# Identifying Arrhythmia from Electrocardiogram Data

Taylor Barrella*, Samuel McCandlish†

December 12, 2014

## 1 Overview

In this project, we use machine learning to determine when a person's heart is beating irregularly[1]. This condition is known as cardiac arrhythmia. To do this, we use data from an electrocardiograph, a device used to measure a trace of a person's heartbeat via electrodes placed on their skin. During a heartbeat, heart muscles electrically polarize and depolarize as they contract, and the machine records these events are deflections on an electrocardiogram (ECG) trace. Our goal is to analyze ECG data to determine whether or not arrhythmia is present.

A normal heartbeat occurs at a relatively steady rate, with the various chambers of the heart contracting for a certain amount of time and in the correct order. Significant deviations from the normal timing of these contractions indicates arrhythmia. This deviation can be classified by the specific area of the heart that is beating abnormally, the heart rate, or the physiological cause.

To identify arrhythmia, we have taken two separate approaches. The first approach, single-beat classification, uses basic knowledge of physiology along with a support vector machine (SVM) to analyze each individual beat in an ECG, reporting any abnormal beats. This approach was highly successful in analyzing an ECG from a previously-seen patient, and moderately successful for a new patient. The second approach uses neural networks to analyze the ECG in a more holistic way, without using any knowledge from physiology. This approach, while less successful, improved significantly over the baseline and appears promising for future work.

## 2 Data

For our analysis, we have used data from the MIT–BIH Arrhythmia Database [1, 2]. This database contains 48 half-hour excerpts of two-channel ECG recordings, of which we exclude five as described below. These were obtained from 47 different subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings (those labeled #1xx) are from a mixed population of hospital patients, while the remaining 25 (labeled #2xx) were selected to display less common arrhythmias that would not be present in a random sample. The sampling rate is 360 samples per second, with a reading made over a 10 mV range.

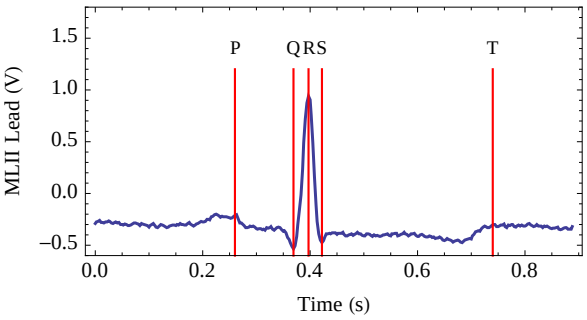


Figure 1: A beat can be decomposed into waves known as P, Q, R, S, and T, which correspond to polarization and depolarization of different parts of the heart.

Each record was annotated by two or more cardiologists, and the annotations were reconciled to yield a single annotation set per recording. Each beat is classified by type, and we group these classes into either 'normal' or 'abnormal.'

For our first model, some pieces of the data were discarded. Four of the 48 ECGs[2] come from patients using a pacemaker, a device that electrically stimulates the heart to beat normally whenever it fails to do so. Paced beats from these patients appear significantly different from normal beats, and we exclude them to avoid difficulty in patients without pacemakers. A more thorough analysis would include these records.

All of the records include recordings from two leads placed on the chest. One lead is always placed in the MLII (modified lead 2) position, while the other varies between the V1, V2, and V5 position. Because of the inconsistency in the position of the second lead, we only use the recording from the MLII lead. In addition, the lead used in record 114 is not clear, so we exclude it from our analysis.

*CS221/CS229; barrella@stanford.edu
†CS229; samsamoa@stanford.edu
[1]We thank CS229 TA Dave Deriso for this suggestion.

