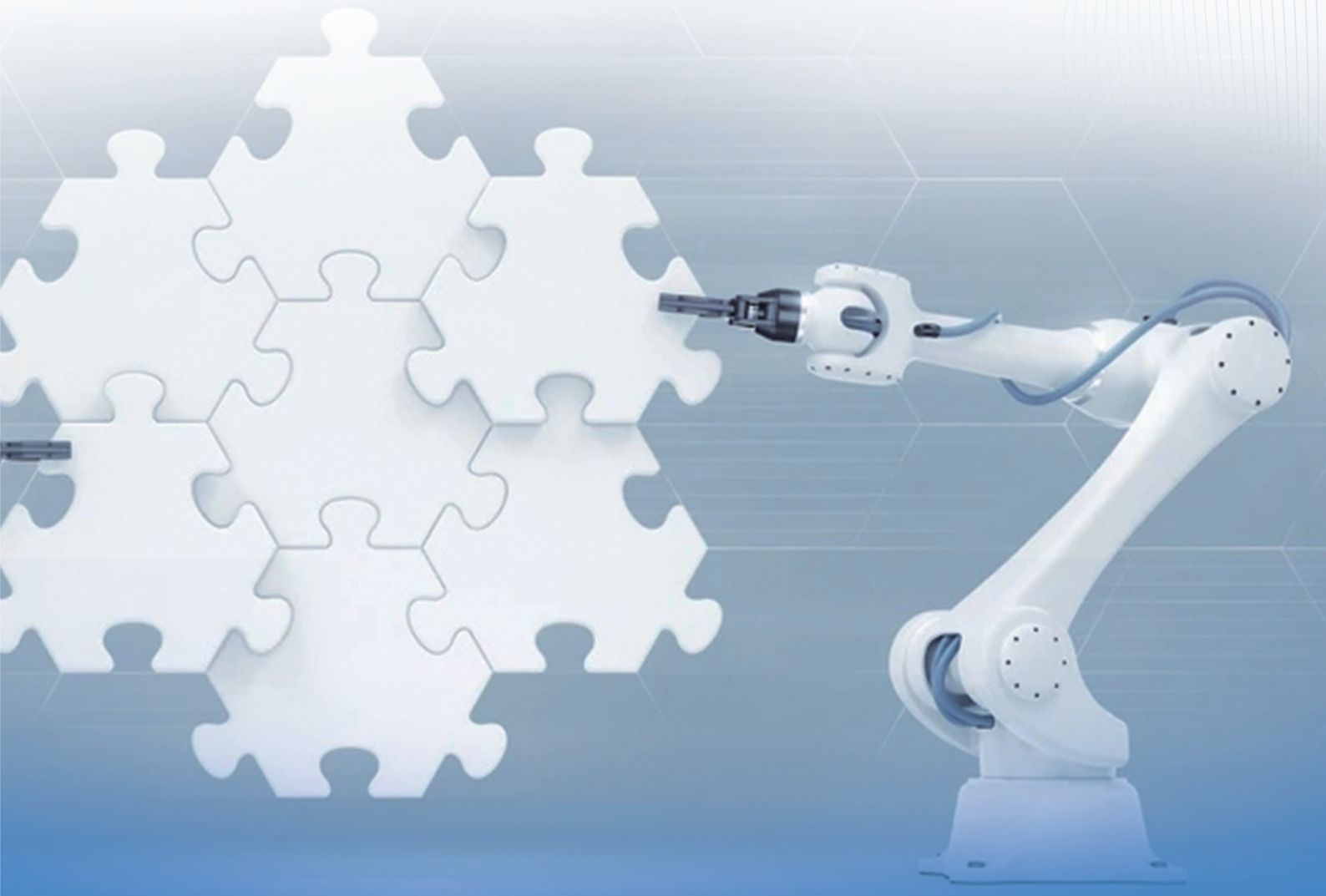
Volume 4·Issue 1·April 2022
ISSN 2661-3220(Online)

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances



Volume 4 Issue 1 • April 2022 • ISSN 2661-3220 (Online)

Artificial Intelligence Advances

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Volume 4 | Issue 1 | April 2022 | Page1-41
Artificial Intelligence Advances

Contents

Articles

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances
https://ojs.bilpublishing.com/index.php/aia

ARTICLE

Metric-based Few-shot Classification in Remote Sensing Image

Mengyue Zhang[1,2] Jinyong Chen[1,2] Gang Wang[1,2*] Min Wang[1,2] Kang Sun[1]

1. The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, Hebei, 050081, China
2. CETC Key Laboratory of Aerospace Information Applications, Shijiazhuang, 050081, Hebei, China

ARTICLE INFO

Article history
Received: 23 November 2021
Revised: 14 February 2022
Accepted: 21 February 2022
Published: 8 March 2022

Keywords:
Few-shot
Metric learning
Remote sensing
Target recognition
Episodic training

ABSTRACT

Target recognition based on deep learning relies on a large quantity of samples, but in some specific remote sensing scenes, the samples are very rare. Currently, few-shot learning can obtain high-performance target classification models using only a few samples, but most researches are based on the natural scene. Therefore, this paper proposes a metric-based few-shot classification technology in remote sensing. First, we constructed a dataset (RSD-FSC) for few-shot classification in remote sensing, which contained 21 classes typical target sample slices of remote sensing images. Second, based on metric learning, a k-nearest neighbor classification network is proposed, to find multiple training samples similar to the testing target, and then the similarity between the testing target and multiple similar samples is calculated to classify the testing target. Finally, the 5-way 1-shot, 5-way 5-shot and 5-way 10-shot experiments are conducted to improve the generalization of the model on few-shot classification tasks. The experimental results show that for the newly emerged classes few-shot samples, when the number of training samples is 1, 5 and 10, the average accuracy of target recognition can reach 59.134%, 82.553% and 87.796%, respectively. It demonstrates that our proposed method can resolve few-shot classification in remote sensing image and perform better than other few-shot classification methods.

1. Introduction

Remote sensing technology is a spatial observation technique, which captures remote sensing image of the earth's surface. It has numerous characteristics, including wide observation range, high efficiency and less restriction to geographic conditions. Therefore, it is widely used in geological research, environmental monitoring, military defense, etc. Moreover, with the progress of aerospace technology, space-borne remote sensing detection technology is developing very rapidly. China's high-resolution satellites have imaging capability of panchromatic,

*Corresponding Author:
Gang Wang,
The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, Hebei, 050081, China; CETC Key Laboratory of Aerospace Information Applications, Shijiazhuang, 050081, Hebei, China;
Email: wanggg@tju.edu.cn

DOI: https://doi.org/10.30564/aia.v4i1.4124

multispectral and radar, which can achieve high-resolution earth observations and capture high-quality remote sensing images of the earth's surface. Recently, in deep learning, researchers have utilized massive remote sensing images to carry out various forms of research and applications, such as remote sensing image target recognition, and proposed neural networks such as Faster R-CNN [1], YOLO [2], R-FCN [3] and SSD [4], which have made significant progress in target recognition.

Target recognition based on deep learning is widely used in the field of remote sensing. This method can achieve promising target recognition performance through training and optimizing by a large number of labeled samples. Generally, it is believed that supervised deep learning algorithms can achieve acceptable performance upon 5000 labeled samples per class. When a data set of at least 10 million labeled samples is used for training, it will meet or exceed human performance. However, in some specific remote sensing scenarios, the number of available samples is very limited. For example, it is exceptionally difficult for dynamic targets such as ships to obtain sufficient samples of multiple types, conditions and sun indidence angles through remote sensing technology. Some ships, whose quantity is small and frequency of dispatches is low, there may be rare samples or no samples. In such a scenario, when samples are rare, the deep learning network will certainly have problems such as over-fitting. Besides, the reduction in the quantity of samples will cause the difference between targets of the same class to become larger, which may deteriorate the performance of target recognition. This situation is a typical few-shot problem.

To solve the few-shot problem, researchers performed Few-Shot Learning (FSL), that is, for a particular task T, the algorithm uses the limited training data D_{train} to learn experience E, thereby improving the performance of the algorithm on the test data D_{test}. Researchers have conducted few-shot target recognition research on natural image datasets, such as data augmentation, meta-learning based methods, metric-learning based methods and so forth. Augmentation of data attempts to create active samples to improve the size of data. Meta-learning based methods introduce meta-learning to make the meta-learner generalize to new tasks. Metric-learning based methods utilize metric learning to learn a mapping space, where targets can be separated easily. All of the methods as mentioned earlier have made great progress in few-shot classification.

The current methods mainly focused on data augmentation, optimizer, transfer learning, however, they did not pay sufficient attention to that in remote sensing scenarios, the scarcity of samples deteriorated the problem that intra-class distance is large and inter-class distance is small.

Thus, in this paper, we develop a new metric-based method to work out few-shot target recognition. First, we proposed a k-nearest neighbor target recognition network, which used k-nearest neighbor [5] to find remarkable features in the measurement module. By this way, the difficult problem caused by large intra-class distance and small inter-class distance can be effectively solved. Furthermore, we introduced episodic training to improve the generalization of the model.

To test our model, we prepared a remote sensing dataset of few-shot classification, which is called RSD-FSC, containing 21 classes of typical remote sensing targets. Experiments are conducted on this dataset. Compared to the Prototype network [12], it improves the 1-shot, 5-shot and 10-shot accuracy from 56.62% to 59.13%, from 67.32% to 82.55% and from 71.18% to 87.80%, respectively.

2. Related Work

There are two main directions of few-shot learning from the existing literature. On one side, researchers use data generation methods to augment samples, such as using Generative Adversarial Networks (GANs) to generate highly realistic new samples through adversarial training of generative networks and judgment networks [6-9]; AJ Ratner et al. introduced a geometric-changes-sequence model based on geometric changes such as horizontal flipping, rotation, scaling, and adding Gaussian noise to achieve random geometric changes to existing samples, complete the amplification of the number of samples and ensure the target in the image without serious distortion [10]; Amit Alfassy et al. proposed Label-Set Operations networks (LaSO), which uses convolutional neural networks to extract the features of several images, and combines the features, such as intersection, union, difference, etc. to get the features of the hidden target in the image and generate new samples [11]. On the other side, the investigator conducted research based on the structure of model and training methods. Prototype network, which is based on the metric learning, gets the feature tensor of images by convolutional neural networks. Then, it calculates the average feature tensor as a prototype for each class, and finally calculates the distance between query images and prototypes as the similarity to recognize the category of targets [12]. Chen Lifu et al. introduced transfer learning to learn a supervised pre-trained convolutional neural network in the source domain, and then use a few samples in the target domain to retrain the model. This training method based on transfer learning improves the target recognition accuracy and convergence speed of the network [13]. Task-Agnostic Meta-Learning (TAML) introduced an optimization strategy to prevent meta-learner bias in several tasks [14].

3. The Proposed Method

3.1 K-nearest Neighbor Target Recognition Network

Metric learning can maximize the inter-class variations and minimize the intra-class variations in a mapping space, therefore, this paper proposed a metric-based k-nearest neighbor target recognition network. The essential idea is that in the feature mapping space, similar targets are close, and different targets are far away. The cosine distance between targets is used as the similarity to judge the category of targets. The flowchart of the k-nearest neighbor target recognition network is shown in Figure 1. It is mainly composed of two parts: a feature extractor and a feature measurement module, which correspondingly perform the extracting features of the targets in remote sensing images and the measurement of similarity between targets.

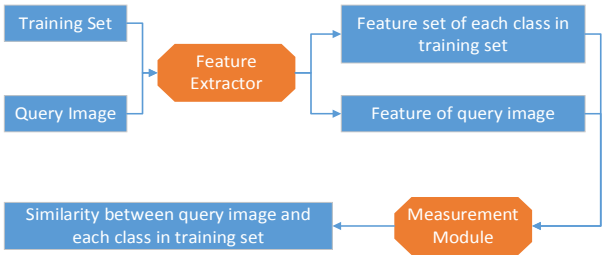


Figure 1. K-nearest neighbor target recognition network

1) Feature extractor

The feature extractor can be any suitable convolutional neural networks without the fully connected layer, which can extract deep local features of the image.

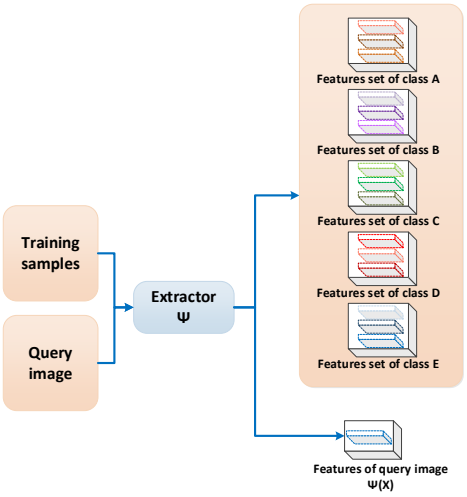


Figure 2. Feature extractor

As shown in Figure 2, the input of the feature extractor are images. The input image X passes through the feature extractor. The feature extractor outputs a feature tensor $\Psi(X)$:

$$\Psi(X)=[x_1,x_2,...,x_n]\in R^{d\times n} \quad (1)$$

Where x_i is a feature kind of the target, which has d n-dimensional feature vectors. Thus, we can get the feature set of each class of targets:

$$\{\Psi(X)^1,\Psi(X)^2,...\}=\{[x_1,x_2,...,x_n]^1,[x_1,x_2,...,x_n]^2,[x_1,x_2,...,x_n]^3,...\} \quad (2)$$

Furthermore, when input a query image q, the feature extractor outputs its feature tensor $\Psi(q)$:

$$\Psi(q)=[q_1,q_2,...,q_n]\in R^{d\times n} \quad (3)$$

Therefore, the feature extractor turns images into deep local feature tensors, which can distinguish different classes of targets. Through the feature extractor, we get the feature tensor of the query image and the feature set of each class of targets of training samples. Besides, the feature extractor outputs the feature tensors to the measurement module to calculate the similarity between targets.

2) Measurement Module

In the measurement module, motivated by the prototype network [12], using the distance between targets in the mapping space to recognize targets. Prototype network gets feature tensors of images by convolutional neural networks, and then calculates the average feature tensor as a prototype for each class, and finally calculates the distance between query images and prototypes as the similarity to recognize the category of targets. However, in few-shot learning, the inter-class variation is small, and the intra-class variation is large, which is problematical, thus, the average feature tensor can't represent each class well. Consequently, this paper uses the k-nearest neighbor to find representative samples to calculate the similarity between query images and every class of training sets.

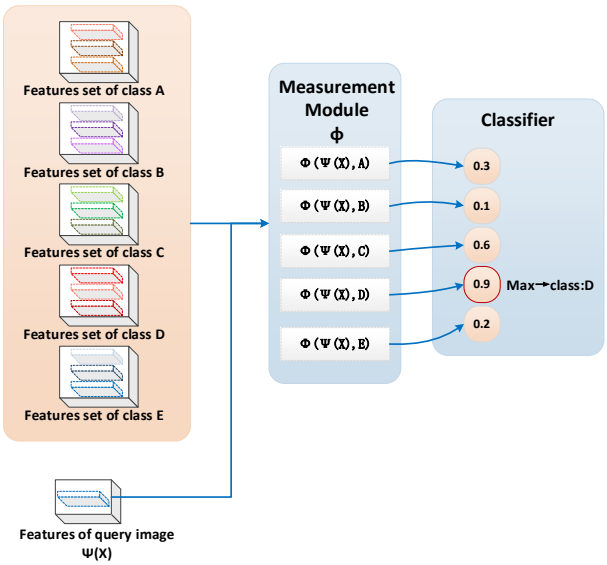


Figure 3. Measurement module

As shown in FIGURE 3, first, the measurement module gets the feature tensor $\Psi(q)$ of the query image and the features set $\{\Psi(X)^1, \Psi(X)^2, \ldots\}$ of each class of targets from the feature extractor. Second, we use k-nearest neighbors to find the top-k feature vectors similar to the feature of the query image in the feature set of each class, and then calculate the cosine distance between the query image and each class of targets, respectively. This cosine distance is defined as the similarity between the query image and each class of targets. The smaller the distance, the larger will be the similarity.

The calculation process is as follows. For each feature component x_i of the query image, using k-nearest neighbor to find the top-k similar feature vectors from the features set of each class targets:

$$\text{Top-k similar feature vectors} = \{\hat{x}_i^1, x_i^2, ..., x_i^k\} \tag{4}$$

Calculate the cosine distance between x_i and each $\hat{x}_i$ for every feature component, and sum k distances up to get the similarity D_i between the query image and each class of targets in individually feature component:

$$\begin{aligned} \cos(x_i, \hat{x}_i) &= \frac{x_i^{\mathrm{T}} \hat{x}_i}{\| x_i \| \bullet \| \hat{x}_i \|} \\ D_i(\Psi(q), c) &= \sum_{j=1}^{k} \cos(x_i, \hat{x}_i) \end{aligned} \tag{5}$$

For each image, there are n feature components. So, for each image, we calculate n times distance D_i and sum up them to get the similarity D between the feature of the query image and the feature set of each class of targets:

$$D(\Psi(q), c) = \sum_{i=1}^{n} D_i(\Psi(q), c) \tag{6}$$

Here, c represents the feature set of various targets of class A, B, C, D, and E.

This metric module considered the challenge when the number of samples is rare, the difference between different kinds of targets is smaller and the distance between the same kind of targets is larger. Moreover, when the quantity of the training samples is small, calculating the similarity between the query image and training images and finding representative training images are possible.

3.2 Episodic Training

In transfer learning, researcher tunes the networks from the source domain with few training samples in the target domain. However, the networks from the source domain are trained with thousands of samples. Thus, it is extremely hard to adjust the source networks to the target domain due to over-fitting. Consequently, this paper adopts a training mode of episodic training. As shown in Figure 4, the model succeeds in few-shot target recognition on the testing set by simulating similar few-shot target recognition on the training set. Concretely, we randomly sample from the training set to construct thousands of few-shot target recognition tasks to improve the model's ability of generalization. When facing new classes of targets, the model can recognize the targets through training with only a few samples.

In the training stage, we randomly collect samples from the training set to form lots of N-way K-shot learning tasks. In target recognition, the N-way K-shot task refers to learning to recognize N types of targets, and each class of target has K labeled samples that can be used as training data [15]. Specifically, in this paper, we carried out 5-way 1-shot, 5-shot and 10-shot target recognition tasks in the training stage of the simulation experiment. Thus, we learnt to recognize 5 kinds of targets in each task, each class of targets had only 1, 5, or 10 labeled training samples. Each task was called 1 episode. We carried out 100,000 stages of training tasks, and the recognition result of each episode was fed back to the network to adjust the parameters of our network.

In the testing stage, where there is the real few-shot scene, we performed 5-way 1-shot, 5-shot and 10-shot target recognition tasks, respectively, to test whether the model can recognize targets with few samples when facing new classes of targets.

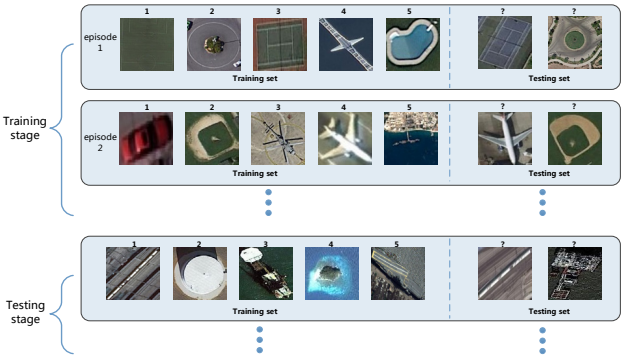


Figure 4. Episodic training

4. Experimental Results

4.1 Datasets

Mini-ImageNet [16], Stanford Dogs [17], Stanford Cars [18], and etc. are datasets that are commonly used in few-shot learning. The pictures in these datasets only contain a single target without complex background, which is helpful for the classifier to learn features of target. However, these datasets mainly contain natural scene images, which are different from remote sensing images. In response to

this problem, this paper constructed a target slices dataset of remote sensing images and conducted few-shot remote sensing image target recognition on this dataset.

Based on the remote sensing image dataset of DOTA [19], NWPU VHR-10 [20-22], RSD-GOD [23] and some remote sensing images from Gaofen-2 satellite, this paper prepared a remote sensing dataset of few-shot classification, which is called RSD-FSC, for target recognition. There are 21 types of targets, including aircraft carrier, oil platform, oil tank, train, island, warship, harbor, missile, large vehicle, swimming pool, bridge, tennis court, airplane, small vehicle, helicopter, roundabout, basketball court, ground track field, baseball court, soccer ball field and airport. The samples are shown in Figure 5. There are about 100,000 remote sensing image target slices with multi-resolution and multi-scale. The number distribution is shown in Table 1.

Table 1. Statistics of RSD-FSC

NO.	Categories	Instances
1	warship	1,008
2	harbor	1,875
3	missile	9,916
4	large vehicle	10,063
5	swimming pool	576
6	bridge	10,131
7	tennis court	524
8	airplane	756
9	small vehicle	598
10	helicopter	127
11	roundabout	265
12	basketball court	159
13	ground track field	163
14	baseball court	390
15	soccer ball field	316
16	airport	4,182
17	oil platform	9,957
18	aircraft carrier	70
19	island	11,420
20	oil tank	45,211
21	train	6,733

4.2 Experimental Setting

Settings for k-nearest neighbor target recognition network

K-nearest neighbor target recognition network was conducted in a Linux system, using Python 3.7.2 and torch 0.4.1.

During the training stage, we chose aircraft carrier, train, harbor, missile, large vehicle, swimming pool, tennis court, airplane, small vehicle, helicopter, roundabout, basketball court, ground track field, baseball court, soccer ball field and airport as the auxiliary training set. We conducted 100,000 few-shot target recognition tasks. For each task, we first randomly sampled 5 classes from 16 classes, which would be recognized. Then, we randomly chose 1,5 or 10 samples for each class as the training set and 10 or 15 samples as the testing set. The results of recognition were fed back to adjust the parameters of the network.

