Platform-Based Particle Filter Combined with Improved ABC Algorithm for IoT Big Data Feature Selection: Postprint

Authors: Wu Ying, Li Xiaoling, Tang Jinglei

Date: 2018-08-13T00:00:00+00:00

Abstract

To address the issues of low computational efficiency and poor scalability in existing Internet of Things (IoT) big data feature selection algorithms, we propose a system architecture based on improved Artificial Bee Colony (ABC) for feature selection. This architecture comprises a four-layer system that can efficiently aggregate effective data and eliminate unnecessary data. The entire system is built upon the Hadoop platform, MapReduce, and the improved ABC algorithm. The improved ABC algorithm is employed for feature selection, while MapReduce is supported by a parallel algorithm capable of efficiently processing large datasets. The system is implemented using MapReduce tools and utilizes particle filtering to eliminate noise. The proposed algorithm is compared with similar methods and evaluated for efficiency, accuracy, and throughput using ten different datasets. Results demonstrate that the proposed algorithm exhibits greater scalability and efficiency in feature selection compared to several other recent algorithms.

Full Text

Preamble

Title: Internet of Things Big Data Feature Selection Method Based on Particle Filter and Improved ABC Algorithm on Hadoop Platform

Authors: Wu Ying[1], Li Xiaoling[1], Tang Jinglei[2]
[1] College of Information & Business, Zhongyuan Institute of Technology, Zhengzhou 451191, China
[2] School of Information Engineering, Northwest A&F University, Yangling Shaanxi 712100, China

Abstract: Aiming at the problem that existing Internet of Things (IoT) big data feature selection algorithms suffer from low computational efficiency and poor scalability, this paper proposes a system architecture that selects features using an improved Artificial Bee Colony (ABC) algorithm. The architecture comprises a four-layer system that can efficiently aggregate effective data while eliminating unwanted information. The entire system is built upon the Hadoop platform, MapReduce, and improved ABC algorithms. The improved ABC algorithm is employed for feature selection, while MapReduce is supported by parallel algorithms that can efficiently process massive datasets. The system is implemented using MapReduce tools and utilizes particle filtering to eliminate noise. The proposed algorithm is compared with similar methods and evaluated for efficiency, accuracy, and throughput using ten different datasets. Results demonstrate that the proposed algorithm exhibits greater scalability and efficiency in feature selection compared to several other recent algorithms.

Keywords: Internet of things; big data; artificial bee colony algorithm; feature selection; particle filter; niche technology

0 Introduction

The Internet of Things (IoT) serves as a crucial link connecting the physical and digital worlds. Advances in IoT technology have enabled the digitization of the physical world with high configuration demands for new applications and services, allowing various objects to be grouped together via the Internet to share information. IoT can sense the physical environment, collect data, transmit or disseminate information, process data for appropriate applications, and enable communication among objects, thereby greatly facilitating people' s daily lives.

However, IoT implementation faces significant challenges. As IoT represents a hybrid of heterogeneous objects fundamentally different from traditional networks, its scalability becomes more complex. Additionally, devices communicating within IoT consume substantial memory and bandwidth. Consequently, IoT tends to generate massive volumes of data, known as big data. To address these limitations, green IoT represents an ideal solution, achieving environmental monitoring to reduce emissions and pollution while lowering operational costs and power consumption. In the current big data landscape, database vendors have introduced various standards and platforms for data aggregation and analysis, but these platforms typically offer limited functionality and cannot be widely applied in IoT big data scenarios.

Based on the above analysis, feature selection emerges as one of the core methods for handling big data. Feature selection encompasses image classification, cluster analysis, data mining, pattern recognition, and image retrieval. Feature selection algorithms fall into two main categories: filter methods and wrapper methods. In filter-based techniques, weight values are computed for each feature, enabling the selection of features with better values to represent the

original big dataset. Wrapper techniques utilize subsets of features to generate a set of candidate features, with accuracy subsequently employed to evaluate the feature set results, often achieving better performance than filter methods. Furthermore, metaheuristic algorithms such as Ant Colony Optimization, Particle Swarm Optimization, Bat Algorithm, and Artificial Bee Colony (ABC) have been proposed to enhance computational efficiency. Existing feature selection algorithms suffer from numerous drawbacks, including difficulties in extracting features from real-time continuous data and inefficiency when processing large volumes of data using traditional tools.

This paper proposes a system architecture for aggregating big data, selecting features using an improved ABC algorithm, and forwarding data to the Hadoop platform for parallel processing. The combination of Hadoop and improved ABC can yield optimal results.

