TECHNISCHE
UNIVERSITÄT
DARMSTADT

Clinical Applications of Brain Imaging, Stimulation, and Modeling

Exercise 7

Talha Bhatti (2667250)

Contents

List of Figures

1 Questions

1.1 What is the "placebo effect" and how is it relevant for clinical trials? (6 points)

- It refers to improvements in symptoms simply due to the belief that one is receiving treatment even when no active therapeutic intervention is provided.
- It is relevant for clinical trials because it can bias results, making it difficult to determine the true efficacy of a treatment.
- To control for the placebo effect, clinical trials often include a sham treatment group.
- In brain stimulation studies, placebo TMS involves adjusting coil positioning or shielding magnetic fields to mimic side effects without actual stimulation.

1.2 What are typical indications that your machine learning algorithm has been either over- or underfitted? What can you do in these cases to improve the algorithm? (8 points)

Overfitting: High accuracy on training data but poor performance on validation/test data. Using more diverse training data and techniques like data augmentation, regularization, dropout help overcome overfitting problems.

Underfitting: Poor performance on both training and validation/test data. It can be overcome by increasing model complexity, optimizing hyperparameters and providing better feature selection.

In general, cross-validation can help optimize model performance and stratified splitting ensures balanced distribution of classes across training and test sets.

1.3 Suppose you have recorded neuroimaging data from an ADHD population. Symptoms in your patients have been assessed with the ASRS scale. Describe which approach you would choose to investigate this data with machine learning. (6 points)

- **Step 1:** Choose an appropriate machine learning approach. Supervised learning for classification or regression, if labelled is available. Unsupervised learning for clustering patients based on brain activity patterns.
- **Step 2:** Feature extraction from neuroimaging data
- **Step 3:** Split data into training, validation and test sets ensuring proper stratification during the data preprocessing part. Perform feature engineering if relevant
- **Step 4:** Perform model training
- **Step 5:** Evaluate model using metrics such as AUC, accuracy, precision, recall etc.

1.4 From a patient health perspective, would it be preferable if your machine learning algorithm has a larger FN rate or a larger FP rate? (5 points)

Higher False Negatives: A serious disease might go undetected leading to missed treatment. This is dangerous in critical conditions like epilepsy or stroke detection.

Higher false positives: Patients may receive unnecessary further testing or treatments but the risk of missing a diagnosis is reduced.

In clinical applications, reducing false negatives is usually more important so we want a lower FN rate, while large FP rate can be somewhat compromised on.

1.5 Suppose you train a classifier on 3 groups of similar size in your patient population and a classifier that achieves 52% accuracy on the test set. Is this result larger than expected by chance? Is it clinically useful? (5 points)

- If the data has equal sized group, even random guessing could achieve 33% accuracy.
- So, 52% acuracy seems too low to have a clinical relevance. Although it may depend on the baseline performance.
- Depending on the application, accuracy alone may not alone jusitfy the model's performance and metrices like Precision and Recall could be important. In my experience, Area under ROC curve is generally a good indicator of a classifier's performance.

1.6 What are the advantages and disadvantages of circular coils, figure 8 coils, and H-coils for TMS? (6 points)

- **Circular coils:** Simple design and can simulate a broad area. But they offer low spatial precision.
- **Figure-8 coils:** More focal stimulation, preferred for targeting specific brain regions. But limited depth penetration.
- **H-coils:** Can stimulate deeper brain structures. But less focal and higher stimulation spread.

1.7 Which problems can occur when using purely random splitting of subjects into training, validation, and test sets? (4 points)

- Class imbalance can be one issue from random splitting.
- Data leakage can be a problem where training data is leaked into val and test sets.

- Data sparsity could be a problem where model is trained on only a few patient's data if we don't take patient ID into account (a good approach I learned from a recent project is to group data with same distribution and discard samples within the group when undersampling or splitting).

