Technical Report: EEG Connectivity Analysis Using Pearson Correlation

1 Introduction

Electroencephalography (EEG) is a method used to record electrical activity of the brain. Analyzing EEG data involves understanding the connectivity between different regions of the brain. This report outlines the process of computing and visualizing the connectivity matrix from EEG data using Pearson correlation. The analysis is performed using Python libraries including `pandas`, `numpy`, `matplotlib`, `networkx`, and `mne`.

2 Requirements

- Python 3.8 or later
- Required packages: `pandas`, `numpy`, `matplotlib`, `networkx`, `mne`

3 Installation

To ensure compatibility and successful execution, install the specific versions of the required packages:

```
pip install mne==1.2 numpy==1.21.1 pandas matplotlib networkx
```

Reason: Different versions of packages might have compatibility issues or deprecated functions. Installing specific versions ensures the code runs without errors.

4 Data Description

The EEG data used in this analysis is stored in a text file with the following format:

- **Header Information**: Comments providing metadata about the recording.

- **Data Columns**: Each row represents a time point in the recording.
  - Column 0: Sample index
  - Columns 1-8: EEG data from 8 channels
  - Additional columns: Sensor data, timestamp, etc.

Sample Data

```
%OpenBCI Raw EEG Data
%Number of channels = 8
%Sample Rate = 250 Hz
%Board = OpenBCI_GUI$BoardCytonSerial
0, 61379.36, 49492.89, -16597.06, -21309.75, 6703.91, -3284.86, 7223.10, 1740.11, 0.040, 0.4
1, 60973.46, 48972.47, -16279.02, -21050.13, 7045.02, -2887.53, 7593.09, 2118.70, 0.040, 0.4
...
```

Reason: Understanding the data format is crucial for correctly parsing and extracting the necessary information for analysis.

5 Code Explanation

The following Python script reads the EEG data, computes the Pearson correlation matrix, and visualizes the connectivity matrix.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from mne.io import RawArray
from mne import create_info

# Function to compute connectivity matrix using Pearson correlation
def compute_connectivity_matrix(data):
    correlation_matrix = np.corrcoef(data)
    return correlation_matrix

# Step 1: Load your EEG data from a text file
file_path = 'path_to_your_eeg_data.txt'  # Replace with your actual file path
data = pd.read_csv(file_path, comment='%', header=None)

# Step 2: Extract EEG data and channel names
# The first column is the sample index, next 8 columns are EEG data
eeg_data = data.iloc[:, 1:9].values.T  # Transpose to have shape (n_channels, n_samples)
n_channels, n_samples = eeg_data.shape

# Step 3: Create MNE Info object
```

```
sfreq = 250  # Sample rate (Hz)
ch_names = [f'Channel{i+1}' for i in range(n_channels)]
info = create_info(ch_names=ch_names, sfreq=sfreq, ch_types='eeg')

# Step 4: Create MNE Raw object
raw = RawArray(eeg_data, info)

# Step 5: Compute connectivity matrix (Pearson correlation in this example)
connectivity_matrix = compute_connectivity_matrix(eeg_data)

# Display the actual connectivity matrix
print("Connectivity Matrix (Pearson Correlation):")
print(connectivity_matrix)

# Step 6: Save or Process Connectivity Matrix
np.savetxt('connectivity_matrix.csv', connectivity_matrix, delimiter=',')

# Optional: Visualize the Connectivity Matrix
plt.imshow(connectivity_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
plt.title('Connectivity Matrix (Pearson Correlation)')
plt.xlabel('Channels')
plt.ylabel('Channels')
plt.show()

# Optional: Create a NetworkX graph and visualize
G = nx.from_numpy_array(connectivity_matrix)
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=700, node_color='skyblue', edge_color='k', font_
plt.show()
```

5.1 Explanation

5.1.1 Loading the Data

The script reads the EEG data from the text file using `pandas`, ignoring the header lines starting with %.

Reason: This step is necessary to import the data into a format that can be easily manipulated and analyzed in Python.

5.1.2 Extracting EEG Data

The relevant EEG data columns (1 to 8) are extracted and transposed to fit the required format for MNE processing.

Reason: Extracting and transposing the data ensures that each row corresponds to a channel and each column corresponds to a time point, which is the expected format for further analysis.

5.1.3 Creating MNE Objects

An `Info` object is created to store metadata such as channel names and sampling frequency. A `RawArray` object is created using the EEG data and the `Info` object.

Reason: Creating these objects facilitates the use of MNE's powerful EEG processing capabilities, even though we are primarily using it for data structuring here.

5.1.4 Computing the Connectivity Matrix

The `compute_connectivity_matrix` function calculates the Pearson correlation coefficients between all pairs of EEG channels using `np.corrcoef`.

Reason: Pearson correlation is a widely used measure of linear dependence between two variables. Computing this matrix helps identify how each EEG channel is related to others.

5.1.5 Displaying and Saving the Connectivity Matrix

The computed connectivity matrix is displayed and saved to a CSV file for further analysis.

Reason: Saving the matrix allows for future reference and analysis, while displaying it provides immediate insights into the connectivity structure.

5.1.6 Visualizing the Connectivity Matrix

A heatmap of the connectivity matrix is generated using `matplotlib`, showing the correlation strengths between channels.

Reason: Visualization helps in easily interpreting the data by providing a graphical representation of the connectivity strengths.

5.1.7 Visualizing the Network Graph

A NetworkX graph is created from the connectivity matrix and visualized using `networkx` and `matplotlib`.

Reason: Network graphs offer an intuitive way to visualize relationships and dependencies between EEG channels, highlighting the overall connectivity structure.

6 Output

6.1 Connectivity Matrix (Pearson Correlation)

The connectivity matrix is a symmetric matrix where each element $[i, j]$ represents the correlation coefficient between channel i and channel j. Values close to 1 indicate a strong positive linear relationship, while values close to -1 indicate a strong negative linear relationship. Values close to 0 indicate little to no linear relationship.

6.2 Heatmap of the Connectivity Matrix

The heatmap provides a visual representation of the connectivity matrix, where:

- **X and Y axes** represent the EEG channels (0 to 7).
- **Color intensity** represents the strength of the Pearson correlation between the channels.
  - **Dark red/black** indicates a high positive correlation (close to 1).
  - **Yellow/white** indicates a lower correlation.

6.3 Graph Representation of the Connectivity Matrix

The graph representation provides a visual overview of how each channel is connected to others:

- **Nodes** represent EEG channels.
- **Edges** represent the connections (correlations) between channels.

7 Conclusion

This report demonstrates how to process EEG data, compute the Pearson correlation matrix, and visualize the connectivity matrix using Python. The provided code efficiently handles the data extraction, computation, and visualization, making it a valuable tool for EEG connectivity analysis.

Key Takeaways:

- **Loading and Preprocessing Data**: Ensuring data is in the correct format for analysis.
- **Computing Connectivity**: Using Pearson correlation to understand linear relationships between EEG channels.
- **Visualization**: Enhancing interpretation of data through heatmaps and network graphs.

By following the steps outlined in this report, you can adapt the code to analyze your own EEG data, customize the connectivity measures, and gain insights into brain connectivity patterns.