Strong Typing, Swarm Enhancement, and Deep Learning Feature Selection in the Pursuit of Symbolic Regression-Classification Accuracy

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

Tim May
Insight Sciences Corp

ABSTRACT

Symbolic Classification (SC), an offshoot of Genetic Programming (GP), can play an important role in any well rounded predictive analytics tool kit – especially because of its so called "*WhiteBox*" properties. In these recent papers, algorithms are developed designed to push SC to the level of basic classification accuracy competitive with existing commercially available classification tools [1][9][10], including the introduction of GP assisted Linear Discriminant Analysis (LDA) [15]. In this paper we add a number of important enhancements to our basic SC system and demonstrate their accuracy improvements on a set of theoretical problems and on a banking industry problem. We enhance GP assisted linear discriminant analysis with a modified version of Platt's Sequential Minimal Optimization algorithm [14] which we call (MSMO), and with swarm optimization techniques [7]. We add a user defined typing system, and we add deep learning feature selection to our basic SC system. This extended algorithm (LDA^{++}) is highly competitive with the best commercially available M-Class classification techniques on both a set of theoretical problems and on a real world banking industry problem. This new LDA^{++} algorithm moves genetic programming classification solidly in the top rank of commercially available classification tools.

Keywords

Symbolic Classification, Genetic Programming, Linear Discriminant Analysis, Deep Learning.

1. Introduction

Symbolic Classification (SC), an offshoot of Genetic Programming (GP), can play an important role in any well rounded predictive analytics tool kit – especially because of its so called "*WhiteBox*" properties. In these recent papers, algorithms are developed designed to push SC to the level of basic classification accuracy competitive with existing commercially available classification tools [1][9][10], including the introduction of GP assisted Linear Discriminant Analysis (LDA) [15]. In this paper we add a number of important enhancements to our basic SC system and demonstrate their accuracy improvements on a set of theoretical problems and on a banking industry problem. We enhance GP assisted linear discriminant analysis with a modified version of Platt's Sequential Minimal Optimization algorithm [14] which we call (MSMO), and with swarm optimization techniques [7]. We add a user defined typing system, and we add deep learning feature selection to our basic SC system. This extended algorithm (LDA^{++}) is highly competitive with the best commercially available M-Class classification techniques on both a set of theoretical problems and on a real world banking industry problem. This new LDA^{++} algorithm moves genetic programming classification solidly in the top rank of commercially available classification tools.

Each of the first four genetic programming SC algorithms is briefly explained further, plus a more detailed description of the proposed extended Linear Discriminant Analysis algorithm (LDA^{++}) is presented in this paper. The Platt inspired MSMO algorithm is described in detail and the manner in which the LDA matrix math and Swarm optimizations are tightly integrated is also described in detail herein. The user defined typing system is described in detail herein, and the deep learning feature selection methodology is described in detail

For theoretical testing, 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 theoretical 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 ten classification algorithms are published herein.

No assertion is made that these fiver 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.

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

For real world testing, we use an actual banking data set for loan scoring as it was received. The training data contains 337 features with 36223 rows, while the testing data contains the same 337 features with an additional 85419 rows. The testing and training data are distinct. We include a comparison study of the five new SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison on this real world problem. Also included is the bank's benchmark score, achieved over a multiple month period by the bank's in-house data science team with proprietary tools.

In conclusion we show that, on the theoretical problems, the two best classification algorithms are Gradient Boosted Decision Trees (GBTL) and this paper's extended Linear Discriminant Analysis (LDA^{++}). Furthermore we show, on the real world banking problem, the three best classification algorithms are Gradient Boosted Decision Trees (GBTL), the bank's in-house data science team, and this paper's extended Linear Discriminant Analysis (LDA^{++}).

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**: $y = \text{argmax}(gp_{\mathbf{1}}, gp_{\mathbf{2}},\ldots, gp_{\mathbf{K}})$ where **K** 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 K) which has the highest value, and is strongly related to the Bayesian probability that the training point belongs to the k-th 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 2K weights*).

