Class-specific feature generation for 1NN through genetic programming

Mauricio García Limón, Hugo Jair Escalante, Eduardo Morales and Luis Villaseñor Pineda
Computer Science Department
National Institute for Astrophysics, Optics and Electronics (INAOE)
Sta. Ma. Tonantzintla, Puebla, México
Email: {mauricio.garcia, hugojair, emorales, villasen}@inaoep.com

***Abstract*—This paper introduces a genetic program for class-specific feature extraction for 1NN. Under the proposed method a new feature space is generated for each class in the problem under analysis. Where feature spaces are build by merging the initial features with a genetic program that aims at maximizing classification accuracy of a 1NN classifier. We compare the performance of our method to both, classical-standard techniques (e.g., PCA, LDA) and to solutions based on evolutionary algorithms. Experimental results reveal our method outperforms alternative solutions in a wide variety of data sets.**

I. Introduction

Pattern classification is concerned with the construction of predictive models that can map instances to a predefined set of classes or labels. In a typical pattern recognition system, feature extraction is a pre-processing stage, where the original feature space is transformed into a new space of features to facilitate the modeling problem for learning algorithms [16]. Sometimes finding a good representation requires the expertise of a human, however, this task can be replaced by an automatic feature generator without losing information [11].

The aim of automatic feature extraction methods is to find/learn the most effective features for discrimination among samples of different classes. In addition, feature extraction seeks to construct a much lower-dimensional feature space than the original input space, ignoring redundant and irrelevant information [6]. Therefore, a tradeoff between performance and reduction is desired. Because of its difficulty, it remains a popular problem within pattern recognition [5].

This paper introduces CSFG (Class-Specific Feature Generation), an approach to feature generation, based on genetic programming, for the Nearest Neighbor (NN) classifier. The main idea is to build feature spaces for each class separately, where each feature space discriminates very well instances of one class vs. the rest. New features are created by combining the features of the original input space, where the genetic program determines what features to combine and how to combine them. Although our work focuses on the NN classifier, one of the most popular predictive models so far [15], nothing constrains our method to be applied with alternative models (e.g., neural nets or support vector machines). The method is evaluated in low and high dimensional data sets, and compared against other evolutionary techniques from the state of art. The experimental results show that CSFG is robust and competitive in high dimensional data sets with respect to other methods, this is a very positive result because it is a promising solution to address large scale classification problems.

The rest of this paper is organized as follows. Section II reviews related work on feature extraction. Section III describes in detail the proposed CSFG approach. Section IV presents the experimental framework and the results obtained by the proposed method. Finally, Section V presents conclusions and discusses future work directions.

II. Related Work

Classical feature extraction methods include PCA and LDA [7], yet alternative methodologies are available as well. Mostly notably, solutions based on evolutionary algorithms have reported acceptable performance. For instance, M. C. Bot in [4] presents a method for feature generation through genetic programming for NN classification. The method uses a greedy strategy, which generates features one by one until the prediction accuracy in the training set is not better than a numeric constant; each new feature is generated with a standard genetic program [13]. The method constructs a unique (common for all classes) feature space where the training and testing instances are projected. In general, the method generates between 2 and 3 new features, which has a close-to-optimal reduction performance for most data sets (the optimum being 1 feature), also the authors provide a detailed analysis of the features obtained. Although being competitive, this method was evaluated on data sets with a small number of features, and is compared with methods such as PCA.

The problem of feature extraction has been also approached with multi-objective optimization. Zhang *et al.* in [17] proposed a standard genetic program for feature extraction based on the multi-objective algorithm SPEA-II (Strength-Pareto Evolutionary Algorithm) [3]. The proposed method uses evolutionary operators [12] that favor smaller trees. Three objectives are considered: the complexity of the trees, the classification error and a Bayesian error estimate. The first objective is to avoid the problem of *bloating* [8]; whereas the second objective ensures obtaining discriminative features. On the other hand, the third objective aims to reduce the overlap between classes under the generated input space, also it is used for fast convergence. The main feature of this method is that it always generates a single feature, that is, optimal reduction performance. The method is compared to different feature extraction methods, including that from M.C. Bot [4], and compared favorably. One problem with this approach is that it only addresses binary classification problems (a threshold over the 1D feature space is learned).

