Highly Accurate Symbolic Regression with Noisy Training Data

Michael F. Korns

Analytic Research Foundation, 2240 Village Walk Drive Suite 2305, Henderson Nevada 89052 mkorns@korns.com.

Abstract.

As symbolic regression (SR) has advanced into the early stages of commercial exploitation, the poor accuracy of SR, still plaguing even the most advanced commercial packages, has become an issue for early adopters. Users expect to have the correct formula returned, especially in cases with zero noise and only one basis function with minimally complex grammar depth.

At a minimum, users expect the response surface of the SR tool to be easily understood, so that the user can know apriori on what classes of problems to expect excellent, average, or poor accuracy. Poor or *unknown* accuracy is a hinderence to greater academic and industrial acceptance of SR tools.

In two previous papers, we published a complex algorithm for modern symbolic regression which is extremely accurate for a large class of Symbolic Regression problems. The class of problems, on which SR is extremely accurate, is described in detail in these two previous papers. This algorithm is extremely accurate, in reasonable time on a single processor, for from 25 up to 3000 features (columns).

Extensive statistically correct, out of sample training and testing, demonstrated the extreme accuracy algorithm's advantages over a previouly published base line pareto algorithm in case where the training and testing data contained zero noise.

While the algorithm's extreme accuracy for deep problems with a large number of features, on noiseless training data, is an impressive advance, there are many very important academic and industrial SR problems where the training data is very noisy.

In this chapter we test the extreme accuracy algorithm and compare the results with the previously published baseline pareto algorithm. Both algorithms' performance are compared on a set of complex representative problems (from 25 to 3,000 features), on noiseless training, on noisy training data, and on noisy training data with range shifted testing data.

The enhanced algorithm is shown to be robust, with definite advantages over the baseline pareto algorithm, performing well even in the face of noisy training data and range shifted testing data.

Key words: Symbolic Regression, Abstract Expression Grammars, Grammar Template Genetic Programming, Genetic Algorithms, Particle Swarm.

1 Introduction

The discipline of Symbolic Regression (SR) has matured significantly in the last few years. There is at least one commercial package on the market for several years *http://www.rmltech.com/*. There is now at least one well documented commercial symbolic regression package available for Mathematica *www.evolved-analytics.com*. There is at least one very well done open source symbolic regression package available for free download *http://ccsl.mae.cornell.edu/eureqa*. In addition to our own ARC system (Korns 2010), currently used internally for massive (million row) financial data nonlinear regressions, there are a number of other mature symbolic regression packages currently used in industry including (Smits 2005) and (Kotanchek 2008). Plus there is another commercially deployed regression package which handles up to 50 to 10,000 input features using specialized linear learning (McConaghy 2011).

Yet, despite the increasing sophistication of commercial SR packages, there have been serious issues with SR accuracy even on simple problems (Korns 2011). Clearly the perception of SR as a *must use* tool for important problems or as an *interesting heurism* for shedding light on some problems, will be greatly affected by the demonstrable accuracy of available SR algorithms and tools. The depth and breadth of SR adoption in industry and academia will be greatest if a very high level of accuracy can be demonstrated for SR algorithms.

In (Korns 2012), (Korns 2013), and (Korns 2014) we published both a baseline pareto algorithm and an extreme accuracy algorithm for modern symbolic regression the (*EA*) algorithm. which is extremely accurate for a large class of Symbolic Regression problems. The class of problems, on which the EA algorithm is extremely accurate, is described in detail in those papers and also in this chapter. A definition of extreme accuracy is provided, and an *informal argument* of extreme SR accuracy is outlined in (Korns 2013), and (Korns 2014).

Prior to writing this chapter, a great deal of *tinker-egineering* was performed on the Lisp code supporting both the baseline and the EA algorithms. For instance, all generated champion code was checked to make sure that the real numbers were loaded into Intel machine registers without exception. All vector pointers were checked to make sure they were loaded into Intel address registers at the start of each loop rather than re-loaded with each feature reference. As a result of these engineering efforts, both the baseline and the EA algorithms are now quite practical to run on a personal computer. Furthermore the EA algorithm is extremely accurate, in reasonable time, on a single processor, for from 25 to 3,000 features (columns); and, a cloud configuration can be used to achieve the extreme accuracy performance in much shorter elapsed times.

In this chapter we test the EA algorithm (Korns 2013) and (Korns 2014) and compare the results with the baseline algorithm (Korns 2012). Extensive statistically correct, out of sample training and testing, are used to compare

both algorithms' performance, on a set of complex representative problems (from 25 to 3,000 features), on noiseless training, on noisy training data, and on noisy training data with range shifted testing data.

The EA algorithm is shown to be robust, with definite advantages over the baseline pareto algorithm, performing well even in the face of noisy training data with range shifted testing data.

Before continuing with the comparisons of the baseline and EA algorithms, we proceed with a basic introduction to general nonlinear regression. Nonlinear regression is the mathematical problem which Symbolic Regression aspires to solve. The canonical generalization of nonlinear regression is the class of Generalized Linear Models (GLMs) as described in (Nelder 1972). A GLM is a linear combination of **I** basis functions B_i; i = 0,1, I, a dependent variable y, and an independent data point with M features x = $<x_0, x_1, x_2, , x_{M-1}>$: such that

- (*E1*) $y = \gamma(x) = c_0 + \Sigma c_i B_i(x)$ + **err**

As a broad generalization, GLMs can represent any possible nonlinear formula. However the format of the GLM makes it amenable to existing linear regression theory and tools since the GLM model is linear on each of the basis functions B_i. For a given vector of dependent variables, Y, and a vector of independent data points, X, symbolic regression will search for a set of basis functions and coefficients which minimize **err**. In (Koza 1992) the basis functions selected by symbolic regression will be formulas as in the following examples:

- (*E2*) $B_0 = x_3$
- (*E3*) $B_1 = x_1 + x_4$
- (*E4*) $B_2 = sqrt(x_2)/tan(x_5/4.56)$
- (*E5*) $B_3 = tanh(cos(x_2 * .2) * cube(x_5 + abs(x_1)))$

If we are minimizing the normalized least squared error, NLSE (Korns 2012), once a suitable set of basis functions B have been selected, we can discover the proper set of coefficients C deterministically using standard univariate or multivariate regression. The value of the GLM model is that one can use standard regression techniques and theory. Viewing the problem in this fashion, we gain an important insight. Symbolic regression does not add anything to the standard techniques of regression. The value added by symbolic regression lies in its abilities as a search technique: how quickly and how accurately can SR find an optimal set of basis functions B. The immense size of the search space provides ample need for improved search techniques. In basic Koza-style tree-based Genetic Programming (Koza 1992) the genome and the individual are the same Lisp s-expression which is usually illustrated as a tree. Of course the tree-view of an s-expression is a visual aid, since a

Lisp s-expression is normally a list which is a special Lisp data structure. Without altering or restricting basic tree-based GP in any way, we can view the individuals not as trees but instead as s-expressions such as this depth 2 binary tree s-exp: *(/ (+ x_2 3.45) (* x_0 x_2))*, or this depth 2 irregular tree s-exp: *(/ (+ x_4 3.45) 2.0)*.

In basic GP, applied to symbolic regression, the non-terminal nodes are all operators (implemented as Lisp function calls), and the terminal nodes are always either real number constants or features. The maximum depth of a GP individual is limited by the available computational resources; but, it is standard practice to limit the maximum depth of a GP individual to some manageable limit at the start of a symbolic regression run.

Given any selected maximum depth k, it is an easy process to construct a maximal binary tree s-expression U_k, which can be produced by the GP system without violating the selected maximum depth limit. As long as we are reminded that each f represents a function node while each t represents a terminal node (either a feature v or a real number constant c), the construction algorithm is simple and recursive as follows.

