CS229 Predicting Heart Attacks

Sihang Yu, Xuyang Zheng, Yue Zhao

December 12 2014

Abstract

In this paper, the use of multiple machine learning algorithms for arrhythmia analysis is explored. We present different models built by multi-class supported vector machines (SVM), multi-class Nave Bayes (NB), decision tree and random forest. The performance of the various models in predicting the presence of cardiac arrhythmia and further classifying the instances into 16 pre-defined groups is tested and presented. The random forest classifier outperforms other algorithms with a test accuracy of 76%. We provide a discussion on the results of different models, together with some insight about of data set.

1 Introduction

Cardiovascular disease is the number one cause of death worldwide, claiming more lives than cancer and HIV combined. In this project, our aim is to distinguish between the presence and absence of cardiac, and also further classify one single instance into one of the 16 predefined groups. Class 01 refers to 'normal', classes 02 to 15 refer to different types of arrhythmia and class 16 refers to the rest of unclassified ones. We systematically investigated the multi-class classifiers built by the following machine learning methods: supported vector machines (SVM), Nave Bayes (NB), decision tree and random forest. Specifically, two different algorithms are employed to extend the binary SVMs into a 16-class SVM classifier: One-versus-One algorithm and One-versus-All algorithm. The One-versus-All algorithm is also used to build a 16-class NB classifier. Decision tree is a logic-based algorithm, which can directly classify the instances into 16 classes. Random forest is an ensemble method by constructing a set of trees and outputting the mode of the classes given by the trees. The performance of various methods are presented and discussed.

Our data comes from UC, Irvine's Machine Learning Repository. This database contains 452 rows, each representing a patient instance. There are 280 columns, the first 279 columns denote the 279 features and the last column is the classification (01-16). It also gives us the cardiologist's classification, which we take as a gold standard. We aim to minimize the difference between the cardiologist's classification and ours with the machine learning algorithms.

2 Feature Selection

We apply "minimum redundancy maximum relevancy" (mRMR) feature selection method, which has a better performance than the conventional top-ranking method, to filter out the irrelevant features as well as the highly correlated features.

In terms of mutual information, the purpose of feature selection is to find a feature set S with m features based on two criterions[reference]: maximal relevance(Eq.1) and minimal redundancy(Eq.2).

$$\max D(S,c), \quad D = \frac{1}{|S|}\sum_{x_i \in S} I(x_i; e) \tag{1}$$

$$\min R(S), \quad R = \frac{1}{|S|}\sum_{x_i, x_j \in S} I(x_i, x_j) \tag{2}$$

where $I(x_i, c)$ is the mutual information between the i^{th} feature and the classification, $I(x_i, x_j)$ denotes the mutual information between the i^{th} and j^{th} feature.

We define the operator $\Phi(D, R)$ to combine D and R and consider the following simplest form to optimize D and R simultaneously:

$$\max \Phi(D,R), \quad \Phi = D - R \tag{3}$$

For each model, we try to select different numbers of features and see which amount has the best performance.

3 Modeling

3.1 Naive Bayes

In this project, the features are continuous while the corresponding class labels are discrete (1, 2, ? , 16). Therefore, the standard Nave Bayes algorithm cannot be applied to classify these instances directly. One method to solve this problem is to discretize the features and then implement a multinomial Nave Bayes classifier. Another method is to combine kernel density estimation method into Bayes? theorem. The idea is illustrated in the following formula:

$$P(y=j|x_i) = \frac{\hat{\pi}_j \hat{f}_j(x_i)}{\sum_{k=1}^{K} \hat{\pi}_k \hat{f}_k(x_i)}$$

In the formula, $\hat{f}_j(x_i)$ is the estimated density at the value of xi. It is based on a kernel density fit and only the observation from the class $y = j$ is involved. $\hat{\pi}_j$ is the estimate of the prior probability of class y=j. The idea is essentially like a discriminant analysis without assuming normality. The nave Bayes classifier computes a separate kernel density estimate for each class of y based on the training data for that class. Both methods can allow Nave Bayes classifier to be implemented in this project, where the feature data is continuous. We choose to use the second method with kernel density estimation. Moreover, to extend the Nave Bayes algorithm into a multi-class scheme, the same one-versus-all design used in the Multi-class SVM is employed. 16 ?Class i-versus-All the rest? binary classifiers are trained. During the predicting phase, each binary Nave Bayes classifier returns a score indicating the probability of the instance being Class i. The final classification result is chosen to be the class that has the highest score (probability). The resulting accuracy of multi-class Nave Bayes classifier is 68.33%.