[2]Records #102, #104, #107, and #217

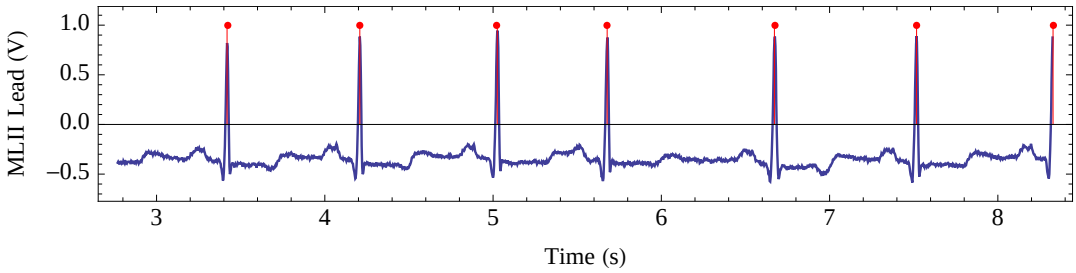


Figure 2: A sample of ECG data, taken from record #100. Detected beats are denoted by a red dot. The fourth beat shown occurred prematurely and is classified as APC, or atrial premature contraction.

# 3 Beat Classification Approach

Arrhythmia is typically diagnosed by detecting irregular beats in an ECG [3]. Hence, in this approach we will separate an ECG recording into its constituent beats and attempt to classify each beat as normal or irregular. Our goal is to use a minimal amount of information about the physiology of the heart to identify and classify beats using machine learning, rather than building a complicated model of each beat.

In this approach, we will build two types of models. The first model is an individualized model, built for a single patient. This would be useful when a person is being tracked routinely for arrhythmia, and a sample of the patient's data has already been evaluated by a cardiologist. The second model is trained on a collection of ECG records, with the goal of identifying arrhythmia in real time on a previously unseen patient. Both models use the same method, and differ only in the training set.

## 3.1 Features

For this analysis, we model an ECG record as a series of beats. Because a beat appears as a sharp peak in the ECG, we are able to extract a list of beats by finding deflections with slope above a pre-determined threshold. Given a peak, we extract a slice of the ECG signal centered at that peak with a pre-determined width. The collection of these slices forms a series of beats.

Each beat can be separated into a series of waves known as the P, Q, R, S, and T waves [3]. These waves represent electrical activity in the various parts of the heart. The R wave, which corresponds to depolarization of the main mass of the ventricles of the heart, can be identified as the sharpest peak in a beat. Other waves are identified as the peaks and troughs in the vicinity of the R wave.

Because we only want to use a minimum amount of physiological knowledge, we extract the following fairly generic features from each beat:

- Maximum and minimum voltage during a beat as well as their relative timing: This captures the main characteristics of the R wave.
- Delay between the current beat's peak and previous beat's peak: This captures the heart rate.
- Mean and mean-squared voltage of the beat: This captures the amplitude and duration of the waves.
- Maximum voltage derivative and its relative timing: This captures the speed of the R wave.
- Mean absolute derivative and mean-squared derivative: This captures the total amount of deflection.

Using only these simple features, we are interested to see how well our model can 'learn' the complicated set of rules used by cardiologists to identify abnormal beats.

## 3.2 Model

We use a support vector machine to classify each beat in a given patient's record as normal or abnormal. To use the highest dimensional effective feature space possible, we used the radial basis function kernel $K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right)$, where $\gamma > 0$ is a tunable parameter. We also include a regularization parameter $C$, since we do not necessarily expect our data to be linearly separable.

We used the LibSVM library [6] to perform training and prediction. To do this, we first scaled our features to lie in the range $[0, 1]$, in order to use the radial basis function most effectively [6]. We then found the optimal model parameters $C, \gamma$ by using the 'grid.py' script included with LibSVM to do a simple grid search, minimizing the error for 5-fold cross-validation.

Using the method described above, we build two different types of models. For the first type, we train a "personalized" model SVM on a single patient record. We choose a random subset of 20% of the beats to be "test" beats, and the remaining 80% to be "training" beats. A typical record contains about 2000 beats. We train the SVM on the training data from a single patient record, so that the test beats can be classified as one would expect in a scenario of ongoing monitoring.