- (U_0): t
- (U_1): (f t t)
- (U_2): (f (f t t) (f t t))
- (U_3): (f (f (f t t) (f t t)) (f (f t t) (f t t)))
- (U_k): (f U_{k-1} U_{k-1})

The basic GP symbolic regression system (Koza 1992) contains a set of functions F, and a set of terminals T. If we let t ∈ T, and f ∈ F ∪ ξ, where $\xi(a,b) = \xi(a) = a$, then any basis function produced by the basic GP system will be represented by at least one element of U_k. Adding the ξ function allows U_k to express all possible basis functions generated by the basic GP system *to a depth of k*. **Note to the reader**, the ξ function performs the job of a pass-through function. The ξ function allows a *fixed-maximal-depth* expression in U_k to express trees of varying depth, such as might be produced from a GP system. For instance, the varying depth GP expression $x_2 + (x_3 - x_5) = \xi(x_2,0.0) + (x_3 - x_5) = +(\xi(x_2\ 0.0)\ -(x_3\ x_5))$ which is a *fixed-maximal-depth* expression in U_2.

In addition to the special pass through function ξ, in our system we also make additional slight alterations to improve coverage, reduce unwanted errors, and restrict results from wandering into the complex number range. All unary functions, such as *cos*, are extended to ignore any extra arguments so that, for all unary functions, *cos(a,b) = cos(a)*. The *sqroot* and *ln* functions are extended for negative arguments so that *sqroot(a) = sqroot(abs(a))* and *ln(a) = ln(abs(a))*.

Given this formalism of the search space, it is easy to compute the size of the search space, and it is easy to see that the search space is huge even for

rather simple basis functions. For our use in this chapter the function set will be the following functions: F = **(+ - * / abs inv cos sin tan tanh sqroot square cube quart exp ln ξ)** (where **inv**(x) = 1.0/x). The terminal set is the features $\mathbf{x}_0$ through $\mathbf{x}_{M-1}$ and the real constant **c**, which we shall consider to be 2^{18} in size.

Our core assertion in this chapter is that the enhanced EA algorithm will achieve, on a laptop computer, in reasonable time, extremely accurate champions for all of the problems in $\mathbf{U}_2$**(1)[25]**, $\mathbf{U}_1$**(25)[25]**, $\mathbf{U}_1$**(5)[150]**, and in $\mathbf{F}_{(x)}$**(5)[3000]** (*note:* $\mathbf{F}_{(x)} = \xi$ *inv abs sqroot square cube quart exp ln cos sin tan tanh*) in reasonable computation times, of a maximum 20 hours (*on an advanced laptop built in Dec 2012*) and a maximum 40 hours (*on an advanced laptop built in Jan 2008*). Most noiseless problems finish far quicker than these maximum time horizons.

Pushing things to the extreme, the enhanced algorithm will achieve extremely accurate champions for all of the problems in $\mathbf{U}_2$**(1)[50]** through $\mathbf{U}_1$**(5)[50]** in a maximum of 160 hours (*on an advanced laptop built in Dec 2012*). Most noiseless problems finish far quicker than these maximum time horizons.

Obviously a cloud configuration will greatly speed up the enhanced EA algorithm, and we will address cloud configurations and extreme accuracy in a later paper. For this chapter, we will develope an extremely accurate SR algorithm which any scientist can use on their personal laptop.

1.1 Example Test Problems

In this section we list the example test problems which we will address. All of these test problems lie in the domain of either $\mathbf{U}_2$**(1)[25]**, $\mathbf{U}_1$**(25)[25]**, $\mathbf{U}_1$**(5)[150]**, or $\mathbf{F}_{(x)}$**(5)[3000]**, where the function set $\mathrm{F}_{(x)}$ = **(ξ inv abs sqroot square cube quart exp ln cos sin tan tanh)**, and the terminal set is the features $\mathbf{x}_0$ thru $\mathbf{x}_{M-1}$ plus the real number constant **c** with **cbit** = 18. Our training data sets will contain 25 features, 150, and 3,000 features as specified. Our core assertion is that the enhanced algorithm will find extremely accurate champions for all of these problems and for **all similar problems** in practical time on a laptop computer.

Similar problems are easily obtained by substituting all other possibilities within $\mathbf{U}_2$**(1)[25]**, $\mathbf{U}_1$**(25)[25]**, $\mathbf{U}_1$**(5)[150]**, or $\mathbf{F}_{(x)}$**(5)[3000]**. For instance one problem in $\mathbf{U}_2$**(1)[25]** is $\mathbf{y} = \mathbf{1.687} + (\mathbf{94.183}*(\mathbf{x}_3*\mathbf{x}_2))$. By substitution, $\mathbf{y} = \mathbf{1.687} + (\mathbf{94.183}*(\mathbf{x}_3/\mathbf{x}_2))$ and $\mathbf{y} = \mathbf{1.687} + (\mathbf{94.183}*(\mathbf{x}_{23}*\mathbf{x}_{12}))$ are also in $\mathbf{U}_2$**(1)[25]**. Another problem in $\mathbf{U}_2$**(1)[25]** is $\mathbf{y} = \mathbf{-2.36} + (\mathbf{28.413}*\mathbf{ln}(\mathbf{x}_2)/\mathbf{x}_3)$. By substitution, $\mathbf{y} = \mathbf{-2.36} + (\mathbf{28.413}*\mathbf{cos}(\mathbf{x}_{12})*\mathbf{x}_6)$ and $\mathbf{y} = \mathbf{-2.36} + (\mathbf{28.413}*\mathbf{sqroot}(\mathbf{x}_{21})-\mathbf{x}_{10})$ are also in $\mathbf{U}_2$**(1)[25]**. Our core assertion is that the EA algorithm not only finds accurate solutions to the 45 test problems listed below, but also to *all other possible test problems* in $\mathbf{U}_2$**(1)[25]**, $\mathbf{U}_1$**(25)[25]**, $\mathbf{U}_1$**(5)[150]**, or $\mathbf{F}_{(x)}$**(5)[3000]**.

- **Deep problems in U_2(1)[25]**
- *..Note: these problems trained on 10,000 examples with 25 features each*
- (*T1*): y = 1.57 + (14.3*x_3)
- (*T2*): y = 3.57 + (24.33/x_3)
- (*T3*): y = 1.687 + (94.183*(x_3*x_2))
- (*T4*): y = 21.37 + (41.13*(x_3/x_2))
- (*T5*): y = -1.57 + (2.3*((x_3*x_0)*x_2))
- (*T6*): y = 9.00 + (24.983*((x_3*x_0)*(x_2*x_4)))
- (*T7*): y = -71.57 + (64.3*((x_3*x_0)/x_2))
- (*T8*): y = 5.127 + (21.3*((x_3*x_0)/(x_2*x_4)))
- (*T9*): y = 11.57 + (69.113*((x_3*x_0)/(x_2+x_4)))
- (*T10*): y = 206.23 + (14.2*((x_3*x_1)/(3.821-x_4)))
- (*T11*): y = 0.23 + (19.2*((x_3-83.519)/(93.821-x_4)))
- (*T12*): y = 0.283 + (64.2*((x_3-33.519)/(x_0-x_4)))
- (*T13*): y = -2.3 + (1.13*sin(x_2))
- (*T14*): y = 206.23 + (14.2*(exp(cos(x_4))))
- (*T15*): y = -12.3 + (2.13*cos(x_2*13.526))
- (*T16*): y = -12.3 + (2.13*tan(95.629/x_2))
- (*T17*): y = -28.3 + (92.13*tanh(x_2*x_4))
- (*T18*): y = -222.13 + (-0.13*tanh(x_2/x_4))
- (*T19*): y = -2.3 + (-6.13*sin(x_2)*x_3)
- (*T20*): y = -2.36 + (28.413*ln(x_2)/x_3)
- (*T21*): y = 21.234 + (30.13*cos(x_2)*tan(x_4))
- (*T22*): y = -2.3 + (41.93*cos(x_2)/tan(x_4))
- (*T23*): y = .913 + (62.13*ln(x_2)/square(x_4))
- **Narrow problems in U_1(2to3)[25]**
- *..Note: these problems trained on 10,000 examples with 25 features each*
- (*T24*): y = 13.3 + (80.23*x_2) + (1.13*x_3)
- (*T25*): y = 18.163 + (95.173/x_2) + (1.13/x_3)
- (*T26*): y = 22.3 + (62.13*x_2) + (9.23*sin(x_3))
- (*T27*): y = 93.43 + (71.13*tanh(x_3)) + (41.13*sin(x_3))
- (*T28*): y = 36.1 + (3.13*x_2) + (1.13*x_3) + (2.19*x_0)
- **Wide problems in U_1(5)[25]**
- *..Note: these problems trained on 10,000 examples with 25 features each*
- (*T29*): y = -9.16 + (-9.16*x_{24}*x_0) + (-19.56*x_{20}*x_{21}) + (21.87*x_{24}*x_2) + (-17.48*x_{22}*x_{23}) + (38.81*x_{23}*x_{24})
- (*T30*): y = -9.16 + (-9.16*x_{24}/x_0) + (-19.56*x_{20}/x_{21}) + (21.87*x_{24}/x_2) + (-17.48*x_{22}/x_{23}) + (38.81*x_{23}/x_{24})
- **Broad problems in $F_{(x)}$(5)[3000]**
- *..Note: these problems trained on 5,000 examples with 3,000 features each*
- *..Note: $F_{(x)}$ = noop inv abs sqroot square cube quart exp ln cos sin tan tanh*
- (*T31*): y = 50.63 + (63.6*cube(x_0)) + (66.54*cube(x_1)) + (32.95*cube(x_2)) + (4.87*cube(x_3)) + (46.49*cube(x_4))

