AI & BIOLOGICAL COMPUTATION

Final Project- Positive and Negative Feeling Recognition Using EEG Dataset

By Mohamad Hosein Faramarzi - 99104095
Bachelor's student at Sharif university of technology

fall semester-1402

Sharif University of Technology
Electrical Engineering Department

Course Proffesor Dr.Sepideh Hajipour

Contents

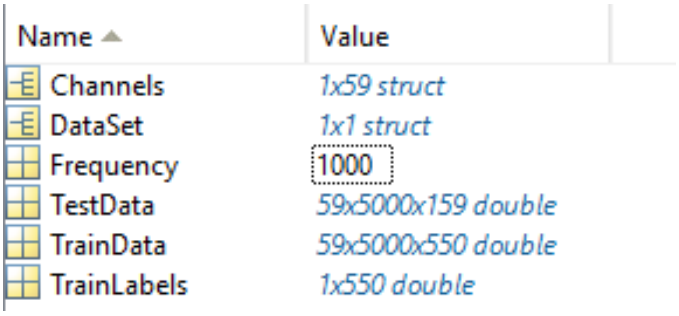
1 Project Description

In this project we aim to predict the positive and negative feeling and this 2 labels form our 2 classes. We have a Dataset of 10 people in the environment of VR (Virtual Reality) and by different stimuli we make positive and negative feeling for them. We have totally 709 Experiments to train our model. 550 of the experiments are used for training model so they have labels 1 and -1 based on the type of feeling and other 159 data are without any label and we should do the prediction. In this report first we describe our method to build this model and after that we check the accuracy of this model to see how well it can do the predictions.

Figure 1: Task and Experiment Environment

2 Method and Code Description

First we load the data set and put each part of the .mat file in each variable described in the figure bellow:



The data is formed by 5 seconds recordings in 2 states of positive feeling and negative feeling. The stares can't be predicted by their raw EEG signals. As you can see in the following figure there is no efficient difference between this 2 experiments in EEG signals.

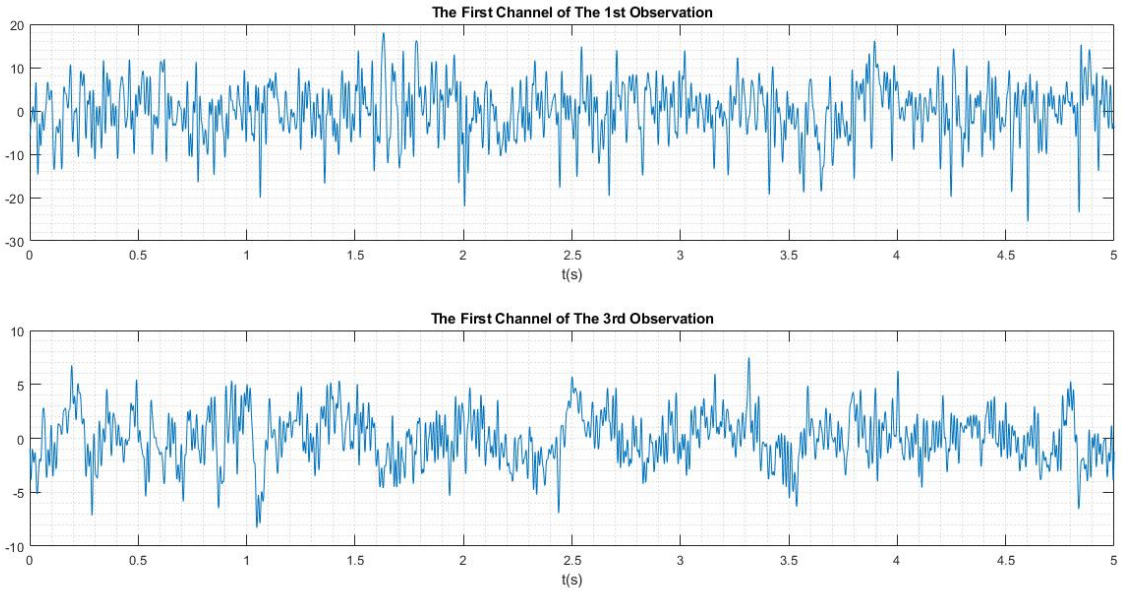


Figure 2: First EEG signal shows + and second one show -

So we have to extract some efficient features by different temporal and frequency characteristic. In next sections we show our method to do this task.

3 Phase 1- Feature extraction and Train Models

3.1 Feature Extraction

Our main data is a complex and high dimensional data that doesn't have any specific patterns to discriminate 2 different classes.In this part we have to do a dimensional reduction so we have some simple features for each 5000D data and based on this features we can discriminate between + and - classes and do the classification. This features can be based on temporal characteristics and also frequency characteristics and we'll describe them in the following.

