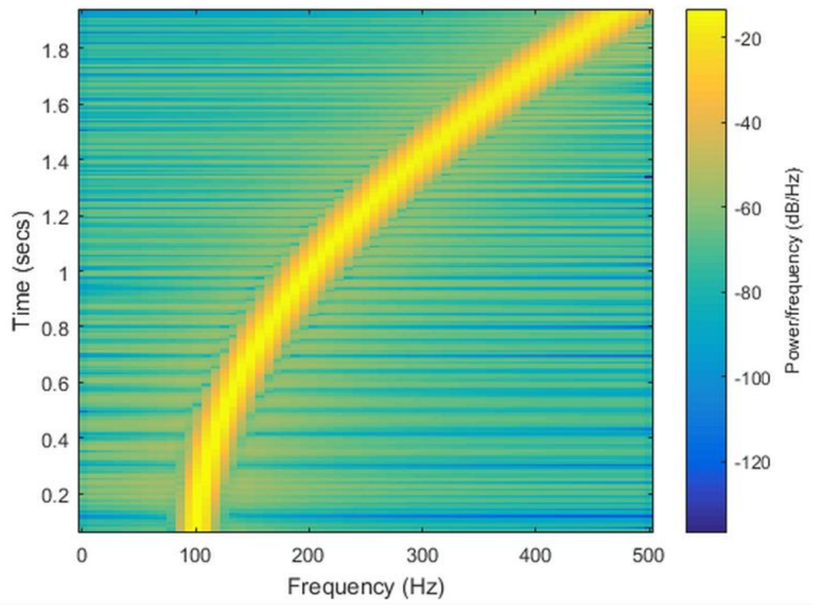


شکل ۸

تا به این جا، ویژگی‌هایی که در صورت تمرین ذکر شده بودند را محاسبه کردیم و در استراکت FData ذخیره کردیم. در ادامه، چند ویژگی دیگر را نیز اضافه نموده‌ایم. این ویژگی‌ها، حاصل جستجوی ما از افراد باتجربه و بررسی مقالات ارائه‌دشه و نتایج مسابقات برگزارشده در زمینه BCI است.

۱۱. تبدیل فوریه زمان کوتاه (Short-Time Fourier Transform - STFT)

تبدیل فوریه زمان کوتاه، ابزاری است برای **برقراری ارتباط بین حوزه زمان و فرکانس** که در آن، محتوای فرکانسی را در طول زمان بررسی می کنیم. در واقع، این تبدیل، سیگنال را به پنجره‌های زمانی کوچک تقسیم می کند و در هر کدام از این پنجره‌های زمانی، به محاسبه تبدیل فوریه می‌پردازد. بنابراین، می‌توان تغییرات انرژی در فرکانس‌های مختلف را در طول زمان بررسی کرد. شکل زیر، نمونه‌ای از این تبدیل است که به شکل نمودار ۲ بعدی زمانی ـ فرکانسی رسم شده است.



شکل ۹

برای محاسبه‌ی STFT در متلب، از تابع spectrogram استفاده می‌شود. در این تابع، **پنجره زمانی** و **میزان همپوشانی پنجره‌های زمانی**، و نیز **تعداد نقاط مورد استفاده در محاسبه fft** را مشخص می‌کنیم.

با استفاده از این تابع، تبدیل فوریه زمان کوتاه را برای سیگنال‌های خود محاسبه کردیم و در استراکت FData ذخیره نمودیم. ضمنا به دلیل این که توابع مربوط به طبقه‌بندی موجود در متلب (svm) قادر به پردازش داده‌های مختلط نبودند، (بدیهی است STFT مقادیری مختلط دارد) و نیز به دلیل این که مقدار انرژی موجود در فرکانس‌های مختلف برای ما مورد توجه است، **اندازه** تبدیل فوریه زمان کوتاه را به عنوان ویژگی در نظر گرفتیم.

۱۲. Common Spatial Patterns

یکی از ویژگی‌های معروف و مورد استفاده برای طبقه‌بندی داده‌های تصور حرکت، استفاده از Common Spatial Patterns یا CSP است.

ایدهٔ اصلی این روش بدین شکل است.

فرض کنید $X_k[n]$ ماتریسی باشد که سطر iام آن داده‌های الکترود iام برای یک دادهٔ دستهٔ k باشد که k یکی از دستهٔ های حرکت بازو، حرکت شست، حرکت پا و استراحت است. قصد داریم ضرایبی بیابیم که اگر داده‌های الکترودهای مختلف هر فعالیت را با آن ضرایب ترکیب خطی کنیم و در نتیجه یک سیگنال برای هر فعالیت به دست آوریم، فیلتر را طوری پیدا می‌کنیم که واریانس سیگنال حاصل برای فعالیت‌های مختلف بیشترین تفاوت واریانس را داشته باشند. (یا در برخی الگوریتم‌ها نسبت واریانس‌های فعالیت‌های مختلف بیشینه شود). یعنی باید

$$w = argmax_w \frac{||wX_k||_2^2}{||wX_2||_2^2}$$

را محاسبه کنیم.

ما در این تمرین از الگوریتم معرفی شده در مقالهٔ زیر استفاده می‌کنیم.

Moritz Grosse-Wentrup, Martin Buss. *"Multiclass Common Spatial Patterns and Information Theoretic Feature Extraction"*, **IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING**, VOL. 55, NO. 8, 2008.

کد الگوریتم آنها از لینک زیر قابل دریافت است. روش آنها مبتنی بر Eigenvector Decomposition ماتریس های کوواریانس داده های گروه‌های مختلف است.

http://mlin.kyb.tuebingen.mpg.de/BCIWebpage/Code.html

مقالهٔ زیر کاری مشابه کار ما ولی برای جداسازی تصور تنها دو نوع حرکت انجام داده است.

Yijun Wang, Shangkai Gao, Xiaorong Gao. "*Common Spatial Pattern Method for Channel Selection in Motor Imagery Based Brain-computer Interface*", **Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference**, 2005.

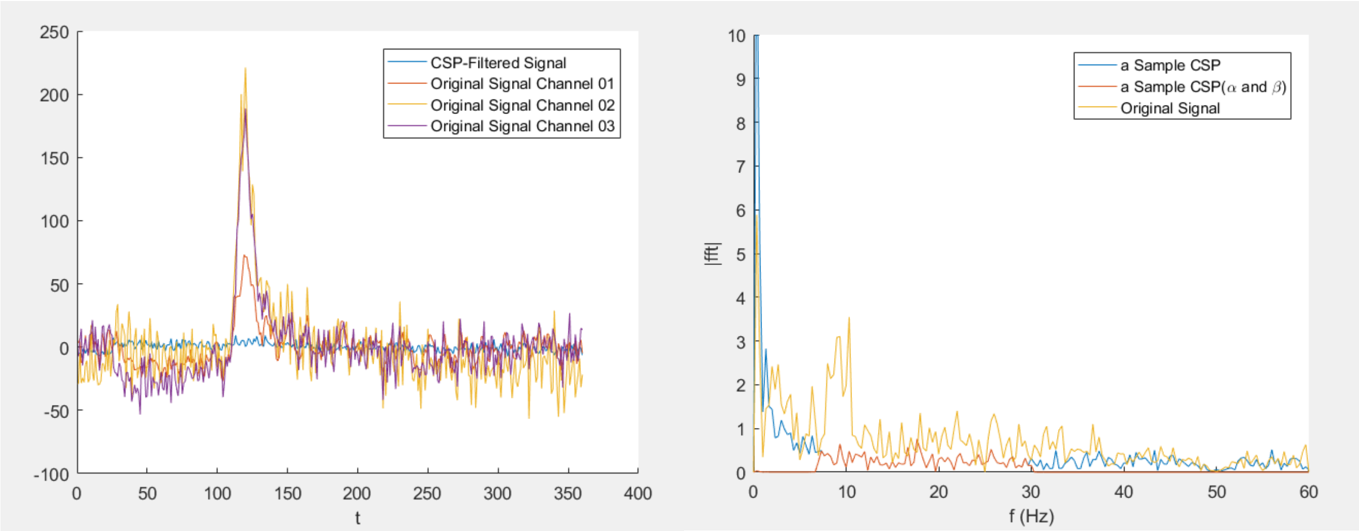
نویسندگان این مقاله توانسته‌اند با استفاده از تنها این روش **با دقتی در حدود ۹۴ درصد** این حرکت‌ها را از هم جدا کند.

