Evaluating Connectivity Measures for Predicting Time Series Data from EEG

Yehia Gewily

Jun, 2024

1 Introduction

Electroencephalography (EEG) is a widely used technique for recording the electrical activity of the brain. Analyzing the connectivity between different EEG channels helps in understanding brain function and predicting various neurological conditions. This report explores different connectivity measures, including Pearson correlation, Phase Locking Value (PLV), Coherence, and Phase Lag Index (PLI), and evaluates their effectiveness in predicting time series data from EEG signals.

2 Connectivity Measures and Their Computation

2.1 Pearson Correlation

Pearson correlation measures the linear relationship between two signals. It is computed as the covariance of the signals divided by the product of their standard deviations.

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

where $\text{cov}(X, Y)$ is the covariance of signals X and Y, and σ_X and σ_Y are their standard deviations.

2.2 Phase Locking Value (PLV)

PLV measures the consistency of the phase difference between two signals. It is computed by converting signals into their analytic representations using the Hilbert transform and then calculating the phase differences.

$$\text{PLV} = \left| \frac{1}{N} \sum_{n=1}^{N} e^{i(\theta_x(n) - \theta_y(n))} \right|$$

where $\theta_x(n)$ and $\theta_y(n)$ are the instantaneous phases of signals x and y at time n, and N is the number of samples.

2.3 Coherence

Coherence measures the frequency-specific correlation between two signals. It is computed as the magnitude squared of the cross-spectral density normalized by the product of the power spectral densities.

$$\text{Coh}_{xy}(f) = \frac{|S_{xy}(f)|^2}{S_{xx}(f)S_{yy}(f)}$$

where $S_{xy}(f)$ is the cross-spectral density of signals x and y at frequency f, and $S_{xx}(f)$ and $S_{yy}(f)$ are the power spectral densities.

2.4 Phase Lag Index (PLI)

PLI measures the consistency of phase differences that are non-zero, reducing the influence of volume conduction. It is computed by analyzing the sign of the phase differences.

$$\text{PLI} = \left| \frac{1}{N} \sum_{n=1}^{N} \text{sign}(\sin(\theta_x(n) - \theta_y(n))) \right|$$

where $\theta_x(n)$ and $\theta_y(n)$ are the instantaneous phases of signals x and y at time n, and N is the number of samples.

3 Code Explanation

The following code computes the above-mentioned connectivity measures for a given EEG dataset and visualizes them. It also prepares the data for further analysis to determine which connectivity measure is best for predicting time series data.

Listing 1: Connectivity Measures Computation and Visualization

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from scipy.signal import hilbert, coherence
from mne.io import RawArray
from mne import create_info

