2020 Fall 개별연구 : 주간 탐구일지

HD EMG 부착 시 따르는 position과 impedence 변화의 보정방법 연구

개별연구생 이혜민
소 속 전기및전자공학부
학 번 20190533
이메일 byj4565@kaist.ac.kr

담당 교수님 박형순
사수님 조성현

● 주제 설명

HD EMG 특성 상 매번 붙이고 떼는 과정을 반복해야하고 이 때마다 항상 같은 위치와 상태로 붙이는 것은 불가능하다. 즉, 매번 HD EMG의 위치와 impedence, 신호의 세기 또는 패턴 등이 같을 수 없기 때문에 이를 보정해줄 필요가 있다.

따라서 다양한 환경에서의 data를 먼저 수집한 후에, 해당 data를 바탕으로 당일의 EMG는 어느 위치에 부착된 것으로 추정되며 따라서 그날 입력되는 신호들은 어떻게 해석하는 것이 적합한지를 판단하고 이에 맞게 손가락이 움직일 수 있도록 해주는 알고리즘을 연구할 예정이다.

● URP 신청 : 11월 중순 마감

예상 계획	9월	초	Ref1의 classification 구현 디버깅
		말	Ref1의 calibration 과정 공부 후 우리의 환경에 맞게 구현 (현재 idea : 1개의 GMM을 기준으로 main muscle activity estimation & position correction)
	10월	초	------- (중간고사 : 10월 19일 - 10월 23일) -------
		말	1명의 일반인을 대상으로 HD EMG를 이용한 실험 (1차)
	11월	초	URP와 논문을 위한 주제 고안, URP 신청서 작성
		말	기존 코드에 실험 데이터 적용 및 알고리즘 수정
	12월	초	------- (기말고사 : 12월 14일 - 12월 18일) -------
		말	알고리즘 개선 및 결과 정리

1. 개별연구 주간 탐구일지

작성일	9월 1주차	2020.09.04. (금)	작성자	이혜민
주간 목표	☑ 가을학기 목표 정리하기 ☐ Ref1의 classification 까지의 과정 디버깅 완료하기			

● Numbering 규칙

type	변경	Mentioning	File name	Inside code
session	전	0 ~ 4	0 ~ 4	sessions[] : 0~4
	후	1 ~ 5	1 ~ 5	
gesture	-	0 ~ 26	0 ~ 26	gestures[] : 0~26
try	-	1 ~ 10	-	gestures[][] : 0~9
channel	전	0 ~ 167 (191)	-	gestures[][][] or [][][][] : 0~167 (191)
	후	1 ~ 168 (192)		
window	전	0 ~ 39	-	gestures[][][] : 0~39
	후	1 ~ 40		

● gestures=np.apply_along_axis(butter_bandpass_filter, 2, gestures)

위 코드로 butterworth bandpass filter 적용함. 그래프로 관찰한 결과, 유의미한 변화가 있는 것으로 보임. 제대로 작동 중. (한 session 소요 시간 : 2m 55s)

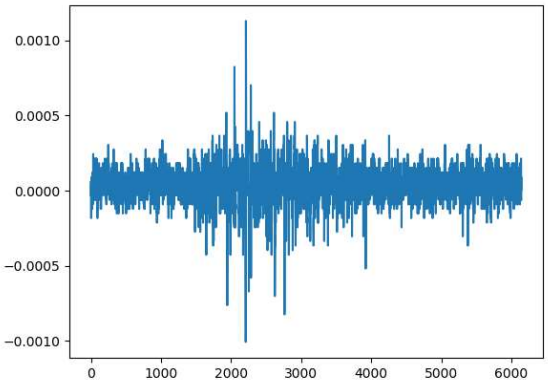

그림 1 . Before bandpass filtering : session5 gesture7 try7 ch81 (before eights deleting)

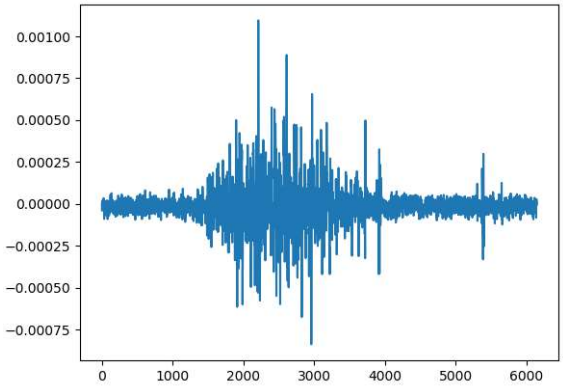

그림 2 . After bandpass filtering : session5 gesture7 try7 ch81 (before eighhts deleting)

● base normalization, median filtering 전후

육안 상으로는 큰 문제 없이 잘 되고있는 것 같다!

