패턴 인식에서 특징 선택을 위한 개미 군락 최적화

Ant Colony Optimization for Feature Selection in Pattern Recognition

오일석*, 이진선**
전북대학교 컴퓨터공학부/영상정보신기술연구소*, 우석대학교 게임콘텐츠학과**

Il-Seok Oh(isoh@chonbuk.ac.kr)*, Jin-Seon Lee(jslee@woosuk.ac.kr)**

요약

이 논문은 특징 선택에 사용되는 개미 군락 최적화의 수렴 특성을 개선하기 위해 선택적 평가라는 새로운 기법을 제시한다. 이 방법은 불필요하거나 가능성이 덜한 후보 해를 배제함으로써 계산량을 줄인다. 이 방법은, 그런 해를 찾아내는데 사용할 수 있는 페로몬 정보 때문에 구현이 가능하다. 문제 크기에 따른 알고리즘의 적용가능성을 판단할 목적으로, 특징 선택에 사용되는 세 가지 알고리즘인 탐욕 알고리즘, 유전 알고리즘, 그리고 개미 군락 최적화의 계산 시간을 분석한다. 엄밀한 분석을 위해 원자 연산이라는 개념을 사용한다. 실험 결과는 선택적 평가를 채택한 개미 군락 최적화가 계산 시간과 인식 성능 모두에서 우수함을 보여준다.

■ 중심어 : | 특징선택 | 그리디 알고리즘 | 유전알고리즘 | 개미 군락 최적화 | 패턴인식 |

Abstract

This paper propose a novel scheme called *selective evaluation* to improve convergence of ACO (ant colony optimization) for feature selection. The scheme cutdown the computational load by excluding the evaluation of unnecessary or less promising candidate solutions. The scheme is realizable in ACO due to the valuable information, pheromone trail which helps identify those solutions. With the aim of checking applicability of algorithms according to problem size, we analyze the timing requirements of three popular feature selection algorithms, greedy algorithm, genetic algorithm, and ant colony optimization. For a rigorous timing analysis, we adopt the concept of atomic operation. Experimental results showed that the ACO with selective evaluation was promising both in timing requirement and recognition performance.

■ keyword : | Feature Selection | Greedy Algorithm | Genetic Algorithm | Ant Colony Optimization | Pattern Recognition |

I. Introduction

One of the most important tasks in building a pattern recognition system is to design discriminatory features. The feature set should be optimal in discriminating different classes of patterns. It also should be small to be desirable in computational load. Practically the over-production method is used, which extracts hundreds of features from a character sample [1]. Some designers extract several different types of

접수번호 : #100119-001
접수일자 : 2010년 01월 19일
심사완료일 : 2010년 04월 02일
교신저자 : 이진선, e-mail : ijslee@woosuk.ac.kr

features and combine them to get a larger feature set. Those feature sets inevitably include useless and/or redundant features. So if we identify and remove them, the resulting classifier becomes more compact which leads to a recognition system requiring less memory and less computation time. In some circumstances, the compact classifier shows improved recognition performance since a smaller-size classifier is less sensitive to the over-fitting [2]. This is the case especially when the training set is small. The role of feature selection is to reduce the feature set size by removing useless, redundant, or least useful features. The algorithms should be both efficient in terms of computation time and effective in finding near-optimal solutions [3].

This paper concentrates on identifying distinguishing characteristics of the feature design in pattern recognition domain. Based on the characteristics, we attempt to design feature selection algorithm which is best applicable to the pattern recognition problems.

Kudo attempted to divide the feature set size into three categories, small to be 0~19, medium to be 20-49, and large to be over 50 [4]. However this categorization is arbitrary and so the feature design benefits little from it. In this paper, we propose a *task-oriented* categorization which reveals the size characteristics of different types of problems. The CR (character recognition) domain is distinguishable from general PR (pattern recognition) and IR (information retrieval) problems in *size* aspect of the feature sets. Usually PR problems use tensd IR (inform. For ra dists, the dina UsuaWDBC (Wisconsin breast cancer) and IPUMS (census dina from Los Angeles and Long Beach areas) in UCI repository have 30 and 61 features, respectively [5]. In IR applications, different words appearing in text documents are used as features. So feature set size is usually very huge, e.g., hundreds of thousands. The CR lies in between PR and IR. It is usual that CR uses hundreds of features [1].

