Transformation of Input Space using Statistical Moments: EA-Based Approach

Ahmed Kattan, Michael Kampouridis, Yew-Soon Ong, and Khalid Mehamdi

***Abstract*— Standard Regression models are presented with n samples from an input space χ that is composed of observational data of the form $(x_i, y(x_i)), i = 1...n$ where each x_i denotes a k-dimensional input vector of design variables and y is the response. When $k \gg n$, high variance and over-fitting become a major concern. In this paper we propose a novel approach to mitigate this problem by transforming the input vectors into new smaller vectors (called Z set) using only a set of simple statistical moments. Genetic Algorithm (GA) has been used to evolve a transformation procedure. It is used to optimise an optimal sequence of statistical moments and their input parameters. We used Linear Regression (LR) as an example to quantify the quality of the evolved transformation procedure. Empirical evidences, collected from benchmark functions and real-world problems, demonstrate that the proposed transformation approach is able to dramatically improve LR generalisation and make it outperform other state-of-the-art regression models such as Genetic Programming, Kriging, and Radial Basis Functions Networks. In addition, we present an analysis to shed light on the most important statistical moments that are useful for the transformation process.**

I. Introduction

Regression is a common task in Machine Learning with a variety of applications in science and engineering. Generally, regression problems can be formalised as follows. Let χ denote the input space that is usually viewed through a finite set of n observational samples $X = \{x_0, x_1, ...x_n\}$ where each $x_i \in \Re^k$ denotes a real-valued random input vector, and $Y = \{y_0, y_1, ...y_n\}$ is the set of corresponding outputs such that $y_i \in \Re$ is a real-valued random output variable, with joint distribution $P(X, Y)$. To this end, all regression models are expected to receive a labelled sample $S = \{(x_0, y_0), (x_1, y_1), ..., (x_n, y_n) \in (X, Y)^n\}$. Since the labels are real-numbers it is very difficult to expect the regression model to predict exactly the correct output. Therefore, it is common to tolerate a range of errors. In order to quantify the quality of the regression model a loss function $L(Y, f(X)) = (Y - f(X)^2)$ is required. The aim is to construct a model $f : X \rightarrow Y$. Hence, given a hypothesis set $\mathbf{H}$ of functions to map X to Y, the regression problem consists of using the samples in S to find a hypothesis $h \in \mathbf{H}$ with small loss (or generalisation error) denoted by:

$$\hat{R}(h) = \frac{1}{n} \sum_{i=0}^{n} L(y_i, f(x_i))$$

In principle, one can make very accurate predictions if S is large enough. For example, using nearest-neighbour methods [1] [9] a prediction for any query point $x_q \in \Re^k$ can be made by finding the average responses of the p closest samples. Closeness implies a metric, which normally is assumed to be the Euclidean distance.

If reasonably large S is available, one could always approximate the theoretically optimal expectation by nearest-neighbour averaging. Since it is possible to find a fairly large neighbourhood of observations close to any x_q, this approach breaks down in high dimensions, and the phenomenon is commonly referred to as the *curse of dimensionality* [9]. In a high-dimensional input space, the distance metric becomes hard to quantify.

In [9], the author gave an example when considering the nearest-neighbour procedure for inputs uniformly distributed in a k-dimensional unit hypercube. In order to extract a hypercubical neighbourhood around a query point we need to capture a fraction r of the observations to make a prediction. A fraction r of the unit volume, the expected edge length will be $e_p(r) = r^{(1/p)}$ [9]. In ten dimensions $e_{10}(0.01) = 0.63$ and $e_{10}(0.1) = 0.80$, while the entire range for each input is only 1.0 [9]. So, to capture 1% to 10% of the data to form a local average, we must cover 63% to 80% of the range of each input variable [9]. This increases the size of the neighbourhood significantly. Also, in many real-world problems S is normally is limited to a small set of observations in which the number of dimensions k is much larger than the number of observations n, often written as $k \gg n$.

In this paper we propose a novel approach to mitigate this problem by transforming the input vectors in X into new smaller vectors $Z = \{z_0, z_1, ..., z_n\}$ where each $z_i \in \Re^g$ and $g < k$. For the transformation procedure we use only a set of simple statistical moments. Genetic Algorithm (GA) has been used to evolve a transformation procedure to transform

Ahmed Kattan with Um Al Qura University, AI Real-World Applications Lab, Department of Computer Science, Kingdom of Saudi Arabia, Michael Kampouridis with University of Kent, School of Computing, United Kingdom, Yew-Soon Ong with the School of Computer Engineering, Nanyang Technological University, Singapore, and Khalid Mehamdi with Um Al Qura University, AI Real-World Applications Lab, Department of Computer Science, Kingdom of Saudi Arabia. email: Ajkattan@uqu.edu.sa, M.Kampouridis@kent.ac.uk, asysong@ntu.edu.sg, kmmehmadi@uqu.edu.sa.

the original input space χ into a new space ξ. GA is used to search for an optimal sequence of statistical moments and their input parameters. We used Linear Regression (LR) as an example to quantify the quality of the evolved transformation procedure (more details in Section III). The contributions of the paper can be formalised as three-folds:

1) We propose a novel approach to transform the high-dimensional input space of regression models using only statistical moments.
2) We provide an analysis to understand the impact of different statistical moments on the evolved transformation procedure.
3) We dramatically improve LR's generalisation and make it competitive to other state-of-the-art regression models such as Genetic Programming (GP), Kriging, and Radial Basis Functions Networks (RBFN).

