1928

K. N. Toosi University of Technology

Faculty of Electrical Engineering

Course: Functional Brain Imaging Systems

Professor: Dr. Ali Khadem

Computer assignment 5

Topic:

Diagnosing Parkinson’s Disease Using EEG Band Power Features

Prepared By:

Mohammadreza Shahsavari, Ramin Tavakoli, Mahdi Molaii

Overview

This homework focuses on classifying subjects with Parkinson's Disease (PD) and Healthy Controls (HC) using EEG data. You will compute power band features from the EEG data and use these features as input for machine learning classifiers.

You are required to complete this homework using Google Colab environment.

Objectives

- Download and prepare EEG data.
- Calculate **EEG band powers** as features.
- Prepare the dataset with these features and their corresponding labels.
- Train an machine learning classifier using the extracted features.
- Evaluate the classifier's performance using metrics such as **accuracy**, **precision**, and **recall**.

Submission Instructions

Google Colab Notebook:

1. All tasks must be implemented in a Google Colab notebook.
2. Use the provided code placeholders and follow the step-by-step instructions to complete the assignment.
3. Submit your completed Google Colab notebook as a `.ipynb` file.

Step 1: Download and Extract the Dataset

Start by downloading the provided EEG dataset, which is a subset of the publicly available UC San Diego EEG dataset for Parkinson's Disease patients. The data provided to you is a small portion of the original dataset that has already been preprocessed and segmented into 2-second EEG intervals. This preprocessing was done as part of one of our previous studies, titled "A Hybrid Deep Spatiotemporal Attention-Based Model for Parkinson's Disease Diagnosis Using Resting-State EEG Signals."

```
# Dowloading the EDF data
import gdown
import os
import zipfile

eeg_arch_url = 'https://drive.google.com/uc?id=1rynIhpIAQc4OzX4nCrrnxOXMsdLt7xWD'

eeg_arch_file_name = 'EEGs.zip'  # Assuming the file is now a ZIP file

gdown.download(eeg_arch_url, eeg_arch_file_name, quiet=False)

eeg_arch_path = os.path.join('/content', eeg_arch_file_name)

# Unzip the file
with zipfile.ZipFile(eeg_arch_path, 'r') as zip_ref:
    zip_ref.extractall()

!ls -l /content
```

By running the above cell, the dataset will be downloaded as a ZIP file and extracted into your Colab's `/content` directory under the name `Preprocessed UC San Diego Dataset`. This folder contains two subfolders:

$$\text{Preprocessed Uc San Diego Dataset} \rightarrow \begin{cases} PD \text{ (Parkinson's Disease EEG Segments)} \rightarrow .mat \text{ files} \\ HC \text{ (Healthy Control EEG Segments)} \rightarrow .mat \text{ files} \end{cases}$$

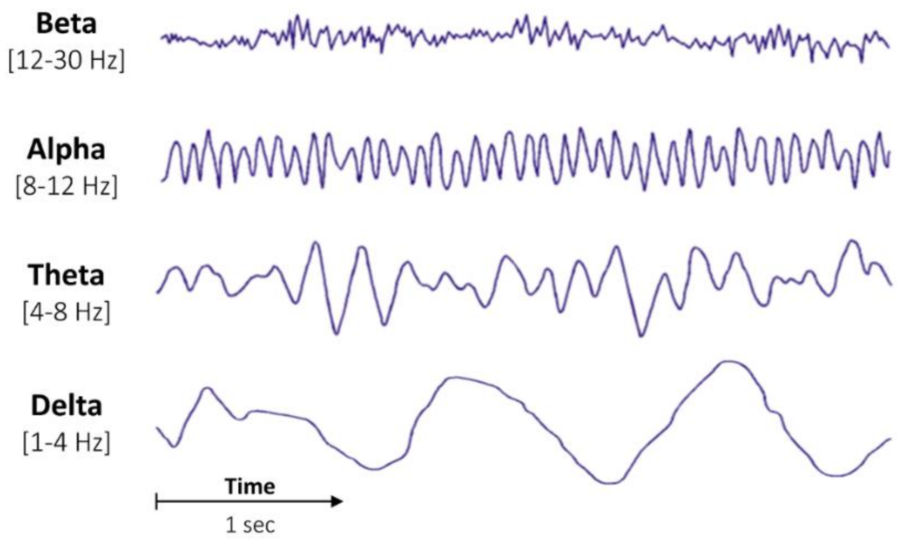
Step 2: Load EEG Signals from .mat Files

The EEG signals are stored in .mat files inside the PD and HC folders. Your task is to load these files, extract the EEG data, and store them for further processing.