---

1 Related Research

Feature selection is a process of selecting feature subsets that can employ search techniques to traverse the solution space. However, this approach appears impractical for identifying massive numbers of features. Consequently, researchers have considered applying swarm intelligence techniques and neural networks to feature selection. Similarly, Hadoop Distributed File System can be utilized for feature selection, as it employs multiple local disks on computer nodes to provide better data locality. In high-performance computing cluster systems, computer nodes connect to a parallel file system called Lustre, which provides an efficient and scalable data storage facility.

The Lustre system is installed on clusters using Lustre as local storage. These local storage systems are suitable for traditional MapReduce functions that can be completed in two steps: read and write operations. Due to the high read/write throughput of Lustre systems, these operations provide high-speed data pathways. The time required for internal transfers in Lustre depends on numerous factors, such as cluster interconnect and data loading, which collectively impact traditional MapReduce functionality when combined.

MapReduce programming can generate massive datasets corresponding to the broad diversity of real-world tasks. MapReduce divides input data into small independent blocks for complete parallel processing. The MapReduce architecture classifies map outputs and sends them to reduce jobs, with task inputs and outputs stored in the file system. The Google File System, inspired by the MapReduce model, utilizes large computer clusters connected via Ethernet switches. Google' s MapReduce solution reduces the cost of widely distributed application clusters. The MapReduce approach simplifies and facilitates the establishment process, operating based on real-time execution without defining pre-planned execution scheduling for nodes. The MapReduce paradigm can execute on distributed nodes, achieving strong fault tolerance and balancing the

load across each cluster, enabling researchers to perform operations more easily and simply. Google' s MapReduce structure was originally a distributed file system for identifying data location and accessibility.

Particle Swarm Optimization techniques can also be applied to feature selection and processing large datasets, reducing system complexity and improving efficiency. Extracting features from large datasets requires substantial time, and employing different noise data filtering algorithms during this process significantly impacts the extracted feature subsets.

Based on the above convolutional methods and traditional Hadoop techniques, systems require selecting optimal features from massive data. To this end, this paper proposes an IoT big data feature selection method based on the Hadoop platform and an improved ABC algorithm. The improved ABC algorithm can effectively select optimal features, while the Hadoop ecosystem provides the necessary infrastructure for processing.

---

2 Proposed Method

2.1 Four-Layer Hierarchical Architecture

The proposed method includes a four-layer architectural model, with each layer providing different functional support to enable efficient read and write operations, as illustrated in [Figure 1: see original paper]. The designed model can facilitate interaction among different objects using shared media. The proposed architectural model can handle various differentiated data generated by applications.

Layer 1: This layer generates and processes data through various objects, then collects and aggregates it. Since data generation involves different numbers of objects, the entire process periodically produces massive heterogeneous data in various formats from different origins. Furthermore, all data have security, privacy, and quality requirements. In sensor data, metadata is always larger than actual measurements. Consequently, early registration and filtering techniques are applied at this layer to filter unnecessary metadata and redundant information.

Layer 2: This layer provides end-to-end connectivity for various devices. Additionally, data generated by different devices converges at this layer and is arranged in appropriate forms.

Layer 3: The feature extraction and processing layer constitutes the main layer of the system architecture, completing the data feature extraction and processing components. Since this work requires both real-time data streams and offline data analysis, third-party real-time tools must be combined with processing servers to provide real-time data processing. Tools such as Storm, Spark, VoltDB, and Hupa can serve as auxiliary components. For instance, MapReduce

can be used to implement data analysis, while the improved ABC algorithm enables better feature extraction from large datasets. At this layer, MapReduce employs the same structure as HDFS. With this system, HIVE, HBASE, and SQL can also be used to manage databases storing historical information.

Layer 4: The service layer is responsible for integrating third-party interfaces to the lowest level of objects and humans. This layer can autonomously function as a single site, merge with other locations, or deploy within cloud interfaces. This layer also implements additional functions; for example, unique global identity management is a key element for handling identification objects throughout the Universe at the application layer. Moreover, the proposed layer construction involves interaction between humans and various intelligent objects, necessitating an intelligent algorithm at the application level that can efficiently interact with people. Service layer functions include session initiation, establishing communication rules, interacting with heterogeneous objects, and session termination.

