Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 1384–1393
Expert Systems with Applications
www.elsevier.com/locate/eswa

Classifier design with feature selection and feature extraction using layered genetic programming

Jung-Yi Lin [a,*], Hao-Ren Ke [b], Been-Chian Chien [c], Wei-Pang Yang [a,d]

[a] Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, HsinChu 300, Taiwan
[b] Library and Institute of Information Management, National Chiao Tung University, Taiwan
[c] Department of Computer Science and Information Engineering, National University of Tainan, Taiwan
[d] Department of Information Management, National Dong Hwa University, Taiwan

Abstract

This paper proposes a novel method called FLGP to construct a classifier device of capability in feature selection and feature extraction. FLGP is developed with layered genetic programming that is a kind of the multiple-population genetic programming. Populations advance to an optimal discriminant function to divide data into two classes. Two methods of feature selection are proposed. New features extracted by certain layer are used to be the training set of next layer's populations. Experiments on several well-known datasets are made to demonstrate performance of FLGP.

Keywords: Feature generation; Feature selection; Pattern classification; Genetic programming; Multi-population genetic programming; Layered genetic programming

1. Introduction and review

1.1. Feature selection

This research concentrate on three research topics: feature selection, feature generation, and classifier design. Feature selection is an important technique of pattern recognition dealing with raw features. It focuses on removing useless, irrelevant, and redundant features. The classification accuracy of data derived by selected features is better than that by no selection. Many research working on feature selection have been proposed (Ahmad & Dey, 2005; Dash & Liu, 1997; Jain & Zongker, 1997; John, Kohavi, & Pfleger, 1994; Kittler et al., 1978; Kohavi & John, 1997; Kudo & Sklansky, 2000; Pernkopf, 2005; Pudil, Novovicova, & Kitter, 1994). John et al. (1994) divide feature selection methods into *filter* group and *wrapper* group. Filter methods applies by measuring the degree of feature relevance and deciding to leave or remove it. By contrast, wrapper methods (Jain & Zongker, 1997; Kohavi & John, 1997) function by cooperating with particular classifiers. Dash and Liu (1997) did a solid survey on many feature selection methods. They categorize feature selection methods into five based on the evaluation of features discrimination ability of: *distance measure*, *information measure*, *dependence measure*, *consistency measure*, and *classifier error rate*. Table 1 shows the five categories (Dash & Liu, 1997). Both Jain and Zongker (1997) proposed taxonomies of feature selection methods as shown in Fig. 1. In Jain and Zongker (1997), experiments with 15 different feature selection algorithms on extracted 18 features of SAR images are made. Kudo and Sklansky also compare excellently the differences among many feature selection methods in Kudo and Sklansky (2000). There are 16 different feature selection methods in comparison with 1-NN classifier, including sequential algorithms and branch-and-bound algorithms. Besides, they also give comments on these methods and

* Corresponding author. Tel.: +886 922 912 530.
E-mail addresses: jylin@cis.nctu.edu.tw (J.-Y. Lin), claven@lib.nctu.edu.tw (H.-R. Ke), bcchien@mail.nutn.edu.tw (B.-C. Chien), wpyang@cis.nctu.edu.tw (W.-P. Yang).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.01.006

Table 1
A comparison of evaluation functions (cited from Dash and Liu (1997))

Evaluation function	Generality	Time complexity	Accuracy
Distance measure	Yes	Low	–
Information measure	Yes	Low	–
Dependence measure	Yes	Low	–
Consistency measure	Yes	Moderate	–
Classifier error rate	No	High	Very high

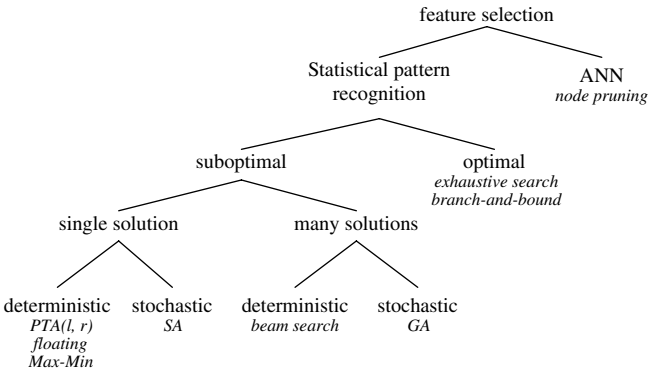

Fig. 1. Taxonomy of feature selection algorithms (cited from Jain and Zongker (1997)).

recommend some useful algorithms, which can provide the number of raw features. Kittler et al. (1978) and Pudil et al. (1994) proposed many feature selection techniques, such as (generalized) sequential forward selection, (generalized) sequential backward selection, (generalized) plus l-take away r selection, sequential forward floating selection, and sequential backward floating selection. Pernkopf compared the accuracy of Bayesian network classifier and k-NN classifiers with different feature selection methods in Pernkopf (2005).

Feature generation (Jolliffe, 1986; Lee & Landgrebe, 1997; Ma, Theiler, & Perkins, 2004; Mao & Jain, 1995; Park & Park, 2004; Wang & Paliwal, 2003) deals with features in different way. It generates representative features or classification-oriented ones (Ma et al., 2004). The former is mainly applied when the given data are not represented by a distinguishable form for the classifier on hand, e.g. handwriting and images. Mostly common feature extraction methods are principle component analysis (PCA) (Jolliffe, 1986) and linear discriminant analysis (LDA). Ma et al. (2004) proposed a general feature extraction framework and two nonlinear feature extraction algorithms, kernel function and mean-STD-norm, cooperating with a SVM classifier. Mao and Jain (1995) proposed several artificial neural network models such as PCA-networks, LDA-networks, and nonlinear discriminant analysis (NDA) network in Mao and Jain (1995). Wang and Paliwal (2003) investigates performance of PCA, LDA, minimum classification error (MCE), and generalized MCE with SVM classifier on vowel databases.

Traditionally, GP maintains a single population to find optimal solutions (Banzhaf, Nordin, Keller, & Framcone, 1998; Koza, 1992). Multipopulation GP (MGP) (Brameier & Banzhaf, 2001; Fernández, Tomassini, & Vanneschi, 2003) developed in recent years uses multiple populations to extend individual diversity and create different evolving environments. Fernández et al. (2003) performed several experiments about parallel and distributed GP (PADGP), isolated MGP (IMGP) ("isolated" means that there is no migration between populations), and traditional single population GP. Their experiments show that PADGP and IMGP usually perform better than traditional SGP. Moreover, MGP with small populations performs better than traditional GP using single large population. Brameier and Banzhaf (2001) combine linear GP and MGP techniques to design a classifier. Individuals represented as strings migrate between demes, i.e. subpopulations, according to their fitness. Fig. 2 (Brameier & Banzhaf, 2001) shows the circle topology where each circle stands for a population and arrows are migration directions.