Zhang et al. in [16] present another a multi-objective

978-1-4673-7121-6/15/$31.00

genetic program. Unlike the method in [17], the new method faces multi-class classification problems as well. The authors use the vectorizing multi-tree representation and evolutionary operators proposed in [12]. The only objective that changes from the method in [17] is the Bayes error estimate, which is replaced in this paper by the Fisher Linear Discriminant (FLD). Again, this methodology is only evaluated on data sets with a small number of features, and is compared with other classifiers.

Recently, García *et al.* in [10] presented a genetic program for the simultaneous generation of prototypes and features with genetic programming. It uses a multi-tree representation, with ad-hoc operators and optimizes the training-set accuracy obtained by a 1NN classifier. This method constructs a feature space for each class in the classification problem addressed. Such method, however, is not compared to other feature generation methods, besides, it is not clear the importance that the feature generation process has by itself. The reduction rate reported in such work was of about 40%, while in our work we are close to 90% or feature space reduction.

This paper presents a method for the generation of class-specific features through genetic programming. Opposed to previous work, our method can be applied to high-dimensional data sets without any problem. Besides, it can deal with any number of classes and it is not tied to a specific classifier (although we implemented it with NN to compare our method with previous work [4], [10]). Most importantly, we show that our proposed method outperforms state of the art techniques. The next section details the proposed method.

III. Feature Generation through Genetic Programming

Genetic programming (GP) is an evolutionary algorithm which has been extensively used in data mining and pattern recognition problems [9], its main feature is the evolution of programs, where each program is usually associated to a modeling problem [13]. The proposed method is capable of obtaining a distinct set of features for each class where the number of features could be different for different categories. The aim is to improve the classification performance of the NN rule. In our proposal, each new feature is the result of combining distinct features from the original data set, by considering a tree structure (one tree per feature), where the leaves represent features and the internal nodes denote arithmetic operators. Genetic programming is used to evolve a population of trees-features trying to maximize the classification performance of the 1NN classifier. The remainder of this section describes the proposed method in detail.

A. Preliminaries

Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a training set of labeled instances, with $\mathbf{x}_i \in \Re^d$ and $y_i \in \mathcal{C} = \{1, \ldots, K\}$, where d is the dimensionality of the data and K is the number of classes in the considered problem. It is desired to generate a *set of sets* of features $\mathcal{F} = \{F_1, \ldots, F_K\}$ (one set of features F_i per class) where each set of features F_i generates a new input space (by the projection of instances) defined as follows: $\mathcal{I}_i = \{(\mathbf{w}_1^i, y_1^i), \ldots, (\mathbf{w}_n^i, y_n^i)\}$, for $i = 1, \ldots, K$ where $\mathbf{w}_j^i \in \Re^{d_i}$ with $d_i << d$, here, d_i is the dimensionality of the feature space for instances in the set $\mathcal{I}_i$ (the instances projected in learned features of class i). This means that all of the instances have the same dimensionality in the feature space d_i, which is not necessarily the same for different classes. Our goal is to learn $\mathcal{F}$ such that the performance of a 1NN classifier is maximized.

B. Representation

A genetic program is proposed to learn $\mathcal{F}$ where each individual codifies a solution to this problem, see Figure 1. Each individual contains the information of the distinct sets of features; formally, a individual is represented by a set of sets of tree-features: $\{F_1, \ldots, F_K\}$, where $F_i = \{f_1^i, \ldots, f_{d_k}^i\}$ is the set of trees that generates the new set of features of the class *i*. Each f_j^i is a tree associated with a single feature. Thus, the leaf nodes of the tree are the features of the training set, and the internal nodes are arithmetic operators.

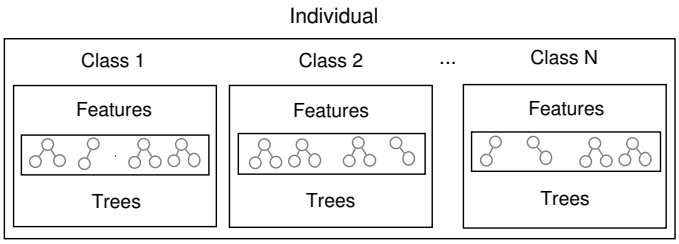

Fig. 1: Representation of an individual: class-specific feature trees.