For the second type of model, we train the SVM on a collection of patient records. We choose a random subset

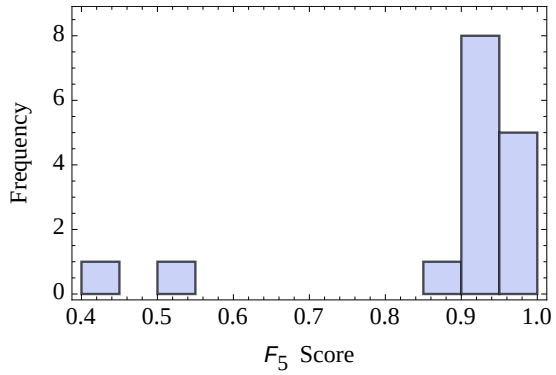


Figure 3: Histogram of $F_5$ scores for the 28 personalized models. All but two records, #202 and #222, were very successful.

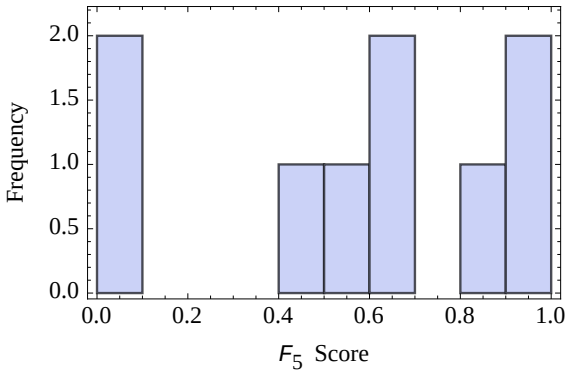


Figure 4: Histogram of $F_5$ scores for the "real-time" model, evaluated on 9 different patient records. All but two records, #212 and #232, were moderately successful.

of 9 of the patient records to be "test" records, and the remaining 34 to be "training" records. We train the SVM on all of the data from the training records, so that the test records can be classified as one would expect in a "real time" hospital scenario.

### 3.3 Results

The "personalized" models were very successful. We trained models for 28 of the records; the other 15 had either too few normal beats or too few abnormal beats to choose a useful testing and training set. To evaluate their performance, we use the F-measure, which is defined for real $\beta$ as

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \qquad (1)$$

In particular, we choose the $F_5$ measure, which is appropriate if we place roughly 5 times as much value in recall as in precision. In our case, false positives are much more acceptable than false negatives. We want to catch as many abnormal beats as possible, in case they are signaling an emergency. The $F_5$ measures are shown in Fig. 4. The models for all but two records, #202 and #222, scored highly. These records contain a large number of normal beats occurring in irregular rhythm, which are difficult to categorize. In future work, it may be beneficial to devise a method to consider normal beats with abnormal rhythm separately.

The "real-time" model was, as expected, less successful than the personalized models. Seven of the nine test records performed moderately well, while records #212 and #232 had perfect precision but very bad recall. Record #212 contained many abnormal beats that looked normal in the MLII lead; an model of both leads would likely have been more successful.

### 3.4 Analysis

The personalized models were surprisingly effective in this setup. This appears to signal that classifying a beat as normal/abnormal can be done relatively unambiguously when given prior access to a classified sample of the subject's beats. We expect that this type of classifier could be useful when a patient is subject to ongoing monitoring. A cardiologist could classify a small number of normal and abnormal beats, then allow the model to automatically classify the rest rather than tediously analyzing hours of ECG recordings.

The real-time model was less effective than the personalized models, but still worked well for many of the test records. The two records that it failed for, #212 and #232, were both in the group of recordings selected to display less common arrhythmias, so it is not surprising that the classifier was not as successful. Future work should include a method for including uncommon arrhythmias in both training and testing sets, perhaps by including ECG data from other databases.

As a further refinement of the model, it would likely be useful to use a sample-dependent regularization weight. This would allow us to ensure that rare forms of arrhythmia are not treated as anomalies by the SVM, and penalize their misclassification more heavily. Because our objective function does not take into account or preference for recall over precision, we expect that using weights would improve our results.

An interesting extension of this problem would be to classify abnormal beats by their type of arrhythmia. This could be done by training a multi-class SVM, which works by training multiple sub-models for each pair of classes. We leave this for future study.