Many classifiers based on GP have been proposed recently (Bojarczuk, Lopes, & Freitas, 1999; Chien, Lin, & Yang, 2003; Falco, Cioppa, & Tarantino, 2002; Freitas, 1997; Kishore, Patnaik, Mani, & Agrawal, 2000; Konstam et al., 1998; Lin, Chien, & Hong, 2002; Loveard & Ciesielski, 2001). Kishore et al. (2000) considered a k-class classification problem as a combination of k two-class classification problems and then generated corresponding expressions or discriminant functions for each class, and so did we (Chien et al., 2003; Lin et al., 2002). These methods need k runs for a k-class classification problem. To generate classification rules, Freitas (1997) proposed Tuple-Set-Descriptor (TSD), a logical formula to represent an individual. Muni, Pal, and Das (2004) proposed a method to solve multi-class problem by representing each individual as a multitree. Each tree stands for a candidate solution corresponding to each class. To evolve an individual is equivalent to evolve k trees simultaneously.

Researches on feature selection and feature extraction using evolutionary computation boom rapidly. The use of genetic algorithm (GA) methods can be found in Oh,

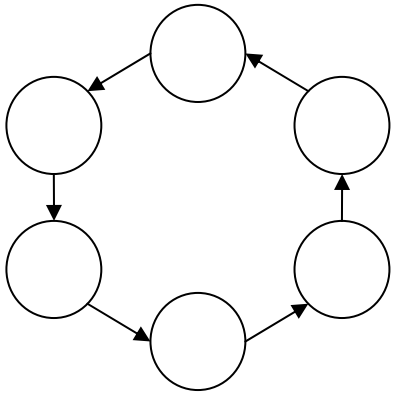

Fig. 2. The circle topology MGP.

Lee, and Moon (2004), Raymer, Punch, Goodman, Kuhn, and Jain (2000) and Siedlecki and Sklansky (1989). Also, some papers focus on methods using genetic programming (GP) (Hong, Jack, & Nandi, 2005; Kotani, Ozawa, Nakai, & Akazawa, 1999; Muni, Pal, & Das, 2006; Muni et al., 2004; Otero, Silva, Freitas, & Nievola, 2003; Rizki, Zmuda, & Tamburino, 2002; Sherrah, Bogner, & Bouzerdoum, 1996; Smith & Bull, 2004). Sherrah et al. (1996) used GP to perform feature selection before applying particular classifiers. The uncontained features in the result are removed. Kotani et al. (1999) and Hong et al. (2005) used GP to generate new features and employed other classifiers with the generated features to perform classification tasks. SVM and ANN are used to be classifiers with generated features on bearing faults problem. Rizki et al. (2002) proposed a hybrid evolutionary learning algorithm, HELPR, performing feature extraction from raw input data. Smith and Bull (2004) combined GP and GA to do feature generation and feature selection with C4.5 classifier, and so did Otero et al. (2003). Muni et al. (2006, 2004) proposed an approach to construct multi-class classifier with online feature selection named GP_{mtfs}, multitree genetic programming based feature selection. They proposed two crossover algorithms with the multitree GP technique to find minimum number of useful features.

This paper proposes a method named FLGP to design a classifier combined with feature selection and feature extraction. The details of FLGP are described in Section 2. In Section 3, experiments results on several datasets are presented and analyzed. Conclusions are finally drawn in Section 4.

2. FLGP

This section aims to itemize FLGP. At first, basic GP terms, including terminal, operation, individual, population, and genetic operators are going to be introduced. Secondly, layers and the relations between layers are described.

We define the classification problem as follows.

Let T be the training set for a K-class classification problem including n training samples and TS be the test set. A training sample of T is a pair of class label and m significant real-valued elements

$$T = \{t_i | t_i = (c_i, x_i), c_i \in \{c_1, \ldots, c_K\},$$
$$x_i = (a_{i1}, a_{i2}, \ldots, a_{im}), m > 0, 1 \leqslant i \leqslant n, a_{ij} \in \mathbb{R}\}.$$

It aims to find a discriminant function F to be the classifier that provides best classification accuracy on TS, where

$$F : \mathbb{R}^m \rightarrow \mathbb{R}.$$

2.1. Single population

A population P is a set of individuals. The size of P is predefined as N. Let I be an individual, then

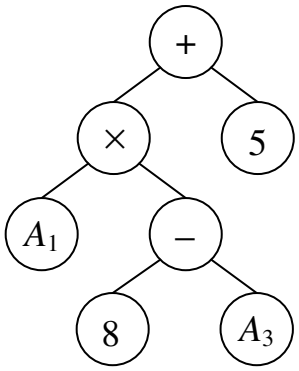

Fig. 3. Binary tree representation of individual $(A_1 \times (8 - A_3)) + 5$.

$$P = \{I_1, I_2, \ldots, I_N\}.$$

An individual I in this work is represented by a binary tree, for example, an individual $(A_1 \times (8 - A_3)) + 5$ is shown in Fig. 3. A tree has three different types of nodes: *terminals*, *constants*, and *operations*. Let S_t, S_c and S_{op} be the terminal, the constant, and the operation set respectively. Terminals of S_t are variables related to features. S_c is a set of predefined constants. S_{op} is the set of operators corresponding to S_t and S_c. S_{op} containing only $\{+, -, \times, \div\}$ is sufficient because of the research results of Kishore et al. (2000). In this way, S_t, S_c, and S_{op} are defined as

$$S_t = \{A_i | 1 \leqslant i \leqslant m, A_i \text{ is the name of } i\text{th feature}\},$$
$$S_c = \{\text{const}_i | \text{const}_i \in N, \text{const}_i = i\},$$
$$S_{op} = \{+, -, \times, \div\}.$$

The size of an individual is decided by its height. An individual containing more nodes makes it possible to have better performance, but also requires more time to calculate. The height of individuals is empirically predefined and is denoted as IH. The binary tree of an individual contains at most $2^{IH} - 1$ nodes.