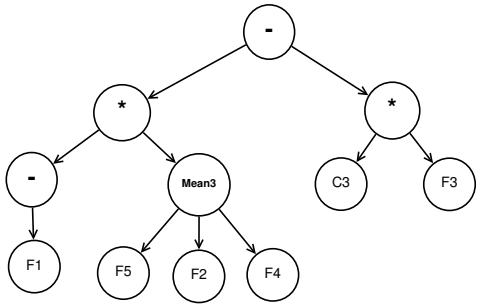

Fig. 2: A feature representation (tree-feature). $\mathcal{F}_y = (-F_1 * \frac{F_5+F_2+F_4}{3}) - (C_3 * F_3)$

The terminal set (from which leaf nodes can be taken) is the set of the original feature vectors of instances in $\mathcal{T}$ that belong to class i, and the following set of constants $\{0.1, 0.2, \ldots, 0.9, 1, 2, \ldots, 9\}$. The function set is shown in Table I; other operators were evaluated for inclusion in the function set (e.g. $sqrt$ and log_2), but better results were obtained with the operators in Table I

C. Genetic Operators

Evolutionary operators are essential in any evolutionary algorithm for generating new promising solution (trees in our case). We considered the following operators:

- **Crossover**: given two individuals, a feature tree from each parent is randomly selected, then a subtree of

TABLE I: Function set

Function	Type	Operation
pow_2	Unary	Returns the power of 2 of one input
$-$	Unary	Returns the negative value of one input
log_2	Unary	Returns the logarithm base 2 of one input
$-$	Binary	Substract 1st value by 2nd value
$+$	Binary	Adds 1st value by 2nd value
$\times$	Binary	Multiplies 1st value by 2nd value
$\div$	Binary	Protected division of the 1st value by 2nd value
max	Binary	Returns the maximum value of two inputs
min	Binary	Returns the minimum value of two inputs
$mean2$	Binary	Returns the mean value of two inputs
$mean3$	Ternary	Returns the mean value of three inputs
$if-then-else$	Ternary	Returns the 2nd value if 1st value = 0; otherwise return the 3rd value

each selected tree is chosen and exchanged. The outputs of this operator are two offspring. This process is repeated for each class associated to the classification problem. The operator is applied to all classes because affecting a class at a time would not have a significant effect in the overall population. The best crossover rate was of 100%, namely, always, for each class two trees were selected randomly, and two subtrees are exchanged. One possible reason for this parameter value is that the representation is highly complex.

- **Mutation**: given an individual, for each class, a randomly selected tree is eliminated and a new one is generated (using the same process as in the initialization). In the experimental evaluation, the best mutation rate was of 50%.
- **Multi-Crossover**: This crossover operator is used due to the representation complexity. Given two individuals, a new offspring is generated by combining trees from the parents. For each class, the offspring tree set is selected from the corresponding trees of the parents with uniform probability. This operator is similar to *uniform crossover* in a genetic algorithm [8]. Finally, the best multi-crossover rate was 100%, as in the crossover operator, this is due to the representation.

D. Fitness Function

The fitness function for the genetic program is the classification performance as obtained by a 1NN classifier over a validation set (a subset of $\mathcal{T}$). The goal is to maximize the classification rate of a 1NN classifier with the learned set of features. For classifying instances (either in training or test sets), they are projected in the feature spaces induced by the $K-$classes, next, an instance is associated with the label of the nearest instance (across classes).

E. Genetic Program

Algorithm 1 presents the pseudo code of the proposed method. The inputs are standard parameters and a validation data set $\mathcal{V} \subseteq \mathcal{T}$. The population is initialized with the ramped-half-and-half strategy, considering a maximum tree deep of D_max, this is to provide population diversity. Therefore, in each class, half of the trees are initialized with the full method, and the other half with the grow technique. For initialization, the proposed method takes as input PF parameters, which specify the number of initial feature-trees. This is just an initialization parameter, once the search is completed, the genetic program (implicitly) adjusts the final number of features for each class. The genetic program uses binary tournament for parent selection; half elitism is used to select the next population [13]: half of individuals are selected from the union of parents and offspring to remain in the new population, and the remaining individuals are selected from the offspring. The algorithm stops when a maximum number of generations is reached. The output of the method is the best individual, which consists of a set of feature-generation functions (i.e., a set of trees, each tree encoding a way to combine original features).

Algorithm 1 Class-Specific Feature Extraction via Genetic Programming.