به کمک الگوریتم مذکور دو مجموعه ضریب بهینه پیدا می‌کنیم. برای پیدا کردن مجموعه ضرایب (فیلتر) بهینه با توجه به توصیهٔ مقالهٔ اول از سیگنال فیلتر شده در باندهای آلفا و بتا استفاده می‌کنیم. با محاسبهٔ ترکیب خطی داده‌های الکترودهای مختلف هر کانال به دو سیگنال به ازای هر فعالیت و Trial میرسیم. این سیگنال ها را CSP1 و CSP2 می‌نامیم.

از این سیگنال‌ها ویژگی‌های زیر را بر می‌گزینیم.

- خود سیگنال CSP1 و CSP2 به عبارتی ۳۶۰*۲ ویژگی
- فرکانس میانگین هر سیگنال یعنی دو ویژگی
- فرکانس میانهٔ هر سیگنال یعنی دو ویژگی

برای شهود بیشتر، به دو تصویر زیر توجه کنید



شکل ۱۰

تصویر سمت راست نمودار تبدیل فوریهٔ سیگنال اصلی یکی از کانال‌ها و نمودار تبدیل فوریه CSP1 و نمودار تبدیل فوریهٔ باند آلفای CSP1 است.

تصویر سمت چپ نمودار زمانی سیگنال چهار کانال پیشانی و سیگنال CSP1 است.

ویژگی های مربوط را در استراکت FData ذخیره کردیم. این کار را برای تمام داده‌های حرکت و تصور حرکت انجام دادیم.

۱۳. سیگنال کانال‌های مختلف

در آخر با مشورت با چند نفر از افرادی که در حوزه ساخت BCI کار کرده‌اند، تصمیم گرفتیم سیگنال تمام الکترود ها هم به عنوان ویژگی بپذیریم. به این موضوع که سیگنال EEG نویز زیادی دارد که لزوما فرکانس بالا نیست و یا با میانگین گیری قابل رفع شدن نیست و احتمال اینکه این نویز باعث شود این ویژگی توانایی جدا کردن کلاس‌های مختلف را نداشته باشد واقفیم اما این ویژگی را انتخاب می‌کنیم تا در بخش‌های بعد اگر JValue مناسبی داشتند در پردازش نهایی لحاظ شوند.

به طور کلی ۶۴*۳۶۰ ویژگی جدید از این جنس داریم.

بخش سوم ــ انتخاب ویژگی‌های مؤثر

۳ــ۱) J-valueها، ماتریس پخشی و تابع Jvalue

در این بخش، با توجه به فرمول ارائه شده برای J-value، ماتریس شامل این مقادیر را تشکیل می‌دهیم. برای این منظور، تابعی به نام Jvalue نوشته‌ایم که ماتریسی $N \times M$ را تشکیل می‌دهد که در آن، N تعداد ویژگی‌های استخراج شده است و اگر k کلاس متفاوت داشته باشیم، بین هر دو کلاس، به ازای هر ویژگی، یک J-value خواهیم داشت که مجموعا برای هر ویژگی، $M = \binom{k}{2}$ عدد J-value بین کلاس‌ها به صورت دوبه‌دو خواهیم داشت.

هدر تابع Jvalue به شکل زیر است:

```
function [J, J_Index, Feature] = Jvalue(X,Y)
```

همان طور که مشاهده می‌شود، تابع Jvalue علاوه بر خروجی ماتریس J که توضیح داده شد، دو خروجی دیگر نیز دارد. خروجی J_Index، ماتریسی است که به تعداد ویژگی‌ها سطر دارد و متناظر با ماتریس J، اعلام می‌کند که هر کدام از سطرهای ماتریس J حاوی کدام ویژگی است. در واقع ماتریس J_Index، حاوی نام ویژگی‌هایی است که مقادیر J-value آن‌ها در ماتریس J ذخیره شده است. علت وجود این خروجی در تابع، آن است که در ادامه و پس از انتخاب ویژگی‌های مؤثر از ماتریس J، بتوانیم نشان دهیم که هر کدام از این ویژگی‌های موثر در اصل، چه چیز بوده اند.

یک نکته در مورد محاسبه‌ی J-valueها در این تابع وجود دارد و آن، وجود عبارت eps در مخرج کسر مربوط به محاسبه J-valueها است که علت قرار دادن آن، برای شرایط خاصی است که واریانس‌ها صفر باشند (ویژگی دقیقاً در کلاس‌های مختلف یکسان باشد) که در این صورت، اگر از فرمول اصلی استفاده کنیم، خروجی‌ها به صورت NaN خواهند بود. حضور eps در مخرج، هیچ خللی در محاسبات وارد نمی‌کند و در این حالت خاص نیز، باعث می‌شود که خروجی‌ها به جای NaN به مقدار معنادار خود، یعنی صفر، مبدل شوند.

خروجی سوم این تابع، ماتریس Feature است که تمامی ویژگی‌ها را در قالب ماتریسی ارائه می‌کند که این ماتریس، آماده‌ی وارد شدن به توابع مربوط به SVM است. بدیهی است بعد از انتخاب ویژگی‌های موثر، باید تنها ستون‌های متناظر با این ویژگی‌ها را از ماتریس Feature انتخاب کرد و به classifier ارائه کرد و نه تمامی محتوای ماتریس Feature را. این کاربردها در ادامه انجام شده و مورد توضیح واقع شده‌اند.

۳ــ۲) انتخاب ویژگی‌ها و تابع Feature_Selector

برای انتخاب ویژگی‌های مؤثر با استفاده از ماتریس J، تابعی به نام Feature_Selector نوشتیم که هدر آن به صورت زیر است:

```
function [TrainSet, labels, feature_list, TestSet] =
Feature_Selector(FData, exe_or_img, number_of_features)
```

ورودی‌های این تابع، FData، استراکت حاوی تمامی ویژگی‌ها، exe_or_img، رشته‌ای که یکی از دو حالت ‘exe’ یا ‘img’ را دارد و مشخص می‌کند که ویژگی‌ها برای حرکت یا تصور حرکت بررسی شوند، و number_of_features، تعداد ویژگی‌هایی که می‌خواهیم در نهایت انتخاب کنیم، می‌باشند.

