Learning to Describe Collective Search Behavior of Evolutionary Algorithms in Solution Space

Lei Liu, Chengshan Pang, Weiming Liu, and Bin Li(✉)

School of Information Science and Technology,
University of Science and Technology of China, Hefei, Anhui, China
binli@ustc.edu.cn

Abstract. Evolutionary algorithms (EAs) are a kind of population-based meta-heuristic optimization methods, which have proven to have superiorities in solving NP-complete and NP-hard optimization problems. But until now, there is lacking in the researches of effective representation method to describe the collective search behavior of the Evolutionary Algorithm, while it is useful for researchers and engineers to understand and compare different EAs better. In the past, most of the theoretical researches cannot directly guide for practical applications. To bridge the gap between theoretical research and practice, we present a generic and reusable framework for learning features to describe collective behavior of EAs in this paper. Firstly, we represent the collective behavior of EAs with a parent-child difference of population distribution encoded by self-organizing map (SOM). Then, we train a Convolutional Neural Network (CNN) to learn problem-invariant features from the samples of EAs' collective behavior. Lastly, experiment results demonstrate that our framework can effectively learn discriminative features representing collective behavior of EAs. In the behavioral feature space stretched by the obtained features, the collective behavior samples of various EAs on various testing problems exhibit obvious aggregations that highly correlated with EAs but very weakly related to testing problems. We believe that the learned features are meaningful in analyzing EAs, i.e. it can be used to measure the similarity of EAs according to their inner behavior in solution space, and further guide in selecting an appropriate combination of sub-algorithm of a hybrid algorithm according to the diversity of candidate sub-algorithm instead of blind.

Keywords: Evolutionary algorithms · Collective behavior · Representation learning · Convolutional neural network · Self-organizing map

1 Introduction

In artificial intelligence, EAs are a kind of stochastic optimization methods. Evolutionary Algorithm (EA) starts with an initial population of solutions, then gradually update them by multiple iterations with the guidance of fitness to obtain optimal solutions. The convergence of EA is usually achieved through iterations, and a set of high dimensional population data is produced in each iteration. So, in the whole

Y. Shi et al. (Eds.): SEAL 2017, LNCS 10593, pp. 196–207, 2017.
https://doi.org/10.1007/978-3-319-68759-9_17

process, a large amount of data is generated. While direct observation and analysis of these data are difficult. Most of the researches mainly focus on the performance of EAs, relatively little attention is paid on the search behavior of EAs. As a result, although EAs have been successfully applied in many domains, there is often full of confusions for users to understand them, let alone to know how the solutions are discovered. Meaningful features can help people understand and compare the behavior of EAs in high-dimensional space, and are useful for people to manipulate EAs in a more reasonable way.

In recent years, a few researchers have dedicated to empirically analyze the collective behavior of EA. In [3], Turkey et al. presented a model that extracts and quantifies features to capture the collective behavior. The proposed model focuses on studying the topological distribution of an algorithm's activity by SOM [1]. It defines two properties called exploitation and exploration behavior, which is an important contribution to investigate the collective search behavior of population-based search algorithms. In [4], the authors defined an indicator of the exploitation behavior of an EA. The above methods are based on the characteristics of manual definition rather than automatic feature learning. In [7], Pang et al. tried to extract discriminative features from the generation-wise collective behavior data of several EAs on fitness landscapes via Slow Feature Analysis (SFA) [14, 15], with purpose to find out whether there exist differences between the searching behavior of different EAs running on the same or different fitness landscapes. It has been shown that the features of one-generation offspring exhibit aggregative and distinguishable nature via unsupervised learning. However, the SFA-based method still has shortcomings: Firstly, when extracting features of a new EA with SFA, all the previous training data must be included, which is computationally expensive. Secondly, it is somehow unnatural to fix the artificially generated parent population and treat its one generation offspring as the samples of EAs' behavior. What we want is a tool that can extract discriminative features of EAs' behavior from the data collected from real runs of the EAs on problems, and after our nets have been well trained, the tool can be used as an off-the-shelf feature extractor that can do feature extraction for any new EAs. To meet this end, a generic and reusable framework is designed and presented in this paper to extract discriminative collective behavior features from data generated by EAs in a real run.