Require: : $\mathcal{T}$: Training Set; $\mathcal{V}$: Validation Set; $Generations$: No. Generations; $Population$: No. Individuals; PF : Proportion of initial features
Ensure: $Best_x$: The best individual, the solution that obtains the highest accuracy in $\mathcal{V}$.
$P \Leftarrow$ *Ramped-half-and-half*$(PF, \mathcal{T})$ %% Initializaiton of solutions
$Best_x \Leftarrow \emptyset$; $FBest \Leftarrow -Inf$; $i = 0$ %% Intialization of variables
while $i <= Generations$ **do**
  for $j = 1 \rightarrow Population$ **do**
    $f(j) \leftarrow fitness(P_j, \mathcal{V})$
    if $f(j) > FBest$ **then**
      $FBest \leftarrow f(j)$; $Best_x \Leftarrow P_j$ %% Update best solution
    end if
  end for
  $Pool \Leftarrow$ *Sampling*(P)
  $Offspring \Leftarrow$ *MultiCrossover*$(Pool)$
  $Offspring \Leftarrow$ *Crossover*$(Offspring)$
  $Offspring \Leftarrow$ *Mutation*$(Offspring)$
  $P \Leftarrow$ *Selection*$(P, Offspring)$
  $i \leftarrow i + 1$
end while
return $Best_x$

IV. Experiments and results

The experimental evaluation of the proposed method was performed on two groups of data sets associated taken from the UCI Machine Learning Repository [1]. The first group comprises low-dimensional data sets ($<=$ 20 features), and the second one consists of data sets with a large number of features. Table II shows a description of the considered data sets. The evaluation methodology consisted on applying 10-fold cross validation to each data set. A Wilcoxon signed-ranks test [14] was performed to compare the results obtained by the proposed and reference methods, with a confidence level of 95%. The performance of CSFG is compared against four state of the art methods for feature generation, including Bot's [4], Zhang's [17] PCA and LDA [7] techniques.

The optimization of evolutionary-based methods (i.e., Bot's and Zhang's techniques) is driven by the classification accuracy, at the end of the search process, accuracy and *Cohen's Kappa Coefficient* [2] are reported to objectively assess the

TABLE III: Performance on low and high dimensional datasets

Dataset	Bot [4]	Y. Zhang *et al.* [17]	CSFG	1NN	PCA	LDA
Accuracy						
Small	0.7203 ± 0.1237 ⋆	0.7214 ± 0.0833	**0.7533** ± 0.1048	0.7009 ± 0.1061 ⋆	0.6502 ± 0.1032 ⋆	0.7277 ± 0.0921
Large	0.7475 ± 0.187 ⋆	0.6704 ± 0.1228	0.8018 ± 0.1847	**0.838** ± 0.1498	0.829 ± 0.1791	0.8379 ± 0.1542
Multi-class	0.8565 ± 0.1253	NI	0.8727 ± 0.1371	**0.9033** ± 0.1261	0.6809 ± 0.1461	0.7495 ± 0.1794
Kappa						
Small	0.3376 ± 0.2835	0.2794 ± 0.2096 ⋆	**0.3889** ± 0.2484	0.316 ± 0.2418	0.1952 ± 0.2261 ⋆	0.3755 ± 0.2160
Large	0.3344 ± 0.269	0.2003 ± 0.1696	0.44 ± 0.3285	**0.5768** ± 0.3599⋆	0.5182 ± 0.3656	0.5470 ± 0.3168 ⋆
Multi-class	0.7875 ± 0.1119	NI	0.7953 ± 0.1441	**0.8478** ± 0.1809	0.5716 ± 0.1828	0.6601 ± 0.1960

⋆ statistically significant difference.

TABLE II: Description of low and high dimensional data sets.

