Genetic Programming Symbolic Classification: A Study

Michael F. Korns
Lantern Credit LLC
2240 Village Walk Drive Suite 2305
Henderson, Nevada 89052
mkorns@korns.com

ABSTRACT

While Symbolic Regression (SR) is a well-known offshoot of Genetic Programming, Symbolic Classification (SC), by comparison, has received only meager attention. Clearly, regression is only half of the solution. Classification also plays an important role in any well rounded predictive analysis tool kit. In several recent papers, SR algorithms are developed which move SR into the ranks of extreme accuracy [7][14]. In an additional set of papers algorithms are developed designed to push SC to the level of basic classification accuracy competitive with existing commercially available classification tools [1][10][15].

This paper is a simple study of four proposed SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison. The four SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm [1]; the MDC algorithm [9], and Linear Discriminant Analysis (LDA) [15]. The five commercially available classification algorithms are available in the KNIME system [16], and are as follows: Decision Tree Learner (DTL); Gradient Boosted Trees Learner (GBTL); Multiple Layer Perceptron Learner (MLP); Random Forest Learner (RFL); and Tree Ensemble Learner (TEL).

A set of ten artificial classification problems are constructed with no noise. The simple formulas for these ten artificial problems are listed herein. The problems vary from linear to nonlinear multimodal and from 25 to 1000 columns. All problems have 5,000 training points and a separate 5,000 testing points. The scores, on the out of sample testing data, for each of the nine classification algorithms are published herein.

Keywords

Symbolic Classification, Genetic Programming, Linear Discriminant Analysis.

1. Introduction

Symbolic Regression (SR) is a well-known offshoot of Genetic Programming; however, Symbolic Classification (SC) by comparison, has received relatively little attention. While regression is important, it is only half of the solution. Classification also plays an important role in any well rounded predictive analysis tool kit. Several recent papers develop algorithms which move SR into the ranks of extreme accuracy [7][14]. Additionally several papers develop algorithms designed to raise SC accuracy to the level of basically competitive with existing commercially available classification tools [1][9][10][15].

This paper is a simple study of four proposed SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison. The four SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm [1]; the MDC algorithm [9], and Linear Discriminant Analysis (LDA) [15]. The five commercially available classification algorithms are available in the KNIME system [16], and are as follows: Decision Tree Learner (DTL); Gradient Boosted Trees Learner (GBTL); Multiple Layer Perceptron Learner (MLP); Random Forest Learner (RFL); and Tree Ensemble Learner (TEL).

A set of ten artificial classification problems are constructed with no noise such that absolutely accurate classifications are theoretically possible. The discriminant formulas for these ten artificial problems are listed herein. The problems vary from linear to nonlinear multimodal and from 25 to 1000 columns such that each classification algorithm will be stressed on well understood problems from the simple to the very difficult. All problems have 5,000 training points and a separate 5,000 testing points. The scores on the out of sample testing data, for each of the nine classification algorithms are published herein.

No assertion is made that these four genetic programming SC algorithms are the best in the literature. In fact we know of an additional enhanced algorithm, *which we have not had time to implement for this study*, M_3GP [10]. No assertion is made that the five KNIME classification algorithms are the best commercially available, only that KNIME is a trusted component of Lantern Credit predictive analytics. This study is simply meant to provide one reference point for how far genetic programming symbolic classification has improved relative to a set of reasonable commercially available classification algorithms.

Each of the four genetic programming SC algorithms is briefly explained further in this paper as follows.

2. AMAXSC In Brief

The simplest naïve genetic programming approach to multiclass classification is arguably a standard genetic programming approach, such as a modification of the baseline algorithm [4], using the **argmax** function to classify as follows,

- **E1**: $y = \text{argmax}(gp_{\mathbf{1}}, gp_{\mathbf{2}},\ldots, gp_{\mathbf{C}})$ where **C** is the number of classes

Each $\mathbf{gp_k}$ represents a separate discriminant function evolved via standard genetic programming. The argmax() function chooses the class (1 to C) which has the highest value, and is strongly related to the Bayesian probability that the training point belongs to the cth class. No other enhancements are needed other than the standard argmax() function and a slightly modified genetic programming system – modified to evolve one formula for each class instead of the usual single formula.

3. MDC In Brief

The Multilayer Discriminant Classification (MDC) algorithm is an evolutionary approach to enhancing the simple AMAXSC algorithm.

