SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2022, Vol. 65 122410:1–122410:10
https://doi.org/10.1007/s11432-021-3316-x

Low-time-complexity document clustering using memristive dot product engine

Houji ZHOU, Yi LI* & Xiangshui MIAO

Wuhan National Laboratory for Optoelectronics, School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan 430074, *China*

Received 14 May 2021/Revised 28 June 2021/Accepted 12 August 2021/Published online 14 January 2022

Abstract Document clustering has been commonly accepted in the field of data analysis. Nevertheless, the challenging issues for the clustering are the massive similarity measurement operations in the von Neumann architecture which result in huge time consumption. Memristive in-memory computing provides a brand-new path to solve this problem. In this article, utilizing the memristive dot product engine, we demonstrate a cosine similarity accelerated document clustering method for the first time. The memristor-based clustering method lowers the time complexity from $O(N \cdot d)$ of the conventional algorithm to $O(N)$ by executing similarity measurement in one step. Focused on the unit-length vectors, an in-situ normalization scheme for the stored vectors in the crossbar array is proposed to provide an efficient hardware training scheme and reduce the normalization steps during the clustering. Utilizing the BBCSport dataset as a benchmark, we further discussed the impact of the non-ideal factors in the memristors, including the available quantized states, the inevitable programming noise, and the device failure. Simulation results indicate that the 6-bit quantized states and 5% programming noise are acceptable for the document clustering tasks. Besides, high resistance states of the failure cells are recommended for higher performance clustering results.

Keywords linear-time clustering, cosine similarity, spherical K-means, memristor, in-memory computing

Citation Zhou H J, Li Y, Miao X S. Low-time-complexity document clustering using memristive dot product engine. Sci China Inf Sci, 2022, 65(2): 122410, https://doi.org/10.1007/s11432-021-3316-x

1 Introduction

Text in documents is the mainly accepted form of data storage in the information world. Organized documents not only improve the efficiency of data searching but also contribute to data analysis and knowledge extraction. Document clustering divides unordered documents into categories with similar characteristics which can be further used in the applications for the data filter, dimension reduction and topic extraction [1–4] (Figure 1(a)). As a fast and high-quality clustering method in the data processing field, K-means and its derivative algorithms have been commonly studied and perform well in document clustering tasks. The kernel part of K-means algorithm focuses on the similarity measurement between the input vector and the prototype centers. Documents in close similarities are placed together as a category. However, massive data streams, as well as the increasing data dimensions, lower the clustering efficiency which opposes the real-time applications. The reason is, intensive serial data communication in the separate storage part and processing unit of the von Neumann architecture hinders the computational efficiency, especially in the similarity measurement processes. The emerging in-memory computing based on the analog memristors is regarded as a reliable solution for this problem by executing the vector-matrix multiplication (VMM) in a single step. In other words, a memristor array can act as a dot product engine (DPE) and significantly accelerates the VMM operations [5–7]. Utilizing the DPE, document clustering tasks are expected to achieve great acceleration by realizing efficient similarity calculation processes. Recent researches of similarity computation acceleration in the memristor array concentrate much more on the distance calculation in the Euclidean space. Ref. [8] reported a competitive learning model which utilizes the results of the dot product in the crossbar array to represent the similarities. This

* Corresponding author (email: liyi@hust.edu.cn)

© Science China Press and Springer-Verlag GmbH Germany, part of Springer Nature 2022

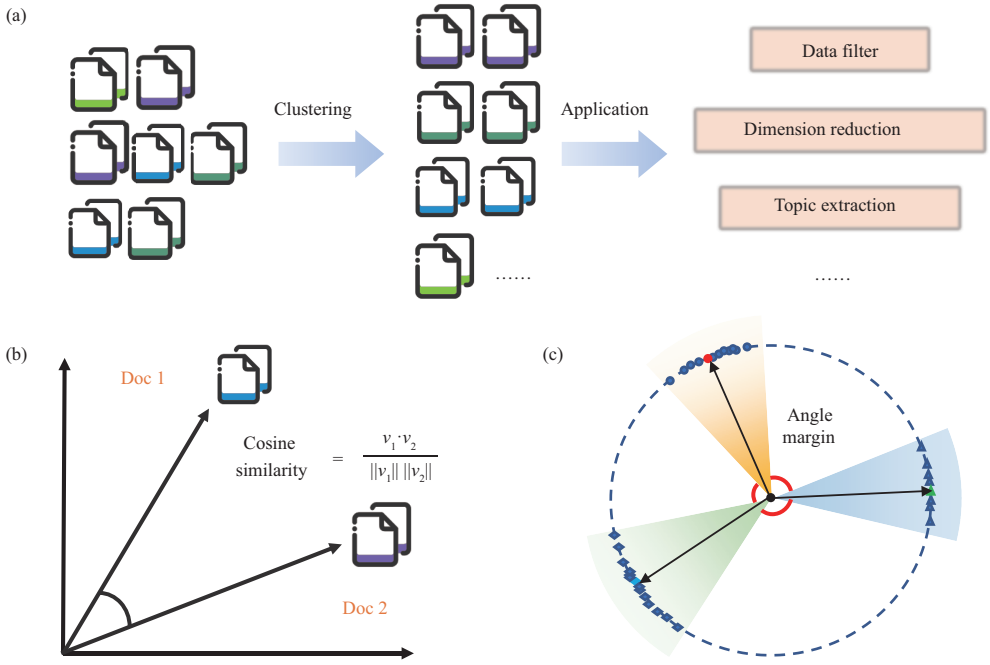

Figure 1 (Color online) Illustration of document clustering. (a) The clustering method divides the unorganized text documents into different groups and finally contributes to various applications including the data filter, dimension reduction and topic extraction; (b) the general cosine similarity in the data clustering tasks measures the angle of different hyper-vectors; (c) the spherical K-means method aims to maximize the angle margins in different groups for the points distributed on the sphere.

publication proposes a new analog similarity computing paradigm based on memristors. In [9], the dot product between the input vector and the normalized dictionary element vector is utilized to measure the Euclidean distance for the sparse coding and the acceleration is experimentally verified in the memristor array. However, this kind of simplification is limited by the unit-length restriction of the vector. Later researches pay much more attention to the similarities of the unnormalized vectors, such as [10, 11], to obtain the accurate distances in the Euclidean space. In [10], the kNN (k-nearest neighbor) algorithm is realized by calculating the complete Euclidean distance in the crossbar array. And Ref. [11] revealed the outperformance of the truly calculated Euclidean distance than the dot product represented one during the K-means clustering in the memristor array.

Although more attention is paid to the acceleration of Euclidean distance, the sparse hyper-vectors of the document representation highlight the cosine similarity during the clustering. Cosine similarity measures the angle of two vectors while ignoring their amplitude and sparsity (Figure 1(b)) which compensates for the short of Euclidean distance. For the conventional cosine similarity calculation, the L2-norm of the two vectors is in the position of the denominator which makes it complex to be implemented in the hardware [12]. The L2 normalization strategy of the vectors is a basic and important scheme to obtain high-quality embedding features and stable training processes in artificial neural networks [13,14]. For the L2-normalized hyper-vectors of the documents, the cosine similarity will be simplified to the form of the inner product of the vectors. Thus, the highly parallel VMM in the DPE makes it a constant time level to calculate the similarities of the input vector and the stored matrix [8, 9, 15]. The spherical K-means (SKM) algorithm focuses on these unit-length vectors which are distributed on the unit hypersphere (Figure 1(c) shows a two-dimensional example) and realizes more effective clustering [16, 17]. By mapping the prototype vectors to the memristors array, the SKM, which heavily relies on the cosine similarities measurements, can be greatly accelerated. However, the weight vector will be numerical updated and the unit-length restriction of the vector will be broken during the training. A challenging issue is to ensure the similarity between the input vectors and the stored weights could be well calculated. The solution for this problem could be the in-situ normalization in the crossbar array or a general cosine similarity calculation method for arbitrary vectors. The former one could be much more suitable for SKM taking advantage of the original unit-length vectors of the documents but still remains unresolved.

In this article, utilizing the memristive dot product engine, we demonstrate a cosine similarity accelerated spherical K-means clustering method for the first time and the document clustering task is employed

as a typical benchmark. This hardware based clustering scheme lowers the temporal complexity from the $O(N \cdot d)$ to the linear-time $O(N)$ level by accelerating the similarity measurements. Attractively, an in-situ vector normalization method in the memristor crossbar array is processed to ensure the validity of dot products as similarities and leads to a flexible and efficient hardware training scheme for document clustering. Simulated with the BBCSport dataset, the simplified normalization method performs well in the clustering tasks and shows a comparable success rate with the software-based one. Further simulation discusses the influence of the non-ideal factors of the memristor device. Simulation results show that 6-bit conductance states, 5% write noise, and high resistance states of the failure cells are recommended for the document clustering tasks. We believe this low-time-complexity document clustering method will contribute to future text analysis applications in the big data area.

2 Background

Documents collected in the real world consist of a sea of symbols and characters, some of which may be useless even unfavorable for the data processing. Thus, data filtering will be performed firstly before the mathematical analysis and produces purely texts. Then these texts in documents will be split into words and further studied. In this part, we mainly give a brief introduction to the operations following the word splitting, including the document representation and the spherical K-means algorithm for document clustering.

2.1 Document preprocessing by TF-IDF method

The documents must be transformed into mathematical models before being processed for the semantic of the documents cannot be recognized by the computers directly. As a pre-processing step of text analysis, the transferring model characterizes the features of the text files in the form of hyper-vectors and determines the quantity of the following analytical operations. Vector space model (VSM) utilizes hyper-vectors that have the same dimensions with the unique words (known as terms) to represent each document in the space [18, 19]. Normally, the frequency of terms is used to represent the weights in the hyper-vectors, respectively. This may produce inaccuracy for the reason that the stop-words, such as "is", will have high weights but make no sense. The term frequency-inverse document frequency (TF-IDF) model can avoid this problem by highlighting the words which appeared in several documents but were rarely shown in the others [20]. The mathematical model is shown as

$$w_{mt} = (1 + \log(f_{mt})) \times \log\left(1 + \frac{N}{f_t}\right), \tag{1}$$

where f_{mt} denotes the frequency of term t in mth document, N is the total number of the documents and f_t is the number of the documents where the term t has appeared. Thus the hyper-vector of the mth document is represented by $w_m = (w_{m1}, w_{m2}, \ldots, w_{mn}), n = 1, \ldots, N$. Especially, the vector w_m will be L2-normalized to lower the influence of the vector length for the similarity comparison. Besides, some densification models, such as Doc2Vec [21], hashing [22, 23], can be used after the TF-IDF model to lower the data dimensions which can reduce the pressure for data storage.

2.2 SKM algorithm

K-means algorithm has been wildly studied for its simple implementation and fast convergence. The aim of this algorithm is to find K prototype vectors in the clustering data space to represent the different classes while samples will be divided into K classes represented by the closest prototype vectors. Here K is a pre-set hyperparameter representing K clusters. The SKM algorithm is developed from the conventional K-means and is proposed to process the clustering tasks where the data vectors are L2-normalized. Thus, the similarity measurement in SKM is simplified to the dot product between the input samples and prototype vectors which exhibits the essence of cosine similarity [24]. Figure 2 shows the detailed training processes of SKM. The temporal complexity of SKM is determined by $O(K \cdot M \cdot N \cdot d)$ where M indicates the max cycling number for the end condition and the d is the dimensions of the data vector. For most of the document clustering tasks, $K \ll M \ll N$, and the time complexity can be approximated as $O(N \cdot d)$. The factor d is mostly introduced by the cosine similarity calculation for

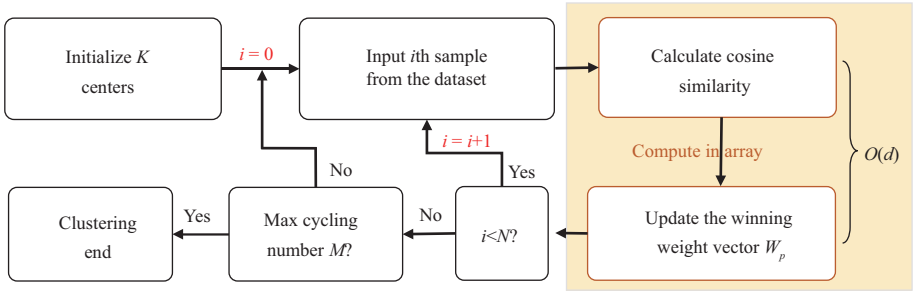

Figure 2 (Color online) Flowchart of the spherical K-means algorithm. The similarity calculation, as well as the weight updating, produces great time consumption during the training. These two parts are indispensable and achievable to be accelerated in the hardware platform.

high dimensional vectors during the training. In this article, using a DPE to accelerate the similarity calculation as well as maintaining its validity to reduce the temporal complexity will be mainly discussed.

