Course Project: Ensemble combination of diverse Classifiers based on AdaBoost Using Genetic Algorithm

Tianyi Fang
PUID:00290865811
Course Name: Nature Inspired Algorithm

Abstract— **AdaBoost, also known as adaptive boosting, as a machine learning ensemble meta-algorithm, convert weak learners to stronger one by learning from former classifiers and give predictions by weighted averaging or voting, in order to improve the prediction accuracy of classification. However, the performance of an ensemble is dependent on the choice of constituent base classifiers. The most widely used weak learner is decision stump. In this article, mainly focused on binary classification problem, I tried to extend the original AdaBoost by combining different weak classifiers, whose mechanisms and performances are different from each other towards different type of data. Then, since there are thousands of weak learners in the final model, I applied GA search algorithm to find the best combination of base weak learners to reduce complexity and increasing interpretability of the final model. My proposed model is named as GAMixBoost Performance are evaluated and compared with single basic classification models such as SVM and K-Nearest Neighbor Algorithm and widely used ensemble models such as Random Forest in real world data sets from UCI- Machine Learning Repository. In some dataset, GAMixBoost performs better than single classifiers and results in a simpler predictive model with similar performance as other ensemble classifiers.**

I. INTRODUCTION

Binary classification problems are the most popular and common issues in nearly every fields. Distinguishing junk email from good one, detecting potential fraud clients, forecasting win or lose of one football team are where binary classification could be used. Since there are hundreds of types of classification algorithms, with their own specific strengths and weaknesses suited for different types of input data and practical objectives, which one we should use in general binary classification case, or how to decide the right learner which matches learning problem appropriately? Vary learners are masters at their own fields but also have their own weaknesses. For example, one of the most famous and widely used classification: decision tree, is an all-purpose classifier that performs well on most problems, can handle different kinds of input data as well as missing data, but is easy to overfitting or underfitting and is greatly sensitive to small changes, which influences its predictability. Instead of improving the performance of a single learner, combining individual models into a group such as bagging and boosting, give us better generalizability to future problems, improvements of performance on massive or miniscule datasets. One of the most frequently used boosting algorithm, AdaBoost, generates weak learners that iteratively learn a larger portion of the difficult-to-classify examples by paying more attention (give more weight) to frequently misclassified examples, then the final decision are based on results of all weak learners and their associated weights. The most common weak learner used with AdaBoost are decision trees(one split with two terminal nodes), which is also called decision stumps.

In my project, I tried different types of classifier as weak learners of AdaBoost, then using genetic algorithm(GA) to select which weak learners should be kept and which should be dropped in order to reduce the complexity of final model, that is, the best subset of weak learners. Since AdaBoost cycles through in a highly repetitive way among a small set of weak classifiers, there are many redundant weak learners in the final AdaBoost model, as mentioned in articles[3]. Thus, only remaining partial percentage of final model could increasing model's interpretability without hurting the model's accuracy.

In the next section, I will briefly introduce how I implement AdaBoost with diverse weak learners. In Section III, I will describe how to reduce the complexity of final model using Genetic Algorithm(GA) as weak learner selection. Algorithm and implementation

*This work was not supported by any organization

[1]Tianyi Fang is master student in Industrial Engineering in Purdue University

[2]This report is done for IE 590 Nature Inspired Algorithm project

steps are shown in Section IV. Experiments results and comparison, based on real world data sets are shown in Section V.

II. ADJUSTED ADAPTIVE BOOSTING

Adaptive Boosting classifier combines weak learner algorithm to form strong classifier. Its algorithm is: taking initially unweighted n examples as input, at each iteration, generates a new weak learner according to examples weights and the result of prediction, then update the weight of examples(increase weight for misclassified examples, remain same weight for correctly classified examples, then implementing normalization), also model's weight is given to the current learner, according to its error rate. After defined iteration T times, AdaBoost generated a set of weak learners and their own weights (as related to how accurate this weak learner is), the final prediction will be done weighted vote.
For the selection of weak learner, the most frequently used weak learner is decision stump, which perform a single test on a single attribute with threshold θ. There are papers using other relatively strong learners, such as SVM or KNN as base learner for AdaBoost[3], [4], which largely improve the model's performance. But still, based on only a single type of learner, it is not that suitable for general cases.
Here in my project, I adjusted the original AdaBoost by using several different relatively weak classifier classes with different parameters, such as decision tree(dt), support vector machine(svm) and K- Nearest Neighbor(knn), etc, as ensemble, but at the same time, do parallel boosting for each classification class. In that way, I hope to generalize the final model, increase both its prediction accuracy and generality.

Initialization Different weak classification types were selected. Then, setting different parameters of each classifier class, each single classifier with either its unique classification type or tuning parameter is treated as one weak learner, named p.learner (parameter learner).
In each iteration Train each p.learner in its own learner class separately, with weighted dataset, computed its weighted error and the weight of each p.learner. Finally, updated weights of each examples for the next iteration learning step, just as what AdaBoost do. The error rate of iteration t for learner G_t for x_i example is

$$\epsilon_t = \frac{\sum_{i=1}^{T} w_t Pr(G_t(x_i) \neq y_i)}{\sum_{i=1}^{T} w_t}$$

The vote $\alpha_t = log(\frac{1-\epsilon_t}{\epsilon_t})$
Updated weight:

$$w_{t+1}(i) = \begin{cases} \frac{w_t(i)}{z_t} e^{-\alpha_t} & if G_t(x_i) = y_i \\ \frac{w_t(i)}{z_t} e^{\alpha_t} & if G_t(x_i) \neq y_i \end{cases}$$

where $z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ is the normalization factor
Model Generation. After learning from weighted examples and gain their own weights by T iteration, the GAMixBoost has generated a pool of all p.learners(with N = $u \times T$ different classification and different parameters, where u = # of classifier class$\times$ # of parameters of each class). In this phase, only train data from the train database are explored, then performance of p.learners are evaluated by cross validation in the validation sets. Additional to the weighted error rate, ROC value of each p.learner is recorded, together with its weight, formed a basic component for GA search for the best combination of p.learner in the next section.

III. GENETIC ALGORITHM FOR BASE WEAK LEARNER SELECTION

Inspired by Knapsack Problem, I treated each p.learner as one item, with its value(ROC value) and weights(model weight), and the bag is like our final model. We always prefer higher bag value but lower weight. So the problem becomes whether to choose this p.learner or not while maximizing the average value of the bag. In order to evaluate the final model more accurately after learner selection, I choose the average value of ROC of selected p.learner as the fitness value of each component, instead of total value in Knapsack Problem. In that case, the measurement of combination of p.learner is still between (0,1), with value closed to 1 indicating better performance.
To limit the number of p.learners, a penalized fitness function is utilized, results in an interpretability advantage over other boosting algorithms due to reduced model complexity. The solution in the final generation with the largest fitness value is chosen as the final solution.