3.1.1 Mean Parameter

Mean-based feature extraction involves calculating the average value of a specific attribute, such as signal amplitude in EEG data, over a defined period. This feature provides insight into the central tendency or baseline of the data, offering a summary of overall signal levels. In emotion classification tasks, variations in mean values can indicate differences in neural activity associated with emotional states.

We do this with **mean** instruction.

3.1.2 Mode Parameter

Mode-based feature extraction focuses on identifying the most frequently occurring value in a dataset. For EEG signals, the mode represents the amplitude or value that occurs with the highest frequency

within a specific time window or channel. This feature can capture the dominant patterns or recurring elements in the signal.

We do this with **mode** instruction.

3.1.3 Variance Parameter

Variance is a feature that measures the spread or dispersion of values in a dataset. It provides information about the degree of variability or fluctuations in the data.

We do this with **var** instruction.

3.1.4 Median Parameter

Median represents the middle value of a dataset when arranged in ascending or descending order, making it less sensitive to extreme values than the mean.

We do this with **median** instruction.

3.1.5 Skewness Parameter

Skewness quantifies the asymmetry of the data distribution; positive skewness indicates a longer tail on the right, while negative skewness implies a longer tail on the left. We do this extraction by **skewness** instruction.

$$\tilde{\mu}_3 = \frac{\sum_i^N \left(X_i - \bar{X}\right)^3}{(N-1) * \sigma^3}$$

$\tilde{\mu}_3$ = skewness
N = number of variables in the distribution
X_i = random variable
$\bar{X}$ = mean of the distribution
σ = standard deviation

Figure 3: Skewness Formula

3.1.6 Kurtosis Parameter

Kurtosis measures the shape and peakedness of the data distribution; high kurtosis indicates a more peaked distribution.We do this extraction by **kurtosis** instruction.

$$K = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i - \mu}{\sigma}\right)^4$$

Figure 4: Kurtosis Formula

3.1.7 Energy and Average Power

he energy of a signal represents the total magnitude or "strength" of the signal over a specified duration. For a discrete signal x[n], the energy E is calculated as the sum of the squared values of each sample. We use **sum(square(...))** instruction.

Average power measures the average strength or intensity of a signal over time. It is related to energy but normalized by the signal's duration. For a discrete signal x[n], the average power P_{avg} is calculated as the ratio of the signal's energy to the number of samples.We use **meansqr()** instruction to extract this feature.

3.1.8 Correlation Parameter

Correlation is a statistical measure that quantifies the degree of linear relationship between two variables. In the context of feature extraction, correlation is often used to assess the relationship between different channels or features within a dataset. For a pair of variables X and Y, the Pearson correlation coefficient (r) is commonly used, and it is calculated as:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Figure 5: Correlation Formula

We do this with **corr** instruction.

3.1.9 Form Factor

The form factor is a feature commonly used in signal processing to characterize the shape or form of a waveform. Specifically, in the context of feature extraction from signals, the form factor is defined as the ratio of the root mean square (RMS) value to the mean value of the absolute amplitudes of a signal.

3.1.10 Frequency Mean

FreqMean is a feature used in the frequency domain analysis of signals, particularly in EEG or time-series data. It calculates the average frequency content of a signal by taking the mean of the spectral frequencies weighted by their respective magnitudes. This feature provides insights into the central tendency of the frequency distribution, offering information about the prevalent frequencies in the signal.

3.1.11 Frequency Median

FreqMean is a feature used in the frequency domain analysis of signals, particularly in EEG or time-series data. It calculates the average frequency content of a signal by taking the mean of the spectral frequencies weighted by their respective magnitudes. This feature provides insights into the central tendency of the frequency distribution, offering information about the prevalent frequencies in the signal.

3.1.12 PeakFrequency

PeakFrequency is a feature that identifies the frequency at which the power spectrum of a signal exhibits its maximum value or peak. It is determined by locating the frequency bin with the highest amplitude in the signal's power spectrum.

3.1.13 MaxFreqIndex (Maximum Frequency Index)

MaxFreqIndex is a feature that represents the index or location of the maximum frequency in the power spectrum of a signal. It indicates the position in the frequency domain where the maximum power occurs. This feature is valuable for precisely identifying the frequency bin associated with the highest power, allowing for a detailed analysis of the frequency composition of the signal.

3.1.14 BandPower

BandPower is a feature that quantifies the power or energy within a predefined frequency band of a signal. It involves integrating the power spectral density within the specified frequency range. BandPower is crucial for analyzing signal strength within specific frequency bands, providing insights into the distribution of power across different frequency components.