تابع در نهایت چهار خروجی می‌دهد که TrainSet، ماتریسی است که حاوی ویژگی‌های انتخابی است و آماده است که وارد توابع مربوط به SVM شود. labels برداری است که برچسب‌های متناظر با TrainSet را در بر دارد و یکی دیگر از ورودی‌های توابع SVM

است، feature_list که نام ویژگی‌های انتخابی را در بر دارد تا بتوانیم در بررسی‌های بعدی، ببینیم کدام ویژگی‌ها کارامدتر بوده‌اند و جزء ویژگی‌های انتخابی نهایی قرار گرفته‌اند. لازم به ذکر است این خروجی، با کمک خروجی J_Index تابع Jvalue که در قسمت قبل مورد توضیح قرار گرفت، به دست می‌آید. در نهایت، خروجی چهارم، TestSet، ویژگی‌های انتخابی را برای داده‌های تست در یک ماتریس قرار می‌دهد تا در ادامه آن‌ها را به طبقه‌بندی‌کننده ارائه کنیم و جواب نهایی را به دست آوریم.

نحوه عملکرد این تابع به این صورت است که ماتریس J را با استفاده از تابع Jvalue تشکیل می‌دهد و سپس در ستون‌های این ماتریس (طبق توضیحاتی که در قسمت قبل داده شد، این ماتریس برای حالت executive به تعداد $\binom{4}{2} = 6$ و برای حالت imagery به تعداد $\binom{3}{2} = 3$ ستون دارد)، به دنبال بزرگ‌ترین مقادیر J-value می‌گردد. هر ستون، J-valueهای بین دو فعالیت مشخص را نشان می‌دهد و چون ما به دنبال آن هستیم که تمامی فعالیت‌ها را از هم جدا کنیم، باید از هر ستون، مقادیر بزرگ J-value را شناسایی کنیم و از ویژگی متناظر با آن استفاده کنیم، چرا که ممکن است یک ویژگی دو فعالیت را از هم جدا کند، ولی برای سایر فعالیت‌های دیگر اثر مناسبی نداشته باشد، بنابراین باید برای جداسازی تمامی حالات دوبه‌دوی فعالیت‌ها، ویژگی‌های لازم را در نظر بگیریم.

با توجه به این توضیحات، در این تابع، تعداد ویژگی‌های خواسته شده را (که ورودی تابع است) تقسیم بر تعداد ستون‌های ماتریس J کرده، و عدد حاصل را به عنوان تعداد ویژگی‌های انتخابی از هر ستون در نظر می‌گیریم. به عنوان مثال به ازای ورودی‌های exe و 120 ویژگی، از هر ستون ۲۰ ویژگی که بیشترین J-valueها را دارند، انتخاب می‌شود. ضمنا برای حالتی که عدد ورودی بر تعداد ستون‌ها بخش‌پذیر نباشد نیز تمهیدات لازم اندیشیده شده است که در آن صورت، تعداد ویژگی‌های استخراجی از ستون‌های مختلف حداکثر در ۱ واحد تفاوت خواهند داشت.

در ادامه نیز، با استفاده از سایر خروجی‌های تابع Jvalue، ویژگی‌های انتخابی را در ماتریس TrainSet قرار می‌دهیم. همچنین، با توجه به ساختار داده‌هایی که در اختیار داریم (و می‌دانیم که هر کلاس، ۲۰ آزمایش دارد)، بردار labels نیز تشکیل داده می‌شود و خروجی‌های تابع Feature_Selector تشکیل می‌شوند.

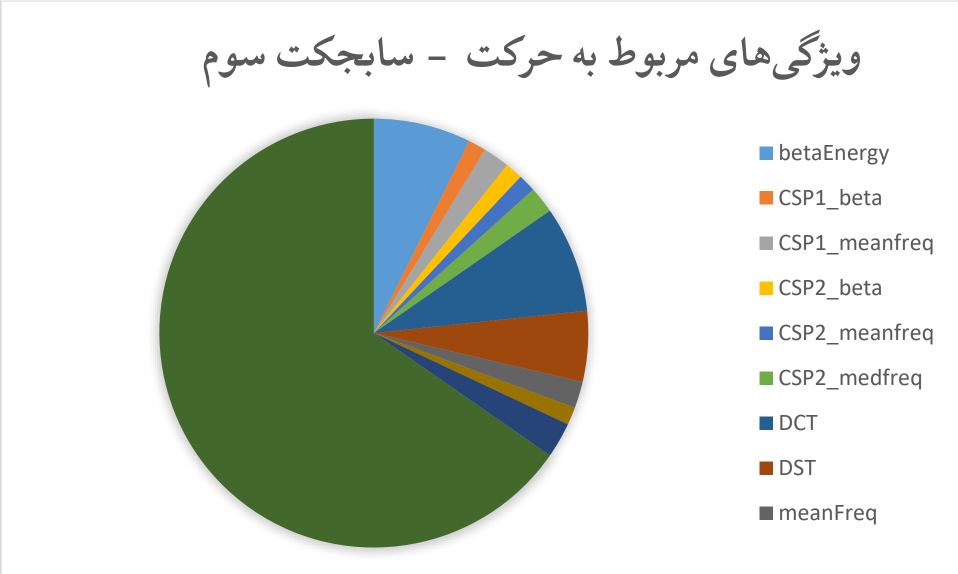
همچنین نحوه محاسبه TestSet نیز کاملاً مشابه روال محاسبه TrainSet است.

بخش سوم* ــ بررسی ویژگی‌های انتخاب شده

با استفاده از تابع فوق **۱۵۰ ویژگی** با بیشترین JValue داده‌های **حرکت** و **۱۰۰ ویژگی** با بیشترین JValue داده‌های **تصور حرکت** را به دست آوردیم. در این بخش درباره این موضوع بحث می‌کنیم که اصولاً کدام یک از ویژگی‌هایی که به دست آوردیم، برای جدا کردن داده‌های حرکت و داده‌های تصور تحرک مناسبند.

۱ ــ۳*) ویژگی‌های انتخاب‌شده برای داده‌های حرکت

چارت های زیر نشان می‌دهند که ویژگی‌های انتخاب شده برای حرکت از چه دسته‌هایی بوده‌اند.



به نظر می‌رسد استفاده از STFT برای جدا کردن داده‌های حرکت بسیار موثر است.

در بین انرژی کانال‌های مختلف، انرژی کانال بتا بیشترین تاثیر را دارد. این موضوع با توجه به اطلاعاتی که در مورد کانال بتا داریم، یعنی

Beta activity is closely linked to motor behavior and is generally attenuated during active movements.

نیز قابل حدس زدن بود. در تحلیل CSP نیز انرژی باند بتای CSP در ویژگی‌های انتخاب شده به چشم می‌خورد.

از بین ویژگی های فرکانسی ذکر شده در جدولِ صورت سوال، فرکانس میانه و میانگین بیشتر قابلیت جدا کردن داده‌ها را داشتند.

ضرایب تبدیل فوریه سینوسی و کسینوسی نیز ۵ و ۶ درصد از ویژگی‌های انتخاب شده را تشکیل می‌دهند.

همان‌طور که حدس می‌زدیم برای جدا کردن داده‌های دسته‌های مختلف، ویژگی‌های زمانی به دلیل نویز زیاد و تا حدی غیر قابل حذف EEG تاثیر چندانی ندارند.

ویژگی های موثر به ترتیب STFT، انرژی باند بتا، ضرایب تبدیل فوریه سینوسی و کسینوسی، فرکانس میانگین و میانه و همچنین چند ویژگی فرکانس استخراج شده از CSP مانند فرکانس میانگین و فرکانس میانه و انرژی باند بتا است.