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

```
# Function to compute instantaneous phase using Hilbert
    ↪ transform
def compute_instantaneous_phase(data):
    analytic_signal = hilbert(data, axis=1)
    instantaneous_phase = np.angle(analytic_signal)
    return instantaneous_phase

# Function to compute Phase Locking Value (PLV) matrix
def compute_plv(instantaneous_phase):
    n_channels = instantaneous_phase.shape[0]
    plv_matrix = np.zeros((n_channels, n_channels))
    for i in range(n_channels):
        for j in range(n_channels):
            phase_diff = instantaneous_phase[i, :] -
                ↪ instantaneous_phase[j, :]
            plv_matrix[i, j] = np.abs(np.mean(np.exp(1j *
                ↪ phase_diff)))
    return plv_matrix

# Function to compute Coherence matrix
def compute_coherence_matrix(data, sfreq):
    n_channels = data.shape[0]
    coh_matrix = np.zeros((n_channels, n_channels))
    for i in range(n_channels):
        for j in range(i, n_channels):
            f, Cxy = coherence(data[i], data[j], fs=sfreq)
            coh_matrix[i, j] = np.mean(Cxy)
            coh_matrix[j, i] = coh_matrix[i, j]  # Symmetric
                ↪  matrix
    return coh_matrix

# Function to compute Phase Lag Index (PLI) matrix
def compute_pli(instantaneous_phase):
    n_channels = instantaneous_phase.shape[0]
    pli_matrix = np.zeros((n_channels, n_channels))
    for i in range(n_channels):
        for j in range(n_channels):
            phase_diff = instantaneous_phase[i, :] -
                ↪ instantaneous_phase[j, :]
            pli_matrix[i, j] = np.abs(np.mean(np.sign(np.sin
                ↪ (phase_diff))))
    return pli_matrix

# Step 1: Load your EEG data from a text file
file_path = 'OpenBCI_GUI-v5-blinks-jawClench-alpha.txt'  #
    ↪ Replace with your actual file path
data = pd.read_csv(file_path, comment='%', header=None)

# Step 2: Extract EEG data and channel names
```

```
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 matrices
correlation_matrix = compute_connectivity_matrix(eeg_data)
instantaneous_phase = compute_instantaneous_phase(eeg_data)
plv_matrix = compute_plv(instantaneous_phase)
coherence_matrix = compute_coherence_matrix(eeg_data, sfreq)
pli_matrix = compute_pli(instantaneous_phase)

# Step 6: Display and save the connectivity matrices

print("Pearson␣Correlation␣Matrix:")
correlation_df = pd.DataFrame(correlation_matrix, index=
    ↪ ch_names, columns=ch_names)
print(correlation_df)
correlation_df.to_csv('correlation_matrix.csv')

print("Phase␣Locking␣Value␣(PLV)␣Matrix:")
plv_df = pd.DataFrame(plv_matrix, index=ch_names, columns=
    ↪ ch_names)
print(plv_df)
plv_df.to_csv('plv_matrix.csv')

print("Coherence␣Matrix:")
coherence_df = pd.DataFrame(coherence_matrix, index=ch_names
    ↪ , columns=ch_names)
print(coherence_df)
coherence_df.to_csv('coherence_matrix.csv')

print("Phase␣Lag␣Index␣(PLI)␣Matrix:")
pli_df = pd.DataFrame(pli_matrix, index=ch_names, columns=
    ↪ ch_names)
print(pli_df)
pli_df.to_csv('pli_matrix.csv')

# Step 7: Visualize the connectivity matrices

def visualize_matrix(matrix, title):
    fig, ax = plt.subplots(figsize=(10, 8))
    cax = ax.imshow(matrix, cmap='hot', interpolation='
        ↪ nearest')
    fig.colorbar(cax)
    ax.set_title(title)
    ax.set_xticks(np.arange(n_channels))
    ax.set_xticklabels(ch_names, rotation=90)
    ax.set_yticks(np.arange(n_channels))
    ax.set_yticklabels(ch_names)
    ax.set_xlabel('Channels')
    ax.set_ylabel('Channels')

    for i in range(n_channels):
        for j in range(n_channels):
            ax.text(j, i, f'{matrix[i,␣j]:.2f}', ha='center'
                ↪ , va='center', color='white' if matrix[i,
                ↪ j] > 0.5 else 'black')

    plt.show()

visualize_matrix(correlation_matrix, 'Pearson␣Correlation␣
    ↪ Matrix')
visualize_matrix(plv_matrix, 'Phase␣Locking␣Value␣(PLV)␣
    ↪ Matrix')
visualize_matrix(coherence_matrix, 'Coherence␣Matrix')
visualize_matrix(pli_matrix, 'Phase␣Lag␣Index␣(PLI)␣Matrix')

def visualize_network(matrix, title):
    fig = plt.figure()
    G = nx.from_numpy_array(matrix)
    pos = nx.spring_layout(G)
    nx.draw(G, pos, with_labels=True, node_size=700,
        ↪ node_color='skyblue', edge_color='k', font_weight=
        ↪ 'bold')
    plt.title(title)
    plt.show()

visualize_network(correlation_matrix, 'Network␣Graph␣of␣
    ↪ Pearson␣Correlation␣Matrix')
visualize_network(plv_matrix, 'Network␣Graph␣of␣PLV␣Matrix')
visualize_network(coherence_matrix, 'Network␣Graph␣of␣
    ↪ Coherence␣Matrix')
visualize_network(pli_matrix, 'Network␣Graph␣of␣PLI␣Matrix')
```

4 Next Steps

To determine which connectivity measure is best for predicting time series data, the following steps can be undertaken:

4.1 Feature Extraction

Extract relevant features from each connectivity matrix. Features could include:

- Mean connectivity values.
- Connectivity strength between specific pairs of regions.
- Network metrics such as node degree, clustering coefficient, and centrality.

4.2 Preparing the Data

Create feature vectors from the extracted features of each connectivity matrix. Combine these feature vectors into a structured format, such as a DataFrame, where each row represents a different sample and each column represents a feature.

4.3 Splitting the Data

Split the dataset into training and testing sets. Typically, 80% of the data is used for training the model, and 20% is used for testing and validating the model's performance.

4.4 Model Selection

Choose appropriate machine learning models for time series prediction. Common models include:

- Linear Regression
- Support Vector Machines (SVM)
- Random Forest
- Long Short-Term Memory (LSTM) networks for more complex temporal dependencies

4.5 Training the Model

Train the chosen model on the training dataset. This involves using the training data to adjust the model's parameters so that it can accurately predict the target variable.

4.6 Evaluation

Evaluate the model's performance on the testing set using metrics such as:

- Mean Squared Error (MSE)
- R^2 Score (Coefficient of Determination)

These metrics help quantify the accuracy and robustness of the model's predictions.

4.7 Comparison

Compare the performance of models trained with features from different connectivity matrices. Determine which connectivity measure results in the lowest prediction error and the highest R^2 score, indicating the best predictive performance.

By systematically evaluating the models, we can identify the most effective connectivity measure for predicting time series data from EEG signals.

5 Conclusion

This report presented various connectivity measures for EEG data, explained the computation and logic behind each measure, and provided a code implementation to compute and visualize these measures. The next steps involve feature extraction and model training to evaluate the predictive performance of each connectivity measure and determine the most effective one for predicting time series data.