Application of Genetic Algorithms and Multiple Hypotheses for Supervised Machine Learning

Sajika Gallege

Department of Computer Sciences
University of Wisconsin-Madison
1210 W. Dayton Street, Madison, WI 53706
sgallege@cs.wisc.edu

ABSTRACT

This paper describes the implementation of a Genetic Algorithm (GA) and a variant of learning multiple hypotheses. The GA is applied to a binary classification of a labeled dataset to evaluate the algorithm's performance in a supervised learning environment. The details include the basic GA implementation and the modifications done to create a variant of learning multiple hypotheses. The Paper also includes the results and a comparison to other Machine Learning Algorithms such as Decision Trees and Perceptrons. The discussion includes advantages such as the ability to parallelize the hypotheses search and drawbacks such as the time taken to select good hypotheses as well as propose modifications to enhance the algorithm.

INTRODUCTION

Genetic Algorithms loosely simulated biological evolution due to natural selection. Concepts such as reproduction, carrying on good traits to the next generation, and survival of the fittest are simulated in genetic algorithms. The population is represented by a set of hypotheses; in this algorithm's context a hypothesis is a bit string. Some hypotheses survive and pass on to the next generation with a probability based on their fitness, whereas some hypotheses crossover and produce siblings which join the new generation. The specific terms and the implementation are detailed in the next sections of the paper. In essence, GAs work by parallelizing the hypotheses space search. This is an interesting area to research because of the similarity to evolution which has proved successful in the natural world. This will also allow a comparison between other Machine Learning Algorithms analyzed over the semester.

In this implementation the GAs are used to learn and classify a labeled data set with fixed length features the label belonging to two classes. Two variants of the GA are implemented and tested in this project. The first version learns the single best hypothesis that meets a certain fitness threshold and applies the hypothesis to classify the test examples. The second version learns a set of hypotheses which are best hypotheses from all the generations and applies it for classification. The results from the GA are then compared to Decision Trees and Perceptrons to determine the performance of the algorithm.

One of the main drawbacks of Gas as discovered from the experiments is the amount of time taken to find a good hypothesis due to the randomness of the search. Limiting the time frame results in producing a non optimal hypothesis; thus the performance of the GA's seems low compared to the other algorithms. The multi hypotheses variant seems to have better performance than the single hypothesis algorithm due to better generalization.

PROBLEM DEFINITION AND ALGORITHM

Task Definition

The goal of the Genetic Algorithm is to learn to classify a dataset with fixed length features. More precisely the learning happens on a labeled set of training examples. Once the learning is complete, the algorithm is expected to be able to classify future (unobserved/ test) examples of a

similar format. The accuracy of the algorithm can be calculated by comparing the number of correctly classified test examples.

The input for the GA is provided in the form of a .names file that contains the data definitions and a .data file which contains the data samples. The .names file follows the format for each feature:

featureName *featureType* *possibleValues*

where

- *featureName* is a *token* (i.e. a character string without quotes and internal spaces)
- *featureType* is either discrete or continuous (for the output category, use output as the feature type)
- *possibleValues* is a comma-separated list of tokens specifying (a) the possible values for a discrete-valued feature *or* (b) the minimum and maximum values for a continuous-valued feature

The .data is a space delimited list of examples with the features appearing in the same order as the .names file.

To apply the GA, the examples in the dataset need to be converted to binary strings. The discrete features were converted to 1-of-n representation by adding a new boolean feature for each distinct value of the discrete feature. The thermometer representation was used for continuous features. The number digits can be specified by the user. The DataReader splits the difference between the upper bound and the lower bound as specified in the .names file into equal intervals. A new boolean feature is added after comparing the continuous value against the intervals.

Algorithm Definition

The Genetic Algorithm works by searching the hypotheses space for the best (fittest) hypothesis as specified by the fitness function. The basic algorithm follows the pseudo code given in Table 9.1 of Machine Learning by Tom Mitchell. The Multi-Hypothesis variant follows pseudo the code given below.

