Human Computer Interfaces Project

Motor Imagery based Game interface

Team ID: SC_33

Department	Level	ID	Name
SC	4	2021170488	محمد مصطفي محمد السيد
SC	4	2021170493	محمود احمد عبدالرحيم السيد
SC	4	20191700783	يوسف محمد أحمد كمال الدين
SC	4	20201700481	عبد الرحمن محمد أحمد عواد

Project Overview

This project implements a Brain-Computer Interface (BCI) system that uses EEG signals to control a Pacman game. The system processes EEG data from motor imagery tasks (right hand, left hand, feet, and tongue movements) and uses machine learning with SVM model to predict the intended movement.

Implementation Details

1. Data Processing and Preprocessing

- **Data Source**: BCICIV_2a dataset containing EEG recordings from 9 patients
- **Sampling Rate**: 250 Hz
- **Trial Duration**: 4 seconds per trial
- **Channels**: 22 EEG channels
- **Classes**: 4 motor imagery tasks (right hand, left hand, feet, tongue)

2. Signal Processing Pipeline

Bandpass Filtering

- Applied bandpass filter (8-30 Hz)
- Used Butterworth filter of order 4
- Implemented using scipy.signal.butter and filtfilt

Notch Filtering

- Applied 50 Hz notch filter
- Quality factor of 30
- Implemented using scipy.signal.iirnotch

PCA (Principal Component Analysis)

- Tested different numbers of components (k=4, 16, 25)
- Results:
  - k=16: Captured ~98% of variance
  - k=4: Captured ~95% of variance
- Selected k=16 as optimal due to highest variance capture with same accuracy

ICA (Independent Component Analysis)

- Applied FastICA with 16 components
- Identified and removed components containing eye blink artifacts
- Reduced to 12 components after artifact removal
- Improved accuracy to 30% after artifact removal

3. Feature Extraction

CSP (Common Spatial Patterns)

- Used MNE's CSP implementation
- Extracted 4 spatial patterns
- Applied to cleaned data after ICA
- Features used for final classification

4. Model Implementation

Model Comparison

Tested multiple classification algorithms to find the best performer:

- **SVM Classifier**
  - Used Support Vector Machine with RBF kernel
  - Parameters:
    - C=100
    - gamma=0.1
    - kernel='rbf'
  - Accuracy: 30%
  - Best performance on Left Hand class (49% recall)
- **Random Forest Classifier**
  - Parameters:
    - n_estimators=100
    - random_state=42
  - Accuracy: 25.18%
  - Most balanced performance across classes
  - Best performance on Left Hand class (32% recall)
- **K-Nearest Neighbors (KNN)**
  - Parameters:
    - n_neighbors=5
  - Accuracy: 24.47%
  - Best performance on Right Hand class (35% recall)
  - Struggled with Feet and Tongue classes
- **Linear Discriminant Analysis (LDA)**
  - Default parameters
  - Accuracy: 26.37%
  - Most balanced performance between Feet and Tongue classes
  - Best performance on Right Hand class (35% recall)

Model Performance Analysis

- All models showed similar performance levels (24-27% accuracy)
- SVM performed slightly better than other models
- Common challenges across all models:
- Low accuracy for Feet class
- Better performance on hand movements than feet/tongue
- High variance in class-wise performance

5. Game Implementation (Pacman)

Features

- Grid-based game (10x10)
- Two modes:
  - Prediction Mode:
    - Uses model predictions to control Pacman
    - 5 coins and 8 walls
    - Moves automatically based on predictions
  - Keyboard Mode:
    - Manual control using arrow keys
    - 10 coins and 15 walls
    - A more challenging layout

Game Elements

- Pacman (yellow circle)
- Coins (white circles)
- Walls (blue rectangles)
- Score display
- Grid visualization

6. Testing and Evaluation

Model Testing

- Load saved model and test data
- Make predictions on test set
- Saves predictions for game use
- Provides detailed classification report
- Maps predictions to game directions:
  - Right hand → right
  - Left hand → left
  - Feet → down
  - Tongue → up

Current Progress and Challenges

1. **Accuracy**: Current model accuracy is around 30%, which is a significant challenge for reliable game control
2. **Feature Engineering**:
   - PCA and ICA implementations have been optimized
   - CSP features show promise but need further tuning
3. **Game Control**:
   - Basic game mechanics are implemented
   - Prediction-based control needs improvement due to model accuracy limitations

Future Improvements

1. **Model Enhancement**:
   - Try different classification algorithms
   - Implement ensemble methods
   - Optimize hyperparameters further
2. **Feature Engineering**:
   - Explore additional feature extraction methods
   - Investigating temporal features
   - Consider channel selection methods
3. **Game Improvements**:
   - Add difficulty levels
   - Implement smoother movement
   - Add visual feedback for predictions
4. **User Experience**:
   - Add calibration phase
   - Implement real-time feedback
   - Improve game interface