Evolving Aggregators for Classification Ensembles to Optimize Prediction Accuracy

Chris Wells cwells18@jh.edu
Johns Hopkins University

Abstract

Ensemble classifiers have become increasingly popular within the field of machine learning due to their ability to generalize, and the use of Evolutionary Algorithms (EAs) to form ensembles is likewise being explored. We attempted in this project to evolve an aggregator that would combine the output of an ensemble of base classifiers to make better predictions than a simple majority vote and to use self-adaptation to further improve the accuracy of the evolved individuals. Unfortunately, none of the implementations attempted were able to match the performance of the majority vote method, and analysis of the evolutionary algorithms showed little learning, producing aggregators with similar levels of fitness over the course of generations. The result points out the inherent problem of attempting to optimize a system that is designed for robustness and the ability to generalize to yet-unknown data.

1. Introduction

Ensembles have become increasingly popular in the field of machine learning due to their simplicity and ability to generalize to unknown data. The aim of the ensemble is not to make a single predictor that can handle any conceivable test data, but to generate a diverse population of predictors and ensure that the base predictors' errors apply to different subsets of the data. If this is the case, then as long as the base predictors perform with better than average accuracy and there are a sufficient number of predictors, the ensemble as a whole will be able to generalize to unknown data better than an individual predictor.

In order to generate an ensemble, a population of base models are trained and used to make predictions which are then combined in some way to produce the final ensemble prediction. In order to not make identical predictions, the base models are trained in different ways: Either by taking random subsets of the training data in a "bagging" approach, or by training successive classifiers over distributions of the training data that change over time in a "boosting" approach.

A simple way to combine the predictions of the base predictors is a majority vote system where each predictor receives a single vote and the prediction with the most votes is the ensemble prediction. We believed that this method, while effective, was not ideal since some of the base predictors would be more accurate and robust than others, having been trained on a "better" distribution of the training set that more closely matches the kind of unknown data that we will be seen in the test set. The problem then, would be exploring the different combinations of classifiers.

Given the ability of Evolutionary Algorithms (EA) to search a solution space and identify optimal values, we objective of this project was to evolve an aggregator that will combine the outputs of a number of base classifiers. **We hypothesized that this evolved aggregator**

would produce more accurate results compared to a simple majority vote due to optimally combining the base classifiers. Furthermore, we hypothesized that incorporating a self-adaptation strategy into the EA would further increase the accuracy of the aggregator.

2. Related Work

This project's inspiration was the the evolutionary generation of classifier ensembles shown by Lacy et al. (2015b) and Souza da Silva et al. (2021). In both cases the combination of the base classifiers was evolved and used to make predictions. The use an evolutionary algorithm to construct a non-linear voting scheme is further investigated by Lacy et al. (2015a).

An alternative method of evolutionary ensemble construction is presented by Kim et al. (2006), where base classifiers are evolved as above, but where the ensembles also compete for base classifiers with small size, optimally-built ensembles are created. Another alternative to create an ensemble of classifiers is presented by Oliveira et al. (2005) using a multi objective genetic algorithm. While more complex than the method we are building on for this project, both ensembles demonstrated improvements over their individual classifiers and against other ensemble methods.

3. Approach

3.1 Evolutionary Algorithms

Our first approach is to evolve an aggregator using the classic genetic algorithm described in Holland (1975), shown below:

Algorithm 1 Simple Genetic Algorithm

$t \leftarrow 0$
Initialize population $P(t)$
Evaluate population fitness using $f(t)$
while not terminated **do**
  $t \leftarrow t + 1$
  Select $C(t)$ from $P(t-1)$
  Recombine and mutate $C(t)$, producing an updated population $C'(t)$
  Evaluate $C'(t)$ fitness using $f(t)$
  Replace population with selections from $C'(t)$ and $P(t-1)$
end while

In this algorithm, a population of potential solutions is generated and in each generation (execution of the while loop), selected individuals are reproduced by creating "children" which can take characteristics of their parents (if crossover is used) or whose characteristics are randomly changed (if mutation is used). The fitness of individuals are judged by a fitness function $f(t)$ and selected children replace individuals from the older generation according to a selection method. As the population changes over the course of generations,

the optimal solutions (those with the best fitness) are discovered. In this way, we hoped to find the optimal combination of base classifiers.