3.2 SVM

Firstly, two different designs of Multi-class SVM classifier are explained. One design is based on the One-versus-One algorithm and the other is based on the One-versus-All algorithm. Secondly, training and prediction procedures using Multi-class SVM (One-versus-One/One-versus-All) classifier are presented.

3.2.1 One-Versus-One

Each time, instances from Class i and Class j are used to train a ?Class i-versus-Class j? SVM binary classifier, which is referred to as i-vs-j classifier in the following paragraphs for simplicity. Therefore, for K-classes classification problem, there are in total K*(K-1)/2 binary classifier trained, i.e., 1-versus-2 classifier ? (K-1)-versus-K classifier. When predicting the class label for a testing instance, the instance is supplied to all K*(K-1)/2 classifiers, and each classifier predicts a class label for this instance. The final classification result is chosen to be the class that receives the most number of votes.

3.2.2 One-Versus-All

Each time all the instances are first re-divided into two groups: Class i and Non-Class-I, and then used to train a ?Class i-versus-All the rest? SVM binary classifier. Therefore, for K-classes classification problem, there are in total K binary classifier trained, i.e., 1-versus-not-1 classifier ? K-versus-not-K classifier. When predicting the class label for a testing instance, the instance is supplied to all K classifiers, and each classifier returns a score which indicates the probability of the instance being Class i. The final classification result is chosen to be the class that has the highest score (probability).

3.2.3 Training and testing procedures

In order to explain the training and testing procedures for each fold, the pseudo-code is presented below with explanations.
(A). For each fold (in total 10 folds):
1.Use 5 features to 250 features (step size is 5 features; thus in total run 50 times)
(i)Use Cross Validation method in the training data to select the best parameters (best penalty parameter c and Gaussian kernel parameter ? as shown in the formulas below) for the Multi-class SVM classifier
(ii).Train the Multiclass SVM classifier using the best parameters selected above with the training data.
(iii).Predict class labels on test samples and calculate the test accuracy.
2.Choose the best accuracy from 50 accuracies. This

gives the best accuracy that Multi-class SVM classifier can produce for this specific fold (with optimum feature size).
(B). Average the best accuracies from all folds to get an overall best accuracy that Multiclass SVM classifier can produce for this given data.

3.2.4 SVM Results

For the parameter selection procedure mentioned above, the penalty parameter c varies from 2-5 to 215 and the Gaussian kernel parameter varies from 23 to 2-15. A picture for a typical parameter selection procedure is presented below. We choose the best parameter pair based on the cross validation accuracies.

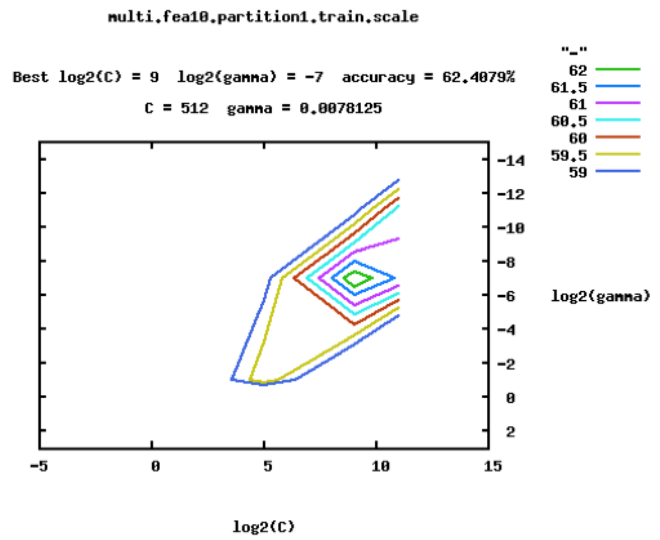
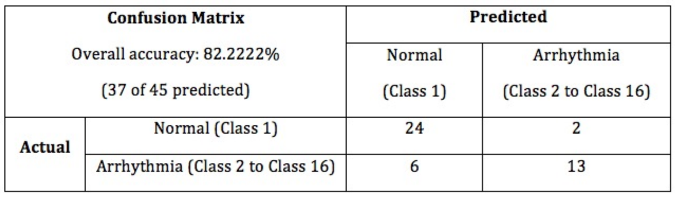