Each $\mathbf{gp_k}$ represents a separate discriminant function evolved via standard genetic programming. The argmax() function chooses the class (1 to K) which has the highest value, and is strongly related to the Bayesian probability that the training point belongs to the k-th class. Given a set of GP evolved discriminant formulas $\{gp_1, gp_2,\ldots, gp_K\}$, the objective of the MDC algorithm is to optimize the choice of coefficient weights $\{w_{10}, w_{11}, w_{20}, w_{21},\ldots,w_{K0}, w_{K1}\}$ 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 K classes, the algorithm attempts to minimize the Mahalanobis distance between the n-th training point and the centroid of the k-th class. The basic training algorithm is as follows.

Algorithm A1: M_2GP Training

1. Input: X, Y, D – where X is an MxN real matrix, Y is an N Vector, D is a scalar
2. For g from 1 to G do

3. Generate: F = {f_1, f_2,…,f_D} set of D solutions
4. Evaluate: Z_s = Eval(f_s(X)) for s from 1 to D – a D-dimensional point
5. Cluster: Z^k in Z for all k from 1 to K – group all the Z which belong to each class
6. For k from 1 to K do
7. C^k = covar(Z^k) – a d by d covariance matrix for each class
8. W^k = centroid(Z^k) – a 1 by D centroid vector
9. $D_k(X_n)$ = sqrt($(Z_n-W^k)*(C^k)^{-1}*(Z_n-W^k)^T$) – for n from 1 to N (*the number of training points*)
10. For n from 1 to N do EY_n = argmin($D_1(X_n)$,$D_2(X_n)$,…,$D_C(X_n)$)
11. For n from 1 to N do E_n = 1 IFF EY_n <> Y_n, 0 otherwise
12. Minimize average(EY)
13. 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 i-th 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**: $CEP = \sum(EY_{[n]} \neq Y_{[n]} \mid \text{for all } n) / N$

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 n-th training point, μ_k is the mean vector for the k-th class, $(C_k)^{-1}$ is inverse of the covariance matrix for the k-th class, $(X_n)^T$ is the transpose of the n-th training point, $(\mu_k)^T$ is the transpose of the mean vector for k-th class, and $\ln(P_k)$ is the natural logarithm of the naïve probability that any training point will belong to the k-th 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**: $lse = \text{fitness}(X, Y, Bf_1, \ldots, Bf_B, \mathbf{c}_0, \ldots, \mathbf{c}_B)$

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

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

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

7.4 Addressing the Problems with LDA Coefficients

The LDA matrix math is made easily computable via an important set of a priori assumptions. These assumptions include that the distribution of the training data is Gaussian, that all covariance matrices are equivalent, and that the covariance matrix for all classes is not singular. Unfortunately these ideal assumptions are not always valid in the course of LDA training, especially since in an SC run, there are tens of thousands to hundreds of thousands of individual LDA training attempts on various nonlinear GP discriminant candidates. Whenever these assumptions are invalid, the resulting LDA coefficients will not be entirely accurate.

To address the problems with LDA coefficients, the LDA^{++} algorithm changes the argument for the LDA matrix math from a statistical justification to a heuristic justification where only approximately accurate results are expected.

Since the LDA *heuristic* is no longer expected to obtain optimal results, we must add optimization steps after the LDA to correct for any possible errors produced. The second (post LDA) step is a set of heuristic matrix math adjustments designed to adjust for any singular covariance matrix conditions. The third step adds Modified Sequential Minimal Optimization (MSMO), and the fourth step adds a Swarm intelligence layer consisting of the Bees algorithm [7].