- Write a function to load EEG signals from `.mat` files using the `scipy.io.loadmat` module.
- Extract only the first 32 EEG channels.

Step 3: Calculate Power Band Features

EEG signals are complex time-series data reflecting the brain's electrical activity. These signals are not monolithic; instead, they are comprised of various rhythmic oscillations occurring at different frequencies. These distinct frequency ranges are known as EEG sub-frequency bands.
The image below displays an EEG signal decomposed into its sub-frequency bands. The four signals shown are the result of processing the raw EEG data through four distinct band-pass filters, each with its specified cut-off frequencies.



The amount of energy or "power" within each band can provide valuable insights into brain function, cognitive states, and neurological conditions. Analyzing the power distribution across these bands is a fundamental and widely used technique in EEG signal processing. It allows researchers and clinicians to identify patterns associated with different mental activities, sleep stages, or the presence of disorders like Parkinson's Disease.

The commonly recognized EEG power bands, along with their typical frequency ranges and general associations, are:

Delta (δ) Band (approx. 0.5 - 4 Hz): This is the slowest band. It is most prominent during deep, NREM (non-rapid eye movement) sleep. In awake individuals, significant delta activity can sometimes indicate certain brain pathologies or be present during some continuous attention tasks.

Theta (θ) Band (approx. 4 - 8 Hz): Theta waves are often linked to drowsiness, light sleep, and the transition to deeper sleep stages, including REM sleep. In awake states, theta activity is associated with focused attention, meditation, memory processing, and emotional states.

Alpha (α) Band (approx. 8 - 13 Hz): Alpha waves are characteristic of a relaxed, wakeful state, especially when the eyes are closed or during quiet contemplation. They are often diminished by opening the eyes or by mental exertion.

Beta (β) Band (approx. 13 - 30 Hz): Beta waves are associated with active thinking, alertness, concentration, problem-solving, and focused mental activity. They are the dominant rhythm in individuals who are alert, anxious, or have their eyes open and are engaged in cognitive tasks.

Gamma (γ) Band (approx. 30 - 100 Hz, sometimes higher): Gamma waves are the fastest EEG frequencies and are thought to be involved in higher cognitive functions, such as perception, learning, memory formation, and consciousness. They are believed to play a role in binding sensory information from different brain areas into a coherent percept.
In this homework, you will calculate the power within these bands from the provided EEG segments. These power values will then serve as features for a machine learning model to distinguish between EEG data from individuals with Parkinson's Disease (PD) and Healthy Controls (HC).

Tasks

Your primary task in this step is to calculate the average power for each of the five standard EEG bands (Delta, Theta, Alpha, Beta, Gamma) for each of the 32 EEG channels extracted in Step 2. You will need to do this for all EEG segments from both the PD and HC groups.

Define Frequency Bands:

- Delta: 0.5 - 4 Hz
- Theta: 4 - 8 Hz
- Alpha: 8 - 13 Hz
- Beta: 13 - 30 Hz
- Gamma: 30 - 100 Hz

The general idea involves two conceptual steps:

1. Obtaining Sub-Band Signals: Imagine you could pass the raw EEG signal for a channel through different band-pass filters. Each filter would isolate the activity (a sub-band signal) for one of the frequency bands above.

2. Calculating Power of Sub-Band Signals: You would then calculate the power of each of these conceptual sub-band signals.

However, for practical implementation, you will achieve this more directly using the Power Spectral Density (PSD).

