

Paper:

A Combination of Shuffled Frog-Leaping Algorithm and Genetic Algorithm for Gene Selection

Cheng-San Yang*, Li-Yeh Chuang**, Chao-Hsuan Ke***, and Cheng-Hong Yang***

*Institute of biomedical engineering, National Cheng Kung University, Tainan, Taiwan 70101

**Department of Chemical Engineering, I-Shou University, Kaohsiung, Taiwan 84001

***Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan 80778

E-mail: chyang@cc.kuas.edu.tw

[Received April 20, 2007; accepted September 22, 2007]

Microarray data referencing to gene expression profiles provides valuable answers to a variety of problems, and contributes to advances in clinical medicine. The application of microarray data to the classification of cancer types has recently assumed increasing importance. The classification of microarray data samples involves feature selection, whose goal is to identify subsets of differentially expressed gene potentially relevant for distinguishing sample classes and classifier design. We propose an efficient evolutionary approach for selecting gene subsets from gene expression data that effectively achieves higher accuracy for classification problems.

Our proposal combines a shuffled frog-leaping algorithm (SFLA) and a genetic algorithm (GA), and chooses genes (features) related to classification. The K-nearest neighbor (KNN) with leave-one-out cross validation (LOOCV) is used to evaluate classification accuracy.

We apply a novel hybrid approach based on SFLA-GA and KNN classification and compare 11 classification problems from the literature. Experimental results show that classification accuracy obtained using selected features was higher than the accuracy of datasets without feature selection.

Keywords: gene expression data, classification, SFLA, GA, KNN

1. Introduction

DNA microarray technology, which enables thousands of gene expression activation levels to be simultaneously monitored and measured in a single experiment, is universally used in medical diagnosis and gene analysis. Many microarray analysis research projects have focused on clustering analysis, used to analyze group genes showing a pattern correlated with gene expression data providing insight into gene interactions and functions [1–3], and classification accuracy, which discriminates between sample classes and predicts the relative importance of individual genes in sample classification [4].

Gene expression data typically has a high dimension and a small sample, which makes general classification testing and training difficult. Obtaining correct classification is also difficult. The purpose of classification is to build an efficient model for predicting data class membership of data, expected to produce the correct label for training data, and to predict the label for unknown data correctly.

Generally, only relatively small numbers of gene expression data are strongly correlated with a certain phenotype compared to the total number of genes investigated, meaning that of thousands of genes investigated, only a small number correlate significantly with a phenotype. To analyze gene expression profiles correctly, feature (gene) selection is crucial to classification.

Classifying microarray data samples involves feature selection and classifier design. Reliable selection for genes relevant to sample classification is needed to increase predictive accuracy and to avoid incomprehensibility. Such selection should be based on the total number of genes investigated. Methods suggested for use in used feature selection have include genetic algorithms [5], branch and bound algorithms [6], sequential search algorithms [7], mutual information [8], tabu search [9], entropy-based methods [10], regularized least squares [11], random forests [12], instance-based methods [13], and least squares support vector machines [14].

We embedded a genetic algorithm (GA) in a shuffled frog-leaping algorithm (SFLA) and to serve as a local optimizer for each generation in implementing feature selection. To classify 11 classification problems taken from the literature, we used the K-nearest neighbor (KNN) with leave-one-out cross-validation (LOOCV) based on Euclidean distance calculation to evaluate the SFLA and GA. Results showed that our proposal achieves superior classification accuracy when applied to the 11 datasets, compared to previous methods. The number of genes needing to be selected can also be significantly reduced.

2. Method

Our proposal is briefly explained in the sections below.

2.1. Shuffled Frog-Leaping Algorithm

The shuffled frog-leaping algorithm (SFLA) combines the benefits of a gene-based memetic algorithm (MA) and social behavior-based particle swarm optimization (PSO) [15]. A MA is a gene-based optimization algorithm similar to a GA. In a GA, chromosomes are represented as a string consisting of a set of elements called “genes.” Chromosomes in MA are represented by elements, called “memes.” MAs and GA differ in that the MA implements a local search before crossover or mutations to determine offspring. After the local search, new offspring that obtain better results than original offspring replace original offspring, thus continuing the evolutionary process.