1.8 Suppose you want to analyze TMS-evoked potentials and TMS-induced responses across multiple trials; what would you need to consider? How would your analysis differ between the one and the other? (6 points)

TMS-evoked potentials (TEP):

- Phase locked to TMS stimulus
- Typically analyzed by averaging across trials to enhance SNR

TMS-induced responses:

- Not phase locked, show later effects on oscillatory activity
- Require time-frequency analysis instead of simple trial averaging

1.9 Which forms of brain/nervous system stimulation are used to aid with stroke rehabilitation? (4 points)

- **Transcranial Magnetic Stimulation (TMS):** Can stimulate the motor cortex to aid in motor recovery
- **Transcranial Direct Current Stimulation (tDCS):** Uses weak electrical currents to modulate cortical excitability and improve plasticity
- **Vagus Nerve Stimulation (VNS):** May support motor rehabilitation when combined with therapy.

2 Python Exercises

2.1 Task 1

```
#%% Import modules

import os
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
```

```
from scipy.stats import f_oneway
from scipy.cluster.hierarchy import dendrogram, linkage

from ex7_functions import make_spider



mpl.style.use('default')




#%% Define frequencies

freqs = np.array([ 2.15,  2.49,  2.88,  3.31,  3.73,
                   4.15,  4.75,  5.35,  5.93,  6.62,
                   7.39,  8.14,  9.02,  9.83,  10.92,
                   11.89, 13.11, 14.78, 16.3,  17.8 ,
                   19.7 , 21.6 , 23.73, 26.55, 28.7 ,
                   31.8 , 34.5 , 37.9 , 42.5 , 46.9 ,
                   52.1 , 59.3 , 65.3 , 71.  , 78.  ,
                   85.92, 95.6 ])

freq_strings     = ['{:.2f}'.format(f) for f in freqs]

NF = len(freqs)



#%% Task 1: Load data and evaluate symptom correlations (10 points)

"""
In this exercise, we will look at a (artificial) clinical dataset.
We will look at the symptoms and then perform some unsupervised learning,
    namely clustering, based on clinical symptoms.
For these 262 patients, 8 different symptom types were measured with
    questionaires.

Load the symptoms from 'symptoms.csv'.

Compute the pairwise cross-correlation between each 2 of the symptoms.
Plot as a heatmap, with yellow-to-red colors, diagonal starting from bottom
    left,
  and add symptom abbeviations/names as ticklabels, and a colorbar with range
    0 to 1.
Which two symptoms have the highest cross-correlation?

"""

symptom_type = np.array(['Depression','Anxiety','Rumination','Functional
    impairment',
                         'Experiental Avoidance','PTSD','Anhedonia','Substance
     abuse'])
```

```
DATA_DIR = 'data'
SYMPTOMS_DIR = os.path.join(DATA_DIR, 'symptoms.csv')
DFA_DIR = os.path.join(DATA_DIR, 'dfa_262_subj.npy')

symptoms_df = pd.read_csv(SYMPTOMS_DIR, sep=';')
dfa_values = np.load(DFA_DIR)

# Compute pairwise correlation
corr_matrix = symptoms_df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='YlOrRd', vmin=0, vmax=1,
    xticklabels=symptoms_df.columns, yticklabels=symptoms_df.columns)

plt.title('Pairwise Cross-Correlation Heatmap of Symptoms')
plt.show()

# Identify the symptoms with the highest cross correlation
corr_unstacked = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).
    astype(bool)).unstack()
max_corr = corr_unstacked.idxmax()
max_corr_value = corr_unstacked.max()
print(f"The two symptoms with the highest cross-correlation {max_corr_value:.3
    f} are {max_corr[0]} and {max_corr[1]}.")
```

```
The two symptoms with the highest cross-correlation 0.545 are PCL and GAD.
```

2.2 Task 2

```
#%% Task 2: Perform hierarchical clustering (10 points)


"""
Apply z-scoring to the symptom data for each of the symptoms.
Create a linkage metric with linkage() from the z-scored symptom data,
    using the 'ward' method and 'euclidean' metric.
Use dendrogram() to plot a dendrogram of the subjects.

Decide what seems like a good number of clusters based on the dendrogram
   (not too low, not too high).
Plot the dendrogram again and select the color threshold to get the desired
    number of clusters.

Use the output from the dendrogram function to assign each subject to a
    cluster.
Alternatively, you can run the method AgglomerativeClustering (with same
    parameters).

"""

scaler = StandardScaler()
```

