2009 International Conference on Machine Learning and Computing
IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore

Catfish Binary Particle Swarm Optimization for Feature Selection

Li-Yeh Chuang[1], Sheng-Wei Tsai[2], Cheng-Hong Yang[2+]

[1]Institute of Biotechnology and Chemical Engineering, I-Shou University, Kaohsiung, Taiwan 80041
[2]Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan 80778

Abstract. The feature selection process constitutes a commonly encountered problem of global combinatorial optimization. This process reduces the number of features by removing irrelevant, noisy, and redundant data, thus resulting in acceptable classification accuracy. Feature selection is a preprocessing technique with great importance in the fields of data analysis and information retrieval processing, pattern classification, and data mining applications. This paper presents a novel optimization algorithm called catfish binary particle swarm optimization (CatfishBPSO), in which the so-called catfish effect is applied to improve the performance of binary particle swarm optimization (BPSO). This effect is the result of the introduction of new particles into the search space ("catfish particles"), which replace particles with the worst fitness by the initialized at extreme points of the search space when the fitness of the global best particle has not improved for a number of consecutive iterations. In this study, the K-nearest neighbor (K-NN) method with leave-one-out cross-validation (LOOCV) was used to evaluate the quality of the solutions. CatfishBPSO was applied and compared to six classification problems taken from the literature. Experimental results show that CatfishBPSO simplifies the feature selection process effectively, and either obtains higher classification accuracy or uses fewer features than other feature selection methods.

Keywords: Feature Selection, Catfish Binary Particle Swarm Optimization, K-nearest neighbor, Leave-one-out Cross-Validation.

1. Introduction

Feature selection is a preprocessing technique for effective data analysis. The purpose of feature selection is the selection of optimal subsets, which are necessary and sufficient for solving the problem. Feature selection improves the predictive accuracy of algorithms by reducing the dimensionality, removing irrelevant features, and reducing the amount of data needed for the learning process. This can be done because not all available features are relevant for the classification process. Recently, feature selection has been successfully employed to effectively solve classification problem in various areas, such as pattern recognition [1], data mining [2, 3], and multimedia information retrieval [4, 5], and other areas where feature selection can be applied to. Furthermore, the filter methods and wrapper methods are two general approaches of feature selection. Filter methods define the relevant features without prior classification of the data [6]. Wrapper methods on the other hand incorporate classification algorithms to search for and select relevant features [7]. The wrapper methods generally outperform filter methods in terms of classification accuracy. The CatfishBPSO introduced in this paper belongs to the wrapper method. A high number of selected features do not necessarily translate into high classification accuracy for many pattern classification problems. In some cases, the performance of algorithms devoted to speed and predictive accuracy of the data characterization can even decrease because features may be irrelevant or misleading, or due to spurious correlations. These factors can have a negative impact on the classification process during the learning stage. Ideally, feature selection method reduces the cost of feature measurement, and increase classifier efficiency and classification accuracy.

[+] Corresponding author. Tel.: + 886 7 381 4526 ext 5639; fax: +886 7 383 6844.
E-mail address: chyang@cc.kuas.edu.tw.

The present study proposes a novel optimization algorithm for feature selection called catfish binary particle swarm optimization (CatfishBPSO). In CatfishBPSO, the so-called "catfish" effect introduces a competition function into a group of individuals. Catfish particles are introduced into the search space if the fitness of *gbest* is not improved (i.e. unchanged) a number of consecutive iteration. These catfish particles are introduced at extreme positions of the search space and will initialize a new search from these extreme positions. The catfish particles open up new opportunities for finding better solutions, and guide the entire swarm to promising new regions of the search space. The K-nearest neighbor method (K-NN) with leave-one-out cross-validation (LOOCV) based on Euclidean distance calculations was used to evaluate the quality of solutions for six classification problems taken from the literature. Experimental results show that CatfishBPSO simplifies the feature selection process effectively, and either obtains higher classification accuracy or uses fewer features than other feature selection methods.