Steps for Feature Extraction

For each EEG segment, and for each of the 32 channels within it:

1 - Calculate Power Spectral Density (PSD) per Channel:

- For the current channel's data within the segment, estimate its PSD.
- Use a standard method like Welch's periodogram (e.g., available in `scipy.signal`).
- Remember to use the sampling frequency (Fs) of 256 Hz.

```
# --- Conceptual Semi-Code: PSD Calculation ---
# current_channel_data = ... (data for one channel in a segment)
# Fs = 256
#
# frequencies, psd_values = calculate_psd_welch(current_channel_data, fs=Fs, ...)
```

2 - Calculate Average Power in Each Band from PSD:

- Using the frequencies and `psd_values` obtained from the PSD, calculate the average power within each of the five defined frequency bands (Delta, Theta, Alpha, Beta, Gamma).
- This involves identifying the PSD values that fall within each band's frequency range and then averaging them.

```
# --- Conceptual Semi-Code: Band Power Calculation (repeated for each band) ---
# For Alpha band (8-13 Hz):
# relevant_psd_for_alpha = psd_values corresponding to frequencies between 8 Hz
and 13 Hz
# power_alpha = average(relevant_psd_for_alpha)
#
# Repeat for Delta, Theta, Beta, Gamma bands.
# This will give you 5 power values for the current channel.
```

3 - Construct Feature Vector for the Segment:

- For each EEG segment, you will have 5 power values (one for each band) from each of the 32 channels.
- This results in `5 (bands) × 32 (channels) = 160` features per segment.
- Combine these 160 features into a single row vector. Ensure a consistent order of features for all segments.

4 - Store Features Separately for PD and HC Groups:

- PD Subjects: Process all EEG segments from the PD folder. Collect the 160-feature vectors for each segment into a 2D NumPy array (e.g., `pd_features`). Each row will be a segment, and columns will be the features.
- HC Subjects: Do the same for all segments from the HC folder, storing them in a separate 2D NumPy array (e.g., `hc_features`).

Step 4: Prepare Labels and Combine Data

To train your machine learning models, you need to associate your extracted features with their correct class labels.

1. **Create Labels:**
    - Assign a label of **0** to all feature vectors extracted from the **Healthy Control (HC)** subjects.

- Assign a label of **1** to all feature vectors extracted from the **Parkinson's Disease (PD)** subjects.

2. **Combine Data:**
   - Concatenate your `hc_features` and `pd_features` arrays into a single feature matrix (e.g., `X`).
   - Similarly, combine their corresponding labels into a single label vector (e.g., `y`). Ensure the order of features in `X` matches the order of labels in `y`.

```
# --- Conceptual Semi-Code ---
# hc_labels = array of 0s, with length matching number of HC segments
# pd_labels = array of 1s, with length matching number of PD segments
#
# X = combine(hc_features, pd_features)  # e.g., using np.vstack
# y = combine(hc_labels, pd_labels)      # e.g., using np.concatenate
```

Step 5: Train an SVM Classifier

You will now train and evaluate several common machine learning classifiers on your prepared dataset.

For *each* of the following five classifiers:

1 - **Split the Dataset:** Divide your combined feature matrix (`X`) and label vector (`y`) into training and testing sets. Use `train_test_split` from `Scikit-learn`.

2 - **Train the Classifier:** Train the classifier on the training data.

3 - **Evaluate the Classifier:** Evaluate its performance on the testing data (see Step 6 for metrics).

Classifiers to Implement:

I. **Support Vector Machine (SVM):** Use SVC from `sklearn.svm`. (Consider using a `kernel='linear'`).

II. **Logistic Regression:** Use `LogisticRegression` from `sklearn.linear_model`.

III. **K-Nearest Neighbors (k-NN):** Use `KNeighborsClassifier` from `sklearn.neighbors`. (You might experiment with different values for `n_neighbors`).

IV. **Decision Tree:** Use `DecisionTreeClassifier` from `sklearn.tree`.

V. **Random Forest:** Use `RandomForestClassifier` from `sklearn.ensemble`.

Hints for Splitting and Training:

- When using `train_test_split`, set `test_size=0.2` and `random_state=1` for reproducibility across your experiments.

```
# --- Conceptual Semi-Code (to be repeated for each classifier type) ---
# from sklearn.model_selection import train_test_split
# from sklearn.svm import SVC # (and imports for other classifiers)
#
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
#
# # Example for SVM:
# model_svm = SVC(kernel='linear', random_state=1) # Or other classifier
# model_svm.fit(X_train, y_train)
# predictions_svm = model_svm.predict(X_test)
#
# # ... then evaluate predictions_svm (see Step 6)
# # Repeat for Logistic Regression, k-NN, Decision Tree, Random Forest
```

Step 6: Evaluate the Classifier

Once each classifier is trained, you need to assess its performance on the unseen test data.