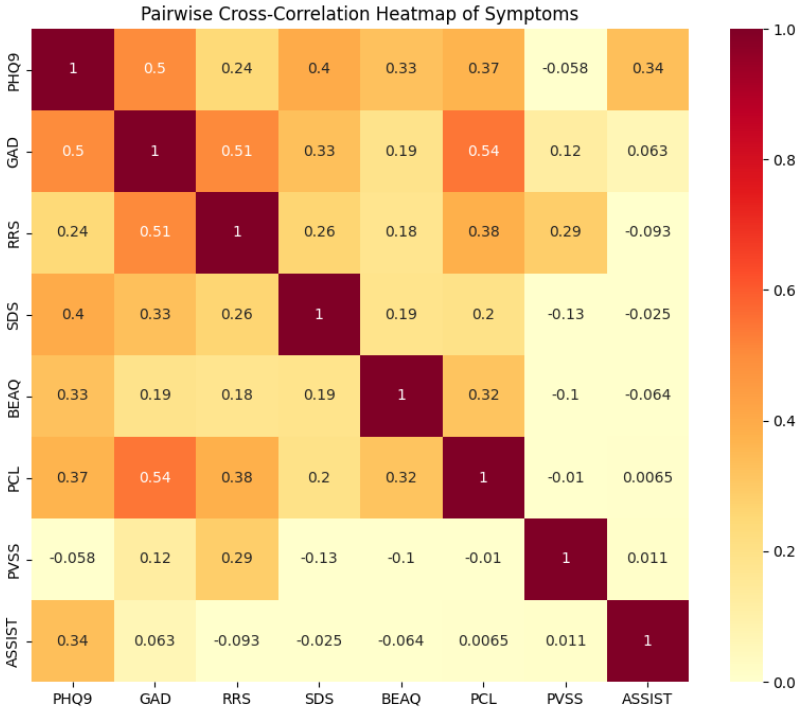

Figure 1: Pairwise Cross-Correlation Heatmap of Symptoms

```
z_scores = scaler.fit_transform(symptoms_df)
linkage_matrix = linkage(z_scores, method='ward', metric='euclidean')


color_threshold = 15

plt.figure(figsize=(12, 8))
dendrogram(linkage_matrix, color_threshold=color_threshold)
plt.title('Dendrogram of Subjects')
plt.xlabel('Subjects')
plt.ylabel('Distance')
plt.show()

n_clusters = 4
clustering = AgglomerativeClustering(n_clusters=n_clusters, metric='euclidean'
    , linkage='ward')
cluster_labels = clustering.fit_predict(z_scores)
print("Cluster assignments for the first 10 subjects:", cluster_labels[:10])
    ment artefact, ICA001 for subject 1 and 3, ICA002 for subject 2 are
    removed. Subject 4 does not show prominent eye movement')
```

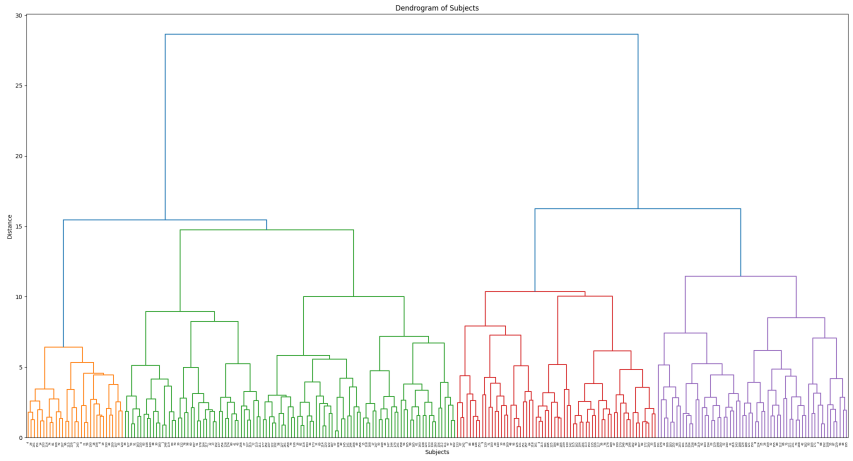

Figure 2: Dendrogram of Subjects

```
Cluster assignments for the first 10 subjects: [3 0 1 2 3 2 3 3 1 2]
```