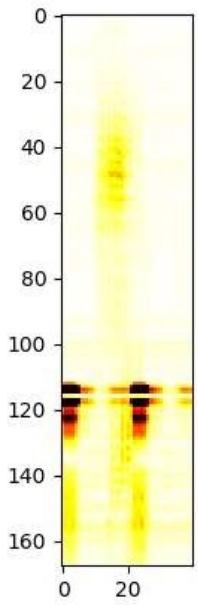

그림 3 . Before base normalization : session5 gesture7 try7

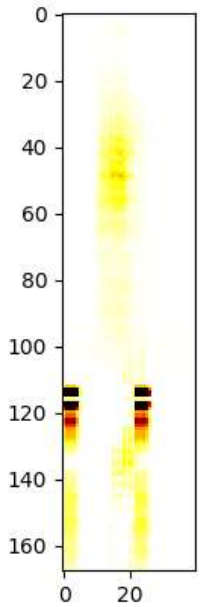

그림 4 . After base normalization : session5 gesture7 try7

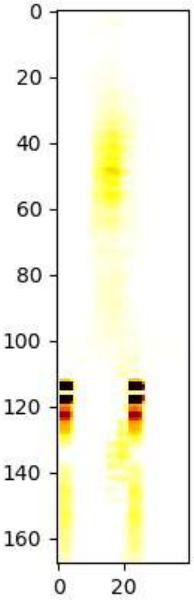

그림 5 . After median filtering : session5 gesture7 try7

● 이때까지 진행상황에 적용한 또 다른 예시

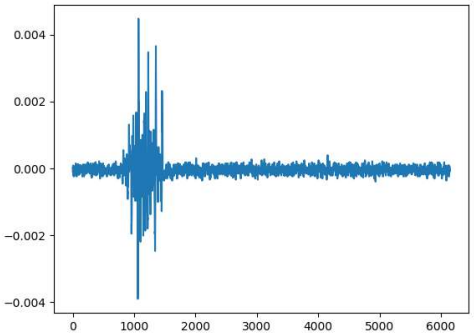

그림 6 . Before bandpass filtering : session5 gesture3 try3 ch51 (before eights deleting)

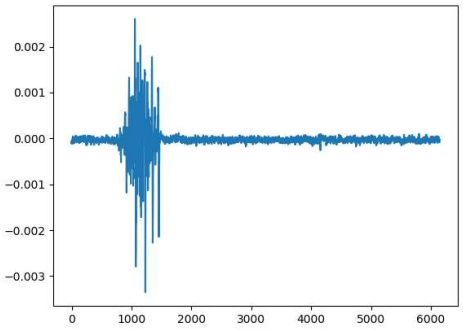

그림 7 . After bandpass filtering : session5 gesture3 try3 ch51 (before eights deleting)