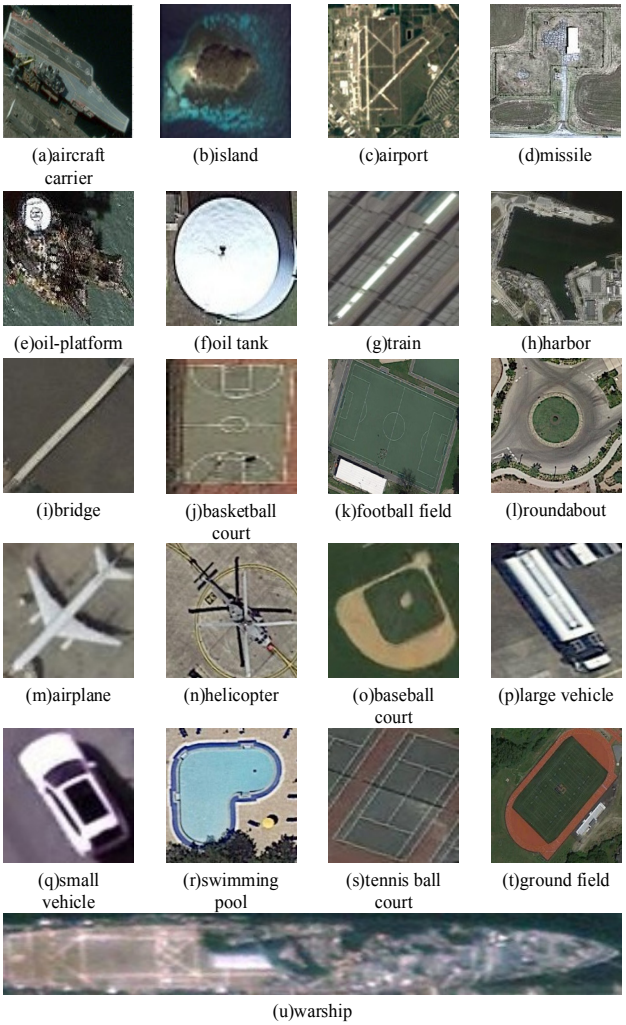


Figure 5. images from RSD-FSC

In the testing stage, to simulate few-shot scenario, we chose oil platform, oil tank, bridge, island and warship as a few-shot set. We conducted 3,000 few-shot target recognition tasks. For each task, based on the few-shot set, we randomly chose 1, 5 or 10 samples for each class as the training set and 10 or 15 samples as the testing set. The results of recognition were counted to calculate the average accuracy of recognition and not fed back to adjust the network's parameters.

Comparison Methods

First, we trained *ResNet-256F* [24], which is a deep neural network, with 1, 5, 10 and thousands of samples for each class from the few-shot set. The aim of the training is to verify whether the deep learning can work out few-shot target recognition.

Second, to prove the efficiency of k-nearest neighbor target recognition network, we conducted a comparative experiment with prototype network [12] and matching network [16] on RSD-FSC. Concretely, we conducted 5-way 1-shot, 5-shot or 10-shot tasks respectively to calculate the average accuracy of recognition.

4.3 Results

The results are shown in TABLE II. Our result showed that when training samples were sufficient, *ResNet-256F* performed efficiently, but when the quantity of training samples was decreased, it can't recognize targets accurately. Interestingly, the accuracy decreased when the quantity of training samples increased from 1 to 5, which was caused by overfitting.

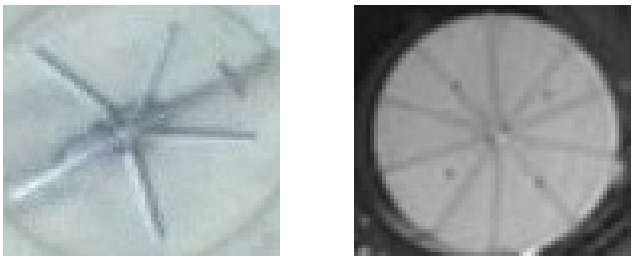
However, our proposed model combining episodic training and metric learning generalized well when facing new emerged classes in the few-shot scenario. After thousands of episodes of all few-shot target recognition tasks, the proposed model overcame overfitting and well recognized the targets with few training samples. Furthermore, the proposed model used *k*-NN in measurement module to calculate the similarity between the query image and each class of targets, which overcame the problem that the inter-class distance was small and intra-class distance was large. Compared with prototype network and matching network, it achieved 7.52% and 10.75% improvements in the 5-way 1-shot classification task. In the 5-way 5-shot classification task, the model achieved 15.24% and 21.20% improvements. In the 5-way 10-shot classification task, our model achieved 16.62% and 18.71% improvements. This was because that these models utilized all the training samples to measure similarity, and didn't effectively limit the classifier due to the effect of smaller inter-class distances and larger intra-class distances.

Table 2. Average Precision of Recognition (%)

Model	1-shot (k=3)	5-shot (k=3)	10-shot (k=3)
Ours	**59.134**	**82.553**	**87.796**
ResNet-256F [24]	26.700	20.000	5.000
Prototype network [12]	51.617	67.315	71.181
Matching network [16]	48.384	61.350	69.091

Moreover, according to the recognition results, we found that compared methods tended to confuse oil-platform with oil tank, train with bridge and helicopter with airplane.

As shown in Figure 6, the helicopter is tend to be recognized as an oil tank by the compared methods. The reason is that the pattern composed by the apron and airscrew is similar to the pattern on the top of the oil tank. However, it can be efficiently recognized by our method.

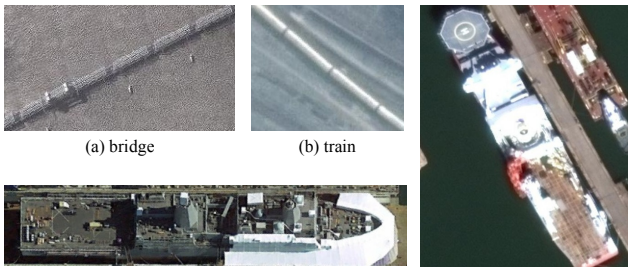


(a) helicopter (b) oil tank

Figure 6. Comparison of helicopter and oil tank

Furthermore, as shown in Figure 7, it's easy to find that textures of bridge and train in remote sensing images are extremely similar, which leads to low recognition accuracy.

Similarly, some oil platforms tend to be confused with the warship, too.



(a) bridge (b) train (c) warship (d) oil platform

Figure 7. Confusion examples

5. Conclusions

In this paper, first, we constructed a dataset RSD-FSC for few-shot target recognition. Second, based on metric learning, we proposed a k-nearest neighbor target recognition network, which can overcome the deterioration of smaller inter-class distance and larger intra-class distance caused by few-shot. Third, we introduced episodic training, which can overcome overfitting and enable the model generalize to new classes of targets in few-shot target recognition. Finally, we carried out 5-way 1-shot, 5-shot and 10-shot experiments using the k-nearest neighbor target recognition network and contrast experiments on the dataset RSD-FSC. The experiment results show that in the few-shot target recognition, the k-nearest neighbor

target recognition network proposed in this paper is useful and superior to the compared networks. The method proposed in this paper is of great significance to promote the research of few-shot target detection and recognition technology in remote sensing images.

Funding

This work was supported in part by the CETC key laboratory of Aerospace Information Applications under Grant No.SXX19629X060.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Ren, S., He, K., Girshick, R., et al., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems. pp. 91-99.

[2] Redmon, J., Divvala, S., Girshick, R., et al., 2016. You only look once: Unified, real-time object detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 779-788.

[3] Girshick, R., Donahue, J., Darrell, T., et al., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 580-587.

[4] Liu, W., Anguelov, D., Erhan, D., et al., 2016. SSD: Single shot multibox detector. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). vol. 9905 LNCS. pp. 21-37. DOI: https://doi.org/10.1007/978-3-319-46448-0_2.

[5] Boiman, O., Shechtman, E., Irani, M., 2008. In defense of nearest-neighbor based image classification. 26th IEEE Conf. Comput. Vis. Pattern Recognition. CVPR. pp. 1-8.

[6] Goodfellow, I.J., et al., 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 3, 2672-2680.

[7] Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[8] Mao, X., Li, Q., Xie, H., et al., 2017. Least Squares Generative Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision. pp. 2813-2821.

[9] Durugkar, I., Gemp, I., Mahadevan, S., 2016. Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673.

[10] Ratner, A.J., Ehrenberg, H.R., Hussain, Z., et al., 2017. Learning to compose domain-specific transformations for data augmentation. Advances in Neural Information Processing Systems. pp. 3237-3247.

[11] Alfassy, A., Karlinsky, L., Aides, A., et al., 2019. Laso: Label-set operations networks for multi-label few-shot learning. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 6541-6550.

[12] Snell, J., Swersky, K., Zemel, R., 2017. Prototypical networks for few-shot learning. Advances in Neural Information Processing Systems. pp. 4078-4088.

[13] Chen, L., Wu, H., Cui, X., et al., 2018. Convolution neural network SAR image target recogniton based on transfer learning. Chinese Space Science and Technology. 38.6(2018), 46-51. (in Chinese)

[14] Jamal, M.A., Qi, G.J., 2019. Task agnostic meta-learning for few-shot learning. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 11711-11719.

[15] Wang, Y., Yao, Q., Kwok, J.T., et al., 2020. Generalizing from a Few Examples: A Survey on Few-shot Learning. ACM Comput. Surv. 53(3), 1-34.

[16] Oriol, V., Charles, B., Tim, L., 2016. kavukcuoglu koray, and W. Daan. Matching Networks for One Shot Learning. Advances in Neural Information Processing Systems. pp. 3630-3638.

[17] Khosla, A., Jayadevaprakash, N., Yao, B., et al., 2011. Novel dataset for fine-grained image categorization: Stanford dogs. Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC). 2(1).

[18] Krause, J., Stark, M., Deng, J., 2013. 3D object representations for fine-grained categorization. Proceedings of the IEEE International Conference on Computer Vision. pp. 554-561.

[19] Xia, G.S., Bai, X., Ding, J., et al., 2018. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3974-3983.

[20] Cheng, G., Han, J., Zhou, P., et al., 2014. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. ISPRS J. Photogramm. Remote Sens. 98, 119-132.

[21] Cheng, G., Han, J., 2016. A survey on object detection in optical remote sensing images. ISPRS J. Photogramm. Remote Sens. 117, 11-28.

[22] Cheng, G., Zhou, P., Han, J., 2016. Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing

Images. IEEE Trans. Geosci. Remote Sens. 54(12), 7405-7415.

[23] Zhuang, S., Wang, P., Jiang, B., et al., 2019. A single shot framework with multi-scale feature fusion for geospatial object detection. Remote Sens. 11(5), 594.

[24] He, K., Zhang, X., Ren, S., et al., 2016. Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, Las Vegas. 2016-December, 770-778.

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances
https://ojs.bilpublishing.com/index.php/aia

ARTICLE

Elderly Fall Detection by Sensitive Features Based on Image Processing and Machine Learning

Mohammad Hasan Olyaei[1*] Ali Olyaei[2] Sumaya Hamidi[3]

1. Faculty of Electrical Engineering, Sadjad University of Technology, Mashhad, Iran
2. Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
3. ARIVET Project, Mashhad, Iran

ARTICLE INFO

Article history
Received: 29 January 2022
Revised: 31 March 2022
Accepted: 2 April 2022
Published: 12 April 2022

Keywords:
Human fall detection
Machine learning
Computer vision
Elderly

ABSTRACT

The world's elderly population is growing every year. It is easy to say that the fall is one of the major dangers that threaten them. This paper offers a Trained Model for fall detection to help the older people live comfortably and alone at home. The purpose of this paper is to investigate appropriate methods for diagnosing falls by analyzing the motion and shape characteristics of the human body. Several machine learning technologies have been proposed for automatic fall detection. The proposed research reported in this paper detects a moving object by using a background subtraction algorithm with a single camera. The next step is to extract the features that are very important and generally describe the human shape and show the difference between the human falls from the daily activities. These features are based on motion, changes in human shape, and oval diameters around the human and temporal head position. The features extracted from the human mask are eventually fed in to various machine learning classifiers for fall detection. Experimental results showed the efficiency and reliability of the proposed method with a fall detection rate of 81% that have been tested with UR Fall Detection dataset.

1. Introduction

The number of elderly people (people with the age of ≥ 65 years) is increasing rapidly [1]. Based on the UK Office for National Statistics (ONS), more than 8.6 million people aged 65 or higher will exist in the next 5 decades [2]. The population of the elderly people in Iran according to the statistics of the United Nations in 2019 is 5272000 and it is predicted that this population will reach 8849000 in 2030 [3]. Existing health care systems face many shortcomings in meeting the care needs of the elderly and disabled. An independent life supported by self-monitoring is a suitable solution to the growing population of the elderly with limited budgets to care for them. Current technologies such as environmental intelligence can help the elderly and disabled to live longer in their homes. Falls

*Corresponding Author:
Mohammad Hasan Olyaei,
Faculty of Electrical Engineering, Sadjad University of Technology, Mashhad, Iran;
Email: mh.olyaei123@sadjad.ac.ir

DOI: https://doi.org/10.30564/aia.v4i1.4419

are a relatively common occurrence among the elderly, which can have significant health consequences. Studies have shown that 28% to 34% of the elderly fall at least once a year [4] and falls are the second leading reason for accidental death in more than 87% of the elderly [5]. Falling is one of the main causes of various types of injuries, such as fractures, muscle bruises or brain injuries in the elderly [6]. It is possible that the injured person will remain lying on the ground for a significant period of time before receiving assistance after the fall. Fall detection and alerting systems are unable to prevent falls.

However, these systems can ensure that the person who falls in quickly receives the help quickly, and this can reduce the effects of the accident [7]. Therefore, significant attention should be paid to develop the automatic human fall detection systems [8].

Currently, some of the advanced technologies for alarming of a fall accident are manually, inactive and based on devices that a person is required to push a button during an accident and wears it as a pendant or bracelet. This device is recommended for the elderly who live alone, as well as vulnerable people who have a history of falls and need to be worn constantly throughout the day. In the event of a fall, the victim is forced to push a button to generate a signal that is sent to crisis telecare center. But this approach depends on the level of awareness of the person at the time of the accident, it is necessary to have full mental awareness and cognizant of the required response, otherwise this device is simply decorative. Automatic human fall detection systems are needed to monitor vulnerable people susceptible to falling. A number of new methods of fall detection were established over the past few years to enable seniors to live in their own homes independently while also having acceptable monitoring tools. And this plan requires large-scale monitoring strategy. Ideally, a comprehensive detection system can do this while conserving the individuals' privacy being monitored. Numerous sensors are used to detect falls, such as cameras, infrared sensors, sound and pressure sensors, accelerometers and others. The development of these sensors has also helped improve fall detection algorithms, especially by reducing the price of cameras. And the development of novel image acquisition like cameras (Xtion, Kinect, etc.). Also, increasing the scope of computer vision has facilitated the deployment and development of computer visual programs. The use of in-depth data due to having an additional dimension has significantly increased the level of reliability and accuracy (in fact it has 2.5 dimensions). The high-level data provided by the OpenNI and Kinect SDK also simplify the division of the human body. With the arrival of Kinect technology in 2011, new research into fall detection has been utilizes using the features of depth sensors like 3D analysis with privacy, thus, more acceptable solutions were provided using deep data (Kinect) in fall detection system.

Research on vision-based monitoring systems to understand human behavior has steadily grown recently [9]. Computer-based image processing systems have provided promising new solutions to help older people stay at home [10]. One of the applications based on video surveillance is the analysis of human behavior and the detection of their unusual behavior.

According to the latest report of the World Health Organization (WHO), every year about 646,000 people die due to falls in the world. The fall is the second leading cause of death due to injury worldwide [11] and at least 1/3 of the elderly fall one or more times a year [12].

There are different types of falls, depending on the direction of the body when falling, and each of them has a different movement. Hence, a detection algorithm is needed to identify different types and minimize false detections. Also, considering privacy issues may negatively affect the choice of a vision-based fall detection system.

In this paper, we focus on the automatic human fall detection from data generated by the camera (or Webcam). This will be based on a monitoring system with low-cost video cameras to track elderly people who living independently.

2. Fall Detection Methods Based on Image Processing

Computer vision-based techniques can provide information about falls and other daily life behaviors. Fall events can be found using an automatic fall detection algorithm via intelligent monitoring systems [13]. Much work was done in the field of visual sensors in fall detection utilizing single, multiple and omni-directional cameras [14-16]. Presently, in-depth cameras such as Microsoft Kinect 2.0 and ASUS Xtion Pro Live was also utilized to detect falls. Kinect is a motion sensing technology that uses an RGB camera and a depth sensor to track a moving object in 3D [17-19]. Depth cameras like the Microsoft Kinect have been used to detect falls. Depth cameras are used to identify human activities. For example, the authors propose a spatial approach based on the skeleton to describe spatiotemporal aspects of a human activity using three-dimensional in-depth images [20]. Two features are calculated from the 3D joint position concluded the spatial and the temporal aspects of the activity. These features are used as input to the random tree algorithm to teach the model of human activity. The trained classified model achieved an accuracy of 80.92%.

The accuracy of a fall detection system must be such that it can be trusted in a real environment. Activities such as lying down may be misdiagnosed and lead to unnecessary alert generation. These movements are described as activities of daily living (ADL) or not falling. Therefore, algorithms must minimize the impact of such events, to increase system reliability, and to reduce false positive (FP).

There are different types of falls, depending on the direction of the body when falling, and each has a different movement. Therefore, a detection algorithm is needed to detect different types and minimize false detections. Also, considering privacy issues may affect negatively the choice of a vision-based fall detection system. A surveillance system should be able to hide some recognizable features, such as people's faces if needed.

Existing research on fall detection systems is normally based on limited amounts of human fall data (which are not real events, and are generated in a tuned or trained manner).

Recording fall events in real life is a complex process that requires costly infrastructure. Several research groups have documented falls in hospitals and nursing homes, but such data is not limited and available to the public due to copyright and privacy restrictions. An alternative approach is used, such as video recordings of human-simulated falls, in which participants try to guide researchers. Data collection is very common in data science, and collecting fall detection videos faces the following issues:

1) Demographics: For example, fall data and daily activity data, people should be of different ages. Likewise, these individuals with various physical features (e.g. weight, height, delivery) should be examined. Finally, the behavioral characteristics of these individuals, such as gait patterns, should be recorded and examined.

2) The quality, quantity and availability of the simulated samples are very important and should be in different places of the house.

3) Scene conditions: Visual obstruction is another issue that needs to be addressed. Dining tables, chairs, sofas and other accessories can act as a barrier in the home scene.

So far, previous methods have often achieved good accuracy by connecting gadgets and devices to the elderly, such as necklaces or mobile phone. But most seniors are dissatisfied with connecting extra gadgets or forget to always have them. For this reason, vision-based methods and the use of a camera or webcam can be a better alternative for these people.

In image processing, feature selection is one of the most important parameters to increase accuracy. It must also have a high processing speed for real-time execution. In our proposed algorithm, the important features that are selected and combined show that it has both speed and accuracy.

3. Our Proposed Algorithm

The proposed fall detection system in this paper is divided into four stages: data collection, foreground segmentation (or moving Human detection), feature extraction and fall detection.

Foreground segmentation is performed to detect moving objects [21]. The steps of the model training algorithm are shown in Figure 1 and the model test in Figure 1-B.

As shown in Figure 1, the videos are first read from the database and then the next steps are performed for all frames of all videos. In each frame, the mask of the person is first identified using the background estimation and Frame Difference methods (BMaker_oly.py) and the background subtraction method. The subtraction method is mainly aimed to convert frame images to a gray scale at any time and then separated from the background image at the pixel level for producing a binary image such as Equation (1) [22].

$$REk(X, Y) = FRk(X, Y) - BA(X, Y). \quad (1)$$

REK is the absolute difference between background and current images. FRK represents the current image and BA denotes the background image. The main idea of the

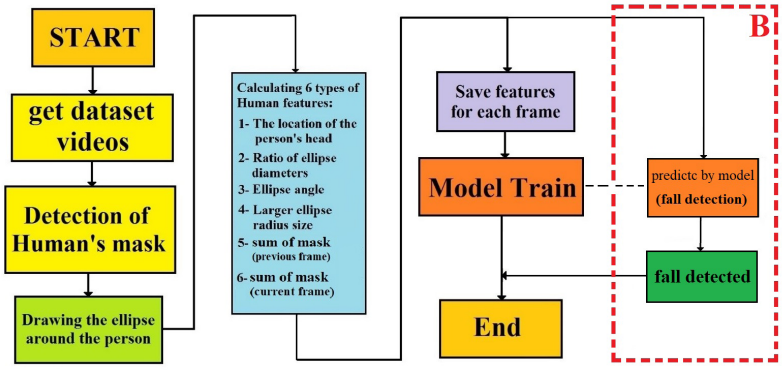


Figure 1. Our proposed algorithm - train and test (B section)

background estimation method (BMaker_oly.py) is that 12 frames of video are randomly selected and images are averaged between them. This averaging causes the moving objects to fade or even be removed and the background to remain. This background can be used to detect a moving object. Figure 2 shows a sample background estimate output for a video.

After this step, an ellipse is estimated around the human mask as shown in Figure 1 using OpenCV library and python language. From the detected and elliptical mask, 6 special features are extracted. These features are head position, ellipse diameter ratio, ellipse angle, larger ellipse radius, and sum of mask, respectively. The proposed method of this detection is the extraction of motion and shape features from previous observations of human falls compared to other daily activities, including severe changes in body shape and form, as well as the study of sudden and unexpected movements.

Many usual fall detection algorithms are oriented by the higher acceleration of fall events have than a daily life activity. Ultimately, the extracted properties are utilized as input vectors for existing classifiers in machine learning for categorizing fall and non-fall events.

Figure 3 shows the sum of mask_final changes of person pixels and the main label for videos of fall-1 and fall-4. It is clear that the sum of motion in frames of 110 and 40 has changed suddenly.

