Name: Fangqi Ma ID: a1756834 The University of Adelaide

Implementation of AdaBoost

(3920_COMP_SCI_X_0009 ISML Assignment 2)

Abstract

Boosting is one of the most necessary ensemble algorithms in machine learning. Adaboost, short for 'Adaptive Boosting', is a useful and basic boosting algorithm. In this report, the principle of implementing AdaBoost algorithm and the result comparison with some other binary classification algorithm is the main task to achieve. Having a deep understanding of boosting algorithm and then the ensemble learning can be another point as well.

1. Introduction

Boosting is one of the most necessary algorithms in machine learning, and Adaboost, short for 'Adaptive Boosting', is a useful and basic boosting algorithm, which usually used for classification and usually has an outstanding performance like random forest classifier. Having a deep understanding about Adaboost can be helpful for further study and research about boosting algorithms. Boosting algorithm uses some weak learner h in the features of each sample x_i in order to get the outputs. Under this scenario, choosing a learning algorithm to have a high accuracy of the outputs can be a hard question. There appears a point that in classification problems, we can use multi-classifier into one training sample as a combination of learning algorithms with different weights to improve the accuracy of final classifier.

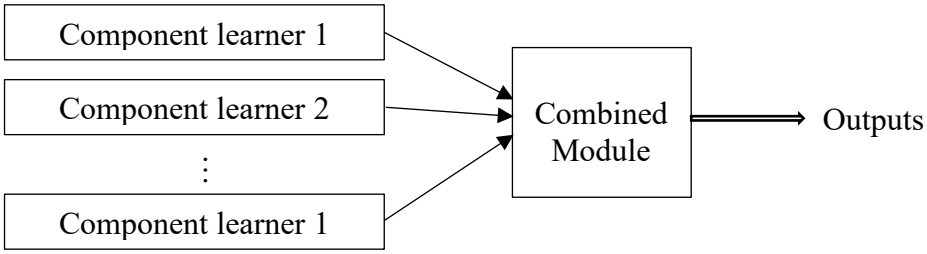
The process of completing the diagnosis prediction by using the Wisconsin Diagnostic Breast Cancer dataset firstly is to choose the decision stump algorithm from machine learning model, which is a one-level decision tree, as the weak learner to train sample data from the training dataset and then use this base algorithm to implement the Adaboost algorithm. The decision stump uses a single feature value as input to do prediction, which is always called 1-rules (Wikipedia). Based on the simple classification weak learner, the Adaboost can be attributed with different weight, such as 0.8: 0.2 that will be more precise than set the weight as 0.5: 0.5, into a classification strong learner. Reasonably, the prediction accuracy of using Adaboost algorithm is a kind of algorithm to unite a bunch of '1' into a complex system and it will be more precise than using single weak learner, and at the end of this assignment, the result of using Adaboost algorithm will be compared with 'build-in' package 'fitemsemble' in MATLAB and the SVM that built in the previous assignment through some dimensions like prediction running time or error rate to draw the conclusion.

2 Adaboost algorithm analysis

Adaboost algorithm is a powerful classification algorithm, the goal of this assignment is to classify the given Wisconsin Diagnostic Breast Cancer dataset, which is reasonably suitable for using Adaboost. One vital point that Zhou (2016) provided is using Adaboost algorithm to choose weak learner as a basic in order to design the Adaboost program, and here, using Decision Stumps is a convenient approach for designing Adaboost program.

2.1 Ensemble learning

Ensemble learning is to construct and combine a family of learners to finish the learning task, which sometimes can be called as multi-classifier system or committee-based learning. The main process of using ensemble learning firstly is to create an individual learner (or component learner) and then uses some approaches to make these as a unit.