3.2 Expression Trees

An expression tree is a data structure where each node in the tree represents an operator, variable, or constant value. Each operator node will have one or more children representing input operators, variables, or constants and all leaf nodes will be variables or constants. A simple expression tree for $a + (b \times 2)$ is shown in Fig. 1.

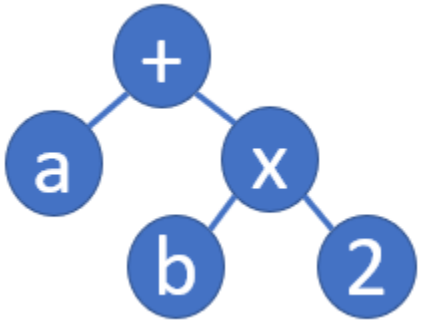

Figure 1: A basic expression tree, representing $a + (b \times 2)$

Using the output of the base classifiers as variables a population of expression trees could be evolved to explore and find the optimal combination of base classifiers. The accuracy of the predictions would be used as the fitness value of the individual expression trees in this case.

3.3 Defining an Individual - Expression Tree Aggregator

In this definition of the individual aggregator as an expression tree, there would be two operations: addition and multiplication representing combinations of base classifier votes or modifying one classifier's vote, either element-wise or operating on every element in a list with a constant. Because we do not want to exclude base classifiers from the ensemble, there will be no subtraction operator. The leaves of the tree then, would be the output of the base classifiers and constant values of the range [0.1, 1.0). In order to avoid biasing the results however, the base classifier outputs must be properly encoded and we have implemented this in the same way as the aggregators created in Souza da Silva et al. (2021), shown below.

Because the base classifier predictions are categorical values, they will need to be one-hot encoded and input to the expression tree as c-dimensional lists, with c being the number of possible output classes. The operations in the tree will then perform element-wise on the inputs and the final prediction will be index of the highest element in the output list. If a list and constant are inputs, the constant is applied to every member of the list. For example, aggregating a simple true/false prediction made by base classifiers a and b in the

Fig. 1, where a predicts 1 and b predicts 0 would be performed as:

$$a = [0, 1], b = [1, 0]$$
$$[0, 1] + (2 \times [1, 0]) = [0, 1] + [2, 0] = [2, 1]$$
$$y_{pred} = 0$$

The comparison of y_{pred} to the true y_{test} would be the fitness of the individual.

The initial population of trees will be randomly generated, with a depth between a range of values. For the sake of simplicity, reproduction will be carried out only using mutation in this implementation. In each generation, the population of trees will be cloned and each cloned individual will be mutated by replacing a randomly-chosen node with a different operator or terminal value. An example of this mutation is given in Fig. 2, with the $\times$ operator replaced by $+$.

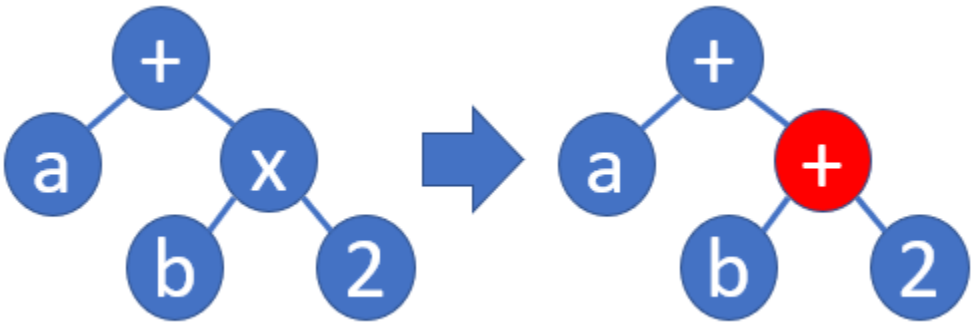

Figure 2: An example of random node mutation, where a randomly-chosen node is replaced by a different operator, variable, or constant value.