2.2 HIABC Algorithm Based on Hadoop and Improved ABC

To elaborate on the architecture of the proposed system, service scenarios are established as shown in [Figure 2: see original paper], including intelligent transportation control departments, intelligent weather forecasting departments, and intelligent hospital and health departments. These components are responsible for collecting heterogeneous data in IoT networks and can serve as the framework' s underlying layer. These components connect to intelligent decision-making and control systems through heterogeneous access technologies such as GSM, Wi-Fi, 3G, and 4G, with the intelligent decision-making system located at the intermediate level of the smart city framework.

A realistic IoT environment contains not only massive amounts of data but also complex computations and multiple applications. IoT system implementation relies on data acquisition and computational analysis. The smart environment concept aims to optimize residential resources, reduce traffic congestion, and provide effective medical services. Acquiring data related to daily operational activities is crucial for achieving these goals. However, data acquisition becomes particularly challenging due to large volumes of data generated by people and other connected devices. Therefore, this work considers converting data into digital formats. Low-cost and energy-efficient sensors have become effective mechanisms for acquiring heterogeneous data from urban IoT. As the number of connected devices increases, cities become smarter, and extensive deployment of heterogeneous sensors within urban and suburban areas facilitates the formation of smart city architectures. These sensors are responsible for collecting real-time data from nearby environments of different categories.

The underlying layer of the proposed architecture comprises multiple components. The key to smart homes is improving energy utilization efficiency in residential buildings. Household appliances are equipped with sensors that determine real-time energy consumption and transmit data to the intermediate

layer. The data processing layer defines a threshold for specific household energy consumption, with the data filtering process performed by data aggregation techniques to identify values exceeding the threshold for further optimization. The primary goal of the vehicle transportation system is to reduce urban traffic congestion. The data processing level defines the average time for transmission between two specified points. Sensors deployed on roadsides collect vehicle entry and exit information between two points. Embedded aggregation techniques determine congested roads by analyzing current travel times at specified locations. The meteorological department' s role is to determine weather conditions and other environmental parameters. For example, sensors deployed at certain locations can monitor urban carbon monoxide concentrations, transmitting collected data to the intermediate layer for appropriate filtering and processing to facilitate decision-making and event generation.

The proposed architecture employs multiple communication technologies, including ZigBee, Bluetooth, Wi-Fi, and cellular networks, to transmit sensed data to the data processing layer for data filtering, analysis, processing, storage, and decision-making. Consequently, this layer is considered the framework' s brain. To perform these tasks, multiple patterns are embedded in this layer. Initially, massive sensed data is filtered through aggregation mechanisms to obtain valuable real-time and offline data. The MapReduce paradigm is used for data analysis, while operations and storage are handled by the Hadoop Distributed File System (HDFS, HBASE, and HIVE).

2.2.1 Particle Filter-Based Data Filtering Aggregation techniques improve data processing efficiency through data filtering, with particle filter (PF) employed in the proposed framework to perform data filtering. PF is an optimal estimator that can remove noise from sensed data. The main idea of PF is to represent the posterior probability density function using a set of special random samples to estimate the minimum variance of different states.

Assuming in PF that $p(x_0)$ represents the initial probability density function of the state, then according to Bayesian theory, the state prediction equation is:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}$$

The state update equation is:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}$$

where

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k$$

Monte Carlo algorithms simplify the entire calculation process by discretizing the integral calculation into a sum of weighted samples. Let $\pi(x_{0:k}|z_{1:k})$ be the

importance function derived from the probability density function, and draw N independent samples from $\pi(x_{0:k}|z_{1:k})$:

$$x_{0:k}^{(i)} \sim \pi(x_{0:k}|z_{1:k}), \quad i = 1, 2, ..., N$$

The posterior expectation of the state can be obtained:

$$E[f(x_{0:k})] = \sum_{i=1}^{N} w_k^{(i)} f(x_{0:k}^{(i)})$$

where $w_k^{(i)} = \frac{p(x_{0:k}^{(i)}|z_{1:k})}{\pi(x_{0:k}^{(i)}|z_{1:k})}$ represents the importance weight.

The resampling method is introduced to solve the particle degeneracy problem in the calculation process. The effective number of particles is defined as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2}$$

When $N_{eff} < N_{th}$, the system performs resampling; otherwise, it proceeds to the next calculation step, where N_{th} represents the threshold determined by actual conditions.