## 4 Neural Network Approach

Instead of using "intelligent" learning based on our own knowledge of physiology, the focus of this section is to

begin to develop a deep learning algorithm. In particular, we have implemented a neural network, or multilayer perceptron (MLP).

## 4.1 Model

A multilayer perceptron uses logistic regression along with additional intermediate layers, called hidden layers. Here, we will be using a single hidden layer. The details of the MLP will be described after the data is explained.

For our current analysis, we take the first five minutes of each half-hour recording. Originally, the data for each example are a $650000 \times 3$ matrix of values. The first column is the time (in seconds). The second column is the reading of the upper lead (in mV). The third column is the reading of the lower lead (in mV). Because the time sampling was verified to be the same for each example, the first column has been stripped from the data. To shorten the sample from thirty minutes to five minutes, the number of rows has been truncated to 108000. Because there are two channels, there are thus 216000 parameters per example.

An MLP takes the input and "learns" the most relevant features. In addition to using the raw amplitudes as input, the model was also run using two forms of preprocessing: discrete fast Fourier transforms (FFTs) and discrete wavelet transforms (using PyWavelets [8]). We used both Haar wavelets and Daubechies (in particular, db3) wavelets.

The target prediction is whether or not the patient has arrhythmia as indicated by the recording data, i.e. 1 if arrhythmia is present and 0 otherwise. For determining this, the oracle is obtained by checking the records in the MIT–BIH Arrhythmia Database [1]. For each record, we check whether there are any beats before the five-minute mark that are classified as something other than "normal." If there are no such irregular beats, the example is labelled 0, for healthy. Otherwise, the example is labelled 1.

Because the data set is small, instead of using simple cross validation, we use 12-fold cross validation. We partition the training examples into 12 subsets $S_i$ of size 4. For each $i = 1, \ldots, 12$, the model is trained on $S_1 \cup \cdots \cup S_{i-1} \cup S_{i+1} \cup \cdots \cup S_{12}$ to obtain a hypothesis $h_i$. The hypothesis is then tested on $S_i$ to get an error $\hat{\varepsilon}_{S_i}(h_i)$. The estimated generalization error of the model is then calculated as the average of the $\hat{\varepsilon}_{S_i}(h_i)$. This generalization error is the metric by which we will evaluate our success.

Ideally, we would use leave-one-out cross-validation (LOOCV). However, training the MLP just once takes minutes. Due to our project timeline, we didn't invest the time for a four-fold increase in training time.

Here is a summary of what has been covered so far:

- Training data is a $48 \times 216000$ matrix of real-valued examples, along with a 48-dimensional vector of labels from $\{0, 1\}$.
- The model is evaluated four times: once using the raw data as features, once using discrete FFTs, once using discrete Haar wavelet transforms, and once using discrete db3 wavelet transforms.
- The model is a multilayer perceptron (MLP) with a single hidden layer. More details follow.
- The model is trained and then evaluated by using 12-fold cross validation.

A tutorial was used to implement the MLP [4]. This uses the Theano package for Python [5].

With a single hidden layer, the MLP works by making a prediction based off of an output vector

$$f(x) = G\left(b^{(2)} + W^{(2)} s(b^{(1)} + W^{(1)} x)\right). \tag{2}$$

The function $G$ is for logistic regression. To accommodate multiclass classification, $G$ is the softmax function

$$G(x; W, b)_i = \frac{e^{(Wx+b)_i}}{\sum_j e^{(Wx+b)_j}}. \tag{3}$$

The function $s$ is a nonlinear activation function for the hidden layer. Here, we choose

$$s(x) = \tanh x. \tag{4}$$

Finally, a prediction is made by

$$h(x) = \arg\max_i f(x)_i. \tag{5}$$

The parameters $W^{(2)}$ (a matrix) and $b^{(2)}$ (a vector) are weights. The additional weights for the hidden layer, $W^{(1)}$ and $b^{(1)}$, are called hyperparameters. These four sets of parameters are learned by training and using backpropagation to calculate the error (cost function). Theano is able to calculate gradients so that backpropagation doesn't have to be implemented.