Due to its speed and efficiency, a tournament selection process will be used. Every generation, individuals will be randomly chosen to compete against each other on the basis of fitness, with the winner being selected for the next generation. The population size p will remain stable, so every generation, the $2 \times p$ individuals consisting of the originals and cloned/mutated offspring will compete against each other in order to form the p-size next generation.

3.4 Defining an Individual - Weighted Aggregator

The poor results of the expression tree aggregator shown in Section 5 revealed several problems with the concept. Over a number of generations, simpler trees proved to be more fit and the top-performing aggregator almost inevitably contained only the output of one or two base classifiers being combined with constants or even themselves. The depth limit for the tree severely restricted the number of classifiers in the ensemble, especially since the leaves might include constant values as well. A simpler tree that could support any number of classifiers and not exclude them was necessary, so we re-thought the concept of the

individual as a weighted aggregator, using Evolutionary Strategy invented by Rechenberg (1965).

Shown in Fig. 3, the individual expression tree now always has the same depth of 3, the same operators, and the same number of variables. Predictions are made by multiplying the output of each classifier by a weight, and summing these weighted predictions like so, with N being the number of classifiers and b_n being the output of a single base classifier:

$$y_{pred} = \sum_{n=1}^{N} w_n b_n$$

As with the original aggregator, the index of the highest value in the c-dimensional list y_{pred} is the predicted class.

Instead of being an expression tree, an individual in this implementation is an N-length list, with each element representing the weight to assign to each base classifier's output. By keeping all weights within the range (0,1], we can ensure that no base classifier is excluded from the output and one or two classifiers do not dominate the ensemble. As with the previous implementation, selection will be conducted via tournament and reproduction will be handled by mutation only for simplicity. With node replacement not being possible in a list however, the mutation operator will be changed.

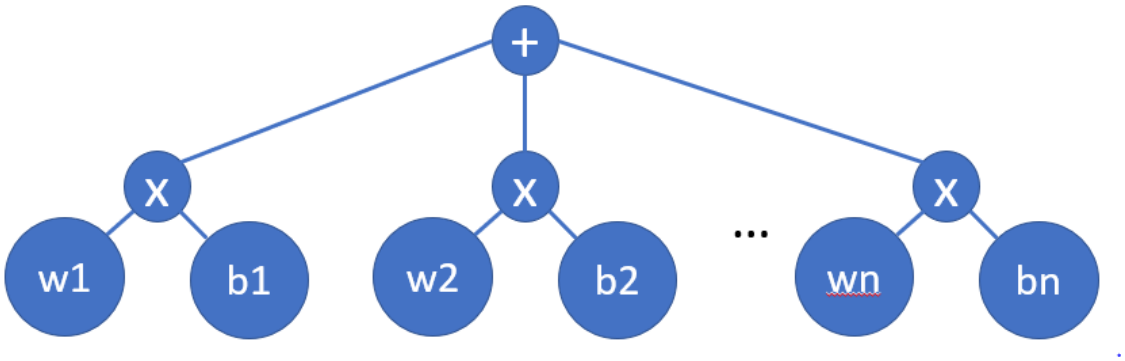

Figure 3: The weighted aggregator. Instead of being represented directly by an expression tree, the individual in the algorithm will consist of an N-length genome containing the weights to multiply each base classifier's output by.

Instead of a node replacement, there is now a static probability P_{mut} that each element in the list will be mutated. Since each weight is a real value between 0 and 1, the mutation will be "noise" calculated from a normal distribution. In order to ensure that the noise can be positive or negative and that large "swings" between the minimum and maximum weight do not occur we have chosen $\mathcal{N}(0, 0.1)$ as our distribution. If chosen for mutation, a given weight w_i will be updated as follows:

$$w_i = w_i + \mathcal{N}(0, 0.1)$$

Different values for the distribution's standard deviation σ will result in different behavior in the evolutionary algorithm as the mutated children will be more or less different than

their parents. Therefore, in choosing a σ value we must balance the need to rapidly search for promising new combinations while "refining" combinations that already work well. A modification to the weighted aggregator will explore the possibility of modifying σ over the evolutionary run in order to ensure this balance.