The remainder of this paper is structured as follows. Section II briefly presents some related works. Section III provides a detailed explanation of the proposed approach. Section IV provides the experimental results and their analysis. Finally, Section V provides some conclusive remarks and set the directions for future research.

II. Related Works

Dimensionality reduction techniques to mitigate the curse of dimensionality problem is a well-explored topic. Many techniques have been developed and used with feature selection and classification problems (e.g., [15], [5]). However, the idea of evolving a transformation procedure to reduce the number of design variables in the regression problems to improve generalisation is relatively little explored thus far. In this section we focus the review on dimensionality reduction and transformation approaches for regression models since these are directly relevant to the work reported in this paper.

Sobester and Nair in [16], presented a GP approach for generating functions in closed analytic form that map the input space of a complex function approximation problem into one where the output is more amenable to linear regression. To achieve this, the authors used a co-evolutionary approach where multiple populations are evolved in parallel. Each population evolves part of the solutions. The system collects the best individual in each evolved population to form a new transformed input vector **z**. The i^{th} element of evolved **z** vector is an output of an evolved function that received the i^{th} input from the original input vector. The proposed approach was evaluated with several benchmark functions and real-world problems. However, the authors claimed that their results are not conclusive and that they merely serve as proof of concept. In addition, the new transformed input vector **z** has the same dimensionality as the original vector.

In [7], the authors presented a GP-based approach for symbolic regression of discontinuous functions in multivariate data sets. The idea is to identify the portions of the input space that require different approximating functions by means of an algorithm referred to as Hyper-Volume Error Separation (HVES). To this end, a preliminary GP evolution run is used to partition the input space based on the error exhibited by the best individual across the training set. The training set is partitioned several times into smaller groups. Although the authors claimed that their approach, in principle, can work with multivariate data-sets, their experiments covered problems of two variables.

In [12], the authors proposed a technique based on latent variables, non-linear sensitivity analysis, and GP to manage approximation problems when the number of input variables is high. The proposed technique was tested with 340 input variable problems. The proposed approach was designed to consider problems where all input variables have similar influence on the model's output. Thus, standard variable pruning techniques are not applicable.

McConaghy [13], presented a deterministic technique, referred to as Fast Function Extraction (FFX), for solving a symbolic regression problem that achieves higher approximation accuracy than standard GP and several state-of-the-art regression techniques. FFX generates a set of basis functions, where each function can be a linear or non-linear combination of the input design variables. Once FFX generates a set of possible basis functions it assigns the best coefficients for each function. FFX execution takes only a few seconds to return simpler models from a large number of design variables. The authors verified FFX on a broad set of real-world problems with different numbers of variables ranging from 13 to 1468. Later, Icke and Bongard [10] hybridised FFX and GP to create an improved learner for symbolic regression problems. In their work, the authors showed that a hybrid deterministic/GP for symbolic regression outperforms GP alone and several state-of-the-art deterministic regression techniques alone on a set of multivariate polynomial symbolic regression tasks. The proposed approach was tested to approximate data-sets of different dimensionality, ranging from 1 to 25 dimensions.

Recently, Kattan et al. [11], proposed a new approach based on GP to transform the original input space into a new input space that has smaller input vectors that are easier to be mapped into their corresponding responses. To achieve this, GP is designed to evolve several non-linear transformation equations that extract different statistical features from different intervals of the original input vectors. Each equation is generated from a different sub-tree in an individual, thus, each tree in the population produces multiple outputs (i.e., transformed output).

TABLE I

STATISTICAL MOMENTS USED IN THE TRANSFORMATION PROCEDURE

Function	Input	Output
Mean, Median, StD, Variance, Geometric Mean, Average Div, Min, Max Copy, Copy $\times$ Intercept	Randomly selected variables from each $x_i \in X$	Real Number

**StD* is Standard Deviation, and *Average Div* is Average Deviation
**Copy* is an operator that copies selected variables without change into the transformed vector
**Copy $\times$ Intercept* is a transformation operator that multiply each selected variable with the intercept value of LR.