At the end by computing all these features we merge all of the temporal features in a single temporal feature array and all of the frequency features in a single frequency feature array.We do this for both Training data and Test data to do classification by efficient features not by original complex data. After that we have to do feature selection

3.2 Feature Selection

Fisher Feature Selection is a powerful technique used in machine learning and pattern recognition to identify and retain the most discriminative features for classification tasks. Named after the statistician and biologist Ronald A. Fisher, this method aims to maximize the separation between classes by selecting features that exhibit significant differences in their means while minimizing the variance within each class.

The primary objective of Fisher Feature Selection is to enhance the performance of classification models by emphasizing features that contribute the most to class separation. By reducing the dimensionality of the dataset to include only the most relevant features, Fisher Feature Selection not only improves computational efficiency but also mitigates the risk of overfitting.

The Fisher Score (F-score) is a key metric in this method, calculated for each feature based on the ratio of the between-class variance to the within-class variance. Features with higher F-scores are deemed more valuable for discrimination. The formula for the F-score is given by:

$$J = \frac{|S_b|}{|S_w|} = \frac{|\mu_0 - \mu_1|^2 + |\mu_0 - \mu_2|^2}{\sigma_1^2 + \sigma_2^2}$$

Figure 6: Fisher score formula

At the end we rank features based on their values of their fisher parameters and after that select 50 of best ranked features as efficient features. So we have now a 50D feature space and then we can easily and efficiently train our neural networks (MLP and RBF) to do the classification task.

3.3 Classification with neural netowrks

After feature extraction we have to train our classifier in final stage.

3.3.1 MLP Neural Network

We implemented a two-output Multi-Layer Perceptron (MLP) network for a binary classification task. First we give our feature exracted train and test data to the model. Few variables (11 different numbers) is considered to contain a range of neuron numbers for evaluation within the hidden layer of the MLP.

Within the inner loop, the dataset is partitioned using a holdout method with a 5-fold cross-validation strategy. For each fold, an MLP network with the current number of neurons is instantiated using patternnet(N). The network is then trained with the training data, and predictions are generated for the validation set. The accuracy of the model for each fold is calculated and accumulated.

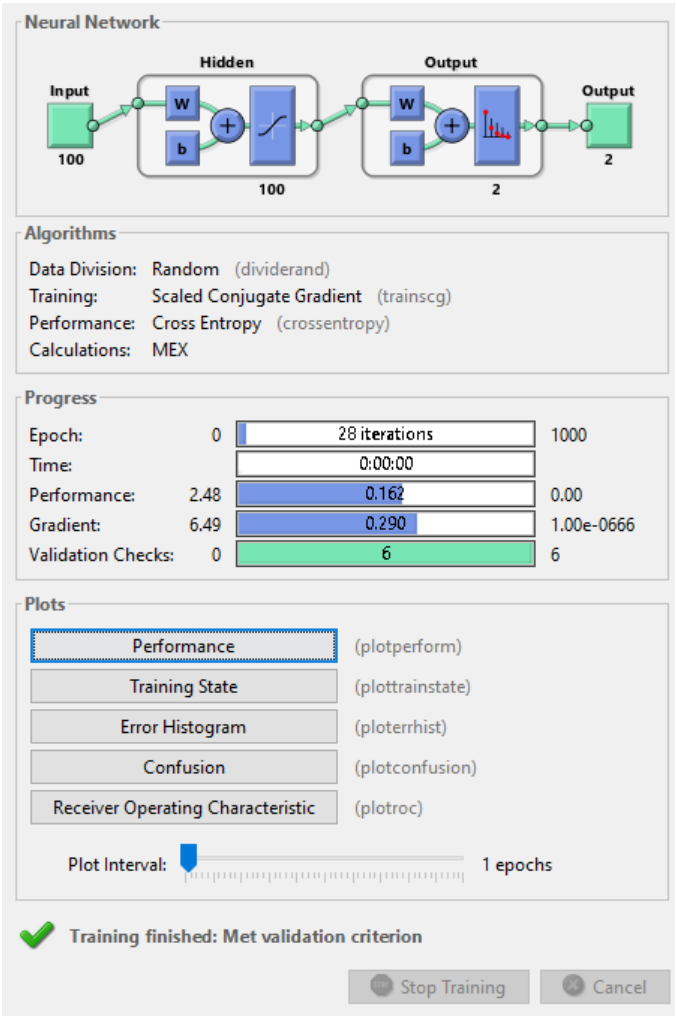


Figure 7: Training MLP

Here is the results of the MLP model we trained:

```
best_neuron_number =

    50

>> 100*max(accuracy)

ans =

   78.9091
```