2. Methods

2.1. Binary Particle Swarm Optimization (BPSO)

PSO is a population based optimization tool, which was originally introduced as an optimization technique for real-number spaces. In PSO, each particle is analogous to an individual "fish" in a school of fish. A swarm consists of *N* particles moving around a *D*-dimensional search space. The process of PSO is initialized with a population of random particles and the algorithm then searches for optimal solutions by continuously updating generations. Each particle makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. The position of the i_{th} particle can be represented by $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The velocity for the i_{th} particle can be written as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The positions and velocities of the particles are confined within $[X_{\min}, X_{\max}]^D$ and $[V_{\min}, V_{\max}]^D$, respectively. The best previously visited position of the i_{th} particle is denoted its individual best position $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, a value called $pbest_i$. The best value of the all individual $pbest_i$ values is denoted the global best position $g = (g_1, g_2, \ldots, g_D)$ and called *gbest*. At each generation, the position and velocity of the i_{th} particle are updated by $pbest_i$ and *gbest* in the swarm. However, many optimization problems occur in a space featuring discrete, qualitative distinctions between variables and between levels of variables. For this reason, Kennedy and Eberhart introduced binary PSO (BPSO), which can be applied to discrete binary variables. In a binary space, a particle may move to near corners of a hypercube by flipping various numbers of bits; thus, the overall particle velocity may be described by the number of bits changed per iteration [8]. In BPSO, each particle is updated can be based on the following equations:

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times r_1 \times \left(pbest_{id} - x_{id}^{old}\right) + c_2 \times r_2 \times \left(gbest_d - x_{id}^{old}\right) \tag{1}$$

$$\text{if } v_{id}^{new} \notin (V_{\min}, V_{\max}) \text{ then } v_{id}^{new} = \max\left(\min\left(V_{\max}, v_{id}^{new}\right), V_{\min}\right) \tag{2}$$

$$S\left(v_{id}^{new}\right) = \frac{1}{1 + e^{-v_{id}^{new}}} \tag{3}$$

$$\text{if } \left(r_3 < S\left(v_{id}^{new}\right)\right) \text{ then } x_{id}^{new} = 1 \text{ else } x_{id}^{new} = 0 \tag{4}$$

In these equations, *w* is the inertia weight that controls the impact of the previous velocity of a particle on its current one, r_1, r_2 and r_3 are random numbers between (0, 1), and c_1 and c_2 are acceleration constants, which control how far a particle will move in a single generation. Velocities v_{id}^{new} and v_{id}^{old} denote the velocities of the new and old particle, respectively. x_{id}^{old} is the current particle position, and x_{id}^{new} is the new, updated particle position. In Eq. 2, particle velocities of each dimension are tried to a maximum velocity $V_{\max}$. If the sum of accelerations causes the velocity of that dimension to exceed $V_{\max}$, then the velocity of that dimension is limited to $V_{\max}$. $V_{\max}$ and $V_{\min}$ are user-specified parameters (in our case $V_{\max} = 6$, $V_{\min} = -6$). The position of particles after updating is calculated by the function $S(v_{id}^{new})$ (Eq. 3) [9]. If $S(v_{id}^{new})$ is larger than r_3, then its position value is represented by {1} (meaning this position is selected for the next update). If $S(v_{id}^{new})$ is smaller than r_3, then its position value is represented by {0} (meaning this position is not selected for the next update) [10].

2.2. Catfish Binary Particle Swarm Optimization (CatfishBPSO)

The catfish effect derives its name from an effect that Norwegian fishermen observed when they introduced catfish into a holding tank for caught sardines. The introduction of catfish, which is different from sardines, into the tank resulted in the stimulation of sardine movement, thus keeping the sardines alive and therefore fresh for a longer time. Similarly, CatfishBPSO introduces catfish particles to stimulate a renewed search by the "sardine" particles. In other words, these catfish particles guide particles which are trapped in a local optimum on towards a new region of the search space, and thus to potentially better particle solutions. In original BPSO, if the distance between *gbest* and the surrounding particles is small, each particle is considered a part of the cluster around *gbest* and will only move a very small distance in the next generation. To avoid this premature convergence, catfish particles are introduced and replace the 10% of original particles with the worst fitness value of the swarm. These catfish particles are essential for the success of a given optimization task. The introduction of catfish particles in original BPSO is simple and can be done without increasing the computational complexity of the process. Catfish particles overcome the inherent defects (premature convergence) of BPSO by initializing a new search over the entire search space from its extreme points. A more detailed description of the entire CatfishBPSO process can be found in [11].