This paper adopts the *edge-based* ACO (ant colony optimization) algorithm, in which pheromone trail is put on the graph edges, for feature selection of CR problems. For a general principle of ACO, we refer the readers to [6]. Since CR uses hundreds of features, the number of edges is tens or hundreds of thousands. So the edge pheromone is manageable. The edge-based version has the advantage of naturally considering the correlations between features [7]. (In IR applications, since the number of features is hundreds of thousands, node-based pheromone were used [8].)

This paper also presents a scheme, called *selective evaluation*, to improve the convergence and recognition performance of ACO. The scheme reduces computational load by excluding unnecessary or less promising candidate solutions. Note that ACO can adopt the scheme because it keeps the valuable information, pheromone trail which helps identify those solutions.

The proposed ACO is compared with two kinds of popular algorithms, greedy algorithm and GA (genetic algorithm) in terms of computational requirements and recognition performance. Both analytic and empirical comparisons have been performed. The results obtained using handwritten numerals showed that the ACO is the most promising method both in computational load and recognition performance. The selective evaluation produced the improved performance. We also explain briefly why the selective evaluation is better than other schemes.

Section II describes conventional feature selection algorithms. Section III describes ACO and the selective evaluation scheme. Section IV presents timing analysis of the algorithms. In Section V,

experimental results and discussions are described. Section VI concludes the paper.

II. Conventional Algorithms

The feature selection problem involves the selection of a subset of d features from a total of D features, based on a given optimization criterion. Let us denote the D features uniquely by distinct numbers from *1* to D, so that the total set of D features can be written as U={1,2,…,D}. X denotes the subset of selected features and Y denotes the set of remaining features. So U=$X \cup Y$ at anytime. $J(X)$ denotes a function evaluating the performance of X. $J()$ may evaluate either the accuracy of a specific classifier on a specific dataset (e.g. the wrapper approach as in [9]) or a generic statistical measurement (e.g. the filter approach). The choice of evaluation function, $J()$, depends on the particular application. In this paper, recognition accuracy of MLP (multi-layer perceptron) was used for the values of $J()$.

The feature selection algorithms can be divided into greedy approach and meta-heuristic approach. Section 1 describes two greedy algorithms, SFS and PTA. A variety of meta-heuristics with effective searching capability are available [10]. Of these, two population-based approaches, genetic algorithm (GA) and ant colony optimization (ACO) are the most popular. Section 2 describes GA. ACO will be presented separately in Section III.

1. Greedy Algorithms

The traditional algorithms such as SFS (sequential forward search), PTA (plus-and take-away), and SFFS (sequential forward floating search) belong to the greedy approach. They search the feature subset by sequentially adding the most significant features and/or deleting the least significant features. They are limited in finding a near-optimal solution, due to their inherent tendency to become trapped at local optimum points [3].

We introduce the basic operations used by greedy algorithms. For the notations, see the first paragraph of Section II. The size of the set, S, is denoted by |S|. Using the *add* and *rem* operations, the SFS and PTA algorithms can be described as follow.

rem: Choose the least significant feature x in X such that x=argmax$_{a \in X} J(X-\{a\})$, and move x to Y.

add: Choose the most significant feature y in Y such that y=argmax$_{a \in Y} J(X \cup \{a\})$, and move y to X.

Algorithm SFS:

1. $X=\Phi$; $Y=\{i|1 \le i \le D\}$;
2. **repeat** *add* **until** $|X|=d$;

Algorithm PTA(l,r) with $l>r$:

1. $X=\Phi$; $Y=\{i|1 \le i \le D\}$;
2. **repeat** { **repeat** *add* l times; **repeat** *rem* r times;} **until** $|X|=d$;

2. Genetic Algorithm

The GA is being actively researched as promising methods for solving combinatorial problems with an exponential search space [11]. The outline of GA is described by Algorithm GA. GA simulates the biological evolution process such as natural selection and genetic operations of crossover and mutation.