The *target class* is what the individuals are trained to fit, for example, TC is *BENIGN* if we are finding an optimal classifier for recognizing *BENIGN* objects. The *positive instances* of T are samples whose class label is the *target class*. Otherwise, they are *negative instances*. An individual I *recognizes* a given training sample t only when $I_f(t) \geqslant 0$, and *repels it* when $I_f(t) < 0$. The fitness function works to evaluate *how well* an individual I is. Let the fitness value of individual I_i be f_i. Classification results of an individual with T can be represented by four values: true positive (TP), true negative (TN), false positive (FP), and false negative (FN), as shown in Table 2. The fitness function used in this work is made of *sensitivity* and *specificity* (Han & Kamber, 2001)

$$\text{Sensitivity} = \frac{\text{TP}}{(\text{TP} + \text{FN})} = \frac{\text{number of positive instances correctly classified by } I_i}{\text{number of positive instances in } T},$$

Table 2
TP, FP, FN, and TN of an individual I_j

	Positive instance	Negative instance
I_j is positive	TP	FP
I_j is negative	FN	TN

$$\text{Specificity} = \frac{\text{TN}}{(\text{FP}+\text{TN})} = \frac{\text{number of negative instances correctly classified by } I_i}{\text{number of negative instances in } T},$$

$$f_i = \text{Sensitivity} \times \text{Specificity}.$$

2.2. The evolution process

GP evolves a population for a number of generations and takes the best individual F as its result. The term *evolve* means a systematic processes which can mimic the natural selection mechanism by performing genetic operators on the population. Three primary genetic operators, *crossover*, *mutation*, and *reproduction*, are performed with predefined probabilities W_c, W_m, and W_r, respectively. The crossover operator produces two new individuals from two existing individuals called parents. The mutation operator replaces a subtree of a randomly selected individual by a randomly generated subtree to obtain a mutant. This mutant may contain new structures never occurred before. The reproduction operator that mimics the natural principle, *the fittest survives*, keeps a selected individual alive to next generation.

In this work, we use a modified version of the basic GP evolution scheme (Banzhaf et al., 1998; Koza, 1992). After evaluating the fitness values of individuals, only the best and second best individuals are going to be reproduced. Crossover and mutation are performed on remaining individuals. Offspring are compared with their parents, and the better ones survive to next generation.

Only the mutation operator can generate individuals with new information. However, in order to avoid random walk, W_m is usually a small value. Under this circumstance, the individuals may not have enough chance to perform mutation. And hence it may diminish the diversity of the population. In particular, individuals may be stuck in certain form when they are already having good fitness value, for example, when $I_i = A_1$ or $I_j = A_5$, the crossover operator just swaps them entirely and will not generate any different offspring. Such situation is also a local optimum problem. Therefore, we propose a method that raises W_m with respect to generation increasing.

Let υ be the current generation and G be the maximum generation. Then W_m at generation υ is set as

$$W_m \text{ at generation } \upsilon = \begin{cases} W_m & \text{if } \frac{f_{MAX}}{f_{AVERAGE}} > 2.0, \\ \left\lceil W_m \times \left(\frac{W_c}{W_m}\right)^{\upsilon/G} \right\rceil & \text{otherwise,} \end{cases}$$

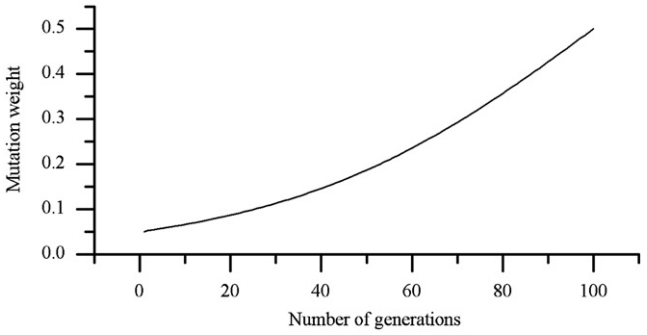

Fig. 4. The curve of W_m with $G = 100$.

where f_{MAX} and $f_{AVERAGE}$ is the best and average fitness value of the population in generation υ, respectively. W_m does not need to change when the best fitness value is double higher than average. Once W_m changes, $W_c = 1 - W_m$. At the last generation, W_m could be equal to W_c. In other words, the probability for W_m to be performed in later generations is estimated 50%. Here an example is given in Fig. 4, which shows the smooth curve of W_m under initial conditions that $G = 100$, $W_m = 0.05$ and $W_c = 0.5$, and the assumption that f_{MAX} is always smaller than $2 \times f_{AVERAGE}$.

A population P stops its evolution process under two conditions. First, P executes G generations. Second, the fitness value of an individual equals to 1.0. Once P stops, the fittest individual is denoted as F.

2.3. Feature selection methods

Here we discuss the feature selection method. This paper proposes two feature selection methods: M1 and M2. M1 is to reduce fitness value of an individual with too many features. M2, on the other hand, controls feature occurrences by tuning their weights during the evolution process.

2.3.1. Method 1 (M1)

This method gives the individuals with more features a lower fitness value to reduce their probability of being selected (Muni et al., 2006). Instead of adopting a new fitness function, we assign new fitness values for individuals once a generation completes. Using a fitness function concerning only the number of features used in an individual may cause an inaccurate individual having higher fitness. Since the accuracy is the most concern, M1 takes into consideration a set of individuals rather than single individual.

At first, we partition the population into several subsets by collecting individuals of similar fitness. For a population $P = \{I_1, I_2, \ldots, I_N\}$, where $f_i \geqslant f_{i+1}$. P is divided into subsets $\{S_1, S_2, \ldots, S_k\}$ by

$$S_1 = \{I_1, \ldots, I_i\}, \quad S_2 = \{I_{i+1}, \ldots, I_{i+j}\}, \ldots, \quad S_k = \{I_k, \ldots, I_N\}.$$

Let I_α be the best one in S_i, the fitness of individuals in $S_i \in [f_\alpha, f_\alpha - \theta]$, where θ is a predefined constant. In this paper, we define $\theta = 0.01$.

Afterward, individuals in a subset S_i are sorted by the number of features they used. For an ordered set $S_i = \{I_{i1}, I_{i2}, \ldots, I_{ij}, \ldots, I_{ik}\}$ the fitness of an individual I_{ij} is assigned to be

$$f'_{ij} = \left(\frac{(f_{\max} - f_{\min})(k - j + 1)}{k}\right) + f_{\min},$$

where $f_{\min}$ and $f_{\max}$ are the minimum fitness value and the maximum fitness value of individuals in set S_i, respectively.

2.3.2. Method 2 (M2)

Diversity of the population would be influenced by irrelevant features, especially when the given problem is a high-dimensional one. The explanation goes to the limitation in both the number of individuals of a population and the number of available nodes of each individual. Instead of tuning the fitness value of each individual, M2 tunes the weights of all features generation by generation. Features with lower weights are unlikely to be chosen when an individual needs a new feature.

For a population P_i, once one generation completes we calculate the terminal usage of individuals. P_i is divided equally into two sets HF and LF. HF is the set of better individuals. We further define the terminal usage φ^{H} and φ^{L} by

$$\varphi^{\mathrm{H}} = \langle \varphi_1^{\mathrm{H}}, \varphi_2^{\mathrm{H}}, \ldots, \varphi_k^{\mathrm{H}} \rangle,$$
$$\varphi^{\mathrm{L}} = \langle \varphi_1^{\mathrm{L}}, \varphi_2^{\mathrm{L}}, \ldots, \varphi_k^{\mathrm{L}} \rangle,$$
$$k = \sum_{q=0}^{i-1} l_q,$$

where φ_j^{H} and φ_j^{L} stand for the occurrence frequency of terminal A_j in HF and LF, respectively.