For the unit-length vectors, cosine similarity will be achieved by the multiplication of the input vector and the weight matrix in the clustering. The results of the multiplication determine the prototype vector to be updated by the winner takes all principle which indicates that the prototype vector corresponding to the maximum value will be further updated. The updating rule is determined by the following formulas [25]:

$$w'_{\text{new}} = w + \eta x_i, \tag{2a}$$

$$w_{\text{new}} = \frac{w'_{\text{new}}}{||w'_{\text{new}}||}, \tag{2b}$$

where w denotes the winning vector to be updated, η is the learning rate, and x_i is the input sample during each training process. w_{new} indicates the finally updated vector for each updating step. Notably, Eq. (2a) determines the amplitude tendency during the training while Eq. (2b) is the weight normalizing step that keeps the updated vector L2-normalized.

3 Design of memristor based SKM

In this part, we mainly explain the mapping rule from the SKM method to the memristive dot product engine. And then the training, as well as the data updating, for the document clustering in the memristor array will be illustrated.

3.1 Mapping SKM to DPE

The SKM can be illustrated by the network structure shown in Figure 3(a). The input layer indicates the input sample to be compared in the original data space and the outputs are the classes. Weights connected to an output neuron represent a prototype vector (the orange lines in Figure 3(a) for example). In order to accelerate the clustering process and reduce time consumption, the memristor crossbar array is employed as the parallel VMM engine during the clustering (Figure 3(b)). Here, the input values of the net are transformed to the voltage pulses which vary in amplitude for different values. The weight matrix is mapped to the conductance of the memristor crossbar array and each column of the array indicates one document prototype vector. Although the limited size of the fabricated crossbar array cannot fit the weight storage greatly, the data splitting and block storage strategy provide inspiring schemes for high-dimensional data computing in non-volatile memory [26,27]. The initial values of the weight matrix are given by the randomly selected K hyper-vectors from the original dataset and programmed by the analog property of the memristor with the target-aware method [28]. The similarity computation during the training relies on the forward reading process of the array which realizes the dot product of the input pattern and the weight matrix. The output currents represent the cosine similarities between the input document and the stored document prototypes. Then the comparing circuit compares the output currents and obtains the biggest one which indicates the most similar document prototype to the input.

Figure 3(c) shows the design of the comparing circuit [10,29]. The output currents from the array are firstly transformed to the voltages linearly by the trans-impedance amplifier (TIA). The voltages will be

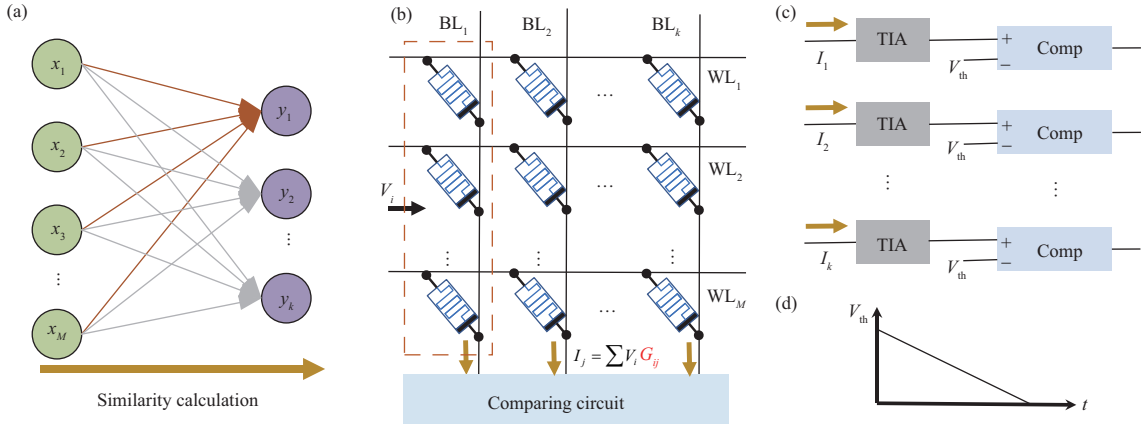

Figure 3 (Color online) Schematic of mapping the SKM to the memristor crossbar array. (a) The network structure of the clustering tasks. (b) Mapping the network to the memristor array. Each column of the array keeps the values of one prototype vector. (c) Comparing circuit with signals for winner takes all. (d) The threshold voltage setting for the winner takes all comparison.

then transferred to the comparator group and compared with the threshold voltage $V_{\rm th}$. The amount of the TIAs and comparators is linearly proportional to the number of categories and is much less than those in the neural networks, such as convolution neural network (CNN) [30], resulting in the acceptable area and power consumption overhead. Figure 3(d) shows how the threshold voltage $V_{\rm th}$ is generated during the winner takes all compassion. As we know, the more similar the input pattern and prototype vector are, the higher voltage the TIA will produce. When $V_{\rm th}$ decreases gradually from a high level, a voltage generated by the closest prototype vector will be larger than the $V_{\rm th}$ firstly. The corresponding comparator will be activated and produces a high voltage pulse indicating the winning vector. This activated pulse will be feedback to the controller for the weight updating. There only exists one vector winning the competition and this vector represents the final cluster for the input when the training ends.

3.2 Hardware training of SKM

The unresolved problem for the training of the SKM is the weight updating scheme. As we have mentioned before, the weight updating for SKM includes two main steps: the numerical updating (Eq. (2a)) and the vector normalization (Eq. (2b)). For the numerical updating, the increment of the weight vector is determined by the known input vector and the weight updating can be achieved by modulating the analog property of the memristor nodes [31–33]. However, for the normalizing step, the stored weights require the in-situ updating by dividing the vector length after numerical updating which depends on complex external circuits or the mixed software based assistance [12,34]. This will increase the complexity of the clustering system and result in much more time consumption. To handle this problem, we developed an in-situ vector L2 normalization scheme in the crossbar array for SKM which reduces the complexity of updating and ensures that the cosine similarity calculation is appropriate accurately during the training.

In more detail, when the learning rate η keeps a small value, the length of the updating vector changes slightly according to (2a) for each numerical updating step. Therefore, the similarity comparison result can hardly be affected by a single numerical updating and the robustness of the clustering system will ignore these errors. However, as the number of numerical updating steps increases, the length changes in this vector accumulate to a magnitude that can actually affect the accuracy of the similarity computation. In this circumstance, an influential control parameter α is introduced to measure the degree of length accumulation. When the length of the stored vector is smaller than the threshold length $L_{\rm th}$, where $L_{\rm th} = 1 + \alpha$, the variation of the vector length L_p is considered negligible to the influence of the cosine similarity and the normalization step is not required. Otherwise, the vector will be L2-normalized. Further, when L_p is considered larger than $L_{\rm th}$, L_p actually exceeds $V_{\rm th}$ by a very small amount for the reason that the learning rate η is a small value and the length of input vector x keeps one. Thus, for the updating step, the constant value $L_{\rm th}$ is regarded as the length of the vector to be normalized and $L_p \approx L_{\rm th} = 1 + \alpha$. If α is set to a small value, the updating step in (2b) can convert to (3) with the Taylor expansion.

