Behavior Research Methods, Instruments, & Computers
2003, 35(2), 202-216

Using genetic programming to discover nonlinear variable interactions

CHRIS WESTBURY
University of Alberta, Edmonton, Alberta, Canada

LORI BUCHANAN
University of Windsor, Windsor, Ontario, Canada

and

MICHAEL SANDERSON, MIJKE RHEMTULLA, and LEAH PHILLIPS
University of Alberta, Edmonton, Alberta, Canada

Psychology has to deal with many interacting variables. The analyses usually used to uncover such relationships have many constraints that limit their utility. We briefly discuss these and describe recent work that uses genetic programming to evolve equations to combine variables in nonlinear ways in a number of different domains. We focus on four studies of interactions from lexical access experiments and psychometric problems. In all cases, genetic programming described nonlinear combinations of items in a manner that was subsequently independently verified. We discuss the general implications of genetic programming and related computational methods for multivariate problems in psychology.

Great progress has been made in decomposing psychological functions into their composite neurological and cognitive subfunctions. The methods of experimental psychology and clinical neuropsychology have succeeded in identifying a large and growing number of well-defined variables that play a functional role in cognition. Understanding how these variables may interact is one problem that poses great difficulty for psychology. In this paper, we introduce the use of genetic programming (GP) to evolve equations that can combine a large number of variables in order to predict a single measure of interest. This technique has wide applicability, but in this paper we focus on two domains: psycholinguistics (predicting reaction times [RTs] in lexical access) and psychometrics (predicting full-scale validation scores from subsets of test questions). GP has several advantages over more traditional techniques for studying the relation of many predictor variables to a single target variable. We highlight some of the more general methodological issues involved in using GP to compute estimator equations across many variables in any domain and, in the conclusion, address some criticisms that may be leveled against its use.

This work was made possible by a National Science and Engineering Research Council grant from the Government of Canada to the first author and a grant from the Social Sciences and Humanities Research Council (MCRI Program) to the second author. We thank Ray Gibbs, Guy Van Orden, Jonathan Vaughan, Richard Plant, and an anonymous reviewer for their helpful comments on earlier versions of the paper. Correspondence concerning this article should be addressed to C. Westbury, Department of Psychology, University of Alberta, P220 Biological Sciences Building, Edmonton, AB, T6G 2E5 Canada (e-mail: chrisw@ualberta.ca).

It is not our intention to suggest that GP is itself a model of psychological processes or that it may in itself stand as a theoretical entity. Our aims in this paper are methodological. We will demonstrate how GP may be used as a useful computational tool for discovering and describing complex relations in data that may otherwise be missed. We will begin by describing GP and comparing it with two other tools that have also been used to describe relations in data.

Three Traditional Approaches to Multivariate Analysis

Much of the progress that has been made in identifying variables relevant to psychological functioning has been conducted using factorial designs that simplify the problem under study by treating a small number of continuous variables of interest as factors. Other variables are usually either blocked to a narrow range or left unassessed, with the assumption being that they are randomly distributed.

The parametric statistics that are usually used to analyze factorially designed experiments make two assumptions. One assumption is that the factors under consideration are normally distributed. In cases in which this is known to be untrue and the nature of the departure from normalcy is understood, adjustments may be made. The second assumption is more important because it cannot always be easily tested or controlled. This is the assumption that the factors under examination are free to vary independently. Conclusions from experiments are compromised whenever one may have reason to doubt either the distributional or the independence assumption.

Another problem with drawing conclusions from factorial manipulations is that they are often limited in their ability to test the full range of a measure. They tend, rather, to test only effects drawn from the extremes. In cases in which the effect of an independent variable on a dependent variable is not linear, testing a narrow range of values must necessarily give an incomplete and even misleading understanding of the nature of the phenomenon under study.

A second technique for analyzing how multiple variables impact on a single variable is multiple regression analysis. This technique makes it possible for investigators to quantify the sensitivity of a target variable to a range of variables (e.g., Balota, Cortese, & Pilotti, 1999; Treiman, Mullennix, Bijeljac-Babic, & Richmond-Welty, 1995). Multiple regression analysis makes the assumption (not necessary in theory, but almost always assumed in practice) that any relationships that do exist between the variables are linear. Assuming linearity is usually necessary, since the alternative is underspecified. If variables are related in a nonlinear manner, the function that relates them needs to be specified before the regression analysis can be carried out. Unfortunately, there is usually no principled way to make such specification. Multiple regression also places restrictions on the number of independent variables that can be examined at one time.

A more recent approach to studying how many variables interact has been to develop connectionist models, which are capable of combining a large number of variables without making assumptions about linearity. These models can be tested against human performance, using the traditional analyses of multiple regression or factorial designs.

Neural networks produce nonlinear solutions to problems of variable combination that are often much better than the best linear solution. The problem with neural networks is that one is not sure what the solution is, because the model represents it in an opaque manner. One must expend a great deal of effort to re-represent it, analyzing the weights and, thereby, translating them into a human-comprehensible expression. In contrast, GP, as we shall see, provides an explicit equation that is already much closer to being humanly comprehensible.

A problem that is common to both regression models and some connectionist models, as well as to many other methods for predicting one value from a set of other variables, is that they suffer from a muddying of two sources of variation. One source of variation is that which is specific to the data set on which the model was built. Such variance is, by definition, an accident of the particular choice of stimuli that make up the set. As such, it would not be replicated in a different set. The other, more scientifically interesting, source of variation is that which is systematically related to the phenomenon being studied. This source of variance is, by definition, likely to be found in any data set, irrespective of its particular composition. Both connectionist and regression models can get around this by implementing cross-validation techniques, discussed below, but these are not always applied. Without their application, the structure of the problem allows either model to include the variance that is specific to that data set, with its idiosyncratic relations. As a result of being allowed to use this source of variation in their calculations these techniques are likely to systematically overestimate how much variation they have accounted for in the phenomenon that one wishes to explain. However, these techniques cannot offer any means of determining how much of the variation they have overestimated, since that is a function of each particular data set.

This problem is particularly vexing because systematic overestimation on a single data set must lead to systematic underestimation in generalizing to other data sets. Any estimator process that accounts for data set specific variance in its estimation will have that variance as an error term in a new data set, since, by definition, that data set specific variance does not apply to the new data set.

It is desirable to capture the nature of multiple-predictor interactions without being subject to the limitations discussed above. To do so, one needs a technique for combining variables that is free from limiting assumptions about the nature of the combinations and that produces algorithmically well-defined predictions that can be directly tested and measured on new data sets, thereby minimizing the error due to the inclusion of data-set–specific error. GP is a technique that can fulfill these desiderata.

Genetic Programming

GP (Koza, 1992) uses natural selection to evolve computational functions. The principle by which it works is simple. One begins with a problem that is defined in such a way as to allow for rank ordering of possible solutions. It is important to stress that this does not require that one know anything at all about the form or content of the desired solution. It only requires that one be able to algorithmically decide whether one solution is incrementally better than another. The goodness value assigned to any solution is called its *fitness value*. The function that assigns the fitness value to each proposed solution is known as the *fitness function.*

To begin the evolutionary process, many purely random mathematical guesses at the solution to the problem are generated through simple concatenation of legal operators and operands in a tree structure (see Figure 1). Any mathematical or logical function can be easily represented in this form. The randomly generated guesses are subsequently rank-ordered using the fitness function. The best among them are *mated* to produce the next generation of problem-solving functions. The mating procedure in GP is illustrated in Figure 1. It consists simply of random subtree swapping of the *parent* problem-solving functions. The random tree-swapping amounts to reusing sections of the calculations that have possible utility for solving the problem at hand. By repeating this process of ranking, selection, and mating of problem-solving functions across many generations, GP is able to sharpen the initial guesses to converge on increasingly higher ranked solutions to the problem. Across many generations of selective breeding, average and best fitness increase. Since fitness is determined here by utility for solving the problem, increases in fitness indicate better solutions to the problem of interest. The process is closely analogous to