In the unhappy event that the covariance matrix, $\mathbf{C_k}$, in equations **E18** and **E19** is singular, then the matrix inversion function, $\mathbf{(C_k)^{-1}}$, will fail with a divide by zero error. Our second heuristic alters the computer code of the covariance matrix inversion function such that, should a divide by zero error be detected, the diagonal of the covariance matrix, $\mathbf{C_k}$, is multiplied by the scalar **1.0001**. In the vast majority of cases multiplying the covariance matrix diagonal by this scalar forces the matrix to be non-singular. After diagonal adjustment the matrix inversion is attempted anew. The resulting LDA coefficient will of course be inaccurate but nevertheless approximately accurate. After diagonal adjustment, should a divide by zero error still be encountered, then the adjusted computer code will divide by the scalar **.0001** instead of zero. Once again the resulting LDA coefficient will of course be inaccurate but nevertheless approximately accurate.

The LDA coefficients, produced by the adjusted matrix inversion function, will be anywhere from accurate to approximately accurate. In order to optimize the coefficients further, we add a layer of Modified Sequential Minimal Optimization (MSMO). We now proceed to describe our MSMO algorithm in the next section.

7.5 Modified Sequential Minimal Optimization (MSMO)

The third heuristic layer of the LDA^{++} algorithm is an opportunistic modification of Platt's sequential minimization optimization algorithm often used to train support vector machines [14]. At the start of the MSMO heuristic, the percent of misclassification errors (CEP) in the training data set is calculated for the LDA coefficients. If the CEP is greater than **10%**, then the MSMO heuristic is skipped on the theory that it is not worthwhile to waste resources on a heuristics which may improve the accuracy of the CEP by a maximum of **2%** or **3%**. Thus MSMO resources are only expended for the better candidates. Better candidates receive more evolutionary activity. Worse candidates receive less evolutionary activity.

At the start of the Modified Sequential Minimal Optimization (MSMO) layer, the candidate contains a swarm pool of a single set of coefficient constants which was produced by the LDA heuristic with its singular matrix modifications. Also the CEP for LDA coefficients in this single entry in the swarm pool is available.

For each repetition of our MSMO algorithm, on the current candidate, the most fit coefficient entry in the swarm pool is chosen. If the CEP for the best coefficient entry is **0%**, then the MSMO algorithm is terminated. If all allocated evolutionary iterations have been exhausted, then the MSMO algorithm is terminated. If the CEP is greater than **0%** then a single erroneous training point is chosen at random, **n**.

Since the chosen training point, **n**, is in error, we know that it's estimated dependent variable, **e**, will not match the actual dependent variable, **y** (i.e. **e≠y**). If **K** is the number of classes, then **0 ≤ e < K** and **0 ≤ y < K**. Since LDA uses the **argmax** function to select the discriminant function with the highest Bayesian probability, we know that two discriminant formulas have the following relationship.

- **E22**: $e = \text{argmax}(D_1(X_n), D_2(X_n), \ldots, D_K(X_n))$

And therefore

- **E23**: $D_e(X_n) \geq D_y(X_n)$

And therefore

- **E24**: $\mathbf{c}_{e0}+\mathbf{c}_{e1}*Bf_1(X_n)+\mathbf{c}_{e2}*Bf_2(X_n)+\ldots+\mathbf{c}_{eB}*Bf_B(X_n) \geq \mathbf{c}_{y0}+\mathbf{c}_{y1}*Bf_1(X_n)+\mathbf{c}_{y2}*Bf_2(X_n)+\ldots+\mathbf{c}_{yB}*Bf_B(X_n)$

Our goal at this step in the MSMO algorithm is to force

- **E25**: $y = \text{argmax}(D_1(X_n), D_2(X_n), \ldots, D_K(X_n))$

Since the **B** basis functions in the **K** discriminants are all the same, our only option is to modify the coefficients. Using equation **E24**, we select the basis function, $\mathbf{Bf_m(X_n)}$, such that the absolute difference

- **E24**: $\text{abs}((\mathbf{c}_{em}*Bf_m(X_n))) - (\mathbf{c}_{ym}*Bf_m(X_n)))$

is greater than all other possible choices from the B basis functions. We then make minimalist random changes to the coefficients $\mathbf{c_{em}}$ and $\mathbf{c_{ym}}$ such that equation **E25** is forced to be true.

