Julia Final Project: ECG Beat Classification

Sandya Subramanian

HST, 2nd year Ph.D. student

Overview: The main goal of my project was to take the MIT-BIH Arrhythmia database, the most commonly used publicly-available database for ECG, and use Knet to do beat classification. When doctors look at an ECG, they can identify specific morphological characteristics in the shape of the beats that allows them to determine what is wrong with the patient's heart, if anything. Doctors don't necessarily look beat by beat, because they can take in the whole picture in an instant. However, the first step for a machine to be able to even approach the same feat is to classify individual beats. My main goal was two-fold: 1) to determine accuracy for binary classification of normal vs abnormal and for multi-class within abnormal and 2) to assess if encoding time dependencies using an LSTM improved ability to classify ECG compared to a simpler method such as MLP. My results showed that for both tasks (binary and multi-class), and MLP alone with 4 layers was able to achieve >98% test set accuracy, and using an LSTM did not improve accuracy, but simplified the structure of the network required (fewer layers). Specific challenges included 1) manually converting the data set to a format compatible with Julia, 2) importing and curating the data set, and 3) installing a GPU and getting Knet to work in a matter of 5 days.

Datasets: I used two databases, primarily the MIT-BIH database, which has data from 40+ subjects, each for half hour, sampled at 350 Hz. The second database is one of primary normal ECG with some motion artifact, that is much smaller snippets (8 seconds long) from 9 subjects, each standing, walking, and jumping. From this I had the separate out the individual beats using standard ECG analysis algorithms such as Pan-Tompkins. The original paper reference for the Pan-Tompkins algorithm is in the Papers folder. The data is in the form of CSVs, after I converted everything over from the original edf format.

MIT-BIH Database
100.csv – actual data consisting of 3 columns: time, channel 1 and channel 2
100_ann.csv – timestamps of annotations
100_anntype.csv – types of annotations (single character code)

Normal ECG with Motion Artifact
test01_00s.csv – standing
test02_45w.csv – walking
test03_90j.csv - jumping

Notebooks: There are six Jupyter notebooks as part of this project. All of the html versions are saved in the Presentation folder.

1. Pan Tompkins Alg – the Pan Tompkins algorithm for beat identification in ECG signal (the original reference paper is in the Papers folder). The Pan-Tompkins is the most commonly used algorithm in clinical settings for ECG analysis.
2. Individual Beat Analysis – separating beats based on the identified peaks and looking at individual beats
3. Curating the Dataset – Turning the individually separated beats into a dataset format with truth labels from the annotations. This also includes combining datasets from multiple subjects for both binary and multi-class classification and splitting into training and testing sets.
4. Classification with MLP – the actual classification using multi-layer perceptron
5. Classification with LSTM – classifying using LSTM, which encodes time dependencies
6. LSTM for Synthetic Data – for fun, using the LSTM like in the assignment to generate synthetic ECG data

Scripts: There are several additional jl files I wrote to take care of more mundane parts of the process. They are also in the folder.

1. Findpeaks.jl – algorithm to find local maxima in signal where you can specify additional conditions like minimum peak height, peak distance, and number of peaks
2. Pan-tompkins.jl – the whole Pan-Tompkins algorithm from the first notebook in one piece
3. Run_pan_tompkins.jl – runs the Pan-Tompkins algorithm for a long signal in 10-15 second chunks
4. Get_breaks.jl – Using the identified peaks, gets the breakpoints for separating beats
5. Get_dataset.jl – Combines a single MIT-BIH subject data into a dataset of their beats with truth labels
6. Get_comb_dataset.jl – combines data from multiple MIT-BIH subjects into datasets with truth labels as integer categories for binary and multi-class classification separately
7. Get_traintest.jl – separates datasets into training and testing sets based on specified proportion that should be test set

Results and Conclusions: For ECG data, it is not necessary to encode the time dependencies, but doing so greatly simplifies the model, especially in the case of binary classification.