In order to obtain a more natural and reasonable description of EAs' collective behavior, deep representation learning techniques are considered in our research, which has proven to be able to learn features from real running data in many applications [22]. To handle the difficulty of representing population distribution universally, SOM is adopted to map high-dimensional solution space to a 2-dimensional grid, in which the neighborhood relationship between samples (population distributions) is maintained. To overcome the limitation of one-generation offspring, a more general representation of EAs' behavior is designed, that is the variation of arbitrary successive 2 generations represented in SOM-based probability model. The new definition of collective behavior describes better the change in distribution between child-parent populations. The whole framework of the proposed method is illustrated in Fig. 1. Firstly, a number of representative evolutionary algorithms [5], classical evolutionary programming (CEP) [6], differential evolution (DE) [8], evolution strategy (ES) [9] and genetic algorithm (GA) [10], are applied to a number of testing problems of different characteristics, e.g.

single/multiple model, with/without dependence between variables, etc., to get collective data. Then, SOM is adopted to transform the collective data into a uniform representation of population distribution. After that, two data sets, training and testing datasets [19], are prepared for training a Convolutional Neural Network (CNN) [2] as a feature extractor. For CNN training, the task is defined as training a classifier to discriminate various EAs. With this setting, it is possible to learn collective behavior features that are only related to EAs themselves, invariant to various problems under solving.

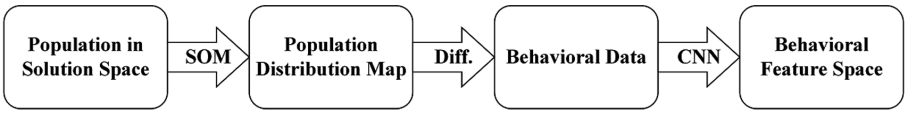


Fig. 1. The schematic of our formulation.

Specifically, in order to promote deeper study in this field, we open our dataset and source code. Our results can be reproduced with the source code and trained models available at https://github.com/liulei13/EAs_analysis.git.

In summary, the main contributions of this paper are listed below:

- A new representation of the collective behavior of EAs based on the difference of SOM-based representation of parent-child population distribution is presented.
- A generic and reusable framework is proposed to learn discriminative features of EAs' collective behavior, empirical studies show that the extracted features are stable with respect to the algorithms.

2 Uniform Representation of Population Distribution of EAs and Definition of Collective Behavior of EA

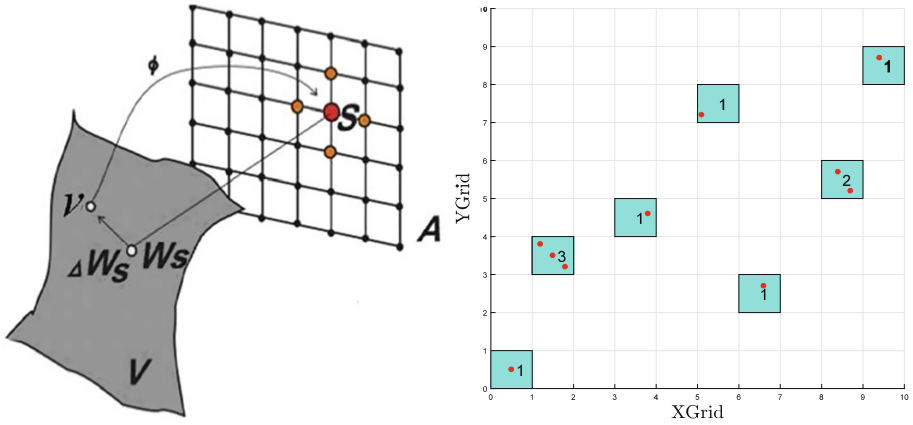
A universal representation of EAs' collective behavior is an important component for preparing samples for feature learning. Since it is expected that the samples should be taken from real runs of EAs on the testing problems of various characteristics, the representation should be valid to represent the change of population distributions. In this paper, SOM [12, 13] is adopted to build a normalized representation of population distribution. Then, the difference of SOM representations between two successive generations is calculated as the representation of EAs' collective behavior.

