Quality versus Quantity of Rules in a Classifier Jury

[Extended Abstract] *

Jeffrey Horn
jhorn@nmu.edu

Matthew J. Holliday
matthewjacobholliday@gmail.com

Dylan J. Elliott
delliott@nmu.edu

Northern Michigan University
Marquette, MI 49855 USA

ABSTRACT

We show that under certain general circumstances there exists a choice of classifier rule length versus number of classifier rules, when given a fixed length classifier system, that maximizes performance of the system.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*

Keywords

evolutionary computation, learning classifier system, Pitt-style, binary classification, time series prediction, jury vote

1. INTRODUCTION

Given a fixed memory budget with which to express a set of classifier rules (i.e., the product of rule length and the number of rules), is it better to have a few long rules or many short rules? Is there an optimal tradeoff between rule length and number of rules? We begin to explore this issue for a Pitt-style learning classifier system (LCS) [2] through experiments on a particular binary time series prediction (BSP) problem. We encode j classifiers (a.k.a., rules) each of length k characters on a chromosome of length $\ell = j * k$, and evolve a population of n such chromosomes on a BSP problem [1]. We conduct a number of experiments varying j,k,ℓ, and n. We find that for our particular LCS algorithm, parameter settings, and BSP problem, the prediction accuracy of our evolved classifier sets does indeed *peak* at an intermediate setting of rule length k. While these results are empirical, and limited to a single instance of BSP and to a specific type of Pitt-style LCS (e.g., employing jury-style voting for rule interactions), we suggest that the bell-shaped performance curve that emerges from six different sets of experiments is general to many LCS systems on many classification problem domains. Given the computational constraints on evolving large classifier systems for real-world problems, such performance curves could be of practical importance.

*A full version of this paper will be available at `http://cs.nmu.edu/~jeffhorn`

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. BACKGROUND

2.1 Binary Time Series Prediction

We use the 10000-bit training set for the Binary Series Prediction (BSP) contests [1] organized by Dan Ashlock for the 2006 WCCI and CEC competitions. The set contains 5106 ones distributed among 4894 zeros by some unknown function. The challenge of a BSP is to predict the next bit in the series given all of the previous bits. A candidate *predictor*, such as a classifier rule set, is evaluated on the entire series by being asked to predict the next bit before being shown the next bit. Thus the minimum score is zero while the maximum score is 10000 correct predictions.

2.2 The LCS: Pitt-style with Jury Voting

Our predictor is a set of rules, each consisting of string of k characters from the set $\{0, 1, \#\}$, called a *rule*. We encode a set of j such rules on a chromosome of length ℓ by simply concatenating them, with $\ell = j * k$. (Note that we restrict our algorithm to handle only complete rules by restricting ℓ to be a multiple of k.) Encoding an entire rule set on a single chromosome is known as the*Pitt-style* LCS [2].

To use the encoded rule set to predict the next bit in the BSP sequence we match each of the j rules against the last k bits seen in the sequence, with '1' matching '1', '0' matching '0', and '#' (a *don't care*) matching both '1' and '0'. If a rule completely matches the last k bits (with no conflicts in all k bit positions), then the rule is said to *fire* which means it "predicts" that the next bit will be a '1'. If there is a conflict in any of the k bit positions then there is no match and the rule is considered to predict a '0' next.

The individual rule predictions can be combined to make a group prediction for the entire rule set in a number of reasonable ways, such as majority voting. We choose a simple system we call *jury voting* whereby if ANY of the rules predicts a '1' then the rule set predicts a '1'; otherwise the set predicts a '0'. Thus it only takes a single rule match to produce a '1' prediction. Put another way, a prediction of '0' requires a unanimous vote (that is, no rule matches the k bits). This is similar to the way a jury functions in that a result of "guilty" requires unanimity whereas a result of "not guilty" requires only a single dissent.

We apply a simple genetic algorithm (sGA) [2] to a population of n chromosomes, generated initially at random. Each generation we apply binary tournament selection using as fitness the number of bits correctly predicted by the chromosomes' decoded rule sets, followed by two-point crossover to create a new population for evaluation.