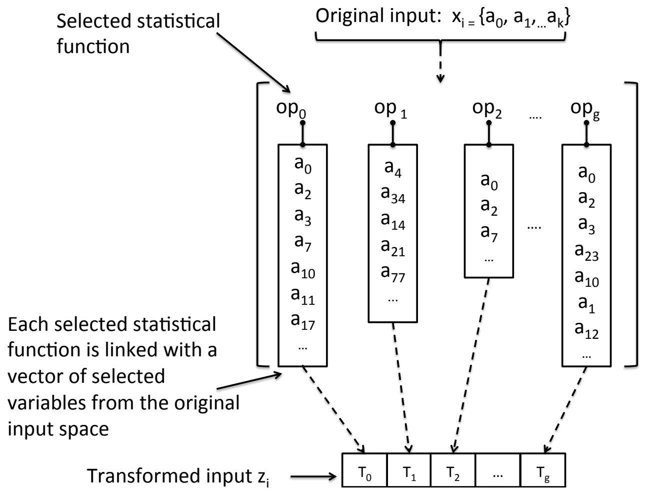
As can be seen, most of the previous works tried to mitigate the curse of dimensionality problem for regression models by transforming the input space into a new input space using linear or non-linear transformation functions. In this paper we show that it is possible to mitigate the curse of dimensionality problem and improve the generalisation by transforming the input space into a new space using only simple statistical moments. This allows the transformed input not only to be smaller but we also to share similar statistical characteristics as the original input space and thus relaxes the learners' complexity.

III. EVOLVE TRANSFORMATION PROCEDURE

The idea is to find a transformation procedure using only a set of primitive statistical features. Thus, we don't only make the transformed input smaller but also make it shares similar statistical characteristics as the original input space and thus relax the learners' complexity. Ideally, one would like to find a single universal transformation procedure that works well across different regression problems. However, this may be extremely difficult given that each regression problem has a unique surface. So, in this paper, we used an evolutionary approach, namely GA, to find a suitable transformation for the given problem. Table I illustrates the set of statistical moments used in the proposed transformation procedure. To this end, GA evolves a transformation procedure that receives input variables from each $x_i \in X$ (note that $X \subset \Re^k$) and transforms it into z_i. The last result is a set $Z = \{z_0, z_1, ..., z_n\}$ where $Z \subset \Re^g$ and $g < k$.

A. GA Representation

As illustrated in Figure 1, GA individuals, are encoded as a sequence of integers to represent a set of selected statistical moments (from the pool of moments in Table I) and each selected statistical moment is linked with a set of selected variables from the original input space (variables are not allowed to be duplicated within the same set). In our representation, we allowed GA individuals to be of variable sizes. Evaluation of individuals will result in a vector of transformed input where each item holds a single number



Selected statistical function
Original input: $x_i = \{a_0, a_1, ... a_k\}$
op_0
op_1
op_2
....
op_g
a_0 a_2 a_3 a_7 a_{10} a_{11} a_{17} ...
a_4 a_{34} a_{14} a_{21} a_{77} ...
a_0 a_2 a_7 ...
a_0 a_2 a_3 a_{23} a_{10} a_1 a_{12} ...
Each selected statistical function is linked with a vector of selected variables from the original input space
Transformed input z_i
T_0 T_1 T_2 ... T_g

Fig. 1. GA individuals' representation.

that represent an abstract of some selected variables from the original input space. For example, a GA individual may transform each original input vector into a single number that represents the mean of some selected variables or it could transform each input vector into two numbers where one represents the mean of some variables and the other represents the variance of some other variables, or it could even transform the each input vector into g numbers where each number represents a statistical feature of some selected variables.

Since we allow GA individuals to be of variable sizes, search operators can shrink or extend individuals. We restrict the maximum size of the individuals to be less than the the number of variables in the original input space. In other words, we do not allow the dimensionality of the transformed space to be bigger than the original space.

The designed GA representation is able to process all data samples in the space $\chi \subset \Re^k$, viewed through the set X, and transform them into a new space $\xi \subset \Re^g$, viewed through the set Z.

B. GA Fitness Function

In order to measure the effectiveness of GA individuals we need to find out whether they will improve regression models performance. This is a challenging problem because the fitness measure needs to be aware of the generalisation level induced by the transformed space. We used average prediction errors of LR as a fitness measure for GA. LR is a very simple algorithm where it considers the family of linear hypotheses:

$$\mathbf{H} = \{x \rightarrow w.\Phi(x) + b : w \in \Re^k, b \in \Re\}$$

and seeks a hypothesis in $\mathbf{H}$ with the smallest loss function. Generally, LR is known to give accurate predictions if the sample inputs are linearly aligned with their corresponding outputs. Moreover, LR is known to perform better when the number of dimensions is limited. Hence, given these features LR can push the GA's evolutionary process to linearly align the transformed inputs with their outputs and minimise the dimensionality of the new space. The GA aims to minimise the following fitness function:

$$fitness = \frac{\sum_{i=0}^{n}(|y_i - LR(z_i)|)}{n} \tag{1}$$

where z_i represents a transformed input vectors.

C. GA Search Operators

The search operators are designed to maintain proper syntax of the individual's representation. Here, we consider three operators to explore the search space. First, we used a crossover operator in which two individuals exchange statistical moments and their parameters, randomly. Second, there is an aggressive mutation operator that replaces a statistical moment and its parameters, randomly selected, with another randomly selected moments from the pool of statistical moments. This aggressive mutation operator allows individuals to shrink or extend. Finally, we used a smooth mutation operator where a parameter of a randomly selected statistical moment is mutated into a new parameter. Note that parameters of a single statistical moment are not allowed to be duplicated.