Individual Representation In this project, binary encoding is applied for representing the combination of p.learner by an individual T_s^i as chromosome, with illustrated in Fig.1. The mapping of individuals in GA algorithm and my p.learner pool is shown in Figure.2. Since p.learners of each learner class are generated separately within their own class type, their weights are normalized within their class. For example, if the first 10 p.learners belong SVM learner, so the total weights of these 10 p.learners are 1. After being selected, the total weight will be normalized again according to the requirement of model complexity.

Id	1	2	3	4	5	6	7	8	...
AUC	0.88	0.7	0.54	0.9	0.67	0.92	0.78	0.69	...
weight	0.1	0.3	0.2	0.4	0.02	0.15	0.33	0.15	...
Select	1	0	0	1	0	1	0	0	...

Fig. 1. Binary Encoding and selection of p.learner in GA

Population The GA begins with a population containing a set of random individuals. The initial population size can be set depends on the problem, while here I used a good rule to determine the size of initial population $\|P\|$, given by [1] as

$$\|P\| = min(5 \times k, (\frac{1}{2} \times e))$$

where k is the size of an individual and e is the maximum number of possible ensemble combinations($e = 2^k$). Thus, a population with $5 \times N$ is created in my project.

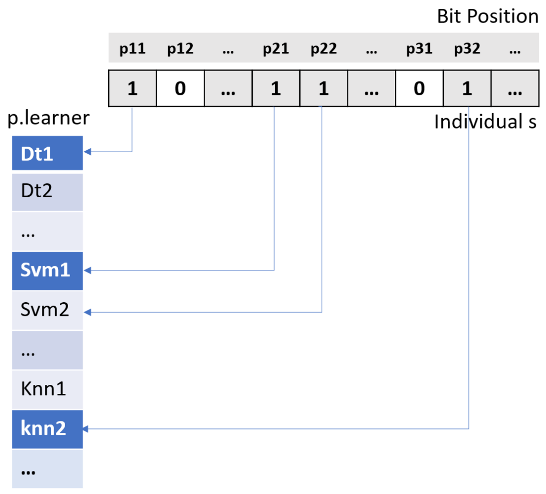


Fig. 2. Representation of an individual in Proposed model and its mapping into the corresponding p.learners for combination

Fitness Evaluation The fitness function of GA measures both the goodness of a solution(average ROC value) with constrain of its complexity (how much weight one individual contains). The following formula:

$$Maximum \quad f(s) = \frac{\sum_{i=1}^{S} AUC_i}{S}$$

$$s.t. \quad \sum_{i=1}^{S} w_i \leq \frac{1}{b} \sum_{i=1}^{N} w_i$$

where S is the number of p.learners in each individual, b is the penalty parameter, defined by user or specific problem.

Penalty for Solution Complexity AdaBoost usually have hundreds or thousands of weak learners, which are found duplicated(repeated and cycled pattern find by[2]). My idea is constructing a simpler model with constrain the maximum number of p.learners, which is measured by their weight. I want the simplified model is $\frac{1}{b}$ of original one. b can be tuned based on specific model complexity requirements.

GA Operation

Operations like elitist selection, probability of mutation, crossover and type of crossover are used. I fixed my elitist rate as 10%.

Elitist selection keep some of the best solutions in this generation from the previous generation, according to their fitness values. In this way, optimal solutions are unlikely to be lost.

Crossover operation determine how the offsprings is generated from the selected parents. I tried two types: Single point crossover, which randomly select a bit position in selected parents, then one child gets half of each parent. Uniform crossover uses a fixed mixing ration between two parents, enables the parent chromosomes to contribute the gene level rather than the segment level. If the mixing ratio is 0.5, the children have approximately half of the genes from the first parent and the other half from the second parent.

Within each type of crossover, I compare different settings of probability of crossover and mutation. Parameter settings are shown in Figure 4.

IV. Algorithm and Implementation Steps

The prototype of entire proposed model is shown in Fig.3.

Input m examples of training set, which are pairs: $(\underline{x_1}, y_1), ..., (\underline{x_m}, y_m)$ are put into each p.learner, where $\underline{x_1}$ belongs to the feature space $\mathbf{R^D}$, y_i is converted into $\{p, n\}$ for every dataset. Initialized the weights of each example as $\frac{1}{m}$.

Mixed Adaptive Boosting For each learner class, do AdaBoost for T iterations. Save each p.learner of each iteration and each learner class, along with its own ROC and weight, as one component of individual in the next GA search phase.

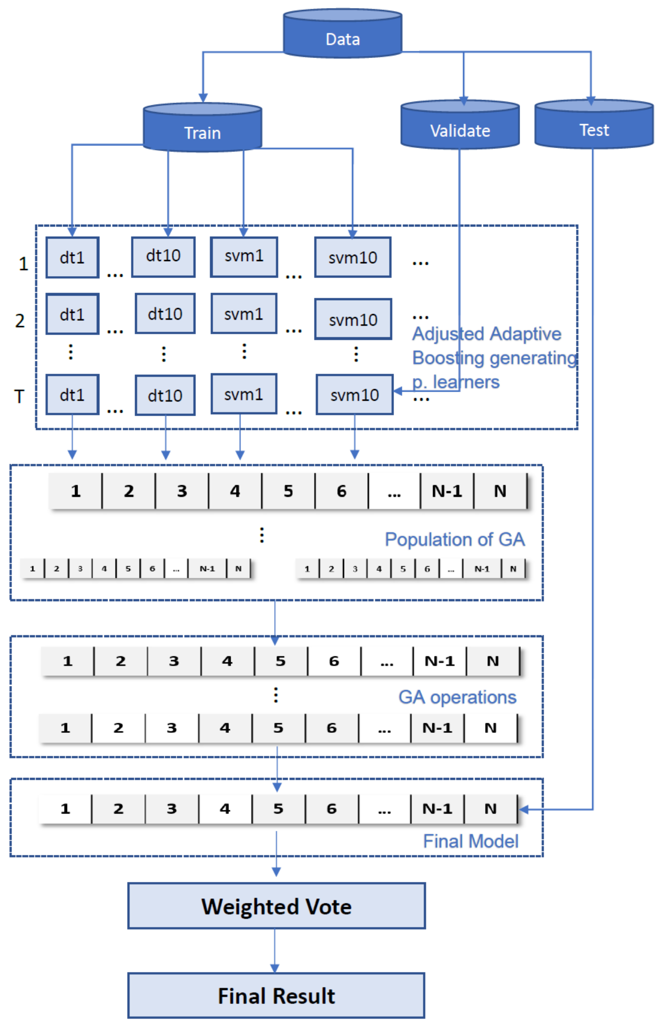


Fig. 3. Algorithm of GAMixBoost

GA selection Random generated a population of individuals, which are the combination strategy of p.learner. Then applied mutation, crossover and elitist to find the best individual with highest fitness value, that is our simplified strong learner.
Prediction Put test data into the simplified strong learner, evaluate the model's performance.
Termination The GA search algorithm will finish searching if any of the following conditions is satisfied:

- The total number of generation has reached 1000
- The fitness of the best individual of the population has remained stationary for 400 consecutive generations.
- The fitness value has reached the global optimal value.