۲ ـ۳*) ویژگی‌های انتخاب شده برای داده‌های تصور حرکت

برای داده‌های تصور حرکت، ۱۰۰ ویژگی با بیشترین JValue را در نظر می‌گیریم. چارت مربوط به این ویژگی‌ها در صفحه بعد رسم شده است.

همان‌طورکه انتظار داشتیم ویژگی CSP بیشترین تاثیر را در جداکردن داده‌های تصور حرکت دارد. دلیل اصلی اضافه‌کردن CSP لیست ویژگی‌ها نیز جداکردن داده‌های تصور حرکت بود. مقالهٔ .Wang et.al نیز که در بخش معرفی CSP ذکر شدهمین ادعا را داشت.

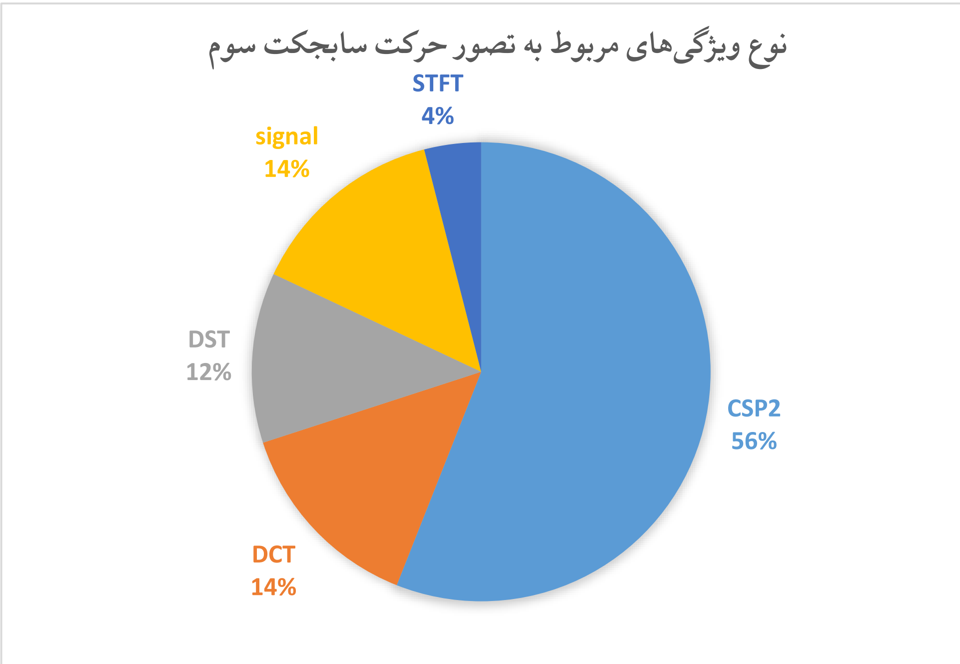
برعکس حرکت که در آن STFT تاثیر فراوانی داشت در اینجا تنها سه درصد ویژگی‌ها را STFT تشکیل می‌دهد.

تبدیل فوریه سینوسی و کسینوسی باز هم برای جداکردن داده‌های تصور حرکت مناسب هستند.

نکته‌ی جالبی که در این‌جا به چشم می‌خورد این است حدود ده ویژگی از جنس خود سیگنال، که انتظار داشتیم برای جداکردن داده‌ها مناسب نباشند نیز JValue بالایی دارند.

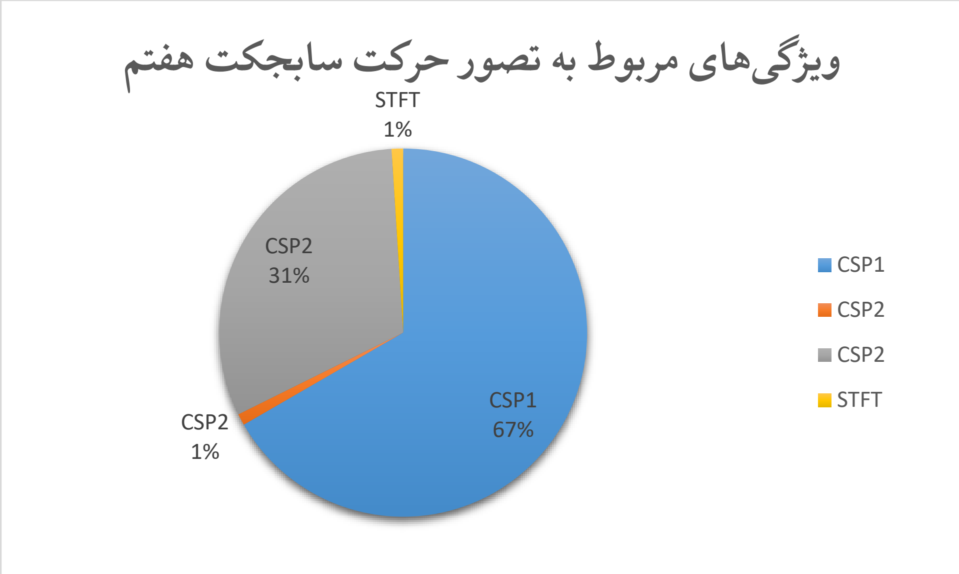
برخلاف حرکت، انرژی باندهای فرکانسی JValue بالایی نداشت و همچنین خود سیگنال CSP مهم بود و نه فیچرهای مستخرج از آن.

به طورکلی JValue های مربوط به داده‌های حرکت بزرگتر از JValue های داده‌های تصور حرکت هستند که این موضوع پذیرفتنی است.



اطلاعات و نمودارهای فوق متعلق با سابجکت سوم هستند. (Data{1}).

دیگر سابجکت ها نیز از الگویی تقریبا مطابق با این پیروی می‌کنند به جز تصور حرکت سابجکت هفتم یا (Data{4}) که ۱۰۰ ویژگی با بیشینه JValue آن به این صورت است.



به نظر می‌رسد برای این سابجکت، روش Common Spatial Pattern روشی بسیار مناسب است. بیش از نود و پنج درصد ویژگی‌های انتخاب شده برای این سابجکت از جنس CSP هستند.

البته باز هم لازم به ذکر است که سابجکت هفت موردی خاص بوده و سایر سابجکت‌ها شباهت زیادی به سابجکتی که در بالا بررسی شد دارند (سابجکت سه با Data{1}).

بخش چهارم ـ آموزش دادن طبقه‌بندی کننده و اعتبارسنجی طبقه‌بندی انجام‌شده

۴ـ۱) آموزش و طبقه‌بندی برای چند کلاس و تابع MultiSVM

می‌دانیم که با استفاده از توابع متلب، می‌توان طبقه‌بندی کننده برای دو کلاس را آموزش داد، اما برای داشتن چند کلاس، باید ابتدا طبقه‌بندی‌کننده‌هایی برای هر کدام از این کلاس‌ها آموزش دهیم، سپس با استفاده از تمامی آن‌ها، به طبقه‌بندی داده‌های جدید بپردازیم. (البته تابع fitcecoc در متلب، این کار را انجام می‌دهد ولی ما در این پروژه، برای طبقه‌بندی چند کلاسی، از طبقه‌بندی‌کننده‌های دوکلاسی استفاده کرده‌ایم)

برای رسیدن به این مقصود، تابع MultiSVM را با هدر زیر نوشته‌ایم:

```
function [result] = MultiSVM(TrainingSet, Label, TestSet)
```

این تابع با دریافت ماتریس ویژگی‌ها و بردار برچسب‌ها، ابتدا با استفاده از تعداد انواع برچسب‌ها، متوجه تعداد کلاس‌ها می‌شود و به همان تعداد، طبقه‌بندی کننده آموزش می‌دهد. برای آموزش هر طبقه‌بندی‌کننده، داده‌های یک کلاس را به عنوان گروه اول، و سایر داده‌های مربوط به تمامی کلاس‌های دیگر را به عنوان گروه دوم در نظر می‌گیرد. به این ترتیب، هر طبقه‌بندی‌کننده، وظیفه تشخیص یکی از کلاس‌ها را به عهده خواهد داشت.