GeneticLearner(*Fitness, Fitness_threshold, p, r, m*)

Fitness: A function that assigns an evaluation score, given a hypothesis.
Fitness_threshold: A threshold specifying the termination criterion.
p: The number of hypotheses to be included in the population.
r: The fraction of the population to be replaced by Crossover at each step.

m: The mutation rate.
n: Number of best hypotheses to be learned
H: Array of n best hypotheses

Initialize population: P ← Generate *p* hypotheses at random
Evaluate: For each *h* in *P*, compute *Fitness(h)*
Add: $\max_h$ *Fitness(h)* to *H*
While [*H.best*] < *Fitness_threshold* do

Create a new generation, P_s:

1. *Select*: Probabilistically select (1 - r)p members of *P* to add to P_s. The probability $Pr(h_i)$ of selecting hypothesis hi from P is given by

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

2. *Crossover*: Probabilistically select pairs of hypotheses from P, according to $Pr(h_i)$ given above. For each pair, $< h_1, h_2 >$, produce two offspring by applying the Crossover operator. Add all offspring to *P*
3. *Mutate*: Choose *m* percent of the members of *P*, with uniform probability. For each, invert one randomly selected bit in its representation.
4. *Update*: P ← Ps
5. *Evaluate*: for each h in P, compute Fitness(h)
6. *Add*: $\max_h$ *Fitness(h)* to *H* if *H.worst* < $\max_h$ *Fitness(h)*

Return the set of hypotheses *H* that has the highest fitness

The Fitness is evaluated by using

$$Fitness\ (h) = \left[\frac{correct(h) - \frac{NumZeros}{|h|}}{|trainingSet|}\right]^2$$

A hypothesis *h* is considered correct for a given training sample *x* as follows: After a bitwise and operation, (*h and x = y*) if a disjunction (*or*) within the features and a conjunction (*and*) of the features match the actual label for the sample.

Example: Assume the following features with the possible values and binary encoding

Color:	red,	green,	blue
Encoding:	100,	010,	001
Size:	small	med.	large
Encoding:	100,	010,	001

Samples:
S1: 010 100 (green, small) label: 1
S2: 100 100 (red, small) label: 1

Hypotheses:
H1: 110 100 (red or green and small)
H2: 010 101 (green and small)

If we evaluate S1 using H1
010 100 and 110 100 = 110 100 = [1 v 1 v 0] ^ [1 v 0 v 0] = predict 1
If we evaluate S2 using H1
100 100 and 110 100 = 110 100 = [1 v 1 v 0] ^ [1 v 0 v 0] = predict 1
So H1 would be correct for both S1 and S2

Similarly, if we evaluate S1 and S2 using H2 we get
010 100 and 010 100 = 010 100 = [0 v 1 v 0] ^ [1 v 0 v 0] = predict 1
100 100 and 010 100 = 000 100 = [0 v 0 v 0] ^ [1 v 0 v 0] = predict 0
So H2 would only be correct for S1.

Thus, H1 would get a higher fitness score; also we enforce a penalty for having zeroes to avoid over fitting. Note that a more general hypothesis contains more ones, and zeroes make the hypothesis less general.

For classifying the test examples a procedure similar to the above example is used. For the variant version of multiple hypotheses all the learned hypotheses are combined together using a bitwise 'or' to create a more general hypotheses, the same procedure described above is used for classification.

EXPERIMENTAL EVALUATION

Methodology

The performance of the Genetic Algorithm is evaluated by the test set accuracy. The primary dataset used for this experiment is the fuel consumption of the 2010 model year motor vehicles obtained from http://www.fueleconomy.gov. The Algorithm's task is to learn to classify if a given car is a guzzler (a vehicle is classified as a guzzler if the fuel consumptions is greater than 18 miles per gallon). The features of the dataset include manufacture country, engine displacement, number of cylinders, number of gears, transmission type, drive type, vehicle classification, air aspiration method and variable valve timing. This dataset contains real data and has a very practical application as the learned concept can reflect the contribution of the features to the fuel consumption of a vehicle. The Weather dataset by Andrew W Bender and the Internet

usage dataset by Natasha Eilbert obtained from the CS760 shared data repository were also used to evaluate the Genetic Algorithm.

The data is formatted as a .names file and a .data file as described in the task definition. For cross validation purposes, samples were randomly picked from the dataset to create 10 train/ test folds, such that the test set consists of 10 percent of the data and the train set consists of 90 percent of the data. The datasets used for the experiment contain roughly 1000 samples. By using the same train/test folds on the other Machine Learning Algorithms and comparing the test set accuracies using a paired t-test we are able to determine if the GAs are statistically significantly different from the other algorithms.

