Multiobjective Genetic Programming Feature Extraction with Optimized Dimensionality

Yang Zhang and Peter I Rockett

Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S1 3JD, UK. hegallis@gmail.com,p.rockett@shef.ac.uk

Abstract. We present a multi-dimensional mapping strategy using multiobjective genetic programming (MOGP) to search for the (near-)optimal feature extraction pre-processing stages for pattern classification as well as optimizing the dimensionality of the decision space. We search for the set of mappings with optimal dimensionality to project the input space into a decision space with maximized class separability. The steady-state Pareto converging genetic programming (PCGP) has been used to implement this multi-dimensional MOGP. We examine the proposed method using eight benchmark datasets from the UCI database and the Statlog project to make quantitative comparison with conventional classifiers. We conclude that MMOGP outperforms the comparator classifiers due to its optimized feature extraction process.

1 Introduction

Normally, the discriminability between classes in a pattern space can be enhanced by feature extraction to obtain more desirable classification performance. Feature extraction as a pre-processing step in a pattern recognition system needs to target the extraction of discriminatory information from the input patterns in order to optimize the classification performance. This is implemented by projecting the pattern space into a new *decision space* in which the discriminability between classes is maximized.

Although a large amount of research has focused on designing ever-more sophisticated and powerful classifiers, such as support vector machines and neural networks, feature extraction has remained an important step in the design of pattern recognition systems. Due to the lack of a generic feature extraction methodology, conventional feature extraction is usually problem-specific and designed largely on the basis of domain-dependent knowledge or assumptions about class distributions. Some general purpose methods such as principal component analysis (or the Karhunen-Loève transform) are mostly constrained to linear transformations. Non-linear mappings like supervised learning networks (*e.g.* multilayer perceptrons and radial basis functions) provide model-free or semi-parametric methods to design non-linear mappings/discriminant functions but in such paradigms, the feature selection/extraction process is subsumed into a single entity within the classifier. Hence the optimality is hard to guarantee.

Depending on the criteria used, feature extractions can be grouped into two categories: representational and classificational. In the first approach, the objective is to maintain the fidelity between the original input and projected spaces. The major concern of these methods is to maintain the discriminatory information of the original input space during what is typically, dimensionality reduction. For the classificational approach, the objective is to enhance the discriminability between classes in the new projected decision space. The present work is targeted at optimizing the discriminability in the transformed space. We show that the determination of an optimal multi-dimensional mapping does not necessarily imply dimensionality reduction.

Typically, a prospective solution in GP can be interpreted as a sequence of operations [1]. Langdon [2] developed a multi-tree program with redefined genetic operations to reduce the extent of building block disruption; see also Koza's work involving automatically defined functions (ADFs) [1]. The CoacH system [3] specifies multiple trees within a single individual in the GP population. Sherrah [4] has attempted to address the issue of generating $n-$to$-m$ mappings $(m > 1)$ by applying a multi-tree chromosome structure but had to impose some severe restrictions to compensate for the limitation of his single objective function. Bot [6] described a hybrid algorithm using GP to evolve a (*near-*)optimal feature extraction stage and added extra transformed features, one-at-a-time. Each new feature was retained only if it yielded an improvement in classification error above a pre-defined threshold. Bot's approach is a greedy algorithm and hence suboptimal. Muni *et al.*. [5] applied a multi-tree representation to produce an q-class classifier $(q > 2)$ by simultaneously evolving q independent dichotomizers. Recent work on GP-evolved feature extraction/classification has been reviewed in [8] where two distinct research strands have been identified: Either GP is used to evolve the whole classifier or the tree produces a mapping to a real-valued decision space which forms the input to a conventional classifier.

Zhang & Rockett [7][8] proposed a generic framework to produce optimal feature extractors independent of domain-specific knowledge and class distributions using multiobjective genetic programming (MOGP). Through a multiobjective optimization process, their 1D mappings comprised a series of mathematical transformations projecting input patterns into a one-dimensional decision space. Classification performance was markedly enhanced. In this paper, we seek to extend that work on evolving feature extraction stages to produce $n-$to$-m$ mappings of optimal dimensionality. For a given classification problem, generally an optimal value of m exists due to the well-known *peaking phenomenon* [17] – here m becomes a free parameter within the optimization. In Section 2 we describe our method to find the (*near-*)optimal set of m extracted features. In order to substantiate the effectiveness of the proposed method, in Section 3 we report results of quantitative comparison with eight conventional classifiers on two groups of datasets from the UCI and Statlog databases. After analysis of the multi-dimensional feature extraction process, we offer conclusions in the final section.