3.5 Self-Adaptation

As described in Engelbrecht (2007), instead of using a single σ value to determine the mutation strategy across every attribute of the individual, we can first allow each attribute i to have its own strategy σ_i that is incorporated into the individual's genome. While previously, a weighted aggregator for N classifier's could be expressed as an N-length list:

$$< w_1, w_2, ..., w_N >$$

We can now define each individual as a $2N$-length list:

$$< w_1, w_2, ..., w_N, \sigma_1, \sigma_2, ..., \sigma_N >$$

Using the log-normal self-adaptation method, the strategy parameters σ_i can be adjusted for each individual like so:

$$L = \tau' N(0,1) + \tau N(0,1), \text{ using the learning rate } \tau$$
$$\tau' = \frac{1}{\sqrt{2\sqrt{N}}}, \tau = \frac{1}{\sqrt{2N}}$$
$$\sigma_i'(t) = \sigma_i(t) e^L$$

A simple reinforcement-based method can also be incorporated that will include a positive θ value for improved fitness and a negative value for decreased:

$$\theta(t) = \begin{cases} 0.5 & \text{if } fitness_{new} - fitness_{old} > 0 \\ 0 & \text{if } fitness_{new} - fitness_{old} = 0 \\ -1 & \text{if } fitness_{new} - fitness_{old} < 0 \end{cases}$$

For the sake of simplicity, we are only looking back one generation and using the latest "old" fitness value for this implementation. The final update rule for the strategy parameter σ_i then, will be:

$$\sigma_i'(t) = \sigma_i(t) e^{\theta(t)|L|}$$

4. Experiment Design

We hypothesized that an evolved aggregator for an ensemble of base classifiers will produce more accurate results compared to a simple majority vote of the base classifiers due to producing an optimal combination of the base classifiers. To test this we produced an ensemble of base classifiers, recorded their accuracy using a majority vote, and compared this to the accuracy produced by the aggregator's prediction.

Furthermore, we hypothesized that incorporating a self-adaptation strategy into the algorithm would further improve the aggregator's performance. Using the simple reinforcement based method discussed in our approach with the weighted aggregator, we compared the accuracy of the weighted aggregator produced by the modified algorithm with the original weighted aggregator.

4.1 Datasets

We will be using a selection of the datasets used by Lacy et al. (2015b) from the UCI repository: Breast Cancer (Wisconsin) by Street et al. (1995), Liver Disorders by BUPA (1995), and Chess (King-Rook vs. King-Pawn) by Shapiro (1989) . The Breast Cancer and Liver datasets were used as they had the highest and lowest accuracy scores in Lacy et al. (2015b). Since both datasets had a relatively low number of instances and a low number of numerical features, we also included the Chess dataset, which was much larger and was composed of categorical features. Full dataset information is listed in Table 1.

Dataset	**Instances**	**Num. Features**	**Cat. Features**
Breast Cancer (Wisconsin)	699	9	0
Chess	3196	0	36
Liver	341	6	0

Table 1: Datasets used in this project

4.2 Defining a Base Classifier

For the base classifier, we used a simple DecisionTreeClassifier with the default settings. Since we did not specify a maximum depth, we expected that these individuals would have poor generalization, with a high accuracy on the training set and significantly lower accuracy on the test set. The accuracy was the averaged value of the results taken with 5-fold Cross Validation (CV).

The results are shown in Table 2. As expected, the DecisionTreeClassifier was able to fit perfectly to the training set and suffered a loss of accuracy in the test set. While the drop-off was minor in the Breast Cancer (Wisconsin) and Chess datasets, it was much more significant in Liver, so there was more scope for our ensembles to improve the accuracy of the latter.