It is difficult to draw any assumptions on how the Genetic Algorithm would perform compared to other Machine Learning algorithms. The multi-hypotheses Genetic Algorithm is expected to perform better than the basic version, since it learns a set of hypothesis which are best from all the generations. To draw a human analogy it would be like having Leonardo da Vinci, Isaac Newton, and Albert Einstein on the team versus having only one of them in a given team. To compare the multi-hypotheses Genetic Algorithm (GA_MH) and basic single-hypothesis Genetic Algorithm (GA_SH) ROC curves and Recall-Precision curves are used. The ROC curves display the False Positive Rate (FPR) verses True Positive Rate (TPR) and the Recall-Precision curve displays the Recall versus the Precision.

The performance of the Genetic Algorithm can be evaluated based on the values of the parameters such as *Fitness threshold, p:* population, *r:* fraction of replaced by crossover, *m:* mutation rate, *n:* number of hypotheses. Nevertheless a complete evaluation of the dependence on the parameters was not carried out due to time constraints. The following (arbitrary) settings were used for the experiments.

*Fitness_threshold:*45/50/ 55
p: 70
r: 0.5
*m:*0.1
n: 5

A tuning set which is 20 percent of the training set was randomly selected to evaluate the *fitness* of the candidate hypotheses.

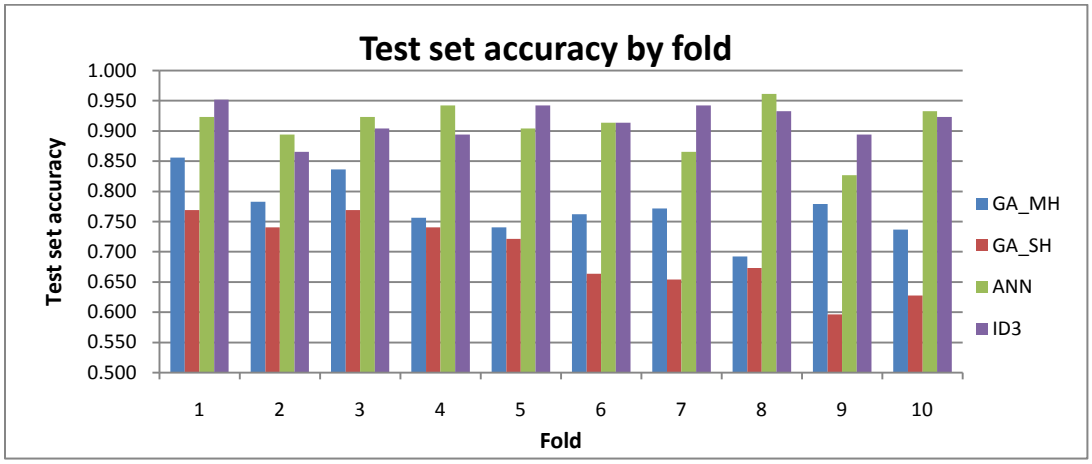
Results

The First Table shows the accuracies of the ten test folds from the Cars dataset. The comparison includes Genetic Algorithm with multi-hypotheses (GA_MH), Genetic Algorithm with single-hypothesis (GA_SH) , Artificial Neural Net-Perceptron (ANN) and Decision Tree (ID3).

Fold#	GA_MH	GA_SH	ANN	ID3
1	0.8557692308	0.7692307692	0.9230769230	0.9519230769
2	0.7826923077	0.7403846154	0.8942307690	0.8653846154
3	0.8365384615	0.7692307692	0.9230769230	0.9038461538
4	0.7563972940	0.7403846154	0.9423076920	0.8942307692
5	0.7403846154	0.7211538462	0.9038461540	0.9423076923
6	0.7619230769	0.6634615385	0.9134615380	0.9134615385
7	0.7715384615	0.6538461538	0.8653846150	0.9423076923
8	0.6923076923	0.6730769231	0.9615384620	0.9326923077
9	0.7788461538	0.7561538462	0.8269230770	0.8942307692
10	0.7369230549	0.6278264033	0.9326923080	0.9230769231
mean	0.7713320349	0.6954749480	0.9086538461	0.9163461538
st. Dev	0.0474519885	0.0608034154	0.0390578770	0.0272152945
p-Value(MH)		0.0015873178	0.0001325595	0.0000177112
p-Value(SH)	0.0015873178		0.0000006564	0.0000026870