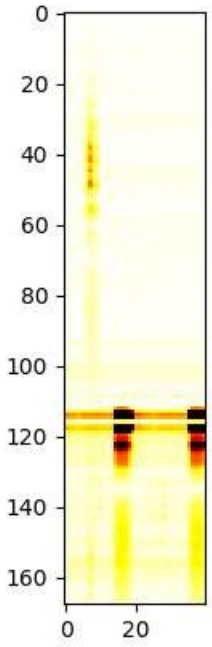

그림 8 . Before base normalization : session5 gesture3 try3

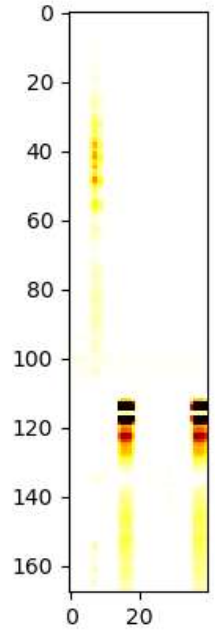

그림 9 . After base normalization : session5 gesture3 try3

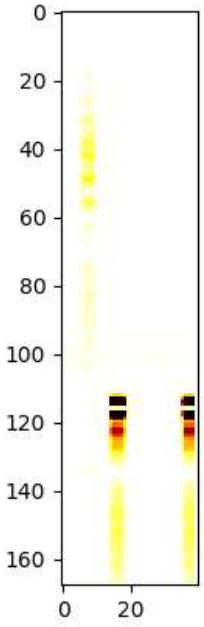

그림 10 . After median filtering : session5 gesture7 try7

2. 개별연구 주간 탐구일지

작성일	9월 1주차	2020.09.05. (토)	작성자	이혜민
주간 목표	☑ Ref1의 classification 까지의 과정 디버깅 완료하기			

● Active window extracting

예전에는 아래와 같은 segmet가 만들어졌었다.

```
PS C:\Users\byj45\OneDrive\4 KAIST\20 Summer Indivisual study\code> & C:/Users/byj45/AppData/
Local/Programs/Python/Python37/python.exe "c:/Users/byj45/OneDrive/4 KAIST/20 Summer Indivisu
al study/code/Building_ref1.py"
0번째 gesture의 각 try의 segment 길이들 : 6 21 17 6 24 11 9 16 13 5
1번째 gesture의 각 try의 segment 길이들 : 3 12 12 5 7 11 11 18 9 10
2번째 gesture의 각 try의 segment 길이들 : 6 5 7 6 5 6 7 16 6 6
3번째 gesture의 각 try의 segment 길이들 : 6 5 5 5 5 4 4 6 5 5
```

그림 11 . 여름방학 때의 session1 gest0,1,2,3 active windows extracting 결과

그런데 이번에 코드를 단순화하면서 다시 구성하니 모든 segment의 길이가 5 또는 6으로 extracting 되었다. 추측하건데 active window를 예전에는 잘못 판별했나 보다.

```
0번째 gesture의 각 try의 segment 길이들 : 6.0 5.0 5.0 6.0 5.0 6.0 5.0 6.0 6.0 6.0
1번째 gesture의 각 try의 segment 길이들 : 6.0 6.0 6.0 5.0 6.0 6.0 6.0 6.0 6.0 6.0
2번째 gesture의 각 try의 segment 길이들 : 6.0 5.0 5.0 5.0 5.0 6.0 6.0 6.0 6.0 6.0
3번째 gesture의 각 try의 segment 길이들 : 6.0 6.0 5.0 6.0 5.0 6.0 8.0 6.0 6.0 6.0
4번째 gesture의 각 try의 segment 길이들 : 6.0 13.0 10.0 14.0 12.0 9.0 12.0 6.0 6.0 9.0
```

그림 12 . 현재 session5 gest0,1,2,3,7 active windows extracting 결과

● Session4 gesture 0,1,2,3,7 confusion matrix

디버깅 전에는 **accuracy가 35%**였던 데에 비해 지금은 **75%** !!!

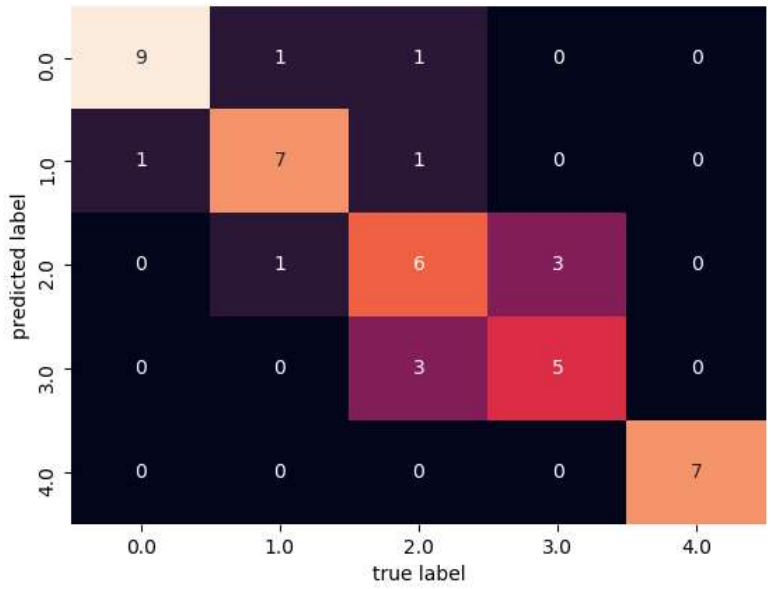

그림 13 . 2020.09.05. Confusion matrix : session5 gesture0,1,2,3,7

● 모든 session을 한 번에 넣었을 때

Accuracy : **61%**

매우 정상적이다!

