Arrhythmia Classification for Heart Attack Prediction

Michelle Jin

Introduction

Proper classification of heart abnormalities can lead to significant improvements in predictions of heart failures. The variety of patient attributes that factor into arrhythmia classification and the number of resulting arrhythmia classes make this a complex problem to solve. Here, I use UCI's arrhythmia database[1] and apply a supervised learning algorithm, implemented in Matlab, to train and classify test data into 16 separate classes of heart conditions.

The dataset is contains 452 patients and 279 features per patient. The features include patient attributes such as age, height, weight, and gender as well as quantized ECG data such as the amplitude and width of the R, Q, and S waves. The patients are separated into 16 different classes, including one class of normal, 14 classes of various heart conditions (coronary heart disease, AV block, etc.), and one class of other. A small fraction of features are missing in the original dataset as well.

I implemented a random forests training algorithm combined with feature selection and data sampling techniques for maximum classification accuracy across all 16 classes. In general, the work presented here used 70% of the samples for training and 30% of the samples for testing.

Data Preprocessing

There are 408 missing attributes in the dataset as denoted by '?'. This corresponds to 0.32% of the overall dataset.

I tried several different methods to fill in this missing data. These methods included replacing missing features with '0's as well as the mean of each particular missing feature across all patients. The difference between each method was not observably significant in the final classification results, as the missing attributes did not represent a large portion of the data. Finally, the method I went with was to fill in each missing attribute with the corresponding feature of the patient's nearest neighbor by Euclidean distance. This seemed the most reasonable, although its results on accuracy were not significant.

Feature Selection

I implemented forward search feature selection based strictly on the correlation between each feature and the overall classification accuracy using a random forests approach. My feature selection algorithm used 10-fold cross validation to select the best 100 features out of the 279 presented. For feature selection purposes, accuracy was described as the number of misclassified patients across all classes/the total number of patients.

As shown in Figure 1, overall classification error rate decreased significantly up to the 10th feature added, after which there was no significant improvement with additional features.

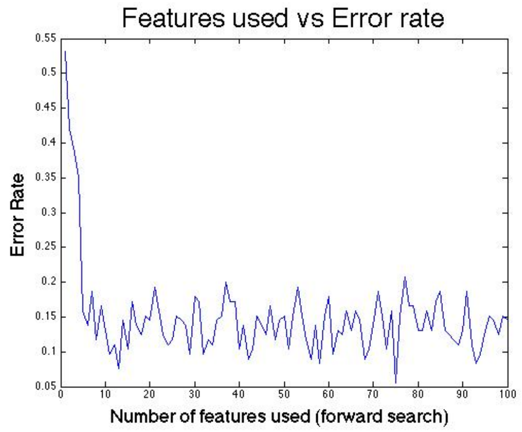

Fig. 1. Number of features used versus test error rate. Error rate no longer decreases with additional features after about the 10th feature added.

In this project, I did not find it to be the case that as the number of features added increased, the training error increased while test error decreased. Rather, training and test error plateaued after around the 10th feature. Both the testing and training error rate hovered around 15% overall misclassifications, signifying that this was not a variance/bias issue.

Here, reducing the number of features used through feature selection is beneficial not because of increased accuracy, but because of decreased processing time, which would have greater implications if similar prediction techniques was applied in real-time. As long as we have the 10 best features, we can be assured that accuracy results will not significantly improve with additional features.

Test and Training Data Sampling

Several traits of the original dataset heavily limit classification accuracy. Out of the 452 patients in the dataset, over half of them are in class 1 (normal), and several classes have five or fewer samples in them. The unbalanced nature of the dataset makes it much more likely that underrepresented classes are misclassified. In order to counter this effect, I preprocess the data through both sampling and skewing before any training and testing is done.

Sampling is done by taking the original distribution and resampling at random with replacement until I draw the same number of samples as the original dataset. Random sampling with replacement has the effect of redistributing the skew of the data to create more balanced classes. Although better than the original dataset, the sampled data still has underrepresented classes. Next, I skew the data by replicating each class that contains less than 10 samples. Thus, all classes that contain less than 10 patients have each of those patients presented twice in the dataset after skewing.

The class distribution of the dataset, from the original distribution, through sampling, and finally through skewing, is presented in Table 1.

Table 1: Distribution of data: original, after sampling, and after skewing.

