In Proceedings of the International Conference on Cognitive Modeling, 2004. 308-313
Mahwah, NJ: Lawrence Erlbaum.

Using a Genetic Algorithm to Optimize the Fit of Cognitive Models

Kevin Tor (tor@cse.psu.edu) and Frank E. Ritter (fer2@psu.edu)
School of Information Sciences and Technology and Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16801

Abstract

This paper analyzes the use of a genetic algorithm (GA) for auto-fitting a non-linear, multivariable model to data. The example model and task used to illustrate the effectiveness of a GA is the Tower of Nottingham task. Empirical data collected from adults completing the task were used to create an initial model. This model's variables were adjusted with a GA to simulate the performance of a child completing the task. The optimized fit was better than the previous fit generated by hand. We note the steps required to duplicate the process with other models.

Introduction

Achieving the perfect picture on a television can be a painstaking problem. Settings for hue, color, and other variables interact. As the people on the screen turn red, then purple, the user frenetically switches back and forth from tint to brightness making slight adjustments.

Adjusting the fit of a cognitive model to data, like adjusting a television, can depend on many parameters. Changing only one may not be enough to achieve the desired outcome. Hand optimization of a model can take an extraordinary amount of time in situations involving the adjustment of multiple parameters to understand how they influence problem solving. This occurs, for example, when modifying parameters to represent development or the effect of behavior moderators on cognition.

Complications also come from the size of the search space. The search space for a model of multiple parameters grows exponentially with the number of parameters. Quick solutions for these problems, which are attained by hand, can be simply the result of luck and cannot be relied on for consistency. For this reason, it is better to pursue an option involving automatic search (Ritter, 1991). These problems appear to have local minima and certainly have noisy evaluation functions, so we chose to explore the use of Genetic Algorithms (Davis & Ritter, 1987). Other approaches should also be tried, such as gradient descent.

Overview of Example Problem

To provide a basis for testing the effectiveness of a GA in optimizing the fitting of a model, an example problem must be used. Here, we use a cognitive model simulating a child putting together a puzzle. Empirical data was taken from both children and adults who constructed the puzzle. This data was used as a comparison for the model simulation.

Puzzle Construction. The puzzle is a pyramid consisting of five levels and a top. It is known as the Tower of Nottingham puzzle or, simply, the Tower Task (Wood & Middleton, 1975). The puzzle is constructed one level at a time and each level is made up of four pieces. The bottom level is constructed first followed by the next largest. Typically, two levels are not constructed simultaneously. Thus, the problem solver is concerned with finding and manipulating at most four pieces of the puzzle at a time.

Model construction and fit to the data. Adults provided the initial empirical data for the task. A model was created and fit to the adult's data by adjusting a set of parameters used by various prominent developmental psychologists in their theories of 'what develops' (Jones, 1998; Jones, Ritter, & Wood, 2000). These parameters range from memory capacity to understanding whether the puzzle's edges are flush. Adults have an accepted set of values for the parameters and a model was created to fit the adult data. Therefore, using this model and data collected from five seven-year-old children, the parameters were adjusted to fit the behavior of children. The problems lay in determining which parameters need adjustments, and what the appropriate ranges are for these parameters, and how much of an adjustment they need.

Use of the genetic algorithm. Once the desired parameters have been chosen and a set of ranges determined, the GA can commence fitting the model. In this case, a randomly generated bit string represents an individual or child. In the GA metaphor, this is a genotype. The bit string is parsed in a programmer-defined way to become the parameters of the genotype's runs. (The model that realizes the parameters is the phenotype). Here, a gene might represent processing speed and another whether block edges were checked.

After each genotype completes a series of constructions (the number of times the genotype puts the puzzle together), a fitness function provides a value based on a statistical measure of performance over those runs for how close the model's performance fits the data. This fitness measure is then multiplied in our GA by the fraction of successful constructions. In some cases, a phenotype does not complete the puzzle or does not complete it properly, resulting in a penalty on average performance.

This process occurs for all phenotypes in the population. Once the entire population has completed its construction sets, a best phenotype is chosen. If the model deems the best phenotype close enough to the children's performance, the GA terminates and the model is considered fit. Otherwise, a new generation is created by selecting some genotypes (bit strings) from the phenotypes (successful models) that performed well and creating new genotypes through combination and mutation.

Preview of Results. The GA finds a good fit to the data. With this particular GA, a run was conducted of 10 genotypes per population each constructing the model three times. In the first full run, the model ran for 77 generations. Possible solutions based on this population set were found (Tor, 2004). Further runs with larger populations and more constructions per genotype will be needed to achieve complete optimization of the model.