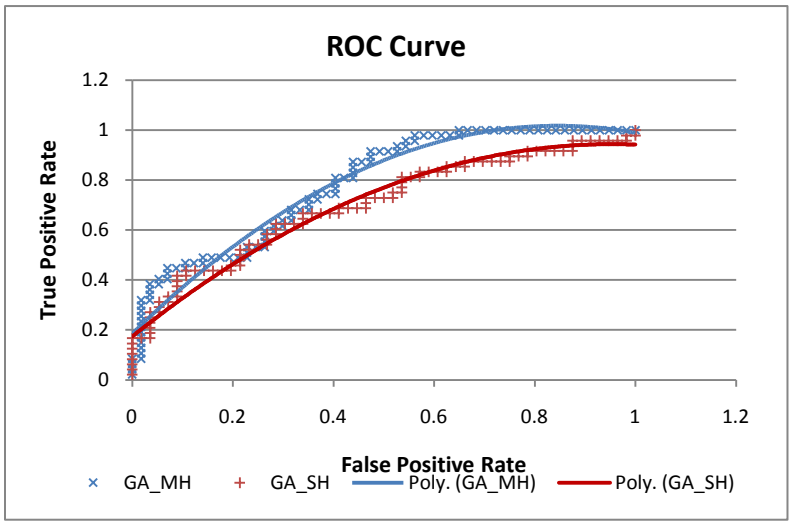
The Last two rows show the two tailed p-values of Multi Hypotheses and Single Hypothesis Genetic algorithms in comparison to each other as well as the Perceptron and the Decision Tree.

The Following bar chart shows the test set accuracies by the fold for algorithms mentioned above.



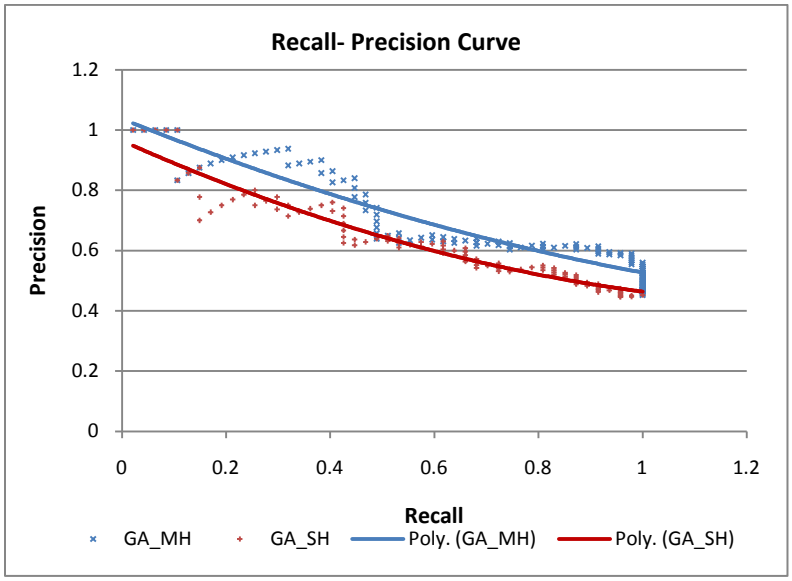
It is evident from the p-values and the bar chart that the Multi Hypotheses GA is statistically significantly better than the Single Hypothesis GA. It is also evident that the Perceptron and the Decision Tree are both statistically significantly better than the Multi Hypotheses GA and the Single Hypothesis GA.

The ROC curve given below shows the FPR versus TPR of the Multi Hypotheses GA and the Single Hypothesis GA.



It can be observed from the ROC curve that the Multi Hypotheses GA has higher TPR and lower FPR than the Single Hypothesis GA.

The Recall - Precision curve below shows the Recall versus Precision of the Multi Hypotheses GA and the Single Hypothesis GA.

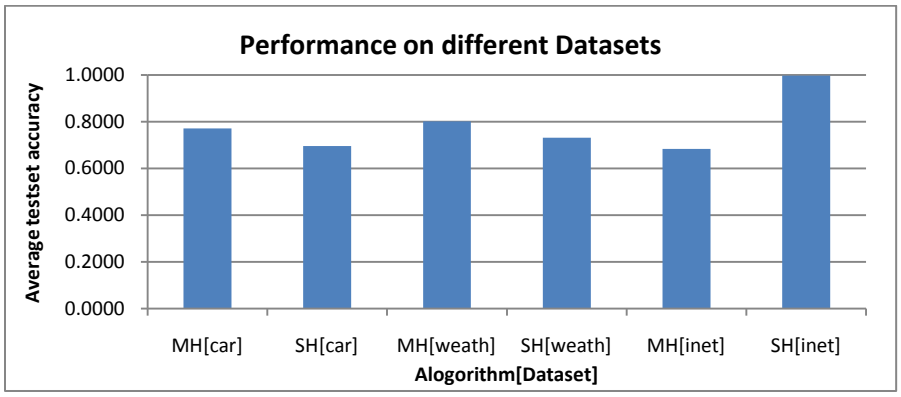


It can be observed from the Recall - Precision curve that the Multi Hypotheses GA has higher Precision and less recall than the Single Hypothesis GA.