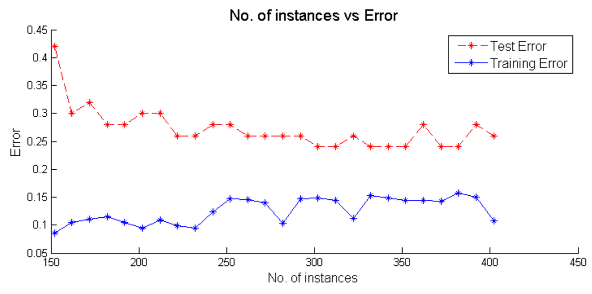
Figure 1: SVM parameter selection

The table below presents the confusion matrix for one of the classifiers used. Due to the limited number of testing samples, the 16*16 confusion matrix are mostly filled by 0s which posts a difficulty for reading and interpreting the results; thus the numbers from Class 2 to Class 16 are merged and are presented in the category Arrhythmia. From the table, we can see that the overall accuracy of 82.2% is not due to a large number of unbalanced data but the model actually captures the feature distributions for different classes and gives a fair prediction for testing samples. We might also notice that given an testing instance belonging to "Arrhythmia" category, the probability that it will be predicted as Arrhythmia is 13/19 or 68.4% , which is noticeably lower than the overall accuracy of 82.2%. One explanation is that the training samples for Classes 2 to 16 are remarkably fewer than those for Class 1; thus the characteristic of Class 2 to Class 16 might be less captured by the model than Class 1. True Negative rate (Arrhythmia classified as Arrhythmia) is expected to increase given more training data from Class 2 to Class 16.



Confusion Matrix Overall accuracy: 82.2222% (37 of 45 predicted)		Predicted	
		Normal (Class 1)	Arrhythmia (Class 2 to Class 16)
Actual	Normal (Class 1)	24	2
	Arrhythmia (Class 2 to Class 16)	6	13

The training and test errors versus the number of training instances for one of the classifiers are shown in the picture below. The big gap between training error and test error suggests that the method suffers from high variance problem and the test error is expected to decrease given more data.



To conclude for SVM method, Multiclass SVM methods using One-versus-One classifiers or One-versus-Rest classifiers present comparable accuracies of 72.67% and 70.89%, respectively. The accuracies for both classifiers are expected to improve given more training data, especially from Class 2 to Class 16.

3.3 Decision Tree

3.3.1 Grow the tree

Decision tree is a kind of logic-based algorithm, which is one of the most successful techniques for supervised classification learning. Figure.N shows a diagram of a decision tree. Each interior node corresponds to one of the features. The edges to children denote the possible values or value intervals of that feature.

Each leaf denotes the classification given the values of the input features represented by the path from the root to the leaf. Here are the main steps to build a decision tree: firstly choosing rules to split on, then, dividing the training instances with the rule into disjoint subsets, repeating recursively for each subset till the leaves are almost pure. For an ideal tree, the instances reaching the same leaf should be in the same class, which means the leaf node is pure. However, the ideal tree can cause severe over fitting especially in our case when the training date is extremely insufficient.

3.3.2 Optimize the tree

In this part, we set the option "prune" on, change the number of features, the complexity of the tree (minParent), and the split criterions (gdi, deviance or twoing), and calculate the corresponding accuracy. "minParent" is a number k such that impure nodes must have k or more observations to be splitted. The smaller the minParent value, the more complex the decision tree. In this way, we find the best feature size and minParent, which leads to the highest accuracy. Figure 1 shows the relations between feature size, minParent of the tree and test accuracy (using gdi as the split criterion, which has better performance than deviance and twoing). We find that this model has the best performance when minParent is 37 and feature size is 165, the corresponding test accuracy is 71.33%, training accuracy 79.7%.