In these two diagrams, it is quite clear that in sum of motion mask, at the moment of the fall, creates a clear peak. This means that when we have a peak in sum of motion mask, it means that we have most likely fallen, because in this frame, there was a sudden movement. This proves that the sum of mask can be used as a very good feature.

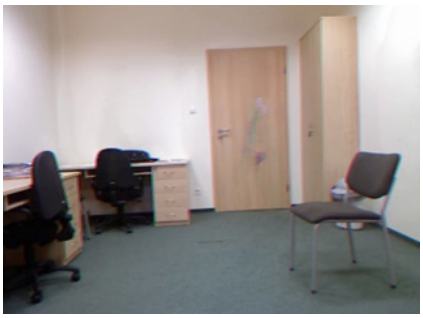


Figure 2. Sample output of background estimation

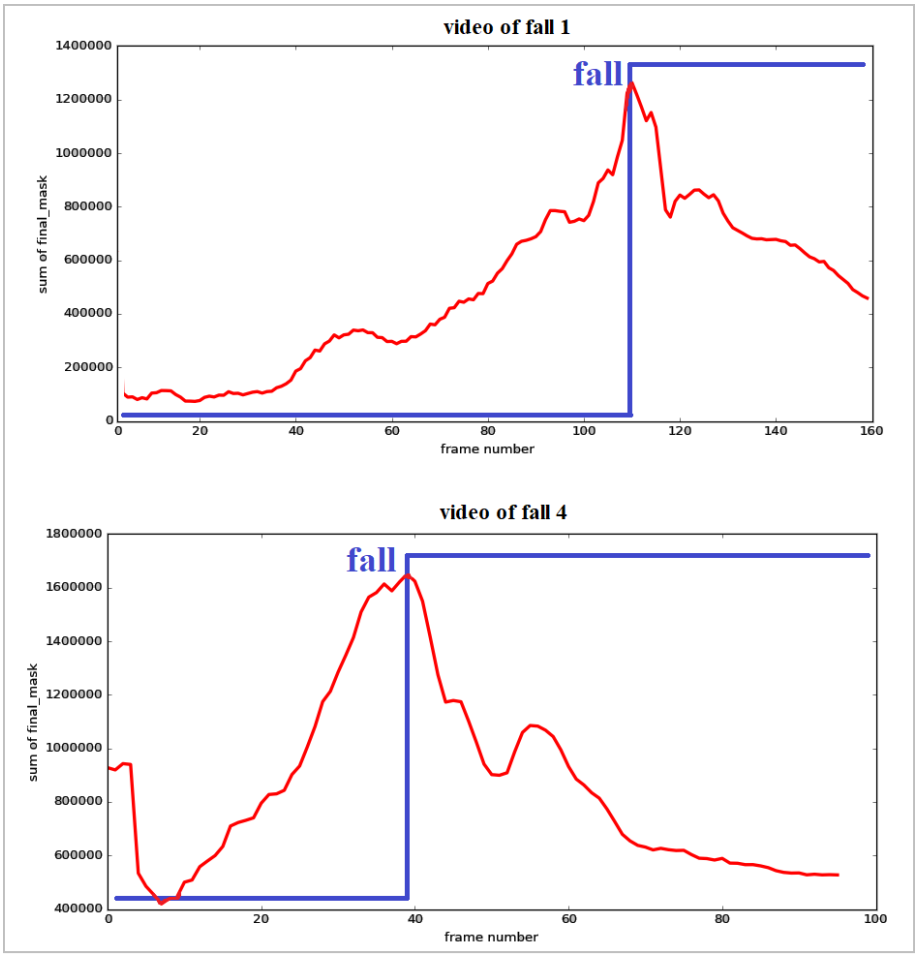


Figure 3. Sum of motion and label

4. Simulation Results

In this paper, we have used UR dataset [23] and the powerful Python language.

Seventy (30 falls + 40 activities of daily living) sequences are included in this dataset. Using 2 Microsoft Kinect cameras and correspondent accelerometric data, fall events are recorded. Only one device (camera 0) and accelerometer were used to record ADL events. To collect sensor data, x-IMU (256Hz) and PS Move (60Hz) devices were used.

The UR Fall Detection dataset includes video frames from the front and top of the head taken by two Kinect cameras. The first camera is installed at a height of 1 meter from the floor and the second camera is located at the ceiling at a height of 3 meters. There are 30 fall scenarios recorded by two Kinect sensors and an accelerometer. Daily activities include walking, sitting, bending over, lifting an object off the floor, and lying down. Normal activities (30 scenarios) using a Kinect sensor that is parallel to the floor and collects 10 sequences of activities such as falling, such as lying on the floor fast and lying on the bed. Five participants were asked to perform daily activities and fall simulations. The dataset was recorded at 30 frames per second. In video sequences containing fall events, the number of images is 3,000, but the number of images in video sequences with ADLs is 10,000. Falling in different directions is simulated according to the camera view. Various types of fall accidents have been recorded, including falling backward, forward, and sideways. Participants performed the fall while standing and sitting on a chair. Figure 4 shows examples of frames taken from the UR fall detection dataset. Only RGB camera video data was used for the suggested fall detection approach.

The dataset is organized as follows. Sequence of depth and RGB images for camera 0 and camera 1 are contained in each row, that we only used its camera 0 videos and RGB section of each frame.

In general, three labels are recorded for each frame. If the person is normal, the label is –1. If a person starts to fall, the label=0. Finally, after a complete fall, the label is 1. You can see an example of these three situations in the Figure 4.

Machine learning-based methods compared in this paper include support vector machine (SVM), k-nearest neighbors (KNN), Adaptive Boosting (Adaboost), Multiple-layer perceptron neural network (MLPNN) and Elman neural network (ENN).

In this paper, to more accurately detect the human mask, the mask from the background estimate and frame difference and the motion mask from the subtractor model (changes in a person's movement) are added together logically. For example, look at Figure 5.

Figure 5A is the result of detecting and drawing an ellipse, 5B is the result of motion changes, 5C is the result of background estimation, and finally Figure 5D is the result of the logical adding of two masks.

Figure 6 shows an example of the implementation of the steps in Figure 1.

In the background subtraction method, if the amount of brightness due to the pixel difference is less than the threshold value, the object is considered as the background pixel and the value 0 in the binary image is assigned to it. Otherwise, the pixel is taken as the background and assigned a value of 1 in Equation (2) [24].

$$D_K(X,Y) = \begin{cases} 0 & background \quad R_K(X,Y) > T \\ 1 & target \quad R_K(X,Y) < T \end{cases} \tag{2}$$

By applying the proposed algorithm and using 5 machine learning methods, the results of performance evaluation of the models are shown in Table 1. By observing and comparing the methods with each other, it can be concluded that the highest accuracy is for Adaboost method and it shows in Figure 7.

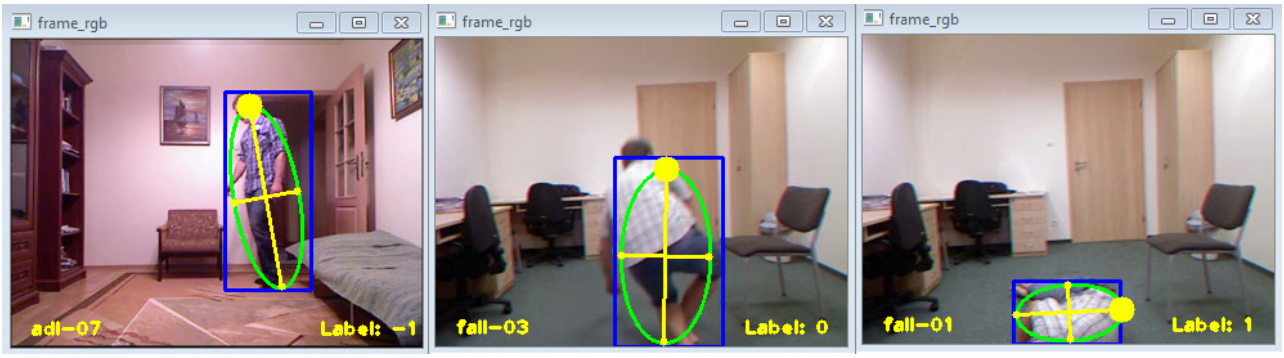


Figure 4. sample of three classes, "–1" means person is not lying, "0" is temporary pose, "1" means person is lying on the ground

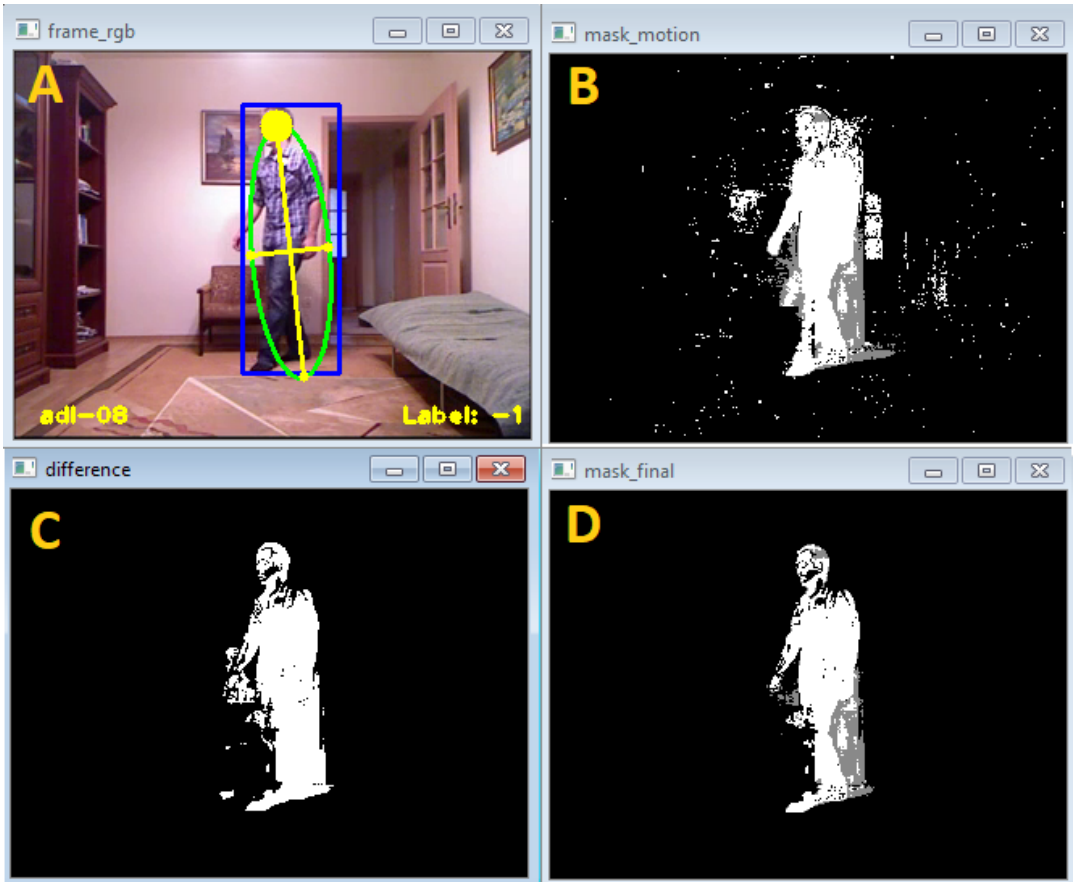


Figure 5. A) output image, B) motion mask, C) Human mask by Frame difference, D) final Human mask

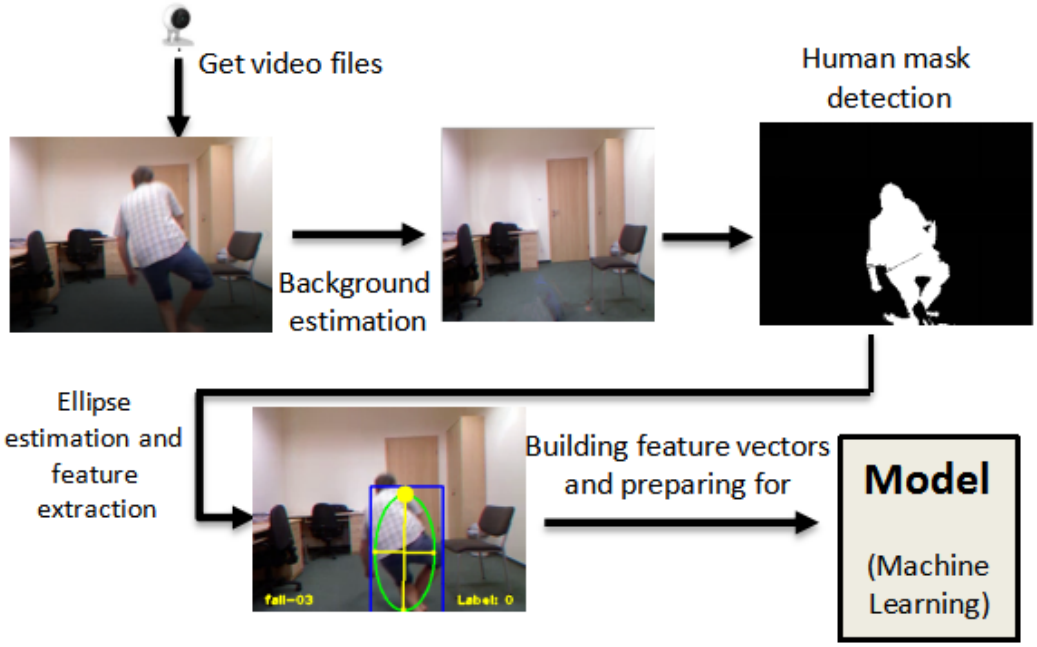


Figure 6. An example of the output of this algorithm

Table 1. Detection results of the proposed algorithm using UR dataset

Method	SVM	KNN	MLP	ENN	Adaboost
Accuracy	70%	65%	69%	54%	81%

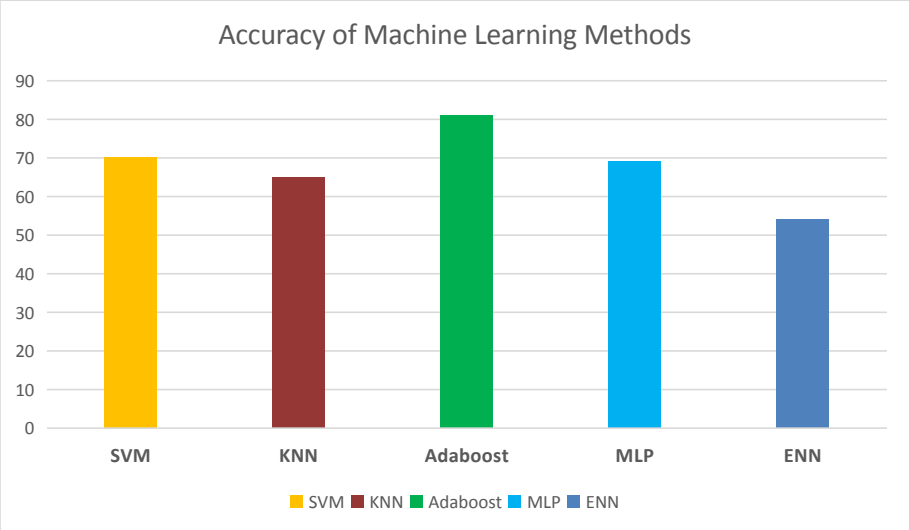


Figure 7. Accuracy of machine learning methods with proposed algorithm for fall detection using UR dataset

5. Conclusions

The algorithm introduced in this paper is a new purpose to fall detection automatically to support a better life for the elderly. The features of the mask extracted from the human play an essential duty in the robustness and effectiveness of human fall detection. Our main focus and purpose in this paper is to combine the features extracted from moving human, which are suitable for real-time fall detection only with images received from the webcam. By combining features such as the coordinates of the human head, various parameters of the estimated ellipse around the human, and the motion features of the human, we have created a very suitable feature vector that demonstrates high accuracy with machine learning algorithms. This type of extraction and combination of features and the use of machine learning models create a new algorithm. Experimental results in Table 1 show that the proposed fall detection algorithm is one of the best fall detection methods in the literature. The advantage of this method over the previous methods is the high speed in real-time image processing, and the reason is the use of easy steps in feature extraction and processing. In addition, there is no need to connect gadgets and devices to the elderly, and it can be used only by placing a camera or webcam in the place of movement. It is considered that the main reason for the success of the method is the strong and effective selection features to distinguish between fall and non-fall activities. The highest accuracy with the proposed algorithm for detecting and combining features and using the Adaboost method is 81%. The results of Table 1 show that this method can also be used in practice. This suggests that the proposed fall detection method can effectively predict the fall.

Conflict of Interest

There is no conflict of interest.

References

[1] Wu, G., 2000. Distinguishing fall activities from normal activities by velocity characteristics. Journal of Biomechanics. 33(11), 1497-1500. DOI: https://doi.org/10.1016/s0021-9290(00)00117-2

[2] Nash, A., 2019. National population projections: 2018-based. Office for National Statistics. https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationprojections

[3] McNicoll, G., 2002. World Population Ageing 1950-2050. Population and development Review, 28(4), 814-816. https://www.un.org/en/development/desa/population/publications/pdf/ageing/WorldPopulationAgeing2019-Highlights.pdf

[4] Leone, A., Diraco, G., Distante, C., et al., 2008. multi-sensor approach for people fall detection in home environment. Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2. https://hal.inria.fr/inria-00326739

[5] Diraco, G., Leone, A., Siciliano, P., 2010. An active vision system for fall detection and posture recognition in elderly healthcare. 2010 Design, Automation & Test in Europe Conference & Exhibition(DATE 2010). pp. 1536-1541. DOI: https://doi.org/10.1109/DATE.2010.5457055

[6] Perner, P., Imiya, A., 2011. Machine learning and data mining in pattern recognition. Springer-Verlag Berlin Heidelberg. pp. 1-264. https://link.springer.com/book/10.1007/978-3-642-23199-5

[7] Chen, Y.T., Lin, Y.Ch., Fang, W.H., 2010. A hybrid human fall detection scheme. 2010 IEEE International Conference on Image Processing. pp. 3485-3488. DOI: https://doi.org/10.1109/ICIP.2010.5650127

[8] Xiang, Y.J., Arora, J.S., Abdel-Malek, K., 2010. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. Structural and Multidisciplinary Optimization. 42(1), 1-23. DOI: http://dx.doi.org/10.1007/s00158-010-0496-8

[9] Gjoreski, H., Lustrek, M., Gams, M. 2011. Accelerometer placement for posture recognition and fall detection. Seventh International Conference on Intelligent Environments. pp. 47-54. DOI: https://doi.org/10.1109/IE.2011.11

[10] Bergen, G., Stevens, M. R., Burns, E. R. 2016. Falls and fall injuries among adults aged 65 years United States. MMWR Morbidity and mortality weekly report. 65(37), 993-998. DOI: https://doi.org/10.15585/mmwr.mm6537a2

[11] Ma, X., Wang, H.B., Xue, B.X., et al., 2014. Depth-based human fall detection via shape features and improved extreme learning machine. IEEE journal of biomedical and health informatics. 18(6), 1915-1922. DOI: https://doi.org/10.1109/JBHI.2014.2304357

[12] Jansen, B., Deklerck, R., 2006. Context aware inactivity recognition for visual fall detection. Pervasive Health Conference and Workshops. pp. 1-4. DOI: https://doi.org/10.1109/PCTHEALTH.2006.361657

[13] Nait-Charif, H., McKenna, S.J., 2004. Activity summarisation and fall detection in a supportive home environment. Proceedings of the 17th International Conference on Pattern Recognition. 4(1), 323-326. DOI: https://doi.org/10.1109/ICPR.2004.1333768

[14] Dai, J.P., Bai, X.L., Yang, Zh.M., et al., 2010. Mobile phone-based pervasive fall detection. Personal and Ubiquitous Computing. 14(7), 633-643. DOI: http://dx.doi.org/10.1007/s00779-010-0292-x

[15] Dubitzky, W., Granzow, M., Berrar, D., 2007. Fundamentals of data mining in genomics and proteomics. Springer Science & Business Media. https://link.springer.com/book/10.1007/978-0-387-47509-7

[16] Hazelhoff, L., Han, J.G., 2008. Video-based fall detection in the home using principal component analysis. International Conference on Advanced Concepts for Intelligent Vision Systems. pp. 298-309. DOI: http://dx.doi.org/10.1007/978-3-540-88458-3_27

[17] Leclercq, S., 2000. In-company same-and low-level falls: From an understanding of such accidents to their prevention. International journal of industrial ergonomics. 25(1), 59-67. DOI: https://doi.org/10.1016/S0169-8141(98)00095-X

[18] Xiang, Y.J., Arora, J.S., Abdel-Malek, K., 2010. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. Structural and Multidisciplinary Optimization. 42(1), 1-23. DOI: https://doi.org/10.1007/s00158-010-0496-8

[19] Pai, Y.Ch., Patton, J., 1997. Center of mass velocity-position predictions for balance control. Journal of Biomechanics. 30(4), 347-354. DOI: https://doi.org/10.1016/S0021-9290(96)00165-0

[20] Popescu, M., Li, Y., Skubic, M., et al., 2008. An acoustic fall detector system that uses sound height information to reduce the false alarm rate. 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 4628-4631. DOI: https://doi.org/10.1109/IEMBS.2008.4650244

[21] Hartikainen, S., Lönnroos, E., Louhivuori, K., 2007. Medication as a risk factor for falls: critical systematic review. The Journals of Gerontology Series A: Biological Sciences and Medical Sciences. 62(10), 1172-1181. DOI: https://doi.org/10.1093/gerona/62.10.1172

[22] Yazdi, M., Bouwmans, T., 2018. New trends on moving object detection in video images captured by a moving camera: A survey. Computer Science Review. 28(1), 157-177. DOI: https://doi.org/10.1016/j.cosrev.2018.03.001

[23] Kwolek, B., Kepski, M., 2014. Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer Methods and Programs in Biomedicine. 117(3), 489-501. DOI: https://doi.org/10.1016/j.cmpb.2014.09.005

[24] Jadhav, M.J.J., 2014. Moving object detection and tracking for video survelliance. International Journal of Engineering Research and General Science. 2(4), 372-378. https://ijergs.org.managewebsiteportal.com/files/documents/MOVING-46.pdf

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances
https://ojs.bilpublishing.com/index.php/aia

ARTICLE

Safety-critical Policy Iteration Algorithm for Control under Model Uncertainty

Navid Moshtaghi Yazdani[1*] Reihaneh Kardehi Moghaddam[1] Mohammad Hasan Olyaei[2]

1. Department of Electrical Engineering, Mashhad branch, Islamic Azad University, Mashhad, Iran
2. Department of Electrical Engineering, Sadjad University of Technology, Mashhad, Iran

ARTICLE INFO

Article history
Received: 18 January 2022
Revised: 31 March 2022
Accepted: 2 April 2022
Published: 11 April 2022

Keywords:
Safe-critical
Optimal controller
Reinforcement learning
Lyapunov
Sum-of-Square

ABSTRACT

Safety is an important aim in designing safe-critical systems. To design such systems, many policy iterative algorithms are introduced to find safe optimal controllers. Due to the fact that in most practical systems, finding accurate information from the system is rather impossible, a new online training method is presented in this paper to perform an iterative reinforcement learning based algorithm using real data instead of identifying system dynamics. Also, in this paper the impact of model uncertainty is examined on control Lyapunov functions (CLF) and control barrier functions (CBF) dynamic limitations. The Sum of Square program is used to iteratively find an optimal safe control solution. The simulation results which are applied on a quarter car model show the efficiency of the proposed method in the fields of optimality and robustness.