The Next Table shows the test set accuracies of the Multi Hypotheses GA and the Single Hypothesis GA on different data sets. The cars data set contains car models and the classification task is the fuel economy, the weather dataset contains atmospheric data and the classification task is the temperature, and the internet dataset contains demographics of people and the classification task is internet usage.

	Cars		Weather		Internet	
Fold#	GA_MH	GA_SH	GA_MH	GA_SH	GA_MH	GA_SH
1	0.8558	0.7692	0.8000	0.6900	0.7402	1.0000
2	0.7827	0.7404	0.7200	0.6500	0.4724	1.0000
3	0.8365	0.7692	0.7400	0.7500	0.7953	0.9764
4	0.7564	0.7404	0.8200	0.7700	0.7559	1.0000
5	0.7404	0.7212	0.7700	0.7600	0.4236	0.9921
6	0.7619	0.6635	0.7600	0.8500	0.8031	1.0000
7	0.7715	0.6538	0.8200	0.6400	0.8504	1.0000
8	0.6923	0.6731	0.8900	0.7700	0.4488	1.0000
9	0.7788	0.5962	0.8300	0.6700	0.7638	1.0000
10	0.7369	0.6278	0.8700	0.7700	0.7874	1.0000
mean	0.7713	0.6955	0.8020	0.7320	0.6841	0.9969
st Dev	0.0475	0.0608	0.0549	0.0668	0.1658	0.0076

The bar chart below show the averages accuracies from the different data sets



It can be observed from the bar chart and the table that the Single Hypothesis GA has almost 100 percent accuracy on the internet dataset. The Multi Hypotheses GA has better average accuracy on the on the cars data set and the weather data set.

Discussion

The results of the paired t-test, ROC curve and the Recall - Precision curve show that the Multi Hypotheses GA performs better than the Single Hypothesis GA. This validates our initial assumption that a more general model learned by multiple hypotheses yields better results than learning a single hypothesis.

It is also evident that the Perceptron and the Decision Tree algorithms performed better than both of the GA variations. This could be due to lowering the fitness threshold in order to reduce the running time of the GA. Because of the random nature of the GA it is naturally expected to take a longer time to find good hypotheses; so by attempting to reduce the running time we also reduce the effectiveness of the GA.

The results from applying the GA to different data set show that the Single Hypothesis GA clearly archives near perfect accuracy on the internet dataset this could be because the data set has one hypothesis that classifies the dataset perfectly and the Single Hypothesis GA was able to find it. The above dataset can be considered an anomaly. The Multi Hypotheses GA has better average accuracy on the on the cars data set and the weather data set.

RELATED WORK

The GABIL system described by DeJong et al. (1993) uses a GA to learn boolean concepts represented by a disjunctive set of propositional rules which is very similar to the Single Hypothesis GA implemented in this paper. The Multi Hypotheses GA builds up on this idea by saving the best hypotheses from all the generations resulting in better performance as observed from the comparison results.

FUTURE WORK

The major shortcoming of the GA is that it performed poorly compared to the other Machine Learning algorithms such as Perceptrons and Decision Trees. It is reasonable to believe that the lower accuracy of the GA is due to constraining the running time and not tuning the algorithm parameters properly. So it would be worthwhile to revisit the GAs with more time at hand to allow the GA sufficient time and well tuned parameters in the future to re-evaluate the performance.

CONCLUSION

We can conclude that the Multi Hypotheses GA performs better than the Single Hypothesis GA, so it is encouraging to continue working on Multi Hypotheses GAs. From the performance on different datasets we can conclude that GAs performs generally well on other datasets. It is also clear that GAs need sufficient time to provide good results and further testing is necessary to verify whether the GAs can perform as well as the other Machine Learning algorithms.

REFERENCES

DeJong, K. A., Spears, W. M., & Gordon, D. F. (1993). *Using genetic algorithms for concept learning.* Washington: Naval Research Laboratory.

Goldberg, G. (1989). *Genetic algorithms in search, optimization, and machine learning.* MA: Addison-Wesley.

Mitchell, T. M. (1997). *Machine Learning.* Portland: McGraw-Hill.