ECG Event Classification using Machine Learning

Shane McNicholas
Department of Electrical and Computer Engineering, Temple University
shane.mcnicholas@temple.edu

Introduction: This project aims to create two machine-learning systems for predicting heart conditions from ECG signals. The ECG data consists of eight channels and can output six classes: 1dAVb, RBBB, LBBB, SB, AF, and ST. Recordings are 7.33 seconds long and sampled at 300 Hz. Various machine learning techniques will be utilized, including preprocessing steps like normalization and feature selection. Adjusting hyperparameters during model training is crucial for performance. Postprocessing of model predictions will also be explored for optimization. The paper will detail the project's techniques and discuss the chosen models.

Before incorporating data into a model, it is crucial to clean the data through preprocessing steps. This project uses raw signal data, which requires normalization and filtering to remove noise and ensure the model is trained on the critical features. Robust scaling is an insensitive normalization algorithm that uses features like median and interquartile range to normalize each sample, resulting in most data being centered around the mean but keeping outliers.

Feature selection is a crucial preprocessing step in machine learning, as it plays a crucial role in the model's performance. Neural network algorithms can extract features from normalized signal data, but non-neural network algorithms may need to reshape the data for more efficient input. After normalization, each channel is concatenated onto each other, resulting in a vector with eight times the number of samples in the data.

Various methodologies are used to extract features from time-series data, but this project opted to keep the signal in the time domain and extract statistical features such as mean, standard deviation, minimum and maximum amplitude, peak-to-peak range, and power spectral density mean and total power. These features aim to have statistical significance to the data and have a relationship with one another and the output labels.

Algorithm No. 1 Description: This project uses a Random Forest estimator, which uses a set number of decision trees to model data. Each tree produces an output, and the outputs are averaged to determine an overall prediction. Random Forests don't overfit, but they are heavily dependent on the features provided. If features don't correlate enough, they may perform less than ideal. Random Forests also rely on their hyperparameters less than neural networks, which can be tuned. Despite this, the hyperparameters are unlikely to break the system. Overall, Random Forests is a simple algorithm with competitive performance, making it an effective candidate for this project.

The input to the Random Forest is a feature matrix, seven features long. The data for this project can have multi-label outputs, meaning that a single sample can belong to more than one class. The SKLearn Random Forest model cannot inherently produce multi-label output. The compromise is to train a Random Forest binary classifier for each output label independently of one another. When decoding with the RF, each sample will go through the six estimators for each label. Outside of the multi-label output, it is necessary to take note of the model's hyperparameters. For this project, all hyperparameters were set to SKLearn's default values. However, parameters such as the criterion function and amount of decision trees in a forest can affect model performance. For this project, the final parameters for the RF featured a Gini criterion function with a hundred decision trees per forest.

Algorithm No. 2 Description: The second model evaluated during this project is a Multi-Layer Perceptron (MLP). An MLP is a basic artificial neural network that can have one or more hidden layers. The model begins with an input layer, which, in this project's case, will be eight times the number of samples in a recording due to the concatenated signal data chosen for neural networks. The input layer then passes the

input vectors to the neurons on the hidden layer. Each neuron on the hidden layer is fitted to weight backpropagation. Once fitted, the values in the input vector are multiplied by the weights on the hidden layer's neurons. This idea continues for every hidden layer in the network until the output layer is reached. Unlike Random Forest, the SKLearn MLP can make multi-output predictions that are out of the box. In the case of this project, the output layer of the MLP is six neurons, one for each class. Like most neural networks, MLPs are very sensitive to their hyperparameters, so they must be appropriately tuned.

There are plenty of different hyperparameters for SKLearn's MLP API. Most of the time, performances tended to be similar despite changing the parameters. However, the parameters did affect how long it took for the model to converge. The activation function, which enables complex modeling, was tested at "logistic" and "ReLU," where ReLU yielded the best results. Similarly, the fitting optimizer worked best with adaptive moment generation (ADAM) rather than any of the other functions. Surprisingly, hidden layer architecture did not have a significant impact on the results of the model. Many combinations were tried with varying hidden layers and neuron counts, but none stood out. Ultimately, a hidden layer architecture of (100, 50, 25, 10) was chosen for simplicity's sake. However, the more neurons in the network, the longer the training times were. Other hyperparameters, such as learning rate, are also essential to consider. Fortunately, the ADAM optimizer handles many of the user's hyperparameters.

The postprocessing step of thresholding was added to both systems before decoding to optimize performance. This step captures the prediction probabilities of each system, allowing a minimum threshold for each label prediction. If the probability exceeds the threshold, the label is present in the sample. The thresholds for both systems were tuned independently, with a single threshold used for all labels. The best results were achieved with thresholds of 0.55 and 0.15 for the MLP and RF, respectively.

Results: After tuning, both systems showed modest results, with the MLP showing competitive performance on the dev/ and eval/sets. Random Forest had the most consistent results, losing only 2% accuracy from training data to development and evaluation sets. The MLP had macro accuracies of over 90% for all three datasets. However, the F1 scores were not as impressive as for both systems. Random Forest had somewhat consistent macro F1 scores, with each set around 40%. The MLP had an F1 score of 90% for training data but quickly halved when decoding unseen data. The RF did not perform nearly as well as the MLP.

	Data Set 15 – Macro Accuracy		
Algorithm	**Train**	**Dev Test**	**Eval**
RF	87.26%	85.76%	85.71%
MLP	98.89%	90.23%	90.11%

Table 1. Macro accuracy performance of each algorithm on all three subsets.

	Data Set 15 – Macro F1 Score		
Algorithm	**Train**	**Dev Test**	**Eval**
RF	44.01%	36.37%	36.26%
MLP	94.61%	48.20%	48.10%

Table 2. Macro F1 score performance of each algorithm on all three subsets.

Conclusions: The performance of two systems, Random Forest and MLP, was impressive due to their simplicity and quick training time. The lack of performance in Random Forest is likely due to the selected features, which did not correlate well with each other or labels. To improve the performance of RF, a complete analysis of features in the data and extracting the ones with the highest statistical significance and correlation is recommended. For MLP, hyperparameter tuning and possibly reconfiguring the input layer are suggested. The MLP's performance was likely nearing its peak due to its simplicity, but a deeper neural network with different layers and pooling types may be necessary for optimal performance. Understanding the data for machine learning is challenging without prior knowledge of ECG signals. It is recommended that an engineer and a physician work together to gain expertise in exploiting the data. Although the systems can be improved, the results demonstrate the power of machine learning in signal processing.