D. GA Training

GA runs for a fixed number of generations where individuals are evaluated as explained in Equation 1. Before the run starts we generate two disjoint sets: training and validation. The training set is used to evaluate individuals where LR uses a two-folds cross-validation approach. The best individual in each generation is further tested with the validation set. Finally, we select the individual that yields the best performance on the validation set across the run. For the GA, we set the number of generations and population size to 50, crossover, mutation, and aggressive mutation rates were set to 0.7, 0.1, and 0.2, respectively. Finally, we used standard tournament selection of size 2.

One may argue that the proposed GA approach eventually does a simple variable selection and especially that it uses some selective moments such as Min and Max as well as a copy operator (as illustrated in Table I). In order to investigate this assumption, in preliminary experiments, we compared the proposed approach against variable selection approach based on standard GA (that uses a binary representation). We found that our proposed approach achieved far better results, which verifies that it does more than a simple variable selection.

IV. Empirical Tests and Analysis

A. Settings

A set of experiments has been conducted to evaluate the proposed approach. We tested the effects of the transformation procedure on LR and compared the results against five regression models, namely, RBFN, Kriging, LR, piecewise LR [6], and GP. These models were selected because they are some of the most important techniques in the literature. Piecewise LR is basically dividing the input samples into groups based on their Euclidean distance and training a single LR model for each group, thus, in principle, it allows LR to approximate non-linear surfaces. For the GP, we used similar settings as the GA (see Section III-D), except that we used standard sub-tree crossover at 0.7 and standard sub-tree mutation at 0.3. In addition, GP was equipped with standard arithmetic operators and 10 random constants from the range $[-1, 1]$. Finally, to compare the proposed approach against standard dimensionality reduction technique we included Principle Component Analysis (PCA) [2] in the experiments.

Experiments included the following five benchmark functions; *F1 = Rastrigin, F2 = Schwefel, F3 = Michalewicz, F4 = Sphere,* and *F5 = Dixon & Price* [14]. For each test function, we trained all regression models to approximate the given function when the number of variables is 100, 500, and 1000. The total number of benchmark test problems is 15 (i.e., 5 test functions $\times 3$ different variables sizes). For all test problems, we randomly generated three disjoint sets: a training set of 100 points, a validation set of 50 points, and a testing set of 150 points from the interval $[-1, 1]$. All techniques have been compared based on the average of absolute errors on the testing set.

In addition, we included three additional real-world problems. The problems we used for this set of experiments comes from the field of financial forecasting. In this area, traders believe that patterns exist in historical data and that these patterns will repeat themselves in the future. Consequently, it is worth identifying these patterns, so that we can exploit them in the future and make a profit. A very common way of doing this is by using the method of technical analysis [4]. Technical analysis uses indicators, which are formulas that measure different aspects of a given financial dataset, such as trend, volatility, and momentum.

An example of such indicators is the Moving Average (MA), which calculates the averages of a given dataset under sliding windows of a fixed length L. For our experiments, we are using 6 different indicators (Moving Average, Trade Break Out, Volatility, Filter, Momentum, and Momentum Moving Average), with 3 different L values for each indicator: $100, 500$, and 1000. Thus, each experiment has 6×100, 6×500, and 6×1000 variables, respectively. The data were divided into three sets as follows: 40% for training, 10%

validation, and 50% testing.

B. Results

1) Benchmark Results: For the benchmark functions, Table II shows improvements of predictions after testing our proposed transformation procedure with LR (referred to as *LR+Z* in the table) in comparison to all nine other algorithms. Note that the transformation procedure is coming from a stochastic evolutionary process, therefore, we run the GA 20 independent times and use the best evolved transformation in the comparison. To assure a fair comparison, we also run both GP and GP+PCA 20 times for each problem and report their best results. Each time GA evolves a transformation procedure, GP runs for 5 independent times and the best result is reported. As it can be seen from Table II, LR+Z remarkably outperforms all competitors. The non-parametric Friedman test in Table III ranked LR+Z first with a rank of 1.0, while the second and third ranking algorithms were GP and LR+PCA, with a ranking of 2.07 and 4.74, respectively. Subsequent analysis of the post-hoc Bonferroni-Dunn test [3], [8] found that the LR+Z's ranking was significant at 5% level when compared to seven of the nine algorithms from our experiments (the only exception was the GP). This is an important result, because it confirms our approach's superiority, across a number of other state-of-the-art algorithms.

Furthermore, Table IV presents a summary of the dimensionality of the transformed space. Interestingly, there seems to be no clear relation between number of dimensions in the transformed space and the original space. For all three dimensions in the comparison, GA managed to find a single dimension (abstracted from several variables), using only one statistical moment, to represent the new space. Although, GA managed to achieve this dramatic abstraction in the dimensions, we noted that this does not necessarily lead to producing the best prediction results. For example, Figure 2 depicts two instances of a 1D-transformed inputs for two different problems.