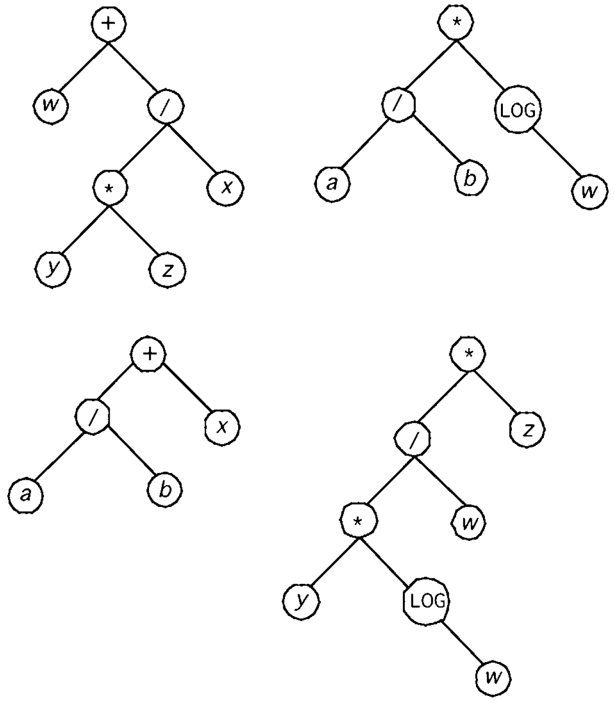

Figure 1. Any mathematical equation can be expressed as a tree. For example, the tree at the top left expresses the equation $w + [(y * z) / x]$. The one beside it expresses the equation $(a / b) * \log(w)$. We can mate any two equations by randomly swapping subtrees that compose them to produce children—equations that have the same elements as their parents. The two trees at the bottom are children of the two at the top. This is an illustrative example only, since the same two parents might equally well produce a large number of other offspring.

selective breeding in biology, where the breeder decides which animal is good enough to be allowed to breed. Following repeated breeding sessions, we select the best solution that has evolved (for formal analyses, see Koza, 1992; see Holland, 1992, for a closely related analysis). We used a modified version of code written and patented by Koza (his original unmodified code is downloadable from ftp://ftp.aic.nrl.navy.mil/pub/galist/src/koza.gp.txt). With minimal familiarity with the Common Lisp programming language and access to Koza's book, it is easy to adapt that code to evolve solutions to problems such as the ones we will consider below.

Estimation problems, such as the ones we consider in this paper, have a natural fitness function. Good solutions may be defined as those with a high absolute correlation of the estimate with the data to be estimated. Because correlations are well defined and continuous across the range of -1 to 1, it is possible to compare the output of any two estimator equations and unambiguously decide which one is better—the one with the larger absolute correlation with the data.

As a technique for modeling relations between variables, GP is completely general. It does not require any a priori assumptions about variable distribution or about the nature of the relationships between variables. Importantly, it can capture and combine both linear and nonlinear relations of predictor variables to the target variable. Those relations can be *radically* nonlinear—that is, they can include discontinuities that are defined by logical relations. For example, we may encounter situations in which Variable A acts in one way within a certain range of Variable B, but in an entirely different way outside of that range. An equation can describe such behaviors easily if it has *if/then/else* operators, by simply specifying the following: If the value of B is within a specified range, use Equation 1 to describe A's behavior; else, use Equation 2. GP can use logical operators as naturally as it uses any other formally specifiable operator.

The problem of including data-set–specific variance may be restated, in ecological terms suited to GP, by saying that equations evolved on a single data set are too well adapted to their evolutionary niche (the initial data set) and utterly unable to "prosper" in the slightly different niche offered by data sets outside that niche. This is the problem of overfitting, discussed in more detail in the General Discussion section at the end of this paper. The general solution to the problem of overfitting is cross-validation—insisting that any solutions put forth be applicable not only to the original data set, but also to new data sets. In order to address the problem when GP is used, we developed a technique called *averaged multitest fitness* that builds cross-validation into the fitness function. In its repeated application during evolution, this technique is analogous to the k-fold cross-validation technique (Stone, 1974) that is now often used in training neural networks. K-fold cross-validation cross-validates a predictor function by deriving it k times on k different subsets of the data set of a size that is $(k - 1)/k$ of the total data set. Error feedback is obtained from the withheld $1/k$th of data. Averaged multitest fitness implements an analogous cross-validation technique by defining a fitness function that directly rewards (by selecting for) the ability of an evolved equation to generalize to a different corpus. Instead of testing the ability of each predictor equation to estimate a value from one data set, one tests every equation's ability to estimate the value for 10 randomly selected subsets of that data set. The fitness value of that equation is then calculated as the average correlation of the estimator function output with the value to be estimated over all 10 of these runs. By requiring an equation to perform well at estimating a value for 10 randomly selected subsets of a data set, instead of for the entire data set, the averaged multitest fitness confers a higher replication probability upon those equations that perform well in varying *adaptive environments*—that is, data sets. It thereby minimizes the possibility of developing overspecialized adaptations to a single data set.

Like many computational models, GP output is generated by a heuristic and, thus, offers solutions that are neither determinate nor unique. Different runs lead to different outcomes. However, since evolved functions are explicit in their mathematical specification of how the variables under study combine, they can be used as predictions that can be directly tested on a novel corpus, where the amount of variance for which they account can be quantified. As well as being amenable to formal analysis, such functions can be easily represented in a graphic form. This makes it easy to point to possible relationships in the data and to analyze the variability between different evolved solutions in order to gain insight into the reliability of those relationships. Thus, in addition to providing solutions to complex problems, these techniques enable us to quantify our confidence in the solutions and, consequently, gain insight into their utility.

Our purpose in the remainder of this paper is not to present novel scientific results. The data we consider are either described in detail elsewhere or relate to toy problems of trivial scientific interest. Our goal in this paper is only to illustrate how GP may be used to gain insight into a diverse set of problems. We will briefly outline four different problems to which we have applied the technique and will present evidence, in each case, of the utility of the evolved solution.

EXAMPLE 1 The Effect of Orthographic Neighborhood on Lexical Word Access

In order to explicitly test the ability of GP to uncover testable relations, we chose to have it discover an interaction that had already been studied extensively and, therefore, was relatively well understood: the effect on lexical decision RTs of orthographic frequency and orthographic neighborhood (ON) size (Coltheart's *N*, first defined in Coltheart, Davelaar, Jonasson, & Besner, 1977). Lexical decision is a widely used task in which subjects are asked to decide as quickly as possible whether a presented letter string is a legal word or a nonword (e.g., *frip*). Orthographic frequency is the frequency with which a word is encountered in written text, expressed as the number of encounters per million words (e.g., the word *good* has a frequency of 911 occurrences per million, whereas the word *husk* has a frequency of 2 occurrences per million). ON size is defined as the number of different words that can be created by changing one letter of a word while maintaining letter positions (e.g., the neighborhood of the word *man* includes the words *ban*, *mad*, *mat*, etc.).

English lexical decision studies of ON typically produce a frequency-modulated facilitatory neighborhood size effect. Low-frequency words with large ONs are responded to more rapidly than low-frequency words with small ONs (see Andrews, 1997, for a review). This facilitatory effect disappears with high-frequency words (Andrews, 1992; Sears, Hino, & Lupker, 1995).

In this initial example, we investigate whether GP could evolve an equation that captured and extended our understanding of this well-documented frequency-modulated ON effect.

Method

Subjects. We asked 120 University of Alberta 1st-year undergraduate psychology students to perform lexical decisions for academic credit. Each subject was randomly assigned to one of four experimental conditions.

Stimuli. The subjects were asked to make lexical decisions on 600 randomly selected four-letter words. These were divided into four sets of 150 words each, with each subject seeing only 150 of those words. The same set of 150 four-letter pronounceable nonwords (e.g., *frip*) was used in each of the four between-subjects conditions.

Procedure. The subjects were instructed to decide as quickly and accurately as possible whether randomly presented single items were real English words. They responded with keypresses to the computer keyboard, using the dominant hand for *yes* responses and the nondominant hand for *no* responses (using the "z" and "?" keys). In every trial, a 50-msec blank screen was followed by a 250-msec fixation cross that appeared at the center of the computer display. Following the fixation, the item appeared and remained on the screen until the subject entered a response, using the designated keys. We discarded erroneous responses and responses with RTs less than 250 msec or greater than 2,500 msec. The remaining RTs were averaged together to get an average lexical decision time for each of the 600 words.

As an additional cross-validity check, we used a cross-validation technique known as the hold-out method. We divided the data set into two subfiles. One file, containing 450 randomly selected words, was the *development set*, which was used to derive the regression equation and evolve the predictor equation. The remaining 150 words were set aside as the *test set*, used to test the ability of linear regression and GP-evolved equations to predict validation scale scores on a new data set.

GP. We included frequency and ON measures in the input file for the GP, forcing it to maximize the function that related these two variables to RT.