- (*T32*): y = -9.16 + (-9.16*square(x_0)) + (-19.56*ln(x_{123})) + (21.87*exp(x_{254})) + (-17.48*x_3) + (38.81*x_{878})
- (*T33*): y = 0.0 + (1*square(x_0)) + (2*square(x_1)) + (3*square(x_2)) + (4*square(x_3)) + (5*square(x_4))
- (*T34*): y = 65.86 + (79.4*sin(x_0)) + (45.88*cos(x_1)) + (2.13*tan(x_2)) + (4.6*sin(x_3)) + (61.47*cos(x_4))
- (*T35*): y = 1.57 + (1.57/x_{923}) + (-39.34*sin(x_1)) + (2.13*x_2) + (46.59*cos(x_{932})) + (11.54*x_4)
- (*T36*): y = 50.63 + (63.6*sqroot(x_0)) + (66.54*sqroot(x_1)) + (32.95*sqroot(x_2)) + (4.87*sqroot(x_3)) + (46.49*sqroot(x_4))
- (*T37*): y = 92.25 + (53.53*square(2.3*x_0)) + (88.26*cos(x_1)) + (42.11/x_4) + (29.0*cube(x_3)) + (93.6*tanh(x_4))
- **Broad problems in U_1(5)[150]**
- *..Note: these problems trained on 10,000 examples with 150 features each*
- (*T38*): y = -9.16 + (-9.16*x_{124}*x_0) + (-19.56*x_{120}*x_{21}) + (21.87*x_{24}*x_{26}) + (-17.48*x_{122}*x_{23}) + (38.81*x_{123}*x_{24})
- (*T39*): y = -9.16 + (-9.16*x_{124}/x_0) + (-19.56*x_{20}/x_{92}) + (21.87*x_{102}/x_2) + (-17.48*x_{22}/x_{143}) + (38.81*x_{23}/x_{149})
- (*T40*): y = -9.16 + (-9.16*cos($_0$)) + (-19.56*x_{20}/x_{21}) + (21.87*square(x_{125})) + (-17.48*x_{22}/x_{23}) + (38.81*tanh(x_{24}))
- **Dense problems in U_1(25)[25]**
- *..Note: these problems trained on 10,000 examples with 25 features each*
- (*T41*): y = 50.63 + (63.6*cube(x_0)) + (66.54*square(x_1)) + (32.95*quart(x_2)) + (4.87*cube(x_3)) + (46.49*square(x_4)) + (62.85*quart(x_5)) + (90.45*cube(x_6)) + (63.28*square(x_7)) + (42.15*quart(x_8)) + (73.03*cube(x_9)) + (92.2*square(x_{10})) + (77.99*quart(x_{11})) + (56.67*cube(x_{12})) + (72.51*square(x_{13})) + (49.77*quart(x_{14})) + (56.94*cube(x_{15})) + (54.76*square(x_{16})) + (23.11*quart(x_{17})) + (56.03*cube(x_{18})) + (51.98*square(x_{19})) + (11.71*quart(x_{20})) + (33.82*cube(x_{21})) + (46.25*square(x_{22})) + (32.98*quart(x_{23})) + (36.06*cube(x_{24}))
- (*T42*): y = -9.16 + (-9.16*x_4*x_0) + (-19.56*x_0*x_1) + (21.87*x_1*x_2) + (-17.48*x_2*x_3) + (38.81*x_3*x_4) + (3.1*x_4*x_5) + (59.81*x_5*x_6) + (93.1*x_6*x_7) + (.81*x_7*x_8) + (9.21*x_8*x_9) + (-5.81*x_9*x_{10}) + (-.01*x_{10}*x_{11}) + (4.21*x_{11}*x_{12}) + (68.81*x_{12}*x_{13}) + (-8.81*x_{13}*x_{14}) + (2.11*x_{14}*x_{15}) + (-7.11*x_{15}*x_{16}) + (-.91*x_{16}*x_{17}) + (20.0*x_{17}*x_{18}) + (1.81*x_{18}*x_{19}) + (9.71*x_{19}*x_{20}) + (8.1*x_{20}*x_{21}) + (6.1*x_{21}*x_{22}) + (18.51*x_{22}*x_{23}) + (7.1*x_{23}*x_{24})
- (*T43*): y = 0.0 + (1*square(x_0)) + (2*square(x_1)) + (3*square(x_2)) + (4*square(x_3)) + (5*square(x_4)) + (6*square(x_5)) + (7*square(x_6)) + (8*square(x_7)) + (9*square(x_8)) + (10*square(x_9)) + (11*square(x_{10})) + (12*square(x_{11})) + (13*square(x_{12})) + (14*square(x_{13})) + (15*square(x_{14})) + (16*square(x_{15})) + (17*square(x_{16})) + (18*square(x_{17})) + (19*square(x_{18})) + (20*square(x_{19})) + (21*square(x_{20})) + (22*square(x_{21})) + (23*square(x_{22})) + (24*square(x_{23})) + (25*square(x_{24}))
- (*T44*): y = 65.86 + (79.4*sin(x_0)) + (45.88*cos(x_1)) + (2.13*tan(x_2)) + (4.6*sin(x_3)) + (61.47*cos(x_4)) + (30.64*tan(x_5)) + (51.95*sin(x_6)) +

$(47.83*\cos(x_7)) + (4.21*\tan(x_8)) + (37.84*\sin(x_9)) + (62.57*\cos(x_{10})) + (4.68*\tan(x_{11})) + (32.65*\sin(x_{12})) + (86.89*\cos(x_{13})) + (84.79*\tan(x_{14})) + (31.72*\sin(x_{15})) + (90.4*\cos(x_{16})) + (93.57*\tan(x_{17})) + (42.18*\sin(x_{18})) + (47.91*\cos(x_{19})) + (41.48*\tan(x_{20})) + (39.47*\sin(x_{21})) + (48.44*\cos(x_{22})) + (34.75*\tan(x_{23})) + (56.7*\sin(x_{24}))$

- (*T45*): $y = 1.57 + (1.57*x_0) + (-39.34*\sin(x_1)) + (2.13*x_2) + (46.59*(x_3/x_2)) + (11.54*x_4) + (30.64*\ln(x_5)) + (51.95*\text{abs}(x_6)) + (47.83*(x_7*x_3)) + (4.21*\text{quart}(x_8)) + (37.84*x_9) + (62.57*\text{square}(x_{10})) + (4.68*\text{sqroot}(x_{11})) + (32.65*(x_{12}/x_3)) + (86.89*x_{14}) + (84.79*\tan(x_{15})) + (31.72*\text{cube}(x_{16})) + (90.4*(x_{17}*x_{18})) + (93.57*(x_{17}/x_{16})) + (42.18*\sin(x_{18})) + (47.91*\cos(x_{19})) + (41.48*\ln(x_{20})) + (39.47*\text{square}(x_{21})) + (48.44*x_{22}) + (34.75*(x_{23}*x_{24})) + (56.7*x_{24})$

2 Training with Zero Noise

Comparing the SR performance of the baseline algorithm and the EA algorithm, on noiseless training data, using statistical best practices out-of-sample testing methodology, requires the following procedure. For each sample test problem, a matrix of independent variables is filled with random numbers between -10 and +10. Then the specified sample test problem formula is applied to produce the dependent variable. These steps will create the training data (each matrix row is a *training example* and each matrix column is a *feature*). A symbolic regression will be run on the training data to produce the champion estimator. Next a matrix of independent variables is filled with random numbers between -10 and +10. Then the specified sample test problem formula is applied to produce the dependent variable. These steps will create the testing data. The fitness score is the root mean squared error divided by the standard deviation of Y, NLSE. The estimator will be evaluated against the testing data producing the final NLSE for comparison.