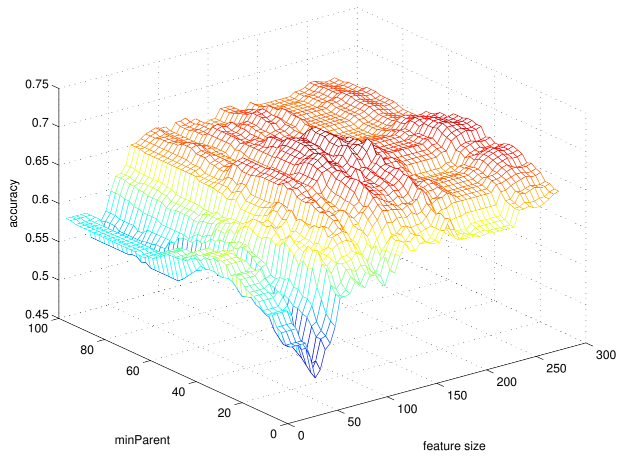


Figure 2: The relations between feature size, minParent and test accuracy

3.4 Random Forest

3.4.1 Grow the trees in the forest

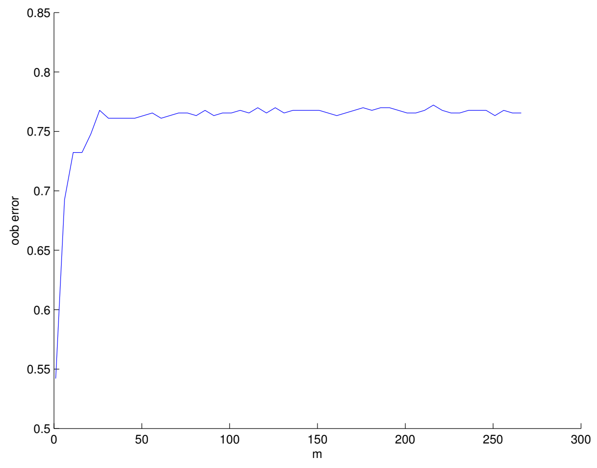
Random forest is an ensemble method for classification that operate by constructing a set of decision trees and outputting the class that is the mode of the classes output by individual trees. Unlike single decision tree, which is likely to suffer from high Variance or high Bias, Random Forests can find a natural balance between the two extremes. For each tree in the forest, we construct our training set by sampling with replacement within the training instances (bootstrap sampling). Besides, if there are M features, a number $m < M$ is specified such that at each node, m features are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing. Unlike single tree, in the forest each tree is grown to the largest extent, i.e. there is no pruning and minParent should be 1.

3.4.2 Error estimation

In random forests, there is no need for cross-validation to get an unbiased estimate of the test set error. The out-of-bag (oob) error estimation is usually used in this forest model. Since each tree is constructed by bootstrap sampling from the original data, some of the instances are left out and not used in the construction of the tree. Taking these left out instances as test set and predicting their classifications with the corresponding tree. At the end of the run, take j to be the class that got most of the votes every time the certain instance n was left out. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

3.4.3 Results

The following figure shows the test accuracy (calculated via the oob error estimation algorithm) varies with the number of features m with 1500 decision trees in the forest. We can see as m increases, the accuracy reach the limit about 77%.



4 Conclusion

Table 1: Testing accuracy for different classify

Model	Testing Accuracy
SVM (One-versus-One)	72.67%
SVM (One-versus-All)	70.89%
Naive Bayes	68.33%
Decision Tree	71.33%
Random Forest (mRMR)	76.89%

Without loss of generality, in the above table we use the test accuracy gained via 10-fold cross validation for the random forest just as the other models (the accuracy of random forest shown in the last section is calculated via out-of-bag error estimation algorithm). The test accuracy of random forest is significantly better than other models. The single decision tree needs to be pruned to get the balance between bias and variance, while random forest can achieve both low bias and low variance. Bagging and other resampling techniques are usually used to reduce the variance in ensemble methods. To make a prediction, all of the models in the ensemble are polled and their results are averaged. Random Forests works by training numerous decision trees each based on a different resampling of the original training data, in which numerous replicates of the original data sets are created. In Random Forests the bias of the full model is equivalent to the bias of a single decision tree (which itself has high variance). By setting prune off and minParent 1, we minimize the bias of a single tree. By creating many of these trees, and then taking the mode of their predictions, the variance of the final model can be greatly reduced over that of a single tree. In practice the only limitation on the size of the forest is computing time as an infinite number of trees could be trained without ever increasing bias and with a continual decrease in the variance.

5 Future Work

Firstly, the current results of our models are limited by the small amount of data and the missing data in some features. Incorporating more accurate and precise patient instances may improve our accuracy results.

Secondly, based on the models we have optimized, we can create our own ensemble methods, which may improve the overall performance.

Thirdly, we can try to improve our classifier to be able to work on real-time data such as ECG signal.

Reference

1. Peng H, Long F, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2005, 27(8): 1226-1238.

2. Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm