מטרות התרגיל:

- היכרות עם יסודות עולם למידת המכונה.
- זיהוי תכוניות מתוך נתוני EEG באמצעות איורים ו-spectrograms (visual data exploration).
- בניית מודל classification במטלב.

מבוא תיאורטי:

ממשקי מוח-מחשב (ממ״ח) מהווים ערוץ תקשורת ישיר בין המוח האנושי ליחידת מחשוב חיצונית. בעבר נמצא[1] שבעת דמיון של תנועת גפיים, ישנם תדרים ספציפיים בקורטקס המוטורי המגיבים סלקטיבית לגפיים שונות. בניסוי עליו מושתת עבודה זו, השתמשו בממ״ח המבוסס על EEG על מנת להקליט נתונים בעת שאנשים דמיינו ביצוע פעולה באמצעות הגפיים שלהם. המטרה הסופית הייתה לזהות בזמן אמת מתוך הנתונים מתי הנבדקים מדמיינים פעולה עם גפיים ימניים ומתי הם מדמיינים פעולה עם גפיים שמאליים. כיוון שניתוח בזמן אמת של כל הנתונים לוקח משאבים חישוביים רבים, ניתחנו נתונים offline על מנת לחלץ מספר קטן יחסית של תכוניות אשר ירכיבו מודל אשר יכול להבדיל בין דמיון של ״ימין״ לבין דמיון של ״שמאל״.
כאשר המודל ״מוכן״*, ניתן להשתמש בו גם על נתונים בזמן אמת בצורה מהירה יחסית.
* ניתן (ואף מומלץ) להמשיך לעדכן את המודל ככל שמשתמשים בו, על מנת לשפר את אחוזי דיוק הניבוי שלו.

סקירת נתונים:

בתרגיל זה קיבלנו 2 קבצי mat המכילים (בין היתר) נתונים גולמיים שנאספו ע״י 3 אלקטרודות EEG מנבדק אחד, אך התבקשנו להתייחס רק לאלקטרודות C3 ו-C4.
הנתונים הכילו הקלטות של 6 שניות כך שלאחר 2.25 הנבדק התבקש לדמיין תנועה בגפיים ימניים/שמאליים בהתאם לחץ שהופיע על המסך שמולו.
תדר הדגימה הינו 128 הרץ ולכן כל אלקטרודה סיפקה וקטור באורך של $128 * 6 = 768$ תאים כך שכל תא בוקטור מייצג נקודת זמן שונה (1/128 שניות).
הקובץ הראשון, motor_imagery_train_data.mat, הכיל 128 ניסויים והשתמשנו בו על מנת לאמן את המודל, כל ניסוי (trial) תוייג עבורנו כניסוי ״שמאל״ או ״ימין״. הקובץ השני, motor_imagery_test_data.mat, הכיל 32 ניסויים לא מתוייגים, ועליהם ביצענו ניבוי עבור כל ניסוי האם הוא ״ימין״ או ״שמאל״.

[1] Hassan, M. A., Ali, A. F., & Eladawy, M. I. (2008, December). Classification of the imagination of the left and right hand movements using eeg. In *2008 Cairo International Biomedical Engineering Conference* (pp. 1-5). IEEE.

שיטה:

את הנתונים שקיבלנו חילקנו לשני חלקים: נתונים בהם התרחש דמיון של הפעלת גפה ימנית ונתונים בהם התרחש דמיון של הפעלת גפה שמאלית. בשני חלקים אלה נלקח זמן שנע בין 6-2.25 שניות (הזמן בו התרחש ביצוע המטלה עצמה). לאחר מכן, על מנת לזהות תכוניות אינפורמטיביות לטובת הסיווג ביצענו מניפולציות שונות על הנתונים:

- ויזואליזציה של הנתונים הגולמיים של 20 ניסויים אקראיים עבור כל צד.
- פונקציית PWelch אשר משתמשת במיצוע על פני חלונות על מנת להפחית רעש בנתונים הגולמיים. לאחר מכן הצגת הנתונים ב-Power Spectra אשר מציגה עוצמת תדר עבור תחום התדרים הנבחרים.
- Heatmap אשר מציגה אנרגיה של התדרים הנבחרים כפונקציה של זמן.
- Histogram – מאפשרת בדיקת אפקטיביות ההפרדה של התוכניות הנבחרות לצדדים השונים. ההיסטוגרמה מציגה ספירת של עוצמות תדרים בימין ומשמאל עבור תחום תדרים מסוים.

באמצעות סקירת השיטות לעיל, חילצנו **32** תכוניות שונות מנתוני הEEG.

רשימת התכוניות:

C3 features:

15.8-17.2 Hz; 8.2-10.3Hz; 8.5-9.8Hz; Root total power; spectral moment; spectral entropy; log(bands)*; sqrt(bands)*; Relative log power(bands)*.

C4 features:

16.5-17.3 Hz; 16.4-16.8Hz; 15.7-16.8Hz; Root total power; spectral moment; spectral entropy; log(bands)*; sqrt(bands)*; Relative log power(bands)*.

C3 – C4 features:

8.5-10.4Hz; 15-17Hz; sqrt(bands)*.

* הכוונה היא הפעלת הפונקציה על התדרים שצוינו עבור כל אלקטרודה.

- השתמשנו בPrincipal component analysis (שיטה הדוחסת את הנתונים ומפחיתה מימדים) לשני צרכים. ראשית על מנת להטיל את הנתונים הגולמיים על הprincipal components והצגת התוצאה בדיאגרמות פיזור של שניים ושלושה מימדים כדי לראות אם ניתן להפריד את סוגי הניסויים (ימין/שמאל) לינארית. שנית, כיוון שהשתמשנו במספר רב של תכוניות, רצינו להפחית את המימדים על מנת להימנע מoverfitting (התאמת יתר לנתוני הtraining אשר עשויה לפגוע בניבוי הסופי). לאחר בדיקת מספר אפשרויות, ראינו ששימוש ב**8 PCs** סיפק את התוצאות הטובות ביותר.
- מודל הסיווג בו השתמשנו הוא מסוג linear discriminant analysis והשתמשנו ב k-fold cross validation (CV) על מנת לאמן את המודל. k-fold CV מחלקת את הנתונים ל-k חלקים ומאמנת את המודל על k-1 מהחלקים (training set) ובודקת אותו על החלק הנותר (validation set). החלוקה לקבוצות זרות בין הtraining לvalidation חשובה כיוון שאחרת המודל כבר יודע את התשובות לכל הניבויים וכך יש סכנה גבוהה לoverfitting שבעקבותיו הניבוי על סט המבחן יהיה גרוע. חשוב לא לבחור k גדול מדי המוביל לvalidation set קטן, שכן כך יכולת ההכללה של המודל לעולם האמיתי (ובפועל כאן, סט המבחן) תהיה נמוכה. כדוגמת קיצון אם נגיע למצב של סט validation בגודל אחד

ובמקרה ננבא עבורו את התשובה הנכונה, נקבל אחוזי דיוק של 100%, אך כמובן שכאשר ננסה לנבא עבור סט המבחן נקבל תוצאות הרבה פחות טובות. כלומר, בחירת k גדול מדי פוגעת ביכולת שלנו לדעת עד כמה המסווג שלנו טוב. בחירת k קטן מדי תקטין את כמות הנתונים עליהם המודל יכול להתאמן, ובכך תפגע ביכולת הלמידה שלו וההכללה שלו.