2) Learning from Evolution: Because the performance of the evolved transformation procedures is good, in most cases, it would be interesting to understand what evolved solutions actually do. When looking at the evolved sequences of moments, one quickly realises that they are not easy to understand them or to capture a clear relation. Therefore, we visually project statistical moments in heat maps according to their contribution to good solutions for each problem, in Figure 3. The idea of this figure is inspired from the work of Smits et al. [15]. If a statistical moment is absolutely essential to produce a good transformation procedure then it must be present in individuals with good fitnesses. Other less essential statistical moments may be present in both good individuals and inferior ones, so their fitness will be closer to the average fitness over all individuals. To this end,

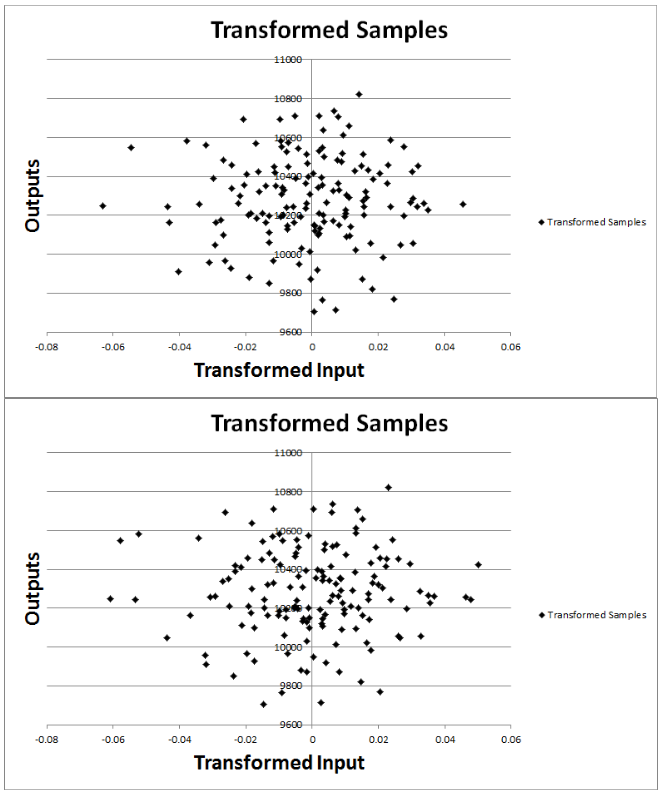


Fig. 2. Sample of transformed testing set using an evolved transformation procedure. The upper figure shows a transformation of F1, D1000 and the lower figure shows a transformation of F2, D500. LR+Z error of the upper and lower figures are 188.404 and 5.13, respectively

in order to rank the importance of statistical moments, we equally distribute the fitness value of each individual over all moments present in that individual and project these values in heat maps as presented in Figure 3. It is clear from the heat maps that each problem has its unique characteristics. However, interestingly, there is a consensus among all maps that the operators *copy* and *copy* $\times$ *intercept* do not contribute to the construction of good transformation procedures. Also, all maps agree that the *Average Deviation, Geometric Mean, Min* and *Max* are important across all problems. We still do not have a full understanding of the effect of these moments on the transformed space. In future research we will focus on this aspect.

3) Real-world Problems: For the three real-world problems included in the experiments, we added a new algorithm in the comparison. Namely, GP with random training sub-set selection (referred to as GP+RTS). Random training sub-set selection is a common technique used in the GP literature to overcome the over-fitting problem. The idea is to randomly select a different sub-set of training samples in each generation. We conducted two sets of experiments for the real-world problems. For the first set, we eliminated non-essential statistical moments as suggested by the heat maps