2.2.2 Data Storage and Processing The proposed scheme stores and processes data within the Hadoop framework, utilizing MapReduce to analyze filtered data. MapReduce operates in two steps: first, mapping transforms the filtered dataset into another set of data, then the data created during the mapping process is combined to generate a reduced set of values. Data storage and processing play crucial roles in implementing smart cities. As shown in [Figure 2: see original paper], the proposed framework leverages multiple technologies such as HDFS, HBase, and HIVE to meet these requirements.

HDFS is Hadoop' s primary storage space, providing distributed storage that satisfies the scalability requirements of big data processing. To support autonomous decision-making, real-time read/write capabilities across the entire cluster are essential. Therefore, HBase is used to improve Hadoop' s processing speed by providing real-time lookup functions in memory cache, while also enhancing system availability and fault tolerance. HIVE provides query and management capabilities for massive data residing on Hadoop clusters. Since SQL cannot be used to query HIVE, HiveQL is employed to query data on Hadoop clusters.

2.2.3 HIABC Algorithm This paper proposes the HIABC algorithm for feature selection in big datasets. The Artificial Bee Colony (ABC) algorithm is a stochastic search metaheuristic global optimization algorithm consisting of three components: food sources, employed bees, and unemployed bees. Specifically:

(a) Food sources represent solutions to a given problem. (b) Employed bees are used to identify different food sources and store information to share with other bees in the hive. (c) Unemployed bees are divided into two categories: onlooker bees and scout bees. Onlooker bees receive shared information from employed bees to find higher-quality food sources; when employed bees become exhausted searching for food sources, they transform into scout bees and attempt to discover new food sources.

The main process of the ABC algorithm is as follows:

Employed Bees Phase: Each employed bee visits a food source and explores its neighborhood. To extract features, neighbors are created from the bit vector of the initial food source. A random number R_{ij} $(0 < R_{ij} < 1)$ is generated at each position of the bit vector. If this value is less than the modification rate parameter MR, the feature is injected into the subset. The neighbor is generated as:

$$v_{ij} = x_{ij} + \Phi_{ij}(x_{ij} - x_{kj})$$

where x_{ij} and x_{kj} are food sources, and Φ_{ij} is a random variable.

Onlooker Bees Phase: If the quality of a newly discovered food source is superior to that of the explored food source, the neighboring food source is considered the latest food source, and this information is shared with other bees.

Scout Bees Phase: In HIABC, data size grows exponentially, so this process continues until optimal parameters are selected in Hadoop. A niche-based population elimination mechanism is then executed to select the best individuals. N features are randomly created and submitted to the classifier, newly discovered sources are assigned to scout bees, and employed bees resume their tasks.

The ABC algorithm may experience stagnation during the search process. Therefore, this paper introduces an improved niching technique to reduce stagnation. If the distance between two scout individuals is less than a set threshold θ, the individual with smaller fitness value is penalized to increase its probability of elimination in subsequent evolution. After this step, superior individuals become dispersed in the constrained space, better maintaining population diversity. In the algorithm, when scout bees search within a subpopulation, if within a fixed number of evolutionary generations the best individuals of two consecutive generations satisfy:

$$L(x_i, x_j) \leq \theta \quad \text{and} \quad |F(x_i) - F(x_j)| \leq \sigma$$

where $2 \leq i, l \leq j \leq L$, $i \neq j$, $L(x_i, x_j)$ represents the distance, θ represents the set threshold, σ represents the standard deviation of fitness values of subpopulation individuals, and L represents the maximum evolutionary generations within the subpopulation, then this subpopulation has experienced stagnation and must be eliminated and reinitialized.

In a specific scenario of HIABC, each food source is associated with a bit vector (of size K, where K is the total number of features). Positions in the vector correspond to the total number of features to be evaluated. In this case, if the feature value equals 1, it indicates the feature is part of the evaluation subset; if the feature value equals 0, it indicates the feature is not part of the evaluation subset. Additionally, food sources store their quality information, given by the accuracy of the feature subset specified by the classifier.

The steps of the HIABC algorithm for feature selection are as follows:

a) In the Hadoop processing system, when using particle filters to remove noise from datasets, the system employs a forward search strategy to find the best and minimum number of features. In the forward search strategy, N food sources contain K features.

b) The feature subset of each food source is assigned to a classifier, which uses accuracy as the fitness value (accuracy is stored in the food source' s fitness).

c) The modification rate parameter (MR) is used to determine neighbors of selected food sources.

d) If the quality of a newly discovered food source is better than the currently explored food source, the neighboring food source is considered the latest food source, and this information is shared with other bees.

e) A niche-based population elimination mechanism is executed to select the best individuals.

f) N features are randomly created and submitted to the classifier, newly discovered sources are assigned to scout bees, and employed bees perform their tasks again.

