Disease Prediction

Yilong Zhao

Abstract

In order to predict the severity of a kind of disease using people's daily health information, three machine learning classifier algorithms and three classifier combiners are implemented in this project. After accomplishing Naïve Bayes, Decision Tree and SVM on the dataset, I made an effort to search more accurate results by implementing Majority Vote Combiner, Borda Count Combiner and Naïve Bayesian Combiner. The final result shows the combiners generally work slightly better. Despite the defects of the dataset, the final error rate is achieved at around 35% using classifier combiners.

1. Introduction

Healing the disease is the responsibility of the doctors, but even the doctors may sometimes need help. Besides, it will be very helpful if the patients can know before hand the potential disease they may have. Last but not the least, many severe disease may be prevented if the patients can do a regular check themselves using their normal health information. In this project, by using the previous physical history of condition data of the people and their severity of a specific kind of disease (Heart disease in this project) as training data, I implement several machine learning algorithm in order to predict the severity of this disease when get physical information from people.

2. Dataset Description

V.A Medical Center, Long Beach and Cleveland Clinic Foundation provide the dataset I use. After trimming the attributes of the dataset, the actual attributes used in the project listed in the Appendix A. There are 300 training data and 70 test data.

3. Approach of basic machine learning algorithms

3.1 Naïve Bayes Classifier

Naïve Bayes Classifier is based on Bayes Rules:

$$P(C \mid F1, F2, ..., Fn) = \frac{P(C) * P(F1, F2, ..., Fn \mid C)}{P(F1, F2, ..., Fn)}$$

Also presented as:

$$posterior = \frac{prior * likelihood}{evidence}$$

After we get the posterior probability, we can assign the class label which has the largest posterior probability to the test data.

Two assumptions are made during the implementation for simplifying the problem:
1. Assume attributes are independent from each other.
2. All attributes have Gaussian distribution.
These two assumptions are in some way unrealistic. First of all, since the attributes are all about the

information of patients' health, some of them may have some connections. One example is people tend to have higher heart rate when they have higher body temperature. Besides, almost all the attributes hardly have a Gaussian distribution; there is also no accurate way to learn the distribution type of the attributes based on the dataset.

Based on the assumptions above, the final formula used in the project is:

$$P(C \mid F1, F2, \ldots, Fn) = \frac{1}{Z} * P(C) * \prod_{i=1}^{n} P(Fi \mid C)$$

2.2 Decision Tree

Decision tree is a decision support tool using a tree-like model of decisions and their possible consequences. The basic idea is to find the attribute that can minimize the impurity of the dataset each time. The impurity measurement used in the project is information entropy:

$i(r) = -\sum_{c \in C} P_c^{(r)} \log P_c^{(r)}$ Where $P_c^{(r)} \equiv P(y = c \mid x \in R_r)$

Two stop criteria for splitting the decision tree are implemented:
1. Stop splitting if current node contains less than N examples
2. Stop if the decrease of impurity is less than M

In order to simplify the implementation, dataset is modified so that all the value of the attributes are either 0 or 1:

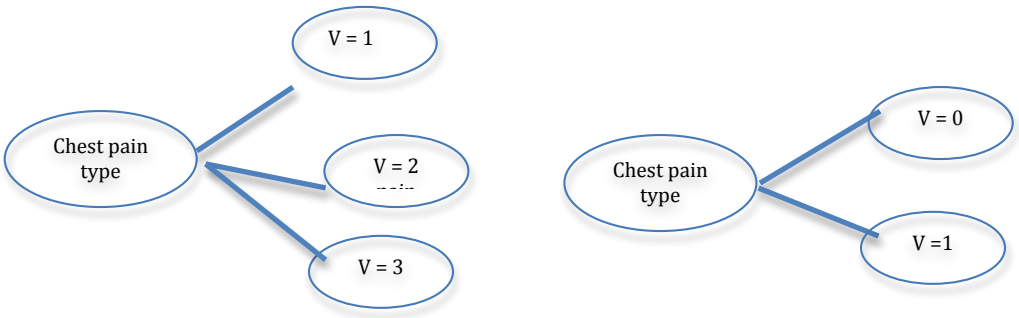


Fig 1 Arbitrary Decision Tree to Binary Decision Tree

2.3 SVM – Support Vector Machine

SVM classifier solves binary classification problem by trying to separate data points which have p dimensions with a (p-1) – dimension hyper plane. The best plane is the one has the largest margin between classes. Since it is only designed for binary classification problem, one-to-one mechanism is implemented in order to expand the algorithm to multi-class classification problem:
Train SVM for k (k-1) times; assign the class label that has the largest vote between all trained SVM.