The schematic of our formulation is shown in Fig. 1. Firstly, we transfer population in solution space to collective distribution by SOM which can preserve the topological structure of the population in solution space. Then we compute behavioral data in solution space by our definition of the behavior of EAs. Finally, we utilize a trained feature extractor to construct behavioral feature space for different problems. In each feature space, we can easily identify different EAs that only need the previous generations of population data (one generation or ten generations in succession).

2.1 Representing Population Distribution of EAs by SOM

A straightforward way to represent the EAs' population is to chain the individuals sequentially. But such representation suffers from a drawback that it may change when the individuals change their positions in the chain. Since the distribution of the population is what we need really in feature learning, a normalized representation that describes the distribution and will not change with respect to the individual's position in the population. SOM not only can maintain intrinsic neighbor relations but also have strong learning ability and generalization ability, therefore, SOM is adopted in this paper as a normalized representation of population distribution.

SOM [12] is a neural network based on competitive learning, with the ability to map continuous and discrete high-dimensional space V to a two-dimensional or three-dimensional discrete space A. As illustrated in Fig. 2(a), this mapping is a preservative mapping that maintains the original spatial neighborhood.



(a) (b)

Fig. 2. (a) is a description of the high-dimensional space ***V*** to low-dimensional space ***A*** mapping; (b) is a normalized representation that describes the distribution and will not change with respect to the individual's position in the population.

In Fig. 2(b), we make a more intuitive description of the implementation process and results of SOM. After being trained, a population in the initial 30-dimensional solution space is mapped into a two-dimensional grid, where the hitting times of each node are recorded and used to represent the value of the node. For example, this is a 10×10 SOM grid as showed in Fig. 2(b), and 10 individuals are mapped into the two-dimensional grid. In our work, we adopted 100×100 SOM grid to represent a population of EAs and there are 100 individuals at any generation. Certainly, other sizes of the SOM grid, such as 10×10, 50×50 and 200×200, are also achievable. It should be noted that the setting of the size of SOM grid is a tradeoff between performance and compute cost.

2.2 Definition of Collective Behavior of EA

In [7], one-generation offspring of EAs from the same initial parental population is regarded as collective search behavior of EAs. It is reasonable because the offspring was obtained by different EAs from the same initial parental population, therefore the offspring contains most differences between EAs. However, this representation is somehow unnatural because, in real applications, it is hard to require all compared algorithms to start from the same parent population all the time. To make the analysis method more appropriate and universal in a real run, a new definition of collective behavior is designed, i.e. the difference of population distributions represented in SOM map in two successive generations of an EA. The formula of the definition for calculation is as follows:

$$\text{Behavior}(g)^{(k,t)} = \left\| SOMmap(g+1)^{(k,t)} - SOMmap(g)^{(k,t)} \right\| \tag{1}$$

Where $\|\cdot\|$ is a norm of the SOM map in $\mathbb{R}^{p\times p}$, g is an ordinal number of generation, k is the id of EA and t means an ordinal number of run times

The new definition describes the variation of the population distribution of EAs, and we assume that it satisfies independent and identically distributed (i.i.d.). We believe that the same algorithm will exhibit the same behavior in any generation on the same landscape, thus, there is a steady and learnable pattern for an EA. So, the definition of collective search behavior of EAs is very suitable as a training data to train a CNN model. We compared the L1 norm and the L2 norm to compute collective search behavior data, and find that there is no difference in the results. In our dataset, we use the L1 norm to define collective search behavior data. An illustration of our new definition of EAs' behavior is shown in Fig. 3.