Dataset	**Instances**	**Training Accuracy**	**Test Accuracy**
Breast Cancer (Wisconsin)	699	1.000 $\pm$0.000	0.923 $\pm$0.021
Chess	3196	1.000 $\pm$0.000	0.977 $\pm$0.011
Liver	341	1.000 $\pm$0.000	0.664 $\pm$0.056

Table 2: Datasets and base classifier accuracy

Next was the creation of an ensemble of DecisionTreeClassifiers. We used a simple bagging strategy introduced by Breiman (1996), where a fraction f of the training set was sampled with replacement and used to train a number N of base classifiers. To find the

optimal f and N values, a number of combinations were tested. The results are shown in Table 3. Again, the accuracy was the averaged value of the results using 5-fold CV.

For the Breast Cancer and Liver datasets, the ensemble showed clear improvement over the individual DecisionTreeClassifier and highlighted its better generalization. In the case of the Chess dataset however, the ensemble did not show any additional improvement even after greatly increasing the classifier population, possibly due to the already high accuracy of the base classifier. The highest-performing of the ensembles, bolded in Table 3, will be compared against the evolved aggregators. To ensure a fair comparison, these aggregators will use the same N number of classifiers where possible and identical f values.

4.3 Evolving an Express Tree Aggregator

The process of evolving tree aggregator consisted of two stages: Training the base classifiers followed by evolving the aggregator. To ensure that the aggregator is able to generalize, the training data was itself split with 80% of the instances used to train the base classifiers and the held-out 20% used to evaluate the aggregator trees as they evolved. After a specified number of generations, the most fit aggregator was used to make predictions on the the test data which had been held back from both the base classifier and aggregator training and its accuracy was compared to the predictions made by a simple majority vote. In order to measure how complex the evolved trees are, the number of nodes of the most fit tree were also recorded at the end of the run.

We initially specified the population size and termination criteria of the evolutionary algorithm on the basis of what could be completed in a reasonable time, settling on a population of 100 and 25 generations, with the idea that either parameter could be increased if the results did not converge by then. As shown in the results below, the fitness values largely remained static and experiments with larger populations increased the time of runs while offing no change in the fitness progression, so these values remained the defaults for the evolutionary runs.

As stated in the approach, all non-leaf nodes in the trees were operations with 2 inputs - either addition or multiplication. All leaf nodes were either the output of one of the base classifiers or a constant value of the range [0.1, 0.9]. After cloning a new generation, each child had one node randomly mutated. Given restrictions on tree depth to ensure completion of the evolutionary run in a reasonable time, only 20 base classifiers were used in the ensemble, with the f value used the same as the highest-performing ensemble in Table 3. Accuracy was the averaged value of the results using 5-fold CV.

4.4 Evolving a Weighted Aggregator

As with the expression tree above, the process of evolving the weighted aggregators consisted of two stages: Training the base classifiers followed by evolving the aggregators. The base classifiers were trained on 80% of the training set while the aggregators' fitness was evaluated using the remaining 20%. After cloning a new generation from the population of 100, mutation was conducted by adding gaussian noise with a static probability of 10% for each attribute. With the self-adapting aggregator the sigma value was adjusted after the fitness was re-evaluated.

Parameter Settings	DT Accuracy	Ensemble Accuracy
Breast Cancer (Wisconsin)	0.923 ±0.021	
N=100, f=0.2		0.957 ± 0.027
N=100, f=0.35		0.960 ± 0.028
N=100, f=0.5		0.960 ± 0.024
N=300, f=0.2		**0.961 ± 0.026**
N=300, f=0.35		0.960 ± 0.027
N=300, f=0.5		0.960 ± 0.027
N=500, f=0.2		0.959 ± 0.026
N=500, f=0.35		0.959 ± 0.026
N=500, f=0.5		0.957 ± 0.029
Chess	0.977 ±0.011	
N=100, f=0.2		0.957 ± 0.030
N=100, f=0.35		0.973 ± 0.013
N=100, f=0.5		0.972 ± 0.016
N=300, f=0.2		0.961 ± 0.025
N=300, f=0.35		0.974 ± 0.013
N=300, f=0.5		0.973 ± 0.016
N=500, f=0.2		0.961 ± 0.025
N=500, f=0.35		**0.975 ± 0.012**
N=500, f=0.5		0.975 ± 0.015
N=1000, f=0.2		0.961 ± 0.026
N=1000, f=0.35		0.974 ± 0.013
N=1000, f=0.5		0.975 ± 0.015
Liver	0.664 ±0.056	
N=100, f=0.2		0.710 ± 0.037
N=100, f=0.35		**0.754 ± 0.048**
N=100, f=0.5		0.713 ± 0.036
N=300, f=0.2		0.754 ± 0.049
N=300, f=0.35		0.725 ± 0.034
N=300, f=0.5		0.716 ± 0.030
N=500, f=0.2		0.725 ± 0.032
N=500, f=0.35		0.728 ± 0.031
N=500, f=0.5		0.730 ± 0.034