Suppose that current layer is L_i, the weight of terminals is set by

$$\text{weight of } A_j = \left\lceil \frac{\varphi_j^{\mathrm{H}} + 1}{\varphi_j^{\mathrm{L}} + 1} \right\rceil \Big/ \left\lceil \frac{\sum \varphi^{\mathrm{H}} + 1}{\sum \varphi^{\mathrm{L}} + 1} \right\rceil.$$

Terminals always have positive weights. We do not remove any of them during the evolution process.

2.4. Layers

So far, we have proposed the methods on evolving populations to obtain accurate functions with feature selection. Such functions are combinations of features, which are not only capable of classifying objects but also generating new features. The new features are results of feature extraction. Instead of using new features only, this paper makes populations evolve with all original features and new features. Some original features evolving with new features have chances to be selected in successive layers so that the nesting problem can be avoided.

A layer of FLGP is a set of populations. Populations in a layer do not have any communication. The evolution process of each population could be run in parallel or in sequence. Let the number of populations in ith layer L_i be l_i, we have

$$L_i = \{P_{i1}, P_{i2}, \ldots, P_{il_i}\}.$$

Let $l_0 = m$ and T_0 be the original given training set. Populations in L_1 execute their learning process through T_0. In general, layer L_i needs a particular training set T_{i-1}. Functions $(F_{i1}, F_{i2}, \ldots, F_{il_i})$ are generated after evolution processes of all populations of L_i complete. Such functions construct new features to fill up a new training set T_i

$$T_0 = \{t_{0j} | t_{0j} = (c_j, x_{0j}), x_{0j} = (a_{0j1}, a_{0j2}, \ldots, a_{0jm})\},$$
$$T_1 = \{t_{1j} | t_{1j} = (c_j, x_{1j}), \quad x_{1j} = (a_{1j1}, a_{1j2}, \ldots, a_{1jm}, a_{1j(m+1)}, \ldots, a_{1jl_1})\},$$
$$\vdots$$
$$T_i = \left\{ t_{ij} | t_{ij} = (c_j, x_{ij}), \quad x_{ij} = \left(\underbrace{a_{ij1}, \ldots, a_{ijm}}_{\text{original features}}, \underbrace{a_{ij(m+1)}, \ldots, a_{ij(k+l_i)}}_{\text{new features generated by } L_0, L_1, \ldots, L_i} \right), k = \sum_{q=0}^{i-1} l_q \right\},$$

$$c_j \in \{c_1, \ldots, c_K\}, \ m > 0, \ 1 \leqslant j \leqslant n, \ a_{ijk} \in R.$$

The value of a new feature a_{ijk} is obtained by

$$a_{ijk} = \begin{cases} a_{(i-1)jk} & \text{if } k \leqslant \sum_{q=0}^{i-1} l_q, \\ F_{k - \sum_{q=0}^{i-1} l_q}(t_{(i-1)j}) & \text{otherwise.} \end{cases}$$

The terminal set S_{t} used in layer L_{i+1} is defined as S_{t}^{i+1}. S_{t}^{i+1} contains not only terminals of the S_{t}^{i} but also new feature names with respect to new features

$$S_{\mathrm{t}}^{i+1} = \bigcup_{j=1}^{i} S_{\mathrm{t}}^{j} = \left\{ A_1, A_2, .., A_{\left(\sum_{q=0}^{i} l(q)\right)} \right\}.$$

Layer L_{i+1} begins its evolution process with T_i after T_i is constructed and S_{t}^{i+1} is prepared. We define FLGP as

$$\mathrm{FLGP} = \{(L_i, T_{i-1}, T_i) | 0 < i \leqslant \Gamma\},$$

where Γ is the number of layers. In this paper, we define $l(\Gamma) = 1$ to obtain single function as a result. After FLGP completes, we will have $\sum_{q=0}^{\Gamma-1} l_q + 1$ features. The last feature $A_{\sum_{q=0}^{\Gamma-1} l_q + 1}$ is denoted as A_{F} to indicate the training results. The algorithm of FLGP evolution process is shown in Algorithm 1. Fig. 5 illustrates the architecture and relationship between layers.

Algorithm 1 (*FLGP evolution process*).

(1) Let $T_0 \leftarrow T$, $P\_COUNT \leftarrow 1$, $L\_COUNT \leftarrow 1$.
(2) Perform evolution process on population P_{P_COUNT} in layer L_{L_COUNT}, with training set $T_{L\_COUNT-1}$.
(3) $P\_COUNT \leftarrow P\_COUNT + 1$.
(4) Repeat (2) unless $P\_COUNT > l_{L\_COUNT}$.
(5) Evaluate $F_{L\_COUNT1}, F_{L\_COUNT2}, \ldots, F_{L\_COUNTl_{L\_COUNT}}$ with $T_{L\_COUNT-1}$ to generate new feature values.
(6) Create training set T_{L_COUNT} with values evaluated at step (5).

