Signal Processing with Machine Learning

Mehdi Khantan
Department of Electrical and Computer Engineering, Temple University
Khantan.mehdi@temple.edu

Introduction:

Signal processing is involved in many devices we utilize in our daily lives. Voice recognition, health monitoring, radar, and wireless communications are some examples of technologies that use various methods of signal processing. In modern systems, a dominant part of signal processing is being done with the help of machine learning techniques. Data mining for signal data could be done by the means of different pattern recognition methods, measuring statistical properties of the signals such as mean, variance, covariance, and energy of the signal per time step. Furthermore, modern advancements in neural network techniques and deep learning, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) are widely used in data mining from different signals. The goal of this assignment is to process rime series by the means of machine learning. Signals need to be classified into four different classes (1 to 4).

In this assignment, three data sets are provided for training, development, and evaluation. Start, stop, and class numbers of the train and dev data sets are given as annotations and the goal is to find the start and stop times of the eval data set and classify the data between them to the four given classes. To process the signals, we need to dig into the signals and find a way to distinguish the differences between classes. As depicted in figure 1 there are some silence moments in the signal which are classified as class 0. Also, there are some pulse-like signals in between the silent moments which have been classified into classes 1 to 4. Pulse lengths are between 10 to 500-time steps and each class has been distributed uniformly. The histogram of class 1 is depicted in figure 2 as an example.

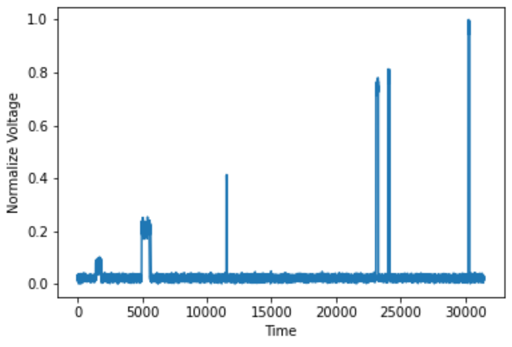

Figure 1-A sample of signals

To do this assignment, two approaches have been done. The first one is a classical approach and the second one is based on deep learning.

Approach:

To distinguish the classes, three aspects including the start time, stop time, and class type of the pulses needed to be found.

Start and stop time of the signals can be easily found by taking derivative of the signals. Since there may be some picks in derivative for the zero classes in the, it is better to get mean of some number of the samples and calculate the derivative over them. To make this happen, a sliding window at the size of half of the minimum length of the classes which is 10 in these data sets has been selected and moved through the whole signal. The results were close to the actual start and stop times with a plus-

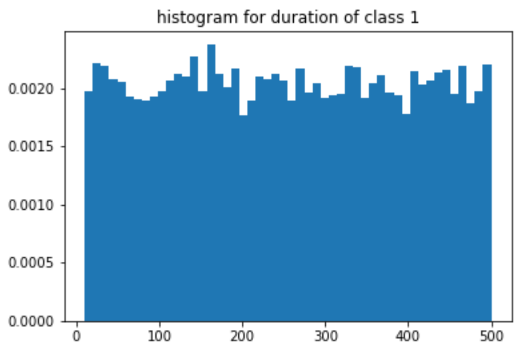

Figure 2: Histogram of time duration for class 1

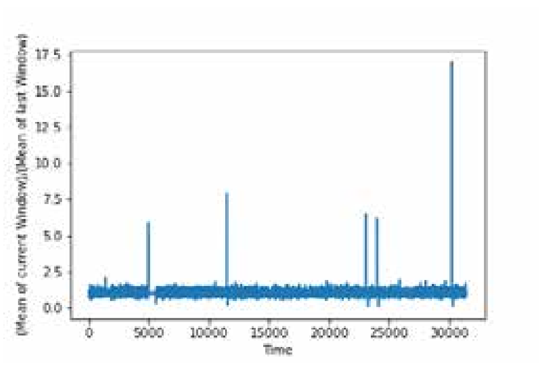

Figure 3: Derivative of the signal in figure 1

minus of 5 time steps. Making an overlap of 3 time steps on the sliding windows helped the accuracy of start and stop times. Figure 3 shows the theory behind this assumption. The calculation of the threshold for finding the start and stop time can be done by means of machine learning techniques. After finding the start stop times, the next step is finding the class of each pulse. Different features have been tested for these signals and the best accuracy had been found in using mean and variance of the signals between start, stop times. This also reduced the time of processing because the processing was done only between the start, stop times and not on the silent sections.

Accuracy on:	KNN n=1	KNN n=5	MLP	RNF	Ridge
Train	61.60%	44.40%	30.80%	27.20%	27.60%
DEV	36.60%	28.80%	24.40%	23.60%	23.70%

Table 1: Results

KNN Classifier:
K nearest neighborhood was used to classify the data to four classes. Since this algorithm relies on distance for classification, and the features were in different scales normalizing the training data helped to improve its accuracy dramatically. For n=1 the best accuracy was found in the train data set, but it was clearly overfitted. The results of the classification on dev data set proved the initial assumption on this. KNN with n=5 the results were 22.2% less accurate.

RNF Classifier:
Random Forest classification was also tested on the data sets and the results were 27.20% and 23.60% accuracy on the train and dev datasets respectively. To implement this classifier Scikit-learn library of python ahs been used. Best results were achieved with 10 number of estimations and maximum depth of 8.

Ridge Classifier:

Ridge classifier is based on the regression which on the first step converts the target values into {-1, 1} and then treats the problem as a regression task (multi-output regression in the multiclass case). With this classifier the results were 27.6% and 23.70% for train and dev dataset respectively.

MLP Classifier:
Multi-layer Perceptron classifier (MLP) has been used to classify the data by means of neural networks. For this classifier three hidden layers were used and the best accuracy occurred with 10, 20, and 88 hidden layers for the first to the third hidden layer respectively on 1000 iterations. The accuracy for the train dataset was 30.80% and for the dev data set was 24.40%.

Conclusion:
The results show that KNN can work better as a classification for these datasets. Also, another approach for these datasets can be convolutional neural network (CNN). An initial trial of using CNN on TensorFlow library has been done and results were acceptable. TensorFlow was developed by the Google Brain team for internal Google use in research and production and was released under the Apache license in 2015.