- **E2**: $y = \text{argmax}(w_{\mathbf{10}}+(w_{\mathbf{11}}*gp_{\mathbf{1}}),\ w_{\mathbf{20}}+(w_{\mathbf{21}}*gp_{\mathbf{2}}),\ldots,\ w_{\mathbf{C0}}+(w_{\mathbf{C1}}*gp_{\mathbf{C}}))$ where **C** is the number of classes, the $\mathbf{gp}_k$ are GP evolved formulas, and the $\mathbf{w_{ij}}$ are real weight coefficients (*there are 2C weights*).

Each $\mathbf{gp_k}$ represents a separate discriminant function evolved via standard genetic programming. The argmax() function chooses the class (1 to C) which has the highest value, and is strongly related to the Bayesian probability that the training point belongs to the cth class. Given a set of GP evolved discriminant formulas $\{gp_1, gp_2,\ldots, gp_C\}$, the objective of the MDC algorithm is to optimize the choice of coefficient weights $\{w_{10}, w_{11}, w_{20}, w_{21},\ldots,w_{C0}, w_{C1}\}$ so that equation **E2** is optimized for all X and Y.

The first step in the MDC algorithm is to perform a Partial Bipolar Regression on each discriminant entry i.e. $w_{\mathbf{k0}}+(w_{\mathbf{k1}}*gp_{\mathbf{k}}) = Y_k + e$. This produces starting weights for $w_{\mathbf{k0}}$ and $w_{\mathbf{k1}}$ which are not very good but are much better than random. The second step in the MDC algorithm is to run a Modified Sequential Minimization on selected discriminant entries. This produces much better weight candidates for all discriminant functions, but still not perfect. Finally, the MDC algorithm employs the Bees Algorithm to fully optimize the coefficient weights.

The MDC algorithm is discussed in much greater detail in [9].

4. M_2GP In Brief

The M_2GP algorithm is described in detail in [1]. Briefly the M_2GP algorithm generates a D-dimensional GP tree instead of a 1-dimensional GP tree. Assuming that there are C classes, the algorithm attempts to minimize the Mahalanobis distance between the nth training point and the centroid of the kth class. The basic training algorithm is as follows.

Algorithm A1: M_2GP Training

```
Input: X, Y, D – where X is an MxN real matrix, Y is an N Vector, D is a scalar
For g from 1 to G do
Generate: $F = \{f_1, f_2,\ldots,f_D\}$ set of D solutions
Evaluate: $Z_s = Eval(f_s(X))$ for s from 1 to D – a D-dimensional point
Cluster: $Z^k$ in Z for all k from 1 to C – group all the Z which belong to each class
For k from 1 to C do
  $C^k = covar(Z^k)$ – a d by d covariance matrix for each class
  $W^k = centroid(Z^k)$ – a 1 by D centroid vector
  $D_k(X_n) = sqrt(\ (Z_n-W^k)*(C^k)^{-1}*(Z_n-W^k)^T\ )$ – for n from 1 to N (the number of training points)
For n from 1 to N do $EY_n = argmin(D_1(X_n),D_2(X_n),\ldots,D_C(X_n))$
For n from 1 to N do $E_n = 1$ IFF $EY_n <> Y_n$, 0 otherwise
Minimize average(EY)
Return F, C, M
```

The M_2GP algorithm is discussed in much greater detail in [1].

5. LDA Background

Linear Discriminant Analysis (LDA) is a generalization of Fischer's linear discriminant, which is a method to find a linear combination of features which best separates K classes of training points [11], [12], [13]. LDA is used extensively in Statistics, Machine Learning, and Pattern Recognition.

Similar to the arguments leading up to the M_2GP algorithm [1], we argue that any symbolic regression system can be converted into a symbolic classification system. In this paper we start with the baseline algorithm published in [4]. Our baseline SR system inputs an N by M matrix of independent training points, **X**, and an N vector of dependent values, **Y**. The SR system outputs a predictor function, **F**(X) ~ Y where F is the best least squares estimator for Y which the SR system could find in the allotted training time. The format of F is important, and consists of one or more basis functions $\mathbf{Bf_b}$ with regression constants $\mathbf{c_b}$. There are always B basis functions and B+1 coefficients. The following is the format of F.

- **E3**: $y = \mathbf{c}_0+\mathbf{c}_1*Bf_1+\mathbf{c}_2*Bf_2+\ldots+\mathbf{c}_B*Bf_B$

There are from 1 to B basis functions with 2 to B+1 real number coefficients. Each basis function is an algebraic combination of operators on the M features of X, such that $Bf_b(X)$ is a real number. The following is a typical example of an SR produced predictor, F(X).

