Class Detecting System for Time Bound Signals

Joshua Cassell
Department of Electrical and Computer Engineering, Temple University
Joshua.cassell@temple.edu

Introduction: The goal for ECE 8527/4527: *Introduction to Machine Learning and Pattern Recognition*, final project requires students to utilize a non-neural network and a neural network algorithm to construct class detection systems using 10,000 train data sets and 2,000 dev data sets. The system is tested for performance on 2,000 blind eval data sets. Both systems must successfully classify class events 0 – 4, start time and end time of the event and give the confidence level of each prediction. My approach was to find features like signal duration and amplitude from the test data to set up decision parameters for classifying eval data.

Data Analysis The data presented was a 1D array with times and amplitudes of a signal. The train data contained 20,000 sets of data. Each set contains thousands of signals with different amplitudes. In the training data each set stated the number of events, event class, start, end time, and duration for each event. Each event class has various duration times, amplitudes and occurrences. When plotting the data set (see fig. (1)) with amplitude vs time, the start/stop times and events become obvious. However, the amplitude and duration for each event class are not the same. For instance, an event labeled class 1 can have a duration of 200 and 11, 88 …, and same goes for the amplitude. This makes it very difficult to find useful classification features.

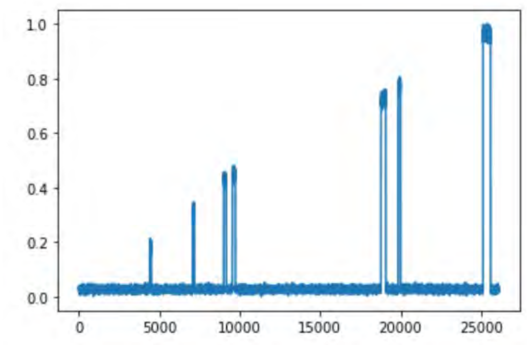

Figure 1: Train data with time vs amplitude graph.

Algorithm No. 1 Description: The classification method that I used was purely created from scratch using python in the jupyter notebook environment and will be referred to as System 1. To detect an event class, I used a sliding window approach to take in frames of instances. The frame size can be changed for optimization. When working with a frame of the data computation time is fast. For each frame my code finds the minimum and maximum amplitude values using the min() and max() python functions. The min amplitude is subtracted from the max amplitude and if the value is greater than the threshold value then we detect an event. The threshold value is a parameter that can be changed to better classify events. The threshold must be low enough to detect a difference in amplitude but also high enough to ignore any background noise. For each frame I detect the start and stop times for and event by looping through each amplitude and subtracting the previous amplitude by the current amplitude. If the absolute value of the difference is over the threshold value, then I detect a start/stop time at the time associated with that amplitude. To differentiate between start and stop time I use an if statement to determine if previous amplitude is greater or current amplitude is greater. A larger current amplitude will result in a start time and a larger previous amplitude will mean an end time. When start and stop times are determine then the duration is the stop time minus the start time and the average amplitude for that duration is calculated by summing all the amplitude in that time duration and then dividing by the time duration.

Using the extracted time duration and amplitude the next step is to have the system extract all the correct start/stop times, duration, and event classes that was stated in the train data and use this to make decision parameters. So far, my code only uses duration times mapped to the correct class from a single test file in order to choose class for the detected events. The decision is made by a simple if else statement that looks at the detected duration times and matches them to the proper class with that same duration time. If the duration time does not find any corresponding times, then the event is randomly given a class between 1 and then number of events detected by the system. If the system was to work perfectly it would give us the correct start/stop times, duration and amplitude values for our data, but the class classifier still uses only duration time a random guessing to detect the correct class of an event.

Proposed Algorithm Description: A Bayesian maximum likelihood classifier was supposed to be used to learn the similarities between class, time duration and amplitude values. The data outputted from the system explained above would have an extra feature which would be amplitude. If I could output a data set with the start/stop times, duration times, and averaged amplitudes for each event class, then I could make a PDF function for each duration and amplitude for each class. With the same system used above I would insert the maximum likelihood classifier and compute the likelihood of the detected duration and amplitude in order to determine the correct class for each event.

Results: I was not able to run my system on all the required training, dev and eval data sets so I could not score my results. In table (1) we have 0 percent for Train, Dev Test, and Eval sections. In table (2) I show the error rate computed for a single test file for which most of my system was modeled on. The error rate is 66.66%, which is pretty high. The high error rate is a result of the random guessing function for the event class. In this particular training data, there was no events for 5 through 9. My code was able to find the correct rise and stop times for a few of the events but when a misfire occurs, or the number of events is overshot then start/stop times are mistakenly assigned to the wrong event and that is why there is an empty row for the target values in table (2).

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
System 1	00.00%	00.00%	00.00%
System 2	00.00%	00.00%	00.00%

Table 1. Scoring Package has not been used for Systems Due to System Incompletion

Target Features			**Simulated Results**		
Event Class	**Start Time**	**End Time**	**Event Class**	**Start Time**	**End Time**
3	2851	3050	3	2851	3050
1	7382	7849	1	7382	7849
2	16046	16405	4	16046	16405
2	21559	21569	7	16404	21603
4	21570	21608	9	21560	21609
4	22253	22539	4	22253	22539
3	27585	27655	7	22504	27603
1	28191	28503	3	27586	27656
			1	28191	28504
Error Rate: 66.66%					

Table 2. Results from a single train file, the target features are the correct features for the training data. Simulated Results are features detected using class perditions from System 1.

Conclusions: The system that I have created is fairly decent at finding stop and start times, detecting events and computing time duration and average amplitude measurements for each event. The classification method used is fairly poor and works with less than 30% accuracy on the training data file it was optimized for. The maximum likelihood classifier which was not implemented in this project would probably not have had much impact on improving class detection accuracy. The likelihood classifier would only work well if there were a strong correlation between event time duration and amplitude, which can not be determined at the moment because correct event features have not been extracted for analysis. From looking at the data the duration and amplitude seem to be random. This system is not able to detect start\stop times if two signals share the same amplitudes at one signal's stop time and other signal's start time, which happens at least once in each data set resulting in problematic detection. To improve this system pre data processing would be helpful. If the data could be manipulated to show features that clearly coordinate to event classes, my approach would have been more successful.