The initial 2,500 equations at generation 0 were randomly generated Lisp functions trees that used the available variables (as well as randomly generated real numbers) as leaves and 10 one-, two-, or three-argument functions (listed in Table 1) as internal nodes. From these, the GP program selected the individual equations to be used for breeding at the next generation. We used a breeding scheme called *fitness-proportionate selection with greedy over selection* (Koza, 1992). Under this scheme, the fitness scores of each evolved equation is divided by the total summed fitness of its generation. The equations that account for the top 16% of the total fitness are selected for reproduction with an 80% probability, whereas the equations accounting for the bottom 84% of the total fitness are chosen with a 20% probability. The GP was allowed to calculate 75 generations per run, after which it began again with a new random set of 2,500 equations. At each generation, all 2,500 predictor equations were allowed to estimate RTs for 10 randomly chosen 30-word subsets of the 450 words in the fitness corpus. They were ranked by the fitness function according to how well their average estimate of the RTs of those 10 subsets correlated with the actual RTs.

We adopted a convention of running eight runs of 75 generations, selecting the equation whose output was maximally correlated with the RTs as the best predictor. The value of this parameter, as well as the population size and subset size, was chosen arbitrarily. The population size and run length are very generous by the standards set out in Koza (1992). Koza often solved complex problems with much smaller population sizes and shorter runs.

Results

Two estimator functions from the eight 75-generation runs were about equally good predictors of the RTs. Their output correlated with RTs at .47 and .48 ($p < .01$). The functions they produced are graphed across a range of variation in Figure 2, and the predicted RTs are graphed against the actual RTs in Figure 3. One of the two func-

Table 1
The 10 Functions Available for Use in Evolved Functions

Single-argument functions	
Protected square root	Square root of absolute value of argument
Protected log	Natural log of absolute value of argument, or 0 if argument is 0
Two-argument functions	
+	Adds Arg1 to Arg2
*	Multiplies Arg1 by Arg2
−	Subtracts Arg2 from Arg1
Protected /	Arg1/Arg2 with error protection: division by 0 returns 0
Equal	Returns 1 if Arg1 = Arg2; else returns 0
<	If Arg1 < Arg2, returns 1, else 0.
>	If Arg1 > Arg2, returns 1, else 0.
Three-argument function	
If	If Arg1 is 0 returns Arg2; else returns Arg3

tions shown in Figure 2 (Equation 2) collapsed all ONs greater than two into the same values across the range of frequency. The other equation (Equation 1) differentiated between ONs at every level over which they were defined.

These two independent estimates were almost identical to each other, correlating at .98. When the predictor equations were tested against the 150 RTs from words that were not in the original input set, their estimates correlated with those new RTs at .61 and .60 ($p < .01$). The two estimate sets of these new data points correlated with each at .99.

For the purposes of comparison, we also ran a multiple regression on the 450 word input set, using ON and frequency as predictors of the RTs. The model produced estimates that correlated at .22 ($p < .01$) with the RTs. When we applied the regression equation to the 150 words that had been kept aside, the correlation was .20 ($p < .01$). The GP model therefore accounted for over four times more variance in original input set (23.2% vs. 4.8%) and (more important) over nine times more variance in the unseen data set (36.5% vs. 4.0%) than the linear regression equation did.

Discussion

Other than the aforementioned collapsing of ONs greater than two, the two equations graphed in Figure 2 describe an extremely similar relation between ON, frequency, and RT, as their correlation at near-identity levels indicates. They both clearly capture two known characteristics of the relationship between these three variables. The first is the frequency-modulated facilitatory neighborhood size effect. The effect of ON is clear when frequency is low and disappears when frequency is high. The second is the declining effect of frequency (independent of ON) on RT. When frequency is low, relatively small increases have a relatively large facilitatory effect on RT. However, the flattening of the curve as fre-

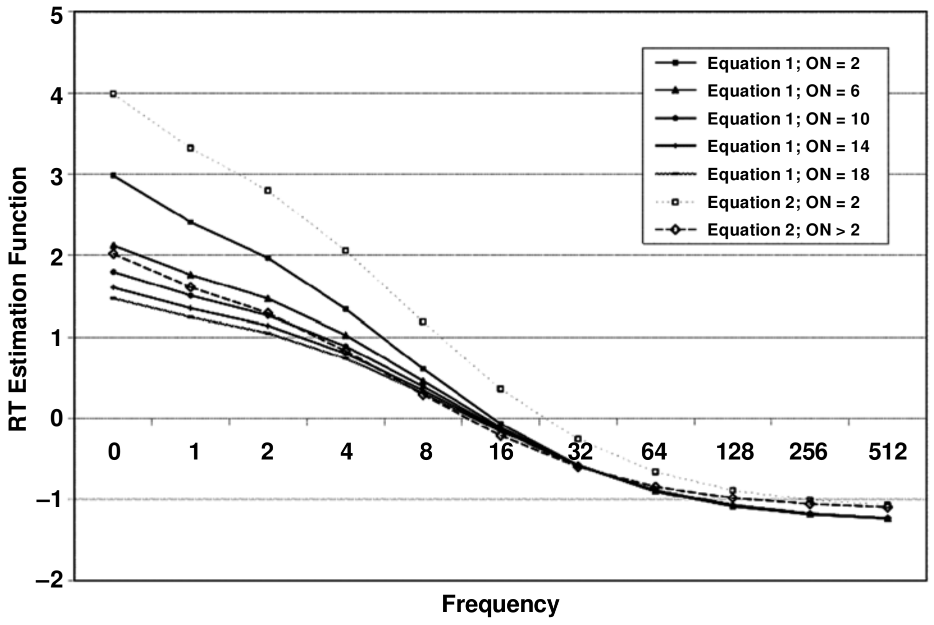

Figure 2. Graph across a range of variation of the two best-evolved functions relating orthographic neighborhood (ON) to frequency. Equation 1 was sensitive to changes across the range of ON; Equation 2 distinguished only between ON ≤ 2 and ON > 2.

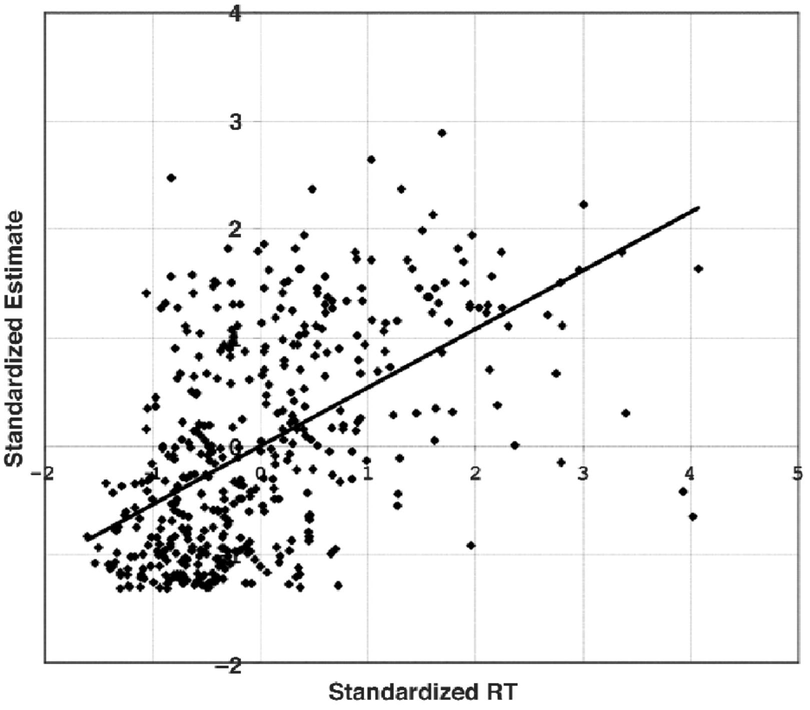

Figure 3. The most highly correlated evolved predictor equation in Example 1: standardized predictions graphed against standardized reaction times (RTs; r = .48).

quency increases shows that even large changes in frequency have little effect on RT when frequency is high.

This graphical display of the function allows for the formulation and testing of new hypotheses about the shape of the function relating ON and frequency that would be difficult or impossible to formulate using only factorially designed experiments. To illustrate this, we use Equation 1, which was sensitive across the range of ON, as our measure.

One hypothesis that is suggested by an inspection of Figure 2 is that the effect of ON disappears, on average, somewhere between a frequency of 8 and 16 occurrences per million, since it is in this frequency range that the lines from the different ON sizes merge indistinguishably. To test this, we divided the 450 word corpus into three ON bands: 0–7, 8–16, and >16. We then conducted an analysis of variance on the RTs with log(OFREQ +1) covaried out ($p \leq .001$). There was a significant difference in RTs between the three frequency-controlled ON groups [F(2,446) = 4.2, $p < .02$]. The average adjusted RTs, in order of increasing ON band, were 671 msec (*SD* = 10 msec) with an ON of 0–7, 637 msec (*SD* = 10 msec) with an ON of 8–16, and 628 msec (*SD* = 8 msec) with an ON greater than 16. A post hoc Tukey's test confirmed what was clear from the inspection of these values: The significant effect was due to differences between the 0–7 band and the other two bands ($p < .05$ in both cases), with no significant difference between those two higher bands ($p > .5$). With the effect of frequency partialled out, there is no difference between words with an ON of 8–16 and those with a higher ON. As was hypothesized, this suggests that ON has a significant effect on word access RTs only for words with frequencies below about eight occurrences per million. It should be noted that this does not diminish the importance of ON as a variable, since the vast majority of English words fall into this frequency range: 80.4% of all four-, five-, and six-letter words in the WordMine database (Buchanan & Westbury, 2000) and an even higher percentage of longer words.

The formula graphed in Figure 2 also allows for the formulation of a hypothesis about the size of ON effects. When the standardized estimates across all frequencies of words with high ON ($\geq$8) and low ON ($\leq$2) are averaged, the evolved formula estimates that there should be a difference of about 0.3 standard scores between the RTs for words in these groups. The actual difference in standardized RTs for words in these high-ON (average = 646 msec) and low-ON (average = 665 msec) categories is close to this estimate at 0.2 standard scores.

As a final simple check of the accuracy of the equation graphed in Figure 2, it is possible to verify the intercept estimate. The equation suggests that the average RT to all words in the frequency range between 8 and 32 should be −0.1 *SD*s from the mean RT. The actual standardized mean RT in the fitness corpus to words in that range is very close to this estimate at −0.04 *SD*s from the mean RT.

In summary, GP was able to deduce the known relationship between ON and frequency. It captured substantially more of the variance in RTs than did a linear regression equation. Moreover, its representation of that relationship provides more details than does the series of factorial experiments that originally uncovered that relationship. By explicitly showing the shape of the curve describing the relation, we can see at what frequency value the ON effect disappears and can estimate the size of that effect across the entire range of frequency. Without committing ourselves to the stochastically produced GP model as *truth*, these elements of the relationship may be considered as testable hypotheses that can guide further research.

EXAMPLE 2
Orthographic and Phonological Neighborhood Interactions in Lexical Decision

The second example we will briefly discuss is also drawn from the lexical access field, illustrating how nonlinear methods, such as GP, may be able to discern a relationship between predictors where linear methods must necessarily fail.

Westbury and Buchanan (2001) presented the results of an experiment that looked at the interaction between ON, described above, and phonological neighborhood (PN). PN is defined in an analogous manner to ON but consists of the number of words that can be obtained by substitution of one single phoneme. For example, the word *rough* is in the PN, but not in the ON, of the word *rut*. To look at the interaction, Westbury and Buchanan manipulated the number of items that were common to the two neighborhoods. That experiment found a significant interaction between ON and PN on lexical access times: Lexical decision RTs were faster as the total number of unique items in the combined ONs and PNs increased.

Method

After this result was obtained, we used GP in a manner analogous to that described above, with the intention of generating an idealized graph of the effect. We entered orthographic frequency, ON, PN, and the number of elements they held in common as predictors, using a 489-word corpus with lexical decision RTs taken from an on-line database (Spieler & Balota, 1997). Among these four predictors, only frequency was significantly correlated with RTs ($r = -.30$, $p < .01$). The other three predictors correlated with the RTs with $|r| \leq .02$. A stepwise linear multiple regression using the same four variables produced a regression equation that used only frequency and, therefore, did not correlate with the RTs better than did frequency alone, at .30.

GP ran eight 75-generation runs on the 489-word corpus. The most highly correlated evolved predictor function correlated with RTs at .39 ($p < .01$), accounting for approximately 1.7 times as much variance (15.2% vs. 9%) in the data as the multiple regression equation.

Although this evolved equation was better than the linear regression, we were unconvinced that such a weak effect size could explain our robust experimental finding. We suspected that the effect was not fully explained by the variables we had manipulated and entered into the predictor equations. We therefore decided to add in variables that might account for some of the missing variance, focusing on variables relating to the phonological and orthographic frequency of the word beginnings and endings. We added the eight variables listed in Table 2 and controlled for frequency by using only words with frequencies $\leq$ occurrence per million.

A stepwise linear regression using the new variable set was able to add only a single variable, LASTTPFREQ, obtaining a multiple r value of .26, accounting for just 6.7% of the variance.

We again ran GP for eight 75-generation runs. The most highly correlated evolved predictor equation was a nonlinear equation that contained eight variables and correlated with the RTs at .85 (see Figure 4). This equation accounted for over 10 times as much variance in this data set as the multiple regression equation (72.2% vs. 6.7%).

Results

The evolved equation was complex. We simplified the evolved equation by hand and obtained a human-comprehensible equation containing only four variables that correlated with RTs in the database, with $r = .64$, accounting for about 41% of the variance. This equation suggested that there was an interaction between PN, the frequency of the beginning of the word, and the frequency of the end of the word.

In order to verify this relation independently, we conducted another lexical decision experiment that manipulated exactly these variables—PN and the frequencies of the first and last orthographic triplets. The method and results are presented briefly here.

We used only five- and six-letter words of mid-frequency (between 10 and 70 occurrences per million). We selected 12 words in each of the first triplet frequency (probable/improbable) × second triplet frequency (probable/improbable) × whole word PN (large/small) conditions for each subject, from as large a pool of words as we could find in each category (between 12 and 112 words; see Table 3). Each subject's input file was constructed and randomized independently and included 96 randomly selected, orthographically legal nonwords matched to the words on length. Forty subjects participated, undertaking a lexical decision task identical to that described above.

Here we report only correct response RTs for words responded to between 400 and 1,500 msec after presentation. There were main effects of the first triplet frequency [$F(1,39) = 27.2$, $p < .001$], of the second triplet

Table 2
The Eight Variables Added as Predictors of the ON/PN Overlap Effect in Example 2

FIRSTTGN	Occurrences in dictionary of first trigram in initial position in words of same length
FIRSTTGFREQ	Average frequency of words in FIRSTTGN class
LASTTGN	Occurrences in dictionary of last trigram in final position in words of same length
LASTTGFREQ	Average frequency of words in LASTTGN class
FIRSTTPN	Occurrences in dictionary of first triphone in initial position in words of same length
FIRSTTPFREQ	Average frequency of words in FIRSTTPN class
LASTTPN	Occurrences in dictionary of last triphone in final position in words of same length
LASTTPFREQ	Average frequency of words in LASTTPN class

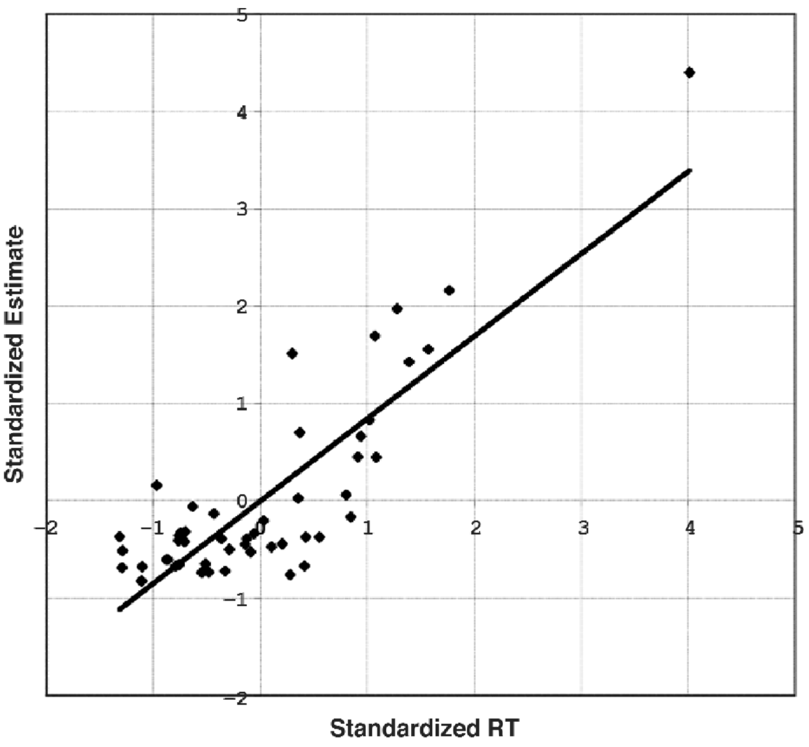

Figure 4. The most highly correlated evolved predictor equation in Example 2: standardized predictions graphed against standardized reaction times (RTs; $r = .85$).

frequency [$F(1,39) = 21.8$, $p < .001$], and of the PN [$F(1,39) = 11.2$, $p < .01$]. Although there were no significant two-way interactions, the three-way interaction whose significance had been captured by the GP was indeed significant [$F(1,39) = 5.4$, $p < .05$; see Figure 5].

On the basis of these preliminary findings (the GP results and their subsequent experimental confirmation), we are currently conducting a series of lexical decision experiments that will enable us to understand this interaction more thoroughly.

Discussion

We have briefly discussed two sets of results that compare GP with multiple regression, using two different data sets. In the first case, GP was able to account for 1.7 times as much variance as the linear regression. In the second case, it accounted for over 10 times as much variance as the regression equation.

However, more important, GP was able to uncover a relation that had been previously unsuspected but that has been subsequently confirmed by experiment and to specify the nature of the relation in sufficient detail to enable that experiment. It is important to emphasize here that the relation described by the GP was *radically nonlinear*, by which we mean that it used logical connectors that changed the predictor equation used on the basis of the values of other predictor variables. Such equations can never be captured by regression equations.

Table 3
Stimulus Characteristics From Verification Experiment in Example 2

	Trigram Frequencies				Trigram Frequencies			
PN	First	Last	*N*	PN	First	Last	OFREQ	Length
High	probable	probable	69	16.51	432.38	518.35	30.84	5.14
High	probable	improbable	24	15.79	361.21	30.50	20.83	5.38
High	improbable	probable	112	16.44	28.32	687.18	20.22	5.35
High	improbable	improbable	29	16.00	25.34	30.79	15.41	5.10
Low	probable	probable	13	0.69	384.31	311.31	40.85	5.92
Low	probable	improbable	12	0.50	301.50	26.83	17.42	5.67
Low	improbable	probable	25	0.40	26.72	357.76	19.28	5.80
Low	improbable	improbable	67	0.30	27.55	27.55	19.04	5.64

Note—Columns 1–3 give factorial categories; columns 5–7 give the numeric values on the basis of which stimuli were entered into those categories. The trigram probabilities represent the average number of times one would encounter that trigram in that position in reading 1 million words of text. First/Last, first or last position of trigram. PN, phonological neighborhood; OFREQ, orthographic frequency.

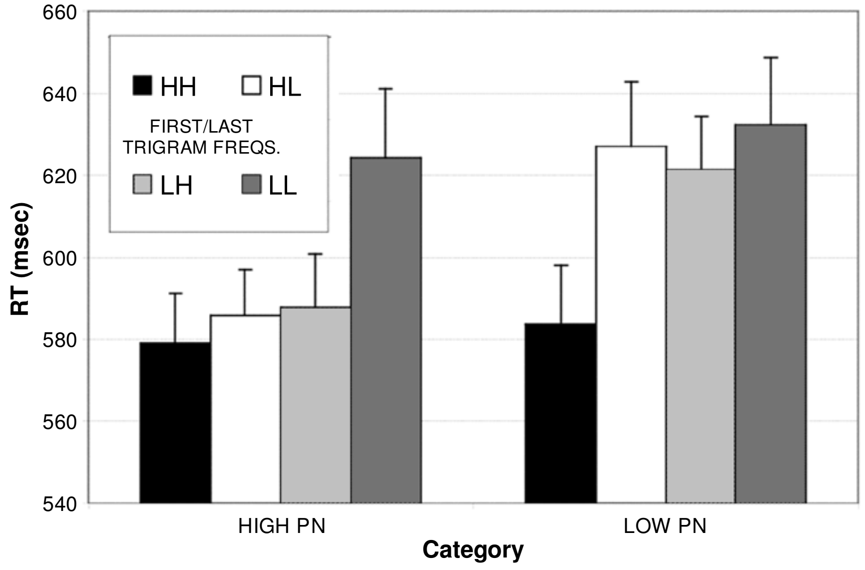

Figure 5. Reaction times (RTs; +*SE*) for a three-way interaction between frequencies of components of the word and its number of phonological neighbors (PNs). H, high; L, low.

GP IN PSYCHOMETRICS

The final examples we wish to consider here use GP to help in designing psychometric instruments. Two difficult problems facing designers of psychometric tests are item selection and item weighting. A test designer must select items for inclusion in the psychometric instrument from a larger pool of candidate items. After the items are selected, the designer must decide whether any subsets of items should be differentially weighted. We have experimented with using evolutionary programming techniques to automate these two steps.

We evolved predictor equations across item sets for two psychometric instruments that were independently developed as a term assignment in a 4th-year undergraduate course in test measurement and design at the University of Alberta. Although these tests were never intended to be used in real settings, they were judged to satisfy the basic requirements for a psychometric instrument. They had clear questions with good face value, they included validation scores, and they had a consistent method of scoring that showed some variability within the population and that allowed summing of question scores as a predictor of the validation scores. They were administered to many subjects.

The validation scores in both cases were self-ratings provided by subjects after they had answered the questions on the test. The values were treated as the "true" value of the construct. Such validation scores may be suspect, since subjects may not know or may not be willing to share their "true score." However, for the purposes of the present, purely formal exercise, this makes no difference; one can think of the tests as predicting only self-ratings, rather than true scores.

As with the ON × frequency example above, we split each of the two data sets into two subsets: a larger development set for deriving the regression equation and evolving the predictor equation and a smaller test set to test the ability of the equations to predict validation scale scores on new data.

Test 1: Geekiness

The first test was designed to measure the construct of *geekiness*, the extent to which a person is a geek. This test was validated against self-rated geekiness on a Likert scale. The test consisted of 76 questions with Likert scale responses. Cronbach's alpha (a standard measure of internal reliability) was .92, suggesting that the test was highly reliable.

The validation set contained 59 subjects. The test set contained 30 subjects.

We compared the ability of three techniques to estimate the subjects' self-ratings from the questionnaire answers (Table 4). Each technique was first used on the development set and then applied to the test set.

One technique was correlating the summed score with the self-ratings. The questions were designed so that a higher score would be higher on the construct, so summed scores were expected by design to correlate with self-ratings. In concrete terms, a self-professed geek would have a high score. On the development set, the correlation of summed scores was .54. On the test set, it was .59.

The second technique was multiple regression. We entered the question scores as predictors for the self-rating scores. A multiple regression equation correlated with the self-ratings on the development set at .70 ($p < .01$). When the same regression equation was used to predict scores on the unseen test set, the correlation was a nonsignificant .20 ($p > .05$).

The question scores were entered into a GP program to predict the self-rating scores. The best evolved equation produced values that correlated with the self-rating scores at .89 ($p < .01$). When that equation was used to predict scores on the test set, its predictions correlated with subjects' self-ratings to a significant degree ($r = .56, p < .01$). These results are particularly noteworthy in light of the fact that the evolved equation used only 12 of the 76 responses. GP managed to find a nonlinear weighting scheme that used a small fraction of the responses to achieve a predictive accuracy that was nearly identical to the summed score around which it had been explicitly designed. We now have at our disposal a fairly robust and easy to administer 12-question test of geekiness.

Test 2: Test Anxiety

The second instrument was designed to measure the construct of *exam anxiety* in students. This test was validated against self-rated exam-related anxiety. After elimination of bad questions, using item analysis techniques, the test consisted of 17 questions with 4-point responses. Cronbach's alpha was .81, suggesting a high degree of reliability. The validation set contained 57 subjects. The test set contained 25 subjects.

We compared the ability of the same three techniques to estimate the subjects' self-ratings from the questionnaire answers (Table 4).

The summed score correlated with self-rated exam anxiety at .77 ($p < .01$) on the development set and at .54 ($p < .01$) on the test set.