TABLE II

SUMMARY OF RUNS ON 15 BENCHMARK PROBLEMS

Algorithm	RBFN	RBFN+PCA	Kriging	Kriging+PCA	LR	LR+Z	Piecewise LR	LR+PCA	GP	GP+PCA
F1, D = 100	8.1E+01	8.4E+01	8.1E+01	8.1E+01	1.5E+03	**5.8E+01**	1.2E+06	8.1E+01	7.8E+01	8.4E+01
F1, D = 500	1.9E+02	2.1E+02	1.9E+02	1.9E+02	1.8E+06	**1.2E+02**	2.5E+08	1.9E+02	1.8E+02	1.9E+02
F1, D = 1000	2.3E+02	2.3E+02	2.3E+02	2.3E+02	8.0E+06	**1.7E+02**	3.7E+08	2.3E+02	2.3E+02	2.3E+02
F2, D = 100	2.2E+00	2.5E+00	2.2E+00	3.1E+00	6.3E+02	**2.0E+00**	2.4E+03	2.2E+00	2.1E+00	2.2E+00
F2, D = 500	4.6E+00	4.6E+00	4.6E+00	4.7E+00	4.7E+04	**4.4E+00**	1.5E+04	4.6E+00	4.5E+00	4.6E+00
F2, D = 1000	2.4E+04	3.7E+04	2.4E+04	2.5E+04	1.9E+08	**1.6E+04**	5.8E+09	2.5E+04	2.4E+04	2.4E+04
F3, D = 100	3.7E+00	3.5E+00	3.7E+00	1.5E+01	2.7E+05	**3.0E-01**	4.7E+07	3.7E+00	3.7E+00	7.2E+00
F3, D = 500	7.5E+00	8.3E+00	7.5E+00	9.4E+00	4.2E+08	**8.8E-01**	5.8E+08	7.6E+00	7.4E+00	1.8E+01
F3, D = 1000	1.2E+01	1.3E+01	1.2E+01	1.3E+01	1.9E+08	**9.5E-01**	4.3E+09	1.2E+01	1.1E+01	4.9E+01
F4, D = 100	2.5E+00	2.9E+00	2.5E+00	2.2E+01	6.4E+01	**3.2E-01**	1.6E+04	2.5E+00	2.4E+00	2.4E+00
F4, D = 500	5.5E+00	6.4E+00	5.5E+00	5.6E+01	2.7E+04	**3.6E-01**	1.4E+06	5.5E+00	5.4E+00	5.5E+00
F4, D = 1000	8.3E+00	1.0E+01	8.3E+00	5.5E+01	2.8E+05	**8.2E-01**	1.4E+07	8.3E+00	8.1E+00	8.4E+00
F5, D = 100	7.7E+02	1.1E+03	7.7E+02	8.6E+02	5.6E+04	**5.0E+02**	4.8E+06	7.7E+02	7.7E+02	7.8E+02
F5, D = 500	9.4E+03	1.0E+04	9.4E+03	9.7E+03	1.5E+08	**5.3E+03**	1.0E+09	9.4E+03	9.3E+03	9.5E+03
F5, D = 1000	2.4E+04	3.7E+04	2.4E+04	2.5E+04	1.9E+08	**5.5E+03**	1.0E+09	2.5E+04	9.3E+03	2.4E+04

* **Bold** numbers are the lowest.

TABLE III

AVERAGE RANKINGS OF THE ALGORITHMS FOR BENCHMARK PROBLEMS

Algorithm	Ranking
RBFN	4.30
RBFN+PCA	6.67
Kriging	4.57
Kriging+PCA	7.13
LR	9.07
LR+Z	1.00
LR+Cluster	9.94
LR+PCA	4.74
GP	2.07
GP+PCA	5.54

TABLE IV

SUMMARY OF DIMENSIONALITY OF TRANSFORMED SPACE

	D = 100				D = 500				D = 1000			
Function	Mean	Best	Median	StD	Mean	Best	Median	StD	Mean	Best	Median	StD
F1	4.25	1.00	4.00	2.19	7.95	3.00	4.50	5.95	52.10	1.00	7.50	190.73
F2	5.30	1.00	5.00	2.59	7.65	7.65	5.50	6.51	12.06	2.00	5.00	11.26
F3	16.55	5.00	16.00	7.60	10.55	1.00	11.50	5.37	64.00	9.00	10.00	214.65
F4	6.00	1.00	5.50	3.33	10.60	3.00	9.00	5.63	802.60	17.00	999.00	392.80
F5	8.25	3.00	8.50	2.98	6.90	3.00	6.00	2.90	10.70	1.00	5.00	10.90

TABLE V

SUMMARY OF RUNS ON 3 REAL-WORLD PROBLEMS

Algorithm	RBFN	RBFN+PCA	Kriging	Kriging+PCA	LR	LR+Z	Piecewise LR	LR+PCA	GP	GP+PCA	GP+RTS
D = 100	24.41	178.91	24.50	24.59	22.65	**18.88**	278.39	24.48	21.09	22.18	21.15
D = 500	16.32	18.66	16.32	16.32	79.12	**8.22**	432.01	9.70	9.70	12.70	10.15
D = 1000	16.32	16.36	16.32	16.34	33540.40	**6.04**	1887.49	16.04	8.70	11.95	8.22

* **Bold** numbers are the lowest.

(See section IV-B.2). For the second set, we included all statistical moments. We found that the evolution has been accelerated in the first experimental set in comparison to the second set of experiments. However, as this was not consistent across the three different dimensions, we decided to report the results of the second experimental set only. We leave it as a future work to further look into this.

Table V presents the results. Similar to the previous experiment set, for each problem we run GA 20 times and report the performance of the best evolved transformation procedure. Also, each of GP, GP+PCA and GP+RTS received the same number of runs, to assure fairness. Each time GA evolves a transformation procedure all GP systems run 5 independent times and return the best results. It is clear from the table that LR+Z comes in first place in all three problems. Results of GP is little competitive.