The baseline algorithm and the EA algorithm will be trained on each of the 45 sample test problems for comparison. The baseline algorithm halts automatically when it achieves an extremely accurate champion on the training data. The EA algorithm algorithm halts automatically when it achieves an extremely accurate champion on the training data; but the EA algorithm also halts automatically when it has exhausted it predefined search pattern. Each algorithm will be given a maximum of 20 hours for completion, at which time, *if the SR has not already halted*, the SR run will be terminated and the best available candidate will be selected as the final estimator champion.

In each table of results, the **Test** column contains the identifier of the sample test problem (*T01 thorugh T45*). The **WFFs** column contains the number

Table 1. Baseline Accuracy Zero Noise

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	2K	0.03	0.0000	0.0000	yes
T02	2K	0.02	0.0000	0.0000	yes
T03	2K	0.03	0.0000	0.0000	yes
T04	11K	0.11	0.0000	0.0000	yes
T05	812K	9.00	0.0000	0.0000	yes
T06	1246K	20.00	0.5364	0.7727	no
T07	112K	1.29	0.0000	0.0000	yes
T08	1221K	20.00	0.0034	0.1354	no
T09	1240K	20.00	0.0484	0.9999	no
T10	1242K	20.00	0.0185	0.9999	no
T11	1117K	20.00	0.0317	0.9999	no
T12	1414K	20.00	0.0244	0.9999	no
T13	5K	0.05	0.0000	0.0000	yes
T14	9K	0.09	0.0000	0.0000	yes
T15	724K	20.00	0.8540	0.9348	no
T16	884K	20.00	0.0077	0.9999	no
T17	10K	0.10	0.0000	0.0000	yes
T18	360K	4.51	0.0000	0.0000	yes
T19	73K	0.86	0.0000	0.0000	yes
T20	356K	4.41	0.0000	0.0000	yes
T21	908K	20.00	0.0560	0.0222	no
T22	908K	20.00	0.0568	0.0602	no
T23	621K	8.21	0.0000	0.9999	no
T24	5K	0.05	0.0000	0.0000	yes
T25	77K	0.88	0.0000	0.0000	yes
T26	17K	0.18	0.0000	0.0000	yes
T27	79K	0.85	0.0000	0.0000	yes
T28	10K	0.10	0.0000	0.0000	yes
T29	870K	20.00	0.1324	0.1334	no
T30	900K	20.00	0.0290	0.0099	no
T31	900K	20.00	0.2104	0.2289	no
T32	179K	8.06	0.0000	0.0000	yes
T33	280K	20.00	0.2435	0.2398	no
T34	283K	20.00	0.2028	0.2412	no
T35	251K	20.00	0.0511	0.0540	no
T36	333K	20.00	0.4524	0.4755	no
T37	255K	11.97	0.0000	0.0000	yes
T38	275K	20.00	0.7453	0.8026	no
T39	282K	20.00	0.0403	0.9999	no
T40	249K	20.00	0.0022	0.9999	no
T41	854K	20.00	0.0455	0.0645	no
T42	978K	20.00	0.8415	0.9999	no
T43	507K	20.00	0.3838	0.8082	no
T44	517K	20.00	0.0062	0.9999	no
T45	517K	20.00	0.0024	0.9999	no

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noiseless training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.3551 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **19 absoutely accurate**)

Table 2. Extreme Accuracy Zero Noise

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	1K	0.01	0.0000	0.0000	yes
T02	1K	0.01	0.0000	0.0000	yes
T03	34K	0.13	0.0000	0.0000	yes
T04	20K	0.11	0.0000	0.0000	yes
T05	135K	0.26	0.0000	0.0000	yes
T06	243K	0.40	0.0000	0.0000	yes
T07	137K	0.29	0.0000	0.0000	yes
T08	255K	0.42	0.0000	0.0000	yes
T09	2935K	2.19	0.0000	0.0000	yes
T10	5087K	3.94	0.0000	0.0000	yes
T11	576K	0.69	0.0000	0.0000	yes
T12	198K	0.40	0.0000	0.0000	yes
T13	1K	0.01	0.0000	0.0000	yes
T14	37K	0.15	0.0000	0.0000	yes
T15	1432K	1.31	0.0000	0.0000	yes
T16	1963K	1.70	0.0000	0.0000	yes
T17	3869K	3.30	0.0000	0.0000	yes
T18	3927K	3.31	0.0000	0.0000	yes
T19	972K	1.05	0.0000	0.0000	yes
T20	644K	0.78	0.0000	0.0000	yes
T21	8268K	6.96	0.0000	0.0000	yes
T22	25365K	15.35	0.0000	0.0000	yes
T23	25675K	15.66	0.0000	0.0000	yes
T24	1K	0.01	0.0000	0.0000	yes
T25	1K	0.01	0.0000	0.0000	yes
T26	1K	0.01	0.0000	0.0000	yes
T27	1K	0.01	0.0000	0.0000	yes
T28	1K	0.01	0.0000	0.0000	yes
T29	453K	0.60	0.0000	0.0000	yes
T30	143K	0.31	0.0000	0.0000	yes
T31	113K	2.05	0.0000	0.0000	yes
T32	51K	1.10	0.0000	0.0000	yes
T33	232K	3.00	0.0000	0.0000	yes
T34	1471K	13.47	0.0000	0.0000	yes
T35	715K	7.36	0.0000	0.0000	yes
T36	139K	2.44	0.0000	0.0000	yes
T37	465K	5.02	0.0000	0.0000	yes
T38	599K	4.99	0.0000	0.0000	yes
T39	134K	1.21	0.0000	0.0000	yes
T40	255K	2.23	0.0000	0.0000	yes
T41	24K	0.38	0.0000	0.0000	yes
T42	1901K	8.25	0.0000	0.0000	yes
T43	119K	1.14	0.0000	0.0000	yes
T44	80K	0.81	0.0000	0.0000	yes
T45	216K	1.87	0.0000	0.0000	yes

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noiseless training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.0000 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **45 absolutely accurate**)

of regression candidates tested before finding a solution. The **Train-Hrs** column contains the elapsed hours spent training on the training data before finding a solution. The **Train-NLSE** column contains the fitness score of the champion on the noiseless training data. The **Test-NLSE** column contains the fitness score of the champion on the noiseless testing data. The **Absolute** column contains **yes** if the resulting champion contains a set of basis functions which are algebraically equivalent to the basis functions in the specified test problem.

For the purposes of this algorithm, ***extremely accurate*** will be defined as any champion which achieves a normalized least squares error (NLSE) of **.0001** or less on the **noiseless testing data**. In the tables of results, in this chapter, the noisless test results are listed under the **Test-NLSE** column header.

Obviously *extreme accuracy* is not the same as *absolute accuracy* and is therefore fragile under some conditions. Extreme accuracy will stop at the first estimator which achieves an NLSE of **0.0** on the noiseless training data, and *hope* that the estimator will achieve an NLSE of **.0001** or less on the testing data. Yes, an extremely accurate algorithm is guaranteed to find a perfect champion (*estimator training fitness of 0.0*) if there is one to be found; but, this perfect champion may or may not be the estimator which was used to create the testing data. For instance in the target formula *y = 1.0 + (100.0*sin(x_0)) + (.001*square(x_0))* we notice that the final term *(.0001*square(x_0))* is less significant at low ranges of x_0; but, as the absolute magnitude of x_0 increases, the final term is increasingly significant. And, this does not even cover the many issues with problematic training data ranges and poorly behaved target formulas within those ranges. For instance, creating training data in the range -1000 to 1000 for the target formula *y = 1.0 + exp(x_2*34.23)* runs into many issues where the value of **y** exceeds the range of a 64 bit IEEE real number. So as one can see the concept of *extreme acuracy* is just the beginning of the attempt to conquer the accuracy problem in SR.