2.3. K-Nearest Neighbor

The K-nearest neighbor (K-NN) method is one of the most popular nonparametric methods [12, 8] used for classification of new objects based on attributes and training samples. K-NN consists of a supervised learning algorithm which instantly classifies the result of a query instance based on the majority of the K-nearest neighbor category. The advantage of the K-NN method is that it is simple and easy to implement. K-NN is not negatively affected when training data are large, and indifferent to noisy training data [12]. In this study, the feature subset was measured by the leave-one-out cross-validation (LOOCV) of one nearest neighbor (1-NN), and calculated using their Euclidean distance. In LOOCV procedure, if *n* data will be classified, the data are divided into one testing and *n*-1 training samples at each iteration during the evaluation process, and finally a classifier is constructed by training the *n*-1 training samples. Then, the category of testing sample can be judged by the classifier. The pseudo-codes for CatfishBPSO and 1-NN are given below.

CatfishBPSO Pseudo-code

```
Begin
  Randomly initialize particles swarm
  while (number of iterations, or the stopping criterion is not met)
    Evaluate fitness of particle swarm by 1-NN( )
    for $n$ = 1 to number of particles
      Find $pbest_i$ and $gbest$
      for $d$ = 1 to number of dimension of particle
        update the position of particles by Eq. 1-4
      next $d$
    next $n$
    if fitness of gbest is the same three times then
      Sort the particle swarm via fitness from best to worst
      for $n$ = number of Nine-tenth particles to number of particles
        if rand number > 0.5 then
        for $d$ = 1 to number of dimension of particle
          The position of catfish = 1 (Max of the search space)
        next $d$
      else
      for $d$ = 1 to number of dimension of particle
        The position of catfish = 0 (Min of the search space)
      next $d$
    next $n$
  end if
  next generation until stopping criterion
end
```

1-NN pseudo-code

```
Begin
  for $i$ = 1 to sample number of classification problem
    for $j$ = 1 to sample number of classification problem
      for $k$ = 1 to dimension number of classification problem
        $dist_i = dist_i + (data_{ik} - data_{jk})^2$
      next $k$
      if $dist_i < nearest$ then
        $class_i = class_j$
        $nearest = dist_i$
      end if
    next $j$
  next $i$
  for $i$ = 1 to sample number of classification problem
    if $class_i$ = real class of testing data then correct = correct + 1
    end if
  next $i$
  Fitness value = correct / number of testing data
end
```

3. Experimental results and Discussion

In order to investigate the effectiveness and performance of the CatfishBPSO algorithm for classification problems, we chose six classification problems from the literature in our experiment. These data sets were obtained from the UCI Repository [13]. The data format is shown in Table 1. Three types of classification problems, small-, medium-, and large-sized groups, were tested. If the number of features is between 10 and 19, the sample group can be considered small; the Vehicle and Segmentation problems constitute such small sample groups. If between 20 and 49, the sample group test problems are of medium size. Groups in this category included the WDBC, Ionosphere, and Satellite problems. If greater than 50, the test problems are large sample group problems; this group included the Sonar problem. The 1-NN method with leave-one-out cross-validation (LOOCV) was used to evaluate all data sets.

Table 1. Format of classification test problems

Data sets	Number of samples	Number of classes	Number of features	Evaluation method
Vehicle	846	4	18	1-NN
Segmentation	2310	7	19	1-NN
WDBC	569	2	30	1-NN
Ionosphere	351	2	34	1-NN
Satellite	6435	6	36	1-NN
Sonar	208	2	60	1-NN

Table 2. The number of generations setting

Data sets	SGA	HGA (1)	HGA (2)	HGA (3)	HGA (4)	Catfish-BPSO
Vehicle	10000	600	200	120	80	70
Segmentation	10000	600	200	120	80	70
WDBC	20000	600	200	120	80	70
Ionosphere	25000	600	200	120	80	70
Satellite	25000	600	200	120	80	70
Sonar	40000	600	200	120	80	70

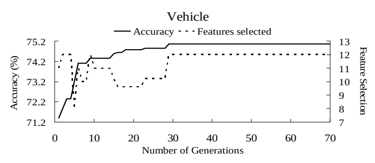
Legend: SGA: simple genetic algorithm, HGA: hybrid genetic algorithm (Oh *et al.* [15]), CatfishBPSO: the proposed method