1. Introduction

Safety is an integral part and a central requirement for any safe-critical system such as power systems, automatic devices, industrial robots, and chemical reactors. Considering the increasing demand for safe systems in the future generation of industrial systems, and also the importance of an interaction with systems surroundings and uncertainties, there is a real need for the development of safe controllers, which can meet the already-mentioned demand. In the absence or violation of these safety conditions, the system is likely to suffer from some faults, including the system stabilization problem and its simultaneous survival in the given safety system, which lead to the rise of multiple serious challenges to designing controllers. The optimal control design, as well as the safe control design for the feedback state, is discussed separately in the literature review. Developing such safe controllers to optimize the performance of dynamic systems with uncertainties, primarily resulted from lack of safe optimal controllers with uncertainty conditions.

*Corresponding Author:
Navid Moshtaghi Yazdani,
Department of Electrical Engineering, Mashhad branch, Islamic Azad University, Mashhad, Iran;
Email: navid.moshtaghi@ut.ac.ir

DOI: https://doi.org/10.30564/aia.v4i1.4361

1.1 Related Works

The official design for the stabilization of non-linear dynamic systems is often obtained by employing the Control Lyapunov Functions (CLFs). The optimal feedback controllers necessary for general non-linear systems can be designed by solving Hamilton-Jacobi-Bellman equations (HJB), which have been done approximately by through the use of Galerkin method [1] and neural networks method [2-8]. However, due to the lack of robustness and computational infeasibility for online performance, the open-loop form of calculating these solutions seems problematic. Consequently, in this paper the optimal control of constrained systems equipped with penalty functions in the performance function [9]. However, the application of these methods is only limited to linear state constraints.

Today real-time safety in dynamic systems has gained large attention, followed by the introduction of the barrier functions, through which the risk of the system states entering the given non-safety zones can be removed [10-15]. Also, control methods using CLF and CBF have been considered as successful methods to achieve safety-stability control. Some researchers have shown that for the performance of movement tasks (manipulation and locomotion), CLF-based quadratic programs (CLF-QP) with constraints can be solved online [16,17]. They have also combined CBFs with CLF-QP to effectively for the effective management of safety constraints in real time. By the by, an itemized information on the system model is expected for every one of these CLF-based and CBF-based techniques.

Taylor et al. addressed how a minimization method for experimental risk can lead to the uncertainties in CLF and CBF constraints [18,19]. Westernbroek et al. have additionally proposed a reinforcement learning-based method to learn model uncertainty compensation for the input-output linearization control [20]. Learning-based control is also obtained in dynamic systems with high uncertainty in spite of safety constraints [21,22]. Moreover, probabilistic models such as Gaussian process can be used to learn about model uncertainties [23,24]. Using these methods, the comprehensive investigation of the learned model or policy is permitted; however, they can scale inadequately with state dimension and involving them in high-ordered systems won't be simple.

1.2 Contributions and Outline

Prajna et al. introduced a policy iteration algorithm as a way to build the safe optimal controller for a class of certain nonlinear systems [25]. However, due to the difficulty of practically obtaining accurate system information, an online training method is presented in this study to replace identifying system dynamics with an iterative algorithm featured with real data. In this paper, the effect of model uncertainty is, also, investigated on CLF and CBF dynamic constraints. For each of them, the purpose of the RL agent and the policy to be learned will be defined. The Sum-of-Square program is utilized to iteratively discover an optimal safe control solution. Finally, in order for the efficiency of the proposed method to be validated, a simulation example is employed.

The remaining part of the present paper is organized as follows: Section 2 formulates the problem and presents a new safe optimal control framework. Section 3 presents reinforcement learning for optimal safe control under uncertain dynamics, and Section 4 provides the numerical examples to validate the efficiency of the proposed method.

1.3 Notations

The term C^1 denotes the set of all continuous differential functions. Then, P denotes the set of all existing functions in C^1 that are positive, definite and proper. The polynomial $p(x)$ is Sum-of-Squares (SOS) (i.e., $p(x) \in \mathrm{P}^{SOS}$ in which P^{SOS} is a set of SOS polynomials, $p(x) = \sum_1^m p_i^2(x)$ where $p_i(x) \in \mathrm{P}\ i = 1,...,m$). Function $\mathrm{K}: \mathrm{R}^n \to \mathrm{R}^n$ is an extended class K function and $K(0) = 0$. ∇V Alludes to the gradient of the V function: $\mathrm{R}^n \to \mathrm{R}^n$.The Li derivative of function h with respect to f is defined as $L_f V(x) = \frac{\partial h}{\partial x} f(x)$. For any positive integer t1 and t2 where t2≥t1, $\bar{\mathrm{n}}_{t1,t2}(\mathrm{x})$ is the vector of all distinct monic monomial sets $\binom{m+t_2}{t_2} - \binom{m+t_1-1}{t_1-1}$ in $x \in \mathrm{R}^n$ with minimum degree of t_1 and maximum degree of t_2. Moreover, $\mathrm{R}[x]_{t_1,t_2}$ represents a set of all polynomials in $x \in \mathrm{R}^n$ with degrees less than t_2 and greater than t_1.

2. Problem Formulation and Details

In this part, we talk about safety, stability and optimization of the control systems. The initial results of each are also mentioned. Then the formulas of the optimal safe control design will be performed.

2.1 Optimal Control of Dynamical Systems

Consider the following nonlinear system:

$$\dot{x} = f(x) + g(x)u \tag{1}$$

In which $\mathrm{x} \in \mathrm{R}^n$ is the system state vector, $\mathrm{u} \in \mathrm{R}^m$ is the control input vector, $\mathrm{f}: \mathrm{R}^n \to \mathrm{R}^n$ and $\mathrm{g}: \mathrm{R}^n \to \mathrm{R}^{n \times m}$ are both locally Lipschitz continuous with $f(0) = 0$. We expect the system as a stabilizable one.

The main goal of standard optimal control design is to

find a control policy to minimize the predefined performance index over the system trajectories (1) defined as follows:

$$J(x_0,u)=\int_0^{\infty} r(x(t),u(t))dt \tag{2}$$

In relation (2), $r(x,u)=q(x)+u^T Ru$, $q(x)$ and $R(x)$ can be considered as reward function, positive definite function and positive definite matrix, respectively. The reward function $r(x,u)$ is defined such that optimizing (2) guarantees the achievement of control objectives (e.g., minimizing the control effort to achieve the desired transient response) as well as system stability.

The presence of an optimal stabilizing solution is ensured under mild assumptions about the reward function and system dynamics [26].

Assumption 1. Considering system (1), there exists a Lyapunov function $V\in\mathcal{P}$ and a feedback control policy u which satisfies the following inequality:

$$\mathrm{L}(V,u)=-(L_fV(x)+L_gV(x)u)-r(x,u)\geq 0 \quad x\in \mathrm{R}^n \tag{3}$$

The system stability conditions are guaranteed by this assumption, implying that the cost $\forall x_0\in R^n, J(x_0,u)$ is Finite.

Theorem 1. Theorem 10.1.2 [26] considers system (1) with performance function (2), there must be a positive semi-definite function $V^*(x)\in C^1$ satisfying the Hamilton-Jacobi-Belman (HJB) equation as follows:

$\mathrm{H}(V^*)=0$

In which

$$\mathrm{H}(V)=q(x)+L_fV(x)-\frac{1}{4}L_gV(x)R^{-1}(x)(L_gV(x))^T=0,\quad V(0)=0 \tag{4}$$

Therefore, the following feedback control

$$u^*(x)=\frac{1}{2}R^{-1}(x)(L_gV^*)^T(\mathrm{x}) \tag{5}$$

Optimizes the performance index (2) and results in the achievement of asymptotic stability of the equilibrium $x=0$. Also, the optimal value function is given as follows:

$$V^*(x_0)=\min_u J(x_0,u)=J(x_0,u^*),\quad \forall x_0\in \mathrm{R}^n \tag{6}$$

Assumption 1 appears that it is vital to solve the HJB Equation (4) to find an optimal control solution.

Assumption 2: There are proper mappings $V_0:\mathrm{R}^n\to\mathrm{R}$ and $\mathrm{u}:\mathrm{R}^n\to\mathrm{R}^m$, such that $V_0\in\mathrm{R}[x]_{2,2r}\cap\mathrm{P}$ and $\mathrm{L}(V_0,u)$ are SOS.

2.2 About Control Barrier Functions and Its Relation with Safe Control of Dynamical Systems

In a safety-critical system, it is important to prevent its state starting from any initial conditions in X_0 set to enter some special unsafe regions like $X_u\in X$. To design a safe controller, control barrier functions (CBF), inspired by Control Lyapunov Function (CLF), can be employed. Now Equation (1) and the function $h:R^n\to R$ can be considered as follows:

$$\begin{aligned} &h(\mathrm{x})\geq 0,\ \forall x\in \mathrm{X}_0,\\ &h(\mathrm{x})<0,\ \forall x\in \mathrm{X}_u \end{aligned} \tag{7}$$

The following function is also defined as:

$$\mathrm{L}=\{\mathrm{x}\in \mathrm{X}\mid h(\mathrm{x})\geq 0\} \tag{8}$$

Having ZCBF $h(x)$, the admissible control space $S(x)$ is defined as follows:

$$S(x)=\{u\in U\mid L_fh(x)+L_gh(x)u+K_b(h(x))\geq 0\},x\in X \tag{9}$$

The following theorem demonstrates the way a controller is designed using the ZCBF concept to ensure that the forward invariance of the safe set and system stability.

Theorem 2. For $\mathrm{L}\subset R^n$ given in (8) and a ZCBF defined by h in (9), each controller $\mathrm{u}\in\mathrm{S}(\mathrm{x})$ for the system (1) presents a safe set L forward invariant.

The barrier functions for exponential controls are introduced. They are improved in a work by Ams et al. [27,28].

This translates to the r^{th} time-derivative of $h(x)$

$$h^{(r)}(x\,u)=L_f^r h(x)+L_gL_f^{r-1}h(x)u$$

The authors expanded the CBFs having an arbitrary relative degree $r\geq 1$ to $h(x)$ functions. To do so, we define $z=col\left(h(x),L_fh(x),L_f^2h(x),...,L_f{}^{r-1}h(x)\right)$. As well, we assume that u can be selected so that $L^r{}_fh(x)+L_gL_f{}^{r-1}h(x)u=\mu$ for $\mu\in U_\mu\subset\mathrm{R}$ which is a slack input. We have:

$$\begin{aligned} &\dot{z}(x)=f_bz(x)+g_b\mu\\ &h(x)=p_bz(x) \end{aligned}$$

Where, f_b,g_b,p_b are,

$$f_b=\begin{bmatrix}0&1&0&\cdots&0\\0&0&1&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&1\\0&0&0&\cdots&0\end{bmatrix},g_b=\begin{bmatrix}0\\0\\\vdots\\0\\1\end{bmatrix},p_b=\begin{bmatrix}1&0&\cdots&0\end{bmatrix}$$

If a set $\mathrm{L}\subset R^n$ is defined as the super level set for an r-times functions which are continuously differentiable, then h is considered as an exponential control barrier function (ECBF) for the control system (1). Therefore, the acceptable space SE(x) (if $K_\alpha\in\mathrm{R}^r$ exists) is defined as follows,

$$SE = \sup_{u \in U} [A + K_\alpha z(x)] \geq 0$$

Where, $A = L_f^r h(x) + L_g L_f^{r-1} h(x) u$

As Assumption 3, the admissible control space S (x) can be considered not empty.

3. Reinforcement Learning for Safe Optimal Control under Uncertain Dynamics

In this part, the potential inconformity between the model and the plant elements is examined, while there is paucity of accurate knowledge of the true plant vector fields g , f . Moreover, its effects on the dynamics of CLF and CBF will be examined.

Allow the substantial model utilized in the controller to be characterized as follows:

$$\dot{x} = \hat{f}(x) + \hat{g}(x) u \tag{10}$$

Assume that the vectors $\hat{f}: R^n \to R^n, \hat{g}: R^n \to R^n$ are Lipschitz continuous and Where,

Problem 1. (Safe Optimal Control under uncertainty dynamics): Find a controller that solves the following equation:

$$\begin{aligned} u^* &= \arg\min \int_\Omega V dx + k_\delta \delta^2 \\ \text{st.} \quad & H(V) \leq \delta \\ & \hat{A} + K_\alpha z(x) \geq 0 \end{aligned} \tag{11}$$

In relation (11), Ω is an area in which the system performance is expected to be improved, $k_\delta > 0$ is the design parameter that acts as a trade-off between the system aggressiveness toward performance and safety, and δ is the Stability relaxation factor. Note that δ can be defined as the Aspiration level for a performance that shows the level of performance sacrificed as a result of failure in satisfying safety and performance. However, this parameter is minimized to achieve the highest possible performance.

First, the relaxed optimal control problem for system (1) with performance (2) is examined as follows:

$$\begin{aligned} &\min_V \int_\Omega V(x) dx \\ &H(V) \leq 0 \\ &V \in P \end{aligned} \tag{12}$$

In which $H(V)$ is defined by Equation (4) and $\Omega \subset R^n$ is an ideal compact set containing the origin [29]. Problem 1 actually solves a relaxed version of HJB (4) in which the HJB equation is relaxed with the HJB inequality. Ames et al. have shown that the solution of problem 1 is unique and if V^* is a solution for (9), then

$$u^{opt}(x) = -\frac{1}{2} R^{-1}(x) (L_g V^*)^T(x) \tag{13}$$

The stability of the system is guaranteed and V^* plays the role of an upper bound or an overestimate for the actual cost. The superscript opt is used here to indicate that u^{opt} is a performance-oriented controller. However, with a safe control policy u^i, V^i and δ^i are determined to tackle the following optimization subject.

This control policy doesn't confirm system safety.

$$\begin{aligned} &\min_{V^i, \delta^i} \int_\Omega V^i dx + k_\delta \delta_i^2 \\ &L(V^i, u^i) = -L_f V^i - -L_g V^i u^i - r(x, u^i) \geq \delta_i \quad \forall x \in R^n \\ &V^{i-1} - V^i \geq 0 \end{aligned} \tag{14}$$

In SOS framework, this optimization problem is defined as follows:

$$\begin{aligned} &\min_{V^i, h^i} \int_\Omega V^i dx + k_\delta \delta_i^2 \\ &L(V^i, u^i) + \delta_i \text{ is SOS} \quad \forall x \in R^n \\ &V^{i-1} - V^i \text{ is SOS} \end{aligned}$$

Based on Assumption 1, there is a safe control policy u. Now we can write the control policy as $u = u^{opt} + u^{safe}$ in which $u^{opt} = \frac{1}{2} R^{-1}(x) (L_g V^*)^T(x)$ is a part of the controller that is applied to optimize performance regardless of safety and u^{safe} has been added to u^{opt} in order to guarantee safety.

3.1 Deriving u^{opt} under Uncertainty Situation

Lemma 1: Consider system (10). Suppose that u is a global safe control policy and $V_{i-1} \in P$ is also existed. Then the system (11) is feed forward.

Proof: According to the assumptions 1 and 2, $V_{i-1} \in P$. Then by sum of squares, we conclude that

$$\begin{aligned} &V_{i-1}^T \left(f + g u^{opt} + g u^{safe} \right) \leq -u^{opt^T} R u^{opt} - 2 u^{opt^T} R u^{safe} \\ &= -\left| u^{opt} + u^{safe} \right|_R^2 + \left| u^{safe} \right|_R^2 \leq \left| u^{safe} \right|_R^2 \leq \left| u^{safe} \right|_R^2 + V_{i-1} \end{aligned} \tag{15}$$

According to Result 2.11 [24], system (11) is feed forward:

There is a fixed matrix $W_i \in \mathbb{R}^{m \times n_t}$ in which $m_t = \binom{m+t}{t} - 1$ such that $u^{opt} = W_i \vec{n}_{1,t}(x)$. It is also assumed that there is a fixed vector $p \in R^{n_t}$ in which $m_t = \binom{m+t}{t} - m - 1$ so that $V = p^T \vec{n}_{2,2t}(x)$. Then, the following terms can be defined along with the solutions of the system (11):

$$\begin{aligned}\dot{V} &= (L_f V(x) + L_g V(x) u^{opt}) + L_g V(x) u^{safe} \\ &= -r\left(x, u^{opt}\right) - \mathrm{L}\left(V, u^{opt}\right) + L_g V(x) u^{safe} \\ &= -r\left(x, u^{opt}\right) - \mathrm{L}\left(V, u^{opt}\right) + \left(R^{-1} g^T \nabla V\right)^T R u^{safe}\end{aligned}$$

Note that two terms $\mathrm{L}\left(V, u^{opt}\right)$ and $R^{-1} g^T \nabla V$, depend on $\hat{f}$ and $\hat{g}$. Since there is uncertainty in these terms, we should solve them without recognizing $\hat{f}$ and $\hat{g}$.

For a similar abovementioned pair (V, u^{opt}), we can find a fixed vector $b_p \in \mathrm{R}^{n_{2t}}$, in which $m_{2t} = \binom{n+2t}{2t} - m - 1$ and $W_p \in \mathrm{R}^{m \times n_t}$ is a fixed matrix, such that

$$\mathrm{L}\left(V, u^{opt}\right) = b_p^T \vec{n}_{2,2t}(x) \tag{16}$$

$$-\frac{1}{2} R^{-1} g^T \nabla V = W_p \vec{n}_{1,t}(x) \tag{17}$$

Therefore, $\mathrm{L}\left(V, u^{opt}\right)$ and $R^{-1} g^T \nabla V$ are calculated to find b_p and W_p By substituting Equations (16) and (17) in Equation (15), we have:

$$\dot{V} = -r(x, u^{opt}) - b_p^T \vec{n}_{2,2t}(x) - 2\vec{n}_{1,t}^T(x) W_p^T \tag{18}$$

By integrating (18) into the time interval [t, t + δt]:

$$\begin{aligned}&p^T\left[\vec{n}_{2,2t}(x(t)) - \vec{n}_{2,2t}(x(t+\delta t))\right] = \\ &\int_t^{t+\delta t}\left(r(x, u_i) + b_p^T \vec{n}_{2,2t}(x) + 2\vec{n}_{1,d}^T(x) W_p^T R u^{safe}\right) dt\end{aligned} \tag{19}$$

Now, b_p and W_p can be calculated without having accurate information about $\hat{f}$ and $\hat{g}$ by using real online data.

1) Initial value:

Find the pair (V_0, u) that satisfies Assumption 1. Consider a fixed vector p_0 such that $V_0 = p_0^T \vec{m}_{2,2t}(x)$, and $i = 1$.

2) Online data collection:

First, apply $u = u^{opt} + u^{safe}$ to the system and then find an optimal solution (p_i, W_{i+1}) for the following SOS program.

$$\begin{aligned}&\min_{p, W_p} \int_\Omega \vec{n}_{2,2t}(x) dx^T p + K_\delta \delta_i^2 \\ &b_p^T \vec{n}_{2,2t}(x) \, isSOS \\ &(p_{i-1} - p_i)^T \vec{n}_{2,2t}(x) \, isSOS\end{aligned} \tag{20}$$

So, we have $V^i = p_i^T \vec{n}_{2,2t}(x)$. Then, we can derive the value of $u^{opt} = W_p \vec{n}_{1,t}(x)$ and proceed to step 2) where $i \leftarrow i + 1$.

3.2 Reinforcement Learning for CBFs

The control rule for the computed input-output linearization has the following form based on the $\hat{f}$ and $\hat{g}$:

$$\hat{u}(\mathrm{x}, \mu) = \hat{u}^*(\mathrm{x}) + \left(\mathrm{L}_{\hat{g}} \mathrm{L}_{\hat{f}} h(\mathrm{x})\right)^{-1} \mu \tag{21}$$

In which μ is also an auxiliary input.