A multiple regression equation correlated with the self-ratings on the development set at .85 ($p < .01$). When the same regression equation was used to predict scores on the shorter unseen test set, the correlation was .47 ($p < .05$).

A GP-evolved equation correlated at .93 with the development set on which it was evolved. That equation produced estimates that correlated at .49 ($p < .05$) with the test set self-ratings, using 9 of the 17 questions.

The results are similar to those seen on the first test. In both cases, GP was able to account for more variation in the development set than did the raw score around which the test was designed or a multiple regression equation. In both cases, it was able to predict about as much variation in the test set as the summed raw scores, while using a small subset of questions.

Both of these tests use toy data sets. However, by combining the power of GP to evolve nonlinear predictor equations with much larger preliminary question sets, it may be possible to design more accurate psychometric tests than we could without using GP. Moreover, a careful analysis of the evolved predictor equation may provide insights into weighting of questions and into relations between questions and, thereby, into the formal structure of the constructs one wishes to measure.

GENERAL DISCUSSION

In this paper, we have briefly considered four examples of the use of GP in psychological settings. The first example provided an opportunity to test GP's ability to describe variance, by focusing on a situation in which the relation between predictors and a dependent measure was already known in rough outline from a long series of factorial manipulations. We showed that the known relation was captured by two independently evolved equations. Those equations also allowed us to estimate the values of relevant parameters in more detail than the factorial experiments had. This detail motivated explicit and testable hypotheses—for example, about the value of word frequency at which ON effects disappear. In the second example we showed how GP might be useful in a situation in which the relevant parameters and their values were not known. We evolved an equation that described a radically nonlinear relationship, using a small number of variables chosen from a larger set. We then designed a factorial experiment to test this relationship independently and presented data from that experiment in support of the hypothesized relationship. Finally, we considered two toy problems in psychometrics, presenting evidence that GP was able to select and weight items for inclusion in psychometric instruments for measuring two different constructs.

In all of these examples, we compared the achievements of the nonlinear approach of GP with standard linear multiple regression techniques. In every case, the evolved equation was superior to the regression equation. No linear method could have found the complex relation described in Example 2. More important, without some a priori information regarding the nature of the

Table 4
Results From Using Multiple Regression and Genetic Programming (GP) to Conduct Item Analyses in Psychometric Tests

	Test 1		Test 2	
	Development Set	Test Set	Development Set	Test Set
Summed score	.54	.59	.77	.54
Multiple regression	.70	.20	.85	.47
GP	.89	.56	.93	.49

variable interactions, it is unlikely that any human calculation could have arrived at the complex but nonetheless informative equations.

The preceding point is the real strength of the GP method and the point that we wish to emphasize in this paper: the ability to systematically search the huge space of equations that describe relations in a data set (the relation space), in order to highlight regions of that space that are worthy of human attention. The reason that one sees nonlinear descriptions of human data relatively rarely, as compared with linear descriptions, is that, by relaxing the linearity requirement, one radically increases the size of the relation space—that is, one increases the number of potential functions that need to be considered. Without GP, a tremendous amount of human effort might be required to discover a good nonlinear equation. With GP, simple computer programs do the work and provide one with explicit comprehensible descriptors of the effects of interacting variables on human behavior.

Neural networks are also very good at finding solutions to problems of variable relation, but those solutions are represented in a way that makes them even more difficult for humans to understand than the (also often complex) explicit mathematical representations of GP. The explicit nature of the evolved solutions therefore gives GP an advantage over the use of neural networks to search relation space.

The Problem of Degrees of Freedom

This increase in the size of the relation space is the main strength of methods that search relation space for nonlinear variable relations but is also a common source of criticism of those methods. The criticism that we have seen most often raised against the use of GP in solving problems of the kind we have considered in this paper is that it *has too many degrees of freedom* or (equivalently) *is overfitting*—that is, that it may find solutions that fit to particularities in the input set but fail to find solutions that are general.

Our application of two cross-validation techniques—the repeated application of averaged multitest fitness and the post hoc check using the hold-out method—is of course intended to prevent overfitting, by requiring that solutions that are judged good predict well on many different data sets. We have presented empirical evidence above showing that solutions evolved with averaged multitest fitness do cross-validate to new data sets.

However, even with this empirical evidence in hand, it is worth examining the conceptual underpinnings of the claim that GP has too many degrees of freedom. We will first consider the status of operators as degrees of freedom and then address how GP deals with the degrees of freedom in input parameters. Along the way, we will consider the possibility that the significance value of any evolved equation needs to be adjusted in some way to reflect the number of equations that were searched to find it.

A degree of freedom is a parameter in a problem description or problem solution that is free to vary independently of other parameters in the same description or solution. This may seem clear enough, but what constitutes a parameter, especially an independent parameter, is not always as clear in any particular case as it seems from this textbook definition. This is in part because descriptions and solutions come in a bewildering variety of representational forms. How a problem or solution is represented can change our understanding of the apparent degrees of freedom that the problem or solution has.

Degrees of freedom I: The role of operators. It may seem obvious that an increase in the number of allowable operators that are accessible to any modeling system must correspond to an increase in the degrees of the freedom that system has available. In fact, however, it is not the case. The assumption that operators are degrees of freedom stems from a failure to distinguish between the concrete computational representation of any problem and the abstract computational structure underlying a particular representation. The computational operators used in a computation, including the operators evolved in GP, are not a countable property of that computation. Operators are a representational choice, one among an infinite number of ways of representing any particular computational structure.

An easy way to see that this is so is to consider the ultimate reduction of computational operators. Turing proved in 1936 that any computation that could be computed by any computer could also be computed by the simple machine that we now know as a Turing machine. A Turing machine is an idealized computational device that can read and write binary digits to an infinitely long moveable tape. It has three operators. It can read from its tape, it can write a binary digit or punctuation marker, and it can move its tape. Since Turing proved that a Turing machine can compute anything that is computable, it is a fact that any computation can be expressed as a function of three operators—namely, the three operators of a Turing machine.[1]

Although Turing machines are theoretical devices of limited usefulness for real-world computation, the reducibility of operators to lower level functions is of direct practical importance in understanding many real computational situations. The same kind of argument that reduces all possible computational operators to a few Turing machine operators also has bearing when one is considering searching solution spaces, using methods such as GP or neural networks. Since there is no sense in which high-level operators are independent of each other, they cannot be treated as degrees of freedom. One cannot simply count up the number of function calls that a computer program happens to make to obtain a useful measure of the complexity of that computation. Function calls are not a measure of computational complexity, because they have no necessary relation at all to the "real" complexity of any computation. They are merely notational devices to allow a particular system to work in a way that is convenient for the hardware or software environment on which it is running or for the person who is going to read or write the program. This is as true for the *meta-function calls* (multiple operators conjoined in an equation) that one evolves in GP as it is for their component operators. An operator is always a convenient fiction.

Meehl and MacCorquodale (1991), extending discussions by Carnap (1936), Tolman (1938), and Hull (1943), called these kinds of convenient fictions *intervening variables*, in order to distinguish them from *hypothetical constructs*, suggesting that “a failure to separate these leads to fundamental confusions” (p. 262; see also the discussion of the nomological net in Meehl, 1977, and the extensive related discussion in Vaihinger, 1935). By Meehl and MacCorquodale’s definition, intervening variables have to be defined in terms that are reducible to empirical events or properties. They thereby serve as a summary of those empirical entities.

The summarizing role that is assigned by definition to intervening variables guarantees that an intervening variable may be called into question only by calling into question some specified set of empirical facts. Given that they are only selected representations or summaries of facts, rather than facts themselves, intervening variables themselves cannot be denied or challenged, for no ontological claims are ever made about their existence and they are (by definition) not allowed to contain theoretical elements that are not grounded in empirically verifiable measures. As Meehl and MacCorquodale (1991) noted, “the only consideration which can be raised with respect to a given proposed intervening variable . . . is the question of convenience” (p. 260).

This distinguishes them from hypothetical constructs, which contain what Reichenbach (1938) memorably referred to as *surplus meaning*—namely, a claim of existence. Setting aside the philosophical questions raised by Reichenbach’s terminology (which are, however, discussed by Meehl & MacCorquodale, 1991), we can say that hypothetical constructs make claims about what is. Intervening variables make claims about how hypothetical constructs relate.

When mathematical and logical operators are understood to be intervening variables, it becomes clearer why some people believe that GP is cheating in a way that linear regression (they believe) is not. The relation space is identical to the set of all possible intervening variables. Both consist of the set of computable relations between a set of predictors and a value to be predicted. GP differs from regression equations because GP is allowed to search a (possibly extremely large) region of relation space, whereas a regression equation is confined to a single point in it. Recognition of this difference is why concerns about degrees of freedom arise.