- **E4**: $y = 2.3+.9*\cos(x_3)+7.1*x_6+5.34*(x_4/\tan(x_8))$

The coefficients $\mathbf{c}_0$ to $\mathbf{c}_B$ play an important role in minimizing the least squares error fit of F with Y. The coefficients can be evolved incrementally, but most industrial strength SR systems identify the optimal coefficients via an assisted fitness training technique. In the baseline SR algorithm this assisted fitness training technique is simple linear regression (*B=1*) or multiple linear regression (*B>1*).

In symbolic classification problems the N by M matrix of independent training points, X, is unchanged. However, The N vector of dependent values contains only categorical unordered values between 1 and K. Furthermore the *least squares error* fitness measure (LSE) is replaced with *classification error percent* (CEP) fitness. Therefore we cannot use regression for assisted fitness training in our new SC system. Instead, we can use LDA as an assisted fitness training technique in our new SC system.

Our new SC system now outputs not one predictor function, but instead outputs K predictor functions (*one for each class*). These functions are called discriminants, $D_k(X) \sim Y_k$, and there is one discriminant function for each class. The format of the SC's discriminant function output is always as follows.

- **E5**: $y = \text{argmax}(D_1,D_2, \ldots ,D_K)$

The *argmax* function returns the class index for the largest valued discriminant function. For instance if $D_i = \max(D_1,D_2, \ldots ,D_K)$, then $i = \text{argmax}(D_1,D_2, \ldots ,D_K)$.

A central aspect of LDA is that each discriminant function is a linear variation of every other discriminant function and reminiscent of the multiple basis function estimators output by the SR system. For instance if the GP symbolic classification system produces a candidate with B basis functions, then each discriminant function has the following format.

$D_0 = \mathbf{c}_{00}+\mathbf{c}_{01}*Bf_1+\mathbf{c}_{02}*Bf_2+\ldots+\mathbf{c}_{0B}*Bf_B$

$D_1 = \mathbf{c}_{10}+\mathbf{c}_{11}*Bf_1+\mathbf{c}_{12}*Bf_2+\ldots+\mathbf{c}_{1B}*Bf_B$

$D_k = \mathbf{c}_{k0}+\mathbf{c}_{k1}*Bf_1+\mathbf{c}_{k2}*Bf_2+\ldots+\mathbf{c}_{kB}*Bf_B$

The K*(B+1) coefficients are selected so that the ith discriminant function has the highest value when the y = i (*i.e. the class is i*). The technique for selecting these optimized coefficients $\mathbf{c}_{00}$ to $\mathbf{c}_{KB}$ is called linear discriminant analysis and in the following section we will present the Bayesian formulas for these discriminant functions.

6. LDA Matrix Math

We use Bayes rule to minimize the *classification error percent* (CEP) by assigning a training point $X_{[n]}$ to the class k if the probability of $X_{[n]}$ belonging to class k, $P(k|X_{[n]})$, is higher than the probability for all other classes as follows.

- **E6**: $EY_{[n]} = k$, iff $P(k|X_{[n]}) \geq P(j|X_{[n]})$ for all $1 \leq j \geq K$

The CEP is computed as follows.

- **E6**: $CEP = \sum(EY_{[n]} \neq Y_{[n]} \,|\, \text{for all } n) / N$

Therefore, each discriminant function D_k acts a Bayesian estimated percent probability of class membership in the formula.

- **E7**: $y = \text{argmax}(D_1,D_2, \ldots ,D_K)$

The technique of LDA makes three assumptions, (a) that each class has multivariate Normal distribution, (b) that each class covariance is equal, and (c) that the class covariance matrix is nonsingular. Once these assumptions are made, the mathematical formula for the optimal Bayesian discriminant function is as follows.

- **E8**: $D_k(X_n) = \mu_k(C_k)^{-1}(X_n)^T - 0.5\mu_k(C_k)^{-1}(\mu_k)^T + \ln(P_k)$

Where X_n is the nth training point, μ_k is the mean vector for the kth class, $(C_k)^{-1}$ is inverse of the covariance matrix for the kth class, $(X_n)^T$ is the transpose of the nth training point, $(\mu_k)^T$ is the transpose of the mean vector for kth class, and $\ln(P_k)$ is the natural logarithm of the naïve probability that any training point will belong to the kth class.

In the following section we will present step by step implementation guidelines for LDA assisted fitness training in our new extended baseline SC system, as indicated by the above Bayesian formula for $D_k(X_n)$.

7. LDA Assisted Fitness Implementation

The baseline SR system [4] attempts to score thousands to millions of regression candidates in a run. These are presented for scoring via the fitness function which returns the *least squares error* (LSE) fitness measure.