In the initialization stage, random bit generator is used to set the genes of chromosomes in P. In the selection stage of line 4, rank-based roulette-wheel method is used, where i-th chromosome in the sorted list is given the probability of selection $q(1-q)^{i-1}$. For

the genetic operations in line 5, standard multi-point crossover and standard bit-flipping mutation operators are used.

Algorithm GA:

1. Initialize population P;
2. *Evaluate* chromosomes in P;
3. **repeat** {
4. Select two parents;
5. Perform crossover and mutation to get offspring;
6. *Evaluate* the offspring;
7. Replace a chromosome in P with offspring;
8. } **until** (stopping condition);

III. Conventional Algorithms

Section 1 describes conventional ACO. In Section 2, we propose a novel idea of embedding the selective evaluation scheme in ACO.

1. Ant Colony Optimization

The ACO uses a graph to represent the state of solution space, where a node corresponds to a feature [7]. [Figure 1] shows *on-edge* version of pheromone representation. The τ_{ij}, $1 \le i, j \le D$, stores the pheromonal trail deposited between features x_i and x_j.

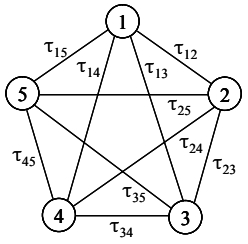

Figure 1. Graph storing pheromone trails when *D=5*

We use Min-Max version of ACO, where τ_{min} and τ_{max} are given as user-defined parameters. The τ_{ij} is controlled to be always within the range $[\tau_{min}, \tau_{max}]$. In the initialization stage, all the edges are set to have maximum value, τ_{max}. In line3, i-th ant walks on the graph and generates a candidate solution (feature subset) X_i. Initially X_i is empty. The ant randomly chooses a node (feature) and adds it to X_i. Then the ant successively chooses next node $x_k \in Y$ with the probability $p(x_k, X_i)$ of Equation (1) and adds it to X_i. In Algorithm ACO, N is the number of ants.

$$p(x_k, X_i) = \frac{\tau(x_k, X_i)^{\alpha} \eta(x_k, X_i)^{\beta}}{\sum_{x_j \in Y} \tau(x_j, X_i)^{\alpha} \eta(x_j, X_i)^{\beta}} \quad (1)$$

We use the elitism in which only the best, called elite and denoted in (2) by X_q, among the candidate solutions is given the chance of updating pheromone in line 4. Equation (2) updates the pheromone trail for the edge connecting node x_i and x_j. The ρ is the pheromone evaporation parameter. By the min() and max() functions, pheromone trail is kept within $[\tau_{min}, \tau_{max}]$. In Equation (3), S_q is the value of $J(X_q)$ and S_{best} is the value of the best solution found so far from the first generation.

$$\tau_{ij} = \min(\max((1-\rho)\tau_{ij} + \delta(x_i, x_j, X_q), \tau_{\min}), \tau_{\max}) \quad (2)$$

$$\delta(x_i, x_j, X_q) = \begin{cases} \dfrac{1}{1 + S_{best} - S_q}, & \text{if } x_i \in X_q \text{ and } x_j \in X_q \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Algorithm ACO:

1. Initialize pheromone trail on every edge, $\tau_{ij} = \tau_{max}$, $1 \le i,j \le D$;
2. **repeat** {
3. **for** (i=1 **to** N) Construct solution (subset) X_i and *evaluate* it;
4. **for** (every edge) Update pheromone trail;
5. }**until** (stopping-condition);

2. Ant Colony Optimization with Selective Evaluation

The aim of this section is to devise an improved version of ACO. Let us analyze the computational time of Algorithm ACO in Section 3. The dominant operation is the subset evaluation in line 3. The other operations are negligible. One generation of the loop performs N subset evaluations where N is the number of ants. In this situation, if we can reduce the number of subset evaluations in a reasonable way, convergence speed can be improved.