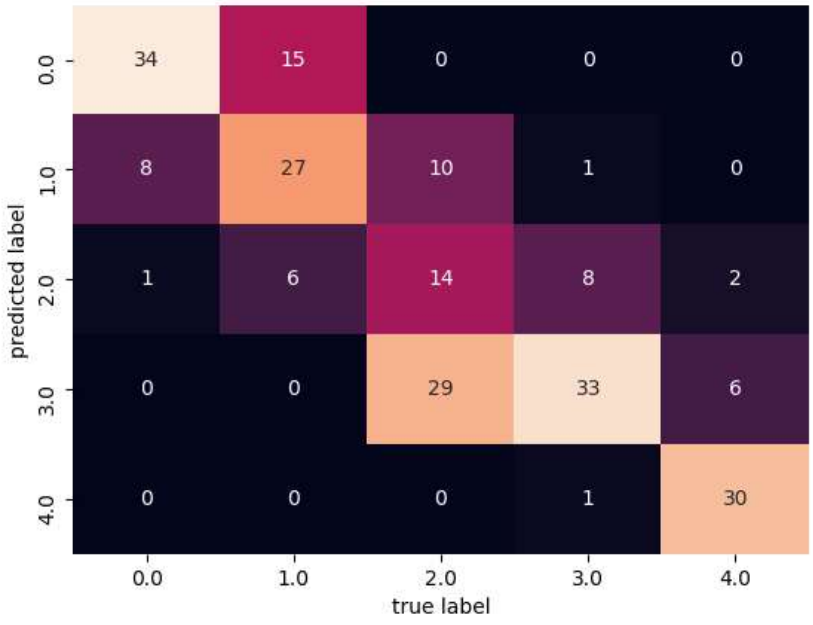

그림 14 . 2020.09.05. Confusion matrix : session1,2,3,4,5 gesture0,1,2,3,7

3. 개별연구 주간 탐구일지

작성일	9월 2주차	2020.09.11. (금)	작성자	이혜민
주간 목표	☐ 최적화하지 않은 후반부 코드 최적화하기 ☐ GMM 공부			

- ACTIVE_gestures

shape : (5, 10, n, 168, 150)

meaning : (gestures, tries, ACTIVE windows, channels, data in each window)

● 각 함수의 실행 시간을 알기 위해서 PRINT_TIME_CONSUMING을 추가함 (session 4만 input)

Loading mat files: 3.58

Discard_useless_data: 0.54

Apply_butterworth_band_pass_filter: 116.55

Divide_continuous_data_into_150_samples_window: 0.35

Compute_RMS: 6.45

base_normalization: 0.00

Median filtering: 0.86

extract_ACTIVE_window_i: 0.00

ACTIVE_filter: 2.81

Repartition_N_Compute_RMS: 1.69

mean_normalization: 0.06

construct_X_y: 0.00

extract_X_y_for_one_session: 130.57

Training: 0.00

Testing: 0.00

Accuracy : 73%

Testing: 139.71

Loading mat files: 3.58 -> 초기 저장 방식을 dictionary에서 list로 바꿈. 3.23

4. 개별연구 주간 탐구일지

작성일	9월 4주차	2020.09.25. (금)	작성자	이혜민
주간 목표	☑ Apply_butterworth_band_pass_filter 최적화			

● 오늘 실행시간 기준으로 수정 전후 비교하여 더 빠른 방법으로 선택

Loading mat files: 3.58

-> 오늘은 약 3초. dict가 더 보기 편할거 같아서 다시 dict로 돌림

Apply_butterworth_band_pass_filter: 116.55

-> 예전처럼 for문으로 바꿨더니 1초 이하로 줄음.. !?

Compute_RMS: 6.45

-> 오늘은 10.36초. for문으로 바꾸니 10.46초. 여러 시도에서 평균적으로 for문이 빠름. for문으로 바꾸자.

16.88?!?!? -> debugging한다고 python list로 구현했더니 엄청 시간이 뛴다. numpy array로 다시 구현했고, 최종 7~8초.

Median filtering: 0.86

-> 오늘은 1.67초. for 1.83초. 평균적으로 비슷. 기존 방식대로 진행.

ACTIVE_filter: 2.81

-> array를 del하기 때문에 어쩔 수 없이 list를 써서, 시간이 많이 걸림.

Repartition_N_Compute_RMS: 1.69

-> 각 try의 active window 수가 달라서 어쩔 수 없이 list 이용.