- **E9**: lse = fitness(X,Y,$Bf_1,\ldots,Bf_B,\mathbf{c}_0,\ldots,\mathbf{c}_B$)

The coefficients $\mathbf{c}_0,\ldots,\mathbf{c}_B$ can be taken as is, and the simple LSE returned. However, most industrial strength SR systems use regression as an assisted fitness technique to supply optimal values for the coefficients before returning the LSE fitness measure. This greatly speeds up accuracy and allows the SR to concentrate all of its algorithmic resources toward the evolution of an optimal set of basis functions $Bf_1,\ldots,Bf_B$.

Converting to a baseline symbolic classification system will require returning the *classification error percent* (CEP) fitness measure, *which is defined as the count of erroneous classifications divided by the size of Y,* and extending the coefficients to allow for linear discriminant analysis as follows.

- **E10**: cep = fitness(X,Y,$Bf_1,\ldots,Bf_B,\mathbf{c}_{00},\ldots,\mathbf{c}_{KB}$)

Of course the coefficients $\mathbf{c}_{00},\ldots,\mathbf{c}_{KB}$ can be taken as is, and the simple CEP returned. However, our new baseline SC system will use LDA as an assisted fitness technique to supply optimal values for the coefficients before returning the CEP fitness measure. This greatly speeds up accuracy and allows the SR to concentrate all of its algorithmic resources toward the evolution of an optimal set of basis functions $Bf_1,\ldots,Bf_B$.

7.1 Converting to basis space

The first task of our new SC fitness function must be to convert from N by M feature space, X, into N by B basis space XB. Basis space is the training matrix created by assigning basis function conversions to each of the B points in XB as follows.

- **E11**: $XB_{[n][b]} = Bf_b(X_{[n]})$

So for each row n of our N by M input feature space training matrix, ($X_{[n]}$), we apply all B basis functions, yielding the B points of our basis space training matrix for row n, ($XB_{[n][b]}$). Our new training matrix is the N by B basis space matrix, XB.

7.2 Class Clusters and Centroids

Next we must compute the K class cluster matrices for each of the K classes as follows

- **E12**: **Define** $CX_{[k]}$ *where* $XB_{[n]} \in CX_{[k]}$ iff $Y_{[n]} = k$

Each matrix $CX_{[k]}$ contains only those training rows of XB which belong to the class k. We also compute the simple Bayesian probability of membership in each class cluster matrix as follows.

- **E13**: $P_{[k]}$ = **length**($CX_{[k]}$) / N

Next we compute the K cluster mean vectors, each of which is a vector of length B containing the average value in each of the B columns of each of the K class cluster matrices, CX, as follows.

- **E14**: $\mu_{[k][b]}$ = column mean of the bth column of $CX_{[k]}$

We next compute the class centroid matrices for each of the K classes, which are simply the mean adjusted class clusters as follows

- **E15**: $CXU_{[k][m][b]} = (CX_{[k][m][b]} - \mu_{[k][b]})$ for all k, m, and b

Finally we must compute the B by B class covariance matrices, which are the class centroid covariance matrices for each class as follows.

- **E16**: $COV_{[k]}$ = **covarianceMatrix**(**transpose**($CXU_{[k]}$))

Each of the K class covariance matrices is a B by B covariance matrix for that specified class.

In order to support the core LDA assumption that the class covariance matrices are all equal, we compute the final covariance matrix by combining each class covariance matrix according to their naïve Bayesian probabilities as follows.

- **E17**: $C_{[m]\,[n]} = \sum(COV_{[k][m][n]} * P_{[k]} \mid \text{for all } 1 \le k \le K)$

The final treatment of the covariance matrix allows the LDA optimal coefficients to be computed as shown in the following section.

7.3 LDA Coefficients

Now we can easily compute the single axis coefficient for each class as follows.

- **E18**: $\mathbf{c}_{[k][0]} = -\,0.5\mu_k(C_k)^{-1}(\mu_k)^T + \ln(P_k)$

The B basis function coefficients for each class are computed as follows.

- **E19**: $\mathbf{c}_{[k][1\ldots B]} = \mu_k(C_k)^{-1}$

All together these coefficients form the discriminants for each class as follows.

- **E20**: $D_k(X_n) = \mathbf{c}_{k0}+\mathbf{c}_{k1}*Bf_1(X_n)+\mathbf{c}_{k2}*Bf_2(X_n)+\ldots+\mathbf{c}_{kB}*Bf_B(X_n)$

And the estimated value for Y is defined as follows.