2 Multi-Dimensional Feature Extraction

To distinguish between the present work and our earlier investigations using a 1-dimensional decision space [7], we term the present method *multi-dimensional multiobjective genetic programming* (MMOGP). As in [7], we optimize a vector of multiple objectives within a Pareto framework.

2.1 The Objective Vector

2.2 Single Tree Representation

We have used a single, vectorizing tree representation in this feature extraction application. Note that the single tree representation scheme used here differs from the multiple-output structure of *MRtree* [15] and Sherrah's *EPrep* [4] approach in that the output of a candidate tree from MMOGP is a multi-dimensional feature vector in the projected decision space, not a set of tentative class labels as in *MRtree*. Furthermore, *MRtree* results are very sensitive to the defined *Modi* parameter [15]. In *EPrep*, the output of the evolved feature pre-processing unit is formed by a special function node – an *Output Point* - operating on the tree [4]. One of the disadvantages of *EPrep* is that the features extracted by Output Points from the lower levels of the tree may be severely correlated with those collected at higher levels. Notably, *EPrep* is able to improve performance only on some, not all problems.

We have added two special node types - *Root* and *Dummy* nodes - to our single tree representation: A tree contains a single instance of a *Root* node, unsurprisingly at the root of the tree where its role is to assemble the scalar outputs from child sub-trees into a (decision space) vector. A *Dummy* node is a special kind of terminal node which allows trees to change dimensionality. If a child of the Root node evolves to a Dummy node, this effectively reduces the dimensionality of the tree mapping by one. If the immediate child of the Root node is a Dummy node, the resulting element in the decision space vector is taken to be zero – which, of course, has no discriminatory power. Similarly, the dimensionality of a tree can increase if a sub-tree is added beneath the Root node which was previously a Dummy node. Since they are just a form of terminal node, Dummy nodes are free to occur anywhere in a tree, can be swapped and counted as normal terminal nodes but during evaluation, they return a constant zero value.

In actuality, there are two simultaneous search processes occurring during the optimization – one locates the optimal dimensionality of the decision space vector (the number and/or the positions of the non-dummy *subtrees*). The second, together with the optimization of the dimensionality, finds the optimal transformations to effect the mappings for each element of the decision space vector. The vectorizing tree structure employed here is illustrated in Figure 1.

Tree Complexity Measure: In mapping to a 1D decision space, Zhang & Rockett [7] employed the total number of tree nodes as a straightforward measure of tree complexity to provide a selection pressure which favored simpler solutions

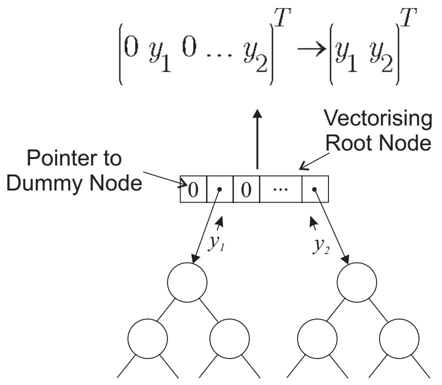
$\left(0\ y_1\ 0 \ldots y_2\right)^T \rightarrow \left(y_1\ y_2\right)^T$
Vectorising Root Node
Pointer to Dummy Node
0
0
...
y_1
y_2

Fig. 1. Illustration of the vectorizing tree chromosome structure

during evolution; this approach has been demonstrated to control tree-bloat in GP applications. With the multi-tree representation here, using total node count as a complexity measure will introduce an undesirable 'hidden' selection pressure which implicitly favors trees with lower dimensionality. Higher-dimensional solutions with more subtrees will tend to contain more nodes. Thus we have used the *mean* tree size as a complexity measure in this work:

$$Complexity = \frac{1}{m}\sum_{i=1}^{m} Size_i$$

where the $Size_i$ denotes the number of nodes in *subtree* i, and m is the number of non-Dummy sub-trees, that is, the dimensionality of the evolved decision space vector.