PSO is an evolutionary algorithm [16] in which individual solutions are called “particle” (analogous to the GA chromosome), but PSO does not apply crossover and mutation to construct a new particle. Each particle changes its position and velocity based on the individual particle’s optimal solution *pBest* and the corporate optimal solution *gBest* until a global optimal solution is found.

The SFLA is derived from a virtual population of frogs in which individual frogs are equivalent to the GA chromosomes, and represent a set of solutions. Each frog is distributed to a different subset of the whole population called a memplex. An independent local search is conducted for each frog memplex, in what is called memplex evolution. After a defined number of memetic evolutionary steps, frogs are shuffled among memplexes [17], enabling frogs to interchange messages among different memplexes and ensure that they move to an optimal position, similar to particles in PSO. Local search and shuffling continue until defined convergence criteria are met.

SFLA have demonstrated effectiveness in a number of global optimization problems difficult to solve using other methods, i.e., water distribution and groundwater model calibration problems [18].

(1) Initial population

An initial population of *P* frogs is created randomly for a *S*-dimensional problem. A frog *i* is represented by *S* variables, such as $F_i = (f_{i1}, f_{i2}, \dots, f_{is})$.

(2) Sorting and distribution

Frogs are sorted in descending order based on their fitness values, then the entire population is divided into *m* memplexes, each containing *n* frogs (i.e., $P = m \times n$). The first frog is distributed to the first memplex, the second frog to the second, the *m* frog to the *m* memplex, and the *m* + 1 frog to the first memplex, etc.

(3) Memplex evolution

Within each memplex, frogs with the best and the worst fitness are identified as X_b and X_w , and the frog with the global best fitness is identified as X_g separately. To improve the worst solution, an equation similar to PSO is used to update the worst solution, e.g., Eqs. (1) and (2):

Change in frog position

$$(D_i) = rand() \cdot (X_b - X_w) \dots \dots \dots (1)$$

New position $X_w =$ current position $X_w + D_i$

$$(D_{max} \geq D_i \geq -D_{max}) \dots \dots \dots (2)$$

where *rand()* is a random number between 0 and 1; and D_{max} is the maximum change allowed in a frog’s position. If this process produces a better solution, it replaces the worst frog. If Eqs. (1) and (2) do not improve the worst solution, X_b of Eq. (1) is changed to X_g and adapted to Eq. (3):

Change in frog position

$$(D_i) = rand() \cdot (X_g - X_w) \dots \dots \dots (3)$$

If Eqs. (1) and (3) do not improve the worst solution, then a new solution is randomly generated to replace that worst frog.

(4) Shuffling

After a defined number of memplex evolution steps, all frogs of memplexes are collected, and sorted in descending order based on their fitness. Step 2 divides frogs into different memplexes again, and then step 3 is done.

(5) Terminal condition

If a global solution or a fixed iteration number is reached, the algorithm stops.

2.2. Genetic Algorithms

A genetic algorithm (GA) is a general adaptive optimization search based on a direct analogy to Darwinian natural selection and genetics in biological systems. It is a promising alternative to conventional heuristics. A GA uses chromosomes, in string form, to represent a solution. Each chromosome consists of a set of elements called “genes” that hold a set of values for optimization variables. In a *S*-dimensional problem, for example, each chromosome *i* is represented by *S* variables, i.e., $C_i = (c_{i1}, c_{i2}, \dots, c_{is})$.

Before the GA is executed, chromosomes are initially randomly produced and represent the population. Different fitness functions must be used for different selection problems, and each fitness of a chromosome must be calculated by this fitness function.

To simulate natural survival-of-the-fittest, the optimal (“best”) chromosomes exchange information to produce offspring chromosomes through crossover and mutation. If the offspring chromosome is superior to the worst chromosome of the population, then it replaces the worst chromosome, and the process continues until a global solution or a fixed number of iterations is reached.

GA have been applied to a variety of problems, such as scheduling [19], machine learning [20], multiple objective problems [21], feature selection [22], data mining [23], and traveling salesman problem [24], as detailed in John Holland [25].

2.3. K-Nearest Neighbor

The K-nearest neighbor (KNN), introduced by Fix and Hodges in 1951 [27], is a popular nonparametric