3. Approach of Classifier Combiners

3.1 Majority Vote Combiner

Majority Vote is one simple classifier combination method based on the Sum Rule. A decision is made to classify a pattern to the class most often predicted by the classifiers.
The implement defect of this approach lies in tackling the even ties cases. When even ties happen the

classifier simply assigns a random label among the cases. However, a better idea is to assign the class based on their prior probability $P(C)$.

3.2 Borda Count Combiner

Borda Count is a quantity defined on the ranked outputs of each classifier. Assume $B_i(j)$ is the number of classes ranked below class w_j by classifier i, then Borda count for class w_j is B_j defined as:

$$B_j = \sum_{i=1}^{L} B_i(j)$$

A pattern is assigned to the class with the highest Borda count.

The tricky part of this method is the way you implement to get the rank of all three classifiers. We can use the posterior probability for Naïve Bayes Classifier and total vote number of each class, which is generated from One-Versus-One SVM mechanism. However, for decision tree classifier, there is no obvious way to get the rank information. The rank information for decision tree classifier used in the project is derived from confusion matrix.
Assume d_{lk} is an entry in confusion matrix, where l is the actual class label and k is the assigned class label. Then we assign the rank when we predict l based on the true classification information.

	Class A	Class B	Class C	Class D
Actual Class A	1000			
Actual Class B	100			
Actual Class C	10			
Actual Class D	50			

When we predict the test data as class A we will assign the rank as ABDC (1000>100>50>10).

3.3 Bayesian Combiner

Bayesian combiner simply uses the product rule with estimates of the posterior probabilities derived from the classifier predictions of each constituent classifier together with a summary of their performance on a labeled training set. Suppose $d_{jk}^{(i)}$, is the number of patterns with true class w_k assigned to w_j by classifier i. The conditional probability a sample x assigned to class w_l in classifier i actually belongs to w_k is estimated as following:

$$P(w_k \mid w_l) = \frac{d_{lk}^{(i)}}{\sum_{k=1}^{C} d_{lk}^{(i)}}$$

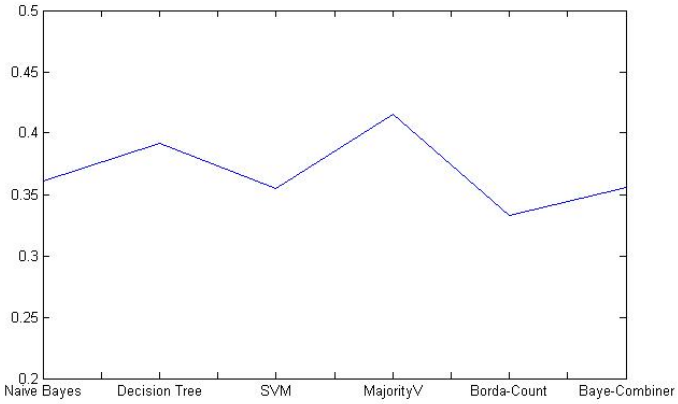
Thus by derivation from above formula and product rule, the decision rule for assigning the pattern to class w_j if

$$\prod_{i=1}^{L} \frac{d_{l,j}^{(i)}}{\sum_{k=1}^{C} d_{l,k}^{i}} > \prod_{i=1}^{L} \frac{d_{l,m}^{(i)}}{\sum_{k=1}^{C} d_{l,k}^{i}}, \text{ for m = 1,…,C; } m \neq j$$

This method needs to train all the classifiers and use their confusion matrix.

4. Result Analysis

Below is the error rate for all implemented algorithms and combiners. The error rate is acceptable and the combiners have smaller error rate generally.



Several things need to mention here:

1. The overall error is relatively high. The main reason lies in the structure of the dataset. The component ratio of classes diffs from each other heavily: The people who have no or light heart disease contributes to the most of the data, and the proportion of people of class 5 (means having the most severe heart problem) is only 5%.
2. The majority vote method has rather high error rate, though the true reason is unknown, it is highly probable that the design defect discussed above generates the problem.

5. Reference

[1] Data set: UCI machine learning repository.
[2] Andrew R. Webb, Statistical Pattern Recognition, Third Edition.
[3] MATLAB Built-in Function of SVM

6. Appendix A

Attributes:

1. #3 (age)
2. #4 (sex)
3. #9 (chest pain type)
4. #10 (resting blood pressure)
5. #12 (serum cholestoral in mg/dl)
6. #16 (fasting blood sugar > 120 mg/dl)
7. #19 (resting electrocardiographic results)
8. #32 (maximum heart rate achieved)
9. #38 (exercise induced angina)
10. #40 (ST depression induced by exercise relative to rest)
11. #41 (slope)
12. #44 (number of major vessels (0-3) colored by flourosopy)

13. #51 (thal : 3 = normal; 6 = fixed defect; 7 = reversable defect)
14. #58 (Disease Severity, prediction attribute)