The above findings are confirmed by the algorithm's ranking across the three different dimensions, according

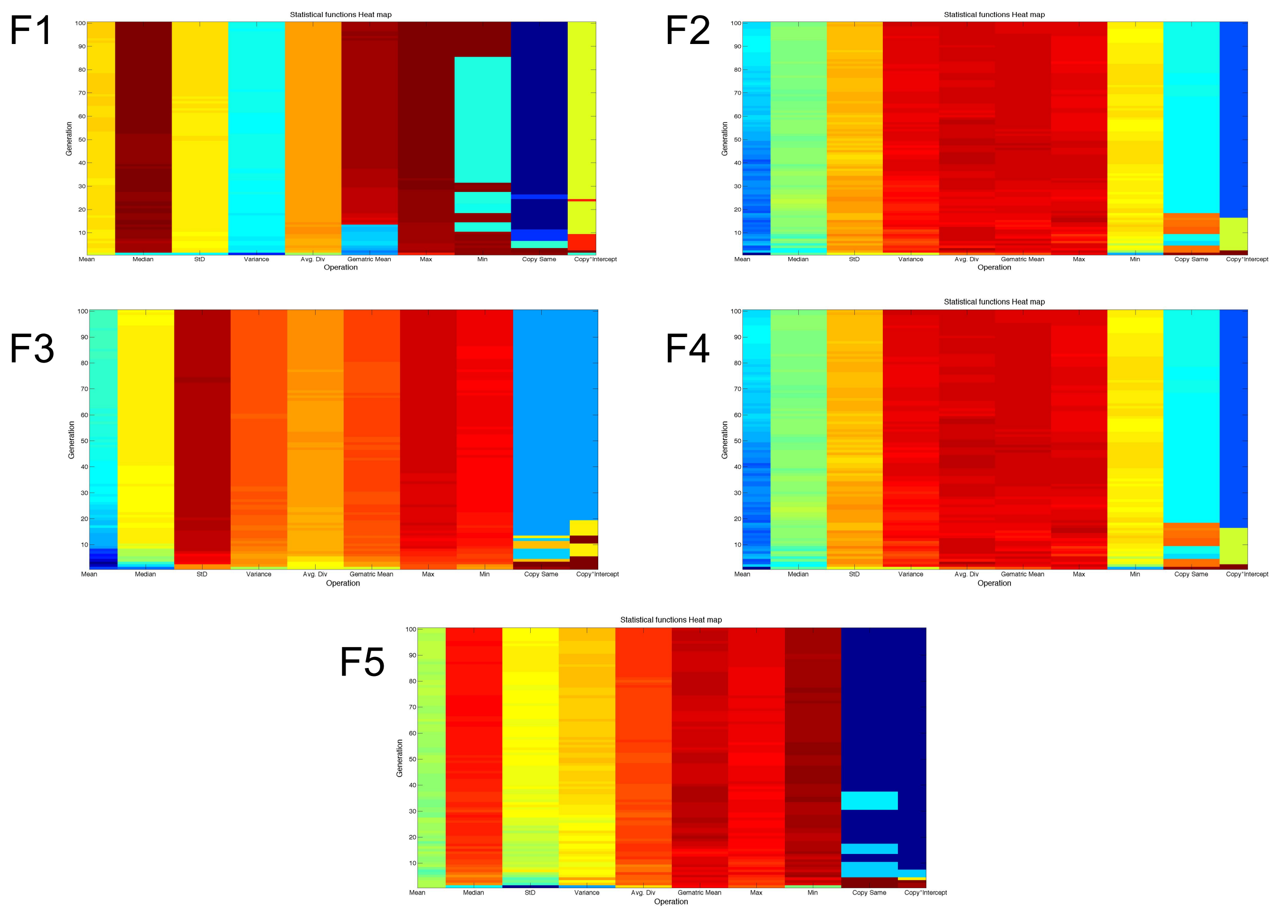


Fig. 3. Heatmaps of statistical moments' importance in terms of their contribution to individuals' fitnesses. Each map is averaged from 60 independent runs (20 runs for each test dimension). The statistical moments, from left to right: Mean, Median, Standard Deviation, Variance, Average Deviation, Geometric Mean, Max, Min, Copy Same, Copy Intercept.

TABLE VI

AVERAGE RANKINGS OF THE ALGORITHMS FOR REAL-WORLD PROBLEMS

Algorithm	Ranking
RBFN	6.33
RBFN+PCA	9.33
Kriging	7.67
Kriging+PCA	7.67
LR	8.67
LR+Z	1.00
LR+Cluster	10.67
LR+PCA	4.83
GP	2.50
GP+PCA	4.33
GP+RTS	3.00

to the Friedman test presented in Table VI. As we can observe, LR+Z is ranked first again with a rank of 1.0, with the GP coming second with a rank of 2.5. The post-hoc test this time, however, only found LR+Z significantly better than LR+Cluster, RBFN+PCA, and LR. The remaining differences were not significant at the 5% level. However, this is not unexpected, as the number of problems we experimented with in the real-world was very low, only three (a single dataset under three different dimensions, 100, 500, and 1000 have been considered). Nevertheless, the fact remains that LR+Z was ranked first among all algorithms in the comparison, and it also consistently outperformed its competitors in terms of Mean and Median results, as can be observed from Table VII.[1] We leave the investigation of more datasets from the real-world as a future work.

Finally, Table VIII presents a statistical summary of the dimensionality of the transformed space. As we can observe, the GA has managed to decrease the input space significantly.