At this point in the MSMO algorithm, the modified coefficients are used to score the candidate obtaining a new CEP. The modified coefficients along with their associated new CEP are inserted into the candidate's swarm pool sorted in order of most fit CEP. Then the most fit coefficient entry in the swarm pool is chosen, and the MSMO algorithm repeats itself until all allocated evolutionary iterations have been exhausted.

7.6 Bees Swarm Optimization

The fourth step adds a Swarm intelligence layer consisting of the Bees algorithm [7].The fourth heuristic layer of the LDA^{++} algorithm is a Swarm intelligence heuristic consisting of the Bees algorithm [7]. The BEES algorithm is too complex to describe in this paper. In summary the BEES algorithm involves a fitness driven evolutionary modification of all coefficients at once using a well-defined Swarm intelligence approach. More details can be found in [7].

At the start of the BEES heuristic, the percent of misclassification errors (CEP) in the training data set are calculated for the most fit coefficients. If the CEP is greater than **5%**, then the BEES heuristic is skipped on the theory that it is not worthwhile to waste resources on a heuristic which may improve the accuracy of the CEP by a maximum of **0.5%** or **1%**. Thus BEES resources are only expended for the better candidates. Better candidates receive more evolutionary activity. Worse candidates receive less evolutionary activity.

8. User Defined Typing System

Unconstrained expressions (models) from Symbolic Classification Systems are often rejected by decision makers regardless of their advantageous fitness scores. Decision makers most often require that SC models make intuitive sense and be logically defensible within the problem domain. This is especially true with SC models for high impact decisions. In general, the greater the impact of trusting an SC model, the more the decision maker needs to understand and believe in the model.

SC systems can be modified with template based logic to constrain the evolution of resulting models to match decision maker specifications [2][4][8]. Template constrained SC systems produce models which are readily accepted by the decision maker since those models always conform to the templates supplied by the decision maker. Unfortunately such template constraint systems are insufficient to accommodate user strong typing rules.

Strong typing rules are important in SC applications where unit types should not be mixed. For instance in a banking system the decision maker might want to prohibit all SC models which add 'personal income' to 'number of defaults', or which multiply 'credit score' by 'race'. SC models which are confused about appropriate typing will be rejected out of hand by decision makers, *even if they are highly accurate*, and may engender distrust of the SC system as a whole.

8.1 User Defined Templates with Constraints

Given any selected maximum depth K, it is an easy process to construct a maximal binary tree fixed constraint template U_K, which can be overlaid on the GP system without violating the selected maximum depth limit K nor limiting the s-expressions which can be produced. As long as we are reminded that each f-node represents a function node while each t-node represents a terminal node (*either a feature or a real number constant*), the template construction algorithm, for building the template U_K, is both 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})

For the arbitrary fixed template Uk, each f-node represents a Lisp function call such as **cos**, **sin**, **+**, **/**, or **ln**, and each t-node represents a terminal feature or constant such as $\mathbf{x_0}$, **34.56**, $\mathbf{x_{10}}$, or **-45.1**. The basic GP symbolic regression system [4] contains a set of functions **F**, and a set of terminals **T**. We let t ϵ T, where T = {x0, …, xM, …IEEE real numbers…}. We let f ϵ F = {+, -, *, /, sin, cos, tan, square, etc}. Now U_K becomes almost another template representation of the irregular s-expressions in the SC. We say almost because U_K is fixed and regular whereas the SC s-expressions are irregular. To complete the template, we must add a special no op function, ξ, to F = F ∪ ξ. The special ξ function allows U_K to express irregular s-expressions be defining ξ(a,b, …) = ξ (a) = a. Now 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 template 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 standard IEEE double long at 2^{64} in size.

User specified constraints can be added to our SC system by explicitly enumerating each node in the template U_K and by adding constraints to each of the explicitly enumerated nodes. For instance an explicit enumeration of nodes in U_2 would be as follows.