Comparison Since the GAMixBoost model are based on AdaBoost and ensemble method, I compared the performance of learners of similar type, such as AdaBoost and Random Forest and the performance of its base weak learner classes.

V. Experiments and Results

A. Datasets

In this section, I compared the performance of my GAMixBoost with Random Forest and AdaBoost on three real world data sets from the UC-Irvine Machine Learning Repository. Basic information of the datasets are shown in Table.1

TABLE I
CHARACTERISTICS OF DATASETS USED FOR EXPERIMENTS

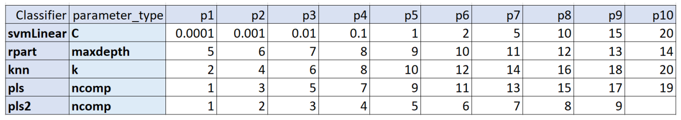
Dataset	Num.of feature	Num. of samples
Wisconsin Breast Cancer	31	699
Pima Indians Diabetes	9	768
Sonar	60	208

B. Parameter Learner Class Selection

The package I use to generate base weak classifiers is Caret in R, which contains 281 types of both single and ensemble learner for machine learning algorithm. How to decide the base weak learner is my first problem. Known decision stump is the most frequently used weak learner for AdaBoost is due to its weak classification ability, which is only a little bit more than 50% accuracy. Since complex model with high accuracy will results in very low error rate, even sometimes really close to 0, which will cause problems when calculating error rate and model weight(as denominator equals to 0). Thus, I focus on finding learners with relatively low accuracy first. Then among all the selected weak learner class, I chose learner classes with only one tuning parameter, which enables my following implementation become easier. Based on these two criteria, I finally use decision tree(rpart2), svm(svmLinear), pls and knn as my weak learner class. Their prediction performance are tested on one real world dataset Sonar and one simulated data, as shown in Figure 4.

C. Base Learner

For the base learner, I used four base learner types: Support Vector Machines with Linear Kernel (svmLinear), decision tree(rpart), K-Nearest Neighbors (knn) and Partial Least Squares Discriminant Analysis(pls). Each learner class contains 10 different parameters. The overall information about base learner are shown in Figure 4.



Classifier	parameter_type	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
svmLinear	**C**	0.0001	0.001	0.01	0.1	1	2	5	10	15	20
rpart	**maxdepth**	5	6	7	8	9	10	11	12	13	14
knn	**k**	2	4	6	8	10	12	14	16	18	20
pls	**ncomp**	1	3	5	7	9	11	13	15	17	19
pls2	**ncomp**	1	2	3	4	5	6	7	8	9	

Fig. 4. Parameters of base classifiers. (pls2 is for Pima dataset)

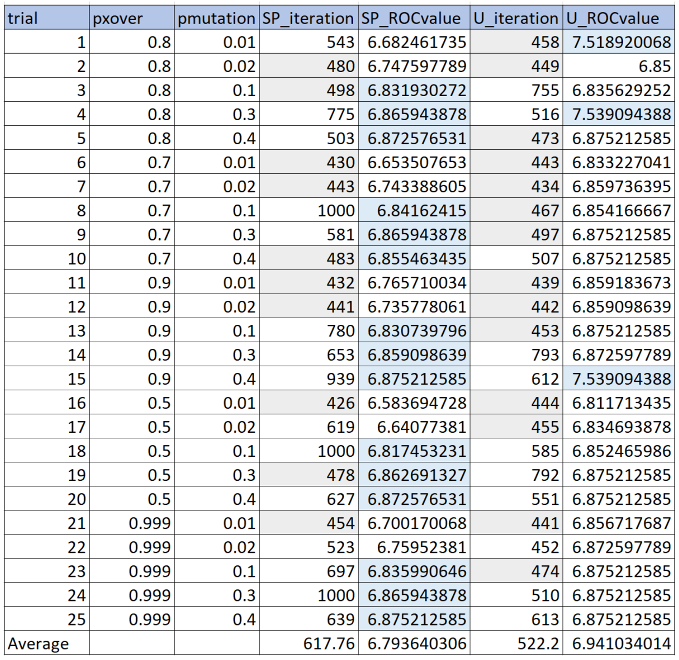
D. *Evaluation Selection*

To evaluate the performance ability of binary classification, there are various of criteria. For example, error rate, Accuracy, Precision , Recall, F-1 Score, etc. The one I used to roughly select the base weak learner is their **Accuracy**; The one used to update weights of examples and calculate the weights for each p.learner is the **error rate** of the validation set (both are shown in Section II); The one I used to set as part of the fitness value is **ROC value**; For the final model comparison, I used many measurements to evaluate the average classification performance of my GAMixBoost for all experiments dataset, which is shown in the result figures and plots.

E. *GA Parameter Tuning*

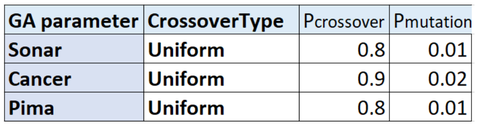
In GA selection, I focused on making balance of the probability of mutation and crossover in different type of crossover, to find the best GA parameters. The Experimental Design and results of tuning parameters for Sonar dataset are shown in Fig.5 as an illustration.