Table 3: Ensemble tuning results

After 25 generations the most fit aggregator's predictions were compared with the majority vote predictions in the same manner as the expression tree. The N and f values used were the same as those of the highest-performing ensemble in Table 3. Accuracy was the averaged value of the results using 5-fold CV.

5. Results and Discussion

5.1 Expression Tree Aggregator

As shown in Table 4, the Expression Tree Aggregator performed poorly, with accuracy comparable to that of the individual DecisionTreeClassifier in all datasets. Although the randomly-generated populations of trees had varying numbers of nodes at initialization, over a number of generations the most fit trees were found to be either equivalent to one base classifier or a simple combination of two or three. For the five folds in the Liver dataset for example, the most fit trees used in the test set predictions were:

- **add**(**add**(C17, C6), C9)
- **add**(C15, C5)
- **mul**(C13, 0.8)
- **add**(C5, C17)
- **mul**(C18, 0.9)

Since so few classifiers were being employed, it unsurprising that the test accuracy was comparable to that of the individual classifier. The results of this aggregation strategy inspired the weighted aggregator approach covered above to ensure that all classifiers were being employed.

Accuracy	**Majority Vote**	**Tree Aggregator**	**Node Count**
Breast Cancer (Wisconsin) $N = 20, f = 0.2$	0.961 ± 0.026	0.933 ± 0.039	4.2
Chess $N = 20, f = 0.35$	0.975 ± 0.012	0.971 ± 0.012	3.0
Liver $N = 20, f = 0.35$	0.754 ± 0.048	0.620 ± 0.038	4.0

Table 4: Average accuracy of the majority vote compared to the Expression Tree Aggregator and average number of nodes in the highest-performing tree

5.2 Weighted Aggregator

As shown in Table 5, the weighted aggregator showed higher accuracy than the expression tree aggregator for the Breast Cancer and Liver datasets. It worsened slightly on the Chess dataset, possibly due to the higher performance of the individual classifier on this data as shown in the initial creation and testing of the ensemble.

The use of minimum and maximum weights also ensured that all classifiers continued to play a role in the ensemble. The means and standard deviation of the weight values in the top-performing individual also showed that weight values were spread out, with some base classifiers clearly being weighed more than others in the final count.

Unfortunately, this still did not result in the aggregator being able to outperform the majority vote. Plotting the average fitness value over a number of generations shown in Figure 4 demonstrated only a slow increase in fitness for Liver until plateauing in about 5 generations, while the other datasets began with high fitness values and stayed there. This indicates that the changes in the population did not significantly impact fitness and that there might not be an "optimal" weight configuration that is as robust as the majority vote.