Misclassification Error: A Fisher linear discriminant (FLD) [16] has been trained during the fitness evaluation phase of each tree to perform classification in the m-dimensional decision space. The misclassification error (0/1 loss) from the Fisher discriminant over the training set is used as the second objective. The training of an FLD can be done quickly in closed-form, thus making a negligible contribution to the time of an iteration; we determine the decision threshold within the FLD classifier [16] by golden section search which is terminated when there is no further improvement in error rate.

Lower Bound on Misclassification Error: Following on from the ideas presented in the one-dimensional MOGP work [7][8], we have found that a further objective is needed to enhance selective pressure in the early stages of evolution when that from misclassification error alone is not sufficient. Partly because the misclassification error turns-out *initially* to be a rather insensitive gage of class separation, we have added the overlap area of the class-conditioned densities along the 1D Fisher projection direction [16] as our third and final objective. This overlap is a measure of the lower bound on the misclassification error, independent of the decision-making mechanism. We estimate this using a straightforward histogramming procedure. The overlap objective is analogous to, although not equivalent to Zhang & Rockett's use of an estimate of Bayes error in the 1D setting [7].

Overall, we have found experimentally that the combination of the three objectives is necessary for the algorithm to rapidly generate a Pareto set of parsimonious solutions which generalize well. Without the Complexity measure the trees bloat and the optimization tends to stagnate. Without the Lower Error Bound objective, convergence is very slow or non-existent. Each of the three objectives thus has a key role to play during the evolutionary process although since we are ultimately considering the classification domain, after we have generated a set of non-dominated solutions whose properties are 'shaped' by the multiple objectives, we select the solution which has the lowest (mean) validation error. This is a critical distinction between the current area and most other uses of multiobjective optimization: The multiple objectives are vital *constraints* during the evolutionary process but do not form part of a natural trade-off at the end of the optimization.

Table 1. MMOGP (PCGP) Settings

Terminal set	Input pattern vector elements
	Dummy nodes
	10 floating point numbers $\in \{0 \ldots 1\}$
Function set	sqrt, log, pow2, -, sin, not
	-, +, *, /, max, min, xor, or, and
	if-then-else
Max. dimensionality	50
Sub-tree preservation probability	0.2
Max. no. of tree evaluations	20,000
Stopping criterion	Max generations exceeded

2.3 Extended Breeding Operators

Crossover - In our single tree representation, the Root node is a special type of output node which only appears at the root of the GP tree. Hence the depth-fair crossover used for one-dimensional MOGP [7] is modified to avoid selecting the root node. Furthermore, we have used a second modification to depth-fair crossover designed to preserve useful genetic building blocks within the feature transformations associated with each dimension. With some probability, the *sub-tree preservation probability*, we perform crossover only within corresponding pairs of sub-trees. For example, given two parents, A and B, we perform cross over within sub-tree 1 of parent A and sub-tree 1 of parent B, then within sub-tree 2 of parent A and sub-tree 2 of parent B, and so on up to sub-tree p such that $p = \min[D_A, D_B]$ where $D_{A,B}$ is the dimensionality of the parents.

Mutation - Like [7], we use a depth-fair, size-dependent mutation except that the mutation operator is modified to allow mutation on the whole tree *including* the Root node. If the Root node is selected based on depth-fair conditions, a new random GP tree of random dimensionality will be created. After the population has been initialized, the mutation operation is responsible for introducing new dummy nodes in the GP tree.

Steady-State MOGP Implementation In a typical multiobjective optimization problem, a family of equivalent solutions exists, the members of which are superior to all the other feasible solutions. None of this family of *non-dominated* solutions can be considered 'better' than any other one from the point of view of the simultaneous optimization of multiple objectives; this family is termed the *Pareto-optimal set.* A traditional way to obtain multiobjective optima is to aggregate the multiple objectives into a single, weighted objective using linear or non-linear combination based on domain knowledge or experience, although this is far from satisfactory. Here we apply steady-state Pareto converging genetic programming (PCGP) to find the Pareto-optimal set for our multiobjective feature extraction problem since this method has been shown to yield smaller trees than competitor methods [8]. Further details of PCGP can be found in [8] and [9]. The population size for all problems in this paper was 200 and the maximum tree depth used to create the random initial population was 5. To create some diversity in the initial population, half of each initial population was created at full tree depth (5, here) and half with random tree depth (≤ 5) although thereafter, no limit was imposed on tree depth during the evolutionary optimization; bloat was controlled solely by the tree complexity objective discussed above. The full list of MMOGP settings used in this work are listed in Table 1.