Class	Labels	Original	Sampled	Skewed
01	Normal	245	235	235
02	Ischemic changes (coronary artery disease)	44	43	43
03	Old Anterior Myocardial Infraction	15	9	18
04	Old Inferior Myocardial Infraction	15	18	18
05	Sinus tachycardy	13	11	11
06	Sinus bradycardy	25	26	26
07	Ventricular premature contraction (PVC)	3	5	10
08	Supraventricular premature contraction	2	1	2
09	Left bundle branch block	9	5	10
10	Right bundle branch block	50	64	64
11	1. Degree AV block	0	0	0
12	2. Degree AV block	0	0	0
13	3. Degree AV block	0	0	0
14	Left ventricule hypertrophy	4	7	14
15	Atrial fibrillation or flutter	5	5	10
16	Others	22	23	23

The result on overall accuracy of this data manipulation is a decrease in the number of misclassifications by ~10%.

Next, I separate the training data from testing data. Following sampling and skewing, I randomly sample 70% of the skewed data without replacement for training purposes. 70% training is high enough that classification error rate remains low and low enough that there is enough test data left to represent all the classes. Then, every sample in the original dataset that is not part of the training class is selected for testing.

Note that after sampling and skewing, the data will have repeats. This only affects the training, and the data selected for training will include repeats so that underrepresented classes in the original dataset get enough representation in training. Every sample used for testing is unique.

Testing error decreases as I increase the percentage of data points used for training. The result is shown in Figure 2. Elsewhere in this report, percentage used for training remains constant at 70%.

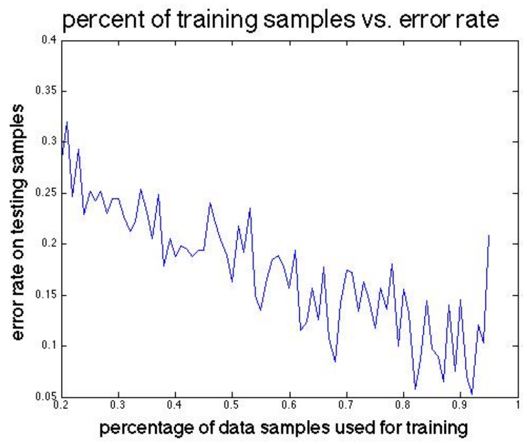

Fig. 2. Overall classification error rate on test samples vs. percentage of data used for training. This runs on a random forests algorithm using the top 20 features.

Training Model

The multi-class nature of this dataset calls for classification using a random forests algorithm. This algorithm contains multiple decision trees and selects the average of the all the trees' classification results as the final classifier output. The overall classification error versus the number of decision trees used is produced in Figure 3. This report uses the results from 50 trees.

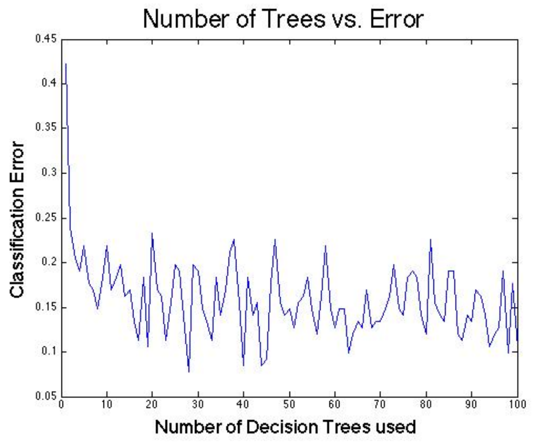

Fig. 3. Number of trees versus overall classification error

The dataset is multi-class, and it contains more features than patient samples. Furthermore, many of these 16 classes contain small numbers of patients, and patient attributes are often highly correlated. With these features, classification through other algorithms, such as SVM, naïve Bayes, and logistic regression did not produce as great results as the random forests classifier did. I explored both an SVM approach as well as a logistic regression approach (for differentiating between normal and all the other unhealthy classes), but overall misclassification error rate for those models hovered around 40% - 50% compared to the 15% error rate of random forests.

The ensemble of decision trees created from the training data is then used to classify the test data.

Results

Defining overall classification accuracy as simply:

$$\frac{\#\, total\ misclassifications}{\#\, test\ samples}$$

I found that overall accuracy hovers around 85% using 20 features and 70% training data across multiple runs. However, because the classes are uneven, overall classification accuracy as defined above may not be the best indicator of classification results.

In order to get a better understanding of accuracy, I also find the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) per class. Using these, I calculated the per class precision, recall, f-score, and acc (redefined). They are described as follows:

$$acc = \frac{(TP\ +\ TN)}{(P + N)}$$

,where P and N are the total number of positives and negatives per class.

Precision, recall, and fscore are by the traditional definitions:

$$Precision = \frac{TP}{(TP + TN)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$fscore = \frac{2 * (precision * recall)}{(precision + recall)}$$

The results are summarized in Table 2. The average of each is the per class result multiplied by the number of test samples for that class over the total number of test samples.