Under the uncertainty situation, it can be written:

$$\begin{aligned}\widehat{\boldsymbol{A}} &= \boldsymbol{L}_{\hat{\boldsymbol{f}}}^{\boldsymbol{r}} \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{L}_{\hat{\boldsymbol{g}}} \boldsymbol{L}_{\hat{\boldsymbol{f}}}^{\boldsymbol{r-1}} \boldsymbol{h}(\boldsymbol{x}) \widehat{\boldsymbol{u}} \\ \boldsymbol{A} &= \widehat{\boldsymbol{A}} + \boldsymbol{\alpha} + \boldsymbol{\beta} \boldsymbol{\mu}\end{aligned} \tag{22}$$

Where α and β are

$$\begin{aligned}\alpha &= L_f^r h(x) - L_g L_f^{r-1} h(x)\left(L_{\hat{f}} L_{\hat{f}}^{r-1} h(x)\right)^{-1} L_{\hat{f}}^r h(x) \\ \beta &= L_g L_f^{r-1} h(x)\left(L_{\hat{g}} L_{\hat{f}}^{r-1} h(x)\right)^{-1}\end{aligned}$$

Terms obtained from the mismatch existing between model and plant. It should also be noted that if α, β are zero, we have the same equation as (22).

Using an estimator made of A that in the form $A = \widehat{A} + \alpha + \beta\mu$.

RL's goal is to learn α, β policies so that $\widehat{A}$ is close to A as much as possible. Thereby, using RL, the uncertainty terms for CBF can be estimated. Therefore, there is a need for designing the reward function to minimize policy estimation errors. Therefore, it can be defined as follows:

$$l = A - \hat{A}$$

The RL factor embraces a policy that considers the uncertainty terms in CBF, which are summed with the SOS constraints as they are extracted from the nominal model, resulting in accurate estimates. One can consider the focal RL problem with the considered reward for a given state x as the summation of the negative objective functions plus an arbitrary penalty (s) selected by the user

$$rl(x, \theta) = -\sum_{i=1}^{b} w_i l_{i,\theta} - s \tag{23}$$

Where b is the number of CBFs. One can solve RL, using common algorithms.

4. Applications

The reason of this part is to demonstrate that our proposed system can make possible the critical safe control, even in the presence of uncertain conditions. Two simulation examples are presented in this section in order to approve the efficiency of the proposed model.

Example 1:

Consider the car quarter suspension model shown in Figure 1. Its non-linear dynamic is defined as follows. However, it is worth mentioning that while the training experiences or the simulations are operating, the car quarter suspension model is assumed to be under the proper dynamics (given its uncertainties) [30].

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{1}{M_b}\left[k_a(x_1 - x_3) + k_n(x_1 - x_3)^3 + c_a(x_2 - x_4) + u\right] \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{1}{M_{us}}\left[k_a(x_1 - x_3) + k_n(x_1 - x_3)^3 + c_a(x_2 - x_4) + k_t x_3 - u\right]\end{aligned} \tag{24}$$

Where, x_1, x_2, and M_b are the car position, velocity, and its mass, respectively. x_3, x_4, and M_{us} are also the wheel position, velocity, and their total mass. K_t, K_a, K_n, and C_a shows the tire hardness, the system of linear pendency, the non-linear suspension hardness, and the damping rate of the pendency system, respectively.

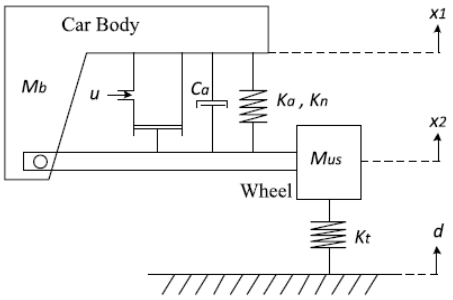


Figure 1. Quarter car model

The uncertainty for the significant model in this experiment is introduced by weighing all the components with a weighing coefficient of 2. During the training (process) of the RL agent, we only know the nominal model.

Let, $M_b \in [250,350]$, $M_{us} \in [55,65]$, $c_a \in [450,550]$, $k_a \in [7500,8500]$, $k_n \in [750,850]$, $k_t \in [90000,10000]$. Then, it can be easily observed that the system establishment has been done in a global level asymptotically, with an absence of input control. The purpose of the proposed method is to design an active suspension control system which lessens the performance index, while retains the global asymptote stability, simultaneously. As well, reducing the disorder effects in the set $\Omega = \{x \mid x \in R^4 \; and \; |x_1| \le 0.03, |x_2| \le 5, |x_3| \le 0.03, |x_4| \le 5\}$ can improve the system performance.

The reinforcement learning factor is taught using a Deep Deterministic Policy Gradient algorithm (DDPG, Silver et al. [31]). The 4 observed state variables, and the CBF component of the simulation constitute the inputs for the actor neural network. The output dimension is equal to which corresponds to 4×1 α_θ^B, and 1×1 β_θ^B.

There exist hidden layers as wide as 200 and 300 in both the actor and the critic neural networks in example 1. This agent is trained by simulation in the interval between $t = 0$, and $t = 80$.

A time step of Ts = 1 is employed (in this regard). The simulations have been carried out on a 6-core laptop with Intel Core™ i7-9400 (2.7 GHz) processor and 4 GB RAM.

Use SOSTOOLS to obtain an initial cost function, V0 for the simulated system having non-determined parameters [32].

Then, we apply the proposed method in which u1=0. The primary condition has been selected randomly. To do the training, we apply the noise from $t = 0$ to $t = 80$ till the convergence is obtained after 8 repetitions.

The obtained control policy is as follows,

$$\begin{aligned} u_8 = {} & -1.76x_1^3 - 5.33x_1^2x_2 + 7.7x_1x_3 + 3.22x_1x_3x_4 - 12.1x_1^2 + 4.43x_1x_2^2 + 0.87x_1x_2^2x_3 \\ & + 0.594x_1x_2x_4 - 4.61x_1x_2 - 6.3x_1x_3^2 - 6.19x_1x_2^2x_3 - 0.174x_1x_4^2 - 2.81\times10^8 x_1x_4 \\ & - 18.1x_1 - 0.73x_2^3 + 0.006x_2^2x_3 + 2.26x_2^2x_4 - 4.07x_2x_3^2 + 1.71x_2x_3x_4 - 4.55x_2x_3 \\ & - 1.35x_2x_4^2 - 4.94x_2x_4 - 2.8x_3^2x_4 + 4.47x_3^2 + 0.241x_3x_4^2 \\ & + 2.62\times10^8 x_3x_4 + 11.1x_3 - 11.62x_3^2 + 6.39x_3^3 + 0.33x_4^3 + 4.61x_4^2 + 10.4x_4 \end{aligned} \quad (25)$$

To test the trained controller, we choose the road disorder as a single-impact as follows,

$$\begin{cases} 0.003(2-\cos(2\pi t)) & t = 60 \\ 0 & otherwise \end{cases} \quad (26)$$

In addition, as an indication of a car carrying a load, an overweight of 260 kg is applied to the vehicle assembly.

So that, the departure of position is relative to the origin. The proposed control policy performance is compared to the primary system performance without any control, as shown in Figure 2. In Figure 3, these two performances of the costs are compared by the constraint wheel position, wheel velocity when they are zero. As can be seen, V_8 has been reduced significantly compared to V_0.

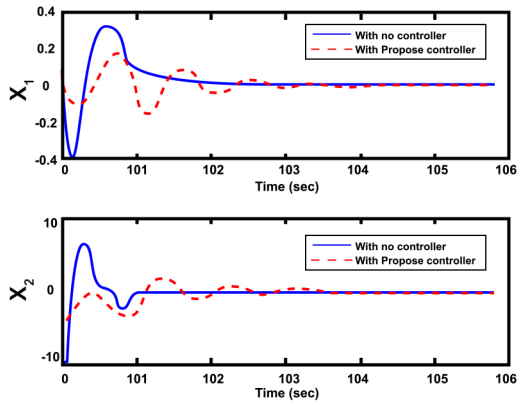


Figure 2. Comparison of performance car position and car velocity

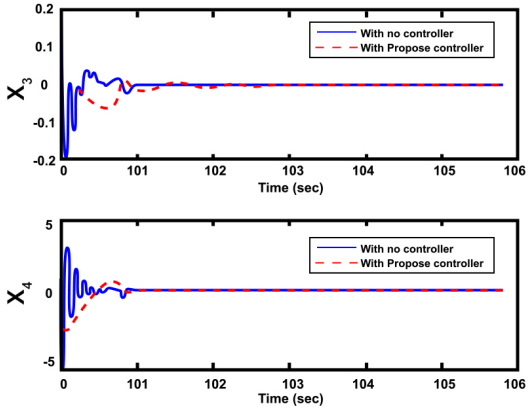


Figure 3. Comparison of performance wheel position and wheel velocity

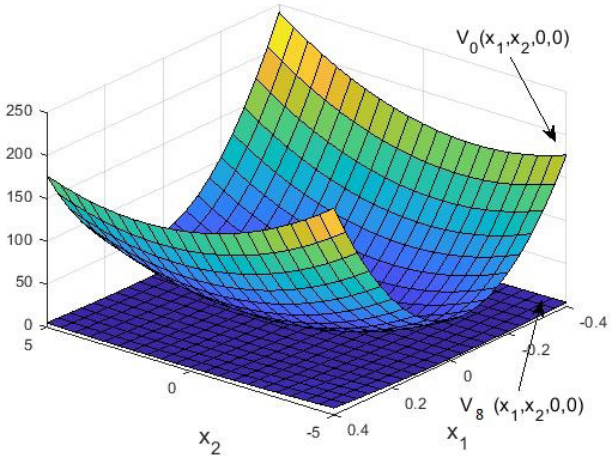


Figure 4. Comparison of learned value functions

Example 2:

Now consider the following system equations:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}x_1^2+x_1x_2^2-x_1x_2\\ 2x_1-x_2\end{bmatrix}+\begin{bmatrix}0 & \alpha_1\\ \alpha_2 & \alpha_1\end{bmatrix}u \tag{27}$$

In which $\alpha_1,\alpha_2\in[0.25,1]$ are uncertain parameters, and $x=[x_1,\ x_2]$ and u are mode and system control, respectively. The unsafe space was coded with a polynomial inequality $X_u=\{x\in R^2\,|\,bf_i(x)<0, i=1,2,3\}$

With the following details:

$$bf_1=-0.5+(x_1+1)^2+(x_2+2)^2<0$$
$$bf_2=-0.5+(x_1+1.5)^2+(x_2-1.5)^2<0$$
$$bf_3=-0.5+(x_1-1.5)^2+(x_2-1)^2<0$$

Using SOS strategies, the system (27) can stabilize, at the source level globally and asymptotically by the following robust control policy [33].

$$u_1^{opt}=\begin{bmatrix}u_1^1\\ u_2^1\end{bmatrix}=\begin{bmatrix}1.192x_1+3.568x_2\\ 1.7x_1-2.905x_2\end{bmatrix} \tag{28}$$

However, the optimality of the closed-loop system has not been fully addressed.

The primary goal of the control is to find more improved safeguard policies under uncertainty using the iterative safeguard policy algorithm. Then, with the help of solving the feasibility study and SOS-TOOLS, we will reach Equation (29) [34]:

$$\mathrm{L}\left(V,u_1^{opt}\right)\quad isSOS,\quad \forall\alpha_1,\alpha_2\in[0.25,1] \tag{29}$$

The V function is obtained as follows:

$$\mathrm{V}_1=7.6626\mathrm{x}_1^2-4.264\mathrm{x}_1\mathrm{x}_2+6.5588\mathrm{x}_2^2-0.1142\mathrm{x}_1^3+1.7303\mathrm{x}_1^2\mathrm{x}_2$$
$$-1.0845\mathrm{x}_1\mathrm{x}_2^2 3.4848\mathrm{x}_1^4-0.361\mathrm{x}_1^3\mathrm{x}_2+4.6522\mathrm{x}_1^2\mathrm{x}_2^2+1.9459\mathrm{x}_2^4$$

If we put α_1=0.5 and α_2=0.5 the initial condition is arbitrarily set to $x_1(0)$=1 and $x_2(0)$=-1.

$$u_1^6=-0.04x_1^3-0.67x_1^2x_2-0.0747x_1^2+0.0469x_1x_2-$$
$$0.986x_1-0.067x_2^3-2.698x_2$$
$$\mathrm{u}_2^6=-0.067\mathrm{x}_1^3-0.09\mathrm{x}_1^2\mathrm{x}_2-0.201\mathrm{x}_1^2+0.025\mathrm{x}_1\mathrm{x}_2^2- \tag{30}$$
$$0.187\mathrm{x}_1\mathrm{x}_2-1.436\mathrm{x}_1-0.1396\mathrm{x}_2^3-0.345\mathrm{x}_2^2-2.27\mathrm{x}_2$$

The V function is as follows:

$$V_6=1.4878x_1^2+0.8709x_1x_2+4.4963x_2^2+0.0131x_1^3+$$
$$0.2491x_1^2x_2-0.0782x_1x_2^2+0.0639x_2^3+0.0012x_1^3x_2+$$
$$0.0111x_1^2x_2^2-0.0123x_1x_2^3+0.0314x_2^4$$

The indefinite cost function and the initial cost function are compared in Figure 5.

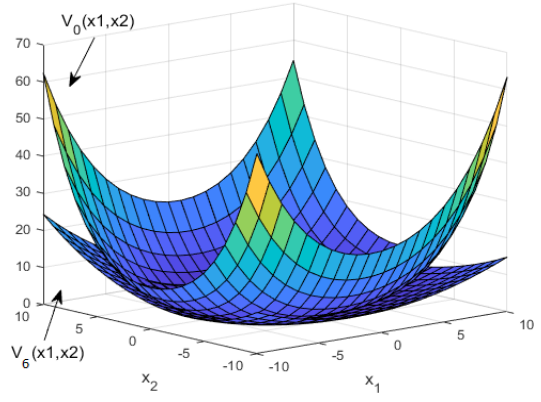


Figure 5. Comparison of learned value functions

Both operator and critical neural networks in example 2 have hidden layers with a width of 100 and 200. The training environment using the learning environment was the same as the previous example, the proposed method was learning took 2 seconds per episode. Control policy Obtained after 5 episodes.

In addition, the safe set is equal to:

$$\ell=\{x\in\mathrm{R}^2\,|\,h(x)\geq 0\}$$

In which:

$$h(x)=0.452-0.0023x_1^2-0.0382x_1-0.014x_2-0.0067x_1x_2-0.0077x_2^2 \tag{31}$$

Note that it is necessary for the safe set to be a member of the complementary set of the unsafe set, as well as being invariable in a way that it never leaves the set in the future. The safe set is obtained using CBF $h\ (x)$. Be attention that barrier certificate is bounded to a second-order polynomial. In Figure 6, the estimated safe sets for both the initial control policy and the optimal control policy are shown.

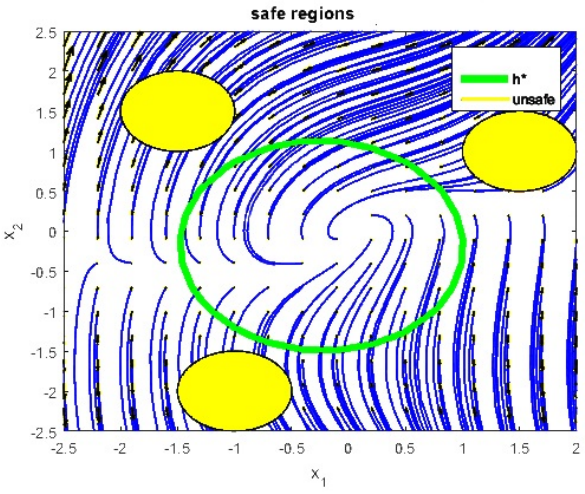


Figure 6. Safe area estimation by the proposed optimal safe controller despite the uncertainty

5. Conclusions

A safe optimization is proposed for the control of dynamics systems under model uncertainty. In order for the performance and safety to be guaranteed, a Hamilton-Jacobi-Bellman (HJB) inequality replaces the HJB equality; besides, a safe policy iteration algorithm is presented certifying the safety of the improved policy and finding a value function corresponding to it. Also, the RL factor was also presented in the proposed method to reduce model uncertainty. The effectiveness of the proposed method is illustrated through two simulation examples.

Conflict of Interest

There is no conflict of interest.

References

[1] Beard, R.W., Saridis, G.N., Wen, J.T., 1997. Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation, Automatica. 33(12), 2159-2177.
DOI: https://doi.org/10.1016/S0005-1098(97)00128-3

[2] Vamvoudakis, K.G., Lewis, F.L., 2010. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem, Automatica. 46(5), 878-888.
DOI: https://doi.org/10.1109/IJCNN.2009.5178586

[3] Lewis, F.L., Vamvoudakis, K.G., 2011. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. IEEE Transactions on Systems. 41(1), 14-25. http://www.derongliu.org/adp/adp-cdrom/Vamvoudakis2011.pdf

[4] Kiumarsi, B., Lewis, F.L., 2015. Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems, IEEE Transactions on Neural Networks and Learning Systems. 26(1), 140-151.
DOI: https://doi.org/10.1109/TNNLS.2014.2358227

[5] Modares, H., Lewis, F.L., Naghibi-Sistani, M.B., 2014. Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems, Automatica. 50(1), 193-202.
DOI: https://doi.org/10.1016/j.automatica.2013.09.043

[6] Wang, D., Liu, D., Zhang, Y., et al., 2018. Neural network robust tracking control with adaptive critic framework for uncertain nonlinear systems, Neural Networks. 97(1), 11-18.
DOI: https://doi.org/10.1016/j.neunet.2017.09.005

[7] Bhasin, S., Kamalapurkar, R., Johnson, M., et al., 2013. A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems, Automatica. 49(1), 82-92. https://ncr.mae.ufl.edu/papers/auto13.pdf

[8] Gao, W., Jiang, Z., 2018. Learning-based adaptive optimal trackingcontrol of strict-feedback nonlinear systems. IEEE Transactions on Neural Networks and Learning Systems. 29(1), 2614-2624. https://ieeexplore.ieee.org/ielaam/5962385/8360119/8100742-aam.pdf

[9] Abu-Khalaf, M., Lewis, F.L., 2004. Nearly optimal state feedback control of constrained nonlinear systems using a neural networks hjb approach. Annual Reviews in Control. 28(2), 239-251.
DOI: http://dx.doi.org/10.1016/j.arcontrol.2004.07.002

[10] Ames, A.D., Grizzle, J.W., Tabuada, P., 2014. Control barrier function based quadratic programs with application to adaptive cruise control. 53rd IEEE Conference on Decision and Control. pp. 6271-6278.
DOI: https://doi.org/10.1109/CDC.2014.7040372

[11] Ames, A.D., Xu, X., Grizzle, J.W., et al., 2017. Control barrier function based quadratic programs for safety critical systems. IEEE Transactions on Automatic Control. 62(8), 3861-3876.
DOI: https://doi.org/10.1109/TAC.2016.2638961

[12] Nguyen, Q., Sreenath, K., 2016. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. 2016 American Control Conference (ACC). pp. 322-328.
DOI: https://doi.org/10.1109/ACC.2016.7524935

[13] Romdlony, M.Z., Jayawardhana, B., 2014. Uniting control Lyapunov and control barrier functions. 53rd IEEE Conference on Decision and Control. pp. 2293-2298.

DOI: https://doi.org/10.1109/CDC.2014.7039737

[14] Xu, X., Tabuada, P., Grizzle, J.W., et al., 2015. Robustness of control barrier functions for safety critical control. Analysis and Design of Hybrid Systems ADHS IFAC Papers Online. 48(27), 54-61. DOI: https://doi.org/10.1016/j.ifacol.2015.11.152

[15] Prajna, S., Rantzer, A., 2005. On the necessity of barrier certificates. 16thIFAC World Congress IFAC Proceedings. 38(1), 526-531. DOI: https://doi.org/10.3182/20050703-6-CZ-1902.00743

[16] Ames, A.D., Powell, M., 2013. Towards the unification of locomotion and manipulation through control lyapunov nctions and quadratic programs. In Control of Cyber Physical Systems. pp. 219-240. http://ames.caltech.edu/unify_ames_powell.pdf

[17] Galloway, K., Sreenath, K., Ames, A.D., et al., 2015. Torque saturation in bipedal robotic walking through control Lyapunov function-based quadratic programs. pp. 323-332. DOI: https://doi.org/10.1109/ACCESS.2015.2419630

[18] Taylor, A.J., Dorobantu, V.D., Le, H.M., et al., 2019. Episodic learning with control lyapunov functions for uncertain robotic systems. ArXiv preprint. https://arxiv.org/abs/1903

[19] Taylor, A.J., Singletary, A., Yue, Y., et al., 2019. Learning for safety-critical control with control barrier functions. ArXiv preprint. https://arxiv.org/abs/1912.10099

[20] Westenbroek, T., Fridovich-Keil, D., Mazumdar, E., et al., 2019. Feedback linearization for unknown systems via reinforcement learning. ArXiv preprint. https://arxiv.org/abs/1910.13272

[21] Hwangbo, J., Lee, J., Dosovitskiy, A., et al., 2019. Learning agile and dynamic motor skills for legged robots. Science Robotics. 4(26), 58-72. https://arxiv.org/abs/1901.08652

[22] Levine, S., Finn, C., Darrell, T., et al., 2016. End-toend training of deep visuomotor policies. Learning Research. 17(1), 1532-4435. https://arxiv.org/abs/1504.00702

[23] Bansal, S., Calandra, R., Xiao, T., et al., 2017. Goal-driven dynamics learning via Bayesian optimization. 56th Annual Conference on Decision and Control (CDC). pp. 5168-5173. DOI: https://doi.org/10.1109/CDC.2017.8264425

[24] Fisac, J.F., Akametalu, A.K., Zeilinger, M.N., et al., 2019. A general safety framework for learning-based control in uncertain robotic systems. IEEE Transactions on Automatic Control. 64(7), 2737-2752. DOI: https://doi.org/10.1109/TAC.2018.2876389

[25] Prajna, S., Jadbabaie, A., 2004. Safety verification of hybrid systems using barrier certificates. In International Workshop on Hybrid Systems: Computation and Control. Springer. 2993(1), 477-492. https://viterbi-web.usc.edu/~jdeshmuk/teaching/cs699-fm-for-cps/Papers/A5.pdf

[26] Yazdani, N.M., Moghaddam, R.K., Kiumarsi, B., et al., 2020. A Safety-Certified Policy Iteration Algorithm for Control of Constrained Nonlinear Systems. IEEE Control Systems Letters. 4(3), 686-691. DOI: https://doi.org/10.1109/LCSYS.2020.2990632

[27] Lewis, F.L., Vrabie, D., Syrmos, V.L., 2012. Optimal control, 3rd Edition. John Wiley & Sons.