The algorithm terminates when the maximum number of cycles is reached.

---

3 Experiments

The proposed architecture is tested and compared with the Random Forest algorithm and algorithms from references [18, 19]. Each method is tested on identical datasets, with ten repeated experiments conducted and the average results taken as final outcomes. All experiments are performed on Hadoop installed in a multi-cluster environment running Ubuntu 14.04 LTS. The proposed feature selection algorithm is implemented in Java.

3.1 Datasets

The proposed optimization and feature selection method is tested on ten datasets from the UCI Machine Learning Repository [20], using a multi-cluster

Hadoop system to test each dataset with different learning algorithms. Table 1 describes each dataset, with analysis primarily based on the number of features in each dataset.

Table 1: Datasets for Testing and Analysis

Dataset	Features	Examples
Indoor Activity Monitoring Sensors		
GPS Trajectories		
Indoor User Movement Prediction from RSS		
3D Road Network		
Wastewater Treatment Plant		
Twitter Dataset for Sentiment Analysis		

3.2 Results and Discussion

The performance evaluation and accuracy of feature selection based on IABC are obtained using 10-fold cross-validation with K different partitions. In this process, one partition serves as the test set while the remaining $K-1$ partitions serve as the training set, repeated ten times with final results averaged across all ten partitions. Additionally, features established during testing are normalized using Z-score normalization, which subtracts the mean of each feature set and divides by its standard deviation.

Different features in UCI datasets affect the performance and accuracy of feature selection algorithms. Table 2 compares the accuracy of selected features versus all features. Table 3 presents comparative analysis results under identical conditions among the proposed algorithm, Random Forest algorithm, and algorithms from references [18] and [19]. The selected features are significantly fewer than the original feature list. Compared with other methods, the proposed feature selection algorithm demonstrates better accuracy on most datasets. While accuracy is inferior on certain datasets such as Air Quality, 3D Road Network, and Cloud, it is nearly identical to other methods on GPS Trajectories and Twitter Dataset for Sentiment Analysis, and superior in remaining cases.

Table 2: Accuracy of Proposed System on UCI Datasets

Dataset	Selected Features	Accuracy (%)	Average Accuracy (%)
Indoor Activity Monitoring Sensors			
GPS Trajectories			
RSS Indoor User Movement Prediction			
3D Road Network			
Wastewater Treatment Plant			
Twitter Dataset for Sentiment Analysis			

Table 3: Accuracy Comparison Between Proposed Scheme and Other Methods

Dataset	Random Forest (%)	Ref [18] (%)	Ref [19] (%)	Proposed (%)
Indoor Activity Monitoring Sensors				
GPS Trajectories				
RSS Indoor User Movement Prediction				
3D Road Network				
Wastewater Treatment Plant				
Twitter Dataset for Sentiment Analysis				

Figure 3 [Figure 3: see original paper] compares classification accuracy results of the four algorithms on the GPS Trajectories dataset with different numbers of selected features. The proposed scheme demonstrates superior classification performance on this dataset compared to the other three feature selection algorithms, with classification accuracy continuously improving as the number of features increases. The proposed algorithm achieves its best accuracy of 85.62% when 12 features are selected.

The system based on IABC is compared with single-node Hadoop and Java-based query systems, as shown in Figure 4 [Figure 4: see original paper]. In single-node Hadoop, data processing occurs without any system optimization. Conversely, the Java-based query system is tested through all other swarm op-

timization methods, with a filtering system used to remove noise from data before passing it to the Hadoop ecosystem. As data volume gradually increases, all three systems can process massive data in real time, but the proposed system requires significantly less processing time than the other two systems under the same data volume. The proposed system can effectively process data in real time and generate results to help objects make decisions. For instance, real-time processing of environmental data helps objects avoid heavily polluted areas.

Figure 5 [Figure 5: see original paper] illustrates throughput testing of the proposed system by increasing data size. Throughput is proportional to dataset size, increasing as dataset size grows. Initially, data processing speeds of the three systems differ minimally. However, as dataset size increases, processing speeds of single-node Hadoop and Java-based query systems decrease substantially, while the proposed scheme maintains high efficiency.