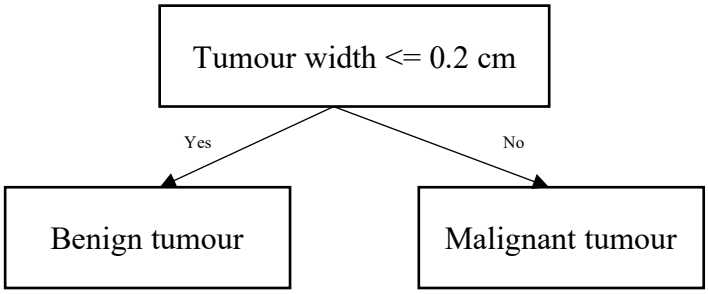


2.2 Weak Learner

Ensemble learning is combined with multiple learners in order to have a more outstanding performance of generalization, which is obvious for weak learner. The ‘weak learner’ is a concept comparing with the ‘strong learner’. Therefore, the weak learner is a learner that its accuracy is slightly better than 50%.

2.3 Decision Stumps

Decision Stumps is a simple weak learner algorithm induced from decision tree, which can be also called a one level decision tree. The meaning of ‘one level’ is using a single attribute to do the prediction instead of using the attribute space. For instance, when the model is to predict the label of tumour is benign or malignant only by using the width of tumour, such as the benign tumour is less and equals to 0.2 cm as well as the malignant tumour is larger than 0.2 cm, which is a fictitious example about one level decision, and actually there is only one attribute or feature contributes to the final result.



Obviously, Decision Stumps is better to be a weak base learner algorithm than being a main learner algorithm in supervised learning. The point of using Decision Stumps is finding the lowest error rate one level decision tree.

2.4 Boosting

Boosting is a machine learning algorithm that improves a family of weak learners into a strong learner in order to have a better performance of prediction. The usual process is to use initial training dataset to train a basic learner, then change the parameter of the weight distribution of each learner. After several times, the learning loops reach to a given value T, which can be finally summed together by different weight.

The most important and basic Boosting algorithm is Adaboost, which is based on additive model of base learner:

$$H(x) = \sum_{t=1}^{T} \alpha_t \, h_t(x)$$

to minimize the exponential loss function:

$$\ell_{exp}(H|D) = E_{x\sim D}[e^{-f(x)H(x)}]$$

The pseudocode for Adaboost (Schapire 2013) is shown in the figure 1 below. The first step of this process is to initialize the sample weight distribution and train the classifier h_t from the training dataset based on the weight distribution D_t with the error estimation of h_t. The further step is to calculate out and select the precise weight of h_t. The final step is to update the sample distribution and then update the $\sum_{t=1}^{T} \alpha_t \, h_t(x)$ to get the *H(x)*, which is also the output of Adaboost.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i\sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 1 Adaboost algorithm process

3 Model implementation

The Wisconsin Diagnostic Breast Cancer dataset in this assignment contains data and textdata, whose the previous one is a 569*30 double datasets with the latter is 569*2 cell that includes data and labels. The purpose here is to use the simply one-level decision trees, Decision Stumps, as the weak learner to implement the AdaBoost algorithm.

3.1 AdaBoost modelling

The first process is to import the data file and rebuild the dataframe. Obviously, the cancer description labels 'Benign' and 'Malignant' should be converted into 1 and -1, which can be convenient for the further implementation. Then, the first 300 samples should be used for training so that the training dataframe x and y will be built based on the 300 samples with another 269 samples will be built as testing sets.

The second process is to train the model. The Decision stumps should be built firstly and, in this part, this implementation uses model 'build_stump.m' (created by Lingqiao Liu) as reference. At the beginning of this process, the sample weight should be initialized to 1. Furthermore, the summation of $\alpha_t h_t(x)$ should be updated after each loop, and in another word, the first base learner h_1 is calculated by the original datasets. Then, the iteration is achieved by using loop in MATLAB in order to generate h_t and α_t, which is exactly the same as the algorithm explanation in the previous section that using model(t).alpha to receive the classifier weights update. Then it is necessary to find the distinguishable points and label the successive points in order to generate the weighted error, threshold and directions of the change. Finally, after finding the expected base learner, the weak learner, based on the minimum of the weighted error and rearrange label into 1 and -1, the best stump will be returned into the AdaBoost trainer.

The second process is to do prediction using the model trained, which is also the test model process. This process is actually an application of the model that we explained above, which means applying the data x_test and y_test into the weak learner Decision Stumps to generate the strong learner. One important point here should be mentioned that the classification error of training and testing model will be recorded in order to analyse the accuracy of algorithm. Implementation time will also be recorded for analysing the performance of algorithm. Another character of this weak learner should be mentioned is the depth of this weak learner is 1 because the Decision Stumps used in this assignment is a one-level decision tree. The number of weak learners used for the AdaBoost algorithm here will be discussed in the latter section.