- (U_2): (f (f t t) (f t t)) = f0(f1(t0,t1),f2(t2,t3))

User specified constraints can be added to SC search commands by explicit user declarations as follows.

- **E25**: regress(f0(f1(t0,t1),f2(t2,t3))) where {f0(cos,sin,tan,*) f2(+,-,*,/) t0(x_4,x_6) t3(x_{10},x_{19},1.0,0.0)}

The above SC search command will produce models constrained by the explicit user declarations as in the following examples (*remember f0, f2, t0, and t3 are constrained as declared above, while f1, t1, and t2 are unconstrained*).

Table 1: Productions from template *f0(f1(t0,t1),f2(t2,t3))* as constrained in **E25**

f0	**f1**	**f2**	**t0**	**t1**	**t2**	**t3**	**Lisp expression**
*	sin	+	x_6	x_{30}	45.7	x_{10}	(* (sin x_6) (+ 45.7 x_{10}))
*	ξ	*	x_4	x_{40}	x_1	x_{19}	(* x_4 (* x_{10} x_{19}))
sin	ln	*	x_6	0.0	x_{21}	1.0	(sin (ln x_6))
cos	square	/	x_4	x_{63}	x_1	x_{10}	(cos (square x_4))

User constraints enhance the probability that a resulting SC model will be accepted by the decision makers; but, alone they are not enough. Adding user specified strong typing to the SC system will also be required for many applications.

8.2 Strong Typing

A user template facility is greatly enhanced if accompanied by the ability to add strong typing rules to the SC search command. These rules may be formed as in the following examples.

- Number = {x_0, x_1, x_{10}, x_{20}}
- Number = {(Number+Number), (Number*Number),(Number/Number)(Number-Number)}
- Income = {x_{21}, x_{31} x_{11}, x_{22}}
- Income = {(Income + Income), (Income * Income),(Income / Income),(Income - Income)}
- Income = {cos(Income), tan(Income),sin(Income)}

The type rules consist of a "type" followed by a set of features, or operator-type combinations which result in the specified type. The type rules must not contain conflicts as in the rules below which contain a type conflict because feature x_{10} cannot be both a Number and a Widget.

- Number = {x_0, x_1, x_{10}, x_{20}}
- Income = {x_{21}, x_{31} x_{10}, x_{22}}

The type rules can be consolidated by type and subdivided into the features, unary operators, and binary operators which result in the specified type as in the following example.

- Number
  - {x_0, x_1, x_{10}, x_{20}}
  - {(Number+Number), (Number*Number),(Number/Number)(Number-Number)}
  - {cos(Number), abs(Number)}
- Income
  - {(Number+Income), (Number*Income),(Income/Number)(Income-Number)}

Taken together, a user template system with constraints and strong typing can greatly enhance the acceptability to the final SC model in the eyes of the domain expert. This can be a critical component of model acceptance.

9. Deep Learning Enhancements

We use the RQL language to develop a deep learning strategy to enhance the LDA^{++} algorithm [2][8]. The first deep learning layer (*RQL Island*) handles feature selection by performing a series of individual LDA (*including the SMO, and Bees Algorithm enhancements*) learning runs on each individual independent feature using the following RQL search specification.

- **E26**: lda (v0,inv(v0),abs(v0),sqroot(v0),square(v0),cube(v0),curoot(v0),ln(v0),cos(v0),sin(v0),tan(v0),tanh(v0))

Each independent feature is given this same nonlinear treatment with the above 12 nonlinear basis functions. The best ranking features (CEP) are selected as the '*of-special-interest*' feature group. When feature selection is complete, a second deep learning layer is run, whose only connected inputs are the '*of-special-interest*' feature group. This general deep learning layer is a standard 20 basis function pareto front LDA evolutionary GP run (*including the SMO, and Bees Algorithm enhancements*) which halts when it has reached a learning plateau.

Once the general pareto front LDA layer is complete, another 41 strongly linked model recurrent deep learning layers are launched. The first of these is another standard 20 basis function pareto front LDA layer (*including the SMO, and Bees Algorithm enhancements*) but which has ALL features connected.