● 시간 최적화 전체완료 후 : 1 session 139.71 -> 17.11초

Loading mat files: 3.40
Discard_useless_data: 0.61
Apply_butterworth_band_pass_filter: 0.93
#
Divide_continuous_data_into_150_samples_window: 0.27
Compute_RMS: 4.58
base_normalization: 0.00
Median filtering: 0.82
extract_ACTIVE_window_i: 0.00
ACTIVE_filter: 2.93
Repartition_N_Compute_RMS: 1.77
mean_normalization: 0.01
construct_X_y: 0.00
extract_X_y_for_one_session: 13.35
Training: 0.00
Testing: 0.00
Accuracy : 75%
main: 17.11

5. 개별연구 주간 탐구일지

작성일	9월 5주차	2020.09.30. (수)	작성자	이혜민
주간 목표	☐ GMM 구현			

6. 개별연구 주간 탐구일지

작성일	9월 5주차	2020.10.02. (금)	작성자	이혜민
주간 목표	☑ Calibration을 위한 과정 정리하기			

현 상태 : 각 session의 gesture 마다 N개씩 data가 존재함. 길이 168의 1D array는 하나의 gesture를 대표함

주의할 점 : GMM(mean)의 개수, session의 개수는 변할 수 있다.

1. (24,7)의 형태로 xy plane에 나타낸다.
2. interpolate 하여 모든 data를 연속된 함수로 만든다.
3. 하나의 gesture를 선택하여 해당 N개의 data를 GMM estimate한다. : 2 component, maximum-likelihood estimation with expectation-maximization algorithm. -> 각 data의 flexor, extensor가 어디에 있는지 2개의 mean 값 return
4. 같은 session의 N개의 mean들을 평균냄. -> 각 session 별로 2개의 mean값 존재
5. 첫 번째 session의 mean 값과 다른 session의 mean 값을 비교하여 xy 평면에서 얼마나 차이가 있는지 계산하고 shift할 만큼을 결정. 이때 두 mean을 통해 결정한 각각의 shift를 평균내서 최종적으로 shift.
6. 5번의 data들에서 LOOCV로 classification하기
7. 6에서 나온 session 개의 accuracy 평균 내기.

7. 개별연구 주간 탐구일지

작성일	10월 1주차	2020.10.03. (토)	작성자	이혜민
주간 목표	☑ 모든 data interpolate 완료하기			

☑ (24,7)의 형태로 xy plane에 나타낸다.
☑ interpolate 하여 모든 data를 연속된 함수로 만든다.
☐ 하나의 gesture를 선택하여 해당 N개의 data를 GMM estimate한다. : 2 component, maximum-likelihood estimation with expectation-maximization algorithm. -> 각 data의 flexor, extensor가 어디에 있는지 2개의 mean 값 return
☐ 같은 session의 N개의 mean들을 평균냄. -> 각 session 별로 2개의 mean값 존재
☐ 첫 번째 session의 mean 값과 다른 session의 mean 값을 비교하여 xy 평면에서 얼마나 차이가 있는지 계산하고 shift할 만큼을 결정. 이때 두 mean을 통해 결정한 각각의 shift를 평균내서 최종적으로 shift.

☐ 5번의 data들에서 LOOCV로 classification하기
☐ 6에서 나온 session 개의 accuracy 평균 내기.

8. 개별연구 주간 탐구일지

작성일	10월 2주차	2020.10.06. (화)	작성자	이혜민
주간 목표	☐ data processing이 제대로 되지 않음. 해결하기			

1. Apply butterworth band pass filter가 제대로 되지 않음

9. 개별연구 주간 탐구일지

작성일	10월 2주차	2020.10.07. (수)	작성자	이혜민
주간 목표	☐ data processing이 제대로 되지 않음. 해결하기			

1. (해결) Apply butterworth band pass filter가 제대로 되지 않음
/////// 9월 25일 수정본이 잘못되었었다 ///////
Apply_butterworth_band_pass_filter: 116.55
-> 예전처럼 for문으로 바꿨더니 1초 이하로 줄음.. !?
//
-> 제대로된 for문으로 돌리고 apply_along_axis랑 비교해봤을 때, 평균적으로 for문이 10초 가량 빠름. 따라서 for문으로 만들기.

2. (해결) mean_normalization까지 다 한 후에 classifying이 제대로 되지 않음
문제 해결. 알고 보니, Compute_RMS를 apply_along_axis에서 for문으로 바꿀 때 잘못함.
/////// 9월 25일 수정본이 잘못되었었다 ///////
Compute_RMS: 6.45
-> 오늘은 10.36초. for문으로 바꾸니 10.46초. 여러 시도에서 평균적으로 for문이 빠름. for문으로 바꾸자.
16.88?!?!? -> debugging한다고 python list로 구현했더니 엄청 시간이 뛴다. numpy array로 다시 구현했고, 최종 7~8초.
//

3. 하나의 session만 넣었을 때는 잘 되지만, 여러 session을 넣었을 때 잘 안 됨
Ready_for_calibration version에서도 그럼.