We want a set of parameters converge to the optimal solution fast, that is with smaller generation, but higher fitness value. From the Fig.5 we can find for the type of crossover, Single-point crossover are more fluctuate than uniform crossover, with extremely large/small iteration values. On the other hand, the iteration value of uniform crossover are more flat and centered around 450 500. For the ROC value, Uniform crossover find solutions with higher fitness value. Choose which set of parameters depends on the problem itself and what we care more. For this Sonar dataset, I chose **Uniform crossover type** since it is more stable with higher converge rate.
For probability of crossover and mutation, there exists trade-offs. Larger probability of mutation results in larger ROC value, but the higher rate of mutation probability is, the larger generations is, which means slow converge. Since uniform crossover has relatively stable ROC value, I choose $P_{crossover}$=0.08, and $P_{mutation}$ = 0.01(the first trial). Others are shown in



trial	pxover	pmutation	SP_iteration	SP_ROCvalue	U_iteration	U_ROCvalue
1	0.8	0.01	543	6.682461735	458	7.518920068
2	0.8	0.02	480	6.747597789	449	6.85
3	0.8	0.1	498	6.831930272	755	6.835629252
4	0.8	0.3	775	6.865943878	516	7.539094388
5	0.8	0.4	503	6.872576531	473	6.875212585
6	0.7	0.01	430	6.653507653	443	6.833227041
7	0.7	0.02	443	6.743388605	434	6.859736395
8	0.7	0.1	1000	6.84162415	467	6.854166667
9	0.7	0.3	581	6.865943878	497	6.875212585
10	0.7	0.4	483	6.855463435	507	6.875212585
11	0.9	0.01	432	6.765710034	439	6.859183673
12	0.9	0.02	441	6.735778061	442	6.859098639
13	0.9	0.1	780	6.830739796	453	6.875212585
14	0.9	0.3	653	6.859098639	793	6.872597789
15	0.9	0.4	939	6.875212585	612	7.539094388
16	0.5	0.01	426	6.583694728	444	6.811713435
17	0.5	0.02	619	6.64077381	455	6.834693878
18	0.5	0.1	1000	6.817453231	585	6.852465986
19	0.5	0.3	478	6.862691327	792	6.875212585
20	0.5	0.4	627	6.872576531	551	6.875212585
21	0.999	0.01	454	6.700170068	441	6.856717687
22	0.999	0.02	523	6.75952381	452	6.872597789
23	0.999	0.1	697	6.835990646	474	6.875212585
24	0.999	0.3	1000	6.865943878	510	6.875212585
25	0.999	0.4	639	6.875212585	613	6.875212585
Average			617.76	6.793640306	522.2	6.941034014

Fig. 5. Results of Tuning parameters for GA selection(Sonar dataset)



GA parameter	CrossoverType	Pcrossover	Pmutation
Sonar	**Uniform**	0.8	0.01
Cancer	**Uniform**	0.9	0.02
Pima	**Uniform**	0.8	0.01

Fig. 6. GA parameters for Sonar, Cancer, Pima Datasets

Fig.6. The Converge results of GA for Sonar dataset are shown in Fig.7 and Fig.8 as an illustration.

F. *Evaluation*

The model performance, based on different evaluation methods are shown in Fig.10 to Fig.21. Taken Sonar as illustration, we can find that GAMixBoost's performance are closed to the performance of AdaBoost, even the best in Accuracy, Specificity and Precision. GAMixBoost has better performance than either of its own base learners, which indicates improvement of model prediction ability by GAMixBoost. However, for the Breast Cancer dataset and Pima dataset, the performance of GAMixBoost is not that good. Possible reasons could be the imbalanced class and the outliers. Sometimes, the knn and SVM preform better even than random forest and adaboost classifier. But in general, the performance of my GAMixBoost are close to other

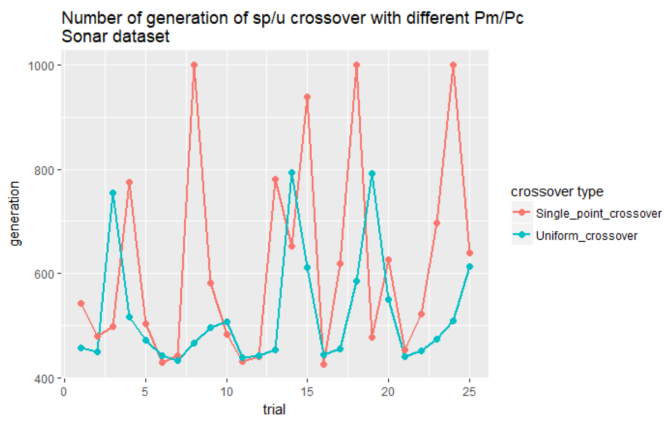


Fig. 7. Converge Results of Tuning parameters for GA selection

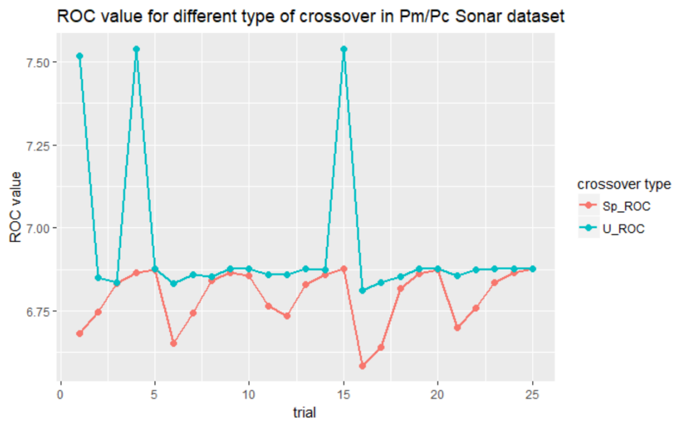


Fig. 8. Fitness value results of Tuning parameters for GA selection

ensemble models, which indicates my model could also be applied in general binary classification cases.

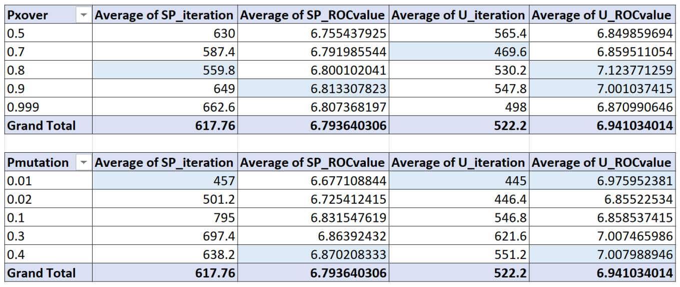
G. Final Result

For each dataset, the final result is generated by GAMixBoost. Final model and information are shown in Figure 18.