- **E21**: y = argmax($D_1(X_n),D_2(X_n), \ldots ,D_K(X_n)$)

8. Artificial Test Problems

A set of ten artificial classification problems are constructed, with no noise, to compare the four proposed SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison. The four SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm [1]; the MDC algorithm [9], and Linear Discriminant Analysis (LDA) [15]. The five commercially available classification algorithms are available in the KNIME system [16], and are as follows: Decision Tree Learner (DTL); Gradient Boosted Trees Learner (GBTL); Multiple Layer Perceptron Learner (MLP); Random Forest Learner (RFL); and Tree Ensemble Learner (TEL).

Each of the artificial test problems is created around an **X** training matrix filled with random real numbers in the range [-50.0,+50.0]. The number of rows and columns in each test problem varies from 5000x25 to 5000x1000 depending upon the difficulty of the problem. The number of classes varies from **Y** = {1,2} to **Y** = {1,2,3,4,5} depending upon the difficulty of the problem. The test problems are designed to vary from extremely easy to very difficult. The first test problem is linearly separable with 2 classes on 25 columns. The tenth test problem is nonlinear multimodal with 5 classes on 1000 columns.

Standard statistical best practices out of sample testing are employed. First training matric X is filled with random real numbers in the range [-50.0,+50.0], and the Y class values are computed from the argmax functions specified below. A champion is trained on the training data. Next a testing matric X is filled with random real numbers in the range [-50.0,+50.0], and the Y class values are computed from the argmax functions specified below. The previously trained champion is run on the testing data and scored against the Y values. Only the out of sample testing scores are shown in the results (**table 1**).

The argmax functions used to create each of the ten artificial test problems are as follows.

Artificial Test Problems

- **T1**: **y = argmax**(D_1,D_2) where **Y** = {1,2}, **X** is 5000x25, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((1.57*x0),(-39.34*x1),(2.13*x2),(46.59*x3),(11.54*x4))
  - D_2=sum((-1.57*x0),(39.34*x1),(-2.13*x2),(-46.59*x3),(-11.54*x4))
- **T2**: **y = argmax**(D_1,D_2) where **Y** = {1,2}, **X** is 5000x100, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((5.16*x0),(-19.83*x1),(19.83*x2),(29.31*x3),(5.29*x4))
  - D_2:=sum((-5.16*x0),(19.83*x1),(-0.93*x2),(-29.31*x3),(5.29*x4))
- **T3**: **y = argmax**(D_1,D_2) where **Y** = {1,2}, **X** is 5000x1000, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((-34.16*x0),(2.19*x1),(-12.73*x2),(5.62*x3),(-16.36*x4))
  - D_2=sum((34.16*x0),(-2.19*x1),(12.73*x2),(-5.62*x3),(16.36*x4))
- **T4**: **y = argmax**(D_1,D_2,D_3) where **Y** = {1,2,3}, **X** is 5000x25, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((1.57*cos(x0)),(-39.34*square(x10)),(2.13*(x2/x3)),(46.59*cube(x13)),(-11.54*log(x4)))
  - D_2=sum((-0.56*cos(x0)),(9.34*square(x10)),(5.28*(x2/x3)),(-6.10*cube(x13)),(1.48*log(x4)))
  - D_3=sum((1.37*cos(x0)),(3.62*square(x10)),(4.04*(x2/x3)),(1.95*cube(x13)),(9.54*log(x4)))
- **T5**: **y = argmax**(D_1,D_2,D_3) where **Y** = {1,2,3}, **X** is 5000x100, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((1.57*sin(x0)),(-39.34*square(x10)),(2.13*(x2*x3)),(46.59*cube(x13)),(-11.54*cube(x4)))
  - D_2=sum((-0.56*sin(x0)),(9.34*square(x10)),(5.28*(x2*x3)),(-6.10*cube(x13)),(1.48*cube(x4)))
  - D_3=sum((1.37*sin(x0)),(3.62*square(x10)),(4.04*(x2*x3)),(1.95*cube(x13)),(9.54*cube(x4)))
- **T6**: **y = argmax**(D_1,D_2,D_3) where **Y** = {1,2,3}, **X** is 5000x1000, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((1.57*tanh(x0)),(-39.34*sqroot(x10)),(2.13*(x2*x3)),(46.59*(x13/x23)),(2.54*square(x4)))
  - D_2=sum((-0.56*tanh(x0)),(9.34*sqroot(x10)),(5.28*(x2*x3)),(-6.10*(x13/x23)),(1.48*square(x4)))
  - D_3=sum((1.37*tanh(x0)),(3.62*sqroot(x10)),(4.04*(x2*x3)),(1.95*(x13/x23)),(0.54*square(x4)))
- **T7**: **y = argmax**(D_1,D_2,D_3,D_4,D_5) where **Y** = {1,2,3,4,5}, **X** is 5000x25, and each $\mathbf{D_i}$ is as follows
  - D_1=sum((1.57*cos(x0/x21)),(9.34*((square(x10)/x14)*x6)),(2.13*((x2/x3)*log(x8))),(46.59*(cube(x13)*(x9/x2))),(-11.54*log(x4*x10*x15)))
  - D_2=sum((-1.56*cos(x0/x21)),(7.34*((square(x10)/x14)*x6)),(5.28*((x2/x3)*log(x8))),(-6.10*(cube(x13)*(x9/x2))),(1.48*log(x4*x10*x15)))
  - D_3=sum((2.31*cos(x0/x21)),(12.34*((square(x10)/x14)*x6)),(-1.28*((x2/x3)*log(x8))),(0.21*(cube(x13)*(x9/x2))),(2.61*log(x4*x10*x15)))