3 Experiments

We re-emphasize: The method we present here is not a classifier, rather a feature extraction framework to design a classifier adapted to an individual problem without prior knowledge or any distributional assumptions. To justify our method, we investigated the design process on a representative range of benchmark datasets from the UCI Machine Learning database [10] and the Statlog project [11]. The key issue is that the generation of the feature extraction stage as well as the dimensionality of the projected decision space are driven by the notion of optimality.

Datasets We make statistical comparisons of classification performance between the results of our MMOGP algorithm and a range of established classifiers on 8 two-class learning problems. From the Statlog database [11] we use: (a) Australian credit approval, (b) German credit and (c) the heart disease datasets. From the UCI Machine Learning database [10] we use: (d) Glass (classifying between float and non-float glasses), (e) BUPA - liver disorder prediction, (f) Wisconsin Diagnostic Breast Cancer, (g) Pima Indians Diabetes and (h) Wisconsin

Breast Cancer datasets. Table 2 lists details of the eight datasets studied in this paper. The last column contains the decision space dimensionalities optimized by MMOGP for each dataset - these will be discussed below.

Table 2. The Eight Datasets Used in this Work

Name	Features	Size and Distributions	Projected Dimensionality
Glass	9	163 = 87 (float) + 76 (non-float)	19
BUPA	6	345 = 200 (Benign) + 145 (Malignant)	19
PID	7	532 = 355 + 177 (Diabetic)	7
WBC	10	699 = 458 (Benign) + 241 (Malignant)	6
WDBC	30	569 = 357 (Benign) + 212 (Malignant)	23
AUS	14	690 = 383 (Positive) + 307 (Negative)	20
GER	24	1,000 = 700 (Positive) + 300 (Negative)	34
HEA	13	270 = 120 (Diseased) + 150 (Benign)	13

Classification Algorithms As the basis for comparison with MMOGP, we have used eight existing classification algorithms with implementations from the Weka Machine Learning system [12]; we used the default parameter settings except where noted below. The classifiers used were: (i) Radial Basis Functions (RBF)– a normalized Gaussian radial basis function network using the k-means clustering algorithm. We estimated the number of clusters, k for a given dataset by considering a random split of the dataset, training the classifier on the first half and calculating a validation error on the second half. We adopted the value of k which gave the lowest validation error for each dataset by this method; (ii) Logistic - a modified multinomial logistic regression model with a ridge estimator; (iii) NNge – Nearest-neighbor-like algorithm using non-nested generalized exemplars; (iv) BayesNet – Bayes Network classifier using the K2 learning algorithm; (v) IB1 – Instance-based learning algorithm using a simple distance measure to find the training instance closest to the given test instance and predict the same class as this training instance; (vi) ADTree – the alternating decision tree learning algorithm; (vii) SMO – Sequential minimal optimization algorithm for training a support vector classifier; (viii) C4.5 – The well-known decision tree algorithm. (This is called J48 in Weka.)

Dietterich [13] has pointed-out that the commonly-used practice of classifier comparison by N-fold cross-validation followed by a t-test is unsound due to the implicit assumptions about independence being violated and has proposed an empirical 5×2 cv test. Alpaydin [14] has modified Dietterich's test to remove the unsatisfactory aspect of the result depending on the ordering of the folds: it is Alpaydin's F-test which we use here to statistically compare classifier performance. We perform five repetitions of splitting the dataset into two folds, treating one fold as the training set and the other as the test set to compute an

F-statistic with which to decide whether to reject the hypothesis that the performances of the two classifiers are identical. (See [14] for full details.) Throughout this work we have used a 95% confidence level to infer a statistical difference.

Table 3 shows the mean test errors from the 5×2 *cv* test for all nine classifiers (8 conventional + MMOGP) for all eight datasets. As explained above, we have used the MMOGP solution which displays the smallest mean error since in this classification problem we are (generally) only interested in the lowest classification rate − classifier structure is generally an unimportant detail. The outcomes of the F-test comparisons are also shown in Table 3 where a tick denotes that MMOGP is statistically superior to the conventional comparator classifier for that particular dataset and a dash denotes no statistical difference. It is noteworthy that MMOGP has the smallest error rates over all dataset/classifier combinations and the F-test results show that over the $8 \times 8 = 64$ paired comparisons, MMOGP is superior in 57 of these comparisons at the 95% confidence level. On the remaining 7 pairwise comparisons, MMOGP is statistically identical to the respective conventional classifiers. Most signficantly, MMOGP is not bettered by any of the comparator classifiers on any dataset.

