ECE 8527 Introduction to Machine Learning and Pattern Recognition Final Paper
or:
How to Not Stick The Landing

Dakota Vadimsky
Department of Electrical and Computer Engineering, Temple University
Dakota.vadimsky@temple.edu

Introduction: For this final project, we had to use everything we have learned throughout this class to construct a classifier that could operate on time-series data which consisted of background data interspersed with events of varying amplitude, duration, frequency, and classification. We were intended to determine with a high confidence level when events started, ended, and what class out of 4 they belonged to. We were given 12,000 training and development files which included these annotations as well as the raw data, and an additional 2,000 files of only raw evaluation data. Although we were dealing with time series data, the sampling frequency of each file remained consistent at 1 sample/sec, and so I treated these waveforms as discrete instead of continuous.

My first step in trying to solve this problem was to preprocess the data in various ways to try to glean some information from it and make it easier to work with. By extracting the comments from each training file, I could easily compare the durations and relative position in each waveform of the various events, and then by applying the start and stop times of each event to their respective waveform I could find the mean amplitude of each event as well. I could also exclude the annotations related to background events to make this data easier to read. Next, I could normalize the data to fall between 0 and 1, and smooth it by setting the amplitudes of each point in each event to be equal to their mean amplitude. This process essentially removed any noise still in the waveform. I also rearranged the data to assign each point not just its time-series index but also its class.

Once I preprocessed the data in these ways, I attempted to segment the data using the sliding window approach. By using my annotations from before, I found the shortest window length for each file, and used that window to segment the waveform and classify each segment based on a majority vote of the annotations. I also calculated the mean for each segment as well. The problem I ran in to was that I did not know how to use this segmented information to teach a model anything about the data, and so after completing the segmentation it eventually sat unused. I had to find another way to try to classify the data.

Algorithm No. 1 Description: The first method I used to attempt to solve this problem was to use Dynamic Time Warping (DTW), pictured in Figure 1. DTW uses a dynamic programming algorithm to find the best path between two discrete, time-variant series. While Euclidean matching can be sufficient when the waveforms being compared are the same length, it is not enough when the waveforms are of varying length and the events within each waveform are possibly translated in one direction or the other. To ensure that the algorithm still takes the length of the waveform into account, it only looks for the next step of the path in a specified radius. This ensures that one waveform in which two events are spaced far apart does not get matched to one where the same events are grouped tightly together. The algorithm first matches the first and last indices of each waveform to each other. It then proceeds through the waveforms, finding the minimum cost at each step until it reaches the end.

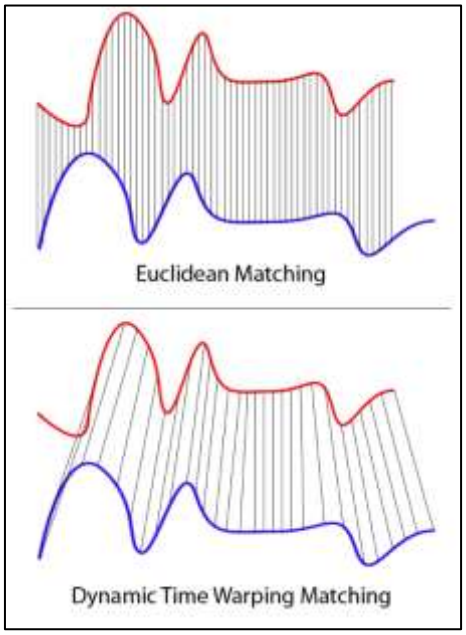

Figure 1: Dynamic Time Warping Example

This algorithm has a time complexity of O(N), which is not terrible, but given the sheer size of the training datasets, does make the job of matching waveforms excruciatingly slow. Reading in the evaluation files one at a time, I compared each to the training files, looking for the waveform that had the closest distance. Once this was found, I pulled the event data to see which class each event belonged to. Because the datasets were the closest matched by distance, they would have the same number of events. By reformatting the evaluation waveform into binary (0 for background noise, 1 for everything else), and then finding where the boundaries were between background and significant events, I was able to determine when events took place and what classes they belonged to. This process of course assumed that the evaluation data would in some way resemble the training data in terms of event placement and classification. It also did not allow for the easy calculation of a confidence level, as the comparison was not made event-by-event, but rather as an entire waveform. This calculation could theoretically be possible, as the distances between the evaluation waveform and all training waveforms could be placed in a Gaussian distribution and a confidence level could be attained using the mean and standard deviation of this distribution.

Algorithm No. 2 Description: For my second method, I attempted to classify the event data using an LSTM. I tried many different approaches to this, mainly revolving around trying to process the data in a way that could allow it to be fed into a neural network. I organized the data from the files into rows and then attempted to pad the rows with background event data, or truncate rows down to a minimum length, or do both to reach a middle ground. I attempted to remove the background so that I could work just on the event data. None of my approaches got me anywhere close to even creating a model, let alone a working model.

2,7067,7178

7064	-1.274346
7065	0.136502
7066	-1.154575
7067	-1.173123
7068	0.160399
7069	32.224335
7070	40.029019
7071	44.524368
7072	42.894937
7073	42.017912
7175	44.56683
7176	45.508448
7177	44.901111
7178	41.990279
7179	44.76587
7180	12.371036
7181	3.012012

Figure 2: DTW Prediction Vs. Actual Event in Raw Data

Results: As of the writing of this paper, I do not have results. My DTW algorithm is slowing working through the evaluation datasets. Running a test version on a handful of evaluation waveforms using only one folder of the training data showed that I could predict within an average of 4 time steps where events started and stopped. My assumption is that, once the algorithm has access to the entire training dataset, its prediction will be even closer.

Conclusions: This assignment made very little sense to me the entire time I was trying to complete it. I attempted to institute some sort of window sliding method, but I had no idea how to use that method to create an actual model that could then be used to classify anything. I could not see how the event classes differed from each other no matter how I rearranged the data. I even attempted to plot the data as an EEG, thinking that would reveal some patterns, but it did not. I thought that perhaps the event amplitudes were scaled as a function of their relative position in the waveform, seeing that the amplitudes increased over time. But when I attempted to scale the events based on this positioning function, they all wound up at roughly the same amplitude, again with no discernible patterns among the various classes. I tried everything I could imagine to extract some features from the events that I could then use to attempt to classify them, but nothing seemed to make any sense. I attempted to cluster the waveforms but seeing as how the events did not occur at consistent locations or with consistent frequency, this seemed to go nowhere. The many applications of time-series classifiers that I found online all proved fruitless as well. If there is any pattern to this data whatsoever, I would love to know what it is and how to find it.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
DTW	--	--	--
N/A	N/A	N/A	N/A

Table 1: Current Results