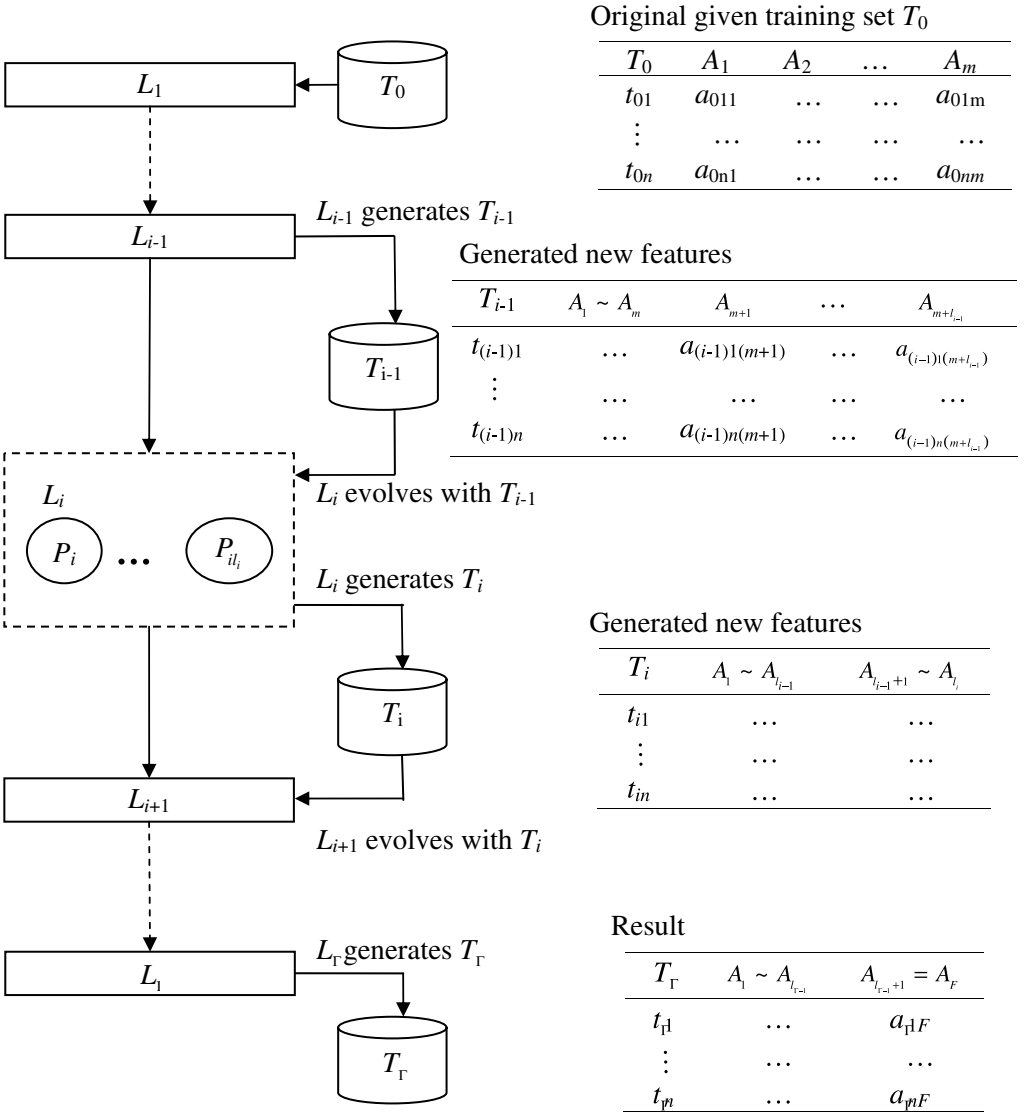

Fig. 5. Illustration of FLGP.

(7) If $L\_COUNT < \Gamma$, then $L\_COUNT \leftarrow L\_COUNT + 1$ and $P\_COUNT \leftarrow 1$. Jump to (2).

Using discriminant function is only suitable for two-class classification problem. A K-class classification problem can be treated as K two-class classification problems and execute K FLGPs regarding to each class. However, a problem called conflict may occur. To solve such problem, several creative and efficient methods have been proposed in Chien et al. (2003), Kishore et al. (2000), Lin et al. (2002) and Muni et al. (2006). This paper concerns two-class classification problem only and performs FLGP for the majority class.

2.5. A brief example of FLGP

This section demonstrates FLGP with the first distribution of *cancer* problem (Prechelt, 1994). FLGP is trained with full training set. Table 3 shows nine training samples of the training set T_0 where $\boldsymbol{M}$ stands for malignant and $\boldsymbol{B}$ stands for benign. The settings of this example are

$$\text{targetclass} = \boldsymbol{M}, \quad \Gamma = 2, \quad \text{IH} = 8,$$
$$FLGP = ((L_1, T_0, T_1), (L_2, T_1, T_2)),$$
$$L_1 = (P_{11}, P_{12}), \quad L_2 = (P_{21}),$$
$$S_t^1 = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\},$$
$$S_t^2 = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}\}.$$

Assume that the best individuals output by P_{11}, P_{12}, and P_{13} are

$$F_{11} = A_9 - ((((A_1 \times (70 \times (A_4 \times A_1))) \times ((A_1 + A_6) - (A_6 - A_2))) \times (A_1/A_2)) \times ((A_1 + (A_6 + (A_2 + (A_4 \times A_3)))) - 1)),$$
$$F_{12} = (A_1 \times (((9 + (40/A_2)) - (8/((A_6/A_4) - (A_2/8)))) - ((((29 - A_1) + A_2) \times (A_1 \times A_9)) \times (((23 \times A_6) \times (15 - A_6)) + A_4)))) - A_6,$$

Table 3
T_0: a 2-class problem and nine training samples

T_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	Class
t_{01}	0.20	0.10	0.10	0.10	0.20	0.10	0.20	0.10	0.10	***M***
t_{02}	0.20	0.10	0.10	0.10	0.20	0.10	0.30	0.10	0.10	***M***
t_{03}	0.50	0.10	0.10	0.10	0.20	0.10	0.20	0.10	0.10	***M***
t_{04}	0.50	0.40	0.60	0.80	0.40	0.10	0.80	1.00	0.10	***B***
t_{05}	0.50	0.30	0.30	0.10	0.20	0.10	0.20	0.10	0.10	***M***
t_{06}	0.20	0.30	0.10	0.10	0.30	0.10	0.10	0.10	0.10	***M***
t_{07}	0.30	0.50	0.70	0.80	0.80	0.90	0.70	1.00	0.70	***B***
t_{08}	1.00	0.50	0.60	1.00	0.60	1.00	0.70	0.70	1.00	***B***
t_{09}	1.00	0.90	0.80	0.70	0.60	0.40	0.70	1.00	0.30	***B***

The original nine features and the two new features, A_{10} and A_{11}, construct T_1 which is shown in Table 4. Here the set of used features of F_{11} and F_{12} are $\{A_1, A_2, A_3, A_4, A_6\}$ and $\{A_1 A_2, A_4, A_6, A_9\}$, respectively.

Next, L_2 uses T_1 to evolve its populations and generates F_{21}. Suppose it is

$$F_{21} = (41/(((A_{10} + ((80 \times A_{10}) + A_4))/A_8)/A_4)) + (9 \times (A_1/A_{11})).$$

Then we combines F_{21} and features $A_1, \ldots, A_{10}$ of T_1 to build T_2 which is shown in Table 5, where A_{11} is the result of the training samples. The set of used features is $\{A_1, A_4, A_8, A_{10}, A_{11}\}$. Since A_{10} and A_{11} are made from the two sets respecting to F_{11} and F_{12} as mentioned above, the set of used features is $\{A_1, A_2, A_3, A_4, A_6, A_8, A_9\}$.

Assume that we have two test data y_1 and y_2. Suppose that FLGPs return y_1 and y_2 regarding to class ***M*** are 0.3 and −0.4, respectively. We can classify y_1 to class ***M*** and y_2 to class ***B***.