The result of AdaBoost model will be plotted as the error curve so that the performance of model can be easily analysed and compared with the result of using fitensemble and SVM. The explanation and comparison will be shown as followed.

3.2 Fitensemble implementation and comparison

In order to verify the performance of the AdaBoost algorithm we built in the previous section, referring a 'built-in' package will be powerful and convincing. In this section, the fitensemble in MATLAB will be referred to do the comparison. Fitensemble in MATLAB is a powerful learner of classification and regression, which can boost decision tree learners 'Tree' to analyse the classification problems (MathWorks). For binary classification, fitensemble has methods like 'AdaBoostM1', 'LogitBoost', 'GentleBoost' and so on (MathWorks). The usual process of using fitensemble package is firstly, train classification ensemble, and then estimate the generalization error of the boosting classification ensemble of the decision trees trained. Finally controlling the depth of the decision trees and find the optimal number of weak learners for this ensemble but in this assignment, it is enough for the comparison that implement fitensemble training, testing and error calculation. The fitensemble plot will be compared with the training and testing error curve of AdaBoost implementation.

3.3 SVM implementation

Support vector machine is usually noted as SVM. When solving binary class linear problems, SVM is very useful because it is convenient to separate the dataset by a hyperplane created as "+" samples and "-" samples with features. In this assignment, the main purpose of using SVM can be regarded as a reference object and the MATLAB cvx package also referred in this section to solve soft margin dual problems. The outstanding function of cvx has already discussed in the previous assignment report, such as applying to solve optimal problems. The usual process of implementing SVM soft margin dual problem is shown as the figure below:

cvx;
variables $w\ b\ \xi\ \alpha\ \gamma$*;*
maximize $L(\alpha) = -\frac{1}{2}\sum\sum \alpha_i\,\alpha_j\,x_i x_j y_i y_j + \sum \alpha_i$*;*
subject to $\alpha_i \geq 0\ \ \sum \alpha_i\, y_i = 0$*;*

Figure 2 SVM soft margin dual problem

In this place, the performance of the SVM implementation, such as implementation time, error rate or accuracy and the value of w, b will also be recorded as the element in order to compare with the performance of AdaBoost.

4 Experimental results

In this section, the result of using strong learner AdaBoost based on the weak learner Decision Stumps will be shown and analysed and the comparison with MATLAB 'built-in' package fitensemble and SVM algorithm.

4.1 AdaBoost result and comparison

In order to have further observation about the learning curve of AdaBoost, the iteration times have already been set as 50, 100 and 200. There arises an important phenomenon that when training model iteration times reach at 20, the model error is 0, which means the number of weak learners is 20. It is reasonable that the testing model can reach at the lowest error rate but in this place, after 20 times iteration, the testing error rate still decreases to a limited extends and finally stabilize at rounded 0.0297 (recorded as the variable testError in MATLAB coding), which is equally with the accuracy 97.03%.

Similarly, the fitensemble training accuracy (recorded as the variable accuracy_fitTrain in MATLAB coding) reach to 100% when the iteration time is 21. Comparing with AdaBoost based on Decision Stumps, the performance of fitensemble prediction is a bit worse but is pretty the same. The fitensemble testing accuracy (recorded as the variable accuracy_fitTest in MATLAB coding) reach at a stable rate after 21 times iteration but still have a fluctuation and finally reach at the accuracy between 97.18% and 97.54%. Obviously for accuracy, the performance of fitensemble testing is more accurate than AdaBoost based on Decision Stumps.

