EEG Signal Segmentation

Mahdi Babaei
Department of Electrical and Computer Engineering, Temple University
mahdi.babaei@temple.edu

Introduction: The objective of this task is to apply machine learning techniques for analyzing raw time-series data, segmenting the data to detect events, and performing classification of each event in a 5-class space, including one background class and four classes of events. The training dataset provided for this assignment comprises 10,000 signals, with each signal containing essential information. Each training file consists of two sections. The first section includes the start and end times, as well as the class values for each event. The second section consists of a 1D feature, represented by a series of floating-point numbers, that emphasizes the pulses in the signal. The recorded signals are mostly sparse, meaning that a large portion of the data in each file is background noise, and the events occur sporadically. This document examines the application of machine learning techniques to address segmentation and classification problems. More precisely, this paper explains two machine learning approaches to segment and classify events for 2000 signals, where the first section of each signal is not included in the analysis.

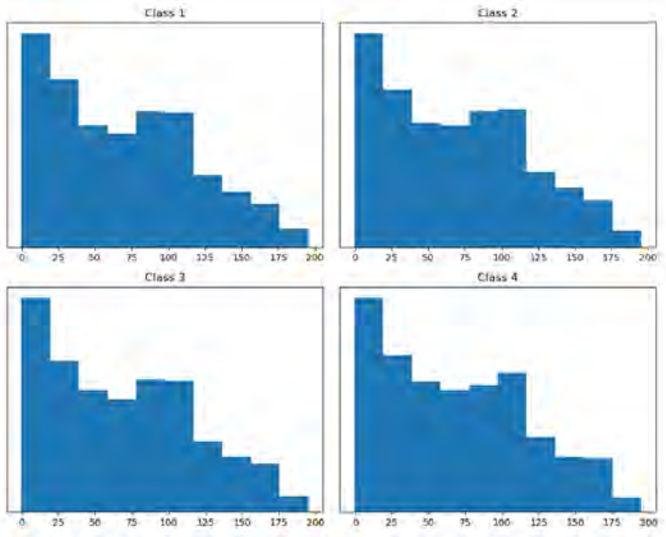

Figure 1. This shows the duration pulses of all classes.

After analyzing the training data, it became apparent that the duration of each class pulse is uniform. This makes it challenging to draw any assumptions about pulse duration. Figure 1 illustrates that the duration pulses for all classes are identical. Additionally, the meaning of each event indicates that there are no differences in pulse strength among different classes (Figure 2). This means that relying solely on pulse duration and feature values will not yield useful information. I have discovered that there is a direct relationship between the frequency of the event class and the amplitudes observed. Specifically, as the frequency of the event class increases, there is a corresponding increase in the amplitudes observed. I attempted to tackle the segmentation problem by breaking it down into two parts. The first part involved detecting event and non-event data through binary classification to identify the occurrence of events. The second part focused on classifying the events.

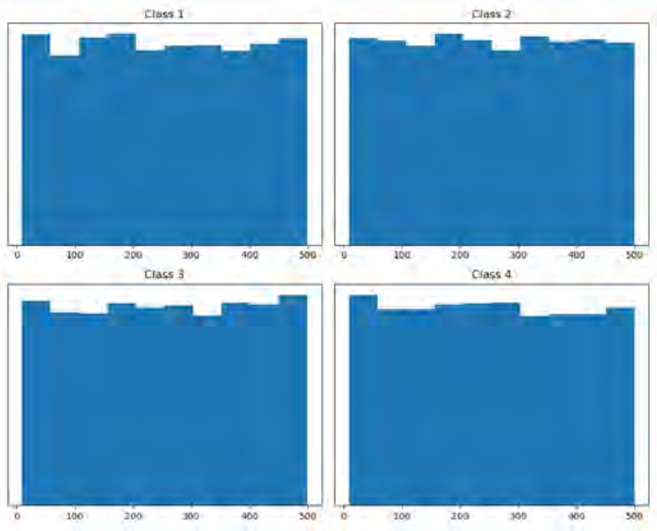

Figure 2. This shows the duration strength of all classes (mean).