In our experiment, the same conditions were used to compare the performance of GA, HGAs and CatfishBPSO algorithms, i.e. population size = 20, and the K = 1 in part of K -Nearest Neighbor. The other parameters of CatfishBPSO are set as follows: generations = 70 (maximum number of iterations), w = 1.0, and $c_1 = c_2 = 2$ [14]. The number of generations for the SGA, HGAs and CatfishBPSO is listed in Table 2. In Table 3, the classification accuracy and selected number of features is shown for the data sets tested with CatfishBPSO and compared to other feature selection methods from the literature [15]. In this paper, each data set was executed for 5 runs by the proposed method and this number is the same as Oh *et al.* used [15]. From Table 3, it can be seen that CatfishBPSO obtained the highest classification accuracy for all classification problems. Fig. 1 plot the number of generations versus the accuracy and the number of features selected for the six data sets, and shows the reciprocal relationship between the achieved accuracy and the number of selected feature. It can be seen that removal of the irrelevant features is benefits to the classification accuracy, and this fact also proves that classification accuracy does not have to be negatively affected by a lower number of selected features. As long as the chosen features contain enough feature classification information, higher classification accuracy can be achieved. In Oh *et al.* [15], the classification accuracies were measured by the four values D/5, 2D/5, 3D/5, and 4D/5 of the total number of features D. However, the optimal number of features for each test problem is unknown. From the experimental result, it can be concluded that the classification accuracy can not breakthrough if the number of features was constraints during the selection procedure. Moreover, the method of Oh et *al.* is rather time-consuming compared to the method we propose here, because the computational complexity for SGA and HGAs can be derived as O(PG) and O(PG+LS) respectively [16], where P is the population size, G is the number of generations, and LS is the local search operation in HGAs. For the CatfishBPSO method, the computational complexity can be derived as O(PG). Consequently, the efficiency of CatfishBPSO is better than SGA and HGAs. Employing the 1-NN method with LOOCV as a fitness function for CatfishBPSO has two distinct advantages; first, the calculation time can be decreased, and secondly, higher classification accuracy can be obtained.

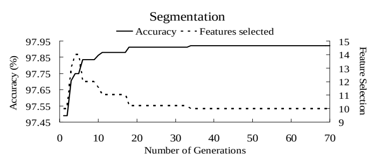
Table 3. Classification accuracy and selected number of features for the tested data sets

Data sets	SFS	PTA	SFFS	SGA	HGA(1)	HGA(2)	HGA(3)	HGA(4)	CatfishBPSO
Vehicle	69.50 (11)	71.75 (11)	73.52 (7)	72.97 (7)	73.52 (7)	73.52 (7)	73.52 (7)	73.52 (7)	**75.06 (12.0)**
Segmentation	92.95 (8)	92.95 (8)	92.95 (8)	92.95 (8)	92.95 (8)	92.95 (8)	92.95 (8)	92.95 (8)	**97.92 (10.0)**
WDBC	94.02 (18)	94.20 (18)	94.20 (6)	93.95 (12)	94.06 (12)	94.38 (6)	94.13 (18)	94.27 (12)	**98.17 (14.2)**
Ionosphere	93.45 (7)	93.45 (7)	93.45 (7)	94.70 (7)	95.38 (7)	95.56 (14)	95.56 (7)	95.50 (7)	**96.01 (8.0)**
Satellite	90.45 (22)	90.70 (29)	91.45 (22)	91.36 (22)	91.37 (22)	91.44 (22)	91.45 (22)	91.41 (22)	**91.53 (26.8)**
Sonar	91.82 (48)	92.31 (48)	93.75 (24)	95.49 (24)	95.86 (24)	95.96 (24)	96.34 (24)	96.15 (24)	**96.92 (30.2)**
Average	88.70	89.23	89.89	90.24	90.52	90.64	90.66	90.63	**92.60**

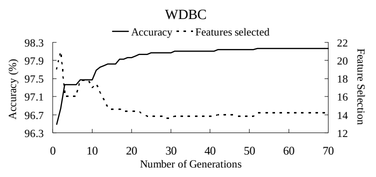
Legend: SFS: sequential forward search, PTA: plus and take away, SFFS: sequential forward floating search, SGA: simple genetic algorithm, HGA: hybrid genetic algorithm (Oh *et al.* [15]), CatfishBPSO: the proposed method. Highest values are in bold-type. For GAs and CatfishBPSO, x and y in x (y) indicates the average accuracy and selected number of features, respectively.