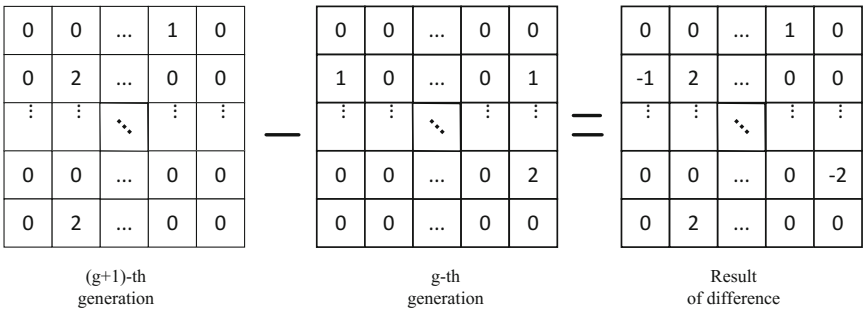


Fig. 3. An illustration of our new definition of EAs' behavior

3 Learning Discriminative Features of Collective Behavior by CNN

As is known to all, deep convolutional neural networks have recently achieved many successes in visual recognition tasks including image classification [2], object detection [18] and scene parsing [21]. If the training data is sufficient, CNN is capable of learning

two-dimensional spatial information. Meanwhile, CNN has the ability to map the high-dimensional nonlinear space into a low-dimensional linear space [20], which is useful to analyze and visualize collective search behavior data of EAs. The method trains an embedder that can transform instances into a feature space.

3.1 Training a CNN as Feature Extractor

It is vital to design a reasonable structure of CNN. We explore different network structures and parameter settings to achieve tradeoffs between performance and speed. Our designed CNN architecture as shown in Fig. 4(b). The flowchart of behavior-to-feature mapping and visualization is shown in Fig. 4(a). Overall, the structure of our CNN contains 2 convolutional layers and 3 fully connected layers.

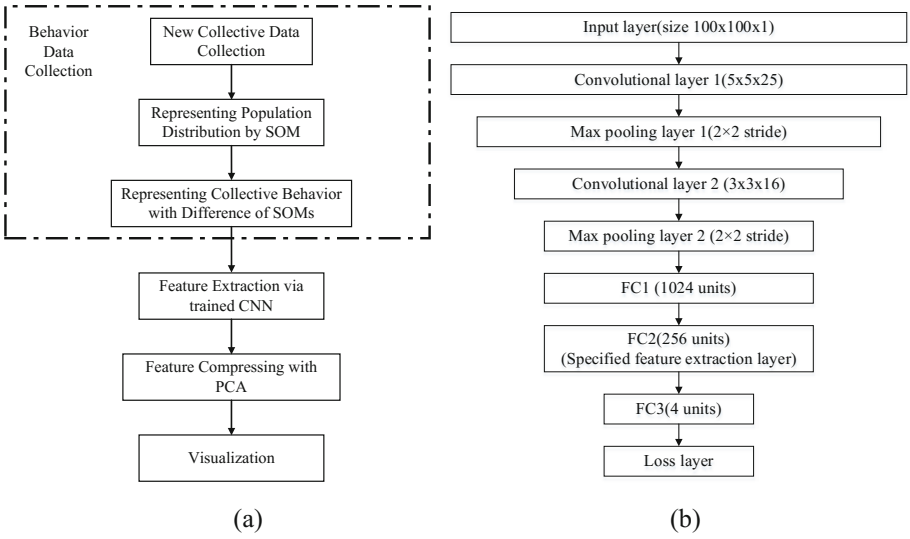


Fig. 4. (a) is the flowchart of behavior-to-feature mapping and visualization; (b) is the architecture of the proposed CNN.