- אימנו את המודל מספר פעמים ולקחנו את ממוצע הניבויים בתור הניבוי הסופי של המודל.

ממצאים שנצפה לקבל בניסוי:

כאמור, ישנם ממצאים[1] שמראים שניתן לזהות דפוסי פעילות שונים בפעילות המוחית בעת דמיון של "ימין" לעומת "שמאל", ולכן נצפה שיהיה אפשר להפריד את סוגי הניסויים. עם זאת, ייתכן שההפרדה המיטבית שניתן לעשות היא לא לינארית (במאמר המצוטט התייחסו לעבודה קודמת בה השתמשו בLDA עם אחוזי דיוק מקסימליים של 73.9%). כיוון שאנו משתמשים במודל לינארי בלבד, ייתכן כי לא נצליח לבצע הפרדה טובה במיוחד.
מבחינת ביצועי המסווג, נצפה לביצועים גבוהים יותר בtraining מאשר בvalidation ובtest כיוון שהמסווג יודע את התשובות הנכונות בעת האימון ולכן יכול להתאים את עצמו אליהן.

הסבר קטעי קוד מרכזיים:

- **Ex6_315025148_204133151.mat:**

בחלק זה של הקוד הראשי ייצרנו heat map עבור הנתונים ובאמצעות התבוננות בממצאים חילצנו את התדרים והזמנים שמופיעים כאן. הסבר מפורט בהמשך.

spectrogram

```
...
...
...
hotmap(rightDataAll,leftDataAll,numTrials,noverlap,fs);
% Extracted bands and times for each band from viewing the heatmaps per electrode
C3bands = {[15.8,17.2],[8.2,10.3],[8.5,9.8]};
C4bands = {[16.5,17.3],[16.4,16.8],[15.7,16.8]};
C3DiffC4Bands = {[8.5,10.4],[15,17]};
C3DiffC4Times = {589:627,537:704};
C3Times = {563:717,499:627,589:627};
C4Times = {550:601,589:627,678:730};
...
...
...
```

(סוף הקוד הראשי). לאחר אימון המודל, טענו את סט המבחן וחילצנו ממנו את התכוניות (הסבר עלget_features בהמשך). השתמשנו (כמו באימון) בפונקציית classify של מטלב עם הפרמטר 'linear'.
הפונקציה מחשבת linear discriminant analysis (LDA) על מנת לנסות לבצע הפרדה לינארית בין שני סוגי הtrials. LDA היא שיטה המוצאת קומבינציה לינארית של התכוניות המפרידה את בין שני (או יותר) classes.

Prediction

```
load motor_imagery_test_data.mat;

testData = data(:,:,1:2);
testFeatures =
get_features(testData,C3bands,C3Times,C4bands,C4Times,C3DiffC4Bands,C3DiffC4Times,imageryStart,imageryEnd,size(testData,1),
window,noverlap,frequencies,fs);

Predictions = classify(testFeatures,features,P_C_S.attribute(4,:));  % 1 = right
...
...
```

- **side_data_extractor.mat:**

מטרת הפונקציה היא לשלוף מהנתונים הגולמיים את השורות עבור הtrials של צד ספציפי, כפי שמוגדר על ידי הוקטור @indicators, בחתך הזמן של הvisual imagery כפי שמוגדר ע"י @imageryEnd, @imageryStart.

```
function [data] = side_data_extractor(rawData,indicators,numTrials,numSamples,numChannels,imageryStart,imageryEnd)
%SIDE_DATA_EXTRACTOR extracts data for @numTrials for a specific side (left/right) from
%@rawData, using @indicators to know which rows to extract. extracts data
%from @imageryStart time point to @imageryEnd time point. @numSamples - total samples.
...
...
...
end
```

- **eeg_visualization.mat:**

הפונקציה מקבלת נתונים גולמיים של @numSubplots (הוגדר 20) ניסויים אשר הוגרלו אקראית בקוד הראשי, ומציגה אותם בsubplots עבור כל ניסוי כך שבכל subplot מוצגת האמפליטודה של האות בכל אחת מהאלקטרודות לאורך זמן ה-visual imagery כפי שמוגדר ע"י @imageryEnd,@imageryStart, כאשר @bin הוא וקטור הזמנים עבור ציר ה-x.

```
function [] = eeg_visualization(handle, data, name, numSubplots, bin, fontSize, imageryStart, imageryEnd)

%EEG_VISUALIZATION visualizes the EEG signal in a single channel for trials from a single class.
%   draws a figure with @numSubplots for each class (left and right) corresponding to @numSubplots random trials.
%   Each subplot plots the data from both channels (C3 and C4).
%   @name - sgtitle name. @bin - time vector. @imageryStart and @imageryEnd - start and finish points for visual imagery.
...
...
end
```

- **calc_spectrum.mat:**

הפונקציה מקבלת את הנתונים הגולמיים כאשר הם כבר מחולקים לפי סוגי הניסוי (ימין/שמאל) ב@leftData ו-@rightData וקוראת לפונקציה הפנימית calc_pwelch_by_sides על מנת לקבל pwelch power spectrum עבור כל אחד מהצדדים.

```
function [pwelchLeft,pwelchRight] = calc_spectrum(leftData,rightData,numChannels,numTrials,frequencies,window,noverlap,fs)
%CALC_SPECTRUM calculates pwelch spectrogram using calc_pwelch_by_sides and
%splits the result into left data (@pwelchLeft) and right data
%(pwelchRight).
%   @leftData - data from "left" trials. @rightData - data from "right" trials. @numChannels - number of channels
%   @numTrials - number of trials. @frequencies - frequency vector. @window, @noverlap window and overlap sizes for pwelch
%   @fs - sampling rate.
pwelchResult = calc_pwelch_by_sides(leftData,rightData,numChannels,numTrials/2,frequencies,window,noverlap,fs);
pwelchLeft = squeeze(pwelchResult(1,:,:));
pwelchRight = squeeze(pwelchResult(2,:,:));

end
```

- **calc_pwelch_by_sides.mat:**

הפונקציה מקבלת את אותם המשתנים כמו calc_spectrum ומחשבת pwelch power spectrum על הנתונים ומחזירה אותו.

```
function [result] = calc_pwelch_by_sides(leftData,rightData,numChannels,numTrials,frequencies,window,noverlap,fs)
% CALC_PWELCH_BY_SIDES calculates the pwelch for each side, channel and trial and then
% averages the pwelch between trials to return a 3D matrix: 2x2xsize(frequencies)
%   @leftData - data from "left" trials. @rightData - data from "right" trials. @numChannels - number of channels
%   @numTrials - number of trials. @frequencies - frequency vector. @window, @noverlap window and overlap sizes for pwelch
%   @fs - sampling rate.
...
...
...
end
```

- **plot_spectrum.mat:**

מייצרת איורים של ספקטרוגרמות העוצמה של pwelch עבור כל אחת מהאלקטרודות וכן עבור חיסור בין הספקטרוגרמות של כל צד בכל אלקטרודה (סה"כ 4 subplots).

```
function [] = plot_spectrum(pwelchRight,pwelchLeft,frequencies,numChannels,chanNames)
%PLOT_SPECTRUM plots a specific channels power spectra in a specific
% method, depends on the call of the function. Both sides are plotted for
% each channel called for plotting.

...
...
...
sgtitle('Power spectra for each channel');

end
```

- **hotmap.mat:**

מקבלת את הנתונים הגולמיים מחולקים לניסויים של צד ימין (@rightDataAll) ושל צד שמאל (@leftDataAll) – לאורך כל הניסוי (לא רק שלב הvisual imagery). הפונקציה מייצרת heatmaps עבור כל צד וכל אלקטרודה (2x2) וכן עבור ההפרש בין הצדדים (בערך מוחלט) עבור כל אלקטרודה – סה"כ 6 subplots. הheatmaps הן של התדרים כפונקציה של הזמן ובעזרתן ניתן לראות את עוצמת האנרגיה בכל תדר לכל נקודת זמן עבור כל צד ואלקטרודה.

```
function [] = hotmap(rightDataAll,leftDataAll,numTrials,noverlap,fs)

%HOTMAP displays time(sec)/frequency(Hz) heatmaps for "right" and "left" trials seprately, as well
%as for the different electrodes and the difference between them.
%   @numTrials is the amount of trials. @noverlap is the overlap for
%   spectrogram function, @fs is the sampling rate.
%   uses 'spectrogram to get the spectral density for each trial and then
%   displays the average of the heatmaps in each subplot.
```

על מנת לייצר את מפות החום בכל subplot, הפעלנו על הנתונים מכל trial (בכל subplot מופיעים מחצית מהtrials) את פונקציית spectrogram של מטלב המבצעת Short-time fourier transform על האות שהיא מקבלת, עם מיצוע של חלון hamming (כברירת מחדל). Spectrogram מחזירה וקטור תדרים, וקטור זמנים ווקטור צפיפות ספקטרלית – הפרמטרים הנדרשים עבור מפת חום של אנרגיה.
לבסוף מיצענו את הפרמטרים (המפות שנוצרו) על פני מספר הtrials (64) והצגנו את התוצאות.

C3 heatmap

```
% C3 Right side
[~, rightFrequencyVector, rightTimeVector, rightSpectralDensity] =
spectrogram(rightDataAll(1,:,1),numTrials/2,noverlap,spectrogramFreqs,fs,'yaxis');
for i = 1:numTrials/2 - 1
  [~, frequencyVectorTemp, timeVectorTemp, spectralDensityTemp] =
spectrogram(rightDataAll(i+1,:,1),numTrials/2,noverlap,spectrogramFreqs,fs,'yaxis');
  rightFrequencyVector = rightFrequencyVector + frequencyVectorTemp;  %[Hz]
  rightTimeVector = rightTimeVector + timeVectorTemp;  %[sec]
  rightSpectralDensity = rightSpectralDensity + spectralDensityTemp;
end
rightFrequencyVector = rightFrequencyVector/(numTrials/2);  %[[Hz]
rightTimeVector = rightTimeVector/(numTrials/2);  % [sec]
rightSpectralDensity = rightSpectralDensity/(numTrials/2);

subplot(3,2,1)
imagesc(rightTimeVector,rightFrequencyVector,rightSpectralDensity);
ylabel('Frequency [Hz]');  xlabel('Time[sec]');
title('C3 - Right')
colormap hot;
...
...
%%%%% [More of the same code for all of the other subplots]
...
...
end
```

- **plot_histogram.mat:**

מייצרת היסטוגרמה של האנרגיה של התדרים של התכוניות שבחרנו על מנת להשוות בין trials של צד ימין וצד שמאל. משתמשת בbandpower על מנת לחלץ את העוצמות של התדרים הרלוונטיים מהנתונים הגולמיים.

```
function [] = plot_histogram(C3Bands,C4Bands,C3DiffC4Bands,rightData,leftData,fs)

%PLOT_HISTOGRAM plots energy histograms for @C3Bands and @C4Bands as well as @C3DiffC4Bands (C3-C4).
%   coloring @rightData and @leftData separately, to see the difference
%   between "left" trials and "right" trials. @fs is the sampling rate.
...
...
...
sgtitle('Energy histograms for each interesting band by channel');
end
```

- **cut_bands_by_times.mat:**

הפונקציה שולפת טווחי תדרים (@bands) בזמנים מסויימים (@times) מהנתונים (@data) על מנת להשתמש בהם כתכוניות בפונקציה get_features.

```
function [features] = cut_bands_by_times(data,bands,times,numTrials,fs)
%CUT_BANDS_BY_TIMES extracts frequency bands and times according to @bands
%and @times from @data.
...
...
...
end
```

- **get_features.mat:**

הפונקציה מקבלת את הנתונים הגולמיים (@data), טווחי תדרים
(@C3Bands,@C4Bands,@C3DiffBands) וטווחי זמנים עבור אותם תדרים
(@C3Times,@C4Times,@C3DiffBands) לאלקטרודות שלוש, ארבע, וההפרש ביניהן בערך מוחלט (בהתאמה).
הפונקציה מחלצת את התכוניות שתוארו בחלק השיטה של הדו"ח, כאשר החישובים עבור
Root total power, spectral moment, spectral entropy and Relative log power
נעשו על פי ההנחיות בעבודה 5 ומחזירה אותן במטריצה @features המכילה 32 תכוניות.

```
function [features] =
get_features(data,C3bands,C3Times,C4bands,C4Times,C3DiffC4Bands,C3DiffC4Times,imageryStart,imageryEnd,numTrials,window,
noverlap,frequencies,fs)

%GET_FEATURES extracts features from the data for classification for each electrode individually and for the difference between
electrodes.
%   bands and times for C3 and C4 electrodes are in @C3bands, @C4bands and @C3Times, C4Times
%   respectively. times and bands for the difference are in @C3DiffC4Times and @C3DiffC4Bands respectively.
%   @imageryStart and @imageryEnd indicate the
%   starting and ending times of visual imagery. @numTrials - number of
%   trials. @window, @noverlap, @fs are parameters for pwelch function (window
%   size, overlap size and sampling rate respectively). @frequencies is the
%   frequency vector.
...
...
...
...

features = [ featsC3 featsC4  freqC3 freqC4 featsC3log featsC4log featsC3diffC4sqrt featsC4sqrt featsC3sqrt rootTotalPowerC3
rootTotalPowerC4 spectralMomentC3 spectralMomentC4 spectralEntropyC3 spectralEntropyC4];

end
```

- **calc_pca.mat:**

פונקציית חישוב PCA שנלקחה מהעבודה הקודמת. מקבלת את מטריצת התכוניות @features מהפונקציה get_features ומחשבת עליה PCA על מנת להפחית מימדים. בתוצאה נשתמש עבור המסווג וכן בדיאגרמות פיזור.

```
function [X] = calc_pca(features,comNum,trialsNum)
%PCA manually calculate principal components analysis on @features matrix
%and returns the projection of the data on the top @comNum eigen vectors
...
...
end
```

- **custom_classifier.mat:**

הפונקציה מקבלת מטריצת תכוניות (@features) ומריצה k-fold cross validation באמצעות פונקציה של מטלב שנקראת cvpartition המבצעת חלוקה לfolds על פי @kfolds ומשתנה האינדיקציה @rightIndicator המסמן אילו ניסויים הם של צד ימין. בחרנו ב k=6 על מנת שתהיה שונות קטנה בין הסטים וגם שיהיו מספיק סטים.
custom_classifier מריצה את ה k-fold CV @runs פעמים וממצעת את התוצאה על מנת להפחית שונות בין ההרצות השונות ובכך להיפטר מרעשים. הפונקציה מחזירה את הניבוי ואחוזי הדיוק ב@prediction ו-@accuracy בהתאמה. בנוסף, הפונקציה מחשבת ומחזירה את אחוזי הדיוק וסטיית התקן של הk-folds CV באחד מהruns הן עבור הtraining (@trainFoldSTD,@trainFoldAcc) והן עבור הvalidation (@foldsSTD,@foldsAcc), נוסף על אחוז הדיוק על הtraining בכל הruns (@trainAcc).
לצורך אימון המודל והניבוי, השתמשנו בפונקציית classify של מטלב עם הפרמטר 'linear', על מנת שתבצע Linear Discriminant Analysis.

```
function [prediction,accuracy,foldsAcc,foldsSTD,stdRuns,trainFoldAcc,trainFoldSTD,trainAcc] =
custom_classifier(features,trialsNum,rightIndicator,runs,kfold,bias)

%CUSTOM_CLASSIFIER run @kfolds cross validation @runs times and saves the
%  average prediction in @prediction. also calculates @accuracy for the
%  average prediction and the standard deviation for the different runs accuracy(@stdRuns).
%  calculates accuracy and standard deviation for the
%  different folds of one run (@foldsAcc, @foldsSTD respectively).
```

Classification

```
y = rightIndicator;  % target variable
answers = NaN(trialsNum,runs+1);  % predictions for each trial and run
trainFoldAcc = zeros(kfold,1);  % accuracy for training for a specific run
trainAcc = zeros(runs,1);  % accuracy for training for all the runs

for run = 1:runs
   CVO = cvpartition(y,'k',kfold);  % creates a cross-validation partition for data.
   err = zeros(CVO.NumTestSets,1);  % hols the errors for each run
   for i = 1:kfold
      trIdx = CVO.training(i);  % training indices
      teIdx = CVO.test(i);  % test indices
      [ytest, trainFoldAcc(i)] = classify(features(teIdx,:),features(trIdx,:),y(trIdx),'linear');  % prediction for current fold
```

```
        % calculate error for each trial in the fold
        for j = 1:CVO.TestSize(i)
            foldAnsers = y(teIdx)';
            err(i) = err(i)+ int8((ytest(j)~= foldAnsers(j)));
        end
        answers(find(teIdx),run) = ytest+bias; % adds manual bias (if one was defined in @bias)
    end
    trainAcc(run) = mean(trainFoldAcc);
end

foldsAcc = err./CVO.TestSize'; % accuracy for folds in the last run
foldsSTD = std(foldsAcc); % standard deviation for folds in the last run
answers(:,runs+1) = ceil(mean(answers(:,1:runs)')); % average out the answers between runs. we chose to round up because our model is
slightlybiased to say more 'left' than 'right
```

Accuracy calculation

```
errors = zeros(trialsNum,runs+1);
for j = 1:runs+1
    for i = 1:trialsNum
        errors(i,j) =  int8((answers(i,j)~= y(i)));
    end
end

trainFoldSTD = std(trainFoldAcc);
stdRuns = std(sum(errors(:,1:runs))/trialsNum);
accuracy = 1 - sum(errors(:,runs+1))/trialsNum;
prediction = answers(:,runs+1);

end
```

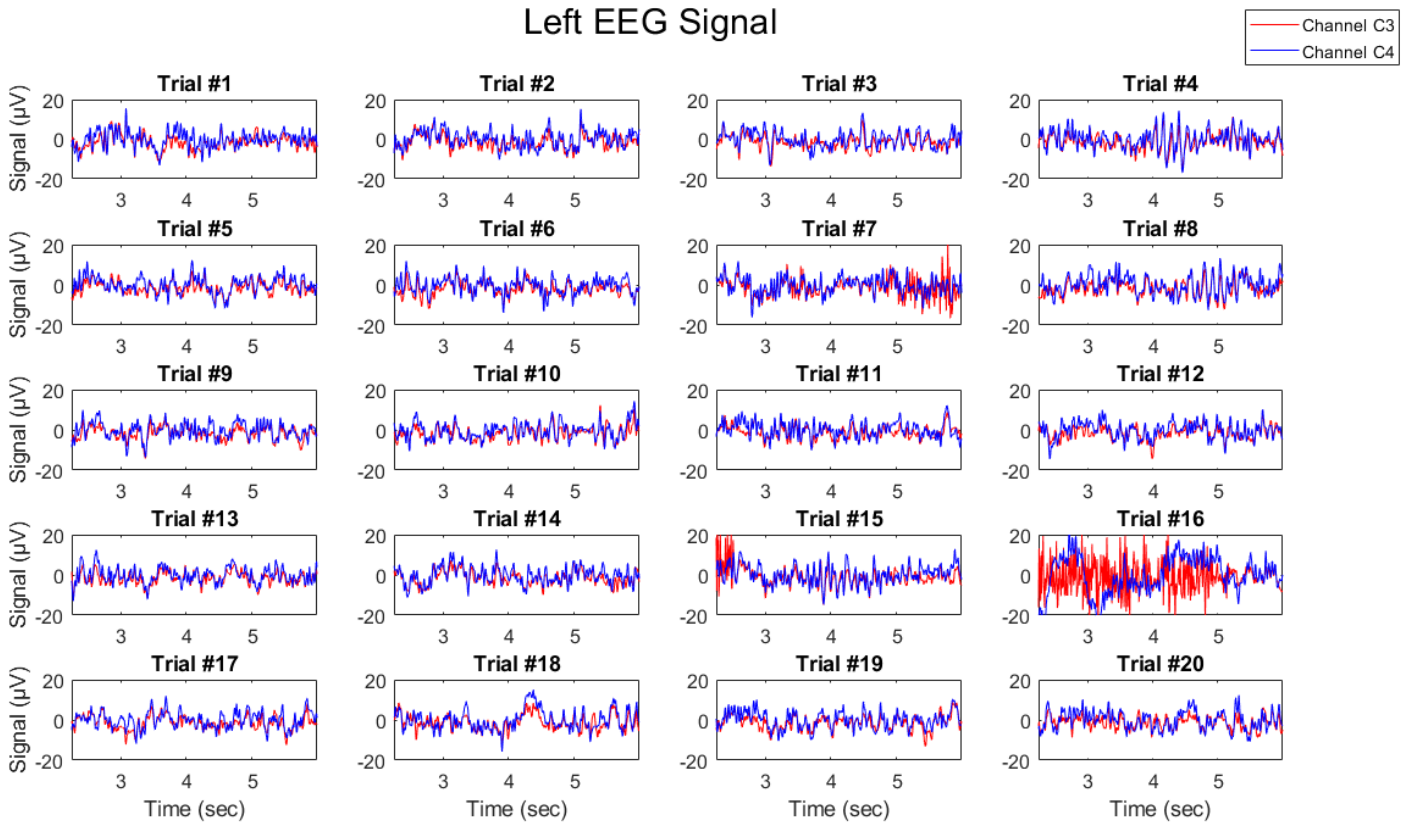
- **scatter_me.mat:**

פונקציה זו מייצרת שתי דיאגרמות פיזור, אחת בשני מימדים המציגה את ההיטלים של הנתונים על שני הוקטורים העצמיים הראשיים, ושנייה בשלושה מימדים המציגה את ההיטלים על שלושת הוקטורים העצמיים הראשיים עם חלוקה לצבעים של סוגי הtrials השונים.

```
function [] = scatter_me(PCs,rightIndicator,leftIndicator,fontSize)%t,patientID,seizureNum,PCs
%SCATTER_ME scatter plots the top 2 PCs and top 3 PCs in two subplots
%  @PCs = projected features data onto top PCs from PCA. @rightIndicator
%  and @leftIndicator are used to color the points in the scatter plot in
%  blue and red respectively
...
...
...
end
```

תוצאות ומסקנות:

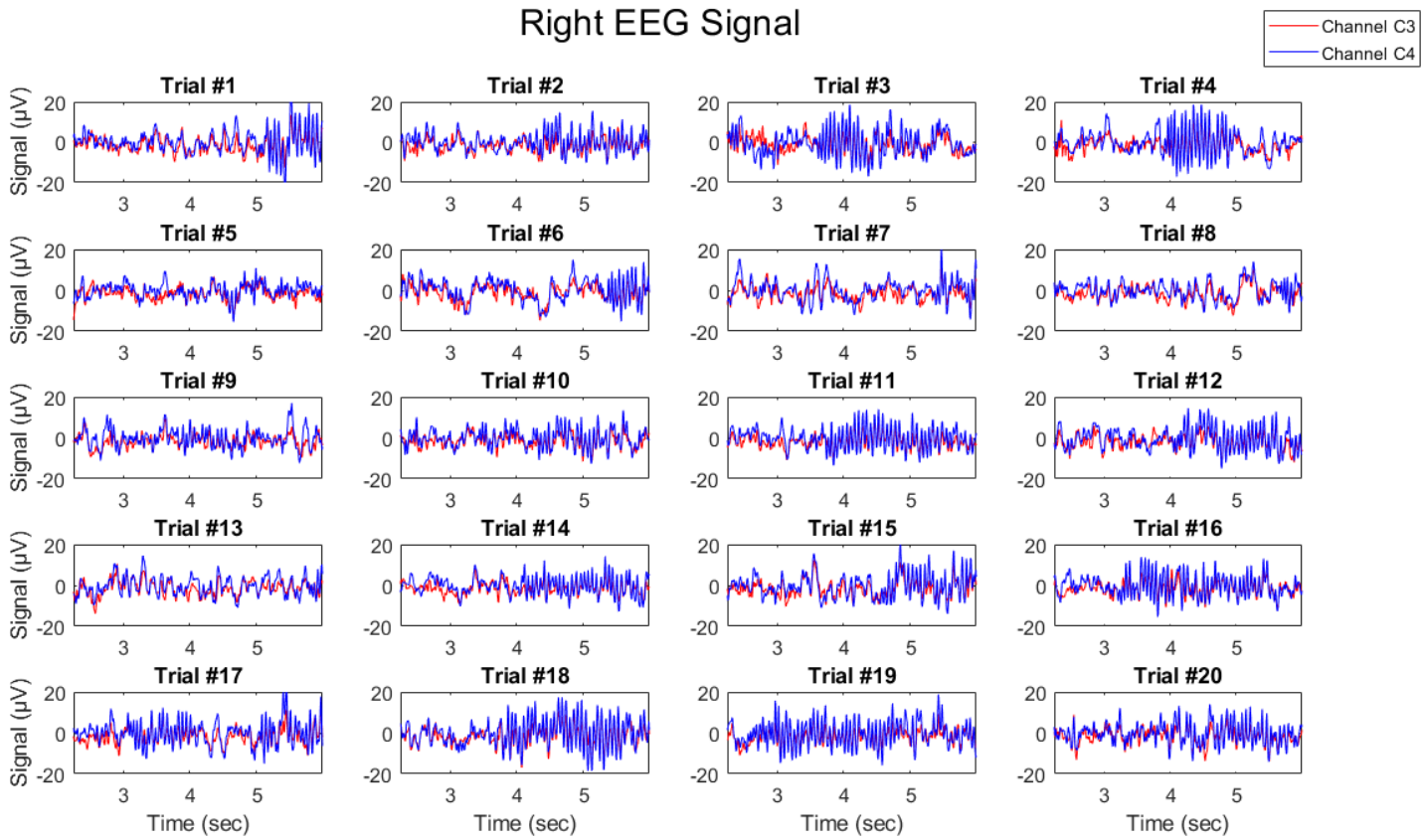
(מ-eeg_visualization.mat)
ויזואליזציה של הנתונים הגולמיים משתי האלקטרודות מ-20 ניסויים אקראיים עבור צד שמאל.



באדום – אות מאלקטרודה C3. בכחול – אות מאלקטרודה C4.
אם מבצעים תקריב משמעותי, ניתן לראות הבדלים יותר ברורים בין הגלים של שתי האלקטרודות, אך קשה להסיק מכך מסקנות אינפורמטיביות.

(מ-eeg_visualization.mat)

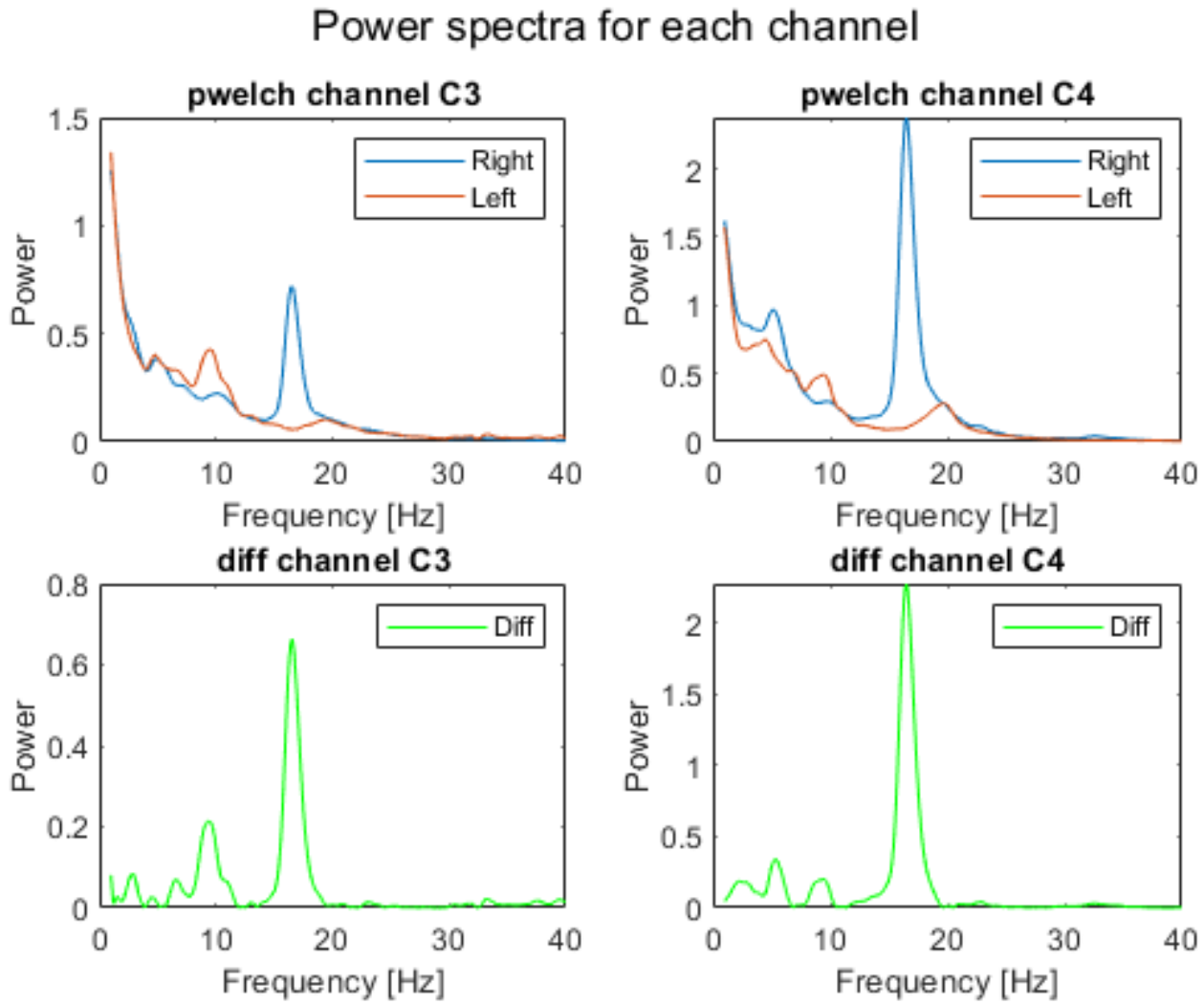
ויזואליזציה של הנתונים הגולמיים משתי האלקטרודות מ-20 ניסויים אקראיים עבור צד ימין.



באדום – אות מאלקטרודה C3. בכחול – אות מאלקטרודה C4.
נראה שיש שונות גבוהה יותר באלקטרודה C4 עבור צד ימין לעומת צד שמאל וכן לעומת אלקטרודה C3, דבר המעיד על כך שאכן יש הבדל בין האלקטרודות ולכן עשוי להועיל לבחון תכוניות על ההפרש בין האלקטרודות (כפי שעשינו). נוסף על כך, הדבר מעיד על הבדלים באות בין הצדדים ועל כן סביר שנצליח לבצע הפרדה.

(plot_spectrum.mat-מ)

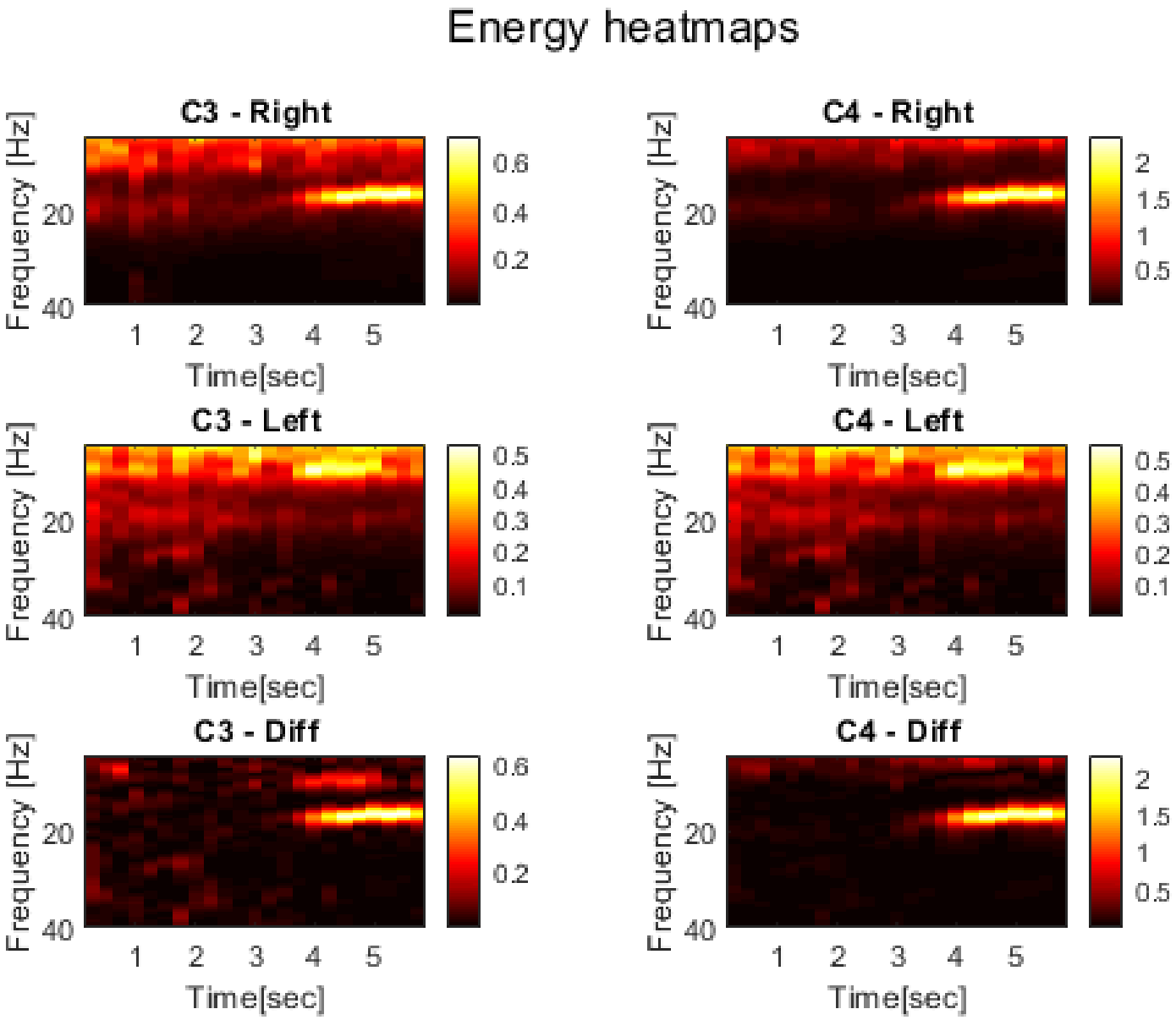
Power spectra using Pwelch's method



בכחול – נתונים מניסויים של צד ימין. באדום – ניסיון של צד שמאל. בירוק – הפרש בין הצדדים בערך מוחלט.
שני האיורים הימניים הם עבור אלקטרודה C4 ושני השמאליים הם עבור אלקטרודה C3.
ניתן לראות שוני משמעותי בספקטרום בין הצדדים. השוני בולט במיוחד בטווח התדרים של 15-17 הרץ בערך, ו7-11 הרץ בערך, אך קשה כאן לדייק את הטווחים, ולכן השתמשנו בheatmaps בהמשך. היות ש power spectrum אינו מציג את מימד הזמן, איננו יכולים לדעת באיזו נקודת זמן בדיוק כל טווח תדרים מתחיל ומפסיק להשפיע. זו סיבה נוספת לשימוש בheat maps אשר כן מאפשרות להתייחס למימד זה.