Here we propose the *selective evaluation* scheme. It actually evaluates only the subsets which are likely to be the elite. In ACO, the pheromone provides valuable information which enables implementation of the selective evaluation. We 'pre-evaluate' the candidate solutions and select the top-ranked solution for the actual evaluation. The pre-evaluation is simply accomplished by adding pheromone values on the edges connecting adjacent nodes on a candidate solution. For example, when a solution is x_2-x_4-x_1, score obtained by the pre-evaluation is $\tau_{24}+\tau_{41}$.

In early generations, the pheromone is unreliable. As generations proceed, the pheromone becomes more reliable. So starting with a high ratio of actual evaluation, we reduce the ratio gradually as time goes. Equation (4) decides the ratio for the generation t. The parameter r is a decreasing factor. The parameter *lower_bound* keeps the ratio to be above its value.

$$q(t) = \max(r^t, lower\_bound) \quad (4)$$

Algorithm ACO with selective evaluation:

1. Initialize pheromone trail on every edge;
2. t=0; // generation
3. **repeat** {
4. **for** (i=1 **to** N) Construct subset X_i;
5. "Pre-evaluate" X_i, $1 \le i \le N$ using pheromone information;
6. Sort X_i, $1 \le i \le N$;
7. **for** (i=1 **to** N) **if** (X_i's rank $\le N^* q(t)$) *Evaluate* X_i;
8. **for** (every edge) Update pheromone trail;
9. t++;
10. } **until** (stopping-condition);

IV. Timing Analysis

If we have a time unit, it would be very helpful to analyze the timing requirements. Kudo and Sklansky used the number of subset evaluations and big-O notation [4]. However it is not very helpful, because the evaluations of subsets with different sizes consume significantly different amounts of computation time. Oh proposed a rigorous method that uses *atomic* operation requiring a nearly fixed amount of CPU cycles [12]. We will adopt this concept to compare the timing requirements of greedy algorithms, GA, and ACO.

Let $t(s)$ be the computation time required to evaluate a feature subset with size s. The value of $t(s)$ depends not only on s, but also on the size of the training sample set, when we use a classifier for the subset evaluation. Since the size of the sample sets is fixed in advance for a given feature selection job, it can be regarded as constant. Therefore, $t(s)$ depends only on the value of s.

We define the linearity property of classification algorithms and then the atomic operation.

Definition 1: When a classifier satisfies both in learning and recognition stages that $t(s)$ is approximately equal to $s^* t(1)$, we say that the classifier is approximately linear.

We can easily prove that the popular classification algorithms such as MLP, SVM, and k-NN are approximately linear. The linearity assumption holds for k-NN classifiers, since the dominant operation is the distance calculation that is linear to the number of features. This assumption also holds for a neural network classifier, MLP, since the amount of computation by the forward classification and backward learning processes is proportional to the number of input nodes.

Definition 2: The evaluation of a single feature is called an *atomic operation*. The time required for the atomic operation is $t(1)$ and it is referred to as the *atomic time*. We abbreviate the atomic time as a.

Note that given a specific character recognition problem with training and test sets, the atomic time is fixed to be a constant. So we can use the atomic time a as a unit in measuring the timing cost of a feature selection algorithm.

Now, we analyze the computation time of feature selection algorithms. Let us first analyze the greedy algorithms. Equation (5) shows the timing requirement for the *rem* and *add* operations. In this notation, the current size of X is s.

$$\left.\begin{aligned} T(rem) &= t(s-1)\cdot s = \alpha(s^2 - s) \\ T(add) &= t(s+1)\cdot(D-s) = \alpha(-s^2 + (D-1)s + D) \end{aligned}\right\} \quad (5)$$

As shown by Algorithm SFS in Section 1, the SFS repeats the *add* operation d times when it selects d features from D ones. So the time required by SFS can be formulated as Equation(6).

$$\sum_{s=1}^{d} \alpha(Ds - s^2 - s + D) \cong \alpha\left(\frac{Dd^2}{2} + \frac{3Dd}{2} - \frac{d^3}{3} - d^2\right) \quad (6)$$

The computation time of GA and ACO can be more easily analyzed. The time of ACO is adT_{ga} since each generation evaluates one subset whose size is d. T_{ga} is the number of total generations. The ACO requires $adNT_{aco}$ where N is the number of ants and T_{aco} is the number of total generations. [Table 1] summarizes the analysis. It is worth noting that the time for GA and ACO is independent on the original feature set size D. The greedy algorithms are influenced by the value of D, and the time becomes prohibitive for large D.