Example Problem

To illustrate the application of GA to this problem, an example model will be optimized. Jones (1998) created a model of child development with multiple parameters. This is an ideal example for running our experiment. The values for the parameters can be incorporated in a GA. Due to the number of parameters being monitored, a substantial number of generations may be needed, showing the effectiveness of the GA in a large search space.

Developmental Theories Examined

Psychologists argue over which changes in which mechanisms explain the changes in thought. There are several theories about what mechanisms change. We worked with a subsample of the set of theories Jones (1998) used. Table 1 shows how the theories were implemented in the model, including the ranges for each parameter. This set of ranges was checked with Lebiere (2003).

Pascual-Leone's (1987) M-Power theory explains that what develops is the number of working memory elements that could be managed at one time. His theory states that as children grow, their memory capacity increases allowing them to have access to more information at a given moment.

Siegler believes that development stems from strategy choice. Children have less ability in choosing an appropriate strategy than adults (Siegler & Shipley, 1995). There are many ways to make strategy choices less rational: increasing noise to decisions, decreasing the, or decreasing the accuracy in checking for obstructions of fit.

Kail (1996) argues that processing speed in humans increases with age. Essentially, a child takes longer to perform a cognitive task than an adult.

Piaget (1952) had a stage theory about how humans developed. Each stage represents a step in complexity of thinking that can be performed and thus understanding the environment around them. Children represented by this model are 7 years-old and, according to Piaget, are at the end of the pre-operational stage and about to enter the concrete operational stage. This means that they have trouble hypothesizing mental manipulations to objects found in the outside world. In this case, the objects would be blocks or block constructions, and it has been operationalized in the model by modifying the accuracy of checking flush edges in the puzzle.

Source of Data

The empirical data was taken from work done in Jones' (1999) thesis. For the adults, he filmed five undergraduate students completing the Tower task. The students had no a priori knowledge of the task and were given five attempts at the puzzle. Throughout the attempts the students were asked to give verbal accounts of their behavior and choices. The exhibited behavior was noted and subsequently coded. This became the basis for the original model.

The childrens' data was taken from observations on videotapes of children performing the task (Reichgelt, et al., 1993). Jones analyzed the children's videotapes the same way the adult data was analyzed.

Tower of Nottingham Model

Jones wrote the model for the Tower of Nottingham Task using the ACT-R 3 cognitive architecture (Anderson, 1993). Jones included a graphical representation of the Tower using Garnet (Myers, et. al., 1990), a toolkit for building user interfaces in Common Lisp.

Jones looked at the developmental theories and wrote a separate model to incorporate each one. He ran each model and adjusted their parameters by hand. Although each model provided favorable results, Jones' best results came from the strategy choice model.

Cornwell (2001) in his thesis furthered the exploration of what develops. He took Jones' models and created a single model combining the theories. This model had multiple parameters to adjust. Hand adjustment was too difficult so a GA was added. Cornwell left the project with a usable model but the graphics implementation made running the GA quite slow. His GA runs were thus limited in number of genotypes and number of generations. We started our project based on his GA and optimized its performance.

The genetic algorithm used on this model is called GAL. It is open-source code downloadable from Carnegie-Mellon (Spears, 1991). The implementation of this particular GA maintains diversity amongst the population with low mutation rate, distinct genotypes across generations, and by limiting the spread of the best individuals. All that needs to be added to the GA is the fitness function, a mapping from binary genotypes to the model parameters, and the termination function, all of which we took from Cornwell (2001).

Optimization of Model Runs

The initial system (Cornwell, 2001) was created in a Macintosh environment and was nearly complete. We ported the program to Red Hat Linux, putting it in a Unix environment. Because Unix often offers access to faster machines and offers a more advanced operating system for testing GA implementations, this port was essential.

Table 1: The incorporation of the developmental theories in the ToN model and GA.