Table 4
Training set T_1 that is generated by L_1

T_1	A_1	⋯	A_9	A_{10}	A_{11}	Class
t_{11}	0.20	⋯	0.10	0.1991	76.1066	***M***
t_{12}	0.20	⋯	0.10	0.1991	76.1066	***M***
t_{13}	0.50	⋯	0.10	1.6225	175.7748	***M***
t_{14}	0.50	⋯	0.10	−7.4600	−24.2714	***B***
t_{15}	0.50	⋯	0.10	0.2633	42.1644	***M***
t_{16}	0.20	⋯	0.10	0.1364	22.7037	***M***
t_{17}	0.30	⋯	0.70	−2.3482	−514.8546	***B***
t_{18}	1.00	⋯	1.00	−440.0000	−9126.0333	***B***
t_{19}	1.00	⋯	0.30	−192.1067	−1135.0109	***B***

Table 5
Training set T_2 that is generated by L_2

T_2	A_1	⋯	A_9	A_{10}	A_{11}	Class
t_{21}	0.20	⋯	0.10	0.1991	0.0489	***M***
t_{22}	0.20	⋯	0.10	0.1991	0.0489	***M***
t_{23}	0.50	⋯	0.10	1.6225	0.0287	***M***
t_{24}	0.50	⋯	0.10	−7.4600	−0.2398	***B***
t_{25}	0.50	⋯	0.10	0.2633	0.1259	***M***
t_{26}	0.20	⋯	0.10	0.1364	0.1161	***M***
t_{27}	0.30	⋯	0.70	−2.3482	−0.1784	***B***
t_{28}	1.00	⋯	1.00	−440.0000	−0.0018	***B***
t_{29}	1.00	⋯	0.30	−192.1067	−0.0098	***B***

3. Experiments

This section will discuss the experiments and analyzes classification results. We select three diagnostic problems, *cancer*, *diabetes*, and *heart*, from the PROBEN1 benchmark set (Prechelt, 1994). These problems are originally from the UCI repository (Blake, Keogh, & Merz, 1998) and have been preprocessed by Prechelt (1994). Values of all sets are normalized to the continuous range [0, 1]. Missing attributes are completed. Every attribute of m possible values is encoded by the 1-of-m method. PROBEN1 divides every problem dataset into three subsets: training, validation, and test. Furthermore, for each dataset PROBEN1 is prepared by three different compositions with different orders of samples. The numbers of training set and test set in different composition might be slightly different. This should raise the confidence degree on the extent of classification results not influenced by the sample distribution of the training set and test set. Summarization of these problems is shown in Table 6.

3.1. Experiments results

Since the number of layers and the population size are made empirically, we design 10 different configurations. Experiments settings are shown in Table 7 and the common parameter values for all settings are shown in Table 8. The selected experiment datasets have already been separated

Table 6
Summarization of selected problems

Problem	Classes	Number of features	Training samples	Majority class	Test data
Cancer	2	9	350	C_1	174
Diabetes	2	8	384	C_2	192
Heart	2	35	152	C_1	75

Table 7
Nine experimental settings

Settings	Γ	l_i	Population size for each
ES1	1	1	5000
ES2		1, 1	2500
ES3		2, 1	1650
ES4	2	3, 1	1250
ES5		4, 1	1000
ES6		1, 1, 1	1650
ES7		2, 2, 1	1000
ES8	3	3, 3, 1	700
ES9		6, 3, 1	100
ES10		3, 6, 1	100

Table 8
Common settings used for all experiments

Variable	Value
S_{op}	{+, −, ×, ÷}
Maximum generation	250
Tournament size	5
Initial W_c	0.95
Initial W_m	0.05
IH	8

into training set and test set. In this way, either further re-separating or using cross-validation method is unnecessary. We perform every dataset with each experiment setting 10 times to evaluate its average performance. In other words, 90 experiments for each setting are conducted. Large population contains more individuals and usually has better performance. To eliminate the influence of population size, we set the population sizes of 2-layer and 3-layer settings to be smaller so that the number of total individuals of all populations is approximate to 5000.

Classification accuracies of FLGP regarding to *cancer*, *diabetes*, and *heart* problems are shown in Tables 9–11, respectively. From those tables, M1 is found better than M2 in either feature selection or accuracy in most cases. In total 90 average accuracies, M1 outperforms M2 66 times. Best accuracies of these nine problems are derived from M1 method.

Table 9
The accuracy and the average number of used features of 10 settings

	Feature Selection	Cancer Distribution 1		Cancer Distribution 2		Cancer Distribution 3	
ES1	M1	0.9747(6.9)	>	**0.9425**(6.7)	>	0.9540(6.5)	<
	M2	0.9701(7.6)		0.9391(7.0)		0.9546(7.1)	
ES2	M1	0.9695(**4.3**)	>	0.9362(4.3)	>	0.9534(4.6)	<
	M2	0.9598(8.4)		0.9356(8.2)		0.9552(7.7)	
ES3	M1	0.9741(5.2)	>	0.9351(4.4)	>	0.9569(5.0)	>
	M2	0.9718(8.0)		0.9333(8.5)		0.9471(8.1)	
ES4	M1	0.9701(5.2)	>	0.9305(4.6)	>	0.9592(5.6)	>
	M2	0.9557(8.6)		0.9414(7.3)		0.9529(8.5)	
ES5	M1	**0.9776**(5.2)	>	0.9339(4.5)	<	**0.9598**(4.6)	>
	M2	0.9684(8.9)		0.9379(8.2)		0.9529(8.0)	
ES6	M1	0.9747(4.8)	>	0.9356(**4.0**)	>	**0.9598**(5.2)	>
	M2	0.9713(7.5)		0.9017(8.5)		0.9218(7.9)	
ES7	M1	0.9741(4.8)	>	0.9356(4.3)	>	0.9563(**4.4**)	>
	M2	0.9701(8.6)		0.9345(8.1)		0.9460(8.5)	
ES8	M1	0.9730(5.0)	>	0.9333(4.7)	<	0.9557(4.7)	>
	M2	0.9621(8.3)		0.9402(7.9)		0.9517(8.2)	
ES9	M1	0.9672(4.7)	<	0.9402(5.1)	>	0.9575(5.1)	>
	M2	0.9741(8.7)		0.9397(7.9)		0.9448(8.1)	
ES10	M1	0.9713(5.1)	>	0.9408(4.8)	>	0.9592(4.8)	>
	M2	0.9339(8.6)		0.9391(8.2)		0.9517(7.8)	

Best accuracy and fewest average number of used features are marked bold; >, =, and < indicate the relation between the accuracy of using M1 and the accuracy of using M2.