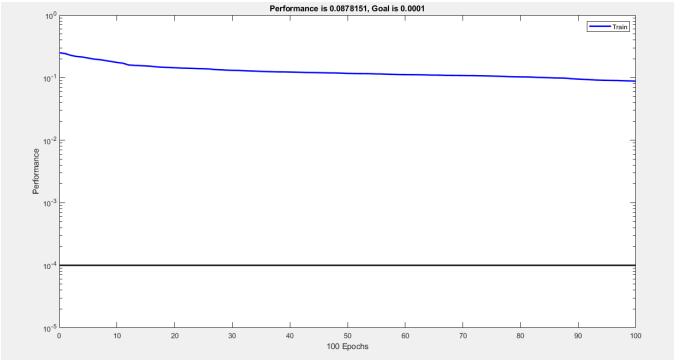
Figure 8: MLP Final Results

3.3.2 RBF Neural Network

We do the same for training radial basis function neural network to. after feature extraction we give the feature extracted train data to **newrb** instruction used for training RBF. 2 parameters have to be optimized.

- spread — Spread of radial basis functions
- MN — Maximum number of neurons

So we put different numbers for these 2 parameters and then train our RBF model and compute accuracy with 5-Fold cross validation. Here you can see the results for the model we trained.




```
best_neuron_number =

    60


best_spreed =

     5


ans =

    0.8127
```

4 Phase 2- Training Models with Evolutionary algorithm

First note that the feature extraction is the same with previous phase and also we use fisher score to rank the features.

4.1 Initialization of Evolutionary Algorithm

Chromosome is initialized as a binary matrix of size 20x200, representing 20 individuals (candidate solutions) with 200 features each. Each individual in the population is generated randomly by setting 100 feature positions to 1 (selected) and the rest to 0 (unselected).

4.2 Evaluation of Initial Fitness

The initial fitness of each individual in the population is calculated based on the selected features' ability to discriminate between classes (Fitness function). This is done for both time and frequency domain features.

4.3 Evolutionary Process (Mutation and Crossover)

The algorithm proceeds through a specified number of generations (generation_step = 200). For each generation, the best individual from the previous generation is retained. Mutation is applied with a probability of 5%, where a random subset of features is modified. Crossover is applied with a probability of 80%, where two individuals exchange information to create new individuals. Fitness is evaluated for the new population.

4.4 Results

Here you can see the results of the evolutionary algorithm we used and after that the training of MLP and RBF models.

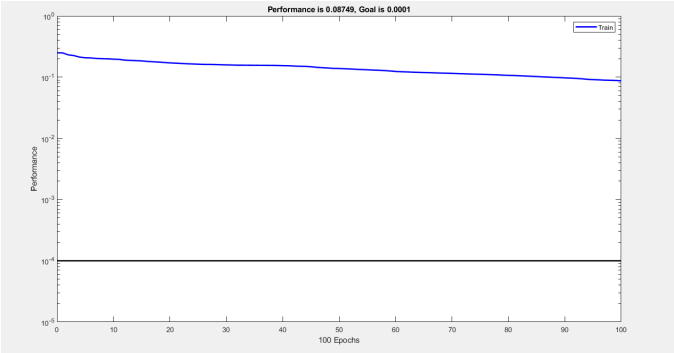
4.4.1 MLP with evolutionary algorithm

Here you can see the parameters and accuracy of the best MLP model :

```
best_neuron_number =

    70

>> max(accuracy)

ans =

    0.7364
```

4.4.2 RBF with evolutionary algorithm

Here you can see the results of the trained RBF model using evolutionary algorithm:




```
best_neuron_number =

   100


best_spreed =

    25


ans =

    0.8091
```

5 Final Results

Here you can see different accuracy of different models:

MODEL	ACCURACY(%)
MLP-Phase1	78.91
MLP-Phase2	73.64
RBF_Phase1	81.27
RBF-Phase2	80.91

Figure 9: Models Accuracy

Our findings show that when it comes to predicting outcomes from our EEG data, the model using Radial Basis Function (RBF) tends to do a better job compared to the one using Multi-Layer Perceptron (MLP). Think of the RBF model as being really good at understanding the tricky patterns and relationships in our brain data, thanks to its special design. It's like having smart experts in different areas of the brain, each focusing on specific details. On the other hand, the MLP model, while also good, might struggle a bit more with these intricate patterns. So, in simple terms, the RBF model seems to be more effective in making accurate predictions for our specific brain-related task.

The final Results of this project is the predicted labels for the test data. We saved our results in 4 file - 2 files for phase 1 (MLP and RBF) and 2 files for Phase 2 (MLP and RBF).