Machine Learning on 8 Channel EEG Data

Leo Berman
Department of Electrical and Computer Engineering, Temple University
Leo.Berman@temple.edu

Introduction: An electroencephalogram (EEG) is a multi-channel signal which describes the electrical activity in the brain via voltages measured in a variety of locations on the scalp. EEG recordings can be interpreted using montages, which redefine channels as the difference of channel voltages. EEG recordings are most commonly stored as raw signals in the European Data Format (EDF). Existing EEG visualization tools, such as EDF Browser and EEGLab, do not allow users to annotate directly over their signal displays. Furthermore, it is not possible to easily add new visualizations to these tools. Our tool displays annotations in a time-aligned format, and allows the direct creation and manipulation of these annotations. We provide an extensible framework that allows for the creation of new visualizations or analytics based on user needs. In addition to the conventional multi-waveform viewing capability to which neurologists are accustomed, we provide a spectrogram and/or energy visualizations. These visualizations are becoming increasingly popular with clinicians as an efficient way to review continuous EEGs (cEEG). In this presentation, we will introduce a software tool that facilitates annotation of EEG signals.

Algorithm No. 1 Description: The approach I took to generating a system is to use a Convolution Neural Network. The hyper parameters I used were 50 filters with a filtersize of (20,1). To my understanding, this will give a small frame that should capture a lot of the detail in the 8 signals while maintaining the granularity of the 8 channels for as long as possible. I used a learning rate of .001 as .01 converged to soon and was overtraining on the train set. I used softmax activation since I had all the labels into one model. Then I put the layer through a two dimensional max pooling with stride 2 in order to reduce the prediction down before flattening. I then added two Dense layers to complete the neural network and reduce it down to the number of features.

Algorithm No. 2 Description: Unfortunately, I did not complete a second algorithm so I will talk about the other algorithm I was initially trying. I attempted to reard the signals and translate them into images. I tried resolutions of both 640x480 and 1280x960. In both cases. I then took these images and tried to train a ResNet50. While I wasn't able to get hard numbers for these layers since my system didn't converge, I was using a two dimensional pooling layer followed by a dropout layer. After this, I put it through a Dense layer with output the size of 64 (number of classes).

Results

	Dataset		
Algorithm	Train	Dev	Eval
CNN	94.80 %	94.74 %	TBD