نهایتاً، دیتای موجود در ماتریس TestSet (که قرار است مشخص شود که متعلق به کدام کلاس است) را به هر کدام از این طبقه‌بندی‌کننده‌ها می‌دهیم، و هرگاه یکی از آن‌ها خروجی ۱ بدهد، یعنی دیتای ورودی مربوط به کلاس متناظر با این طبقه‌بندی‌کننده بوده است و خروجی مشخص می‌شود.

الگوریتم فوق با استفاده از دو تابع svmtrain و svmclassify پیاده‌سازی شده است.

۴ـ۲) اعتبارسنجی طبقه‌بندی و توابع CrossVal و Final_CrossVal

نهایتاً برای بررسی کارایی طبقه‌بندی‌کننده‌ای که آموزش داده‌ایم، از روش k-fold cross validation استفاده می‌کنیم. در این روش، داده‌های آموزش را به k دسته تقسیم می‌کنیم، فرایند طبقه‌بندی را k بار انجام می‌دهیم، به این صورت که از هر دسته، یک بار به عنوان داده‌ی تست، و k-1 بار به عنوان داده‌ی طبقه‌بندی استفاده می‌شود. در هر مرحله (با توجه به این‌که می‌دانیم هر کدام از داده‌های تست در واقع به کدام کلاس تعلق دارند)، درصد خطای طبقه‌بندی‌کننده را در تعیین کلاس داده‌های تست محاسبه کرده، نهایتاً با میانگین‌گیری درصدهای خطای به دست آمده، ملاکی را برای عملکرد طبقه‌بندی‌کننده‌ی خود ارائه می‌دهیم.

الگوریتم فوق را در تابع CrossVal پیاده‌سازی کرده‌ایم:

```
function error_percentage = CrossVal(TrainData, labels, K)
```

ورودی‌های این تابع عبارتند از TrainData، کل داده‌ی آموزش که در اختیار داریم، labels، برچسب‌های متناظر با داده‌ها، و K، مرتبه‌ی k-fold cross validation

یک نکته در مورد این تابع آن که اگر تعداد نمونه‌های موجود بر k بخش‌پذیر نباشد، دسته‌ها را هم‌طول در نظر می‌گیریم و صرفاً ممکن است طول دسته‌ی آخر بیشتر شود.

طول دسته‌ها در برداری به نام L ذخیره می‌شود، سپس در یک حلقه با k بار تکرار، هربار داده‌های دسته‌ی iام به عنوان Test، و مابقی داده‌ها به عنوان Train در نظر گرفته شده، و با استفاده از تابع MultiSVM، طبقه‌بندی داده‌های تست به دست می‌آید و با برچسب‌های

اصلی آن مقایسه می‌شود. به این ترتیب، درصد خطا محاسبه شده و در بردار Error ذخیره شده و نهایتاً پس از پایان این حلقه، میانگین بردار Error به عنوان خروجی اعلام می‌شود.

نکته‌ی دیگر آن که در انتخاب دسته‌ها، از روش تصادفی استفاده کرده‌ایم، یعنی دسته‌ها به ترتیب ورودی انتخاب نمی‌شوند، بلکه به صورت تصادفی گزینش می‌شوند، و همین مسأله باعث می‌شود که اجرا کردن این تابع به صورت متعدد، نتایج دقیقاً یکسانی نداشته باشد.

برای آن که اثر این تصادفی بودن را میانگین‌گیری کنیم، تابع Final_CrossVal را با هدر زیر نوشته‌ایم:

```
function error_percentage = Final_CrossVal(TrainData, labels, K, n)
```

این تابع، ورودی‌هایی مشابه تابع CrossVal دارد و یک ورودی اضافه‌ی دیگر، یعنی n. این تابع، CrossVal را n بار پیاده‌سازی می‌کند و نهایتاً نتیجه را میانگین‌گیری می‌کند تا اثر تصادفی بودن ترتیب انتخاب دسته‌ها میانگین‌گیری شود و نتیجه‌ای نسبتاً دقیق از cross validation به دست آید.

بخش پنجم ـ آموزش دادن طبقه‌بندی کننده و اعتبارسنجی طبقه‌بندی انجام‌شده

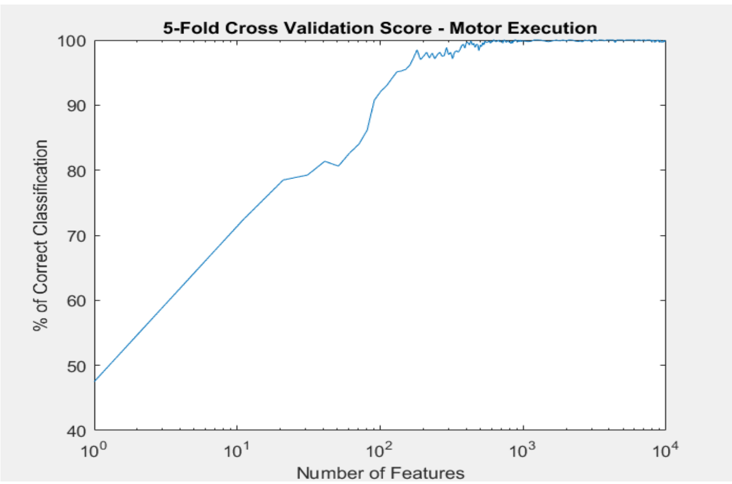
در این بخش با استفاده از ماتریس JValue به انتخاب ویژگی می‌پردازیم. در بخش ششم از PCA استفاده شده است.

اکنون آماده هستیم تا 5-Fold Cross Validation را اجرا کنیم.

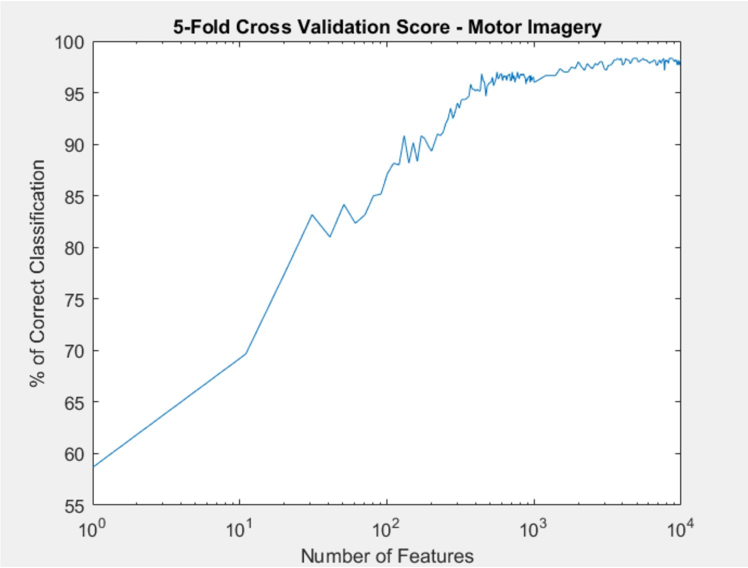
با استفاده از تابع Feature_Selector می‌توانیم n ویژگی با بیشترین JValue را پیدا کنیم. با استفاده از این n ویژگی، Cross Valid. را انجام می‌دهیم و درصد پاسخ‌گویی صحیح را بر حسب تعداد ویژگی انتخاب شده رسم می‌کنیم.