For the purposes of this algorithm, ***absolutely accurate*** will be defined as any champion which contains a set of basis functions which are *algebraically equivalent* to the basis functions in the specified test problem. In the tables of results, in this chapter, the absolute accuracy results are listed under the **Absolute** column header. "Yes" indicates that the resulting champion contains a set of basis functions which are algebraically equivalent to the basis functions in the specified test problem.

As mentioned, each of the problems were trained and tested on from 25 to 3,000 features as specified using out of sample testing. The allocated maximum time to complete a test problem on our laptop environment was 20 hours, at which time training was automatically halted and the best champion was returned as the answer. However, most problems finished well ahead of that maximum time limit.

All timings quoted in these tables were performed on a Dell XPS L521X Intel i7 quad core laptop with 16Gig of RAM, and 1Tb of hard drive, manufactured in Dec 2012 (*our test machine*).

Note: testing a single regression champion is not cheap. At a minimum testing a single regression champion requires as many evaluations as there are training examples as well as performing a simple regression. At a maximum testing a single regression champion may require performing a much more expensive multiple regression.

The results in baseline **Table 1** demonstrate only intermittant accuracy on the 45 test problems. Baseline accuracy is very good with 1, 2, or 5 features in the training data. Unfortunately, Baseline accuracy decreases rapidly as the number of features in the training data increases to 25, 100, and 3000. Furthermore, there is a great deal of overfitting as evidenced by the number of test cases with good training scores and very poor testing scores.

The baseline algorithm also suffers from bloat. This is often the reason for the baseline's frequent failure to discover the absolutely accurate formula. For instance, in test problem **T19**, the correct formula is: y = *-2.3 + (-6.13*sin(x_2)*x_3)*. The baseline algorithm returns a champion of y = *-2.3000000000033-(6.13*((0.008*(x_3*125.0))*sin(x_2)))+(0.0000000000033*tanh(square(x_{23})))*. The first term, *(0.008*(x_3*125.0))*, and the last term, *(0.0000000000033*tanh(square(x_{23})))*, are bloat and will cause serious problems in range shifted data.

In such cases of overfitting, SR becomes deceptive. It produces tantalizing candidates which, from their training NLSE scores, look really exciting. Unfortunately, they fail miserably on the testing data.

Clearly the baseline testing results in **Table 1** demonstrate an opportunity for improved accuracy.

Another serious issue with the baseline algorithm is that negative results have no explicit meaning. For example, Alice runs the baseline algorithm on a large block of data for the maximum time specified. At the conculsion of the maximum specified generations, reqiring a maximum of 20 hours on our laptop, no candidate with a zero NLSE (*perfect score*) is returned. The meaning of this negative result is indeterminate, as one can argue that perhaps if Alice were to run the baseline algorithm for *a few more generations* an exact candidate would be discovered.

Significantly, the EA results in **Table 2** demonstrate extreme accuracy on the 45 test problems. This extreme accuracy is robust even in the face of problems with large number of features. More importantly, the EA algorithm achieved a perfect score on absolute accuracy. In the case of all 45 test problems, the EA algorithm was consistently absolutely accurate.

Notice the extreme search efficiency which **Table 2** demonstrates. Our assertion is that the EA algorithm is getting the same accuracy on $U_2(1)[25]$, $U_1(25)[25]$, $U_1(5)[150]$, and $F_{(x)}(5)[3000]$ as if each and every single element

of those sets were searched serially; and yet we are never testing more than a few million regression candidates.

Another very important benefit of extreme accuracy will only be fully realized when all undiscovered errors are worked out of our *informal argument for extreme accuracy* and when our informal argument is crafted into a complete, peer reviewed, well accepted, formal mathematical proof of accuracy. Once this goal is achieved, we can begin to make **modus tollens** arguments from negative results!

For example, our future Alice runs the EA algorithm on a large block of data for the maximum time specified. At the conculsion of the maximum time of 20 hours on our laptop, no candidate with a zero NLSE (*perfect score*) is returned. Refering to the published, well accepted formal mathematical proof of accuracy, Alice argues (*modus tollens*) that there exists no exact relationship bewteen X and Y anywhere within $U_2(1)[25]$, $U_1(25)[25]$, and $U_1(5)[150]$ through $F_x(5)[3000]$.

3 Training With Noisy Data

Comparing the SR performance of the baseline algorithm and the EA algorithm, on noisy training data, using statistical best practices out-of-sample testing methodology, requires the following procedure. For each sample test problem, a matrix of independent variables is filled with random numbers between -10 and +10. Then the specified sample test problem formula is applied to produce the dependent variable. Then 20% noise is added to the dependent variable according to the following formula: *y = (y*.8) + random(y*.4)*. These steps will create the training data. A symbolic regression will be run on the training data to produce the champion estimator. Next a matrix of independent variables is filled with random numbers between -10 and +10. Then the specified sample test problem formula is applied to produce the dependent variable. No noise is added to the testing dependent variable. These steps will create the testing data. The fitness score is the root mean squared error divided by the standard deviation of Y, NLSE. The estimator will be evaluated against the testing data producing the final NLSE for comparison.

The baseline algorithm and the EA algorithm will be trained on each of the 45 sample test problems for comparison. Each algorithm will be given a maximum of 20 hours for completion, at which time, *if the SR has not already halted*, the SR run will be terminated and the best available candidate will be selected as the final estimator champion.

In each table of results, the **Test** column contains the identifier of the sample test problem (*T01 thorugh T45*). The **WFFs** column contains the number of regression candidates tested before finding a solution. The **Train-Hrs** column contains the elapsed hours spent training on the training data

Table 3. Baseline Accuracy 20% Noise

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	1366K	20.00	0.0355	0.0187	yes
T02	1274K	20.00	0.0010	0.1100	no
T03	1142K	20.00	0.0991	0.9920	no
T04	1284K	20.00	0.0005	0.0536	yes
T05	1155K	20.00	0.0807	0.9999	no
T06	1201K	20.00	0.7324	0.9999	no
T07	1181K	20.00	0.0119	0.9999	no
T08	1214K	20.00	0.0017	0.9999	no
T09	1308K	20.00	0.0448	0.9999	no
T10	1210K	20.00	0.0152	0.9999	no
T11	1230K	20.00	0.0124	0.9999	no
T12	1286K	20.00	0.0189	0.9999	no
T13	1292K	20.00	0.2973	0.4052	no
T14	1242K	20.00	0.8214	0.9999	no
T15	1135K	20.00	0.8849	0.9999	no
T16	1196K	20.00	0.0370	0.9999	no
T17	1230K	20.00	0.1125	0.1339	no
T18	1057K	20.00	0.8900	0.9999	no
T19	1163K	20.00	0.1059	0.0382	no
T20	1227K	20.00	0.0002	0.1992	no
T21	1040K	20.00	0.0120	0.8882	no
T22	934K	20.00	0.0007	0.2953	no
T23	1132K	20.00	0.0001	0.9999	no
T24	1141K	20.00	0.1061	0.1734	no
T25	1054K	20.00	0.0010	0.0657	no
T26	1068K	20.00	0.1070	0.9999	no
T27	1087K	20.00	0.1555	0.9999	no
T28	1112K	20.00	0.2023	0.9999	no
T29	972K	20.00	0.6108	0.9961	no
T30	921K	20.00	0.0115	0.9999	no
T31	247K	20.00	0.1200	0.0607	no
T32	259K	20.00	0.0716	0.0148	no
T33	265K	20.00	0.3946	0.3038	no
T34	288K	20.00	0.3975	0.9999	no
T35	273K	20.00	0.3073	0.8116	no
T36	248K	20.00	0.6486	0.5438	no
T37	309K	20.00	0.1196	0.0677	no
T38	1578K	20.00	0.6697	0.6780	no
T39	1034K	20.00	0.0215	0.9952	no
T40	1505K	20.00	0.1097	0.2328	no
T41	590K	20.00	0.2731	0.2731	no
T42	694K	20.00	0.3174	0.3327	no
T43	780K	20.00	0.4356	0.9263	no
T44	800K	20.00	0.6293	0.8469	no
T45	814K	20.00	0.1069	0.0717	no