Table 2. Accuracy results by class

Class	Acc	Precision	Recall	F-score
01	0.8828	0.8333	0.9589	0.8917
02	0.9586	0.8000	0.6667	0.7273
03	0.9862	0.7143	1.0000	0.8333
04	1.0000	1.0000	1.0000	1.0000
05	0.9793	1.0000	0.5000	0.6667
06	0.9655	1.0000	0.4444	0.6154
07	1.0000	1.0000	1.0000	1.0000
08	1.0000	1.0000	1.0000	1.0000
09	1.0000	1.0000	1.0000	1.0000
10	0.9655	0.8571	0.9000	0.8780
11	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN
14	1.0000	1.0000	1.0000	1.0000
15	1.0000	1.0000	1.0000	1.0000
16	0.9724	1.0000	0.4286	0.6000
avg	**0.9280**	**0.8700**	**0.8552**	**0.8434**

Figure 4 shows the confusion matrix of how the classes are being misclassified:

Discussion

While the majority of classes have high accuracy scores, the lowest accuracy scores occur in classes 5, 6, and 16. These three classes all have low recall scores, meaning that out of all the patients who actually have the disease as marked by 5, 6, and 16, this algorithm has trouble classifying them as their respective classes. In fact, out of the 7 test cases that are in class 16 for the confusion matrix above, more are classified as class 1 rather than class 16 (4 vs 3). A large portion of misclassifications also all have the similarity that they are classified as 1 when the actual target class is something else.

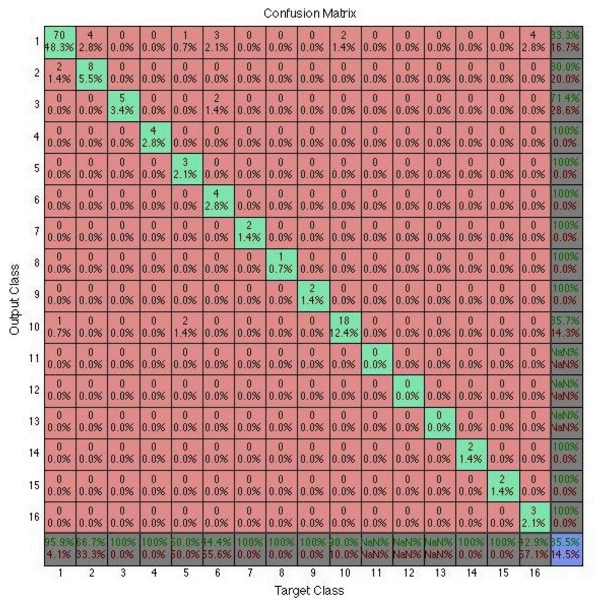

Fig. 4. Confusion matrix. For each class, this shows what the test sample actually is (target class) versus what it is being classified as (output class). Proper classifications are shown in the diagonal, and the overall accuracy rate is shown on the bottom right hand corner.

Conclusions

For this particular dataset, a random forests training algorithm provides the best classification results across all 16 classes. Due to the skewed nature of the original dataset, resampling and re-skewing the data to give more weight to underrepresented classes during training significantly increased all overall accuracy metrics.

The accuracy scores presented here are comparable to other learning algorithms on the same dataset as presented in literature. Typical accuracy metrics range between 70% - 90%, as summarized in Ozcift's paper [2].

Future Work

There are several things that could be done as future work.

First, out of the 279 features presented, many are correlated. Therefore, it is possible to explore dimensionality reduction algorithms, although in this case, the number of training samples exceeded the number of features present. However, there are other things that could be done because much of the data is correlated. For example, rather than viewing features like height and weight as separate attributes, it might be reasonable to combine them and present the pair as a weight-height ratio. Similar things could also be done with the ECG data.

Second, although not explored in this project, a cost matrix is also something that could be very useful to a problem like arrhythmia classification. The cost of misclassifying a sick patient as healthy, or misclassifying a sick patient as having an incorrect type of sickness, might be greater than the cost of misclassifying a healthy individual as sick. The medical implications of various types of misclassifications should be taken into account when designing such learning algorithms. Furthermore, implementing a cost matrix might also prove to be beneficial in improving accuracy, as placing a greater cost on often misclassified classes could improve accuracy results for those classes.

References

[1] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Akin Ozcift, "Random forests ensemble classifier trained with data resampling strategy to improve cardiac arrhythmia diagnosis." Computers in Biology and Medicine 41, 2011.

Note

This work was done partly in collaboration with Lee Tanenbaum. We submitted the project proposal and mid-project report together. However, because we each largely worked on our own standalone pieces of code and found it too hard to integrate in the end, we decided to write separate final reports. I have previously emailed the CS229 staff about this.