The proposed scheme requires only seconds to process data on the scale of gigabytes, as measured on different medical datasets shown in Figures 6 [Figure 6: see original paper] and 7 [Figure 7: see original paper]. Moreover, throughput is maximized as dataset size increases. These results demonstrate that the proposed system with parallel processing achieves better data processing performance.

---

4 Conclusion

This paper proposes a system architecture for feature selection in IoT big data. The proposed scheme is based on a four-layer architectural model that can efficiently aggregate effective data while eliminating unwanted information. The entire system utilizes an improved bee colony algorithm for feature selection, processes data through the Hadoop ecosystem, is implemented based on MapReduce tools, and employs particle filtering to eliminate noise. Results demonstrate that using IABC in the Hadoop ecosystem significantly improves the efficiency of system feature selection.

References

[1] Zhao Huiqun, Li Huifeng, Liu Jinluan. Study on RFID complex event pattern clustering algorithm of Internet of things [J]. Application Research of Computers, 2018, 35(2): 339-341.

[2] Qian Zhihong, Wang Yijun. IoT Technology and Application Research [J]. Acta Electronica Sinica, 2012, 40(5): 1023-1029.

[3] Ahmad A, Paul A, Rathore M, et al. An efficient multidimensional big data fusion approach in machine-to-machine communication [J]. ACM Trans on Embedded Computing Systems, 2016, 15(2): 32-39.

[4] Wei Baoya, Lin Menglei, Zheng Yifeng. Multi-label feature selection algorithm based labeling-importance [J]. Natural Science Journal of Xiangtan University, 2017, 39(4): 1-5.

[5] Lyu Lin, Wei Yongqing, Ren Min, et al. Agglomerative hierarchical clustering based on ant colony optimization algorithm [J]. Application Research of Computers, 2017, 34(1): 114-117.

[6] Tang Y, Guan X. Parameter estimation for time-delay chaotic system by particle swarm optimization [J]. Chaos Solitons & Fractals, 2017, 40(3): 1397-1402.

[7] Shang Junna, Liu Chunju, Yue Keqiang, et al. The multi-agent bat algorithm applied to wireless sensor networks [J]. Chinese Journal of Sensors and Actuators, 2015, 28(9): 1418-1424.

[8] Akay B, Karaboga D. A modified Artificial bee colony algorithm for real-parameter optimization [J]. Information Sciences, 2012, 192(1): 120-142.

[9] Civicioglu P, Besdok E. A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms [J]. Artificial Intelligence Review, 2013, 39(4): 315-346.

[10] Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system [C]// Proc of IEEE Symposium on MASS Storage Systems and Technologies. Washington DC: IEEE Computer Society, 2010: 1-10.

[11] Rong W, Zhang X, Dave C, et al. Smart city architecture: a technology guide for implementation and design challenges [J]. China Commun, 2014, 11(3): 56-69.

[12] Sanchez L, Muñoz L, Galache J A, et al. SmartSantander: IoT experimentation over a smart city testbed [J]. Computer Networks, 2014, 61(6): 217-238.

[13] Cui Lizhen, Wu Di, He Jiaxing, et al. Research and implementation on underground tracking algorithm based on improved particle filter [J]. Application Research of Computers, 2017, 34(5): 1476-1479.

[14] Dean J, Ghemawat S. MapReduce: a flexible data processing tool [J]. Communications of the ACM, 2010, 53(1): 72-77.

[15] Bao L, Zeng J C. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm [C]// Proc of International Conference on Hybrid Intelligent Systems. 2009: 411-416.

[16] Quabab B Y. Niching particle swarm optimization with local search for multi-modal optimization [J]. Information Sciences, 2012, 197(197): 174-191.

[17] Wang Xiaomei, Lin Xiaohui, Huang Xin. Algorithm of forward feature selection and aggregation of classifiers based on feature effective range [J]. Journal of Chinese Computer Systems, 2016, 37(6): 1159-1163.

[18] Yao Dengju, Yang Jing, Zhan Xiaojuan. Feature selection algorithm based on random forest [J]. Journal of Jilin University: Engineering and Technology Edition, 2014, 44(1): 137-141.

[19] Zhang Jin, Ding Sheng, Li Bo. Improved particle swarm optimization algorithm for support vector machine feature selection and optimization of parameters [J]. Journal of Computer Applications, 2016, 36(5): 1330-1335.

[20] Dash M, Choi K, Scheuermann P, et al. Feature Selection for Clustering-A Filter Solution [C]// Proc of IEEE International Conference on Data Mining. 2002: 115-122.

Note: Figure translations are in progress. See original paper for figures.