In the process of training a CNN, the input layer is input with a $100 \times 100 \times 1$ behavior data. Convolutional layer means the output of the previous layer will be convoluted with filters, where the filter is often 5×5 or 3×3. ReLU is the abbreviation of Rectified Linear Units [16] and follow the convolutional layer, which is an activation function defined as $f(x) = max(0, x)$. Max pooling layer is a form of non-linear down-sampling, which partitions the input data into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. In a fully connected layer, neurons have full connections to all activations in the previous layer, as seen in regular Neural Networks. The loss layer specifies how the network training penalizes the deviation between the predicted and true labels and is normally the last layer in the

network. Finally, we use Stochastic Gradient Descent (SGD) algorithms to update parameters of convolutional layers and fully connected layers.

In our model, the loss function is the cross-entropy, which can be calculated by:

$$L = -\sum_{i=1}^{S} l_i \log(O_i = 1|W) - \sum_{i=1}^{S} (1 - l_i) \log(O_i = 0|W) \quad (2)$$

where l_i is the label of EAs, We denotes the collection of all network parameters in our model. S denotes the total number of EAs as the training data of our model.

3.2 Behavior-to-Feature Mapping and Visualization

The purpose of this work is not to get a classifier with high accuracy, but to obtain a feature extractor for characterising different EAs in solution space. Therefore, the behavior-to-feature mapping is trained via CNN. As shown in Fig. 4(b), we feed testing behavior data into trained CNN and a 256-dimensional feature vector is obtained at the fc2 layer. In order to better demonstrate different behavior data in feature space, the principal component analysis (PCA) [17] is employed to reduce the dimension of the feature space. We plot the principal 2 or 3 dimensions for visualization, in which the first 2 dimensions occupy about 75% of the energy. PCA is the main linear technique for dimensionality reduction. It is reasonable to use PCA to deal with the extracted features because the output of the fully connected layer of CNN is linear. This has already been proved that CNN allows mapping the high nonlinear space into a low dimensional linear space [20].

4 Experiment Results and Discussion

In this work, we choose four classical EAs (i.e. GA, ES, CEP, DE) which are well-known optimization algorithms for investigating the ability and legality of the method presented. More details about these algorithms can be referred to paper [7]. Meanwhile, we adopt 6 classical benchmark functions to provide various fitness landscapes for investigation, they are: Ackley function [11], the Elliptic function [11], Rastrigin's function [11], Rosenbrock's function [11], Schwefel's problem 1.2 [11], Sphere function [11], which represent different complex problems. In this paper, the dimensionality of all testing functions are set to 30 and the global optimum of each testing function is the minimum value in the solution space for all experiments. More details about the test functions can be referred to paper [7]. Specifically, the main parameters in experiments are given in Table 1.

The sampling process can be merged into the execution of EAs, which makes the method presented in this paper more reasonable and practical. The EAs under analysis were performed on the selected representative testing problems, the successive populations generated at each iteration were adopted as the population samples. To make the experimental research be tractable and comparable, the range of value for each dimension was set from −32 to 32. It is reasonable because the search behavior of algorithms is believed to be independent of the size of the range. In order to obtain the statistical robust features from the random process of EAs' execution, each algorithm

Table 1. The main parameters in experiments.

Parameter	Value
Population size	100
The dimensionality of individual	30
The size of SOM	100×100
The number of EAs	4
The number of test functions	6

was performed 10000 times on each test function. In the article, the proposed method named as the CNN-based method, and will compare it with the SFA-based method [7].

We designed two experiments to verify our idea: (1) Verifying the validity of our behavioral definition on different testing functions with the SFA-based method; (2) Comparing the CNN-based method with the SFA-based method in their behavioral feature space. (3) Analysing multi-generation offspring with the CNN-based method.

