Classification of Arrhythmia using ECG data

Giulia Guidi & Manas Karandikar

Dataset

Overview

The dataset we are using is publicly available on the UCI machine learning algorithm. It can be found at https://archive.ics.uci.edu/ml/datasets/Arrhythmia. It consists of 452 different training examples and spans 16 different classes. Out of the 452 training examples, 245 is for normal people. We also have examples of 12 different types of arrhythmia. Among those types of arrhythmia, the most represented ones are classes 2 (coronary artery disease) and 10 (Right bundle branch block).

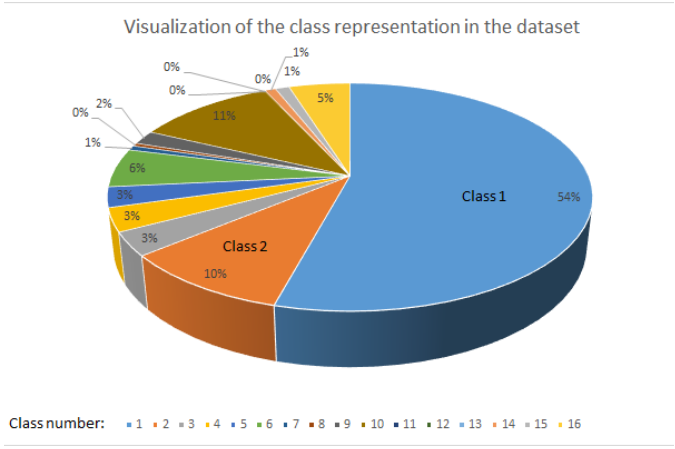


Figure 1: Class distribution in dataset

The 279 features include the age, the sex, the height and the weight of the person as well as some 'workable' information extracted from the electrocardiogram. This number of features is relatively high compare to the number of training examples available.
In order to understand the features we had to take a close look at how electrocardiograms are performed and what is measured.

As different parts of the heart (ventricles and the atria) contract, the electrical activity can be recorded. Thanks to multiple electrodes, 3D reconstruction of the waves is performed. All the pre-processing to convert raw data to workable vector angles/amplitudes has already been done by the creators of the dataset.
Each channel in the dataset is obtained by taking a voltage difference between two electrodes.

Approach to the problem

Prepare the dataset

The first step had to perform was to prepare the dataset. We noticed that some missing values being replaced by NaN were crashing the algorithms. Most of those values were found in feature 14. We also got rid of the rows containing NaN values.
With those changes, we ended up having 278 features and 420 training examples.

Feature selection

In biomedical applications, feature selection is very important as there are usually a lot of different parameters to take into account. With our dataset, we have 279 features and 452 training examples. With this ration of features to training examples, our algorithms can't be accurate. As it is

computationally expensive to enumerate and compare 2^279 (~9E83) models, we decided to perform feature selection with the help of trees classifiers. After running a tree classification algorithm we used only the features that were showing up in the tree.

Clustering

We created our training and testing sets by randomly sorting the indexes and spit ⅘ for training and ⅕ for testing. Each algorithm was run several times and the accuracy was the average each trial.

Algorithms

We explored four main algorithms: tree classifiers, SVM, naive Bayes and random forest.
The results of these algorithms are presented in the following paragraphs.

Decision Trees

Since the big advantage of decision trees is speed of computation and immunity to missing values, we ran CART algorithm in R and in Matlab to understand our feature and data set much better. To optimize the depth of the tree, we minimized the cross validation error as a function of the depth of the tree / number of leaves and then pruned the tree to that depth. A typical cross-validation error plot looks something like Fig 2.