Note that we have no control on the timing requirement of SFS in (6). On the contrary, we can control the time of GA and ACO by changing the number of generations.

Table 1. Timing analysis of SFS, GA, and ACO

Algorithms	Time (Dependent on D?)	User controllable?	Actual timing example*
SFS	α ($Dd^2/2+3Dd/2-d^3/3-d^2$) (yes)	no	53.62 hours for d=64 167.73 hours for d=128
GA	$\alpha\, dT_{ga}$ (no)	yes	Given 72 hours, T_{ga}9579 for d=64 and 4799 for d=128.
ACO	$\alpha\, dNT_{aco}$ (no)	yes	Given 72 hours, T_{aco}320 for d=64 and 160 for d=128.

* When D=256 and α =0.422 seconds (measured for the experiments in Section IV)

V. Experiments

1. Experimental Setup and Results

The experiments have been done using CENPARMI handwritten numeral database. It has 4,000 training and 2,000 test samples [13]. For the original feature set, we used DDD which has 256 features [14]. So in the given problem, D is 256. For the subset

evaluation, MLP was used. Its architecture was (1+*d*)-20-10 where three values represents the number of input, hidden, and output nodes, respectively. It was trained using error back-propagation algorithm and tested using the test set. The test accuracy was used as the evaluation result, i.e., the value of *J()*. The accuracy was calculated as a ratio of the number of correctly recognized samples to the total number of samples. No rejection was allowed.

Under these experimental conditions, the atomic time α was 0.422 second. (We used a Pentium processor with 3.4GHz and 2M cache memory. The main memory was 4G bytes.) The last column of [Table 1] shows the actual timing requirements. The SFS requires 168 hours to select 128 features. It is too time consuming to be used practically. (In the industrial applications, much bigger training set is usually used, eg, 100,000 samples.) The GA and ACO have the advantage of controllability of timing. Assuming 3 days of the computation time is given, we set the stopping-condition (line 8 of Algorithm GA and line 5 of Algorithm ACO) to quit the loop when the timer reaches 72 hours. The last column of [Table 1] shows the maximum generation when 3 days are given.

The GA and ACO used the following parameters. For their *stopping-condition*, 72 hours were given.

- GA: population size=30, mutation rate=0.1, q for rank-based selection=0.25
- ACO: population size=30, α=1, β=0 (ignoring η component in Equation (1)), ρ (evaporation rate)=0.06, $[\tau_{min},\tau_{max}]$=[0.2, 20.0]
- ACO with the selective evaluation: r=0.98, *lower-bound*=0.3

[Table 2] shows the recognition accuracies obtained by SFS, GA, and ACO. Note that SFS used 54 and 168 hours for d=64 and 128, respectively. The GA and ACO obtained the performances using 72 hours for both d=64 and 128. For an objective comparison, we ran GA and ACO 6 times independently, and their average, minimum, and maximum were recorded in [Table 2].

[Table 2] illustrates that ACO produced better solutions than GA. It also shows that the selective evaluation scheme improved the ACO in a significant amount. The scheme increased the average accuracies by 0.18% and 0.23% for d=64 and 128, respectively. These amounted to the error reduction ratio, 6.55% and 10.45%.

The last row is the accuracy when we used the original feature set. It was 97.30%. When we selected half the original set, the accuracy was improved to 98.03% (error reduction ratio 27.04%). So we can say that in this problem instance, we obtained both the compact classifier (about twice faster than the original classifier) and the better accuracy. The CENPARMI training set size is 4000, which is believed to be small, considering the large variations of handwriting style. (The database was collected from real-world mail pieces.)