In this experiment, we re-implement SFA-based method [7] with two behavioral definitions, which are one-generation offspring defined by the author in the paper [7] and our new behavioral definition. The initial population is randomly sampled in the area $[-32, 32]^{30}$. This is different from the author's area in $[5, 10]^{30}$, as this area does not contain the optimal solution 0, which may easily lead to behavior bias on the behavior of algorithms. The one-generation offspring generated by four algorithms with the same initial population on the same benchmark function are collected as the sample data of collective behavior of EAs, our behavioral data compute the difference between the distribution of the current data and the distribution of the initial population, as illustrated in Fig. 3. There are 2000(offspring populations)$\times$ 4(EAs) = 8000 samples in the input sequence, they is mapped into the 2-D feature space that are extracted via the process presented in Fig. 4(a). Another 8000 new behavioral samples are the distribution of the previous samples subtracting the distribution of the same initial population.

4.1 Verifying the Validity of Our Behavioral Definition on Different Testing Functions with the SFA-Based Method

From Fig. 5, it can be observed that the sample distribution of four algorithms exhibits clear aggregative and discriminative nature in the feature space constructed by slow features. Beyond that, for all the six benchmark functions, the behavior of four EAs exhibits a sort of stable similarity relationship, that is the behavior of CEP, DE and GA are more similar compared to that of ES. The above conclusions are consistent with the authors of the SFA-based method. It must be emphasized that the two definitions yield exactly the same result, so we only draw a set of experimental results in Fig. 5. Why the behavior of the two definitions is surprisingly consistent. This is not difficult to analyze, for a fixed initial population, in the calculation process is equivalent to the same component. The first step of SFA will remove the same component, that is to say, the actual participation in the calculation of the data is exactly the same, although they come from different behavioral definitions. At the same time, it also verifies that the

author's behavioral definition has limitations, only a special case of our behavioral definition. In contrast, our behavioral definition is more general and reasonable.

4.2 Comparing of the CNN-Based Method and the SFA-Based Method in Their Behavioral Feature Space

In this experiment, we use the collected behavioral data in 4.1 as testing data. We collected sufficient behavioral data by executing the algorithm under the same conditions for training a feature extractor with CNN, the whole process as described in Sect. 3.1. There are 8000(offspring populations) $\times$ 4(EAs)$\times$ 6(benchmark functions) = 192 thousand samples. During the training stage of the proposed model, the label of CNN is the class of algorithm. After training, the proposed model achieves a precision over 97%. Meanwhile, it also has fast convergence rate on the test data. To sum up, the fast convergence rate of the model shows that our definition of collective search behavior of EAs is learnable and effective. High accuracy illustrates that our method is appropriate and efficient for extracting discriminative features.

The test data is mapped into the 2-D feature space as illustrated in Fig. 4(a) and the result is shown in Fig. 6. Comparing Fig. 6 with Fig. 5, it can be observed that the distribution of testing samples of EAs exhibits more clear and consistent discrimination on the different test functions than SFA-based method. Beyond that, we find there are stable relative (similar) relationships of four EAs in two different feature space constructed by the SFA-based method and the CNN-based method respectively. The amazing consistency implies that the behavior of different EAs only determined by the inner mechanism of the EA. Another advantage of our method is less computational

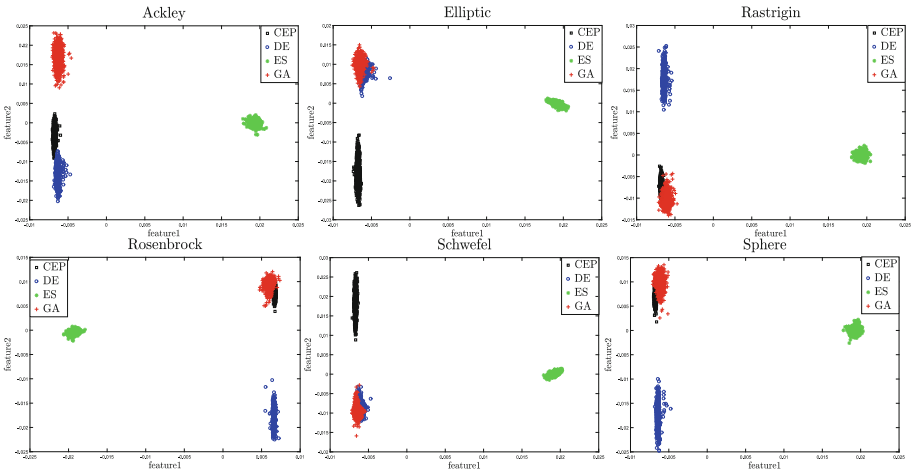


Fig. 5. The distribution of testing samples of 4 EAs on 6 benchmark functions in the behavioral feature space constructed by the SFA-based method.