Theorist	Theory as it Relates to ToN Model	Theory Implementation in ToN Model	GA Implementation (genes)
Piaget	7-year-olds fall at the end of the pre-operational stage—they may not be able to manipulate internal representations of objects and events in the external world.	Conditionally removed mental check to determine if two blocks will have flush outer edges when fit together	Conditionally loaded different production rules for mental fitting operations.
Siegler	Children should be slower and less accurate at both choosing and implementing an effective strategy than adults.	Added noise to rule firings with ACT-R's expected gain noise parameter(:egn)	:egn parameter set to integer value between 2 and 9
		Varied accuracy of check to determine if the outer edges of two blocks would be flush if fit together	probability of correctly checking for flush edges set between 1 and 0.5 (chance)
		Varied accuracy of check to determine if obtruding features would obstruct the fit of two blocks	probability of correctly checking for obstructing features set between 1 and 0.5 (chance)
Pascual-Leone	The number of knowledge structures that can be simultaneously held in memory increases with age.	Limited the number of DMEs that could be active at once with ACT-R's retrieval threshold parameter (:rt)	:rt parameter set to integer value between 0 and 7 [between -2 and 7]
Kail	Processing speed increases with development	Varied the default time allotted to rule firings with the ACT-R's default action time parameter (:dat)	:dat parameter set between 65 and 100 ms [between 40 and 100 ms]
		Varied the time allotted to hand movements	:effort parameter for productions involving hand movements set between 700 and 1,050 ms
		Varied the time allotted to eye movements	:effort parameter for productions involving eye movement speed set between 65 and 100 ms

A fitness function for the genetic algorithm focuses on 24 measures of comparison between the 7 year-old empirical data and the simulated data as determined by Jones (1998). These measures include layer completion times, total completion time, number of construction attempts for a layer, number of errors, and number of blocks examined.

Statistical t-tests are used to measure differences between the real and simulated data. The sum of the 24 t-tests is the major component of the fitness function. The sum is inverted to create a value that increases as the sum of the values of the t-tests decreases. That is, as fit increases, the fitness increases. (This is an example fitness function. It should not be inferred that we wish to accept the null hypothesis this way, merely that the numbers provide a useful and intuitive measure of data fit. Many others, quite possibly better ones, are possible.) Finally, a penalty is assessed by a successful construction ratio consisting of the number of successful constructions over the number of total constructions. A successful construction consists of the complete construction of the puzzle without skipping a level or constructing a level incorrectly. Therefore, the number of successful constructions over the genotype's total constructions is multiplied against the fitness value.

There were a few problems. The state of the model left by Cornwell appeared to have a memory leak. Jones' model was written to run for one construction and was not scalable to accommodate multiple runs. For the purposes of the GA, the model needed to run continually. Attempts to reset the graphics for the next puzzle construction were unsuccessful.

To alleviate these problems, many options were explored. Ultimately, we spawned Unix processes that each executed a model constructing the puzzle. Once a set number of constructions are completed, a fitness value is attained for that genotype and the next genotype begins. Using system processes works with a model that was originally written to run once.

Results

To illustrate how the GA works, a sample, small run is explained. The following is the preliminary set of results for the first full run, consisting of 77 generations of ten genotypes, each evaluated three times. Each evaluation takes about 5 minutes on standard PC, so this represents about 8 days of processing.

The results presented are the best three genotypes found during the first full 77 x 10 x 3 run. These genotypes are then rerun more times to validate the soundness of the achieved fitness values. Then, they are compared to the previous best of Jones.

The model was started with a randomly-generated, initial population of 10 genotypes, each made up of 24-bit strings as specified by the GA. The bit string is parsed and converted into the parameters that are used in the puzzle construction simulation. At this point, the GA runs the resulting model and collects the results of the fitness function for each of the genotypes. It takes the results and manipulates the better genotypes to create the next generation. We also saved the best genotypes.

The best three genotypes found are displayed in Table 2, including their fitness and t-test values. There is some commonality between the three in terms of expected gain noise and the motor and perceptual speeds. The fitness value and the first t-test sum were attained from the initial GA run. The t-test sum is the sum of the t-test values for the 24 measures. The order of the genotypes in the chart corresponds to the order that they were generated.

As can be seen, the GA was finding higher fitness values with time. Genotype 3 performed better than Genotypes 1 or 2 but did not check for flush edges during construction. It also had a longer processing time that the Adult meaning that the child model takes more to apply one of its rules.

Logically, Genotype 3 would not appear to be a good set of parameters. It does not check for flush edges, causing the puzzle to create more incorrect constructions. Also, its retrieval threshold is extremely high, allowing very few pieces of knowledge to appear in memory. Finally, it is drastically different from Genotypes 1 and 2. It may represent a local minimum and provide evidence that a GA is an appropriate optimization technique.

For further analysis these three genotypes were run 10 more times. Genotypes were run until 10 perfect construction sets of 3 runs each were completed. Construction sets where any of the three constructions failed were excluded with no penalty. The 10 fitness values were averaged to get the average t-test sum shown in the table. The max t-test measure, shown in Table 2, is the measure of the genotype that performed worst. As a group, the genotypes appear to perform poorly on timing measures.