Table 2. Comparison of accuracies of SFS, GA, and ACO (Subset evaluation by MLP classifier, Unit=%)

Subset size (d)	SFS	GA	ACO	
			Without selective evaluation	With selective evaluation
64 (0.25D)	97.25	97.12 (96.75, 97.30)	97.25 (97.10, 97.45)	97.43 (97.35, 97.50)
128 (0.5D)	97.50	97.66 (97.50, 97.80)	97.80 (97.70, 97.85)	98.03 (98.00, 98.05)
256 (D)	97.30			

* For GA and ACO, average (minimum, maximum) were recorded for 6 independent runs.

2. Discussions

It is difficult to explain rigorously why the selective evaluation is superior. Here we will explain intuitively only, not in a rigorous manner. Our statistical observation showed that the selected ants were really the elite (i.e., the best ant), but sometimes they were not. However, it was more probable to be elite. So the selective evaluation scheme still works to evolve towards better solutions. Since the selective evaluation can execute more generations, it is probable to find better solutions than conventional ACO. Additionally the cases in which the selected ant is not elite perturb the current graph information, so it might be advantageous in escaping from the premature convergence.

VI. Conclusions

This paper proposed a feature selection algorithm for the pattern recognition. The ACO with the newly proposed selective evaluation scheme was the best among greedy algorithm, GA, and ACO. It consumed a reasonable computation time. One of its practical advantages is that its timing is controllable. In other words, we can say that, if the algorithm is given more computation time, they might find out better solutions than the ones in Table 2. In future work, we will analyze and explain in a more rigorous manner why the selective evaluation is superior.

참 고 문 헌

[1] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition-a survey," Pattern Recognition, Vol.29, No.4, pp.641-662, 1996.

[2] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed., Academic Press, 2006.

[3] J. Kittler, "Feature selection and extraction," in *Handbook of Pattern Recognition and Image Processing*, Academic Press (Edited by T.Y. Young and K.S. Fu), pp.59-83, 1986.

[4] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern recognition," Pattern Recognition, Vol.33, No.1, pp.25-41, 2000.

[5] P. M. Murphy and D. W. Aha, UCI repository for machine learning databases, (http://www.ics. uci.edu/~mlearn/databases/), 1994.

[6] Marco Dorigo and Christian Blum, "Ant colony optimization: a survey," Theoretical Computer Science, Vol.344, pp.243-278, 2005.

[7] Christine Solnon and Derek Bridge, "An ant colony optimization meta-heuristic for subset selection problems," in *System Engineering using Particle Swarm Optimization* (Edited by Nadia Nedjah and Luiza Mourelle), Nova Science publisher, pp.7-29, 2006.

[8] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri, "Text feature selection using ant colony optimization," Expert Systems with Applications, Vol.36, pp.6843-6853, 2009.

[9] P. Langley, "Selection of relevant features in machine learning," Proc. of AAAI Fall Symposium on Relevance, pp.1-5, 1994.

[10] Christian Blum and Andrea Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," ACM Computing Surveys, Vol.35, No.3, pp.268-308, 2003.

[11] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Boston, 1989.

[12] I. S. Oh, J. S. Lee, and B. R. Moon, “Hybrid genetic algorithms for feature selection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.26, No.11, pp.1424-1437, 2004.

[13] C.-L. Liu, “Handwritten digit recognition: benchmarking of state-of-the-art techniques,” Pattern Recognition, Vol.36, No.10, pp.2271-2285, 2003.

[14] I. S. Oh and C. Y. Suen, “Distance features for neural network-based recognition of handwritten characters,” International Journal on Document Analysis and Recognition, Vol.1, pp.73-88, 1998.

저 자 소 개

오 일 석(Il-Seok Oh) 정회원

- 1984년 : 서울대학교 컴퓨터공학과(공학사)
- 1992년 : KAIST 전산학과 박사
- 1992년 9월 ~ 현재 : 전북대학교 컴퓨터공학부 교수

<관심분야> : 컴퓨터비전, 패턴인식

이 진 선(Jin-Seon Lee) 정회원

- 1985년 : 전북대학교 전산통계학과(이학사)
- 1995년 : 전북대학교 전자계산기공학과 박사
- 1995년 3월 ~ 현재 : 우석대학교 게임콘텐츠학과 교수

<관심분야> : 멀티미디어, 패턴인식