(hotmap.mat-מ)

Energy heatmaps



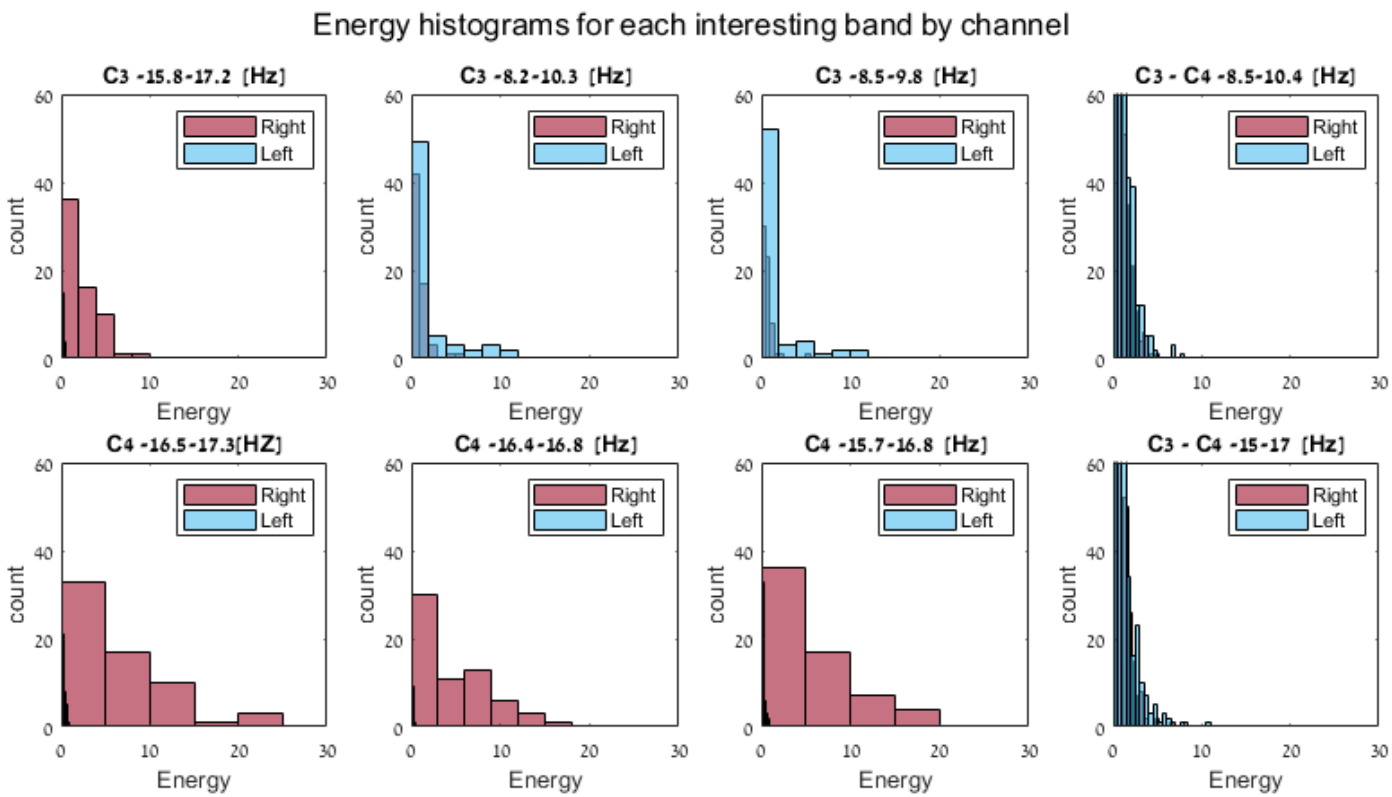
ככל שהצבע מתבהר ככה אמפליטודת התדר בזמן נתון גבוהה יותר.

בהפקת גרף זה השתמשנו בכל זמן הניסוי ולא רק בחלק בו הנבדק מתבקש לדמיין תנועת גפיים. הסיבה לשימוש בכל זמן הניסוי הוא בשל הבקשה למצוא תכוניות שמבדילות בין ימין/שמאל ללא קשר לסיבת ההבדלה. יכול להיות שיש מאפיינים בניסוי בזמנים שהם טרם תהליך הדמיון שגורמים לפעילות מוחית שיכולה לנבא בצורה טובה באיזה ניסוי התבקשו לדמיין הפעלת גפה בכל צד.

ראשית, ניתן לראות ממפות החום שהתוצאות תואמות למה שקיבלנו ב channel Power spectra to each – פעילות חזקה במיוחד סביב 15-17 הרץ וכן סביב 7-11 הרץ. שנית, באמצעות הגדלה משמעותית (zoom in) ניתן לזהות ברזולוציה גבוהה טווחי זמן ותדרים עבור האלקטרודות השונות שיכולים להוות הפרדה טובה בין שני סוגי הניסוי (ניתן לראות איזה טווחים זוהו בגרף זה ושמשו ליצירת המודל בחלק השיטה).

(מ-plot_histogram.mat)