    - D_4=sum((-0.56*cos(x0/x21)),(8.34*((square(x10)/x14)*x6)),(16.71*((x2/x3)*log(x8))),(-2.93*(cube(x13)*(x9/x2))),(5.228*log(x4*x10*x15)))
    - D_5=sum((1.07*cos(x0/x21)),(-1.62*((square(x10)/x14)*x6)),(-0.04*((x2/x3)*log(x8))),(-0.95*(cube(x13)*(x9/x2))),(0.54*log(x4*x10*x15)))

- **T8**: **y = argmax**(D_1,D_2,D_3,D_4,D_5) where **Y** = {1,2,3,4,5}, **X** is 5000x100, and each **D_i** is as follows
    - D_1=sum((1.57*sin(x0/x11)),(9.34*((square(x12)/x4)*x46)),(2.13*((x21/x3)*log(x18))),(46.59*(cube(x3)*(x9/x2))),(-11.54*log(x14*x10*x15)))
    - D_2=sum((-1.56*sin(x0/x11)),(7.34*((square(x12)/x4)*x46)),(5.28*((x21/x3)*log(x18))),(-6.10*(cube(x3)*(x9/x2))),(1.48*log(x14*x10*x15)))
    - D_3=sum((2.31*sin(x0/x11)),(12.34*((square(x12)/x4)*x46)),(-1.28*((x21/x3)*log(x18))),(0.21*(cube(x3)*(x9/x2))),(2.61*log(x14*x10*x15)))
    - D_4=sum((-0.56*sin(x0/x11)),(8.34*((square(x12)/x4)*x46)),(16.71*((x21/x3)*log(x18))),(-2.93*(cube(x3)*(x9/x2))),(5.228*log(x14*x10*x15)))
    - D_5=sum((1.07*sin(x0/x11)),(-1.62*((square(x12)/x4)*x46)),(-0.04*((x21/x3)*log(x18))),(-0.95*(cube(x3)*(x9/x2))),(0.54*log(x14*x10*x15)))

- **T9**: **y = argmax**(D_1,D_2,D_3,D_4,D_5) where **Y** = {1,2,3,4,5}, **X** is 5000x1000, and each **D_i** is as follows
    - D_1=sum((1.57*sin(x20*x11)),(9.34*(tanh(x12/x4)*x46)),(2.13*((x321-x3)*tan(x18))),(46.59*(square(x3)/(x49*x672))),(-11.54*log(x24*x120*x925)))
    - D_2=sum((-1.56*sin(x20*x11)),(7.34*(tanh(x12/x4)*x46)),(5.28*((x321-x3)*tan(x18))),(-6.10*(square(x3)/(x49*x672))),(1.48*log(x24*x120*x925)))
    - D_3=sum((2.31*sin(x20*x11)),(12.34*(tanh(x12/x4)*x46)),(-1.28*((x321-x3)*tan(x18))),(0.21*(square(x3)/(x49*x672))),(2.61*log(x24*x120*x925)))
    - D_4=sum((-0.56*sin(x20*x11)),(8.34*(tanh(x12/x4)*x46)),(16.71*((x321-x3)*tan(x18))),(-2.93*(square(x3)/(x49*x672))),(5.228*log(x24*x120*x925)))
    - D_5=sum((1.07*sin(x20*x11)),(-1.62*(tanh(x12/x4)*x46)),(-0.04*((x321-x3)*tan(x18))),(-0.95*(square(x3)/(x49*x672))),(0.54*log(x24*x120*x925)))