یک نکته در این‌جا قابل ذکر است و آن این است که درصد پاسخ صحیح در K-Fold باید بیشتر از درصد پاسخ صحیح برای داده‌های تست باشد زیرا در محاسبهٔ ماتریس JValue از داده‌های آموزش برای انتخاب ویژگی موثر استفاده شده است. به بیان دیگر ویژگی‌هایی انتخاب کرده‌ایم که برای داده‌های Train بتوانند فعالیت‌های مختلف را از هم جدا کنند پس درصد پاسخ درست K-Fold بایاسی به سمت بالا دارد زیرا در آن از خود داده‌های Train برای اعتبارسنجی طبقه بندی استفاده می‌کنیم. در آخر گزارش روشی برای کاهش این بایاس ارائه می‌کنیم.

شکل زیر نمودار درصد پاسخ صحیح 5-Fold بر حسب تعداد ویژگی‌های انتخاب شده است.



شکل ۱۱



شکل ۱۲

برای اطمینان از الگوریتم 5-Fold یک روش ساده تر و مشابه 5-Fold نیز ابداع کردیم که در آن داده ها به صورت تصادفی به K زیرمجموعه افراز می‌شوند و یک زیر مجموعه از آن به عنوان داده تست و بقیه به عنوان دادهٔ آموزش استفاده می‌شوند. این کار را هزار بار انجام دادیم و میانگین درصد پاسخ درست در هر مرحله را حساب کردیم. جواب‌ها با تقریب بسیار خوبی به K-fold نزدیک بودند.

برای نمونه کد زیر الگوریتم بالا را برای داده‌های حرکت اجرا می‌کند. در این‌جا صد ویژگی با JValue بزرگ را در نظر گرفته‌ایم.

```
DataSet = [];
labels = [];
[DataSet, labels, ~] = Feature_Selector(FData,'exe',100);

for i = 1 : 400
    I = randperm(80);
    TestIndex     = I(1:16);
    TrainIndex    = I(17:80);
    result = MultiSVM(DataSet(TrainIndex,:), labels(TrainIndex), DataSet(TestIndex,:));
    correct(i)  = sum(labels(TestIndex) == result')/length(result)*100;
end
correct = mean(error)
```

این روش نیز همانند 5-Fold به ما درصد پاسخ‌گویی صحیح برابر ۹۲ درصد با انتخاب صد ویژگی می‌دهد. انحراف معیار آن نیز ۸ درصد است. انحراف معیار، می‌تواند معیاری برای خطای طبقه‌بند باشد.

به نظر می‌رسد برای BCI این دقت بالایی است. البته باید توجه داشت که این درصد در K-fold به دست آمده و همان‌طور که در بالا عنوان شده درصد پاسخ صحیح آن بایاسی به سمت بالا دارد. با مطالعهٔ مقالاتی در این زمینه به روش‌های دیگر که به درصدهایی در این حدود می‌رسند نیز برخورد کردیم. برای طبقه بندی حرکت در BCI Competition iii به درصد های نزدیک ۹۹ درصد هم رسیده‌اند. درصدهای مذکور در سایت http://www.bbci.de/competition/iii/results/index.html موجودند.

ویژگی‌هایی مثل CSP برای طبقه‌بندی تصور حرکت و STFT نیز برای حرکت بیشترین JValue ها را داشتند که کاملا منطبق بر انتظارات ما بر مبنای پژوهش‌های انجام شده بود.

در کل نود هزار ویژگی در این تمرین محاسبه کردیم ولی حداکثر توانستیم k-fold را برای ۱۰۰۰۰ ویژگی با JValue بالا تکرار کنیم و نمودارهای بالا score این اعتبارسنجی‌ها بر حسب تعداد فیچرهاست.

یکی از انتظارات ما این بود که به دلیل overfitting، هنگامی که تعداد فیچرها را افزایش می‌دهیم، score ما ابتدا افزایش یافته و در جایی شروع به کاهش کند. این موضوع با مشاهدات ما سازگاری نداشت. ما برای این عدم سازگاری دو علت می‌توانیم ذکر کنیم. اول اینکه این score مربوط به cross validation است و انتظار فوق مربوط به وقتی است که از داده‌ها برای انتخاب فیچر مناسب استفاده نکرده باشیم، مثل بخش آخر تمرین. دوم اینکه شاید این کاهش در تعداد فیچرهای بالاتر که ما امکان سخت افزاری طبقه‌بندی با آن تعداد فیچر را نداشتیم به وقوع بپیوندد.

به علت دلیل اول و برای پرهیز از overfitting باید تعداد فیچرهای مورد استفاده را در حد معقول نگه داریم در اینجا یک حدس می‌زنیم و آن این است که تعداد فیچری که در از آن به بعد نمودار score بر حسب تعداد فیچر در شکل‌های ۱۱ و ۱۲ شروع به نوسان با فرکانس بالا می‌کند را به عنوان آستانه در نظر میگیرم. برای تست حدس خود در بخش آخر و برای طبقه‌بندی داده‌های رندوم، طبقه‌بندی را با چند عدد مختلف به عنوان تعداد فیچر انجام می‌دهیم و بررسی می‌کنیم که جواب آن چه میزان با جوابی که با آستانهٔ حدس زده شده تفاوت دارد.

<u>با توجه به آستانهٔ مذکور باید برای تصور حرکت ۱۰۰ فیچر و برای حرکت ۱۵۰ فیچر در نظر بگیریم.</u>

اگر برای حرکت به جای ۱۵۰ فیچر، ۱۰۰ فیچر در نظر بگیریم برای برای بخش آخر ۸ مورد از ۴۹ مورد از داده‌هایمان در کلاس متفاوتی قرار میگیرند یعنی **حداکثر** ۱۶ درصد در اسکور تاثیر دارند.

اگر برای حرکت به جای ۱۰۰ فیچر، ۱۰ فیچر در نظر بگیریم برای برای بخش آخر ۱۰ مورد از ۳۶ مورد از داده‌هایمان در کلاس متفاوتی قرار میگیرند یعنی **حداکثر** ۲۷ درصد در اسکور تاثیر دارند.

با وجود تغییر زیاد در تعداد فیچر به نظر می‌رسد تفاوت چشمگیری دیده نمی‌شود پس حدسمان را می‌پذیریم و از همین آستانه استفاده می‌کنیم.

برای طبقه‌بندی نهایی داده‌های تست، ابتدا SVM را با تعداد ویژگی‌هایی که در بالا مطرح شد و با تمام داده‌های آموزش train میکنیم و برای هر سابجکت، نتیجه طبقه‌بندی را به دست آورده و در فایلی با نام subi_exe.mat ذخیره کرده و ضمیمه می‌کنیم که I شمارهٔ سابجکت است.