Task: For *each* of the five classical classifiers trained in Step 5 (and the MLP if you attempt the extra points):

1. Calculate and report the following metrics using the predictions on the test set:
   - **Accuracy:** Overall correctness (`accuracy_score` from `sklearn.metrics`).
   - **Precision, Recall, and F1-Score:** Use `classification_report` from `sklearn.metrics` to get these per-class and averaged values.

Presentation of Results:

- It is recommended to summarize the performance metrics (Accuracy, Precision, Recall, F1-Score for class 1 (PD)) for all classifiers in a table for easy comparison.

```
# --- Conceptual Semi-Code (following predictions for a model) ---
# from sklearn.metrics import accuracy_score, classification_report
#
# # Assuming 'y_test' are the true labels and 'predictions_model' are from a
trained model
# accuracy = accuracy_score(y_test, predictions_model)
# report = classification_report(y_test, predictions_model)
#
# print(f"Results for [Classifier Name]:")
# print(f"Accuracy: {accuracy}")
# print("Classification Report:\n", report)
```

Implement a Multi-Layer Perceptron (MLP) Classifier (+10 points)

For extra points, implement, train, and evaluate an MLP (a type of neural network) for the same classification task.

Task:

1. **Choose a Framework:** You can use `MLPClassifier` from `sklearn.neural_network` for a quick

implementation, or delve deeper using libraries like `Keras` (with `TensorFlow` backend) for more customization.

2. **Define MLP Architecture:** If using `Keras/TensorFlow`, you'll need to define the layers (e.g., number of hidden layers, number of neurons per layer, activation functions). For `MLPClassifier`, you can specify `hidden_layer_sizes`. Start with a simple architecture (e.g., one or two hidden layers).
3. **Train the MLP:** Use the same training data (`X_train`, `y_train`) as prepared in Step 5. You might need to experiment with hyperparameters like learning rate, number of epochs, and batch size.
4. **Evaluate the MLP:** Evaluate its performance on the test set (`X_test`, `y_test`) using the same metrics as in Step 6 (Accuracy, Precision, Recall, F1-Score). Include these results in your comparison table.

Brief Hints for MLP:

- Data scaling (e.g., `StandardScaler` from `Scikit-learn`) is often beneficial for neural networks. Apply it to `X_train` (fit and transform) and `X_test` (transform only).
- For `MLPClassifier`, you can set `random_state` for reproducibility.
- Start with default hyperparameters and iterate if time permits.

توجه:

1. حتماً در موعد مقرر به بارگذاری پاسخ تمرین در سامانه VC اقدام کنید (تاخیر قابل قبول نیست)
2. در صورت بروز هرگونه ابهام یا سوال در مورد تمرین حتما با دستیاران آموزشی در ارتباط باشید.

3. لطفاً نام خانوادگی خود را به صورت لاتین به همراه شماره تمرین به عنوان نام فایل بارگذاری شده قرار دهید.

به عنوان مثال :

FIS1_YourFullName_HW1

4. پاسخ ارسالی شما باید در یک فایل ورد نوشته شده و فایل ورد و PDF آن هردو در یک فایل rar ارسال گردند.

5. زمان کافی جهت تحقیق جهت یافتن پاسخ تمارین به دانشجویان گرامی داده شده. لازم به ذکر است که هر شخص باید برداشت و نتایج تحقیقات خود را ارائه دهد پس به همین دلیل از کپیبرداری پاسخنامه یکدیگر شدیدا خودداری فرمایید و در صورت مشاهده مسئولیت عواقب منفی متوجه تمامی افراد خاطی میگردد. همچنین ممکن است جلساتی تحت عنوان پرسش و پاسخ در خصوص گزارش دوستان گرامی برگزار شود

موفق باشید.