cost and reusability for new behavior data. The model presented has the capability to

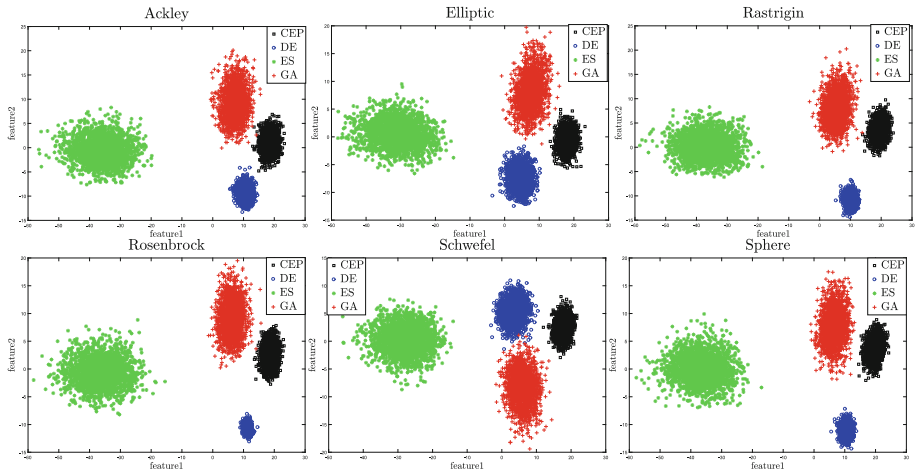


Fig. 6. The distribution of testing samples of 4 EAs on 6 benchmark functions in the behavioral feature space constructed by CNN.

deal with new samples without the need to retrain the system.

4.3 Multi-generation Offspring in Their Behavioral Feature Space Using CNN-Based Method with Our New Definition of Behavior

In this experiment, varying generation number were investigated. Firstly, we collected 1000(run times) × 4(EAs) × 6(benchmark functions) × 5(the number of generations) = 60 thousand samples 80% of which was used as training data and 20% as test data for two specific generation number experiment. Different from the Sect. 4.2, we just fine tuned the training data, then our model achieved precision over 97%, which cost no more than 5 min. It is a big advantage in the calculation of efficiency and reusability without the need for costly re-learning. Therefore, arbitrarily increase in algorithms and test functions is allowed.

In Fig. 7, it can be observed that the distribution of testing samples of EAs also exhibits clearly and steady aggregative and discriminative. Meanwhile, the similarity relationship among 4 EAs is still stable, that is the behavior of CEP, DE and GA are more similar compared to that of ES, which is very weakly related to testing problems. In addition, the experimental results show that our approach allows dealing with multi-generations behavioral data in a real run, which is more close to the nature of the algorithm. Specifically, I have to point out that as the generation number increases, the convergence of the data will become more and more obvious, so we add a scale normalization in the calculation of behavioral data at each generation. The choice of appropriate generation number requires some experience.