Small Datasets							
Dataset	Features	Instances	Classes	Dataset	Features	Instances	Classes
Appendicitis	7	106	2	Hepatitis	19	155	2
Australian	14	690	2	Housevotes	16	435	2
Bands	19	539	2	Movement-libras	90	360	15
Bre	9	286	2	Mammographic	5	961	2
Bupa	6	345	2	Monks	6	432	2
Crx	15	690	2	Pendigits	16	10992	10
Dermatology	34	366	6	Pima	8	768	2
Flare-solar	9	1066	2	Saheart	9	462	2
German	20	1000	2	Shuttle	9	58000	7
Haberman	3	306	2	Splice	60	3190	3
Heart	13	270	2	Texture	40	5500	11
Large Datasets							
Dataset	Features	Instances	Classes	Dataset	Features	Instances	Classes
Ads	1558	3279	2	Gisette	5000	7000	2
Basehock	4862	1993	2	Hiva	1617	4229	2
Gina	970	3468	2	Pcmac	3289	1943	2

performance of the 1NN classifier in the induced feature spaces. The following settings were defined to assess the proposed approach: CSFG was run for 50 generations, with a population size of 100, and maximum tree-depth of 3, and the value of PF is 1% for binary problems, and 40% for multiclass problems.; the above parameters were the best performance obtained in a previous experimental study. In the remainder of this section we evaluate the classification performance of the different methods, their reduction performance and show the decision surfaces generated with each technique.

A. Results

Table III shows the accuracy and Cohen's Kappa rate in low and high dimensional data sets. Regarding small data sets, the method obtains greater accuracy than both all reference methods and the 1NN classifier. Whereas in the case of large data sets, the proposed method outperforms evolutionary-based techniques, but it obtains lower performance than classical methods and 1NN. The previous result suggest our evolutionary program deals better than Bot's and Zhang's methods with the implicit huge search space associated to high-dimensional feature spaces. Although, it is clear that for high-dimensional binary classification tasks, classical methods perform better, again, this can be due to the size of the search space, for which CSFG may require of larger populations and more generations to obtain better solutions. Nevertheless, if we analyze the results for the multiclass problems in Table III, CSFG obtains the best performance among all feature extractors. In such problems, classical approaches fail to obtain a suitable feature space.

Results in terms of Cohen's Kappa are consistent with those obtained in terms of accuracy (remember, this measure express the portion of successes that can be attributed to the classifier itself, i.e., not to mere chance), in small data sets, so with regard to the other approaches (evolutionary methods, PCA, LDA and 1NN classifier) the results obtained by the proposed method is no mere chance in terms of accuracy, i.e. the method proposed is more accurate than the others. However, in large data, it is superior against the evolutionary methods and, it remains competitive compared LDA, PCA and 1NN classifier. Results reported in this section show CSFG is a very competitive method across varied types of data sets (low/high dimensional, binary/multiclass) in terms of classification performance. CSFG obtains similar performance as when using all of the features, but using a lower-dimensional space (see below). Also, CSFG outperforms other evolutionary-based methods and obtains comparable performance to classical techniques. As other evolutionary-based method it has problems with high-dimensional data sets. We can amend this by performing a more intensive search with the genetic program.

B. Reduction performance

The reduction rate allows us estimating the size of the new feature set, a high reduction (i.e., less features) allows the classifier to perform more efficiently. Remember, the proposed method in [17] always generates one feature, so that the reduction rate obtained by this method is the maximum possible. The proposed method in [4] mentions that this method reduces in average between 2 and 3 features, so that the reduction is close to optimal. For the proposed method, initially, the proportion of features for each class is fixed, and after the evolutionary process is able to set the number of features per class. Table IV shows reduction rates for the considered methods (recall 1NN has a reduction rate of 0). It can be seen that the reduction obtained by the proposed method is close to the optimum. In fact, there is no statistically significant difference among the methods. Therefore, we can say that our method offers a good tradeoff between accuracy (compares favorably with the most state of the art and with 1NN using the full input space) and reduction (no difference among the considered methods), i.e. the method obtains a high reduction rate without great difference between CSFG and others approaches on the classifier performance (accuracy). Furthermore, if it is considered as a multi-objective optimization problem, the ***CSFG*** would be on the Pareto front.

C. Analysis of the new feature space

In order to give insight into the type of feature spaces generated by each of the considered methods we show in Figure V the feature spaces generated by each method in the CRX data

TABLE IV: Reduction rate

Dataset	Bot [4]	Y. Zhang *et al.* [17]	CSFG	PCA	LDA
Small	0.8515 $\pm$ 0.084 $\star$	**0.8812** $\pm$ 0.0737	**0.8812** $\pm$ 0.0737	**0.8812** $\pm$ 0.0737	**0.8812** $\pm$ 0.0737
Large	0.9987 $\pm$ 0.0009	**0.9995** $\pm$ 0.0003	0.9944 $\pm$ 0.0025	0.9945 $\pm$ 0.0024	0.9945 $\pm$ 0.0024
Multi-class	**0.9070** $\pm$ 0.0515 $\star$	NI	0.7615 $\pm$ 0.0314	0.7292 $\pm$ 0.0601	0.7292 $\pm$ 0.0601