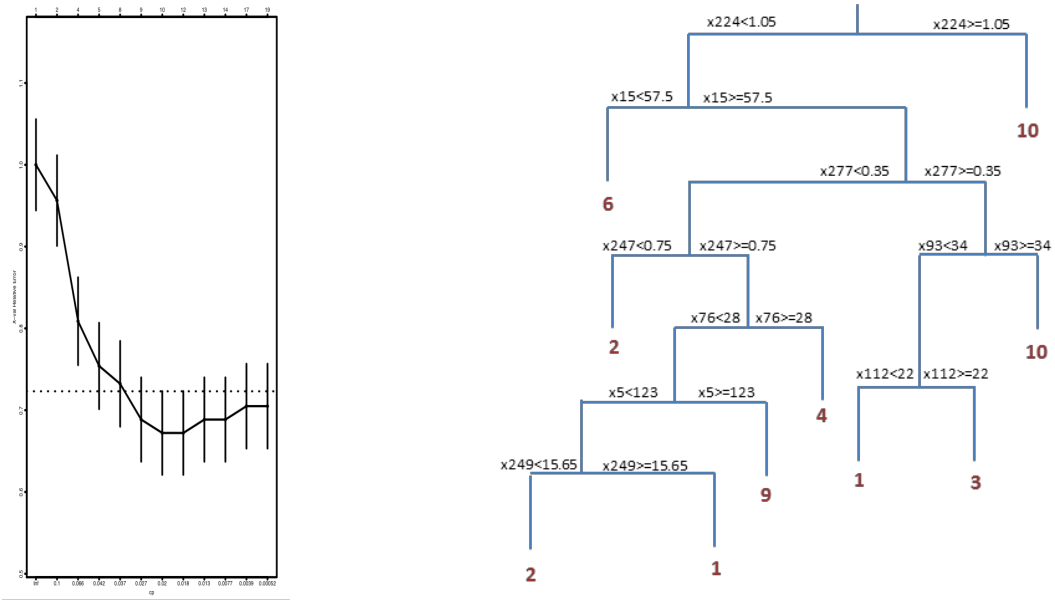


Figure 2: (i) Cross-Validation Error vs Depth of tree (ii) Decision Tree trained on dataset

The prediction accuracy of trees on the testing dataset was around 72%. This is expected to improve by implementing boosting methods like AdaBoost & Gradient Boosting.

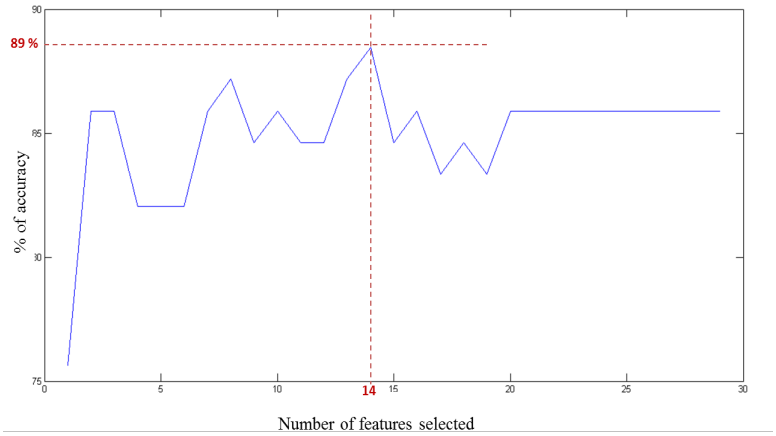
The classification tree constructed was a direct input for feature selection while running the next algorithm, Support Vector Machines (SVM).

Support Vector Machines (SVM)

Using liblinear on Matlab, we tried to run a SVM algorithm on the entire dataset (278 features and 13 classes). The accuracy of the model was only of 20%. This poor result can be explained by several factors. First of all the number of training examples that we used was comparable to the number of features. In addition, there were only 3 classes well represented in the dataset (class 1 for normal people, class 2 for artery coronary disease and class 10 for right bundle branch block). Every other class represented less than *5%* of the dataset (see Fig. 1)

In order to improve the SVM performance, we decided to combine feature selection and class selection.

As far as feature selection is concerned, we used the 30 most important features that we had with decision trees and we ordered them from most important (the first highest one on the tree) to least important.



We decided to extract from the dataset only the data for class 1 (245 training examples) and class 2 (45 training examples).

Starting with only the most important feature, once the SVM had finished to create a model, we added the following most important feature and ran again the algorithm. On each iteration, we computed the accuracy of the model and found that we could maximize the accuracy by selecting only 14 features (see Fig. 3).

Figure 3: Variation of accuracy with number of features

In addition to that, we implemented several types of kernels. We implemented a gaussian kernel and polynomial kernels of degrees 2, 3 and 5. The results can be visualized in Fig 4.We plotted for the 2 most important features (according to the decision trees) the separator as well as the support vectors. The kernel which gave us the maximum accuracy was the polynomial kernel of degree 3.

N° of features	N° of classes	Accuracy	Kernel
278	13	20%	-
278	2	14%	-
278	2	18%	polynomial deg 3
14	2	84%	-
14	2	85%	rbf
14	2	89%	polynomial deg 3