Table 1: Average (10 runs) Number of Bits Correctly Predicted

Chrom. Length (popsize)	Rule Length										
	1	2	3	4	6	8	12	16	24	32	64
24 (1000)	5106	5105	6070	7406	7370	7139	6909	-	6224	-	-
24 (2000)	5106	5105	6781	7406	7601	7148	6917	-	6224	-	-
32 (1000)	5106	5105	-	7406	-	7545	-	6787	-	6026	-
32 (2000)	5106	5105	-	7406	-	7600	-	6840	-	6221	-
64 (1000)	5106	5105	-	7015	-	8132	-	7477	-	6462	4862
64 (2000)	5106	5105	-	7120	-	8179	-	7535	-	6788	4862

3. EXPERIMENTS

3.1 Setup

For all of the runs certain parameters of evolution are fixed: number of generations is 200, mutation rate is set to zero (no mutation), crossover probability is set to 0.9, tournament size is always two, and the *don't care* probability (i.e., the probability of generating a "#" character independently for each position of each chromosome in the initial generation) is 0.5. We use two-point crossover (with crossing points selected uniformly at random from all ℓ inter-loci points), and generational replacement (i.e., replacing an entire population with its offspring each generation).

We conduct runs for three different chromosome lengths: $\ell = 24$, 32, and 64. For each chromosome length we try two population sizes: $n = 1000$ and 2000. For each combination of ℓ and n we average the results of ten runs, each using a different random number seed and hence a different sequence of random numbers. Note that the same ten random seeds were used for each combination of ℓ, n.

3.2 Results

Table 1 shows the average performance of the six experiments (that is, three chromosome lengths at two different population sizes each) in terms of the number of bits (out of 10000 total) in the BSP data correctly predicted by the best individual in the final generation (number 200), averaged over ten different (unique random seed) runs. Figure 1 plots these mean performances as a function of rule length and chromosome length, with lines connecting the results for each series corresponding to a single combination of ℓ and n. The shaded regions show the difference in average performance for two different population sizes (1000 and 2000). In our runs, increasing population size always results in improved or identical performance.

3.3 Analysis

As intuition might suggest, we do see generally higher performance with increased population size and increased chromosome length. Perhaps less intuitive is the apparent peak in performance at an intermediate rule size, when varying the way a fixed chromosome length is divided up into equally sized rules. As can be seen in each of the series of runs for each fixed chromosome length, having six to eight rules of six to eight characters each seems to be better than having more and smaller rules or fewer but larger rules. This empirical result holds across three different chromosome lengths and two different population sizes. These data point to a balance that needs to be found (at least for this algorithm on this problem), a balance between investing expressive power in a few sophisticated individuals versus distributing such processing power out among many simpler agents. To explore just how general this balance might be, we plan to take this approach on other classification problems (e.g., character recognition, feature detection), and with other types of conflict resolution (e.g., majority voting instead of jury voting). We would also like to encode the rule length on the chromosome to see if evolution can find some kind of optimal rule length on its own.

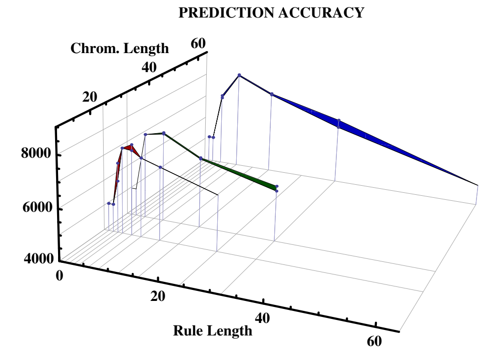

Figure 1: At all three chromosome lengths, the performance data indicate that an intermediate tradeoff of rule size to number of rules is best.

4. ADDITIONAL AUTHORS

Additional authors are the following alumni of Dr. Horn's fall 2012 class at NMU, CS 470 *Artificial Intelligence*: Joshua A. Chomicki (email: `jchomick@nmu.edu`), David A. Pfeiffer (email: `DavidPfeiffer54@gmail.com`), Steven M. Scheel (email: `sscheel@nmu.edu`), Lewis D. Steiner (email: `lsteiner@nmu.edu`), Erik P. Wisuri (email: `ewisuri@nmu.edu`), James M. Zeits (email: `jzeits@nmu.edu`), Jordan M. Bal (email: `jobal@nmu.edu`), Chelsea G. Burton (email: `chburton@nmu.edu`), Micah A. Erickson (email: `micerick@nmu.edu`), and Nicholas D. McIntyre-Wyma (email: `nmcintyr@nmu.edu`).

5. REFERENCES

[1] D. Ashlock. Binary Series Prediction Contest, In *WCCI 2006* and *CEC 2006* competitions, http://eldar.mathstat.uoguelph.ca/dashlock/CEC05/BSP.html, 2006.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, New York, 1989.