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noisy training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.6339 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **2 absolutely accurate**)

Table 4. Extreme Accuracy 20% Noise

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	26861K	19.38	0.0993	0.0059	yes
T02	26897K	19.61	0.0003	0.0000	yes
T03	26922K	19.78	0.1014	0.0133	yes
T04	26910K	18.84	0.0004	0.0000	yes
T05	26877K	18.61	0.0948	0.0000	yes
T06	26922K	18.83	0.1157	0.0000	yes
T07	26948K	18.98	0.0025	0.0000	yes
T08	26982K	19.98	0.0009	0.0000	yes
T09	26897K	20.11	0.0176	0.0000	yes
T10	26877K	19.33	0.0129	0.2877	yes
T11	26924K	20.28	0.6912	0.0747	no
T12	26879K	19.73	0.0043	0.0185	no
T13	26907K	19.99	0.3199	0.0000	yes
T14	26896K	19.96	0.8487	0.2350	no
T15	26930K	20.16	0.6712	0.1581	yes
T16	26870K	21.84	0.0119	0.4315	yes
T17	26949K	21.88	0.1227	0.0000	yes
T18	26865K	21.93	0.8763	0.9999	no
T19	26896K	22.06	0.1085	0.0000	yes
T20	26878K	22.73	0.0007	0.0668	no
T21	26983K	23.44	0.0013	0.0000	yes
T22	26886K	22.74	0.0027	0.0000	yes
T23	26918K	20.46	0.0006	0.0000	yes
T24	26936K	20.23	0.1057	0.0140	no
T25	26866K	19.83	0.0009	0.0157	no
T26	26941K	16.64	0.1074	0.0178	no
T27	26884K	19.27	0.1600	0.0000	no
T28	26908K	16.40	0.2059	0.0000	no
T29	26898K	16.33	0.1168	0.0000	yes
T30	26866K	16.13	0.0036	0.0000	yes
T31	969K	7.06	0.1084	0.0000	yes
T32	1472K	10.63	0.0739	0.0050	no
T33	1159K	8.33	0.2726	0.0000	yes
T34	1123K	8.17	0.0803	0.0000	yes
T35	1038K	7.49	0.0678	0.0000	yes
T36	1089K	8.10	0.5901	0.1083	yes
T37	1031K	7.55	0.1186	0.0124	no
T38	1189K	6.94	0.1128	0.0000	yes
T39	1279K	7.82	0.0426	0.0000	yes
T40	1299K	7.72	0.0732	0.0053	no
T41	28313K	31.2	0.1947	0.0730	no
T42	29246K	41.43	0.1002	0.0534	no
T43	28079K	28.21	0.4036	0.3682	no
T44	28605K	34.88	0.0068	0.0000	no
T45	28385K	32.31	0.0375	0.1803	no

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noisy training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.0698 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **27 absolutely accurate**)

before finding a solution. The **Train-NLSE** column contains the fitness score of the champion on the noisy training data. The **Test-NLSE** column contains the fitness score of the champion on the noiseless testing data. The **Absolute** column contains **yes** if the resulting champion contains a set of basis functions which are algebraically equivalent to the basis functions in the specified test problem.

The added training noise causes many problems. Even *absolute accuracy* is somewhat fragile under noisy training conditions. For instance in case of the target formula $y = 1.0 + (100.0*sin(x_0))$, the SR will be considered *absolutely accurate* if the resulting champion, after training, is the formula $sin(x_0)$. Clearly a champion of $sin(x_0)$ will always achieve a zero NLSE on noiseless testing data, but only *if trained on noiseless training data*. If a champion of $sin(x_0)$ is trained on noisy training data, the regression coefficients will almost always be slightly off and the champion will NOT achieve a zero NLSE even on noiseless testing data. So even an absolutely accurate champion (*containing the correct basis functions*) may not achieve extreme accuracy on noiseless testing data because the coefficients will have be slightly off due to the noise in the training data.

Since we have introduced 20% noise into the training data, we do not expect to achieve *extremely accurate* results on the noiseless testing data. However, we can hope to achieve *highly accurate* results on the testing data. For the purposes of this chapter, ***highly accurate*** will be defined as any champion which achieves a normalized least squares error (NLSE) of **.2** or less on the **noiseless testing data**. In the tables of results, in this chapter, the noisless test results are listed under the **Test-NLSE** column header.

The random noise added is normally distributed and symmetric (*as normally distributed as the* **random** *function can achieve*). The study of asymmetric noise and non-normally distributed noise will be left to another paper.

The results in baseline **Table 3** demonstrate only very intermittant accuracy on the 45 test problems. Baseline accuracy is fragile in the face of training noise. High accuracy on the noiseless testing data is infrequently achieved in 12 of the 45 test problems. Absolute accuracy on the noiseless testing data is rarely achieved in 2 of the 45 test problems. There is a great deal of overfitting as evidenced by the number of test cases with good training scores and very poor testing scores. Furthermore, there is a great deal of bloat which is why absolute accuracy is rarely achieved (*i.e. the baseline algorithm rarely discovers the correct target formula*).

Significantly, the EA results in **Table 4** consistantly demonstrate high accuracy in 40 of the 45 test problems. Noteably, the EA algorithm does achieve frequent absolute accuracy, even in the face of the noisy training data, in 27 of the 45 test problems. This absolute accuracy is robust even in the face of problems with large number of features (*i.e. the EA algorithm frequently discovers the correct target formula*).

Notice the EA's failure to achieve high accuracy in **TestCaseT10**. Even though the EA discovered the absolute accurate basis function, the noisy training data caused the coefficients to be seriously skewed. Additionally, the EA's problem with absolute accuracy in **TestCaseT12** is a case in point. Noteably, the EA algorithm actually does discover the absolute answer; but, on the noisy training data, the final fitness score of the correct answer is worse than the final fitness score of a multivariable formula (*containing the correct formula*). Faced with this better fitness score, the EA chooses the incorrect answer as its primary choice and the correct answer as a secondary choice. The EA has no way of descerning that the added noise has so seriously altered the training landscape.

Nevertheless, even with all these issues, the EA algorithm achieves a level of acuracy and search efficiency which raises SR to new level of performance on noisy training data.

4 Noisy Training With Range Shifting Testing

Comparing the SR performance of the baseline algorithm and the EA algorithm, on noisy training data with range shifted testing data, using statistical best practices out-of-sample testing methodology, requires the following procedure. For each sample test problem, a matrix of independent variables is filled with random numbers between 0 and 1. Then the specified sample test problem formula is applied to produce the dependent variable. Then 20% noise is added to the dependent variable according to the following formula: $y = (y*.8) + random(y*.4)$. These steps will create the training data. A symbolic regression will be run on the training data to produce the champion estimator. Next a matrix of independent variables is filled with random numbers between -1 and 0. Then the specified sample test problem formula is applied to produce the dependent variable. No noise is added to the testing dependent variable. These steps will create the testing data. The fitness score is the root mean squared error divided by the standard deviation of Y, NLSE. The estimator will be evaluated against the testing data producing the final NLSE for comparison.

Notice the range shifted testing data. All training is performed on data between 0 and 1. The SR has never seen a negative number. Furthermore, 20% noise is added to the dependent variable during training. Finally, the testing data is in the range -1 to 0. These are mostly negative numbers which the SR has never seen during training.