The next twenty deep learning layers are standard 20 basis function pareto front LDA layers (*including the SMO, and Bees Algorithm enhancements*) whose only connected inputs are the '*of-special-interest*' feature group; BUT, where each layer has only one of its basis functions participating in evolution. The remaining basis functions are all fixed. Each of the twenty deep learning layers has a different basis function undergoing active evolution while the remaining basis functions are fixed.

The next twenty deep learning layers are also standard 20 basis function pareto front LDA layers (*including the SMO, and Bees Algorithm enhancements*) which have ALL features connected; BUT, where each layer has only one of its basis functions participating in evolution. The remaining basis functions are all fixed. Each of the twenty deep learning layers has a different basis function undergoing active evolution while the remaining basis functions are fixed.

Each of the above 41 deep learning layers are *strongly linked and model recurrent*. At the start, the best model discovered by the second 20 basis function learning layer is fed into each of the 41 strongly linked model recurrent deep learning layers. For example:

- **E27**: lda (Bf_1,Bf_2,…,Bf_{20})

This best scoring model is fed into each of the 41 deep learning layers, where layer 1 performs a general evolutionary search on all 20 basis functions with ALL features connected. Layers 2 through 21 each perform a general evolutionary search on all 20 basis functions with only the '*of-special-interest*' feature connected. And, where each layer evolves only one basis function holding all other basis functions fixed. Layers 22 through 41 each perform a general evolutionary search on all 20 basis functions with ALL the features connected. And, where each layer evolves only one basis function holding all other basis functions fixed.

Since each of these 41 deep learning layers are strongly linked, the moment any one of these layers discovers a model, *which is a global fitness improvement*, all processing stops AND the new model is fed back into all 41 deep learning layers (*hence the term model recurrent*). For instance, after some evolutionary effort, the nth layer discovers a model which is a global fitness improvement, for example.

- **E28**: lda (Bg_1,Bg_2,…,Bg_{20})

This *new improved globally fit* model is fed recurrently into each of the 41 deep learning layers, and the whole strongly linked model recurrent process begins anew until another better globally fit model is discovered.

For purposes of keeping the models "*WhiteBox*" we arbitrarily set the maximum number of basis functions to 20, and we arbitrarily set the maximum depth of each basis function to 3 (*i.e.* $2^3 = 8$ *terminal nodes*). If a basis function is not needed for accuracy the algorithm will set its coefficient to 0.0, and each a basis function will evolve to the optimal shape (*limited by our arbitrary maximum depth of 3*). Obviously these parameters can be adjusted for the particular problem domain.

10. Artificial Test Problems

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

Each of the artificial test problems is created around an **X** training matrix filled with random real numbers in the range [-10.0,+10.0]. 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** = {0,1} to **Y** = {0,1,2,3,4} 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 the training matric X is filled with random real numbers in the range [-10.0,+10.0], 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 [-10.0,+10.0], 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 $\mathbf{D_i}$ 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 $\mathbf{D_i}$ 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 $\mathbf{D_i}$ 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 five SC algorithms (AMAXSC, M_2GP, MDC, LDA, and 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.

11. Real World Banking Problem

For real world testing, we use an actual banking data set for loan scoring as it was received by Lantern Credit. The training data contains 337 features with 36223 rows, while the testing data contains the same 337 features with an additional 85419 rows. The training and testing data are distinct, and the training and testing data are completely anonymized. The 337 independent features contain both categorical and continuous data,. But do nOT contain any FICO or similar agency ranking scores. The dependent variable is 1 for ‘good’ and 0 for ‘bad’ loan.

An extensive user defined typing file of thousands of strong typing rules was constructed, with the help of domain experts.

The objective is to develop a model for scoring incoming 337 feature loan applications. The model must pass the bank's in-house compliance management, and be accepted by the bank's in-house lending management team.