RNF + LSTM: Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve the overall predictive accuracy and prevent overfitting. It is a supervised learning method used for classification and regression tasks. In a Random Forest, many decision trees are built using a randomly selected subset of features and training samples. Each tree makes a prediction, and the final output is the average prediction of all the trees. By using a combination of multiple decision trees, Random Forest can better capture the complex relationships between features in the data. I used Random Forest (RNF) to capture the event/non-event aspect of the approach. Given the training data, the RNF model outputs either 0 for non-event or 1 for event instances. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. The key idea behind LSTM is the use of memory cells, which can selectively retain, or discard information based on the current input and previous history. In LSTM, the memory cells are controlled by three gates: the input gate, output gate, and forget gate. The input gate controls the flow of information into the memory cell, the output gate controls the flow of information out of the memory cell, and the forget gate controls the retention or deletion of information from the memory cell. By adjusting the weights associated with each gate, the LSTM can learn to selectively retain or discard information from the previous time steps, allowing it to effectively model long-term dependencies in the data.

Subsequently, I employed an LSTM model for the second part of the implementation, to capture sequential and time-based information using the provided data on events in the training set. Although the minimum duration for an event is 9, a window with a size of 40 and a stride of 20 was used to slide over the data due to computational limitations. As previously mentioned in the introduction, the data is imbalanced, and I attempted several techniques such as under-sampling and oversampling. However, they could alter the underlying structure of event occurrence. Ultimately, I defined a lower class-weight for background data to influence the loss function. Class weight technique assigns a higher weight to the minority class and a lower weight to the majority class during model training, which helps the model to give more importance to the minority class and thus, improve its performance on the imbalanced data.

MLP + LSTM: MLP stands for Multi-Layer Perceptron, which is a feedforward neural network model that is widely used in machine learning. It is an artificial neural network that consists of multiple layers of nodes or neurons, where each neuron is connected to all the neurons in the previous layer, and the output of each neuron is passed through an activation function. The number of neurons in each layer and the number of layers in the network are hyperparameters that can be tuned to optimize the model's performance. The MLP model is trained using backpropagation, a supervised learning algorithm, to minimize the error between the predicted output and the actual output. Similarly, to the RNF approach, MLP was utilized for the initial binary classification task. This was done to capture the relationship between features and the event/non-event classes. Same as the previous approach, LSTM employed for the second part of the method.

I split the task into two parts: firstly, developing a model capable of distinguishing between background noise and the event classes, and secondly, developing a model to classify the events into their respective classes. For the first part, I generated input vectors of size 40 with a label of 0 or 1 using the first section of the data. Then, I trained MLP and RNF algorithms that are effective at identifying differences in binary classification tasks. However, the output data had overlapping events which required post-processing based on the previous and future events. For the second part, I utilized LSTM, a model well-suited for time series tasks that can learn features independently. I used a window of size 40 and a stride of 20 for sliding over the events to generate training data for both approaches.

Results: To apply the model to sequential datasets, a windowing function was created that slides a fixed window (size 40) across the raw datasets with a stride of 20. The resulting windows were fed into the binary classifier, and the output was post-processed based on the overlapping outputs. The same windowing function was then applied to the predicted events and non-events, and a further post-processing step was implemented to account for overlapping outputs and to smooth the sequential events based on information such as the smallest length of occurred events being 9. Table 1 shows the results. These performances are computed using the following:

$$P = avg[TPR\ for\ C1 - C4] - avg[FPR\ for\ C1 - C4]$$

Equation 1. The equation for the scoring metric of the classification algorithms.

Conclusions: This assignment explored the segmentation of a time-series signal and the classification of events that occurred within it. A hybrid RNF + LSTM model was utilized, followed by a DNN approach using MLP+LSTM. However, due to time constraints and the complexity of the problem, larger window sizes and strides were chosen to produce fewer input data and speed up the process. While the RNF model could have been made more complex, computational limitations meant that parameter optimization was limited to a smaller range.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
RNF + LSTM	59.01	66.62%	-
MLP + LSTM	42.99%	42.07%	-

Table 1. The scoring performance for the classifiers based on Equation 1