The baseline algorithm and the EA algorithm will be trained on each of the 45 sample test problems for comparison. Each algorithm will be given a maximum of 20 hours for completion, at which time, *if the SR has not already*

Table 5. Baseline Accuracy Range Shifting

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	1666K	20.00	0.2314	0.0251	yes
T02	1742K	20.00	0.0356	0.0007	no
T03	1675K	20.00	0.1467	0.0393	no
T04	1757K	20.00	0.0413	0.0003	no
T05	1491K	20.00	0.3619	0.0889	no
T06	1785K	20.00	0.5413	0.4399	no
T07	1896K	20.00	0.0377	0.0579	no
T08	1832K	20.00	0.0336	0.1600	no
T09	1619K	20.00	0.3800	0.9998	no
T10	1655K	20.00	0.8966	0.9998	no
T11	1765K	20.00	0.8836	0.9999	no
T12	1653K	20.00	0.0017	0.0106	no
T13	1788K	20.00	0.5727	0.0824	no
T14	1808K	20.00	0.8814	0.7037	no
T15	1857K	20.00	0.6017	0.0477	no
T16	1426K	20.00	0.0117	0.9999	no
T17	1749K	20.00	0.1115	0.0540	no
T18	1681K	20.00	0.8853	0.9999	no
T19	1770K	20.00	0.3066	0.5472	no
T20	1381K	20.00	0.0011	0.0006	no
T21	1811K	20.00	0.3383	0.1484	no
T22	1838K	20.00	0.0453	0.1881	no
T23	1732K	20.00	0.0000	0.0000	no
T24	1831K	20.00	0.2500	0.0324	no
T25	1884K	20.00	0.0428	0.4828	no
T26	1686K	20.00	0.3089	0.1358	no
T27	1613K	20.00	0.4922	0.0715	no
T28	1468K	20.00	0.8754	0.8360	no
T29	1726K	20.00	0.2747	0.3448	no
T30	1638K	20.00	0.0070	0.5507	no
T31	448K	20.00	0.3765	0.3907	no
T32	453K	20.00	0.2953	0.2615	no
T33	462K	20.00	0.2783	0.1566	no
T34	387K	20.00	0.6321	0.1958	no
T35	534K	20.00	0.1813	0.0617	no
T36	460K	20.00	0.6561	0.2358	no
T37	518K	20.00	0.0974	0.0124	no
T38	1759K	20.00	0.3503	0.4808	no
T39	1734K	20.00	0.0224	0.2714	no
T40	1633K	20.00	0.0124	0.2066	no
T41	571K	20.00	0.4867	0.3647	no
T42	597K	20.00	0.3211	0.3328	no
T43	599K	20.00	0.4478	0.2434	no
T44	635K	20.00	0.6385	0.8469	no
T45	741K	20.00	0.0514	0.9999	no

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noisy training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.3357 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **1 absolutely accurate**)

Table 6. Extreme Accuracy Range Shifting

Test	*WFFs*	*Train-Hrs*	*Train-NLSE*	*Test-NLSE*	*Absolute*
T01	26912K	13.23	0.2308	0.0000	yes
T02	26832K	12.71	0.0326	0.0000	yes
T03	26868K	13.05	0.1586	0.0000	yes
T04	26937K	13.42	0.05426	0.0000	yes
T05	26820K	12.7	0.3626	0.0000	yes
T06	26884K	13.03	0.4908	0.0000	yes
T07	26885K	12.96	0.0527	0.0616	no
T08	26908K	12.21	0.0621	0.0000	yes
T09	26880K	11.85	0.1686	0.0000	yes
T10	26870K	13.04	0.8927	0.9999	no
T11	26862K	12.98	0.8969	0.9999	no
T12	26865K	12.94	0.0017	0.0008	no
T13	26905K	13.27	0.5626	0.0716	no
T14	26914K	13.30	0.8820	0.1826	no
T15	26836K	12.21	0.6426	0.0158	yes
T16	26859K	12.80	0.0198	0.9999	no
T17	26861K	12.96	0.1223	0.0000	yes
T18	26832K	12.66	0.9050	0.9999	no
T19	26895K	13.09	0.3130	0.0635	no
T20	26981K	13.61	0.0013	0.1339	no
T21	26885K	13.06	0.3445	0.0808	no
T22	26956K	13.42	0.0449	0.0172	no
T23	26838K	11.91	0.0002	0.0000	yes
T24	26857K	11.87	0.2541	0.0140	no
T25	26971K	13.49	0.0425	0.0031	no
T26	26984K	13.56	0.3175	0.1258	no
T27	26892K	13.06	0.4942	0.0942	no
T28	26871K	12.88	0.8860	0.9999	no
T29	26896K	13.03	0.1359	0.0000	yes
T30	26882K	12.89	0.0036	0.0000	yes
T31	969K	7.10	0.3842	0.0609	no
T32	969K	7.08	0.2776	0.0287	no
T33	975K	7.09	0.2830	0.0139	no
T34	764K	7.51	0.6563	0.0000	no
T35	1094K	8.02	0.1594	0.1042	no
T36	723K	7.01	0.6640	0.1870	no
T37	1121K	8.15	0.0992	0.0073	no
T38	1560K	8.97	0.2616	0.2316	no
T39	1228K	7.46	0.0069	0.0000	yes
T40	1995K	14.80	0.0052	0.0066	no
T41	28632K	35.34	0.4616	0.9392	no
T42	28700K	37.84	0.2008	0.1363	no
T43	27144K	17.32	0.3983	0.2914	no
T44	27670K	24.78	0.5973	0.8458	no
T45	28694K	37.34	0.0532	0.0020	no

(**Note1***: the number of regression candidates tested before finding a solution is listed in the Well Formed Formulas (WFFs) column*) (**Note2***: the elapsed hours spent training on the training data is listed in the (Train-Hrs) column*) (**Note3***: the fitness score of the champion on the noisy training data is listed in the (Train-NLSE) column*) (**Note4***: the fitness score of the champion on the noiseless testing data is listed in the (Test-NLSE) column with* **.1937 average fitness**) (**Note5***: the absolute accuracy of the SR is given in the (Absolute) column with* **14 absolutely accurate**)

halted, the SR run will be terminated and the best available candidate will be selected as the final estimator champion.

In each table of results, the **Test** column contains the identifier of the sample test problem (*T01 thorugh T45*). The **WFFs** column contains the number of regression candidates tested before finding a solution. The **Train-Hrs** column contains the elapsed hours spent training on the training data before finding a solution. The **Train-NLSE** column contains the fitness score of the champion on the noisy training data. The **Test-NLSE** column contains the fitness score of the champion on the noiseless testing data. The **Absolute** column contains **yes** if the resulting champion contains a set of basis functions which are algebraically equivalent to the basis functions in the specified test problem.

For the purposes of this chapter, ***extremely accurate*** will be defined as any champion which achieves a normalized least squares error (NLSE) of **.0001** or less on the **noiseless testing data**. In the table of results, at the conclusion of this chapter, the noisless test results are listed under the **Test-NLSE** column header.

Obviously *extreme accuracy* is not the same as *absolute accuracy* and is therefore fragile under some conditions. Extreme accuracy will stop at the first estimator which achieves an NLSE of **0.0** on the noiseless training data, and *hope* that the estimator will achieve an NLSE of **.0001** or less on the testing data. Yes, an extremely accurate algorithm is guaranteed to find a perfect champion (*estimator training fitness of 0.0*) if there is one to be found; but, this perfect champion may or may not be the estimator which was used to create the testing data. For instance in the target formula $y = 1.0 + (100.0*sin(x_0)) + (.001*square(x_0))$ we notice that the final term $(.0001*square(x_0))$ is less significant at low ranges of x_0; but, as the absolute magnitude of x_0 increases, the final term is increasingly significant. And, this does not even cover the many issues with problematic training data ranges and poorly behaved target formulas within those ranges. For instance, creating training data in the range -1000 to 1000 for the target formula $y = 1.0 + exp(x_2*34.23)$ runs into many issues where the value of **y** exceeds the range of a 64 bit IEEE real number. So as one can see the concept of *extreme acuracy* is just the beginning of the attempt to conquer the accuracy problem in SR.

Furthermore even *absolute accuracy* is somewhat fragile under noisy training conditions. For instance in case of the target formula $y = 1.0 + (100.0*sin(x_0))$, the SR will be considered *absolutely accurate* if the resulting champion, after training, is the formula $sin(x_0)$. Clearly a champion of $sin(x_0)$ will always achieve a zero NLSE on noiseless testing data, but only *if trained on noiseless training data*. If a champion of $sin(x_0)$ is trained on noisy training data, the regression coefficients will almost always be slightly off and the champion will NOT achieve a zero NLSE even on noiseless testing data. So even absolute accuracy is a tricky proposition with noisy training data.