12. Performance On The Theoretical Problems

Here we compare the out of sample CEP testing scores of the five proposed SC algorithms and five well-known commercially available classification algorithms to determine where SC ranks in competitive comparison. The five SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm [1]; the MDC algorithm [9], Linear Discriminant Analysis (LDA) [15], and Linear Discriminant Analysis extended with SMO and Swarm (LDA^{++}). The five commercially available classification algorithms are available in the KNIME system [16], and are as follows: Multiple Layer Perceptron Learner (MLP); Decision Tree Learner (DTL); Random Forest Learner (RFL); Tree Ensemble Learner (TEL); and Gradient Boosted Trees Learner (GBTL). The following table lists each classification algorithm in descending order of average CEP scores on all ten theoretical test problems. The lower the CEP the more accurate the classification results.

Table 1. Test Problem CEP Testing Results After Deep Learning Enhancements

Test	AMAXSC	MLP	M_2GP	LDA	MDC	DTL	TEL	RFL	LDA^{++}	GBTL
T1	0.0808	0.0072	0.0330	0.0174	0.0330	0.0724	0.0496	0.0492	**0.0138**	0.0308
T2	0.1108	0.0360	0.0656	0.0234	0.0402	0.0740	0.0648	0.0664	**0.0116**	0.0240
T3	0.1436	0.0724	0.1010	0.0182	0.0774	0.0972	0.1526	0.1522	**0.0132**	0.0332
T4	0.1954	0.0472	0.0180	0.0188	0.0162	0.0174	0.0252	0.0260	***0.0194***	*0.0170*
T5	0.1874	0.3250	0.1052	0.1026	0.1156	0.0858	0.0946	0.0920	***0.0712***	*0.0530*
T6	0.6702	0.6166	0.4604	0.5400	0.5594	0.5396	0.6284	0.6286	***0.5420***	*0.3198*
T7	0.4466	0.4598	0.4060	0.4002	0.4104	0.2834	0.2284	0.2292	**0.1522**	0.2356
T8	0.8908	0.4262	0.4006	0.4176	0.4124	0.2956	0.2248	0.2250	**0.2302**	0.2340
T9	0.8236	0.6904	0.7686	0.7450	0.6210	0.6058	0.4334	0.4344	**0.4188**	0.4286
T10	0.8130	0.5966	0.7330	0.7562	0.6440	0.5966	0.4352	0.4296	**0.4186**	0.4286
Avg	0.4362	0.3277	0.3091	0.3039	0.2930	0.2668	0.2337	0.2333	**0.1891**	0.1805

On a positive note, the four new proposed symbolic classifications algorithms are a great improvement over the vanilla AMAXSC algorithm, and the newly proposed LDA^{++} SC algorithm is extremely competitive with the best performer. The top performer overall by a very small margin is the Gradient Boosted Trees Learner (GBTL). The penultimate performer is the newly proposed LDA^{++} SC algorithm.

Is is interesting to note that all four newly proposed SC algorithms performed better overall than the Multiple Layer Perceptron Learner (MLP). Of the four newly proposed SC algorithms, the LDA^{++} algorithm was the best overall performer, and is extremely competitive with the commercially available Gradient Boosted Trees Learner (GBTL). In fact the LDA++ algorithm did better than GBTL on all but three of the test problems.

Clearly progress has been made in the development of commercially competitive SC algorithms. We now have an SC classification algorithm which is highly competitive with GBTL. But, a great deal more work has to be done before SC can radically outperform the Gradient Boosted Trees Learner (GBTL).

13. Performance On The Real World Problem

We include a comparison study of the five new SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison on this real world banking problem. Also included is the bank's benchmark score, achieved over a multiple month period by the bank's in-house data science team with proprietary tools. The bank's benchmark score represents the best that human domain experts can achieve with months of effort and state-of-the-art data science tools.