Table 10
The accuracy and the average number of used features of 10 settings

	Feature Selection	Diabetes Distribution 1		Diabetes Distribution 2		Diabetes Distribution 3	
ES1	M1	0.6885(6.8)	>	0.6969(6.5)	<	0.7052(**5.7**)	<
	M2	0.6833(7.2)		0.7063(7.1)		0.7078(7.3)	
ES2	M1	**0.7276**(**6.1**)	>	0.7000(**5.6**)	>	0.7276(6.1)	>
	M2	0.6906(7.8)		0.6813(8.0)		0.6906(7.8)	
ES3	M1	0.7146(6.8)	>	0.6948(6.8)	>	**0.7354**(6.4)	>
	M2	0.7078(7.9)		0.6917(8.0)		0.7307(8.0)	
ES4	M1	0.6938(7.3)	>	0.7094(6.1)	>	0.7313(6.5)	>
	M2	0.6854(8.0)		0.6766(8.0)		0.7115(8.0)	
ES5	M1	0.7026(7.6)	>	0.7005(6.5)	>	0.7307(7.2)	>
	M2	0.6818(8.0)		0.6693(8.0)		0.7068(8.0)	
ES6	M1	0.6854(5.9)	<	0.7078(6.7)	>	0.7292(5.9)	>
	M2	0.6885(8.0)		0.6536(8.0)		0.6828(8.0)	
ES7	M1	0.6729(7.0)	<	0.6969(6.8)	>	0.7240(6.9)	>
	M2	0.6792(8.0)		0.6677(8.0)		0.6885(8.0)	
ES8	M1	0.6984(7.4)	>	0.6958(7.0)	>	0.7286(6.4)	>
	M2	0.6807(8.0)		0.6724(8.0)		0.6813(8.0)	
ES9	M1	0.6958(6.9)	>	**0.7104**(7.0)	>	0.7266(5.8)	>
	M2	0.6750(8.0)		0.6734(8.0)		0.6974(7.7)	
ES10	M1	0.6984(6.9)	>	0.6938(6.4)	>	0.7057(6.0)	>
	M2	0.6880(8.0)		0.6479(8.0)		0.6906(8.0)	

Best accuracy and fewest average number of used features are marked bold; >, =, and < indicate the relation between the accuracy of using M1 and the accuracy of using M2.

When M1 is used, an individual may drop its fitness value because having too many features would reduce its chance of beating other individuals in tournament selection. M1 intends to discover individuals that used fewer features without loss of fitness. On the other side, M2 aims to reduce the chance of poor features being selected. Weights of features might be converged under M2 given enough number of generations.

In the following, we analyze the experiment results by grouping relevant experiment settings.

Group 1: ES1

ES1 is the only experiment setting that uses single population containing 5000 individuals. It obtains the best accuracies in the second distribution of *cancer* and the first distribution of *heart*. ES1 also shows the performance of traditional single population GP with the two feature selection methods. M1 does not always outperform M2 in the aspect of classification accuracy, and vice versa.

Group 2: ES2, ES3, ES4, and ES5

These settings use two layers. ES2 and ES3 achieve the best accuracies in the first and the third distribution of *diabetes*, respectively. ES5 performs excellent in the first and the third distribution of *cancer*. It is found that the improvement of increasing the number of populations in first layer is not significant. Using a number of smaller populations redeems its lack in population diversity compared to larger population.

Table 11
The accuracy and the average number of used features of 10 settings

	Feature Selection	Heart Distribution 1		Heart Distribution 2		Heart Distribution 3	
ES1	M1	**0.7760(11.0)**	>	0.9120(17.4)	<	0.8333(**11.3**)	>
	M2	0.7520(14.1)		0.9133(12.6)		0.8293(13.3)	
ES2	M1	0.7387(13.6)	<	0.8973(16.4)	=	0.8387(14.4)	>
	M2	0.7680(18.1)		0.8973(19.2)		0.8333(17.0)	
ES3	M1	0.7693(17.4)	>	0.9067(20.2)	>	0.8173(16.5)	<
	M2	0.7560(24.1)		0.9000(23.6)		0.8240(20.0)	
ES4	M1	0.7533(18.9)	<	0.9040(18.2)	>	0.8120(16.8)	<
	M2	0.7560(22.4)		0.8947(24.7)		0.8133(21.9)	
ES5	M1	0.7600(19.7)	>	0.8947(21.6)	>	0.8320(17.5)	>
	M2	0.7573(22.0)		0.8893(25.3)		0.8187(22.0)	
ES6	M1	0.7547(15.9)	>	0.8987(16.9)	>	0.8320(15.2)	>
	M2	0.7520(18.6)		0.8960(20.8)		0.8200(18.1)	
ES7	M1	0.7613(17.6)	>	0.8947(18.7)	<	0.8240(16.9)	>
	M2	0.7467(21.8)		0.8987(21.9)		0.8013(21.6)	
ES8	M1	0.7467(20.4)	<	0.8560(22.5)	<	0.8227(18.2)	>
	M2	0.7480(23.9)		0.9027(25.8)		0.8000(23.9)	
ES9	M1	0.7613(16.2)	<	0.9080(19.9)	=	0.8440(13.9)	>
	M2	0.7640(24.1)		0.9080(23.1)		0.8240(22.4)	
ES10	M1	0.7440(17.6)	<	**0.9200(16.3)**	>	**0.8453**(12.7)	>
	M2	0.7493(21.4)		0.9133(19.8)		0.8240(17.9)	

Best accuracy and fewest average number of used features are marked bold; >, =, and < indicate the relation between the accuracy of using M1 and the accuracy of using M2.

Group 3: ES6, ES7, ES8, ES9, and ES10

These settings use three layers. ES6 achieves the best accuracy in the third distribution of *cancer*. ES9 achieves the best accuracy in the second distribution of *diabetes*. Both the best accuracies in the second and the third distribution of *heart* are obtained by ES10. Using more layers does not always infer better performance.

Group 4: ES3 and ES6

ES3 and ES6 are grouped together because they have the same population size. ES3 outperforms ES6 in four problems and all problems with M1 and M2, respectively. This phenomenon raises a hypothesis that using more populations in first layer would be better than separating populations into more layers.

Group 5: ES5 and ES7

The reason why ES5 and ES7 are categorized into the same group is the same as above. ES5 outperforms ES7 in six problems and seven problems with M1 and M2, respectively. It seems that Group 5 and Group 6 ensure the hypothesis that arranging populations within fewer layers could be a better way.

Group 6: ES9 and ES10

The last group consists of ES9 and ES10. ES9's first layer contains six populations, which is the number of population ES10 used in second layer. ES10 outperforms ES9 in six problems and three problems with M1 and M2, respectively. Using more populations in first layer means that the training data for the second layer have more features. Consider the high-dimensional *heart* distributions and the eight-dimensional *diabetes* distributions. For *heart*, the three populations of the second layer of ES9 evolve 41 features, including 35 original features and 6 new ones. However, ES10's second layer uses six populations to evolve only 38 features. This explains why ES9 performs worse than ES10 in *heart*, but in *diabetes* is the opposite. The three populations of the second layer of ES9 are sufficient to obtain good results with the given 14 features, where six of them are generated by feature extraction. Once the proportion of new features compared to the original ones is higher enough, using more populations in first layer might be a good configuration.

4. Conclusions

This paper proposes a novel method called FLGP to construct classifier with capabilities of feature selection and feature extraction. FLGP employs multi-population genetic programming technique in a proper multi-layer architecture. By means of a number of experiments, we show that FLGP not only achieves high classification accuracy but also completes feature selection and feature extraction simultaneously. The classification accuracy of FLGP is comparable to traditional single population genetic programming.