- **T10**: **y = argmax**(D_1,D_2,D_3,D_4,D_5) where **Y** = {1,2,3,4,5}, **X** is 5000x1000, and each **D_i** is as follows
    - D_1=sum((1.57*lif(x0<x23,sin(x20*x11),cos(x10))),(9.34*(tanh(x12/x4)*x46)),(2.13*lif(x10<0.0,((x321-x3)*tan(x18)),((x156-x31)/tanh(x21)))),(46.59*(square(x3)/(x49*x672))),(-11.54*log(x24*x120*x925)))
    - D_2=sum((-1.56*lif(x0<x23,sin(x20*x11),cos(x10))),(7.34*(tanh(x12/x4)*x46)),(5.28*lif(x10<0.0,((x321-x3)*tan(x18)),((x156-x31)/tanh(x21)))),(-6.10*(square(x3)/(x49*x672))),(1.48*log(x24*x120*x925)))
    - D_3=sum((2.31*lif(x0<x23,sin(x20*x11),cos(x10))),(12.34*(tanh(x12/x4)*x46)),(-1.28*lif(x10<0.0,((x321-x3)*tan(x18)),((x156-x31)/tanh(x21)))),(0.21*(square(x3)/(x49*x672))),(2.61*log(x24*x120*x925)))
    - D_4=sum((-0.56*lif(x0<x23,sin(x20*x11),cos(x10))),(8.34*(tanh(x12/x4)*x46)),(16.71*lif(x10<0.0,((x321-x3)*tan(x18)),((x156-x31)/tanh(x21)))),(-2.93*(square(x3)/(x49*x672))),(5.228*log(x24*x120*x925)))
    - D_5=sum((1.07*lif(x0<x23,sin(x20*x11),cos(x10))),(-1.62*(tanh(x12/x4)*x46)),(-0.04*lif(x10<0.0,((x321-x3)*tan(x18)),((x156-x31)/tanh(x21)))),(-0.95*(square(x3)/(x49*x672))),(0.54*log(x24*x120*x925)))

The Symbolic Classification system for all four SC algorithms (AMAXSC, M_2GP, MDC, LDA) avail themselves of the following operators:

- **Binary Operators**: + - / * minimum maximum
- **Relational Operators**: < <= == <> >= >
- **Logical Operators**: lif land lor
- **Unary Operators**: inv abs sqroot square cube curoot quart quroot exp ln binary sign sig cos sin tan tanh

The unary operators sqroot, curoot, and quroot are square root, cube root, and quart root respectively. The unary operators inv, ln, and sig are the inverse, natural log, and sigmoid functions respectively.

9. Performance On Test Problems

Here we compare the out of sample CEP testing scores of the four proposed SC algorithms and five well-known commercially available classification algorithms to determine where SC ranks in competitive comparison. The four SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm; the MDC algorithm, and Linear Discriminant Analysis (LDA). The five commercially available classification algorithms are available in the KNIME system, and are as follows: Decision Tree Learner (DTL); Gradient Boosted Trees Learner (GBTL); Multiple Layer Perceptron Learner (MLP); Random Forest Learner (RFL); and Tree Ensemble Learner (TEL).

Table 1. Test Problem CEP Testing Results

Test	AMAXSC	LDA	M_2GP	MDC	DTL	GBTL	MLP	RFL	TEL
T1	0.0808	0.0174	0.0330	0.0330	0.0724	0.0308	0.0072	0.0492	0.0496
T2	0.1108	0.0234	0.0656	0.0402	0.0740	0.0240	0.0360	0.0664	0.0648
T3	0.1436	0.0182	0.1010	0.0774	0.0972	0.0332	0.0724	0.1522	0.1526
T4	0.1954	0.0188	0.0180	0.0162	0.0174	0.0170	0.0472	0.0260	0.0252
T5	0.1874	0.1026	0.1052	0.1156	0.0858	0.0530	0.3250	0.0920	0.0946
T6	0.6702	0.5400	0.4604	0.5594	0.5396	0.3198	0.6166	0.6286	0.6284
T7	0.4466	0.4002	0.4060	0.4104	0.2834	0.2356	0.4598	0.2292	0.2284
T8	0.8908	0.4176	0.4006	0.4124	0.2956	0.2340	0.4262	0.2250	0.2248
T9	0.8236	0.7450	0.7686	0.6210	0.6058	0.4286	0.6904	0.4344	0.4334
T10	0.8130	0.7562	0.7330	0.6440	0.5966	0.4286	0.5966	0.4296	0.4352
Avg	0.4362	0.3039	0.3091	0.2930	0.2668	0.1805	0.3277	0.2333	0.2337

On a positive note, the three new proposed symbolic classifications algorithms are a great improvement over the vanilla AMAXSC algorithm. On a negative note, none of the three newly proposed SC algorithms are the best performer. The top performer overall by a good margin is the Gradient Boosted Trees Learner (GBTL).