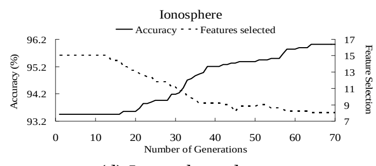
(a) Vehicle data set



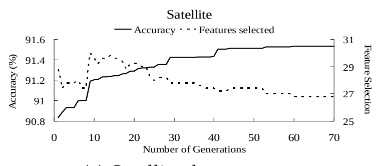
(b) Segmentation data set



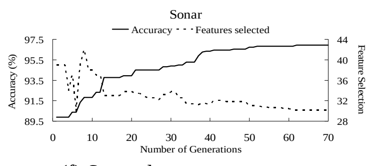
(c) WDBC data set



(d) Ionosphere data set



(e) Satellite data set



(f) Sonar data set

Fig. 1 The number of generations versus the accuracy and the number of features selected for the six data sets

4. Conclusions

In this study, a novel optimization algorithm, CatfishBPSO, is introduced in which the catfish effect is applied to perform feature selection and improve the performance of BPSO. The K-NN method with LOOCV served as an evaluator of the CatfishBPSO fitness functions. In CatfishBPSO, catfish particles initialize a new search from extremes of the search space when the gbest value stays unchanged after three numbers of iterations. Better solutions can be found by guiding the entire particle swarm to more promising regions in the search space. The experimental results show that CatfishBPSO simplifies feature selection by reducing the total number of necessarily selected features effectively, and that it obtained the highest in all of the data set test problems compared to other feature selection methods from the literature. CatfishBPSO can be used as a preprocessing tool to optimize the feature selection process, since it increases either the classification accuracy, reduces the number of necessarily selected features for classification, or does both. From computational complexity and statistical analysis, it can be concluded that the proposed CatfishBPSO method demonstrate higher efficiency than the other methods. In the future, CatfishBPSO could conceivably be applied to problems in other areas.

5. References

[1] S. Gunal, and R. Edizkan. Subspace based feature selection for pattern recognition. *Information Sciences*. 2008, 178 (19): 3716-3726.

[2] M.J. Martin-Bautista, and M.-A. Vila. A Survey of Genetic Feature Selection in Mining Issues. *Proc. of Congress on Evolutionary Computation*. 1999, 2: 1314-1321.

[3] S. Piramuthu. Evaluating Feature Selection Methods for Learning in Data Mining Applications. *Proc. IEEE 31st Ann. Hawaii Int'l Conf. System Science*. 1998, 5: 294-301.

[4] M.S. Lew (Ed.). *Principles of Visual Information Retrieval*. Springer press, 2001.

[5] Y. Liu, and F. Dellaert. A Classification Based Similarity Metric for 3D Image Retrieval. *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 1998, pp. 800-805.

[6] H. Liu, and R. Setiono. A probabilistic approach to feature selection-A filter solution. *Proc. 13th Int'l Conf. Mach. Learn*. 1996, pp. 319-327.

[7] R. Kohavi, and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*. 1997, 97 (1-2): 273-324.

[8] E. Fix, and J.L. Hodges, Discriminatory Analysis-Nonparametric Discrimination: Consistency Properties. *Project 21-49-004, Report 4, US Air Force School of Aviation Medicine, Randolph Field.* 1951, pp. 261-279.

[9] K. Crammer, and Y. Singer. On the learn ability and design of output codes for multiclass problems. *Machine Learning*. 2002, 47 (2-3): 201-233.

[10] J. Kennedy, and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. *Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, IEEE International Conference*, 1997, 5: 4104-4108.

[11] L.Y. Chuang, S.W. Tsai, and C.H. Yang. Catfish Particle Swarm Optimization. *IEEE Swarm Intelligence Symposium 2008 (SIS 2008)*. 2008, pp. 1-5.

[12] T. Cover, and P. Hart. Nearest neighbor pattern classification. *Proc. IEEE Trans. Information Theory*. 1967, 13 (1): 21-27.

[13] P.M. Murphy, and D.W. Aha. UCI Repository of Machine Learning Databases. *Technical report, Department of Information and Computer Science*, University of California, Irvine, Calif., 1994. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[14] J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.

[15] I.-S. Oh, J.-S. Lee, and B.-R. Moon. Hybrid Genetic Algorithms for Feature Selection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2004, 26 (11): 1424-1437.

[16] Z. Zhu, Y.S. Ong, and M. Dash. Wrapper-Filter Feature Selection Algorithm Using a Memetic Framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B.* 2007, 37 (1): 70-76.