[28] Wang, L., Ames, A., Egerstedt, M., 2016. Safety barrier certificates for heterogeneous multi-robot systems. 2016 American Control Conference (ACC). pp. 5213-5218. DOI: https://doi.org/10.1109/ACC.2016.7526486

[29] Ames, A.D., Coogan, S., Egerstedt, M., et al., 2019. Control barrier functions: Theory and applications. In Proc 2019 European Control Conference. https://arxiv.org/abs/1903.11199

[30] Jiang, Y., Jiang, Z., 2015. Global adaptive dynamic programming for continuous-time nonlinear systems. IEEE Transactions on Automatic Control. 60(1), 2917-2929. DOI: https://doi.org/10.1109/TAC.2015.2414811

[31] Gaspar, P., Szaszi, I., Bokor, J., 2003. Active suspension design using linear parameter varying control. International Journal of Vehicle Autonomous Systems. 1(2), 206-221. DOI: https://doi.org/10.1016/S1474-6670(17)30403-2

[32] Silver, D., Lever, G., Heess, N., et al., 2014. Deterministic policy gradient algorithms. International conference on machine learning. pp. 387-395. http://proceedings.mlr.press/v32/silver14.pdf

[33] Papachristodoulou, A., Anderson, J., Valmorbida, G., et al., 2013. SOSTOOLS: Sum of squares optimization toolbox for MATLAB. Control and Dynamical Systems, California Institute of Technology, Pasadena. http://arxiv.org/abs/1310.4716

[34] Xu, J., Xie, L., Wang, Y., 2009. Simultaneous stabilization and robust control of polynomial nonlinear systems using SOS techniques. IEEE Transactions on Automatic Control. 54(8), 1892-1897. DOI: https://doi.org/10.1109/TAC.2009.2022108

BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances
https://ojs.bilpublishing.com/index.php/aia

ARTICLE

Efficient Parallel Processing of k-Nearest Neighbor Queries by Using a Centroid-based and Hierarchical Clustering Algorithm

Elaheh Gavagsaz*

Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

Article history
Received: 27 April 2022
Revised: 19 May 2022
Accepted: 20 May 2022
Published: 26 May 2022

Keywords:
Classification
k-Nearest Neighbor
Big data
Clustering
Parallel processing

ABSTRACT

The k-Nearest Neighbor method is one of the most popular techniques for both classification and regression purposes. Because of its operation, the application of this classification may be limited to problems with a certain number of instances, particularly, when run time is a consideration. However, the classification of large amounts of data has become a fundamental task in many real-world applications. It is logical to scale the k-Nearest Neighbor method to large scale datasets. This paper proposes a new k-Nearest Neighbor classification method (KNN-CCL) which uses a parallel centroid-based and hierarchical clustering algorithm to separate the sample of training dataset into multiple parts. The introduced clustering algorithm uses four stages of successive refinements and generates high quality clusters. The k-Nearest Neighbor approach subsequently makes use of them to predict the test datasets. Finally, sets of experiments are conducted on the UCI datasets. The experimental results confirm that the proposed k-Nearest Neighbor classification method performs well with regard to classification accuracy and performance.

1. Introduction

Due to developments in technology, information gathering has become an ongoing and relatively economical activity. That has led to such an exponential increase in the available data rate. Social networks, sensor networks, mobile phones, and tablets are examples of applications that generate tons of data every day [1-3]. The large volume of data can be useful if the correct knowledge extraction methods could leverage it. There is a major challenge for researchers and industry that standard machine learning methods can not address the volume, diversity, and complexity of the vast data collection [4]. Hence, the current learning methods need to be improved and scaled to leverage such a volume of data.

The k-Nearest Neighbor method is one of the most popular techniques for both classification and regression purposes [5]. Because of the simplicity, clarity, and high performance of the k-Nearest Neighbor algorithm, it is one of the ten most popular data mining methods [6]. It is also a

*Corresponding Author:
Author] Elaheh Gavagsaz,
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran;
Email: elaheh.gavagsaz@srbiau.ac.ir; egavagsaz@yahoo.com

DOI: https://doi.org/10.30564/aia.v4i1.4668

non- parametric lazy learning technique. Non-parametric implies that the k-Nearest Neighbor approach makes no assumptions about data distribution. Lazy learning represents that this technique prevents model generation before a query is generated, as opposed to other methods of creating a model based on the training data. Therefore, the method requires the storage of all the training data. Next, it selects the k nearest training data for each test data item. Then, it measures the distance or similarity of all training data and every element of test data. This pairwise calculation must be repeated against the entire training data for all of the test data items [7,8]. The approach is impractical in big data. Hence, a parallel and distributed computing environment must be used to produce results in a reasonable period of time.

Recent programming models provide ideal environments to cope with these problems by providing all the computational and memory resources we need. MapReduce framework [9,10] is a programming model for distributed computations and large-scale data processing on clusters. Its architecture offers the appropriate scalability and fault tolerance. Hadoop [11] is one of the first implementations of this framework and is also an open-source implementation. This is an effective method for handling data-intensive applications based on data locality [12,13]. This technology is commonly used in diverse fields such as web search, log analysis and data mining. This, with its unquestionable developments, has some limits. One of the major disadvantages of Hadoop is the retrieval of data from a distributed file system. Therefore, it is inefficient for applications like iterative algorithms or multi-step of data.

Spark [14] was designed to overcome the limitations of the Hadoop. It is also an open-source of the MapReduce framework providing in-memory computation. Spark introduces Resilient Distributed Datasets (RDDs) that are read-only datasets with data items distributed over the computing nodes. RDDs are creating an appropriate kind of distributed shared memory to implement iterative algorithms. This in-memory computation results in a significant improvement in efficiency. The most significant use of in-memory data is in machine learning scenarios [15]. Spark has been highlighted for its ease of use as an effective tool and has been used in many scenarios [16-22].

This paper proposes a new k-Nearest Neighbor classification method which uses a parallel centroid-based and hierarchical clustering algorithm (KNN-CCL) to reduce and obtain the appropriate training dataset. This method is a two-phases approach based on MapReduce, implemented within the Spark framework. In the first phase, a parallel centroid-based clustering algorithm has been used to separate the sample of the training dataset into multiple clusters. In the second phase, the cluster nearest to each test data item was chosen as the training dataset. Then, for each test data item, the k-Nearest Neighbor classification was applied to predict this in the training dataset. The experimental results on real datasets confirm that the proposed k-Nearest Neighbor classification method performs well with regard to classification accuracy and performance compared with the conventional classification k-Nearest Neighbor and similar previous method. The main contributions of this paper can be summarized as follows:

- A new model is proposed to improve k-Nearest Neighbor classification method by using a two-phases approach based on MapReduce, implemented within the Spark framework.
- A parallel centroid-based and hierarchical clustering algorithm is applied to separate the sample of training dataset into multiple parts.
- The appropriate cluster for each test data item is determined as new training dataset to reduce the calculation of the k-Nearest Neighbor approach.
- Sets of experiments are conducted on the UCI datasets. The experimental results confirm that the proposed k-Nearest Neighbor classification method performs well in terms of classification accuracy and performance.

The rest of the paper is structured according to this. The related works are briefly reviewed in Section 2. Section 3 deals with the overall system and proposed workflow process for the k-Nearest Neighbor classification. The training process is discussed briefly in Section 4 that consists of the clustering method and its parallelization. Then, Section 5 explains the testing process. The experimental results are summarized in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

Research into the Nearest Neighbor technique has attracted considerable attention because of its simplicity and effectiveness since it was proposed in 1967 [23]. This method has been widely used in various applications including recommendation systems, databases, pattern recognition, data compression, DNA sequencing, etc.

The Nearest Neighbor search is an optimization problem to find a minimum distance or maximum similarity instance set [5,8]. A famous generation of Nearest Neighbor searches is k-Nearest Neighbor. The search for k-Nearest Neighbor finds *k* instances that minimize distance function. This approach has two different implementations: exact k-Nearest Neighbor search, and approximate k-Nearest

Neighbor search. The former performs a pair-wise comparison in all instances. It is obvious that the k-Nearest Neighbor has high computational cost. Although, there are some techniques for indexing the feature space and reducing computational complexity. The second one does not strive to implement a precise k-Nearest Neighbor. It uses subsets of the training data to minimize calculations [24,25].

By applying sophisticated data structures and pruning techniques, the traditional k-Nearest Neighbor methods in big data have attempted to solve the problem in a single machine [26-30]. However, we are just interested in distributed and parallel approaches. Triguero et al. proposed an evolutionary method for classification based on sampling [31]. It uses two phases of MapReduce: In the first phase, after sampling, a decision tree is established in each map. The test set is classified into the second phase by applying the set of trees. This uses a partial k-Nearest Neighbor over the subsets of training data. Du et al. described a clustering of density peaks dependent on k-Nearest Neighbor for large datasets [32]. This represents the key component analysis for processing large-dimensional data. This method is sensitive to the parameters of density estimation.

Deng et al. suggested a novel approach by using two different steps to cluster big data [33]. In the first step, k-means clustering is performed to separate the entire training dataset. Thus, this step creates *k* clusters and cluster centers. In the second step, the nearest cluster center is found for each test data item and the corresponding cluster is used as the training dataset for it. Then, a k-Nearest Neighbor is applied for the classification of the test data. Although distributed solutions are not included in this approach, it works relatively well in terms of accuracy and efficiency.

There are several studies conducting k-Nearest Neighbor join queries in the MapReduce [34,35]. The method described by Moutafis et al. applies five phases of the MapReduce [34]. The data space is subdivided into a grid of cells of equal size. The number of training data is determined per cell at first phase. In the second phase, an initial k-Nearest Neighbor list is generated for each test data, based on the training data in the same cell. In the third phase, the lists from the previous phase are verified by collecting more training data from neighboring cells. For each item of test data, plane-sweep technique is applied when required. The lists of k-Nearest Neighbor are joined into the final lists in the fourth phase. In the final phase, the classification of each test data item is performed based on the class of its neighbors. In this approach, load balancing is improved by the implementation of an adaptive partitioning scheme based on Quadtrees.

Chatzimilioudis et al. have introduced a batch-oriented algorithm called Spitfire [36]. This algorithm has its own distributed procedure and does not follow a MapReduce model. It employs three steps: split, refine, and replicate. At first, the data space is partitioned into disjoint sub-areas. Then, at each split, the k-Nearest Neighbors are calculated and replicated in splits. The final result is determined in the last step. This algorithm is implemented using the distributed file system Tachyon and Parallel Java Library for MPI.

Sun et al. proposed a method to classify facial images using Hadoop and k-Nearest Neighbor [37]. It splits the training dataset into several disjoint parts using the Map phase. This method uses three MapReduce processes for scanning the images, extracting facial features, and recognition for every single item of test data. This method conducts the MapReduce iteratively for each test data item resulting in very time-consuming operations. The exact implementation of the k-Nearest Neighbor algorithm is shown by Maillo et al. [38]. It uses a MapReduce process to classify the test dataset against the training dataset. The training dataset is divided into a certain number of disjoint partitions during the Map phase. The whole test dataset is sent to all maps. Hence, the test dataset is read line by line from the HDFS. The distance of each test data item is measured against the training dataset in each map. The class label and distance of *k* nearest neighbors are saved for each test data item. Each map sends its result to a single reduce task when its processing is complete. Then, the reduce task determines final neighbors for every test data item from the lists obtained during the map phase. Maillo et al. extended this method by the use of multiple reducers and in-memory solutions [39]. These methods have some limitations: first, these techniques add an enormous number of parameters. Second, the test data are inefficiently iterated on the driver. However, this paper introduces an approximate k-Nearest Neighbor algorithm, it has relatively good classification accuracy. In addition, the proposed method reduces the time complexity of the k-Nearest Neighbor classification compared to other distributed k-Nearest Neighbor methods.

3. System Overview

k-Nearest Neighbor algorithm calculates the distance in the training dataset between each test data item and all training data items and returns *k* nearest items. Linear time complexity is required to find the exact *k* nearest neighbors. Let *n* be the number of the training dataset and *d* be the number of dimensions, $O(nd)$ is the computational complexity for each test item [40]. Thus, it is very expensive for big data. This paper presents a new training process for the k-Nearest Neighbor classification. The

proposed method does not perform an exact k-Nearest Neighbor but it obtains high accuracy in classification.

Figure 1 shows proposed workflow process for k-Nearest Neighbor classification. Before the test process, a proposed clustering method will partition a sample of the training dataset into several clusters. The training data are more similar in each cluster than in other clusters. Then, the cluster centers are determined for all of the clusters obtained. The next step shall be to select the nearest cluster center to each test item. For each test item, its corresponding cluster is applied as a new training dataset. k-Nearest Neighbor classification is used to predict class of test items.

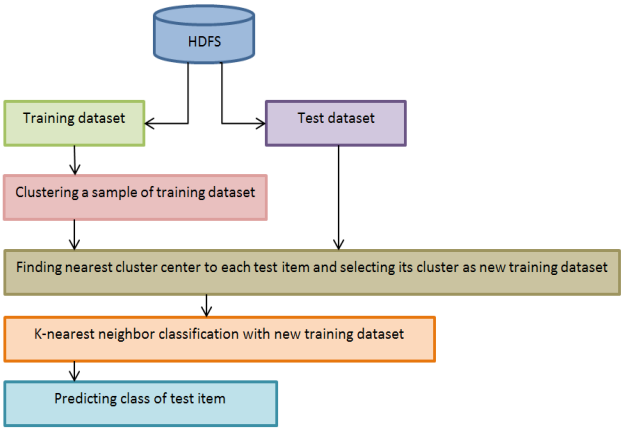


Figure 1. The proposed workflow process for the k-Nearest Neighbor classification.

The proposed algorithm consists of two main processes: training and testing. The training process is designed to partition a sample of the training dataset into appropriate clusters. A proper method of clustering is required for this process. The testing process is organized to find the cluster that is nearest to each test item. Then, the selected cluster is used as the training dataset to classify the test item. The k-Nearest Neighbor algorithm is applied in this process.

4. Training Process

Cluster analysis is one of the fundamental and widely used knowledge-discovery techniques. Clustering algorithms can be used in various applications for data mining such as recommendation systems, data compression, and data preprocessing. As described earlier, clustering is the task of grouping a set of data in such a way that data in the same group are more similar to each other than data in other groups. In other words, data have high similarity within a cluster and low similarity between clusters. An example of the clustering is shown in Figure 2.

The most common methods of clustering can be divided into four main categories: density-based clustering, grid-based clustering, partition-based clustering, and hierarchical clustering. The density-based clustering methods are unsupervised learning techniques [41,42]. Their principal idea is that a cluster is determined by an adjacent area of high data density. Low-density areas are the dividing parts and data in those parts are considered outliers.

The second category of clustering algorithms is the grid-based clustering method [43,44]. A set of grid cells is defined within data space. Then, each data item is allocated to a suitable grid cell. The density of each cell shall be calculated in the next step. In the end, the adjacent groups of dense cells establish clusters.

The partition-based clustering methods establish initial partitions of a dataset into a set of clusters of a given number [45]. Then, they iterate refinements to maximize intra-cluster similarity and inter-cluster dissimilarity. The partition-based clustering approach is also called objective function clustering. The explanation for this is that a certain objective function is minimized. For instance, the k-means algorithm [46] minimizes the mean square error of each data item in a cluster as regards its cluster centroid. This method is simple, fast, and effective. It is widely used in practical applications. However, it depends on selecting the *k* initial points.

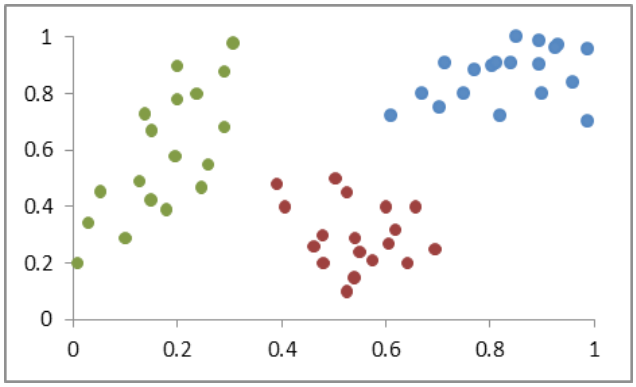


Figure 2. An example of clustering.

The hierarchical clustering algorithms [47] form a hierarchy for clusters. They have two strategies for data decomposition: agglomerative and divisive methods. An agglomerative method is a bottom-up approach. At first, every item in the data is in a cluster. Then, an appropriate criterion merges pairs of clusters as one. A divisive method is a top-down approach. First, all items in the data are in one cluster. Next, the cluster is separated until each item of the data is a separate cluster. These methods reflect the proper accuracy of clustering. For instance, BIRCH [48] is an accurate method that does not rely on initial parameters for its results and can not be influenced by outliers. However, it is not as fast as the k-means method.

The hierarchical clustering methods lose performance and scalability, as they try to form clusters of high quality.

However all studies define unsupervised learning as the most important clustering feature, all major methods require some user information. For instance, the k-means method requires the number of clusters to be obtained and the hierarchical clustering methods usually need to get the termination condition. A new clustering using a binary splitting method is proposed by Mazzeo et al., called CLUBS$^+$, to resolve this major limitation [49]. It is a centroid-based and hierarchical clustering method profiting from the combination of divisive and agglomerative algorithms. First, Masciari et al. demonstrated the idea of divisive and agglomerative corporate behavior [50], and then its functionality defined [51]. CLUBS$^+$ has improved its predecessor approach with two refinements. It surpasses the detection of outliers for the recognition of high-density outliers from low-density clusters areas. It also creates ideal ellipsoid clusters around the centroids. Lanni et al. represented a parallel version of CLUBS$^+$ called CLUBS-p [52]. It scales linearly with respect to dataset size. These characteristics make it a very powerful algorithm for clustering large quantities of data. Because of the listed features, this paper uses a parallel version of CLUBS$^+$ to cluster the training dataset.

4.1 Clustering Using Binary Splitting

CLUBS$^+$ is introduced as a parameter-free clustering algorithm that uses the binary splitting to partition data space. It is also a centroid-based and fast hierarchical clustering algorithm collaborating with the advantages of the divisive and agglomerative techniques. This algorithm consists of four phases: (1) divisive (2) intermediate refinement (3) agglomerative and (4) final refinement. The input dataset is split into several rectangular blocks by a divisive algorithm in the first phase. Then, a refinement process is performed on these blocks. That leads to the formation of new clusters of points. Some points are introduced as outliers and classified into one cluster. It uses an agglomerative algorithm in the third phase to merge some clusters into one cluster. Finally, clusters from the preceding phase and outliers from the second phase are refined and the final clusters and final set of outliers are obtained in this phase. Figure 3 represents the CLUBS$^+$ algorithm operations.

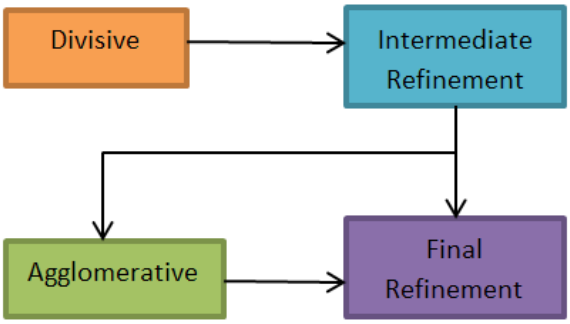


Figure 3. The four phases of the CLUBS$^+$ algorithm.

4.1.1 Problem Definition

By default, a dataset $D=\{p_1, p_2, ..., p_n\}$ is considered where p_i is a d-dimensional point. The aim is to obtain a cluster set $CS=\{C_1, C_2, ..., C_k\}$ for the points in D such that points in the same cluster are the most similar, and points in the different clusters are the least similar. It is notable that each point p_i is assigned only one cluster. In this paper, the Euclidean distance will be used as a measure of similarity between pairs of points. It means that the lower distance between pairs of points represents the greater similarity.

Assume $p \in D$ is a d-dimensional point and p_i is indicating its i-th coordinate. The Euclidean distance of point p to point q is given in Equation (1):

$$\|p - q\| = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2} \tag{1}$$

The following defines several concepts and analytical characteristics that will be used in the CLUBS$^+$ algorithm.

Suppose that $C=\{p_1, p_2, ..., p_m\}$ is a common cluster that includes m d-dimensional points. Let p_{ij} denotes j-th coordinate of p_i. The point $\overline{cp} = (\overline{cp_1}, \overline{cp_2}, ..., \overline{cp_d})$ is the centroid point of C such that $\overline{cp_j}$ is the average of the j-th coordinates of all points in C. The formula $\overline{cp_j}$ is given in Equation (2):

$$\overline{cp_j} = \frac{1}{m} \sum_{i=1}^{m} p_{ij} \quad , (1 \le j \le d) \tag{2}$$

For this algorithm, a further measurement called *WCSS* (Within Cluster Sum of Squares) is used. For each cluster the *WCSS* is defined as follows:

$$WCSS(C) = \sum_{i=1}^{m} \|p_i - \overline{cp}\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{d} (p_{ij} - \overline{cp_j})^2 \tag{3}$$

This measure can be determined effectively using some computed aggregate information for each cluster.

Theorem 1

Assume C is an individual cluster with a set of m d-dimensional points. Then,

$$WCSS(C) = \sum_{j=1}^{d} (Q_j - \frac{S_j^2}{m}) \quad (4)$$

where, $S_j = \sum_{i=1}^{m} p_{ij}$ is the sum of the j-th coordinates and $Q_j = \sum_{i=1}^{m} p_{ij}^2$ is the sum of the squares of the j-th coordinates of the points in cluster C [49].