Next we comment on the dimensions of these parameters. Two of the dimensions are given by the dimension of the input vector ($D = 216000$) and the number of possible classifications for the output ($L = 2$). The other dimension, $D_h$, is the dimension associated with the hidden layer. i.e., $b^{(1)}$ is a vector of dimension $D_h$, $W^{(1)}$ is a $D_h \times D$ matrix, $b^{(2)}$ is a vector of dimension $L$, and $W^{(2)}$ is an $L \times D_h$ matrix. We chose $D_h = 500$.

There are additional comments to be made about the details of the model. The MLP uses $L_2$ regularization, i.e., it tries to keep the $L_2$ norm of the weights low. Training is done using mini-batch gradient descent with a batch size of 2 and learning rate (step size) of $\eta = 0.01$. Five to ten iterations are made through the entire batch of 44 examples (4 being held out for cross-validation).

| | Raw Data | FFTs | Haar Wavelets | DB3 Wavelets |
|---|---|---|---|---|
| logistic regression | 0.521 | 0.5 | 0.521 | 0.521 |
| neural network | 0.25 | 0.229 | 0.25 | 0.25 |

Table 1: MLP errors after using raw data and three forms of preprocessing.

### 4.2 Results

Using the raw data as features, the estimated generalization error is

$$\varepsilon = \frac{1}{12}\sum_{i=1}^{12}\hat{\varepsilon}_{S_i} = 0.25 \tag{6}$$

after training the MLP. This is not very good, but it is an improvement over the baseline estimated generalization error of $\varepsilon_0 = 0.521$. The baseline consists of running logistic regression without the hidden layer (and also without $L_2$ regularization). The results after preprocessing are summarized in Table 1.

### 4.3 Analysis

The baseline error, $\varepsilon_0$, is high, and actually higher than what would be obtained by using a majority prediction rule on our data set (for the MIT–BIH dataset, most of the examples have arrhythmia). This is expected because the features, which are the 216000 raw amplitude readings in this case, are basically useless for a linear classifier. The point here is that even with useless features, using an MLP by adding a hidden layer and nonlinear activation function results in a lower error.

Preprocessing the data improved performance when FFTs were used. However, the two chosen wavelet decompositions did not help. The performance depended not on the type of input, but the number of parameters in the input. (For FFTs, the number of input parameters was half: real input results in an FFT with half as many complex parameters. The imaginary parts were dropped.) It is possible that the chosen wavelet decompositions were inappropriate for our data. FFTs may have been more appropriate because of the nature of the data (somewhat periodic graphs, where significant deviations from periodicity indicate arrhythmia).

Our implementation of the MLP did help, but overall it was still not very successful. Unfortunately, we did not figure out how to adapt other deep learning methods, such as convolutional neural networks (for image classification), to our data.

### 4.4 Future Steps

Given sufficient time, improvements could likely be made by continuing to adjust the learning parameters. A possible idea for helpful preprocessing would be to "normalize" the time-axis of the data. This would be done by setting the unit for the time-axis to be the average time between heartbeats for the sample, truncating each sample to be the same number of heartbeats, and adjusting the samples to have the same initial time offset.

## Bibliography


[1] Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. IEEE Eng in Med and Biol 20(3):45-50 (May-June 2001). (PMID: 11446209) http://www.physionet.org/physiobank/database/mitdb/, http://www.physionet.org/physiobank/database/html/mitdbdir/records.htm.

[2] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215]; 2000 (June 13).

[3] Ashley EA, Niebauer J. Cardiology Explained. London: Remedica; 2004. Chapter 3, Conquering the ECG. http://www.ncbi.nlm.nih.gov/books/NBK2214/

[4] http://deeplearning.net/tutorial/mlp.html

[5] http://deeplearning.net/software/theano/

[6] Chang CC, Lin CJ. LibSVM: A library for Support Vector Machines. March 4, 2013. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[7] Hsu CW, Chang CC, Lin CJ. A Practical Guide to Support Vector Classification

[8] http://pybytes.com/pywavelets/