V. CONCLUSIONS

This paper presents a novel approach to mitigate the curse of dimensionality for regression problems. The idea is based on transforming the input vectors of the data samples into new smaller vectors (called Z set). This is unlike other

[1]We present only LR+Z, GP, and GP+RTS, as these are the only algorithms that follow a stochastic process, thus it is possible to calculate the Mean and Median values only for these algorithms.

TABLE VII

COMPARISON SUMMARY OF 20 INDEPENDENT LR+Z-SET, GP AND GP+RTS.
(FOR EACH EVOLVED Z SET EACH OF GP AND GP+RTS RUNS 5 TIMES AND REPORT THE BEST RESULT)

NOTE: IN TOTAL EACH OF GP AND GP+RTS RANS FOR 20×5 TIMES

Algorithm	LR+Z	GP	GP+RTS
	D = 100		
Average	**20.30**	21.69	21.93
Best	**18.88**	21.09	21.15
Median	**20.30**	21.60	21.91
StD	1.12	0.57	0.62
	D = 500		
Average	**10.16**	12.56	12.20
Best	**8.22**	9.70	10.15
Median	**10.31**	12.02	11.94
StD	1.11	2.23	1.53
	D = 1000		
Average	**7.75**	12.39	11.89
Best	**6.04**	8.70	8.22
Median	**6.65**	11.52	11.31
StD	2.30	3.20	2.17

* **Bold** numbers are the lowest.

TABLE VIII

STATISTICAL MOMENTS USED IN THE TRANSFORMATION PROCEDURE

	Measure			
Dimensions	Mean	Best	Median	StD
100	47.65	9.00	53.00	18.04
500	46.25	1.00	28.00	61.81
1000	28.70	2.00	6.00	52.81

existing works where transformation of input space is done using linear or non-linear transformation functions. In this paper we showed that it is possible to transform the input space into new space using only simple statistical moments. GA has been used to evolve a transformation procedure. GA is used to optimise an optimal sequence of statistical moments and their input parameters. We used LR as an example to quantify the quality of the evolved transformation procedure. Empirical evidences, collected from 18 different benchmark and real-world problems, demonstrate that the proposed transformation approach is able to dramatically improve LR generalisation and make it outperform other state-of-the-art regression models such as GP, Kriging, and RBFN.

The contributions of this paper can be formalised as follows:

1) We propose a novel approach to transform the high-dimensional input space of regression models using only statistical moments.
2) We provide an analysis to understand the impact of different statistical moments on the evolved transformation procedure.
3) We dramatically improve LR's generalisation.

For future work, we will try to understand the effect of different statistical moments on the transformed space. Also, we will explore the idea of making the GA's search space to be adaptive by pruning non-essential statistical moments based on their importance in terms of the contribution to good individuals.

REFERENCES

[1] E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2004.

[2] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[3] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[4] R. Edwards and J. Magee. Technical analysis of stock trends. *New York Institute of Finance*, 1992.

[5] C. Estébanez, R. Aler, and J. M. Valls. Genetic programming based data projections for classification tasks. 2005.

[6] G. Ferrari-Trecate and M. Muselli. A new learning method for piecewise linear regression. In *Artificial Neural Networks ICANN 2002*, pages 444–449, 2002.

[7] C. Fillon and A. Bartoli. Symbolic regression of discontinuous and multivariate functions by hyper-volume error separation (hves). In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 23–30. IEEE, 2007.

[8] S. Garcia, F. Herrera, and J. Shawe-Taylor. An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[9] T. Hastie, R. Tibshirani, and J. J. H. Friedman. *The elements of statistical learning*, volume 1. Springer New York, 2001.

[10] I. Icke and J. Bongard. Improving genetic programming based symbolic regression using deterministic machine learning. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1763–1770, 2013.

[11] A. Kattan, M. Kampouridis, and A. Agapitos. Generalisation enhancement via input space transformation: A gp approach. In *14th European Conference on Evolutionary Computation in Genetic Programming*, page in press. Springer, 2014.

[12] T. McConaghy. Latent variable symbolic regression for high-dimensional inputs. In *Genetic Programming Theory and Practice VII*, pages 103–118. Springer, 2010.

[13] T. McConaghy. Ffx: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*, pages 235–260. Springer, 2011.

[14] M. Molga and C. Smutnick. Test functions for optimization needs. *Test functions for optimization needs*, 2005.

[15] G. Smits, A. Kordon, K. Vladislavleva, E. Jordaan, and M. Kotanchek. Variable selection in industrial datasets using pareto genetic programming. In T. Yu, R. L. Riolo, and B. Worzel, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 6, pages 79–92. Springer, Ann Arbor, 12-14 May 2005.

[16] A. Sobester, P. Nair, and A. Keane. Evolving intervening variables for response surface approximations. In *Proceedings of the 10th AIAA/ISSMO multi-disciplinary analysis and optimization conference*, pages 1–12. American Institute of Aeronautics and Astronautics, 2004. AIAA 2004-4379.