Accuracy	**Majority Vote**	**Aggregator**	μ, σ
Breast Cancer (Wisconsin) $N = 300, f = 0.2$	0.961 ± 0.026	0.960 ± 0.023	0.504, 0.283
Chess $N = 500, f = 0.35$	0.975 ± 0.012	0.969 ± 0.018	0.501, 0.290
Liver $N = 100, f = 0.35$	0.754 ± 0.048	0.716 ± 0.028	0.492, 0.320

Table 5: Average accuracies of the evolved weighted aggregator and average mean and standard deviation of the top individual's weights

5.3 Self-Adapting Weighted Aggregator

As shown in Table 7, the addition of self-adaptation did not impact performance of the aggregator. The results of the 5-fold CV run showed slight improvement over the weighted aggregator in Table 5, but after taking 10 runs for both the unmodified and modified Weighted aggregators on the Liver dataset (results shown in Table 6) and applying the Wilcoxon Ranked Sum Test, the result was found to not be significant for $p < 0.01$ and $p < 0.05$. Due to the longer execution time, we were unable to collect enough samples for the Breast Cancer or Chess datasets.

Run #	**Unmodified**	**Self-Adapting**
1	0.710	**0.724**
2	0.754	**0.783**
3	0.696	**0.724**
4	**0.739**	0.710
5	**0.739**	0.696
6	**0.768**	0.739
7	**0.710**	0.696
8	**0.725**	0.667
9	0.725	**0.739**
10	**0.710**	0.681

Table 6: Run accuracies used in Wilcox Ranked Sum Test

As shown in Fig. 4, the fitness over generations did not significantly differ from the original Weighted Aggregator. Like with the unmodified Weighted Aggregator, the fitness largely stays the same over the course of the run.

The almost unchanged performance of the modified method indicated that there was an problem inherent to our strategy of weighing aggregators. When the aggregator was trained on a selection of training data and optimized to be as accurate as possible on that data, by necessity certain classifiers that would have contributed to test set accuracy were being de-emphasized by being given lower weights. As a result, the test set accuracy did not exceed that produced by the majority vote in any of our implementations.

Accuracy	**Majority Vote**	**SA Aggregator**	μ, σ
Breast Cancer (Wisconsin) $N = 300, f = 0.2$	0.961 ± 0.026	0.960 ± 0.015	0.522, 0.348
Chess $N = 500, f = 0.35$	0.975 ± 0.012	0.970 ± 0.017	0.513, 0.345
Liver $N = 100, f = 0.35$	0.754 ± 0.048	0.725 ± 0.058	0.431, 0.334

Table 7: Average accuracies of the evolved self-adapting weighted aggregator and average mean and standard deviation of the top individual's weights

6. Conclusions

Given the strength of ensemble-based classification, we attempted in this project to evolve an aggregator that would combine the output of an ensemble of base classifiers to make better predictions than a simple majority vote. This would take advantage of the ability of EAs to search a solution space and identify optimal values. We first attempted to create an Expression Tree Aggregator using the classic genetic algorithm., and then a simpler Weighted Aggregator using Evolutionary Strategy, both with and without self-adaption.

The Expression Tree Aggregator did not perform as expected. Instead of exploring a variety of combinations of base classifiers, the simplest trees tended to be selected and the resulting predictions were comparable to those made by an individual classifier. Nor was the architecture of the tree scalable to the large number of base classifiers needed for consistent ensemble performance.

The Weighted Aggregators while almost as accurate as the majority vote, still did not exceed it despite avoiding the problems of the previous implementation. Over a number of generations, the individuals should evolve to better classify the test data and this was not accomplished, with the population of aggregators largely remaining at the same fitness level over the course of the evolutionary run. Incorporating self-adaptation likewise did not improve performance or make the aggregator a better alternative to majority vote. We would suggest that any future work on this concept focus instead on the fitness evaluation, specifically trying to find a value that captures the ability of the aggregator to generalize better than than simply trying to achieve the highest accuracy score.