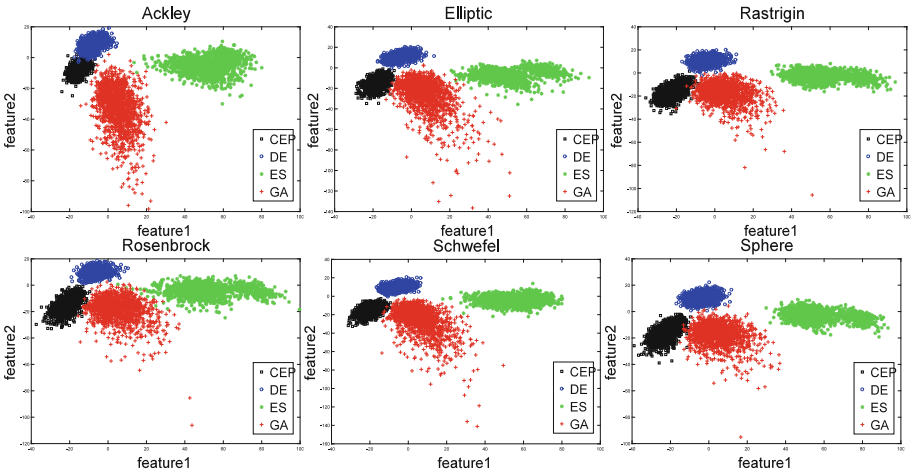


Fig. 7. The distribution of multi-generation offspring samples of 4 EAs on 6 benchmark functions in the behavioral feature space constructed by CNN

5 Conclusions

This paper proposes using a self-organizing map to encode the population distribution of various EAs in the solution space, and a convolutional neural network to extract features which can be used to discriminate the behavior of different algorithms.

Experiment studies are performed with four EAs and six continuous functions, which show that the new definition of EAs' behavior is more accurate and natural, and the CNN-extracted features can more accurately describe the collective search behavior of EAs in the calculation of efficiency, generality, expansibility and reusability than SFA-based method. CNN-derived features are usually difficult to be interpreted, while we can treat it as a black box model in analyzing the behavior of EAs. A very promising application is to calculate the similarity of evolutionary algorithms in the behavioral feature space for algorithm portfolios.

Acknowledgment. The work is supported by the National Natural Science Foundation of China under grand No. 61473271 and No. 61331015.

References

1. Kohonen, T.: The self-organizing map. Neurocomputing **21**, 1–6 (1998)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

3. Turkey, M., Poli, R.: An empirical tool for analysing the collective behaviour of population-based algorithms. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 103–113. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29178-4_11
4. Turkey, M., Poli, R.: A model for analysing the collective dynamic behaviour and characterising the exploitation of population-based algorithms. Evol. Comput. **22**(1), 159–188 (2014)
5. Collins, T.: The application of software visualization technology to evolutionary computation. In: A case study in genetic algorithms. Dissertation, The Open University (1998)
6. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**, 82–102 (1999)
7. Pang, C., Wang, M., Liu, W., Li, B.: Learning features for discriminative behavior analysis of evolutionary algorithms via slow feature analysis. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, pp. 1437–1444. ACM, July 2016
8. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
9. Beyer, H.G., Schwefel, H.P.: Evolution strategies–a comprehensive introduction. Nat. Comput. **1**, 3–52 (2002)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Boston (1989)
11. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark functions for the CEC 2008 special session and competition on large scale global optimization. Nat. Inspired Comput. Appl. Lab. (2009)
12. Duch, W., Naud, A.: Multidimensional scaling and Kohonen's self-organizing maps. In: Proceedings of 2nd Conference on "Eural Networks and Their Applications", Szczyrk, Poland, pp. 138–143 April 1996
13. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
14. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: Unsupervised learning of invariances. Neural Comput. **14**(4), 715–770 (2002)
15. Berkes, P.: Pattern recognition with slow feature analysis. Comput. Neurosci. (2005)
16. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: AISTATS, vol. 15, p. 275 April 2011
17. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdisc. Rev.: Comput. Stat. **2**, 433–459 (2010)
18. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
19. Butail, S., Bollt, E.M., Porfiri, M.: Analysis and classification of collective behavior using generative modeling and nonlinear manifold learning. J. Theor. Biol. **336**, 185–199 (2013)
20. Brahma, P.P., Wu, D., She, Y.: Why deep learning works: a manifold disentanglement perspective. IEEE Trans. Neural Netw. Learn. Syst. **27**(10), 1997–2008 (2016)
21. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1915–1929 (2013)
22. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)