در آخر تصمیم گرفتیم که اثر بایاس ذکرشده، به دلیل استفاده از داده‌های train برای انتخاب فیچرهای مناسب را بررسی کنیم. برای این کار در الگوریتم k-fold در هر مرحله با استفاده از داده‌های train و تشکیل ماتریس JValue برای آنها، ویژگی‌های مناسب را پیدا می‌کنیم. ایراد این روش این است که نمی‌توانیم با آن به یک مجموعه دادهٔ یکتا برای تست داده‌های بخش آخر برسیم اما روشی مناسب برای اعتبارسنجی و بررسی ویژگی‌های انتخاب شده است. این کار را برای داده‌های سابجکت اول تکرار کردیم.

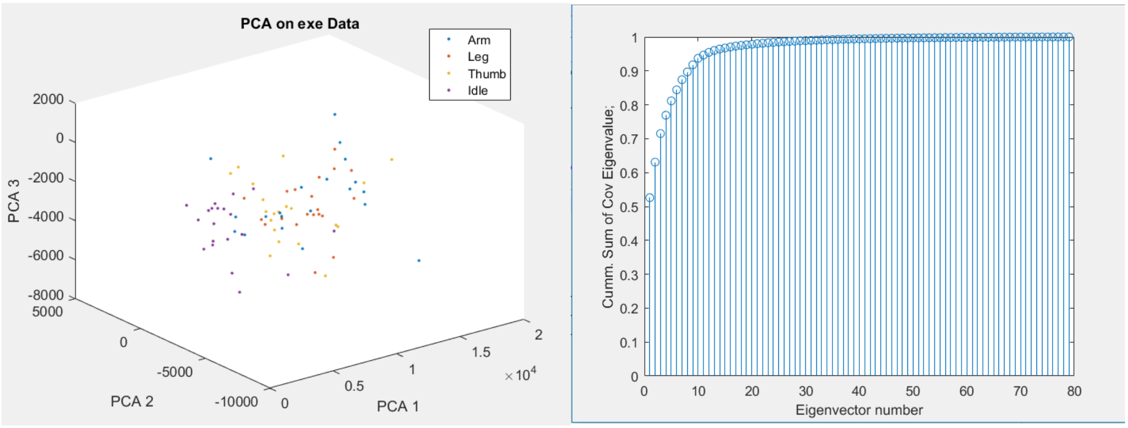
```
correct =

   93.7500   87.5000  100.0000  100.0000   87.5000  100.0000   81.2500   93.7500   87.5000   93.7500
```

این نتایج تفاوت چندانی با نتایج قبل ندارد و به نظر می‌رسد استفاده از همان صد ویژگی برای تصور حرکت و ۱۲۰ ویژگی برای حرکت منطقی باشد. در بخش بعد به‌جای ماتریس JValue از PCA استفاده خواهیم کرد.

بخش شش – PCA

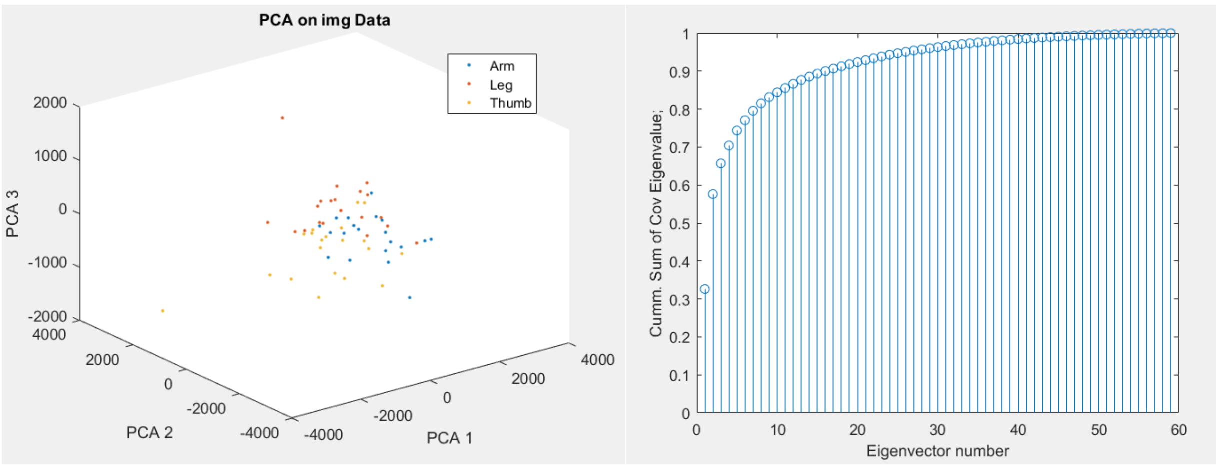
در آخر بر روی ماتریس فیچرهای خود PCA انجام می‌دهیم یعنی ماتریس کوواریانس را قطری میکنیم و بردارهای ویژه را بر حسب مقادیر ویژه متناظرشان sort می‌کنیم. این بردارویژه ها متعامد هستند و در درس نوروساینس آموختیم که جهت‌های PCA جهت های بیشینه واریانس داده در فضا به ما می‌دهند. اصولاً برای n داده، روش PCA حداکثر n جهت با مقدارویژه غیر صفر به ما می‌دهد چون داده‌های ما عضو زیرفضایی n بعدی از فضای بعد بالاتر ما هستند. از PCA می‌توان برای کاهش بعد داده ها استفاده کرد بدین صورت که تنها چند جهت با بیشترین واریانس رو در نظر گرفته و داده‌ها را بر آن تصویر کنیم و از باقی جهات چشم‌پوشی کنیم.



نمودار چپ، نمودار تصویر داده ها بر سه جهت اول PCA است که نگارنده از شدت جدا شدن این داده‌ها توسط تنها سه جهت به وجد آمده است. دیده‌می‌شود جهت اول PCA توانسته تا حد خوبی داده‌های Idle و Thumb را از هم جداکند.

نمودار سمت راست نیز توزیع تجمعی واریانس داده‌ای در جهت‌های مختلف PCA است. با استفاده از ۲۰ جهت اول PCA بیش از نود درصد واریانس کل را خواهیم داشت.

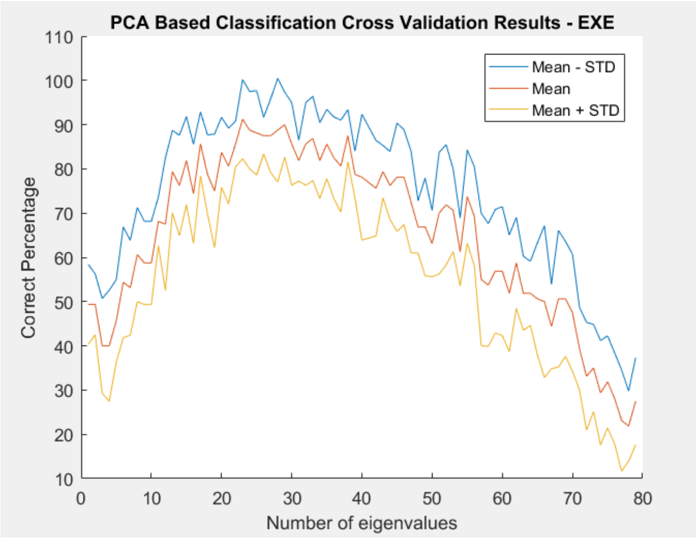
برای داده‌های تصور حرکت نیز این روش را تکرار می‌کنیم. نتایج باز هم خیره‌کننده‌اند.