The *optimal* dimensionalities extracted by multi-dimensional mapping for each dataset are list in Table 2. There is a reduction in dimensionality for WBC and WDBC while the dimensionalities of the projected spaces are identical to that of the original input space for HEA and PID. (The two spaces are not the same, of course.) Optimal performance on the other datasets is obtained with an *increase* in dimensionality over the original pattern spaces. MMOGP algorithm constructs new, typically non-linear features which maximize the class discriminability.

Table 3. Mean Error Comparisons of the 9 Classifiers on 8 Datasets (*5 × 2 cv* Test); F-test comparisons between algorithms for each dataset at 95 % confidence level. A tick represents superiority of MMOGP over the comparator classifer/dataset combination; a dash denotes no statistical difference

Datasets	Classifiers								
	RBF	LOG	NNge	Bayes Net	IB1	ADTree	SMO	C4.5	MMOGP
GLASS	0.354 ✓	0.364 ✓	0.322 ✓	0.311 ✓	0.300 ✓	0.317 ✓	0.392 ✓	0.338 ✓	0.135
BUPA	0.442 ✓	0.383 ✓	0.449 ✓	0.420 ✓	0.388 ✓	0.343 ✓	0.423 ✓	0.391 ✓	0.215
PID	0.255 ✓	0.233 ✓	0.249 ✓	0.249 ✓	0.301 ✓	0.248 ✓	0.222 ✓	0.263 ✓	0.203
WBC	0.048 ✓	0.045 ✓	0.038 –	0.026 –	0.042 ✓	0.043 ✓	0.030 –	0.057 ✓	0.024
WDBC	0.061 ✓	0.068 ✓	0.077 ✓	0.054 ✓	0.046 ✓	0.052 ✓	0.030 –	0.067 ✓	0.024
AUS	0.182 ✓	0.130 –	0.176 ✓	0.139 –	0.202 ✓	0.153 ✓	0.162 ✓	0.162 ✓	0.121
GER	0.288 ✓	0.270 ✓	0.250 ✓	0.274 ✓	0.324 ✓	0.272 ✓	0.266 ✓	0.304 ✓	0.230
HEA	0.178 –	0.185 ✓	0.233 ✓	0.184 ✓	0.244 ✓	0.235 ✓	0.179 ✓	0.242 ✓	0.140

Feature Selection and Extraction Due to space limitations, we show only one representative MMOGP tree in Figure 2 to illustrate the outcome of the multi-dimensional mapping process. This tree is the solution with the lowest error over the HEA dataset. The leaf nodes are labeled as Xn, $n \in \{1 \ldots N\}$

where N is the number of raw attributes in the input pattern vector. The "if-then-else" node returns the value of the second child if the first child value is greater than 0, otherwise the third child value is returned. The "max" node returns the larger value from its leaves while "min" return the smaller. If both leaf node values are larger than 0, "xor" returns 0, otherwise it returns 1.

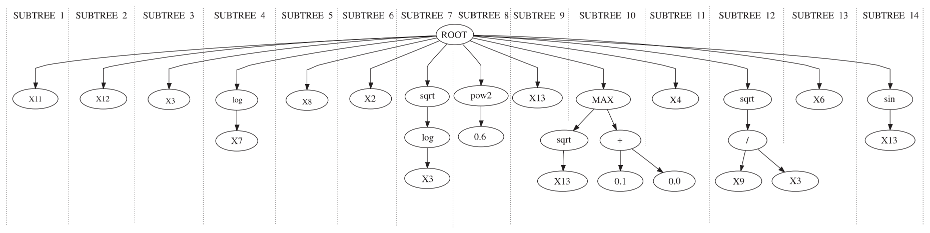

Fig. 2. Example GP tree to perform multi-dimensional mappings on the HEA datasets

The nominal dimensionality of this tree is 14, the number of the sub-trees. Actually, sub-tree 8 in Figure 2 returns a constant (0.6^2) to the final feature vector and hence will not provide any discrimination power. Clearly the objective which minimizes tree size does not exert quite enough selective pressure to remove minor redundancies from the trees – this is an area for future work. Thus, the dimensionality of the output feature vector is 13 and the mapping (listed in Table 2) is 13-to-13. Interestingly, the raw attributes: $X_1, X_5$and X_{10} have not been used at all while some of the new features are constructed from non-linear combinations of other raw attributes which are used repeatedly. During the multiobjective optimization process, feature selection has been *implicitly* conducted together with explicit feature extraction to design/search for the optimal multi-dimensional mappings.