$\star$ statistically significant difference.

set (a binary problem for which all methods obtained similar performance). For CSFG we combined both feature spaces and generated a single decision surface. Interestingly, methods based on evolutionary algorithms (rows 1,2 and 5) generate decision surfaces and feature spaces that attempt to linearly separate the data. Whereas classical methods generate highly nonlinear decision surfaces (rows 3-4). Thus, the it seems the outputs of evolutionary-based methods could be very useful for training linear classifiers as well. Among evolutionary based method, it can be seen that Bot's and Zhang's methods make too many mistakes in the test set (in row 2 all instances are assigned the same label). Our method on the other hand (last column) correctly separates most instances. When compared to classical methods (rows 3,4), nonlinear decision surfaces seem to correctly classify most test instances. In this case 1NN could obtain acceptable performance, nevertheless it is clear from the plots that the separation among samples of different classes is better for CSFG. So other data-analysis tasks could benefit from feature spaces generated with CSFG.

V. Conclusions

This paper introduced an automatic method capable of generating class specific features for classification problems (not necessarily binary tasks) for the NN classifier. The proposed method was extensively evaluated in low and high dimensional data sets and its performance was compared to state of the art techniques. Our experimental study revealed that the proposed method, CSFG, compares favorably in classification performance and similarly in reduction performance when compared to state of the art alterative.

The main conclusions of this work can be summarized as follows:

- CSFG is able to reduce data dimensionality without compromising the classification performance.
- The proposed method obtains a feature reduction rate close to optimum.
- In small data sets the method outperforms the reference methods in terms of accuracy.
- In data sets with a large number of features CSFG obtained results equivalent performance in terms of accuracy to the classifier, making it an alternative to tackle large-scale problems
- The performance of the classification obtained by CSFG gives evidence that the method constructs more accurate and informative new features with regard to methods in state of the art.

Current and future work includes: addressing the problem of feature extraction with a multi-objective genetic program (minimizing the number of features and maximizing classification performance); also we would like to apply/adapt our method for specific tasks where high dimensionality is a serious issue (e.g. computer vision tasks such as image classification).

References

[1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[2] A. Ben-David. A lot of randomness is hiding in accuracy. *Engineering Applications of Artificial Intelligence*, 20(7):875 – 885, 2007.

[3] S. Bleuler, M. Brack, L. Thiele, and E. Zitzler. Multiobjective genetic programming: Reducing bloat using spea2. In *CEC*, pages 536–543, 2001.

[4] M. Bot. Feature extraction for the k-nn classifier with genetic programming. In *EuroGP'2001*, volume 2038 of *LNCS*, pages 256–267, 2001.

[5] S. Ding, W. Jia, W. Su, F. Jin, and Z. Shi. A survey on statistical pattern feature extraction. In *ICIC*, volume 5227, pages 701–708. Springer, 2008.

[6] S. Ding, H. Zhu, W. Jia, and C. Su. A survey on feature extraction for pattern recognition. *Artificial Intelligence Review*, pages 169–180, 2012.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2001.

[8] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. 2003.

[9] Pedro G. Espejo, Sebastián Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *Trans. Sys. Man Cyber Part C*, 40(2):121–144, March 2010.

[10] M. Garcia-Limon, H. J. Escalante, E. F. Morales, and A. Morales. Simultaneous generation of prototypes and features through genetic programming. In *GECCO '14*, pages 517–524, 2014.

[11] I. Guyon, M. Gunn, S.and Nikravesh, and L. A. Zadeh. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[12] T. Ito, H. Iba, and S. Sato. Non-destructive depth-dependent crossover for genetic programming. In *EuroGP '98*, volume 1391 of *LNCS*, pages 71–82, 1998.

[13] R. Poli, W. B. Langdon, and N. Freitag McPhee. *A field guide to genetic programming*. available at `http://www.gp-field-guide.org.uk`, 2008.

[14] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 12 1945.

[15] X. et al. Wu. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, December 2007.

[16] Y. Zhang and P. I. Rockett. A generic multi-dimensional feature extraction method using multiobjective genetic programming. *Evol. Comput.*, 17:89–115, March 2009.