Suppose $CS=\{C_1, C_2, ..., C_k\}$ is a set of clusters over a dataset, $D = \{p_1, p_2, ..., p_n\}$, and each C_i cluster has m_i points. The definition of $WCSS(CS)$ is in Equation (5):

$$WCSS(CS) = \sum_{i=1}^{k} WCSS(C_i) \quad (5)$$

Let $\overline{tcp}$ be the centroid point on dataset D. The $BCSS$ (Between Clusters Sum of Squares) of the cluster set CS is calculated by Equation (6):

$$BCSS(CS) = \sum_{i=1}^{k} (m_i \times \|\overline{cp_i} - \overline{tcp}\|^2) \quad (6)$$

Assume dataset D is a single cluster, and calculate $WCSS(D)$ on the basis of Equation (4). For each arbitrary set of clusters CS over D, we get it [49]:

$$WCSS(D) = WCSS(CS) + BCSS(CS) \quad (7)$$

It implies for every partition of D, the sum of $WCSS$ and $BCSS$ is constant. This method employs another theorem that is used to merge and split clusters.

Theorem 2

Let $CS=\{C_1, C_2, ..., C_k\}$ be a set of clusters over the dataset D and let C_i and C_j be two clusters in CS. Suppose CS_{ij} is obtained from the cluster set CS by exchanging clusters C_i and C_j with their union $\{C_i \cup C_j\}$. Therefore,

$$\Delta WCSS(CS_{ij}) = WCSS(CS_{ij}) - WCSS(CS) \quad (8)$$

The $\Delta WCSS(CS_{ij})$ can be calculated by Equation (9):

$$\Delta WCSS(CS_{ij}) = \frac{m_i \times m_j}{m_i + m_j} \|\overline{cp_i} - \overline{cp_j}\|^2 \quad (9)$$

where, m_i and m_j denote the number of points in C_i and C_j and $\overline{cp_i}$ and $\overline{cp_j}$ represent the centroid points of C_i and C_j respectively [49]. This equation is replaceable as follows:

$$\Delta WCSS(CS_{ij}) = \frac{m_i \times m_j}{m_i + m_j} \sum_{k=1}^{d} (\frac{S_{i,k}}{m_i} - \frac{S_{j,k}}{m_j})^2 \quad (10)$$

where, $S_{i,k}$ ($S_{j,k}$) is the sum of all points in cluster C_i (C_j) in k-th dimension. This equation is applied in the divisive phase. When a cluster is divided into two clusters, this results in a reduction of the $WCSS$. On the other hand, when two clusters are merged into one cluster, this causes the $WCSS$ to grow. This decrease and increase can be determined by Equation (10). These definitions and theorems determine the main strategy of the CLUBS$^+$ method.

4.1.2 The Divisive Phase

In this phase, a top-down partitioning of the dataset is performed to gain rectangular blocks of points. Each block contains points similar to each other, as much as possible. This means this algorithm is attempting to reduce the $WCSS$. This purpose is also followed by other clustering algorithms such as k-means.

Identifying a partition that minimizes the $WCSS$ even for two-dimensional points is an NP-hard problem [53]. A greedy algorithm is applied in the CLUBS$^+$ method. In each step, each block is divided into a pair of blocks to improve the specific clustering criteria.

Given D as an input dataset of n d-dimensional points. D is regarded as a single block, entered into a priority queue, Q. A block is removed from Q and split into a pair of blocks, while Q is not empty. The new blocks replace the previous one in Q if the partition is successful, otherwise, the block will be identified as the final block.

Blocks in Q are sorted by their $WCSS$ in descending order because blocks with larger $WCSS$ have higher splitting priorities. Furfaro et al. demonstrated that as a greedy algorithm is used for hierarchical clustering to minimize the overall variance with a limited number of clusters, the selection of the cluster with the largest variance leads to the best results for each iteration [54]. Therefore, the block with the largest $WCSS$ is evaluated for splitting at each step. Two critical points should be considered for producing efficient splits: (1) calculating the best split and (2) assessing the efficiency of the split.

The calculating of the best split

To find the best split for a block, $WCSS$ should be minimized. In other words, the aim is to partition a block into a pair of blocks to maximize the $\Delta WCSS$ (Equation (9)). Blocks are partitioned using hyperplanes that are orthogonal to some dimension. Each split is identified by a pair <*dim*, *value*> where the separating dimension is *dim* and the separating position is *value*. The points of block B are partitioned into a pair of blocks $\{B_1, B_2\}$ by split <*dim*, *value*>, where B_1 contains the points of B whose *dim*-th coordinate is less than or equal to the *value* and B_2 contains the remaining points.

It is essential to measure cumulative marginal distributions on each dimension before splitting a block. Assume $B = \{p_1, p_2, ..., p_n\}$ is a set of n d-dimensional points and p_{ij} is j-th coordinate of p_i. The marginal sum of the j-th di-

mension of B is defined as Equation (11):

$$ms_j: R \to R^d, ms_j(v) = \sum_{p_i \in B \wedge p_{ij}=v} p_i \quad (11)$$

The marginal count of the j-th dimension of B can be calculated by Equation (12):

$$mc_j: R \to N,\ mc_j(v) = \left|\left\{p_i \mid p_i \in B \bigwedge p_{ij} = v\right\}\right| \quad (12)$$

Therefore, the cumulative marginal distribution at coordinate v of dimension i is the sum (count) of points whose coordinates in the i-th dimension are less than or equal to v:

$$cms_j(v) = \sum_{p_i \in B \wedge p_{ij} \le v} p_i \ , \quad (13)$$

and

$$cmc_j(v) = \left|\left\{p_i \mid p_i \in B \wedge p_{ij} \le v\right\}\right| \quad (14)$$

These functions can be represented as arrays and determined by a linear scanning of the data in B. It is possible to calculate $\Delta WCSS$ in constant time as the cumulative marginals are precomputed. To determine the best split, it is important to find a pair <dim, value> that will maximize $\Delta WCSS$.

Evaluating of the effectiveness of the split

One essential problem in any divisive algorithm is deciding when to prevent the further division of the current clusters. The reason is that the additional division may not improve the overall quality of the current cluster set. There are many criteria in the literature that estimate the quality of the current cluster set [55]. In this paper, the CH-index is identified as the most appropriate criterion [56].

The *CH-index* for a set of k clusters $CS=\{C_1, C_2, ..., C_k\}$ over a dataset D, with $|D|=n$ is defined as Equation (15):

$$CH - index(CS) = \frac{BCSS(CS)}{WCSS(CS)} \times \frac{n-k}{k-1} \quad (15)$$

The new *CH-index* is calculated after each step. If the split increases the new *CH-index*, the split is effective and the divisive phase continues. Otherwise, a "local" criterion is checked according to the existence of a "valley" in the marginal distribution. Even though a split could not increase the *CH-index*, a very large local discontinuity could justify the split of a block anyway. The divisive phase is shown in Algorithm 1. The algorithm for this phase consists of two sub-algorithms that determine the best split and effectiveness of the split.

Algorithm 1 Divisive Phase

Input: $D = \{p_1, p_2, \ldots, p_n\}$, a dataset and a set of n d-dimensional points.
Output: $B = \{B_1, B_2, ..., B_t\}$, a partitioning of D into t blocks.
1: Let $Q=\emptyset$ be a priority queue and $B=\emptyset$.
2: Let D be a single block and is entered into Q.
3: Set t=1, max_CH=0;
4: Set $wcss$= $WCSS(D)$;
5: Set $bcss$= 0;
6: while (Q!=*NULL*)
7: $S \leftarrow Q$.deleteQueue();
8: Compute (*dim*, *value*, $\Delta WCSS$) for the best split of S.
//The *dim* is the splitting dimension and the *value* is the splitting position
// Compute $\Delta WCSS$ based on Equation(10)
9: **If** split is effective **then**
10: {*S1*, *S2*} ← make split (S);
11: Q.addQueue(*S1*);
12: Q.addQueue(*S2*);
13: $wcss = wcss - \Delta WCSS;$
14: $bcss = bcss + \Delta WCSS;$
15: $t=t+1$;
16: $CH = \frac{bcss}{wcss} \times \frac{n-t}{t-1}$;
17: **If** $CH > max\_CH$ **then**
18: $max\_CH = CH;$
19: **end if**
20: **else**
21: $B \leftarrow B \cup \{ S \}$;
22: **end if**
23: **end while**
24: **return** B

4.1.3 The Intermediate Refinement Phase

The overall dataset was partitioned in several blocks in the previous phase. However, some blocks contain outliers or noise points. The aim of the intermediate refinement phase is (1) to recognize and separate blocks containing only noise points, and (2) to create well-rounded clusters for the remaining blocks.

The main idea is the low density of blocks including only outliers. Additionally, due to the random nature of the outliers, the density of these blocks is uniform. First, the blocks are sorted by their densities in ascending order (As shown in Algorithm 2). Then, changes in density between adjacent blocks are observed and the first jump

determines the candidates for outlier-blocks. In the next step, the blocks that are considered to be outlier should be examined. For this reason, each candidate block is considered to have a small hypercube around the centroid point. When the hypercube has significantly less density than the whole block, the candidate block is identified as an outlier block otherwise it is known as a cluster block.

Following the determination of the cluster blocks and outlier blocks, the second target of this phase is pursued. To this end, the blocks near every block B_i in the set are found. Suppose C_k is a cluster identified from cluster set with the centroid point $\overline{cp} = (\overline{cp_1}, \overline{cp_2}, ..., \overline{cp_d})$. The distance between each point $p_i = \{p_{i1}, p_{i2}, ..., p_{id}\}$ and each cluster block C_k is calculated by Equation (16):

$$dist(p_i, C_k) = \sum_{j=1}^{d} \left(\frac{\|p_{ij} - \overline{cp_j}\|}{r_i}\right)^2 \quad (16)$$

where r_i is the radius of the cluster along the i-th dimension that can be estimated by Equation (17):

$$r_i = 3 \times \sqrt{\frac{WCSS_i(C_k)}{n}} \quad (17)$$

such that n is the number of points in the cluster block C_k and $WCSS_i(C_k)$ is calculated by Equation (4) only for i-th coordinates of the points in C_k [57]. It is notable that if $dist(p_i, C_k)=1$, there is an ellipsoid whose center is the centroid point of C_k and whose radius on the i-th dimension is r_i. Consequently, each point p_i is allocated to cluster RC_k with the lowest $dist(p_i, C_k)$, and the minimum distance is less than or equal to 1. Otherwise, p_i is classified as an outlier in RC_0.

Hence, ellipsoids are used in this method instead of spheres. It causes some dimensions to stretch or shrink to the previous phase. However, some blocks contain outliers or noise points. The aim of the intermediate adjust the data in hypercubes. The step could increase the effectiveness of the algorithm. Therefore, each point is assigned to the cluster to minimize its distance from the cluster's centroid relative to the radius of the cluster.

Algorithm 2 Intermediate Refinement Phase

Input: $B = \{B_1, B_2, ..., B_t\}$, a partitioning of D into t blocks.
Output: $RC=\{RC_0, RC_1, ..., RC_u\}$, a clustering of the points where RC_0 is the only block that contains outliers .
1: Let $density = \{d_1, d_2, ..., d_t\}$ be the densities of blocks in B.
2: Let $centroid\_density=\{cd_1, cd_2, ..., cd_t\}$ be the densities of small hypercubes around the centroid points of blocks.
3: $density1 \leftarrow$ Sort ($density$) // in ascending order.
4: for i=1; $i \le t$-1; i++ **do**
5: $Jump[i]= density1[i+1]/ density1[i]$;
6: end for
7: Set sum=0;
8: for i=1; $i \le t$-1; i++ **do**
9: $sum= sum + Jump[i]$;
10: end for
11: $avg= sum / (t-1)$;
12: Set sw= 0;
13: for i=1; $i \le t$-1 && sw==0; i++ **do**
14: If $Jump[i] > avg$ **then**
15: $MDO= density1[i]$; // maximum density of the outlier block
16: sw= 1;
17: end if
18: end for
19: for i=1; $i \le t$; i++ **do**
20: If $density[i] \le MDO$ && $centroid\_density[i] < 2* density[i]$ **then**
21: $outlier \leftarrow outlier \cup \{ B_i \}$;
22: else
23: $cluster \leftarrow cluster \cup \{ B_i \}$;
24: end if
25: end for
26: for each $B_i \in B$ **do**
27: $NB \leftarrow$ near_blocks(B_i, $cluster$) // find indexes of near // blocks to B_i in $cluster$.
28: for each $p_j \in B_i$ **do**
29: Set c=0, d_{min}=1;
30: for each $k \in NB$ **do**
31: If $dist(p_j, C_k) \le d_{min}$ **then**
32: $d_{min}= dist(p_j, C_k)$;
33: $c=k$;
34: end if
35: end for
36: $RC_c \leftarrow RC_c \cup \{ p_j \}$;
37: end for
38: end for
39: return RC

4.1.4 The Agglomerative Phase

The main goal of this phase is to enhance the overall quality of the clustering. Some clusters that were created in the previous phase are merged in this phase. A pair of clusters are evaluated for merging at each stage. The process of merging is conducted provided it leads to the growth of the *CH-index* and the least increase of *WCSS*. This means that they will be replaced by the union of the two clusters. If the merge of these two clusters leads to a decrease in the *CH-index,* the merge is not done.

There is no limit to selecting which clusters to merge, as opposed to the divisive phase. In the divisive phase,

each rectangular block is divided into two rectangular parts. In the agglomerative phase, there is no constraint on the shape of the resulting cluster from the merging process. This phase is very quick, as it is precomputed based on prior information and does not require data access.

4.1.5 The Final Refinement Phase

Some clusters have an irregular shape that is produced in the previous phase. On the other side, the final outliers must be identified. The main goal of the final refinement phase is to enhance the quality of the clusters and identify the final outliers. The final refinement phase is close to the end part of the intermediate refinement phase. Algorithm 3 shows the final refinement phase that the final set of clusters and the final outliers are its output. The complexity of the CLUBS$^+$ algorithm is $O(n \times k)$ where n is the number of the points and k is the number of clusters that were found following the divisive phase [52].

Algorithm 3 Final Refinement Phase

Input: $RC=\{RC_0, RC_1, ..., RC_u\}$, a clustering of the points where RC_0 is the only cluster that contains outliers.
Output: $C=\{C_0, C_1, ..., C_u\}$, a clustering of the points where C_0 is the only cluster containing outliers.
1: *cluster* ← $\{RC_1, RC_2, ..., RC_u\}$;
2: for each $B_i \in RC$ **do**
3: NB ← near_blocks(B_i, *cluster*); // find indexes of near clusters to B_i in *cluster*.
4: for each $p_j \in B_i$ **do**
5: Set $c=0$, $d_{min}=1$;
6: for each $k \in NB$ **do**
7: If $dist(p_j, C_k) \leq d_{min}$ **then**
8: $d_{min} = dist(p_j, C_k)$;
9: $c=k$;
10: end if
11: end for
12: $C_c \leftarrow C_c \cup \{p_j\}$;
13: end for
14: end for
15: return C

4.2 The Parallelization of CLUBS$^+$

As stated earlier, the MapReduce framework is used to parallelize our algorithms. The basic idea is to split the dataset on a large scale into small datasets. The small data sets are then distributed to the nodes (workers) for calculations. The nodes (workers) are coordinated by a master node. The datasets are processed by Map function and Reduce function, and the final results are then obtained.

Analysis of the CLUBS$^+$ method steps is essential to parallel its execution. Some operations require entire access to the data. Therefore, the most important operations are checked to parallel them in the four phases of the method.

4.2.1 The Parallelization of Divisive Phase

The divisive phase consists of two sub-algorithms that find the best split and evaluate the effectiveness of the split. For these objectives, the measurement of the cumulative marginals is required in this phase. This calculation requires access to the entire dataset. Assume input dataset D is loaded into a distributed file system (HDFS) by the data fragments called splits. Each split S_i is sent to a mapper for processing. Therefore, data set $D\{S_1, S_2, ..., S_t\}$ is sent to the mappers. For each dimension, the vectors *ms* and *mc* (Equation (11) and Equation (12)) can be calculated independently and in parallel on each split S_i. Then, the final vectors *ms* and *mc* can be acquired by summing and counting the obtained vectors on all S_i. The reducers are getting the final results according to the MapReduce paradigm. Other operations can be handled by the master node.

4.2.2 The Parallelization of the Refinement Phases

The density of the blocks has to be measured in the intermediate phase to assess the outlier blocks. After the divisive phase, the master node obtains a list of the blocks with their range and density. Furthermore, the density of each block around the centroid is required to find out the outlier blocks. For this purpose, a small range is deemed for each block around the centroid. Then, the number of points that fall in each of these specified areas is counted in parallel. The only mathematical operation involved in these phases is the sum, which is completely parallelizable. For every part of a block's data, the number of points falling within a specified range can be counted independently. Then, the total count is obtained by the sum of partial counts. As described before, this count permits the master node to identify the outlier blocks.

Another important operation performed during both refinement phases is the assignment of points to the clusters, which can be performed independently on each worker node. Cluster information is in the master node that has to be sent to all the worker nodes. Then, each worker node will measure the distance between points and clusters and allocate points to the appropriate cluster. After the assignment of the points, some information must be updated by the master node. To this end, each worker node computes certain information for the master node, such as the new centroid, the number of points, and the new radius. The

master node would then get updated information about clusters by aggregating all the values from the worker nodes.

4.2.3 The Parallelization of the Agglomerative Phase

In this phase, some clusters are merged to improve the overall effectiveness of the clustering. For this purpose, some precomputed prior information about clusters is required and does not require access to the entire data. The master node has the required information and the operation can be performed by a worker node.

Four phases of the CLUBS$^+$ method have been reviewed for parallelization. The parallelization of this method is used to cluster the training dataset for the next step of the proposed k-Nearest Neighbor classification.

5. Testing Process

Assume that the parallel CLUBS$^+$ algorithm produces u clusters and cluster centers, then the cluster with the nearest cluster center is found for each test item. The cluster found is being used as the new training dataset for the test data item. The k-Nearest Neighbor algorithm is used to classify the test data item into the new training dataset. Due to the selection of a cluster with high similarity to the test data item, the proposed approach also has relatively high classification accuracy. The proposed method (KNN-CCL) is represented in the Algorithm 4.

Algorithm 4 KNN-CCL algorithm

Input: $X=\{x_1, x_2, ..., x_n\}$, the training dataset.
$Y=\{y_1, y_2, ..., y_m\}$, the test dataset.
Output: $CL=\{cl_1, cl_2, ..., cl_m\}$, the class label of each test data item.
1: Produce u clusters by Algorithm 1 to Algorithm 3.
2: $C=\{C_1, C_2, ..., C_u\}$;
3: for each $y_i \in Y$ **do**
4: Set min=0, $d_{min}=M$; // M is a very big value
5: for each $C_j \in C$ **do**
4: Compute distance $dist\ (y_i, C_j)$ between y_i and the cluster center of C_j.
5: If $dist(y_i, C_j) \leq d_{min}$ **then**
6: $d_{min}= dist(y_i, C_j)$;
$min=j$;
5: end if
6: end for
6: Use C_{min} as new training dataset for y_i.
7: Find k-nearest neighbor for y_i in C_{min}.
8: $knn_i=\{nn_1, nn_2, ..., nn_k\}$;
8: Predict the class label cl_i for y_i based on knn_i.
9: $CL \leftarrow CL \cup \{cl_i\}$;
10: end for
11: return CL

The size of the new training dataset in this method is much smaller than the size of the original training dataset. Hence, the calculation of the k-Nearest Neighbor algorithm is reduced, and the KNN-CCL algorithm improves the classification quality. Furthermore, some previous methods require the number of clusters or the termination condition to be obtained. The CLUBS$^+$ algorithm requires none of them and the aforementioned issues can not affect the overhead of clustering and classification accuracy.

6. Experiments

This section describes the factors and points related to the experimental study. It also analyzes the results obtained from various experimenal studies.

6.1 Experimental Framework

The following measures are considered in this paper for evaluating the performance of the method proposed:

- Classification accuracy: this represents the number of correct classification with respect to the total number of instances.
- Execution time: the total execution time spent by classifiers in the classification of a given test set against a training dataset shall be collected. It includes reading and distributing all data and all computations carried out using comparative methods.

As described in previous sections, the proposed method KNN-CCL uses the parallel CLUBS$^+$ algorithm to partition the training dataset. Then, it will provide the appropriate cluster for each test data item as a new training dataset. Ultimately, it uses the k-Nearest Neighbor classification to predict each test data item in the new training dataset. The whole training dataset is not used in experiments. A random sampling method [21] is applied to sample 20% of the training dataset. We took the classification of k-Nearest Neighbor (KNN) as a baseline to demonstrate the effectiveness of the KNN-CCL algorithm and made a comparison between KNN, KNN-CCL, and LC-KNN [33]. LC-KNN uses k-means clustering to separate the training dataset. We used a similar random sampling approach before the clustering process to sample 20% of the training dataset. We also repeat the k-means algorithm 10 times and use the cluster centers afterward. Each experimental

group is repeated three times, and the average values are reported for a more reliable result.

Three real datasets will be used for the experimental study. Pendigits, Letter, and Satimage are extracted from the UCI machine learning repository [58]. The Pendigits is a dataset about pen-based recognition of handwritten digits. This includes 10992 instances, 16 features, and 10 classes. The Letter is a letter recognition dataset. It contains a total of 20000 instances, 16 features, and 26 separate classes (capital letters in the English alphabet). The Satimage is a dataset of satellite images. This involves 6435 instances, 36 features, and 7 classes.

All experiments were carried out on the same environment, an Intel Xeon E5-2680v2 with a total of 10 cores (20 threads) and 128 GB of memory. The experiments were executed using Spark 1.6.1 and Scala 2.10.5.