2.3 Task 3

```
#%% Task 3: Make spider plots (6 points + 5 bonus points)

"""
Next, we'll make some spider plots to visualize how the symptom profiles in
    the
   individual clusters look relative to the cohort mean.
```

```
For each cluster and symptom, get the mean z_score over all subjects in the
    cluster.
For each cluster, plot the mean values in a spider plot.

Try to use the same (or similar) color for each cluster as was used in the
    dendrogram.
Use the cluster's index plus its number of subjects in the title for its plot.

For bonus points, describe each cluster's clinical symptom profile
   relative to the cohort mean (1-2 sentences each).

"""

cluster_means = pd.DataFrame(z_scores, columns=symptoms_df.columns)
cluster_means['Cluster'] = cluster_labels

cluster_colors = ['orange', 'green', 'red', 'purple']

# Create spider plots for each cluster
for cluster_id in range(n_clusters):
    cluster_data = cluster_means[cluster_means['Cluster'] == cluster_id]
    mean_values = cluster_data[symptoms_df.columns].mean()
    make_spider(mean_values, symptoms_df.columns, cluster_colors[cluster_id],
                f'Cluster {cluster_id+1} (n={len(cluster_data)})')
plt.show()

cluster_descriptions = []
for cluster_id in range(n_clusters):
    cluster_data = cluster_means[cluster_means['Cluster'] == cluster_id]
    mean_values = cluster_data[symptoms_df.columns].mean()
    description = f"Cluster {cluster_id}: Shows higher levels of "
    # Identify symptoms significantly above or below the cohort mean
    above_mean = mean_values[mean_values > 0.5].index.tolist()
    below_mean = mean_values[mean_values < -0.5].index.tolist()
    if above_mean:
        description += f"elevated {', '.join(above_mean)}"
    if below_mean:
        if above_mean:
            description += " and "
        description += f"reduced {', '.join(below_mean)}"
    if not above_mean and not below_mean:
        description += "symptoms close to the cohort mean."
    else:
        description += " compared to the cohort mean."
    cluster_descriptions.append(description)

print(cluster_descriptions)
```

Figure 3: Spider plots of mean subject z-scores in clusters

Cluster 0: Shows higher levels of reduced GAD, RRS compared to the cohort mean.
Cluster 1: Shows higher levels of elevated PHQ9, GAD, RRS, SDS, BEAQ, PCL compared to the cohort mean.
Cluster 2: Shows higher levels of elevated GAD, RRS and reduced ASSIST compared to the cohort mean.
Cluster 3: Shows higher levels of reduced PHQ9, GAD, RRS, SDS, BEAQ, PCL compared to the cohort mean.

2.4 Task 4

```
#%% Task 4: Plot DFA exponents for clusters (8 points + 3 bonus points)

"""
Now, assume that DFA exponents have been estimated in source-reconstructed MEG
     data
  for our patients, and we want to see if our clusters differ in their
  spectral distribution of DFA values.

Load the DFA values from dfa_262_subj.npy file.
Compute for each cluster and frequency the mean over its subjects and all 200
    parcels.
In one figure, plot the mean DFA exponent as a function of frequency for each
    cluster.

For bonus points: What do you think the results indicate about the different
    clusters?

"""

print(dfa_values.shape)
```

```
mean_dfa_per_cluster = {}

for cluster_id in range(n_clusters):
    subject_indices = np.where(cluster_labels == cluster_id)[0]
    # Average over subjects and parcels
    mean_dfa = dfa_values[subject_indices].mean(axis=(0, 2))
    mean_dfa_per_cluster[cluster_id] = mean_dfa

plt.figure(figsize=(12, 8))

for cluster_id, mean_dfa in mean_dfa_per_cluster.items():
    plt.plot(freqs, mean_dfa, label=f'Cluster {cluster_id}', linewidth=2)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Mean DFA Exponent')
plt.title('Mean DFA Exponent as a Function of Frequency for Each Cluster')
plt.legend()
plt.grid(True)
plt.show()
```