The notion that methods that are allowed to search relation space are “cheating” stems from two facts. One is that such methods have access to a subspace containing a great many possible intervening variables. The second is that the location and shape of that searched subspace may be a function of which operators one gives the system. Notwithstanding the discussion above of Turing machine equivalence of computations, it is true that some operators can never be used within any limited computational system. For example, in the GP system, there is no recursion or iteration operator, so a general square root operator could never be evolved by natural selection from addition, subtraction, multiplication, and division, despite the fact that Newton proved it was possible to compute square roots using these four functions with iteration. If one did not provide a square root operator, the system could not search the vast subspace of relation space that cannot be accessed without a square root operator. In this way, the adding of operators can (although it need not always) increase the number of computable solutions to any problem. This makes the adding of operators seem closely analogous to the adding of independent input variables, which also increase the number of possible solutions and which do increase the degrees of freedom of the problem.

However, the idea that it is cheating to allow arbitrarily directed searches of subspaces of relation space derives from the confusion of intervening variables with hypothetical constructs that Meehl and MacCorquodale (1991) warned against. One has no right to repeatedly search through hypothetical constructs until one finds some that satisfy one’s criteria. This would be akin to searching for an entity that happened to have the distributional properties one needed to predict some dependent measure and then adding that entity as a post hoc addition to one’s theories of that measure.

However, if one is searching the space of intervening variables, one is by definition making no ontological claims, but only claims of representational convenience. Therefore, one is free to search the space as thoroughly as one likes. There can be no question of violating statistical or empirical assumptions by allowing ourself to examine large numbers of intervening constructs, since their properties are not ontological but are, rather, defined entirely by the very utility one seeks to maximize by searching through them. In anthropomorphic terms, we might say that the intervening variables we call operators are computational products of the imagination, imaginative suggestions about which region of relation space to search. Those suggestions are informed by information one provided to the search system (GP or neural network) about the kinds of solutions one would most like to see, for reasons of aesthetics, representational convenience, or theoretical allegiance. One tells the computer what kinds of solutions are acceptable by defining the high-level operators with which it may communicate its findings and by defining some way of recognizing good solutions—error feedback in neural networks and a fitness function in GP.

It is illuminating to consider linear regression in terms of a search of suggested subspaces of relation space. The common practice of using linear regression in contemporary psychological research does not, of course, stem from that fact that the single well-defined point in relation space that is specified by a regression equation has a guaranteed status as the best possible solution. There clearly is no such guarantee. The practice stems from the fact that regression equations are simple and well defined.

Simplicity and well-definedness are scientific virtues to which all scientists must aspire. However, oversimplicity is not. As the examples in this paper demonstrate, if one limits oneself to regression equations, one is very likely to miss intervening variables that may be only a little more computationally complex and only a little distance off in relation space but that may be very much better for the purposes at hand, however defined. Of course, this claim is already trivially appreciated by all of us. Even the most devoted proponent of linear regression will not fail to take a nonlinear transformation of a highly skewed variable (such as a logarithm of word frequency) before it is entered into a regression equation.

In championing technologies such as GP and neural networks, researchers are suggesting that we trade algorithmic uniformity for explanatory elegance. We give up the useful tool of a common language for describing relations (regression equations) in return for the chance to find another description that may be more suitable for a specified purpose, such as (but not necessarily limited to) accounting for variance in some dependent measure.

One way of thinking of this trade is by making an analogy to Type I and Type II errors. Methods of describing relationships that do not search relation space at all (such as regression equations) make a very cursory test of the claim that there is a significant relation between hypothetical constructs and a dependent measure, because they test only one possible relation between those constructs. They thereby run a high risk of making a Type II error—incorrectly rejecting a relationship that actually does exist. On the other hand, methods that are able to search the relation space run an increased risk of making the Type I error that amounts to overfitting the data—that is, they risk incorrectly announcing evidence in favor of a relationship that does not actually exist. Both Type I and Type II errors mislead by misrepresenting matters of empirical fact. However, it may be argued that Type I errors are easier to correct than Type II errors, because the error is explicitly stated and, therefore, amenable to obvious experimental disconfirmation (see the discussion of error tradeoffs in Hays, 1994, p. 284). In Example 2 above, for example, we ran the risk of making a Type I error. However, we may have some confidence that we did not, because we explicitly conducted an independent experimental test of the claim in a follow-up experiment. Type II errors may be harder to detect and correct than Type I errors, because a Type II error is a nonspecific error of omission, rather than a specific error of commission. Having concluded, on the basis of failing to find a single specific relationship between variables, that there is no relationship between those variables, one cannot easily know what experiment to conduct to cross-check that conclusion. The proper experiment might be any one of the infinite number of experiments that test the infinite number of other possible relations.

We are certainly not recommending that researchers exhibit a cavalier disregard for Type II error. Nor do we intend to claim that the point to which GP takes us must necessarily be the best possible location in relation space, anymore than anyone could reasonably claim optimality after conducting a regression. Perhaps if one had added one more operator to our operator set, let the system run for a few generations longer, or added one more predictor, one might have found a location that was better, because it was specified in a way that was computationally more compact, because it accounted for more variance, because it was more closely tied to a theoretical network of ideas, or because it met some other pragmatic criterion. One always risks committing an error that might be called a Type III error—accepting a solution that, although correct as far as it goes, is nonetheless nonoptimal.

We do claim—and provide the examples above to demonstrate—that GP may guide one to places in relation space that are worth exploring more closely. Because relation space is infinitely large by definition (even a single parameter can be transformed in an infinite number of ways—e.g., by squaring it, taking its logarithm, dividing it by its own square, etc.), an automated advance scout, such as GP, is useful to have.

Another criticism sometimes leveled against GP is that it is "cheating" to hand-simplify an evolved solution. When one understands that GP is nothing other than a tool to help one find a convenient set of intervening variables—a convenient representation of a problem of interest—it should be clear that there is nothing that rules out the precedent, simultaneous, or subsequent use of other tools that maximize representational convenience, including hand-simplification of evolved solutions. One real limitation of GP is that the evolved solutions are sometimes extremely unwieldy and complex. They may even include *junk DNA*—subfunctions that demonstrably contribute nothing to the solution but are passed along because they do no harm. For example, it is not uncommon to find *if–then–else* statements that always evaluate to true or to false and can, therefore, be replaced by their inevitable consequent. It is rare that an evolved equation will be immediately comprehensible. The simplification of an evolved equation amounts to a very fine-grained, slow, and directed search through a region of relation space that has been identified by GP as worthy of further exploration. Such hand-guided search must be subject to the same rigor as the automated search was, with an emphasis on cross-validation of any proposed representation.

We have heard arguments that such searches through relation space constitute a violation of statistical assumptions, requiring that one divide one's standard of statistical significance by the number of relations considered, as if each relation were an independent t test. Since GP searches hundreds of thousands of relations, this insistence would render it statistically useless if it were correct. However, it is not. Searches through relation space are routinely conducted by scientists, without application of any statistical correction for the number of relations that are considered. When one sits down with a pencil and paper to diagram one's understanding of some phenomenon in order to decide which experiments to conduct, one is using one's prior understanding and one's own imagination to search through relation space. No one would argue that one should adjust the *p* values of conducted experiments in proportion to the number of experiments one imagined conducting but did not or in proportion to the number of ways one imagined one might graph the data. GP is a way of cloning one's imagination. By providing it with hypothetical constructs and operators, one supplies the prior understanding that constrains the search for an appropriate representation of relationships in the data. The fitness function instantiates and provides a quantifiable measure of what constitutes a solution worth imagining.

Equating a search through relation space to imagining ways of representing a problem is not a rejection of the role of theory or of the hypotheticodeductive method in science. It is simply a recognition of the fact that theoretical claims and empirically based descriptions of the relations between theoretical elements must be separate. Theories can postulate which hypothetical constructs may play a role in any particular phenomena and may certainly attempt to specify the precise nature of that relation. However, when it comes to questions of the real nature of the relation between those constructs, theoretical claims cannot trump empirical facts. If a systematic relation between hypothetical constructs can be reliably demonstrated, by any means, theory is not in a position to rule this empirical finding in or out; it can only adapt to the finding, as it must adapt to any other relevant empirical evidence. For example, a theory might claim that ON and frequency combine in a linear way in their effect on RTs in lexical decision. However, if it can be shown that lexical decision RTs may, in fact, be more reliably predicted by using a nonlinear relationship, the theoretical claim must be rejected or, at least, demoted to the Type III error mentioned above—a suboptimal description that is, nevertheless, correct as far as it goes.