When applying FLGP in practice, its configuration is made empirically. Some outcomes are derived from the experiments, including:

1. Feature selection method M1 is preferable.
2. Layer architecture improves classification accuracy.
3. Given a fix number of populations, rather than separating them into more layers, using fewer layers is preferable.
4. For a high-dimensional problem, it is preferable that using fewer populations in successive layers.

Implementation of FLGP can be achieved by either parallel distribution computing environment or serial computing environment. Since the populations are independent to each other, we can perform evolutionary process of each population on a number of different computers at different time and combine results to build a layer afterward.

Our further research will focus on discovering the relation of the number of original and new features. Furthermore, in order to examine the quality of selected features and extracted ones, we will use them with other classification algorithms.

References

Ahmad, A., & Dey, L. (2005). A feature selection technique for classificatory analysis. *Pattern Recognition Letters, 26*, 43–56.

Banzhaf, W., Nordin, P., Keller, R. E., & Framcone, F. D. (1998). *Genetic programming: an introduction on the automatic evolution of computer programs and its application*. San Francisco, CA: Morgan Kaufmann.

Blake, C., Keogh, E., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Irvine, University of California, Department of Information and Computer Science. Available from http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bojarczuk, C. C., Lopes, H. S., & Freitas, A. A. (1999). Discovering comprehensible classification rules using genetic programming: a case study in a medical domain. In *Proceedings of genetic and evolutionary computation conference, Orlando, FL, USA* (pp. 953–958).

Brameier, M., & Banzhaf, W. (2001). A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation, 5*(1), 17–26.

Chien, B. C., Lin, J. Y., & Yang, W. P. (2003). Learning effective classifiers with *Z*-value measure based on genetic programming. *Pattern Recognition, 37*, 1957–1972.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*(3), 131–156.

Falco, I. D., Cioppa, A. D., & Tarantino, E. (2002). Discovering interesting classification rules with genetic programming. *Applied Soft Computing, 23*, 1–13.

Fernández, F., Tomassini, M., & Vanneschi, L. (2003). An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines, 4*, 21–51.

Freitas, A. (1997). A genetic programming framework for two data mining tasks: classification and generalized rule induction. In *Proceedings of second annual conference on genetic programming, Stanford University, USA* (pp. 96–101).

Han, J., & Kamber, M. (2001). *Data mining: concepts and techniques*. Morgan Kaufmann.

Hong, G., Jack, L. B., & Nandi, A. K. (2005). Feature generation using genetic programming with application to fault classification. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 35*(1), 89–99.

Jain, A. K., & Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(2), 153–158.

John, G. H., Kohavi, R., Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of 11th international conference on machine learning, New Brunswick, NJ* (pp. 121–129).

Jolliffe, I. T. (1986). *Principal component analysis*. New York: Springer.

Kishore, J. K., Patnaik, L. M., Mani, V., & Agrawal, V. K. (2000). Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation, 4*(3), 242–258.

Kittler, J. (1978). Feature set search algorithms. In C. H. Chen (Ed.), *Pattern recognition and signal processing* (pp. 41–60). Netherlands: Sijthoff and Noordhoff.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*, 273–324.

Konstam, A. (1998). Group classification using a mix of genetic programming and genetic algorithms. In *Proceedings of the 1998 ACM symposium of applied computing, Atlanta, GA, USA* (pp. 308–312).

Kotani, M., Ozawa, S., Nakai, M., & Akazawa, K. (1999). Emergence of feature extraction function using genetic programming. In *Proceedings of third international conference on knowledge-based intelligent information engineering system, Adelaide, Australia* (pp. 149–152).

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MA: Cambridge: MIT Press.

Kudo, M., & Sklansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition, 33*, 25–41.

Lee, C., & Landgrebe, D. A. (1997). Decision boundary feature extraction for neural networks. *IEEE Transactions on Neural Networks, 8*(1), 75–83.

Lin, J. Y., Chien, B. C., & Hong, T. P. (2002). A function-based classifier learning scheme using genetic programming. In *Proceedings of sixth Pacific-Asia conference on knowledge discovery and data mining, Taipei, Taiwan* (pp. 92–103).

Loveard, T., & Ciesielski, V. (2001). Representing classification problems in genetic programming. In *Proceedings of the 2001 congress on evolutionary computation, Seoul, Seoul, Korea* (pp. 1070–1077).

Ma, J., Theiler, J., & Perkins, S. (2004). Two realizations of a general feature extraction framework. *Pattern Recognition, 37*, 875–887.

Mao, J., & Jain, A. K. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks, 6*(2), 296–317.

Muni, D. P., Pal, N. R., & Das, J. (2004). A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation, 8*(2), 183–196.

Muni, D. P., Pal, N. R., & Das, J. (2006). Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on System, Man, and Cybernetics – Part B: Cybernetics, 36*(1), 106–117.

Oh, I. S., Lee, J. S., & Moon, B. R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1424–1437.

Otero, F. E. B., Silva, M. M. S., Freitas, A. A., & Nievola, J. C. (2003). Genetic programming for attribute construction in data mining. In *Proceedings of 6th European conference on genetic programming, Essex, UK* (pp. 384–393).

Park, C. H., & Park, H. (2004). Nonlinear feature extraction based on centroids and kernel functions. *Pattern Recognition, 37*, 801–810.

Pernkopf, F. (2005). Bayesian network classifiers versus selective *k*-NN classifier. *Pattern Recognition, 38*, 1–10.

Prechelt, L. (1994). PROBEN1—a set of neural network benchmark problems and benchmarking rules. Tech. Rep. 21/94, Univ. Karlsruhe, Karlsruhe, Germany.

Pudil, P., Novovicova, J., & Kitter, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters, 15*, 1119–1125.

Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., & Jain, A. K. (2000). Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation, 4*, 164–171.

Rizki, M. M., Zmuda, M. A., & Tamburino, L. A. (2002). Evolving pattern recognition systems. *IEEE Transactions on Evolutionary Computation, 6*(6), 594–609.

Sherrah, J., Bogner, R. E., & Bouzerdoum, A. (1996). Automatic selection of features for classification using genetic programming. In *Proceedings of Australian and New Zealand conference on intelligent information systems, Adelaide, SA, Australia* (pp. 284–287).

Siedlecki, W., & Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters, 10*, 335–347.

Smith, M. G., & Bull, L. (2004). Using genetic programming for feature creation with a genetic algorithm feature selector. In *Proceedings of 8th international conference on parallel problem solving from nature, Birmingham, UK* (pp. 1163–1171).

Wang, X., & Paliwal, K. K. (2003). Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition. *Pattern Recognition, 36*, 2429–2439.