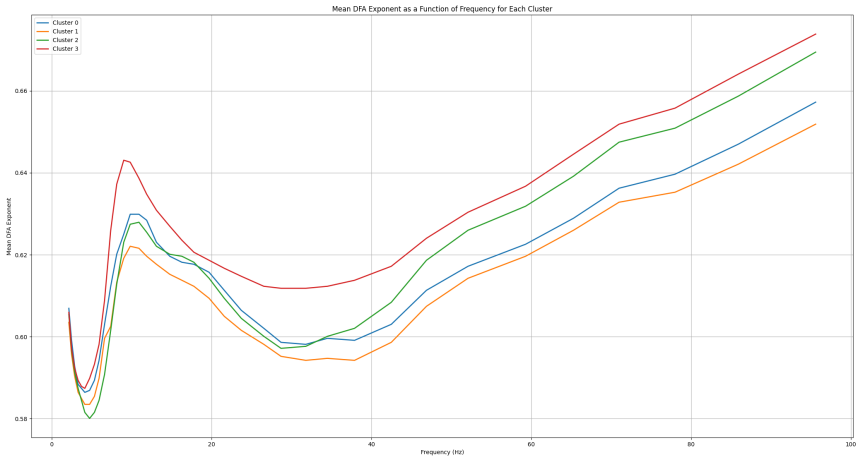

Figure 4: Mean DFA Exponent as a Function of Frequency for Each Cluster

2.5 Task 5

```
#%% Task 5: Compare DFA between all clusters (8 points)

"""
Now let's see if our clusters have significant differences in DFA exponents.

Compute the mean DFA (over parcels) per subject and frequency.
Choose a statistical test suited for comparing multiple groups and compute
  the test-statistic and p-value for each frequency.
Is the result statistically significant for any frequencies?
Is the result what you expected? Why, or why not?
```

```
"""

mean_dfa_per_subject = dfa_values.mean(axis=2)

# Perform ANOVA for each frequency to compare clusters
p_values = []
f_statistics = []
alpha = 0.05

for freq_idx in range(mean_dfa_per_subject.shape[1]):
    cluster_data = [mean_dfa_per_subject[cluster_labels == cluster_id,
    freq_idx] for cluster_id in range(n_clusters)]
    f_stat, p_val = f_oneway(*cluster_data)
    f_statistics.append(f_stat)
    p_values.append(p_val)

significant_freqs = [(freqs[i], p_values[i]) for i in range(len(p_values)) if
    p_values[i] < alpha]

print(significant_freqs)
print(len(significant_freqs))
```

There are no frequencies at which the DFA exponents differ significantly among the clusters
This could be because the symptom based clustering may not strongly correlate with global neural DFA properties.

2.6 Task 6

```
#%% Task 6: Compare DFA between all clusters for each parcel (8 points + 3
    bonus points)

"""
Now, do the statistical test again, but for each frequency and for each parcel
    .
At which frequency do we have the highest number of significant parcels?

For bonus points: Let's think about multiple comparisons.
    Is the number of significant results higher than would be expected by
    chance?
"""

n_subjects, n_freqs, n_parcels = dfa_values.shape

pvals = np.zeros((n_freqs, n_parcels))

for i_freq in range(n_freqs):
    for i_parcel in range(n_parcels):
        data_by_cluster = [
            dfa_values[cluster_labels == c, i_freq, i_parcel]
            for c in range(n_clusters)
```

```
        ]
        # Perform one way ANOVA
        _, p_value = f_oneway(*data_by_cluster)
        pvals[i_freq, i_parcel] = p_value

num_significant = np.sum(pvals < alpha, axis=1)

# Identify which frequency has the largest number of significant parcels
best_freq_idx = np.argmax(num_significant)
max_freq = freqs[best_freq_idx]
max_significant_count = num_significant[best_freq_idx]

#print("Number of significant parcels (p < 0.05) at each frequency:")
#for i_freq in range(n_freqs):
#    print(f"  Frequency {freqs[i_freq]:.2f} Hz: {num_significant[i_freq]}
    significant parcels.")

print(f'Highest number of significant parcels are {max_significant_count} at {
    max_freq} Hz')

expected_by_chance = alpha * n_parcels  # for a single frequency
print(f'By chance alone, we\'d expect reoughly {expected_by_chance:.1f}
    parcels to be significant at p < {alpha}')
if max_significant_count > expected_by_chance:
    print("We have more significant parcels than expected by chance at that
    frequency.")
else:
    print("We do not exceed the rough expectation of false positives at that
    frequency.")


#%%
```

```
Highest number of significant parcels are 23 at 7.39 Hz
By chance alone, we'd expect reoughly 10.0 parcels to be significant at p < 0.05.
We have more significant parcels than expected by chance at that frequency.
```