The average t-test analysis showed that Genotypes 1 and 2 were initially performing correctly as they did not vary much from the initial t-test sum. They did experience regression to the mean as the average t-test sum did increase slightly over the initial t-test score. However, Genotype 3 had a unique, initial performance for those parameters leading to the belief that three constructions per genotype may not be enough to attain stable measures. In order for the GA to improve the genotypes, it would need to run with more constructions and more genotypes per generation.

The GA's performance should be considered successful based on these preliminary results. The initial fitness values of the three best genotypes did increase as the generations progressed. A termination function, created as specified by the GA, was not used in this run. This was done purposely to see how close a fit could be reached

Comparison to Jones' Results

The results attained in this study should be compared against other data previously found in this area, that is, Jones' hand optimization models. The easy method is to take the best model from his research and compare it to the three best genotypes discussed earlier. Jones found that the best performance came in the implementation of the Siegler model (Jones et al., 2000). This model affects the strategy choice by adding noise to the decision process. The model with these parameters was run until 10 successful puzzle construction sets were found in the same manner as with the three best genotypes. This is illustrated in Table 2

The average t-test sum for the Siegler model turned out to perform better than two of the top-performing genotypes. However, Table 2 shows that Jones' Siegler model took 67 attempts to get 10 puzzle construction sets that finished. This shows that his Siegler parameter set often cannot solve the Tower task. In fact, the parameters find the solution only about 1 in 7 times whereas the GA-computed parameters solve the task at a rate of about 1 in 1.5 to 2. The problem is that there was no penalty assessed for poor performance in Jones' analysis. Forcing the Siegler model to finish 10 perfect sets did give a good average t-test sum but may not be a good measure of its fit to the children's data. It also suggests that we need to go back and examine how often Wood's children completed the task on their own. As this task was designed to study contingent tutoring, the model predicts that many children cannot solve the task without assistance.

These results take a small step towards answering the question of what develops. The results in Table 2 suggest that multiple aspects develop. The best fitting model modified the children's model in several dimensions, providing support for several theories. The fit (and direction of parameter modifications) is not clear enough and good enough to suggest that we have a complete understanding of this task and data. There remains what appears to be some noise in the fit, and there may be multiple parameter sets that fit the data well. We will have to run the GA longer to get more stable and clear results.

Conclusions

This work shows that a genetic algorithm can be used to optimize models' fit to data where the fit consists of a large search space due to multiple model parameters. Hopefully, genetic algorithms will be used to optimize other complex, nonlinear models. The following is a discussion of possible improvements to this approach and how implementing a GA to optimize other should be approached.

Table 2: Comparison of GA found models with Siegler Model as implemented by Jones.

	Genotype 1	Genotype 2	Genotype 3	Siegler (Jones)	Adult
Check flush edges	T	T	Nil	T	T
Expected gain noise	2	2	2	6	0.4
Flush edge accuracy (%)	90	95	100	100	100
Obstruction detection accuracy (%)	100	100	75	100	100
Retrieval Threshold	3	3	6	0	0
Processing Speed (s)	0.065	0.065	0.08	0.05	0.05
Eye Speed (s)	0.095	0.095	0.085	0.2	0.2
Hand Speed (s)	0.95	0.95	0.8	0.55	0.55
Average fitness (x 10 sets)	46.7	47.4	55.1	46.9	
Number of runs to get 10 correct	16	19	18	67	10
Maximum t-test measure	3.45 (Size 4 Layer Time)	3.31 (Time Taken To Complete)	6.28 (Time Taken To Complete)	2.66 (# of Incorrect Disassembled)	

Improvements

The GA needs some measure to compare its performance against. This measure comes from the empirical data set. Given this statistical measure for comparison, the GA can optimize the model. Next is the idea of how close a fit is sufficient. Determining a measure of an appropriate fit is arguably the most complicated part of the GA application.

In this example, the GA did not perform as well as we would have liked; the largest problem appears to be a matter of having enough puzzle constructions per genotype to get stable evaluations. Three constructions may not be enough to get acceptable results where 5, 10, or even 20 may. Running the model on a larger scale will cause the t-tests to give results closer to the averages seen in Table 2 and remove the possibility of the unique performance seen in Genotype 3, where a higher fitness value was achieved initially but the average was significantly more.

Another plan is to adjust the GA to keep the best performing genotypes in the current population. the GA we used prefers diversity and uses the best performing genotypes to create the next generation but they, themselves, may not appear in the next generation. This would allow the fitness value for a particular genotype to be updated keeping performance realistic.

The successful construction ratio used to penalize a poor run's fitness can be squared to increase the penalty. The bit string could be lengthened to 48 bits providing more granularity for the parameters, creating more options in the range.