As mentioned, each of the problems were trained and tested on from 25 to 3,000 features as specified using out of sample testing. The allocated maximum time to complete a test problem on our laptop environment was 20 hours, at which time training was automatically halted and the best champion was returned as the answer. However, most problems finished well ahead of that maximum time limit.

All timings quoted in these tables were performed on a Dell XPS L521X Intel i7 quad core laptop with 16Gig of RAM, and 1Tb of hard drive, manufactured in Dec 2012 (*our test machine*).

Note: testing a single regression champion is not cheap. At a minimum testing a single regression champion requires as many evaluations as there are training examples as well as performing a simple regression. At a maximum testing a single regression champion may require performing a much more expensive multiple regression.

The results in baseline **Table 1** demonstrate only intermittant accuracy on the 45 test problems. Baseline accuracy is very good with 1, 2, or 5 features in the training data. Unfortunately, Baseline accuracy decreases rapidly as the number of features in the training data increases to 25, 150, and 3000. Furthermore, there is a great deal of overfitting as evidenced by the number of test cases with good training scores and very poor testing scores.

In such cases of overfitting, SR becomes deceptive. It produces tantalizing candidates which, from their training NLSE scores, look really exciting. Unfortunately, they fail miserably on the testing data.

Clearly the baseline testing results in **Table 1** demonstrate an opportunity for improved accuracy.

Another serious issue with the baseline algorithm is that negative results have no explicit meaning. For example, Alice runs the baseline algorithm on a large block of data for the maximum time specified. At the conculsion of the maximum specified generations, reqiring a maximum of 20 hours on our laptop, no candidate with a zero NLSE (*perfect score*) is returned. The meaning of this negative result is indeterminate, as one can argue that perhaps if Alice were to run the baseline algorithm for *a few more generations* an exact candidate would be discovered.

Significantly, the EA results in **Table 2** demonstrate extreme accuracy on the 45 test problems. This extreme accuracy is robust even in the face of problems with large number of features.

Notice the extreme search efficiency which **Table 2** demonstrates. Our assertion is that the EA algorithm is getting the same accuracy on $U_2(1)[25]$, $U_1(25)[25]$, $U_1(5)[150]$, and $F_{(x)}(5)[3000]$ as if each and every single element of those sets were searched serially; and yet we are never testing more than a few million regression candidates.

Another very important benefit of extreme accuracy will only be fully realized when all undiscovered errors are worked out of our *informal argument for extreme accuracy* and when our informal argument is crafted into a complete, peer reviewed, well accepted, formal mathematical proof of accuracy. Once this goal is achieved, we can begin to make **modus tollens** arguments from negative results!

For example, our future Alice runs the EA algorithm on a large block of data for the maximum time specified. At the conculsion of the maximum time of 20 hours on our laptop, no candidate with a zero NLSE (*perfect score*) is returned. Refering to the published, well accepted formal mathematical proof of accuracy, Alice argues (*modus tollens*) that there exists no exact relationship bewteen X and Y anywhere within $U_2(1)[25]$, $U_1(25)[25]$, and $U_1(5)[150]$ through $F_x(5)[3000]$.

5 Conclusion

In a previous paper (Korns 2011), significant accuracy issues were identified for state of the art SR systems. It is now obvious that these SR accuracy issues are due primarily to the poor surface conditions of specific subsets of the problem space. For instance, if the problem space is exceedingly choppy with little monotonicity or flat with the exception of a single point with fitness advantage, then no amount of fiddling with evolutionary parameters will address the core issue.

In (Korns 2013), an EA algorithm was introduced with an informal argument asserting extreme accuracy in a number of noiseless test problems. This enhanced algorithm contains a search language and an *informal argument*, suggesting a priori, that extreme accuracy will be achieved on any single isolated problem within a broad class of basic SR problems. In (Korns 2014), the EA algorithm was enhanced to include extreme accuracy on noiseless large feature test problems.

In this paper we test the enhanced EA algorithm measuring levels of extreme accuracy on problems with noisy training data, and with range shifted testing data. The results support the view that the persuit if high accuracy algorithms in noiseless training data also conveys distinct and measurable advantages with noisy training data and and range shifted testing data. In fact, for both noiseless training data and when trained on noisy training data, then tested on range shifted testing data, the enhanced EA algorithm is measurably faster and more accuracte than the baseline algorithm. This places the Extreme Accuracy algorithm in a class by itself.

The new EA algorithm introduces a hybrid view of SR in which advanced evolutionary methods are deployed in the extremely large spaces where serial search is impractical, and in which the intractable smaller spaces are first identified and then attacked either serially or with mathematical treatments.

All academics and SR researchers are heartily invited into this newly opened *playground*, as a plethora of intellectual work awaits. Increasing SR's demonstrable range of extreme accuracy will require that new intractable subspaces be identified and that new mathematical treatments be devised.

Future research must explore the possibility of developing an Extreme Accuracy algorithm for the related field of symbolic multinomial classification.

Finally, to the extent that the reasoning in this *informal argument*, of extreme accuracy, gain academic and commercial acceptance, a climate of *belief* in SR can be created wherein SR is increasingly seen as a "**must have**" tool in the scientific arsenal.

Truely knowing the strength's and weaknesses of our tools is an essential step in gaining trust in their use.

References

1. Gregory S Hornby (2006). Age-Layered Population Structure For reducing the Problem of Premature Convergence, in *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation.* ACM Press, New York.
2. Michael Korns (2010). Abstract Expression Grammar Symbolic Regression, in *Genetic Programming Theory and Practice VIII.* Springer, New York. Kaufmann Publishers, San Francisco California.
3. Michael Korns (2011). Accuracy in Symbolic Regression, in *Genetic Programming Theory and Practice IX.* Springer, New York. Kaufmann Publishers, San Francisco California.
4. Michael Korns (2012). A Baseline Symbolic Regression Algorithm, in *Genetic Programming Theory and Practice X.* Springer, New York. Kaufmann Publishers, San Francisco California.
5. Michael Korns (2013). Extreme Accuracy in Symbolic Regression, in *Genetic Programming Theory and Practice XI.* Springer, New York. Kaufmann Publishers, San Francisco California.
6. Michael Korns (2014). Extremely Accurate Symbolic Regression for Large Feature Problems, in *Genetic Programming Theory and Practice XII.* Springer, New York. Kaufmann Publishers, San Francisco California.
7. Mark Kotanchek, Guido Smits, and Ekaterina Vladislavleva (2008). Trustable Symbolic Regression Models: Using Ensambles, Interval Arithmetic and Pareto Fronts to Develop Robust and Trust-Aware Models, in *Genetic Programming Theory and Practice V.* Springer, New York.
8. John R Koza (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge Massachusetts.
9. John R Koza (1994). Genetic Programming II: Automatic Discovery of Reusable Programs. The MIT Press, Cambridge Massachusetts.
10. John R Koza, Forrest H Bennett III, David Andre, Martin A Keane (1999). Genetic Programming III: Darwinian Invention and Problem Solving. Morgan
11. Trent McConaghy, (2011). FFX: Fastm Scalable, Deterministic Symbolic Regression Technology, in *Genetic Programming Theory and Practice IX.* Springer, New York.
12. J.A., Nelder, and R. W. Wedderburn (1972). Generalized linear Models, in *Journal of the Royal Statistical Society*, Series A, General, 135:370-384.
13. Poli, Riccardo, McPhee, Nicholas, Vanneshi, Leonardo, (2009). Analysis of the Effects of Elitism on Bloat in Linear and Tree-based Genetic Programming, in *Genetic Programming Theory and Practice VI.* Springer, New York.
14. Guido Smits, and Mark Kotanchek (2005). Pareto-Front Exploitation in Symbolic Regression, in *Genetic Programming Theory and Practice II.* Springer, New York.
15. Michael Schmidt, Hod Lipson (2010). Age-Fitness Pareto Optimization, in *Genetic Programming Theory and Practice VI.* Springer, New York.