VI. Conclusions and Improvements

A. Conclusions

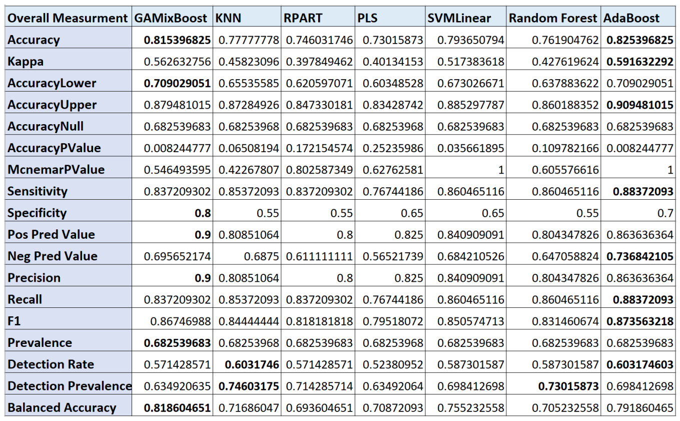
This project presents a genetic algorithm based search method, named GAMixBoost, for constructing heterogeneous ensembles of classifiers, which are generated by adaptive boosting method. As we know, adaptive boost combines a group of weak learners into a strong learner, by double weighted learning: each iteration, new learner in AdaBoost dealing with the previous updated weighted misclassified examples and gain its own weight based on its performance. While as



Pxover	Average of SP_iteration	Average of SP_ROCvalue	Average of U_iteration	Average of U_ROCvalue
0.5	630	6.755437925	565.4	6.849859694
0.7	587.4	6.791985544	469.6	6.859511054
0.8	559.8	6.800102041	530.2	7.123771259
0.9	649	6.813307823	547.8	7.001037415
0.999	662.6	6.807368197	498	6.870990646
Grand Total	**617.76**	**6.793640306**	**522.2**	**6.941034014**

Pmutation	Average of SP_iteration	Average of SP_ROCvalue	Average of U_iteration	Average of U_ROCvalue
0.01	457	6.677108844	445	6.975952381
0.02	501.2	6.725412415	446.4	6.85522534
0.1	795	6.831547619	546.8	6.858537415
0.3	697.4	6.86392432	621.6	7.007465986
0.4	638.2	6.870208333	551.2	7.007988946
Grand Total	**617.76**	**6.793640306**	**522.2**	**6.941034014**

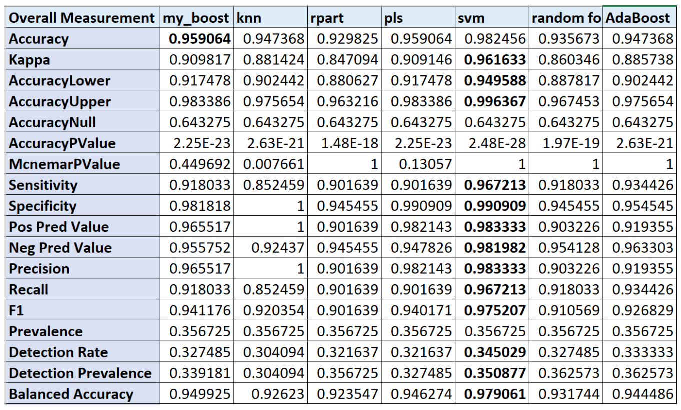
Fig. 9. Average results of different Pm and Pxover parameters for GA selection for Sonar Dataset



Overall Measurment	GAMixBoost	KNN	RPART	PLS	SVMLinear	Random Forest	AdaBoost
Accuracy	**0.815396825**	0.77777778	0.746031746	0.73015873	0.793650794	0.761904762	**0.825396825**
Kappa	0.562632756	0.45823096	0.397849462	0.40134153	0.517383618	0.427619624	**0.591632292**
AccuracyLower	**0.709029051**	0.65535585	0.620597071	0.60348528	0.673026671	0.637883622	0.709029051
AccuracyUpper	0.879481015	0.87284926	0.847330181	0.83428742	0.885297787	0.860188352	**0.909481015**
AccuracyNull	0.682539683	0.68253968	0.682539683	0.68253968	0.682539683	0.682539683	0.682539683
AccuracyPValue	0.008244777	0.06508194	0.172154574	0.25235986	0.035661895	0.109782166	0.008244777
McnemarPValue	0.546493595	0.42267807	0.802587349	0.62762581	1	0.605576616	1
Sensitivity	0.837209302	0.85372093	0.837209302	0.76744186	0.860465116	0.860465116	**0.88372093**
Specificity	**0.8**	0.55	0.55	0.65	0.65	0.55	0.7
Pos Pred Value	**0.9**	0.80851064	0.8	0.825	0.840909091	0.804347826	0.863636364
Neg Pred Value	0.695652174	0.6875	0.611111111	0.56521739	0.684210526	0.647058824	**0.736842105**
Precision	**0.9**	0.80851064	0.8	0.825	0.840909091	0.804347826	0.863636364
Recall	0.837209302	0.85372093	0.837209302	0.76744186	0.860465116	0.860465116	**0.88372093**
F1	0.86746988	0.84444444	0.818181818	0.79518072	0.850574713	0.831460674	**0.873563218**
Prevalence	**0.682539683**	0.68253968	0.682539683	0.68253968	0.682539683	0.682539683	0.682539683
Detection Rate	0.571428571	**0.6031746**	0.571428571	0.52380952	0.587301587	0.587301587	**0.603174603**
Detection Prevalence	0.634920635	**0.74603175**	0.714285714	0.63492064	0.698412698	**0.73015873**	0.698412698
Balanced Accuracy	**0.818604651**	0.71686047	0.693604651	0.70872093	0.755232558	0.705232558	0.791860465

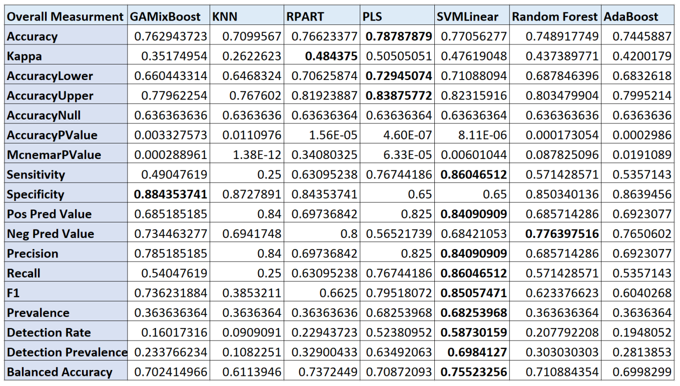
Fig. 10. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Sonar data(a)

the number of base classifier increases, the number of possible ensembles that can be created rises exponentially, and the final model are difficult to interpret. Here GA search helps in exhaustive search for constructing the best combination of base classifiers, then based on the limited numbers of base learner, weighted voting technique are used for combining the base classifier's predictions in a single final decision. In that case, the final model contains only a subset of weak learners in the boosted model, which keep the original model's prediction ability, but in a simpler version.
My GAMixBoost algorithm are separated into three main parts. First, generates a pool of base learners by alternating the parameters of four classifier types. That is the initial base learners, named p.learner. Then, going through adaptive boosting on train set within their own class type. At each iteration, being trained in weight-updated dataset and being evaluated by its error rate, which will give the p.learner a model weight. After T iteration, a larger pool of solution of p.learner are generated by binary encoding, for GA algorithm to search for the best combination of p.learners. The fitness value of each learner are measured by ROC and