[17] Y. Zhang and P. I. Rockett. A generic optimising feature extraction method using multiobjective genetic programming. *Applied Soft Comp.*, 11:1087 – 1097, 2011.

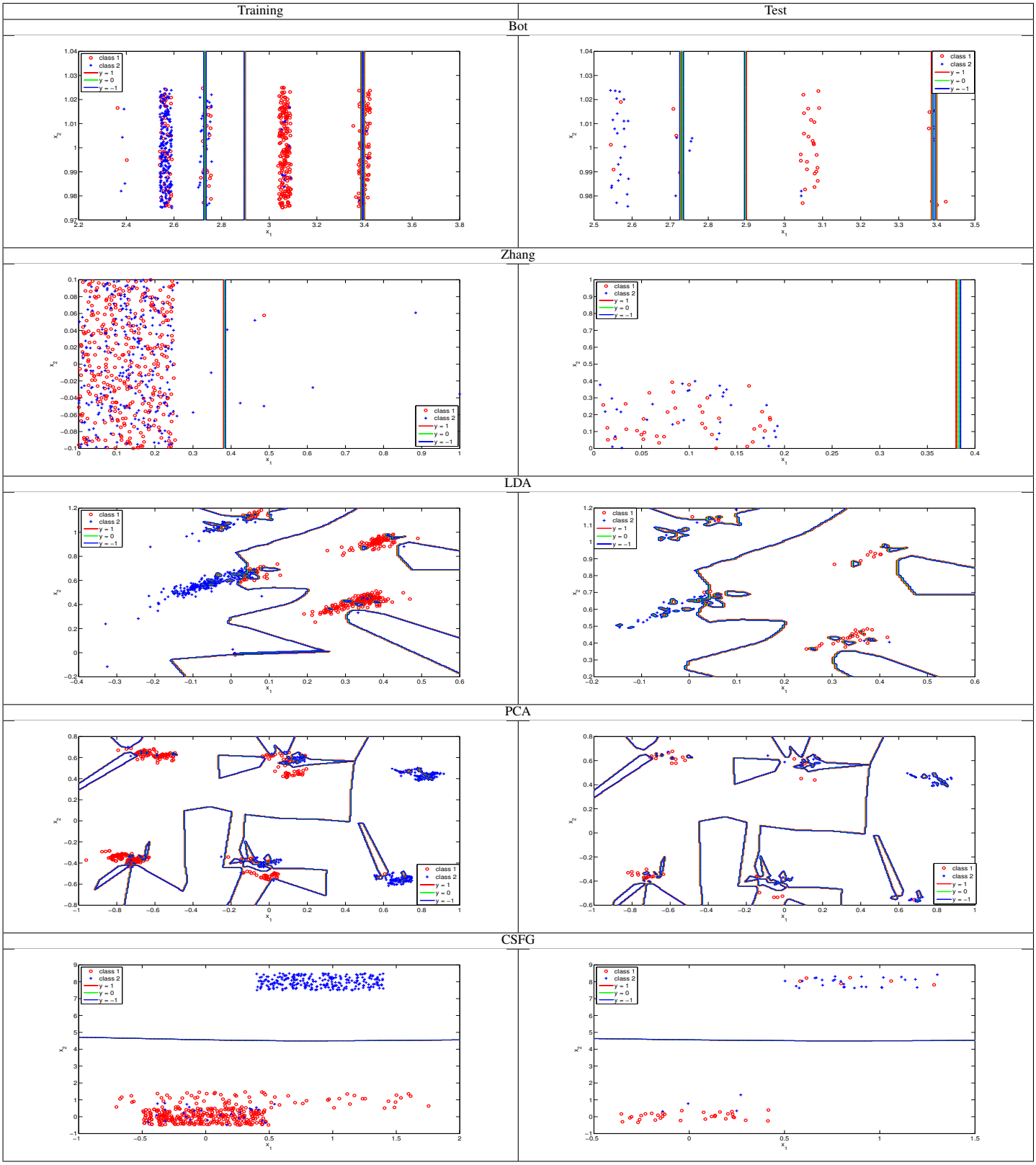

TABLE V: Decision surfaces with feat. spaces obtained by (top to bottom): Bot's, Zhang's, LDA, PCA and CSFG methods; training (left) and test (right) instances are shown.