Is is interesting to note that all three newly proposed SC algorithms performed better overall than the Multiple Layer Perceptron Learner (MLP). This is significant; as it is the first time we've seen a genetic programming SC algorithm beat one of the top commercially available classification algorithms.

Of the three newly proposed SC algorithms, surprisingly the MDC algorithm was the best overall performer; although all three SC algorithms were grouped very close together in performance.

Clearly progress has been made in the development of commercially competitive SC algorithms. But, a great deal more work has to be done before SC can outperform the Gradient Boosted Trees Learner (GBTL).

10. Conclusions

Several papers have proposed GP Symbolic Classification algorithms for multiclass classification problems [1], [9], [10], [15]. Comparing these newly proposed SC algorithms with the performance of five commercially available classifications algorithms shows that progress has been made. The three newly proposed SC algorithms now outperform one of the top commercially available algorithms on a set of artificial test problems of varying degrees of difficulty.

It may be significant that, of the commercially available classifiers, the best overall performers are all *tree learners*. While the random forest learner (RFL) and the tree ensemble learner (TEL) enjoy enhanced performance over the decision tree learner (DTL), it is the gradient boosted tree learner (GBTL) which clearly enjoys the overall top performer slot on these ten artificial test problems.

It will be interesting to see if gradient boosting can be adapted to the three newly proposed SC algorithms (M_2GP, MDC, and LDA) such that they will also enjoy enhanced performance.

11. ACKNOWLEDGMENTS

Our thanks to: Thomas May from Lantern Credit for assisting with the KNIME Learner training/scoring on all ten artificial classification problems.

12. REFERENCES

[1] Ingalalli, Vijay, Silva, Sara, Castelli, Mauro, Vanneschi, Leonardo 2014. *A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems*. Euro GP 2014 Springer.

[2] Korns, Michael F. 2013. *Extreme Accuracy in Symbolic Regression*. Genetic Programming Theory and Practice XI. Springer, New York, NY.

[3] Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by means of Natural Selection.* The MIT Press. Cambridge, Massachusetts.

[4] Korns, Michael F. 2012. *A Baseline Symbolic Regression Algorithm*. Genetic Programming Theory and Practice X. Springer, New York, NY.

[5] Keijzer, Maarten. 2003. *Improving Symbolic Regression with Interval Arithmetic and Linear Scaling*. European Conference on Genetic Programming,

[6] Billard, Billard., Diday, Edwin. 2003. *Symbolic Regression Analysis*. Springer. New York, NY.

[7] Korns, Michael F. 2014. *Extremely Accurate Symbolic Regression for Large Feature Problems*. Genetic Programming Theory and Practice XII. Springer, New York, NY.

[8] Korns, Michael F. 2015. *Highly Accurate Symbolic Regression for Noisy Training Data*. Genetic Programming Theory and Practice XIII. Springer, New York, NY.

[9] Korns, Michael F. 2016. *An Evolutionary Algorithm for Big Data Multiclass Classification Problems*. Genetic Programming Theory and Practice XIV. Springer, New York, NY.

[10] Munoz, Louis, Silva, Sara, M. Castelli, Trujillo 2014. *M3GP Multiclass Classification with GP*. Euro GP 2015 Springer.

[11] Fisher, R. A. 1936. *The Use of Multiple Measurements in Taxonomic Problems*. Annals of Eugenics 7 (2) 179-188.

[12] Friedman, J. H. 1989. *Regularized Discriminant Analysis*. Journal of American Statistical Association 84 (405) 165-175.

[13] McLachan, Geoffrey, J. 2004. *Discriminant Analysis and Statistical Pattern Recognition.* Wiley. New York, NY.

[14] Korns, Michael F., 2013. Extreme Accuracy in Symbolic Regression. In Soule, Terrance, and Wortzel, Bill, editors, Genetic Programming Theory and Practice XI, New York, New York, USA. Springer.

[15] Korns, Michael F., 2017. Evolutionary Linear Discriminant Analysis for Multiclass Classification Problems. In GECCO Conference Proceedings ‘17, July 15-19, Berlin Germany 2017.

[16] Michael R. Berthold and Nicolas Cebron and Fabian Dill and Thomas R. Gabriel and Tobias K\"{o}tter and Thorsten Meinl and Peter Ohl and Christoph Sieb and Kilian Thiel and Bernd Wiswedel, 2007. KNIME: The Konstanz Information Miner}. In Studies in Classification, Data Analysis, and Knowledge Organization, Springer, Berlin. ISBN 978-3-540-78239-1.