Overall Measurement	my_boost	knn	rpart	pls	svm	random fo	AdaBoost
Accuracy	**0.959064**	0.947368	0.929825	0.959064	0.982456	0.935673	0.947368
Kappa	0.909817	0.881424	0.847094	0.909146	**0.961633**	0.860346	0.885738
AccuracyLower	0.917478	0.902442	0.880627	0.917478	**0.949588**	0.887817	0.902442
AccuracyUpper	0.983386	0.975654	0.963216	0.983386	**0.996367**	0.967453	0.975654
AccuracyNull	0.643275	0.643275	0.643275	0.643275	0.643275	0.643275	0.643275
AccuracyPValue	2.25E-23	2.63E-21	1.48E-18	2.25E-23	2.48E-28	1.97E-19	2.63E-21
McnemarPValue	0.449692	0.007661	1	0.13057	1	1	1
Sensitivity	0.918033	0.852459	0.901639	0.901639	**0.967213**	0.918033	0.934426
Specificity	0.981818	1	0.945455	0.990909	**0.990909**	0.945455	0.954545
Pos Pred Value	0.965517	1	0.901639	0.982143	**0.983333**	0.903226	0.919355
Neg Pred Value	0.955752	0.92437	0.945455	0.947826	**0.981982**	0.954128	0.963303
Precision	0.965517	1	0.901639	0.982143	**0.983333**	0.903226	0.919355
Recall	0.918033	0.852459	0.901639	0.901639	**0.967213**	0.918033	0.934426
F1	0.941176	0.920354	0.901639	0.940171	**0.975207**	0.910569	0.926829
Prevalence	0.356725	0.356725	0.356725	0.356725	0.356725	0.356725	0.356725
Detection Rate	0.327485	0.304094	0.321637	0.321637	**0.345029**	0.327485	0.333333
Detection Prevalence	0.339181	0.304094	0.356725	0.327485	**0.350877**	0.362573	0.362573
Balanced Accuracy	0.949925	0.92623	0.923547	0.946274	**0.979061**	0.931744	0.944486

Fig. 11. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Wisconsin Breast Cancer data(a)



Overall Measurment	GAMixBoost	KNN	RPART	PLS	SVMLinear	Random Forest	AdaBoost
Accuracy	0.762943723	0.7099567	0.76623377	**0.78787879**	0.77056277	0.748917749	0.7445887
Kappa	0.35174954	0.2622623	**0.484375**	0.50505051	0.47619048	0.437389771	0.4200179
AccuracyLower	0.660443314	0.6468324	0.70625874	**0.72945074**	0.71088094	0.687846396	0.6832618
AccuracyUpper	0.77962254	0.767602	0.81923887	**0.83875772**	0.82315916	0.803479904	0.7995214
AccuracyNull	0.636363636	0.6363636	0.63636364	0.63636364	0.63636364	0.636363636	0.6363636
AccuracyPValue	0.003327573	0.0110976	1.56E-05	4.60E-07	8.11E-06	0.000173054	0.0002986
McnemarPValue	0.000288961	1.38E-12	0.34080325	6.33E-05	0.00601044	0.087825096	0.0191089
Sensitivity	0.49047619	0.25	0.63095238	0.76744186	**0.86046512**	0.571428571	0.5357143
Specificity	**0.884353741**	0.8727891	0.84353741	0.65	0.65	0.850340136	0.8639456
Pos Pred Value	0.685185185	0.84	0.69736842	0.825	**0.84090909**	0.685714286	0.6923077
Neg Pred Value	0.734463277	0.6941748	0.8	0.56521739	0.68421053	**0.776397516**	0.7650602
Precision	0.785185185	0.84	0.69736842	0.825	**0.84090909**	0.685714286	0.6923077
Recall	0.54047619	0.25	0.63095238	0.76744186	**0.86046512**	0.571428571	0.5357143
F1	0.736231884	0.3853211	0.6625	0.79518072	**0.85057471**	0.623376623	0.6040268
Prevalence	0.363636364	0.3636364	0.36363636	0.68253968	**0.68253968**	0.363636364	0.3636364
Detection Rate	0.16017316	0.0909091	0.22943723	0.52380952	**0.58730159**	0.207792208	0.1948052
Detection Prevalence	0.233766234	0.1082251	0.32900433	0.63492063	**0.6984127**	0.303030303	0.2813853
Balanced Accuracy	0.702414966	0.6113946	0.7372449	0.70872093	**0.75523256**	0.710884354	0.6998299

Fig. 12. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Pima Diabetes data(a)

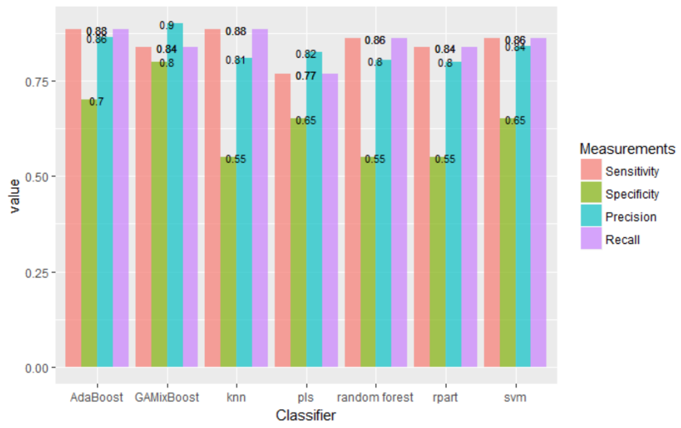


Fig. 13. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Sonar data(b)

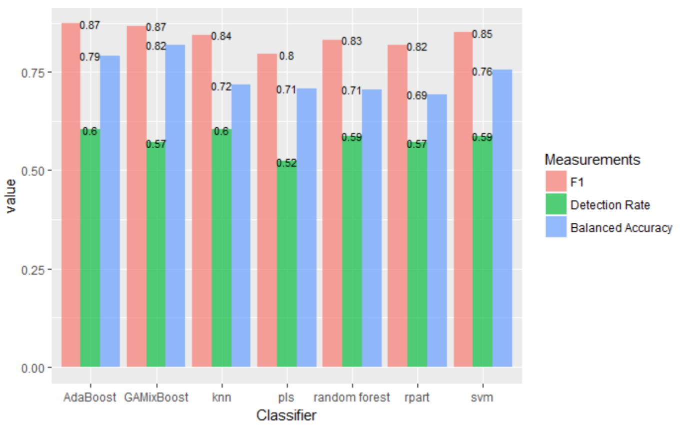


Fig. 14. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Sonar data(c)