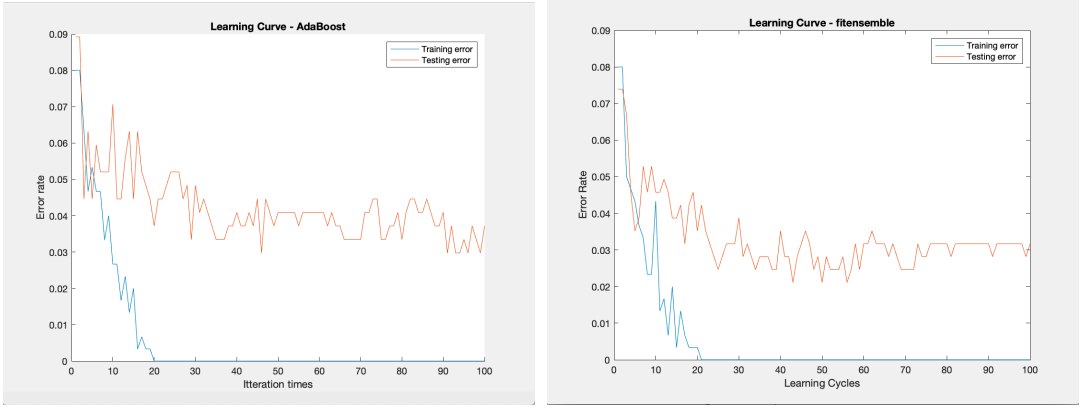


Figure 3 AdaBoost vs fitensemble comparison

4.2 AdaBoost and SVM comparison

From the previous section, the AdaBoost based on Decision Stumps has a good performance for binary classification, the testing accuracy can reach at 97.03%. After implementing SVM algorithm by referring cvx function in MATLAB (mainly because the libsvm package in my MAC has variable name clash and not easy to compile), the accuracy has already recorded in

MATLAB coding variable accuracy_svmTrain and accuracy_svmTest (CVX Research). The accuracy_svmTrain is 96.00% and accuracy_svmTest is 92.19%, which is reasonable that training accuracy is higher than testing because the model is trained by the original data. The SVM algorithm is not an ensemble algorithm, so it does not make sense to plot the learning process.

accuracy_svmTest	0.9219
accuracy_svmTrain	0.9600

As is shown above, the AdaBoost based on Decision Stumps accuracy is higher than using SVM designed by cvx and the difference is 4.86%.

4.3 Implementation times

The implementation time is another important element to measure the performance of algorithm. In this assignment, the implementation time is recorded in a structure and has shown below.

t_train_ada	0.4942
t_test_ada	0.0031
t_train_fit	1.4397
t_test_fit	0.8978
t_train_svm	1.4062
t_test_svm	1.5951e-04

Different development environment can affect the implementation time results. In this assignment, the time recorded with MATLAB R2018b Update 5 (9.5.0.1178774) and macOS 10.14.6.

The result can be obviously seen that the AdaBoost based on Decision Stumps model training and testing times in total is less than using fitensemble and SVM.

5 Discussion and conclusion

The conclusion can be drawn that using AdaBoost algorithm based on Decision Stumps has a higher accuracy and less implementation time in solving binary classification problems, but the ensemble process is more complicated and more importantly, the understanding of this algorithm can be an obstacle for beginners.

In total, there is one vital hypothesis that when using boosting algorithm like AdaBoost, the error of base learners should be i.i.d (independent and identically distributed) in order to have a higher accuracy after many times iterations. In this assignment the only weak learner is a one-level decision tree algorithm Decision Stumps, in the further research it is possible to use different base learner in order to have 'diversity'. Another point is the method to achieve re-weighting and then convert weak learner into strong learner in order to implement boosting algorithm.

Reference

MathWorks, *fitensemble Documentation*, viewed on 28 September, <https://au.mathworks.com/help/stats/fitensemble.html>.

Patel, S 2017, Chapter 6: Adaboost Classifier, Machine Learning 101, Medium, <https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06>.

CVX Research, *CVX User's Guide*, viewed on 1 October, < http://cvxr.com/cvx/doc/index.html>.

Wikipedia, *AdaBoost*, viewed on 22 September, < https://en.wikipedia.org/wiki/AdaBoost>.

Wikipedia, *Decision Stumps*, viewed on 22 September, <https://en.wikipedia.org/wiki/Decision_stump>.

Schapire, R.E. 2013, Explaining AdaBoost, *Empirical Inference*, pp. 37-52.

Zhou, Z 2016, *Machine Learning*, Tsinghua University Press, Beijing, pp. 171-177.