4 Conclusions

In this paper we have proposed a generic, multi-dimensional, multiobjective genetic programming method to design an *optimal* feature extraction pre-processing stage for pattern recognition with the optimal dimensionality of the decision space being determined. The method is domain-independent and makes no assumptions about the class distributions or prior information. The single-tree structure is shown to be an effective way to represent the multi-dimensional mapping problem. We have carried-out an extensive range of comparisons with 8 conventional classifiers over a 8 benchmark datasets and conclude that MMOGP yields error rates which, in the vast majority of cases are statistically superior to the best of the conventional classifiers or, at worst, statistically identical.

References

1. J.R.Koza. (1994) Genetic Programming II, Automatic Discovery of Reusable Programs. *The MIT Press,* Cambridge, Massachusetts.
2. W.B. Langdon, (1998) Genetic Programming and Data Structures: Genetic programming + Data Structures = Automatic Programming. *Kluwer Academic Publishers*, London.
3. S.Raik & B.Durnota, (1994) *The Evolution Of Sporting Strategies*, in R Stonier and X Yu (eds), *Complex Systems '94: Mechanisms of Adaption, IOS Press.*
4. J.R.Sherrah, R.E.Bogner & A.Bouzerdoum. (1997) The Evolutionary Pre-Processor: Automatic Feature Extraction for Supervised Classification using Genetic Programming. *Genetic Programming 1997: Proc.of the 2nd Ann. Conf.* pp 304-312.
5. D.P.Muni, N.R.Pal, & J.Das, (2004) A Novel Approach to Design Classifiers using Genetic Programming, *IEEE Trans. on Evolutionary Computation*, vol. 8, no. 2, pp.183–196.
6. M.C.J.Bot, (2001) Feature Extraction for the k-Nearest Neighbor Classifier with Genetic Programming, In *Genetic Programming, Proc.of EuroGP 2001*, pp. 256–267.
7. Y.Zhang & P.I.Rockett, (2005) Evolving Optimal Feature Extraction using Multi-objective Genetic Programming: A Methodology and Preliminary Study on Edge Detection," In *GECCO 2005*, pp. 795-802.
8. Y.Zhang & P.I.Rockett, (2006) Feature Extraction using Multi-objective Genetic Programming, in *Y.Jin (ed), Multi-Objective Machine Learning*, pp.79-106. Springer.
9. R.Kumar & P.I.Rockett, (2002) Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimization by Steady-State Evolution: A Pareto Converging Genetic Algorithm, In *Evolutionary Computation*, vol.10, no. 3, pp. 283-314.
10. C.L.Blake & C.J.Merz, (1998) UCI Repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html].Irvine,CA: University of California, Department of Information and Computer Science.
11. D. Michie, D.J. Spiegelhalter & C.C.Taylor, (1994) Machine Learning, Neural and Statistical Classification, *Ellis Horwood*.
12. I.H.Witten & E.Frank, (2005) Data Mining: Practical Machine Learning Tools and Techniques (2^{nd} Edition). *Morgan Kaufmann*.
13. T.Dietterich, (1998) Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, *Neural Computation*, vol. 10, no. 7, pp. 1895-1923.
14. E.Alpaydin, (1999) Combined 5×2 cv F-test for Comparing Supervised Classification Learning Algorithms, *Neural Computation*, vol. 11, no. 8, pp. 1885-1892.
15. Y.Zhang & M.Zhang, (2004) A Multiple-Output Program Tree Structure in Genetic Programming, *Tech. Report CS-TR-04/14*, Victoria University, New Zealand.
16. R.O.Duda, P.E.Hart & D.G.Stork. (2000) Pattern Classification (2nd ed.), *Wiley Interscience*.
17. F.Camastra. (2003) Data Dimensionality Estimation Methods: A Survey. *Pattern Recognition* vol. 36, no. 12, pp.2945-2954.