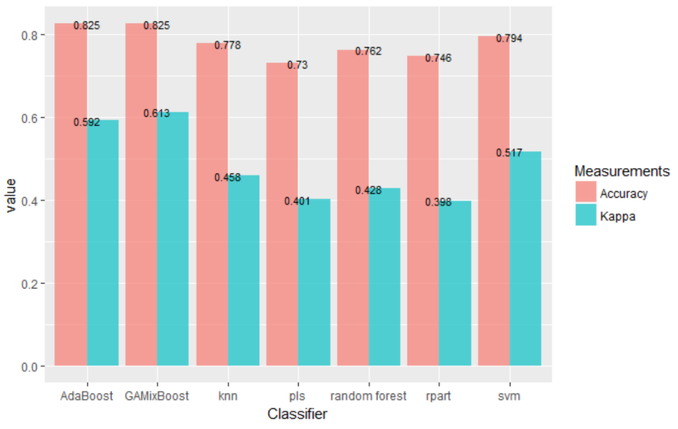


Fig. 15. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Sonar data(d)

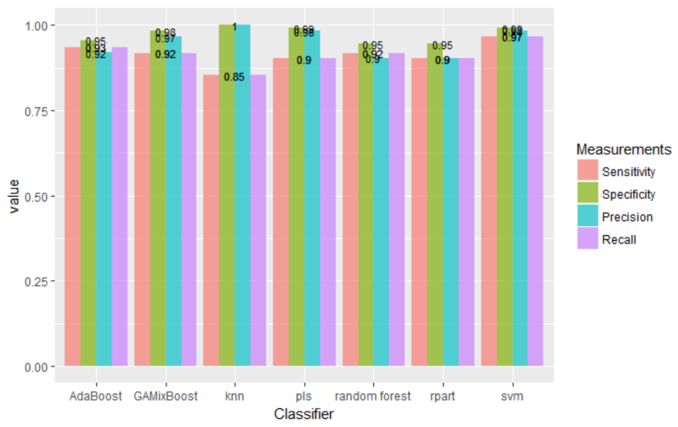


Fig. 16. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Wisconsin Breast Cancer data(b)

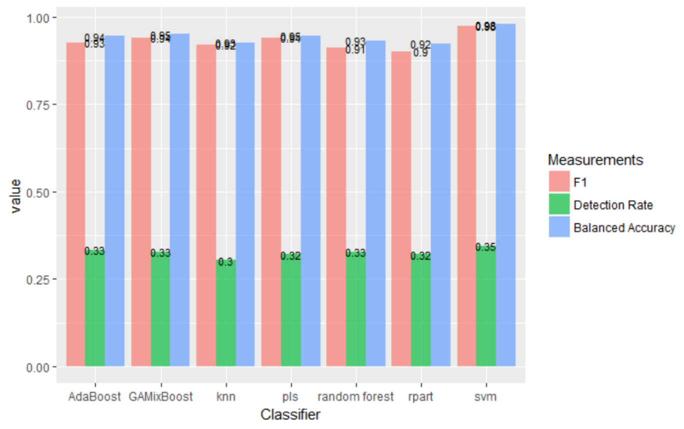


Fig. 17. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Wisconsin Breast Cancer data(c)

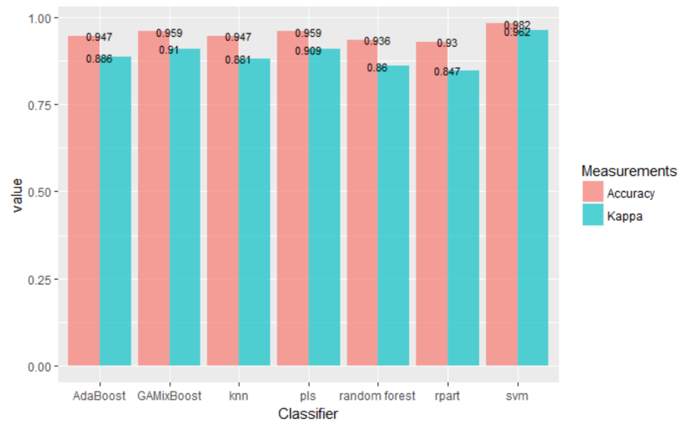


Fig. 18. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Wisconsin Breast Cancer data(d)

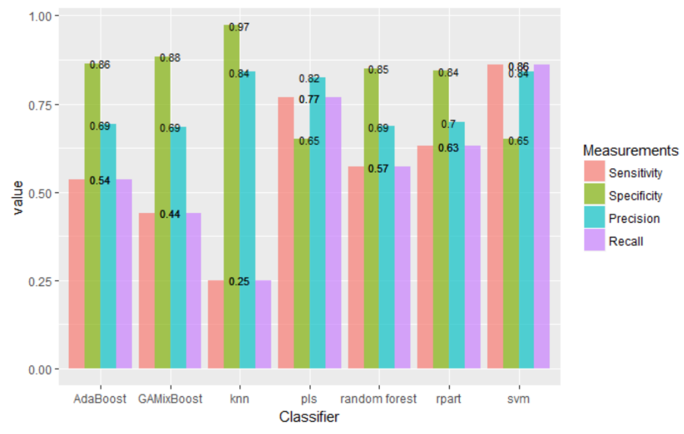


Fig. 19. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Pima Diabetes data(b)

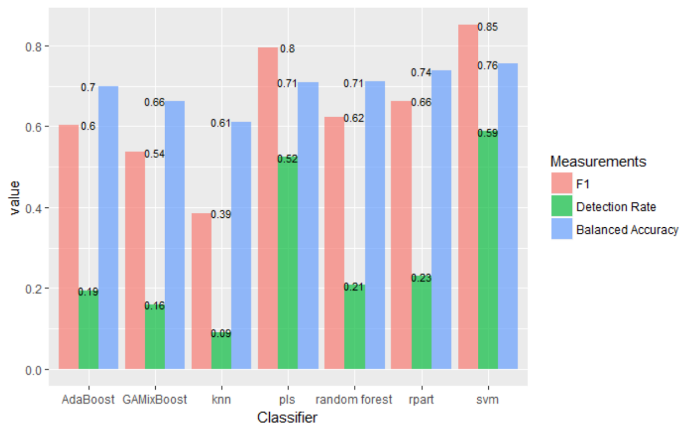


Fig. 20. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Pima Diabetes data(c)