Ultimately we believe that the objective of optimizing as we attempted to perform in this project was in conflict with the creation of a good ensemble and that attempting to replace the majority vote with a set of optimized weights was fundamentally flawed. An ensemble's robustness comes from the relative equality of the base classifiers, but as we refined and

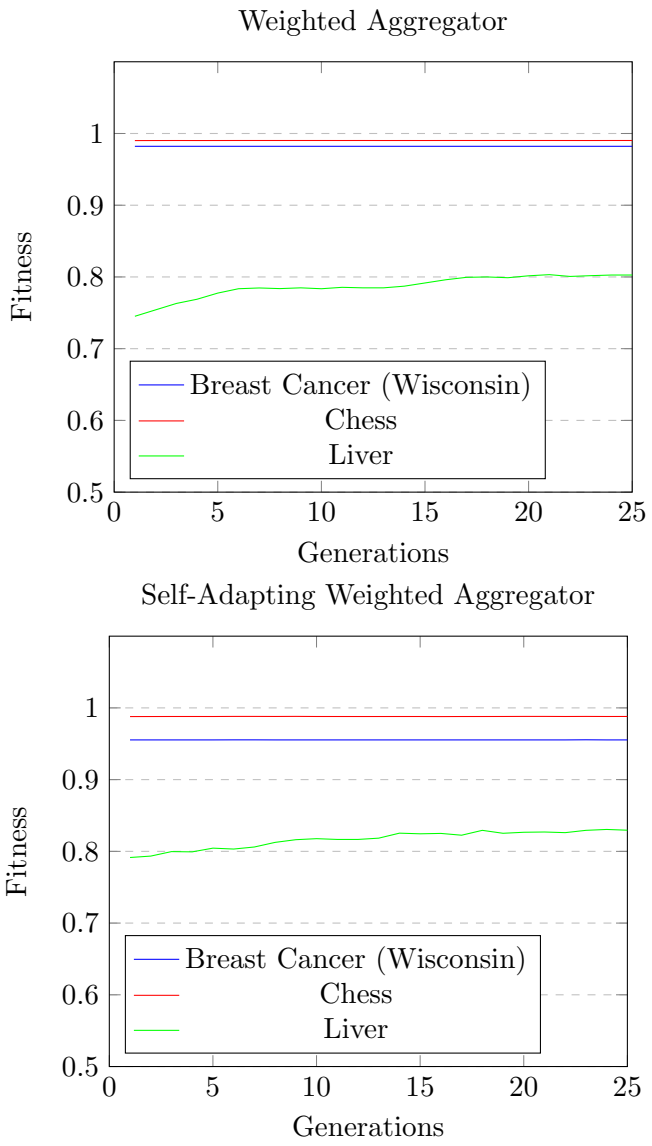

Figure 4: Average fitness over generations

optimized our weights on a training set we inherently biased our ensemble toward some of the base classifiers over others and reduced its ability to react to new information in the test set. **Given the additional complexity of our two-tiered training approach and the poorer overall performance, we would not recommend evolving an aggregator for an ensemble instead of making predictions using a simple majority vote.**

References

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

BUPA. Liver disorders data set, 1995. URL `https://archive.ics.uci.edu/ml/datasets/liver+disorders`.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction*, pages 187–211. John Wiley Sons, Ltd, 2007.

John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

YongSeog Kim, W. Nick Street, and Filippo Menczer. Optimal ensemble construction via meta-evolutionary ensembles. *Expert Systems with Applications*, 30(4):705–714, 2006.

Stuart E. Lacy, Michael A. Lones, and Stephen L. Smith. A comparison of evolved linear and non-linear ensemble vote aggregators. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 758–763, 2015a.

Stuart E. Lacy, Michael A. Lones, and Stephen L. Smith. Forming classifier ensembles with multimodal evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 723–729, 2015b.

Luiz S. Oliveira, Marisa Morita, Robert Sabourin, and Flávio Bortolozzi. Multi-objective genetic algorithms to create ensemble of classifiers. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 592–606, 2005.

I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Library Translation 1122*, 1965.

Alen Shapiro. Chess (king-rook vs. king-pawn), 1989. URL `https://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King-Pawn)`.

Ronnypetson Souza da Silva, Márjory Da Costa-Abreu, and Stephen Smith. Investigating the use of an ensemble of evolutionary algorithms for letter identification in tremulous medieval handwriting. *Evolutionary Intelligence*, 14(4):1657–1669, 2021.

W. Nick Street, Dr. William H. Wolberg, and Olvi L. Mangasarian. Breast cancer (wisconsin), 1995. URL `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)`.