Energy histogram for each band by channel

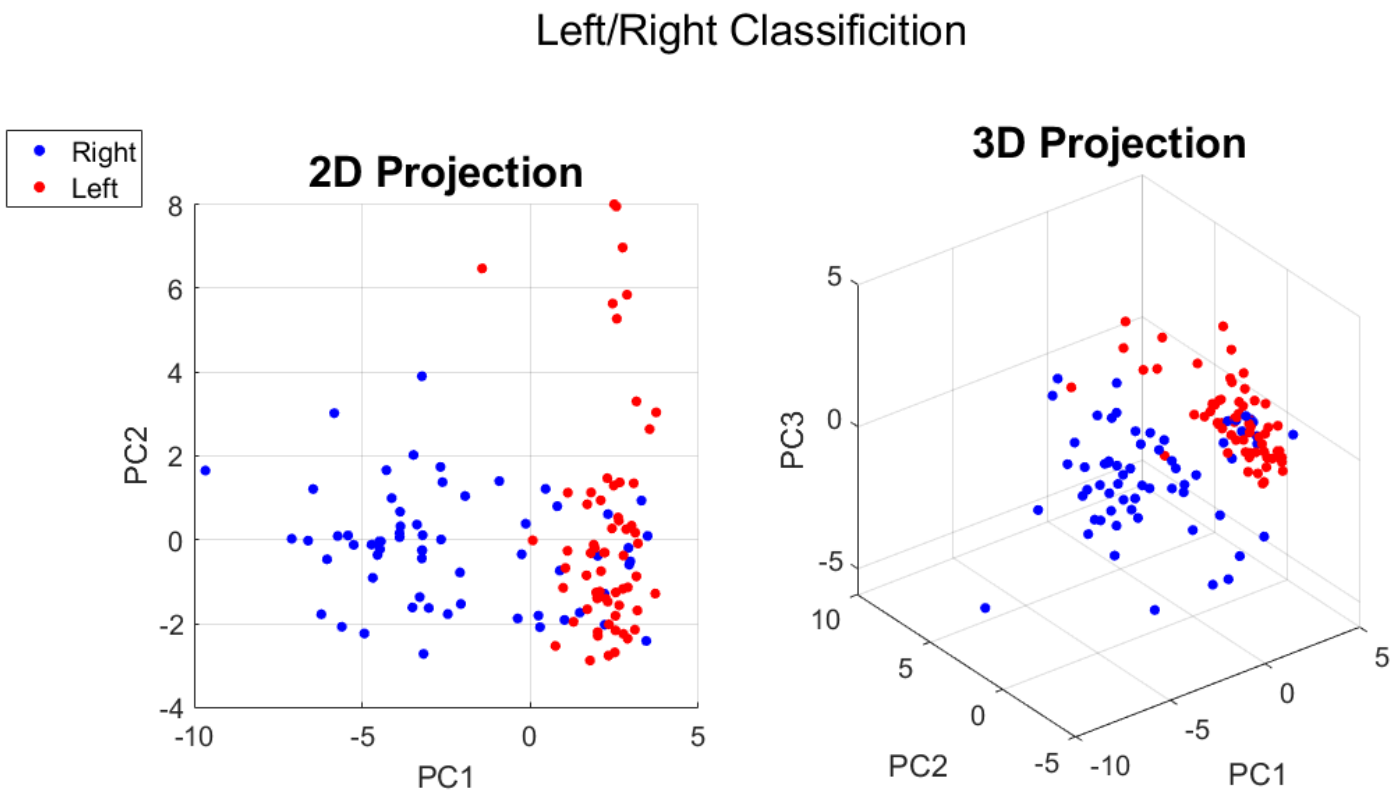


באדום – נתונים מניסויים של צד ימין. בכחול – ניסיון של צד שמאל.

על מנת לבחון האם טווחי התדרים שמצאנו עבור כל אחת מהאלקטרודות יכולים ליצור הפרדה טובה בין ניסויי ימין לניסויי שמאל השתמשנו בהיסטוגרמות. ניתן לראות שהגרפים של טווחי התדרים של אלקטרודות C3 ו-C4 יוצרות הפרדה ברורה בין צד ימין לשמאל. לעומת זאת לפי ההיסטוגרמה של הפרשי האלקטרודות לא ניתן לראות באופן מובהק שיש הפרדה בין צד ימין לשמאל. עם זאת, ייתכן שיחד עם תכוניות נוספות, התכוניות האלו כן יתרמו לניבוי, ובכל אופן המסווג וה-PCA ייתנו לתכוניות 'משקל' בהתאם למידת התרומה שלהן לניבוי.

(scatter_me.mat-מ)

PCA – 2D & 3D visualization



בכחול – נתונים מניסויים של צד ימין. באדום – ניסויים של צד שמאל.

ניתן לראות בגרפים הטלה של הנתונים עבור כל צד על 2/3 הווקטורים העצמיים השייכים לערכים העצמיים הגדולים ביותר. ממבט על הגרפים אפשר לראות שניתן להפריד בצורה יחסית טובה בין ניסויי ימין לבין ניסויי שמאל בעזרת קו לינארי, כך שההפרדה יותר טובה עם 3 PCs.

תוצאות הליך הסיווג

```
## Complete model results (validation): ##
Run #5 Accuracy (training): 0.89847%   SD: 0.0070364
Run #5 Accuracy (validation): 0.88203%   SD: 0.050997
Whole model accuracy (training): 90.0646%   SD: 0.0021075
Whole model accuracy (validation): 89.0625%   SD: 0.0065364


## Example with 1 feature: ##
Run #5 Accuracy (training): 0.82815%   SD: 0.014908
Run #5 Accuracy (validation): 0.82828%   SD: 0.075064
1-feature model accuracy (training): 82.91%   SD: 0.00082935
1-feature model accuracy (validation): 83.5938%   SD: 0.0042791
```

כאמור, הרצנו את הk-fold CV מספר פעמים (5) על מנת למצע את התוצאה ולנקות רעשים. הצגנו כאן הן תוצאות של הרצה אחת של הCV על הtraining set (שורות 1-2) והן את התוצאות של הממוצע של ההרצות (שורות 3-4).
ניתן לראות מספר דברים. ראשית, כצפוי, אחוזי דיוק האימון גבוה מאחוז דיוק הvalidation (בשני המקרים), שכן התיוגים (ימין/שמאל) 'שקופים' למודל בזמן האימון. שנית, ניתן לראות כי המודל השלם (כל ההרצות) הינו בעל סטיות תקן קטנות יותר מאשר המודל של הרצה אחת, כך שנראה שהרעיון של ניקוי הרעשים עובד. שלישית, אחוזי הדיוק של המודל השלם מעט גבוהים יותר מאחוזי הדיוק של המודל הרצה אחת.

נוסף על התוצאות לעיל, צירפנו גם תוצאות באותו פורמט, אך הפעם עבור שימוש בתכונית אחת (לעומת המודל ה"רגיל" בו יש 8 תכוניות לאחר הPCA). קל לראות שמודל בעל תכונית אחת הינו משמעותית פחות טוב בניבוי. תוצאה זו אינה מפתיעה שכן האות הגולמי של EEG רועש ומדובר בתופעה מורכבת, שלא היינו מצפים לנבא בכזאת קלות.

אחוז דיוק על סט המבחן במודל

ציונים / 100.00

93.75

כשביצענו אימות באתר המודל עבור מידת הדיוק של המודל שלנו על motor_imagery_test_data.mat יצא שאחוז הדיוק של הניבוי שלנו הוא 93.75%. נשים לב שציון זה גבוה בכ3-4% מהדיוק שהגענו אליו על סט האימון. הסבר אחד לתופעה זו היא שסט המבחן מכיל ניסויים "קלים" יותר לניבוי, כלומר שהם יותר מובחנים אחד משהני. הסבר שני הוא שהיה לנו מזל והמודל שלנו מתאים יותר להתפלגות של סט המבחן. אמנם לא ציפינו לתוצאה כזו, אך היא אינה מאוד מאוד מפתיעה כיוון שמדובר בסט עם מעט דגימות, ולכן סביר שתהיה ביניהן שונות גדולה יחסית, שיכולה להוביל לכל מיני תוצאות "מוזרות".
אפשר גם להסיק מתוצאות אלו שהמודל שיצרנו לא עשה overfitting על סט האימון.