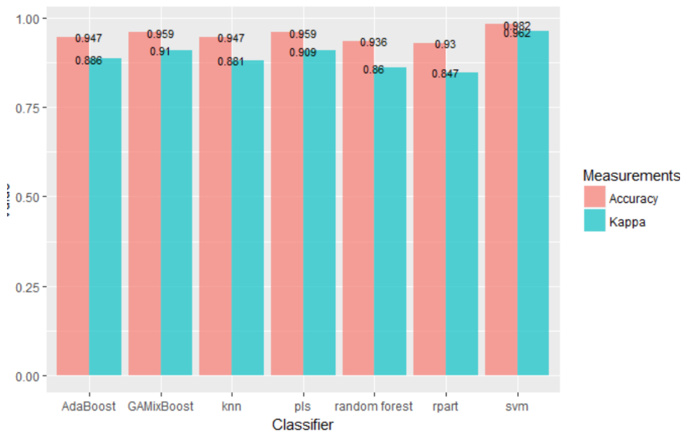
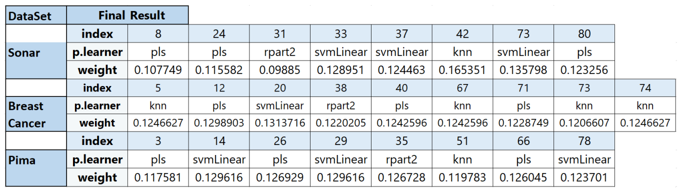


Fig. 21. Classification performance of GAMixBoost, base learners and common ensemble classifiers for Pima Diabetes data(d)

penalized by model complexity.
Based on test result on several real world datasets from UCI Machine Learning Repository, existed ensemble classifiers such as Random Forest and AdaBoost, and base learner used in GAMixBoost model are used to compare with my GAMixBoost model. The results show that GAMixBoost performs almost the same as AdaBoost, and better than its base weak learner in most evaluation measurements such as Accuracy, Sensitivity and Precision in Sonar dataset. In the Wisconsin Breast Cancer dataset, although the best performance model is SVM, but my GAMixBoost model performs better than Random Forest and AdaBoost in Accuracy, Specificity, Precision and F-1 Score.
For the GA search part, there exist some trade-offs between the probability of mutation and probability of crossover, here I focus on the crossover type, which leads to more stable(less fluctuation) converge and



DataSet	Final Result									
	index	8	24	31	33	37	42	73	80	
Sonar	p.learner	pls	pls	rpart2	svmLinear	svmLinear	knn	svmLinear	pls	
	weight	0.107749	0.115582	0.09885	0.128951	0.124463	0.165351	0.135798	0.123256	
	index	5	12	20	38	40	67	71	73	74
Breast Cancer	p.learner	knn	pls	svmLinear	rpart2	pls	knn	pls	knn	knn
	weight	0.1246627	0.1298903	0.1313716	0.1220205	0.1242596	0.1242596	0.1228749	0.1206607	0.1246627
	index	3	14	26	29	35	51	66	78	
Pima	p.learner	pls	svmLinear	pls	svmLinear	rpart2	knn	pls	svmLinear	
	weight	0.117581	0.129616	0.126929	0.129616	0.126728	0.119783	0.126045	0.123701	

Fig. 22. Result of GA selection for Sonar, Wisconsin Breast Cancer, Pima Diabetes datasets.

faster converge combination of P_m and $P_{crossover}$, with all other parameters fixed.

B. Future Improvement

Since I only selected the fixed four classification types as my base weak learner, with only one parameter varying to generate the pool of weak learner, more things can be done to improve the model performance. For example, different classifiers type could be used as base weak learner for boosted learning, with trials of both relatively poor learning ability, such as Single Rule Classification(OneR), or more completed learners such as Neural Network. Also, more parameters can be tried for GA search, such as population size, selection type and penalty functions. In the fitness function, problems about the outliers and imbalanced classes, which are critical problems for AdaBoost, can be solved by adding penalization parts, as illustrated in [8] and [9]. But again, expensive computational costs and worse interpretability of final model are issues should be paid more attention.

References

[1] Cox E. Fuzzy modeling and genetic algorithms for data mining and exploration. Elsevier/Morgan Kaufmann; 2005.

[2] Y. Yao, Z. Fu, X. Zhao and W. Cheng, ”Combining Classifier Based on Decision Tree,” icie, vol. 2, pp.37-40, 2009 WASE International Conference on Information Engineering, (2009).

[3] K. C. Ying, S.-W. Lin, Z. J. Lee, and Y. T. Lin, ”An ensemble approach applied to classify spam e-mails,” Expert Syst. Appl., vol. 37, pp. 2197-2201, (2010).

[4] Tolba, M., Moustafa, M. (2016). GAdaBoost: Accelerating Adaboost Feature Selection with Genetic Algorithms. arXiv preprint arXiv:1609.06260.

[5] Oh, Dong-Yop. GA-Boost: a genetic algorithm for robust boosting. The University of Alabama, 2012.

[6] Elyan, Eyad, and Mohamed Medhat Gaber. ”A genetic algorithm approach to optimising random forests applied to class engineered data.” Information Sciences 384 (2017): 220-234.

[7] Zhang, Chun-Xia, Jiang-She Zhang, and Sang-Woon Kim. ”PBoostGA: pseudo-boosting genetic algorithm for variable ranking and selection.” Computational Statistics 31, no. 4 (2016): 1237-1262.

[8] Haque, Mohammad Nazmul, Nasimul Noman, Regina Berretta, and Pablo Moscato. ”Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification.” PloS one 11, no. 1 (2016): e0146116.

[9] Oh, Dong-Yop, and J. Brian Gray. ”GA-Ensemble: a genetic algorithm for robust ensembles.” Computational Statistics 28, no. 5 (2013): 2333-2347.

[10] Zhang, Zili, and Pengyi Yang. ”An ensemble of classifiers with genetic algorithmBased Feature Selection.” The IEEE intelligent informatics bulletin 9, no. 1 (2008): 18-24.

[11] de Arajo Padilha, Carlos Alberto, Dante Augusto Couto Barone, and Adrio Duarte Dria Neto. ”A multi-level approach using genetic algorithms in an ensemble of Least Squares Support Vector Machines.” Knowledge-Based Systems 106 (2016): 85-95.

[12] Fu, Zhiwei, Bruce L. Golden, Shreevardhan Lele, S. Raghavan, and Edward A. Wasil. ”A genetic algorithm-based approach for building accurate decision trees.” INFORMS Journal on Computing 15, no. 1 (2003): 3-22.