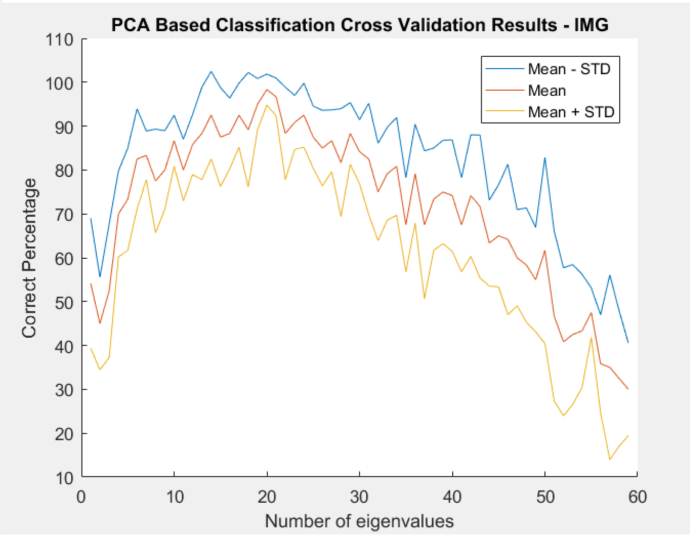


وجود این نتایج ما را ترغیب می‌کند تا برای دسته‌بندی از تصویر داده‌ها بر چند جهت اول PCA استفاده کنیم. به طورکلی این روش، روش بسیار مرسوم‌تری در ساخت BCI است.

در این روش باید تعداد بردارویژه‌ای را پیداکنیم که با استفاده از تصویر داده‌ها بر آن جهت‌ها و طبقه‌بندی به وسیله آن به بهترین جواب برسیم. همانند قسمت قبل این کار را برای تمام ۸۰ حالت ممکن تکرار می‌کنیم و با استفاده از 5-Fold Cross Validation بررسی می‌کنیم که با چه تعداد فیچر به بیشترین دقت می‌رسیم.

نمودار های زیر، Score ما در 5-Fold Cross Validation بر حسب تعداد بردارویژه انتخاب شده است.





در این نمودار به وضوح Over Classification مشهود است. اگر تعداد فیچرها را از حدی بیشتر بگیریم، درصد تشخیص شروع به کاهش می‌کند.

از انحراف معیار نتایج K-Fold به عنوان معیاری از خطای Classification استفاده می‌کنیم.

نتایج به وضوح بهتر از نتایج به دست آمده با ماتریس‌پخشی هستند.

بیشینهٔ نمودارها در نقاط زیر به دست آمدند.

	تعداد بردار ویژه	درصد میانگین	انحراف معیار
تصور حرکت	20	98.33%	3%
حرکت	20	91.33%	5%

برای هر دو ست داده، ۲۰ بردار ویژه در نظر میگیریم و داده‌های تست را طبقه‌بندی می‌کنیم.

به دلیل‌زمان بالای پردازش، داده‌های تست سه سابجکت اول با این روش نیز طبقه‌بندی شدند. نتیجه دو طبقه‌بندی ما در حدود ۷۰ درصد با یکدیگر یکسان شد که بسیار نشانه خوبی است!

بخش هفت – جمع‌بندی و نکات نهایی

فرایند کلی حل این تمرین برای استخراج ویژگی‌های مناسب به این شکل بود که ابتدا ویژگی‌های ذکر شده در دستور کار را مورد بررسی قرار دادیم، سپس با بررسی مقالات موجود و نیز نتایج مسابقات برگزارشده در زمینه BCI، به جستجوی ویژگی‌های جدید برای تشخیص بهتر کلاس‌های گوناگون پرداختیم. در این میان، آن‌چه مشاهده شد، یک ویژگی با عملکرد چشمگیر برای فعالیت‌های حرکتی و یک ویژگی با عملکرد چشمگیر برای فعالیت‌های تصوری بود که این دو ویژگی عبارتند از:

STFT(Short-Time Fourier Transform) و CSP(Common Spatial Pattern)

در ادامه با تشکیل ماتریس‌های J-value، به انتخاب ویژگی‌های مناسبی پرداختیم که بیشترین مقادیر J-value را دارا باشند تا بتوانند عمل جداسازی کلاس‌های گوناگون را به بهترین نحو انجام دهند.

در ادامه با اعتبار طبقه‌بندی را از طریق k-fold cross validation بررسی کردیم و اثر تعداد ویژگی‌های انتخابی را بر درصد تشخیص صحیح بررسی نمودیم. نهایتاً مشاهده کردیم که با افزایش تعداد ویژگی، درصد پاسخگویی بالا می‌رود اما با توجه به مسأله overfitting و نوسانات موجود در نمودار مربوط به درصد پاسخگویی، حد بهینه‌ای را برای تعداد ویژگی‌ها انتخاب کردیم و با ویژگی‌های انتخاب شده، داده‌های تست را طبقه‌بندی کردیم.

در پایان نیز به بررسی این نکته پرداختیم که درصدهای پیش‌بینی‌شده از طریق cross validation احتمالاً نمی‌توانند به خوبی، نتیجه‌ی طبقه‌بندی داده‌های تست را از نظر درصد پاسخگویی پیش‌بینی کنند که علت این امر، آن است که در محاسبه J-valueها، از میانگین کلی داده‌ها و نیز واریانس آن‌ها استفاده می‌شود و این یعنی اگرچه در cross validation داده‌ها را به دو دسته‌ی تست و آموزش تقسیم می‌کنیم، اما ردپایی از داده‌های تست نیز در ملاک‌های طبقه‌بندی وجود دارد، بنابراین نمی‌توانیم ادعا کنیم که cross validation دقیقا عمل مشابه با داده‌های تست خارجی را شبیه‌سازی می‌کند. برای بررسی بیشتر این امر، J-valueهای مربوط به داده‌های تست و آموزش را در فرایند cross validation به شکل جداگانه برای آموزش حساب کردیم و با محاسبه‌ی درصدهای حاصل، سعی کردیم بایاس موجود را از بین ببریم که نهایتاً منجر به تأیید نسبی نتایج اولیه شد.

در آخر برای انتخاب ویژگی مناسب از روش مرسوم PCA استفاده کردیم. در این روش مشکلات روش JValue وجود نداشت و توانستیم نتایجی با تطابق بالا با انتظارات تئوری بگیریم. در این روش با انتخاب ۲۰ بردار ویژه اول به درصد تشخیص حرکت و تصور بالای نود درصد رسیدیم.

References

[1] Moritz Grosse-Wentrup, Martin Buss. *"Multiclass Common Spatial Patterns and Information Theoretic Feature Extraction"*, **IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING**, VOL. 55, NO. 8, 2008.

[2] Yijun Wang, Shangkai Gao, Xiaorong Gao. "*Common Spatial Pattern Method for Channel Selection in Motor Imagery Based Brain-computer Interface*", **Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference**, 2005.

[3] Hohyun Cho, Minkyu Ahn, Sangtae Ahn, Moonyoung Kwon, and Sung Chan. *EEGdatasets for motor imagery brain–computer interface*, doi: 10.1093/gigascience/gix034.

[4] Seyed Navid Resalat and Valiallah Saba, *a Study of Various Feature Extraction Methods on a Motor Imagery Based Brain Computer Interface System*, **Basic Clin Neurosci.** *2016.*

[5] Jaime Camacho, Vidya Manian, *Real-time single channel EEG motor imagery based Brain Computer Interface*, **World Automation Congress (WAC)**, *2016.*

[6] BCI competition Results, http://mlin.kyb.tuebingen.mpg.de/BCIWebpage/Code.html.