As representational methods become more sophisticated, researchers may even find reason, as Landauer and Dumais (1997) have, to "eschew the conventional stance that the theory is primary and the simulation studies are tests of it" (p. 211). When reliable relations between theoretically motivated elements are discovered that theory either predicts against or cannot explain, those relations must be taken as a piece of evidence that bears on the validity of the theory, on a par with any other empirically grounded finding.

Degrees of freedom II: Variables. The second problem in addressing problems related to degrees of freedom concerns the matter of determining independence of Meehl and MacCorquodale's (1991) hypothetical constructs: the theoretically relevant terms that are known or assumed to correspond to empirically measurable properties that are given to GP as inputs. A fear is that if one provides too many independent inputs to GP, it may be able to *overfit* those inputs, evolving a solution that maximizes the fitness function in some spurious way.

GP can avoid this problem in three ways.

The simplest way is to avoid giving the method inputs that might have a spurious relation to the problem—that is, to avoid using GP for wild fishing expeditions. In all the examples considered in this paper, the problem of spurious relations between the inputs was sidestepped in just this way, simply by excluding inputs whose contributions were not of theoretical or practical interest. In the lexical access Examples 1 and 2 above, we were specifically interested in the relationship between a small number of variables, and so we entered just those variables. The problem of overdetermination from having too many variables clearly does not arise in these cases. In the psychometric Examples 3 and 4, we entered a great number of variables, but we had pragmatic reasons for being indifferent as to the theoretical reality of the evolved solutions. We did not care how many variables, or which ones, were used in the final equation. So long as the set of questions proposed by GP predicted more variance in the validation measure or used fewer questions than the summed score did, that solution was not spurious by definition. All we were trying to do was improve on the summed score. In all cases we have considered in this paper, the solution we had after GP was better than the one we had before, by every relevant criterion we had for evaluating a solution.

The second way that GP avoids spurious solutions is by using repeated feedback of its own proposed solutions. Recall that any equation offered by GP as the best of a set of runs has been ranked against thousands of other solutions, perhaps dozens of times. Spurious solutions are by definition solutions that are not actually good at the task for which they are being evolved but that are mistakenly taken as being good solutions. Such solutions are by their very definition highly unlikely to beat nonspurious solutions in the rankings, especially repeatedly, since by definition nonspurious solutions really are good solutions. Although a solution offered by GP might surprise one by showing that some variable set one had not imagined as being predictive of some measure actually was predictive, it would be odd and improbable to call any solution *spurious* that had been ranked dozens of times as the best solution among thousands of candidates. It would be like denying that a person knew the rules of chess after watching him win hundreds of games against hundreds of skilled opponents. How could anything appear to be so good so often, if it were in fact not good? This, of course, is the point behind the insistence on the importance of cross-validation (Stone, 1974) and the reason we have built cross-validation into our selection mechanism.

If one does have specific criteria for evaluating what constitutes a good or acceptable solution, one should use those criteria. The third way that GP can avoid spurious solutions is to rule them out computationally by extending the fitness function. So long as one has an algorithmically specifiable means of recognizing solutions that are unsatisfactory for any theoretical, statistical, or pragmatic reason, one can breed them out, because any algorithmically specifiable criterion of what specifies a bad solution can be included in the fitness function. For example, it would be very easy to write a fitness function for a psychometric problem that gave fitness values of 0 to any solution with fewer than some specified number *N* of questions in it and, thereby, to search only for a good *N*-question solution. It would be equally easy to specify, in the fitness function, some limits on the allowable correlations between parameters in any evolved solution (to encourage maximal independence of predictors in cases in which the predictors were correlated) or on the number of elements one would allow to appear in any solution (to encourage simple solutions). Placing such constraints on the fitness function reduces the size of relation space. Such reduction may be a good or bad thing, depending on whether or not the excluded region may contain any useful relations that are now beyond reach.

Conclusion

For many researchers, the simple linearity imposed on us through our standard statistical techniques is dissatisfying. Van Orden, Holden, and Turvey (in press) have recently referred to the insistence on searching for linear solutions to scientific problems in psychology as "linear imperialism." Few would argue that the brain is a linear system, and fewer still would argue that most behavior can be linearly described. The attraction of linearity is that it is a *common tongue* that is relatively simple to understand. However, human behavior and functioning are not simple to understand. We are complex beings with cognitive processes that are numerous and ever-changing. GP offers us, with limited costs, the power to manage and explore some of that complexity.

REFERENCES

Andrews, S. (1992). Frequency and neighborhood effects on lexical access: Lexical similarity or orthographic redundancy? *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **18**, 234-254.

Andrews, S. (1997). The effect of orthographic similarity on lexical retrieval: Resolving neighborhood conflicts. *Psychonomic Bulletin & Review*, **4**, 439-461.

Balota, D. A., Cortese, M. J., & Pilotti, M. (1999). Item-level analyses of lexical decision performance: Results from a mega-study. *Abstracts of the Psychonomic Society*, **4**, 44.

Buchanan, L. & Westbury, C. (2000). *Wordmine database: Probabilistic values for all four to seven letter words in the English language* [on line]. Retrieved February 24, 2003, from http://www.wordmine.org.

Carnap, R. (1936). Testability and meaning: Pts. I–III. *Philosophy of Science*, **3**, 419-471.

Coltheart, M., Davelaar, E., Jonasson, J. T., & Besner, D. (1977). Access to the internal lexicon. In S. Dornic (Ed.), *Attention and performance VI* (pp. 535-555). Hillsdale, NJ: Erlbaum.

Hays, W. (1994). *Statistics* (5th ed.). New York: Harcourt Brace.

Holland, J. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* Cambridge, MA: MIT Press.

Hull, C. L. (1943). *Principles of behavior.* New York: Appleton-Century.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, **104**, 211-240.

Meehl, P. (1977). Construct validity in psychological tests. In P. Meehl (Ed.), *Psychodiagnosis: Selected papers* (pp. 3-31). New York: Norton.

Meehl, P., & MacCorquodale, K. (1991.) On a distinction between hypothetical constructs and intervening variables. In C. Anderson & K. Gunderson (Eds.), *Paul E. Meehl: Selected philosophical and methodological papers* (pp. 249-263). Minneapolis: University of Minnesota Press.

Reichenbach, H. (1938). *Experience and prediction.* Chicago: University of Chicago Press.

Sears, C. R., Hino, Y., & Lupker, S. J. (1995). Neighborhood size and neighborhood frequency effects in visual word recognition. *Journal of Experimental Psychology: Human Perception & Performance*, **21**, 876-900.

Spieler, D. H., & Balota, D. A. (1997). Bringing computational models of word naming down to the item level. *Psychological Science*, **8**, 411-416.

Stone, M. (1974). Cross-validation choice and assessment of statistical predictions. *Journal of the Royal Statistical Society,* **36B**, 111–147.

Tolman, E. C. (1938). The determiners of behavior at a choice point. *Psychological Review*, **45**, 1-41.

Treiman, R., Mullennix, J., Bijeljac-Babic, R., & Richmond-Welty, E. D. (1995). The special role of rimes in the description, use, and acquisition of English orthography. *Journal of Experimental Psychology: General*, **124**, 107-136.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **2**(42), 230-265.

Vaihinger, H. (1935). *The philosophy of "as if": A system of the theoretical, practical, and religious fictions of mankind* (2nd ed.). London: Lund Humphries.

Van Orden, G. C., Holden, J. G., & Turvey, M. T. (in press). Self-organization of cognitive performance. *Journal of Experimental Psychology: General.*

Westbury, C., & Buchanan, L. (2001, November). *Interactions between orthographic and phonological neighborhood.* Poster presented at the 42nd Annual Meeting of the Psychonomic Society, Orlando, FL.

Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media.

NOTE

1. In fact, even fewer operators are necessary. Wolfram (2002) has proven, in a roundabout but rigorous fashion, that a particular one-dimensional cellular automaton, which might be characterized as having a single operator, since it is a finite state machine, is Turing complete—that is, it can, in theory, compute anything that can be computed.

(Manuscript received November 19, 2002;
revision accepted for publication February 26, 2003.)