Applying the Optimization to Other Models

Given a model that needs to be fit and empirical data to fit the model to, optimization can be accomplished by following a set of guidelines. The model should have multiple parameters needing adjustment and fine granularity in the range of acceptable values for those parameters. This provides a large and expensive search space that makes a GA a useful choice.

The genotypes need a way to be represented in the GA. In this project bit strings were used because the GA operated on bit strings. Arrays of the parameter values can also be used depending on the GA implementation. Then, simple functions to parse the strings or arrays can be written to incorporate the values into the model. Theoretically, these representations are equivalent; the bit string is easier for the GA to manipulate, and the symbolic representations are easier for modellers.

A set of measures need to be created for use in the comparison of the simulation run data and the empirical data. T-tests or other statistical measures, such as correlations, should be used to provide a fitness value for the model's correspondence to the data.

Graphics are a nice addition to a running simulation model but can be problematic when optimizing fits. A decision should be made as to whether the model being optimized needs graphics. If graphics are added, they should be as independent of the model as possible and capable of being shut off. Graphics slow models down.

GAs need to run for many generations, which can take time. The run of a genotype's solution should take the smallest amount of time possible. This will help minimize the overall run time of the GA and allow the search space to be better explored. A summary of these lessons are shown in Table 3.

Table 3: List of Using a GA with another Model.

1. Multiple parameter model
2. Empirical data and measures of comparison
3. Candidate strings and parsing function
4. Fitness function that increases as fit improves
5. Graphics that can be "reset" by model
6. Minimized genotype run time
7. Loads of time

Further Work

Using this methodology to fully understand what develops in children will not be complete until a larger population of models has been examined over further generations. We have arranged supercomputer support from the U.S. Army Research Laboratory in Maryland to perform this analysis. The processing power found at their facilities will allow the runs to be completed on the order of 100x faster.

Acknowledgments

The previous work by Gary Jones and Jason Cornwell provided the foundation for this project. Thanks to the ACT-R community and Christian Lebiere for providing their input. Ian Schenck and three anonymous reviewers provided useful comments.

References

Anderson, J.R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum.

Cornwell, J. B. (2001). *Using genetic algorithms to create and optimize models of development.* Unpublished undergraduate honor's thesis, Pennsylvania State University, University Park, PA.

Davis, L. W., & Ritter, F. (1987). Schedule optimization with probabilistic search. *Proceedings of the Third Conference on Artificial Intelligence Applications*. IEEE Computer Society. 231-236.

Jones, G. (1998). *Testing mechanisms of development within a computational framework.* Unpublished doctoral dissertation, University of Nottingham, Nottingham, England.

Jones, G., Ritter, F. E., and Wood, D.J. (2000). Using a cognitive architecture to examine what develops. *Psychological Science, 11*, 1-8.

Kail, R. (1996). Nature and consequences of developmental change in speed of processing. *Swiss Journal of Psychology*, 55, 133-138

Myers, B. A., Guise, D. A., Dannenberg, R. B., Vander Zanden, V., Kosbie, D. S., Pervin, E., Mickish, A., & Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer*, *23*, 71-85.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Pascual-Leone, J. (1987). Organismic processes for neo-Piagetian theories: A dialectical causal account of cognitive development. In A. Demetriou (Ed.). *The neo-Piagetian theories of cognitive development: Towards an integration.* Amsterdam: North-Holland.

Piaget, J. (1952). *The origins of intelligence in children.* New York: International Universities Press.

Reichgelt, H., Shadbolt, N., Paskiewicz, T., Wood, D. and Wood, H. (1993) EXPLAIN: On implementing more effective tutoring systems. In A. Sloman, D. Hogg, G. Humphreys, D. Partridge, & A. Ramsey (Eds.), *Prospects for artificial intelligence* (pp. 239-249). Amsterdam: IOS Press.

Ritter, F. E. (1991). Towards fair comparisons of connectionist algorithms through automatically generated parameter sets. In Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. 877-881. Hillsdale, NJ: Lawrence Erlbaum.

Siegler, R. S., & Shipley, C. (1995). Variation, selection and cognitive change. In T. Simon & G. S. Halford (Eds.). *Developing cognitive competence: New approaches to process modeling*, 31-76. Hillsdale, NJ: Lawrence Erlbaum.

Spears, W. M. (1991). [Computer Software]. GAL. U. S. Navy Center for Applied Research in Artificial Intelligence, USA.

Tor, K. (2004). *Using a genetic algorithm to optimize the fit of cognitive models*. Unpublished Master's thesis. Pennsylvania State University, University Park, PA.

Wood, D., & Middleton, D. (1975). A study of assisted problem solving. *British Journal of Psychology, 66*, 181-191.