6.2 Experimental Results

This section describes and discusses the experimental results. First, we consider the number of clusters as a cluster to compare the performance of the algorithms in terms of classification. Therefore, we set k=1 and make use of classification accuracy and execution time as evaluations of the classification tasks. Our experimental results are reported in Table 1. Table 1 demonstrates that the proposed KNN-CCL method has improved 5~9 times over the KNN, and 2~4 times over the LC-KNN in terms of execution time. This means that the proposed distributed solution reduces the execution time of the proposed algorithm compared to the other two algorithms. In classification accuracy evaluation, the KNN-CCL and LC-KNN are 4%~10% lower and 2%~11% lower than KNN. Although LC-KNN uses a serial algorithm and algorithm KNN-CCL uses a parallel algorithm and distributes data between clusters, algorithm KNN-CCL can be equivalent or even better than algorithm LC-KNN in terms of accuracy evalution. Therefore, the KNN-CCL performs well in terms of classification accuracy and execution time with k=1 according to experimental results.

A group of experiments on three datasets is performed by selecting different k values for KNN, KNN-CCL, and LC-KNN. For this group of experiments, for particular, three methods were conducted on the datasets with k=1, 5, 10, 15, and 20, respectively. Comparison of the execution time of three algorithms KNN, KNN-CCL, and LC-KNN with different k values is shown in Figure 4.

From Figure 4, we observed that the execution times for three algorithms are linear with varying k values. In other words, for three algorithms, greater k values slightly increase the execution time over different datasets. As concluded from Figure 4, despite the increase in the number of clusters, the execution times of the algorithms will not differ much. The proposed algorithm KNN-CCL requires less execution time in all cases. The reason for this is that the KNN-CCL uses a parallel centroid-based and hierarchical clustering method to provide the appropriate training dataset for each test data item. The LC-KNN method uses the k-means algorithm which requires it to repeat the k-means algorithm by 10 times to obtain the appropriate training dataset. Thus, The LC-KNN method takes more execution time than the KNN-CCL method. On the other hand, this strategy (KNN-CCL) takes advantage of parallel execution to speed up computation.

Figure 5 illustrates the classification accuracy evaluation of the three algorithms KNN, KNN-CCL, and LC-KNN over three datasets with various k values. In Figure 5(a), the classification accuracy of KNN-CCL for the Pendigits dataset is 5%~12% lower than KNN and the classification accuracy of LC-KNN is 2%~10% lower than KNN. KNN-CCL and LC-KNN got almost the same results in most situations. The classification accuracy of three methods over the Letter dataset is shown in Figure 5(b). The KNN-CCL is 9%~22% lower than the KNN and LC-KNN is 11%~29% lower than the KNN. Obviously, the KNN-CCL algorithm provides better results than the LC-KNN. In Figure 5(c), the classification accuracy of both KNN-CCL and LC-KNN is 2%~9% lower than KNN over the Satimage dataset. KNN-CCL and LC-KNN obtained almost the same outcomes for the majority of cases. We observed, from Figure 5, that the classification accuracy of the proposed method KNN-CCL and LC-KNN were approximately equivalent.

Table 1. classification accuracy and execution time (seconds) of three algorithms on three datasets.

Dataset	KNN-CCL		LC-KNN		KNN	
	Accuracy	Time	Accuracy	Time	Accuracy	Time
Pendigits	0.927	0.7	0.9537	2.3	0.9774	5.71
Letter	0.8602	1.14	0.8459	3.2	0.9565	10.94
Satimage	0.8501	0.31	0.8667	1.28	0.8944	1.62

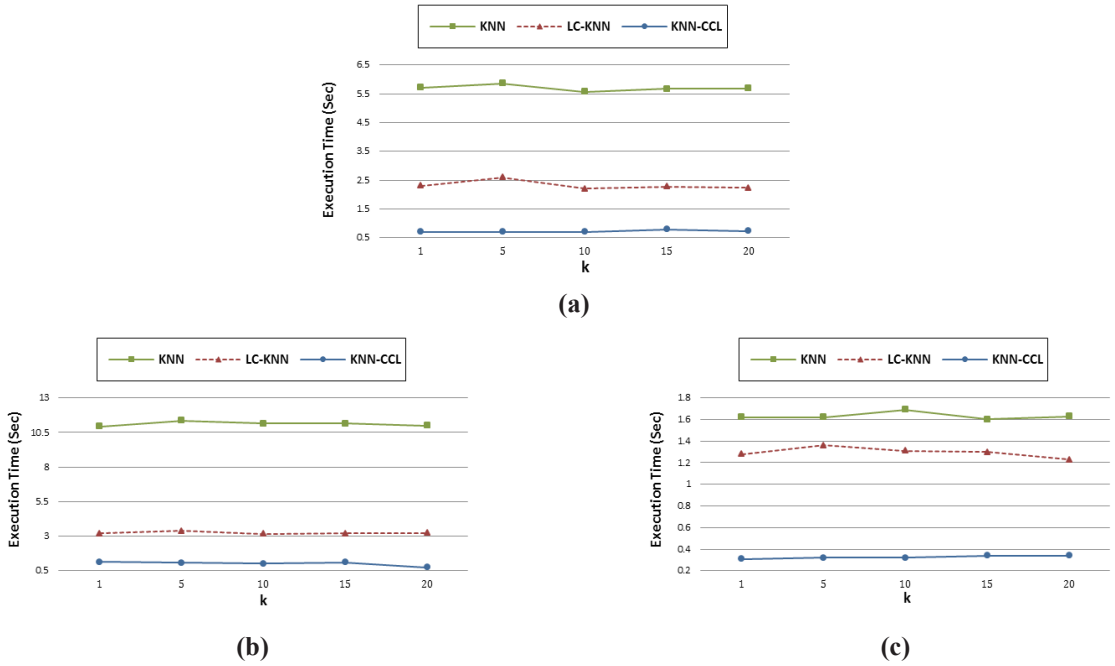


Figure 4. Execution time on three datasets at different values of k: **a** Pendigits **b** Letter **c** Satimage.

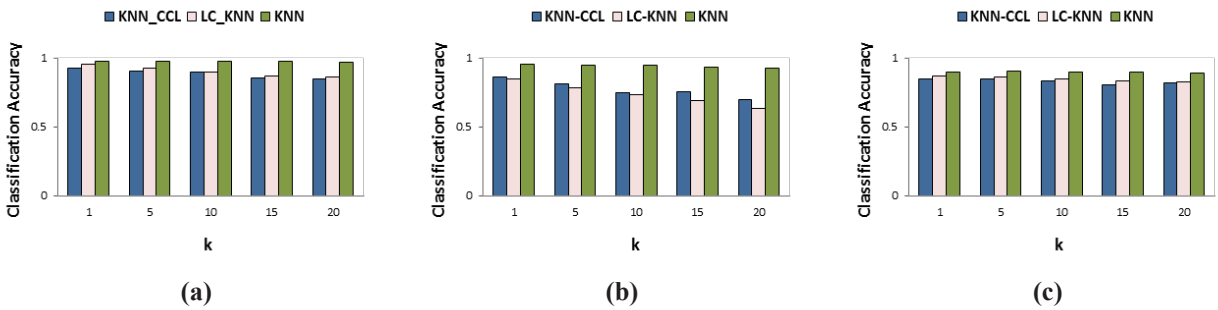
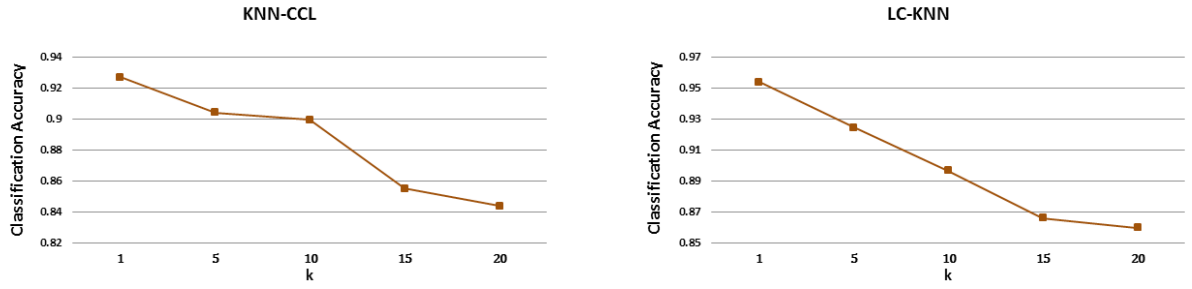


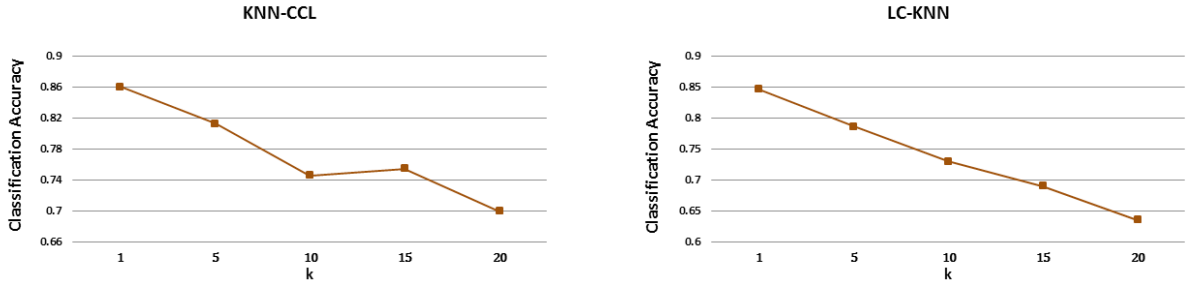
Figure 5. Classification accuracy on three datasets with varying *k* values: **a** Pendigits **b** Letter **c** Satimage.

In the end, we are going to study the classification accuracy of two KNN-CCL and LC-KNN algorithms to classification accuracy of two algorithms over three datasets to *k*=1, 5, 10, 15, and 20. As the value of *k* increases, the overall accuracy of classification decreases. This is because the smaller the training dataset would be, the greater the classification accuracy. Therefore, the difference between the samples is important, and the accuracy of classification will be reduced. In this case, it can be concluded that an acceptable *k* value should be selected. As shown in Figure 6, when two algorithms are more accurate, the *k* value should be set as small as possible. According to this analysis, if higher accuracy is required, a smaller number of clusters should be selected.

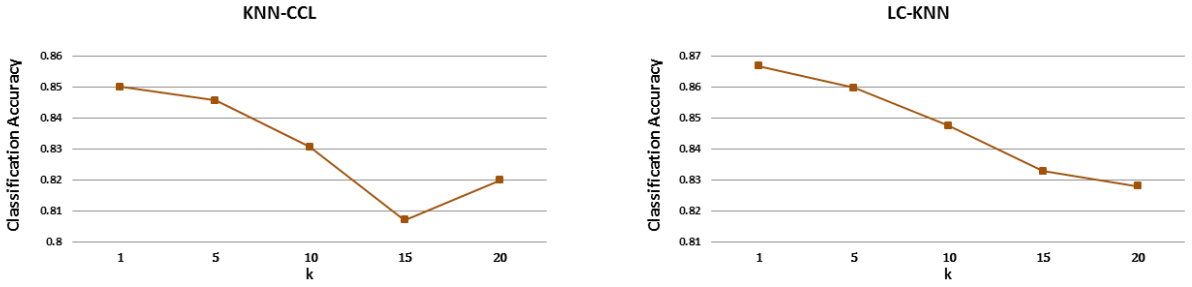
Therefore, according to experimental results, we can infer that the KNN-CCL and LC-KNN algorithms perform well with small values of *k* in terms of classification accuracy, and the proposed method KNN-CCL performs very well in terms of execution time.



(a) Pendigits dataset.



(b) Letter dataset.



(c) Satimage dataset.

Figure 6. Classification accuracy of KNN-CCL and LC-KNN on three datasets with different k.

7. Conclusions

In this paper, an efficient parallel processing of the k-Nearest Neighbor classification method was presented and evaluated. The proposed method is denominated as KNN-CCL uses a parallel centroid-based and hierarchical clustering algorithm to split the sample set of training dataset into several clusters. The introduced clustering algorithm uses four stages of successive refinements and generates high quality clusters. And the suitable cluster was selected as new training dataset for each test data item to decrease the calculation of classification. The k-Nearest Neighbor classification was used for each test data item to predict it in the new training dataset.

The KNN approach was considered the baseline, and a group of comparative experiments was performed by KNN, LC-KNN, and KNN-CCL. The experimental results carried out revealed that the proposed method can handle large dataset and performs well in terms of accuracy and performance in classification.

Conflict of Interest

There is no conflict of interest.

Funding

The authors received no specific funding for this work.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Alharthi, A., Krotov, V., Bowman, M., 2017. Addressing barriers to big data. Business Horizons. 60(3), 285-292.

[2] Anagnostopoulos, I., Zeadally, S., Exposito, E., 2016. Handling big data: research challenges and future directions. The Journal of Supercomputing. 72(4), 1494-1516.

[3] Akoka, J., Comyn-Wattiau, I., Laoufi, N., 2017. Research on Big Data – A systematic mapping study. Computer Standards & Interfaces. 54, 105-115.

[4] Minelli, M., Chambers, M., Dhiraj, A., 2013. Big data, big analytics: emerging business intelligence and analytic trends for today's businesses. John Wiley & Sons. 578.

[5] Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE transactions on information theory. 13(1), 21-27.

[6] Wu, X., Kumar, V., 2009. The top ten algorithms in data mining. CRC press.

[7] Chen, Y.H., Garcia, E.K., Gupta, M.R., et al., 2009. Similarity-based classification: Concepts and algorithms. Journal of Machine Learning Research. 10, 747-776.

[8] Weinberger, K.Q., Saul, L.K., 2009. Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research. 10, 207-244.

[9] Li, J., Liu, Y., Pan, J., et al., 2020. Map-Balance-Reduce: An improved parallel programming model for load balancing of MapReduce. Future Generation Computer Systems. 105, 993-1001.

[10] Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the Acm. 51(1), 107-113.

[11] White, T., 2012. Hadoop: The definitive guide. O'Reilly Media, Inc.

[12] Chen, C.P., Zhang, C.Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences. 275, 314-347.

[13] Guo, Z., Fox, G., Zhou, M., 2012. Investigation of data locality in mapreduce. Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). 419-426. IEEE.

[14] Zaharia, M., Chowdhury, M., Das, T., et al., 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI 12). 15-28.

[15] Bei, Zh.D., Yu, Zh.B., Luo, N., et al., 2018. Configuring in-memory cluster computing using random forest. Future Generation Computer Systems. 79, 1-15.

[16] Tang, Zh., Zhang, X.Sh., Li, K.L., et al., 2018. An intermediate data placement algorithm for load balancing in Spark computing environment. Future Generation Computer Systems. 78, 287-301.

[17] Gonzalez-Lopez, J., Cano, A., Ventura, S., 2017. Large-scale multi-label ensemble learning on Spark. 2017 IEEE Trustcom/BigDataSE/ICESS, 893-900. DOI: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.328

[18] Harnie, D., Saey, M., Vapirev, A.E., et al., 2017. Scaling machine learning for target prediction in drug discovery using Apache Spark. Future Generation Computer Systems. 67, 409-417.

[19] Hernández, Á.B., Perez, M.S., Gupta, S., et al., 2018. Using machine learning to optimize parallelism in big data applications. Future Generation Computer Systems. 86, 1076-1092.

[20] Singh, H., Bawa, S., 2017. A MapReduce-based scalable discovery and indexing of structured big data. Future generation computer systems. 73, 32-43.

[21] Gavagsaz, E., Rezaee, A., Javadi, H.H.S., 2018. Load balancing in reducers for skewed data in MapReduce systems by using scalable simple random sampling. The Journal of Supercomputing. 74(7), 3415-3440.

[22] Gavagsaz, E., Rezaee, A., Javadi, H.H.S., 2019. Load balancing in join algorithms for skewed data in MapReduce systems. The Journal of Supercomputing. 75(1), 228-254.

[23] Bhatia, N., 2010. Survey of nearest neighbor techniques. arXiv preprint arXiv:1007.0085.

[24] Zhu, X., Zhang, L., Huang, Z., 2014. A sparse embedding and least variance encoding approach to hashing. IEEE transactions on image processing. 23(9), 3737-3750.

[25] Zhu, X., Zhang, S., Zhi, J., et al., 2010. Missing value estimation for mixed-attribute data sets. IEEE Transactions on Knowledge and Data Engineering. 23(1), 110-121.

[26] Connor, M., Kumar, P., 2010. Fast construction of k-nearest neighbor graphs for point clouds. IEEE

transactions on visualization and computer graphics. 16(4), 599-608.

[27] Liu, T., Moore, A.W., Gray, A., et al., 2004. An investigation of practical approximate nearest neighbor algorithms. Advances in neural information processing systems. 17.

[28] Raginsky, M., Lazebnik, S., 2009. Locality-sensitive binary codes from shift-invariant kernels. Advances in neural information processing systems. 22.

[29] Silpa-Anan, C., Hartley, R., 2008. Optimised KD-trees for fast image descriptor matching. 2008 IEEE Conference on Computer Vision and Pattern Recognition. 1-8. IEEE.

[30] Zhu, X.F., Huang, Z., Cheng, H., et al., 2013. Sparse hashing for fast multimedia search. ACM Transactions on Information Systems (TOIS). 31(2), 9.

[31] Triguero, I., Peralta, D., Bacardit, J., et al., 2015. MRPR: A MapReduce solution for prototype reduction in big data classification. neurocomputing. 150, 331-345.

[32] Du, M., Ding, S., Jia, H., 2016. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. Knowledge-Based Systems. 99, 135-145.

[33] Deng, Zh.Y., Zhu, X.Sh., Cheng, D.B., et al., 2016. Efficient kNN classification algorithm for big data. Neurocomputing. 195, 143-148.

[34] Moutafis, P., Mavrommatis, G., Vassilakopoulos, M., et al., 2019. Efficient processing of all-k-nearest-neighbor queries in the MapReduce programming framework. Data & Knowledge Engineering. 121, 42-70.

[35] Zhang, C., Li, F., Jestes, J., 2012. Efficient parallel kNN joins for large data in MapReduce. Proceedings of the 15th international conference on extending database technology. 38-49.

[36] Chatzimilioudis, G., Costa, C., Zeinalipour-Yazti, D., et al., 2015. Distributed in-memory processing of all k nearest neighbor queries. IEEE Transactions on Knowledge and Data Engineering. 28(4), 925-938.

[37] Sun, K., Kang, H., Park, H.H., 2015. Tagging and classifying facial images in cloud environments based on KNN using MapReduce. Optik. 126(21), 3227-3233.

[38] Maillo, J., Triguero, I., Herrera, F., 2015. A mapreduce-based k-nearest neighbor approach for big data classification. 2015 IEEE Trustcom/BigDataSE/ISPA. 2, 167-172. IEEE.

[39] Maillo, J., Ramírez, S., Triguero, I., et al., 2017. kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data. Knowledge-Based Systems. 117, 3-15.

[40] Wu, X., Zhang, C., Zhang, S., 2005. Database classification for multi-database mining. Information Systems. 30(1), 71-88.

[41] Ester, M., Kriegel, H.P., Sander, J., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd. 96(24),226-231.

[42] Sander, J., 2010. Density-Based Clustering, in Encyclopedia of Machine Learning, C. Sammut and G.I. Webb, Editors. Springer US: Boston, MA. pp. 270-273.

[43] Amini, A., Wah, T.Y., Saybani, M.R., et al., 2011. A study of density-grid based clustering algorithms on data streams. 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). 3, 1652-1656. IEEE.

[44] Wang, W., Yang, J., Muntz, R., 1997. STING: A statistical information grid approach to spatial data mining. Vldb. 97, 186-195.

[45] Gerlhof, C.A., Kemper, A., 1993. Partition-based clustering in object bases: From theory to practice. International Conference on Foundations of Data Organization and Algorithms. Springer, Berlin, Heidelberg. 301-316.

[46] MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1(14), 281-297.

[47] Johnson, S.C., 1967. Hierarchical clustering schemes. Psychometrika. 32(3), 241-254.

[48] Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: an efficient data clustering method for very large databases. ACM sigmod record, 25(2), 103-114.

[49] Mazzeo, G.M., Masciari, E., Zaniolo, C., 2017. A fast and accurate algorithm for unsupervised clustering around centroids. Information Sciences. 400, 63-90.

[50] Masciari, E., Mazzeo, G.M., Zaniolo, C., 2013. Pacific-asia conference on knowledge discovery and data mining. Springer, Berlin, Heidelberg. 111-122.

[51] Masciari, E., Mazzeo, G.M., Zaniolo, C., 2014. Analysing microarray expression data through effective clustering. Information Sciences. 262, 32-45.

[52] Ianni, M., Masciari, E., Mazzeo, G.M., et al., 2020. Fast and effective Big Data exploration by clustering. Future Generation Computer Systems. 102, 84-94.

[53] Muthukrishnan, S., Poosala, V., Suel, T., 1999. On rectangular partitionings in two dimensions: Algorithms, complexity and applications. International Conference on Database Theory. Springer, Berlin, Heidelberg. 236-256.

[54] Furfaro, F., Mazzeo, G.M., Saccà, D., et al., 2008. Compressed hierarchical binary histograms for summarizing multi-dimensional data. Knowledge and Information Systems. 15(3), 335-380.

[55] Arbelaitz, O., Gurrutxaga, I., Muguerza, J., et al., 2013. An extensive comparative study of cluster validity indices. Pattern Recognition. 46(1), 243-256.

[56] Caliński, T., Harabasz, J., 1974. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods. 3(1), 1-27.

[57] Zhang, T., Ramakrishnan, R., Livny, M. ,1996. BIRCH: an efficient data clustering method for very large databases. ACM sigmod record, 25(2), 103-114.

[58] Dua, D., Graff, C., 2017. Machine learning repository. University of California, Irvine, School of Information and Computer Sciences. Available online at: http://archive. ics. uci. edu/ml.

BILINGUAL
PUBLISHING CO.
Pioneer of Global Academics Since 1984

Tel: +65 65881289
E-mail: contact@bilpublishing.com
Website: ojs.bilpublishing.com

ISSN 2661-3220 01