The bank's benchmark score is compared to the five proposed SC algorithms and five well-known commercially available classification algorithms to determine just where SC now ranks in competitive comparison. The five SC algorithms are: simple genetic programming using argmax referred to herein as (AMAXSC); the M_2GP algorithm [1]; the MDC algorithm [9], Linear Discriminant Analysis (LDA) [15], and Linear Discriminant Analysis extended with MSMO and Swarm (LDA^{++}). The five commercially available classification algorithms are available in the KNIME system [16], and are as follows: Multiple Layer Perceptron Learner (MLP); Decision Tree Learner (DTL); Random Forest Learner (RFL); Tree Ensemble Learner (TEL); and Gradient Boosted Trees Learner (GBTL).

Table 2. Banking Problem CEP Testing Results After Deep Learning Enhancements

AMAXSC	MLP	M_2GP	LDA	MDC	DTL	TEL	LDA^{++}	Benchmark	RFL	GBTL
0.9101	0.4523	0.4472	0.4282	0.4174	0.3165	0.3084	0.2856	**0.2841**	0.2658	0.2621

Each CEP score represents the percent of applications which were misclassified (*lower is better*). On a positive note, the four new proposed symbolic classifications algorithms are a great improvement over the vanilla AMAXSC algorithm, and the newly proposed LDA^{++} SC algorithm is extremely competitive with the best performer. The top performer overall by a 2% margin is the Gradient Boosted Trees Learner (GBTL). It is interesting to note that the LDA^{++} result included user defined types - thus rendering basis functions which meet the preconditions established by the bank. This greatly improves the probability of model acceptance.

Is is interesting to note that of the four newly proposed SC algorithms, the LDA^{++} algorithm was the best performer, and is extremely competitive with the bank's in-house benchmark. Both the Random Forest Learner (RFL) and the Gradient Boosted Trees Learner (GBTL) are extremely competitive top performers. However, despite achieving a 2% advantage over the Bank's benchmark, both the RFL and GBTL models will face a difficult uphill battle for acceptance by the bank's in-house compliance management. Each of the RFL and GBTL models contain hundreds to thousands of embedded decision trees. They are very difficult to understand and doubly difficult to support cogently within the logic of the banking domain.

On the other hand, the newly proposed LDA^{++} SC algorithm produces an easily understood "*WhiteBox*" model with twenty or less very simple user type compliant basis functions. In its first training run, the LDA^{++} algorithm has produced a model that is highly competitive with the bank's in-house model (*which took their in-house data science team months to construct*). Furthermore, the LDA^{++} model is already in a form that is easily understood by the bank's in-house data science team, easy for them to work with, and compatible with their pre-specified constraints.

Clearly progress has been made in the development of commercially competitive SC algorithms. We now have an SC classification algorithm which is highly competitive with the bank's in-house benchmark. But, a great deal more work has to be done before SC can radically outperform the Gradient Boosted Trees Learner (GBTL) in terms of raw accuracy alone.

14. 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. All four newly proposed SC algorithms performed better overall than the Multiple Layer Perceptron Learner (MLP). Of the four newly proposed SC algorithms, the LDA^{++} algorithm was the best overall performer by a good margin, and is extremely competitive with the commercially available Gradient Boosted Trees Learner (GBTL). In fact the LDA^{++} algorithm did better than GBTL on all but three of the theoretical test problems.

Clearly progress has been made in the development of commercially competitive SC algorithms. We now have an SC classification algorithm which is highly competitive with GBTL. But, a great deal more work has to be done before SC can radically outperform the Gradient Boosted Trees Learner (GBTL) on the basis of raw accuracy alone. We must perform comparative tests on a much wider range of theoretical problems, and we must perform comparative tests on a wide range of real world industry problems.

15. ACKNOWLEDGMENTS

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

16. 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] Karaboga, D, Akay, B.,. 2009. *A Survey: Algorithms Simulating Bee Swarm Intelligence*. In Artificial Intelligence Review, 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] Platt. J., 1998. *Sequential Minimal Optimization: A Fast Algorithm For Training Support Vector Machines*. Technical Report. Advances In Kernel Methods – Support Vector Learning.

[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.