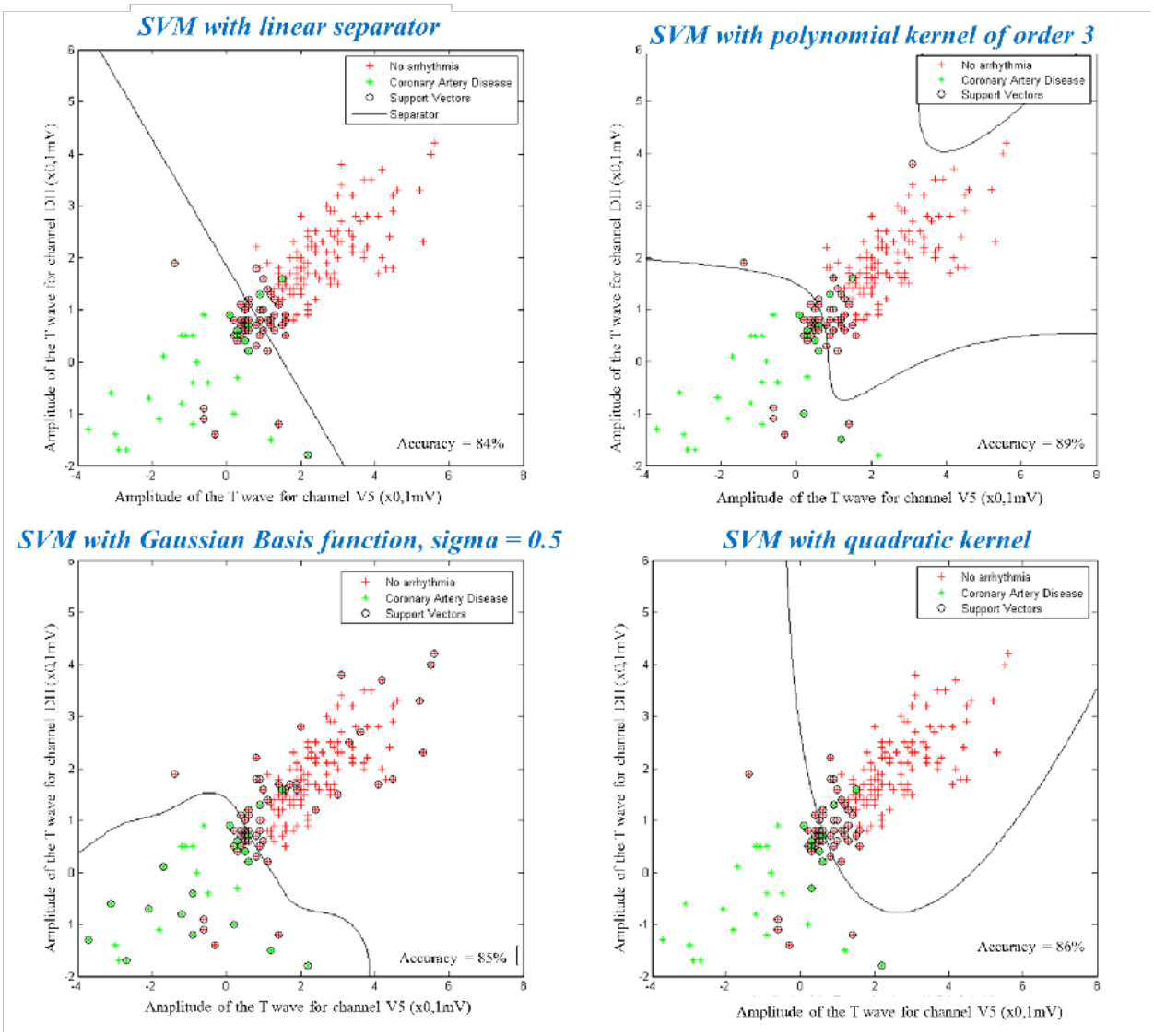


Figure 4: Visualization of 2D plots for SVM with different types of kernels

Naive Bayes and Random Forest

The naive Bayes algorithm relies on a strong hypothesis; the value of any feature is independent of the existence of any other feature.

In most of the real life examples, the naive Bayes hypothesis is never satisfied but the algorithm predicts with a good enough accuracy the classes.

The first step for us was to transform continuous data into multinomial. We used a kernel smoothing density estimate to model the probability density function of continuous variables.

The accuracy of the algorithm was on average 68%, which is fairly good for Naive bayes on such a large feature set.

We tried to use a random forest algorithm running a matlab version of the Brieman and Cutler's random forest package on fortran/R. After running the algorithm 20 times, we averaged the accuracy of each iteration of each model. We found that the average accuracy of random forest was 78%.

Conclusion

The accuracy & speed for various algorithms that we implemented is tabulated below.

Algorithm	Accuracy	Training speed	Testing speed
Decision Trees	78%	0.11s	0.001s
SVM for 2 classes and 279 features	80%	1.4s	0.033s
SVM for 2 classes and 11 features	86%	0.3s	0.011s
Naive Bayes	68%	8.5s	1.55s
Random Forest	78%	23s	0.1s

Note: training and testing speed are estimated with Matlab time summary. It is the time to execute the training or testing function.

As can be seen, classification trees provide a good accuracy with extremely fast computation time. The predictive accuracy is expected to be improved by implementing more complex boosting algorithms like AdaBoost and Gradient Boosting. SVM with feature selection gives the highest accuracy amongst all the algorithms implemented.
In spite of the fact that the dataset is immensely skewed towards a few classes and contains missing values, the implemented algorithms exhibit good level of accuracy in prediction. Performance is expected to significantly improve with a larger and more distributed dataset.

Contributions:

The idea for the project was conceived by Dave Deriso and was substantiated by the authors. The implementation and analysis of the algorithms has been done by the authors.

References:

[1] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and **C.-J. Lin**. LIBLINEAR: A library for large linear classification *Journal of Machine Learning Research* 9(2008), 1871-1874.
[2] H. Altay Guvenir, Burak Acar, Gulsen Demiroz, Ayhan Cekin "A Supervised Machine Learning Algorithm for Arrhythmia Analysis." Proceedings of the Computers in Cardiology Conference, Lund, Sweden, 1997
[3] Random Forest Matlab Package https://code.google.com/p/randomforest-matlab/