$$w_{\rm new} = w_p - \alpha w_p. \tag{3}$$

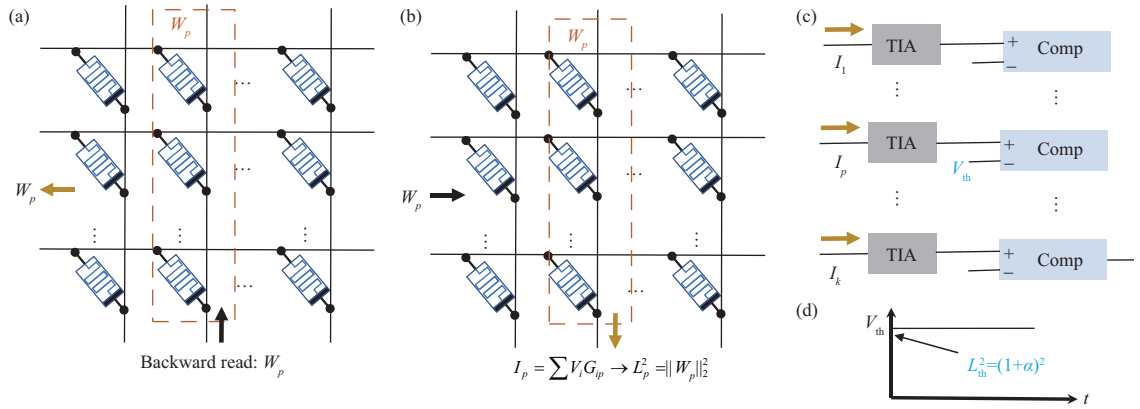

Figure 4 (Color online) Illustration of comparing the length of L_p with the threshold length $L_{\rm th}$ in the memristor array. (a) A backward read step to obtain the selected stored vector W_p in the array; (b) a forward read step to calculate the L_p^2; (c) signals applied to the comparing circuit for the length comparing; (d) the threshold voltage setting for the length comparing.

This equation is the same form as the numerical updating step in (2a) and shares similar updating operations which ensure consistent data updating process. Here, w_p is the weight vector before the normalization. Therefore, the traditional normalization scheme is greatly simplified by the influential control parameter α and provides an in-situ updating solution.

Figure 4 shows the operations to compare the length of the L_p with the threshold length $L_{\rm th}$ in the memristor array. A backward read voltage is applied to the BL_p and the output currents indicating the stored weights will be obtained in the crossbar array (Figure 4(a)). Data conversion and fast temporary storage blocks are required for the backward reading which can be commonly seen in the in-situ training systems [31, 32, 35]. Then the readout values are encoded with forward read voltages and applied to the pth column. The output current indicates the length L_p^2 (Figure 4(b)). The calculated L_p^2 will further be compared with the threshold length in the comparing circuit. Figure 4(c) shows how to set the threshold voltage $V_{\rm th}$ during the length comparison. Here, we focus much more on the pth column in the output currents while the currents in the other column will be suppressed by applying bias voltages to the array [11]. In the comparing circuit for length comparison, the threshold voltage $V_{\rm th}$ takes a constant value which indicates the $L_{\rm th}^2 = (1+\alpha)^2$ (Figure 4(d)). If the voltage transferred by the pth TIA is larger than the threshold voltage, the output of the comparator will generate a positive pulse and will give feedback to the controller to start L2 normalization of the weight.

4 Validation and analysis for DPE based document clustering

In this part, the validity of the proposed clustering scheme as well as the impact of the non-ideal factors of the DPE is discussed. The BBCSport dataset, consisting of five categories including athletics, cricket, football, rugby, and tennis, is employed as a benchmark to investigate the memristive document clustering tasks. The simulation framework of document clustering is mainly realized by the steps shown in Figure 2. Key steps, including the similarity measurement and weight updating, are achieved in the modeled DPE where the analog properties of memristors are simulated with the python platform [36]. Figure 5(a) shows the clustering results where the original data is downscaled to two dimensions with the principal component analysis. The five categories can be almost perfectly distinguished in different areas of the two-dimensional plane which indicates the great clustering results with the accuracy of 96.06% against the software obtained 96.7%. The accuracy evolution as well as the average loss during the training is displayed in Figure 5(b). The final accuracy, as well as the evolution traces, of the DPE-based simulation, shows comparable values with the software-based one. However, the average loss of the memristive method cannot be optimized to the ideal level for the quantized states during the weight mapping. Figure 5(c) shows part of the updating tendency for the length of a weight vector during training. The weight vector can be perfectly normalized by (3) once its length reaches the threshold length (here, the threshold length is 1.3). Thus, the in-situ normalization in the array is proved valid and reliable.

Two main aspects which are influential for the DPE-based clustering system are considered, including the parameters in the hardware clustering scheme and the non-ideal factors of the memristor cells in

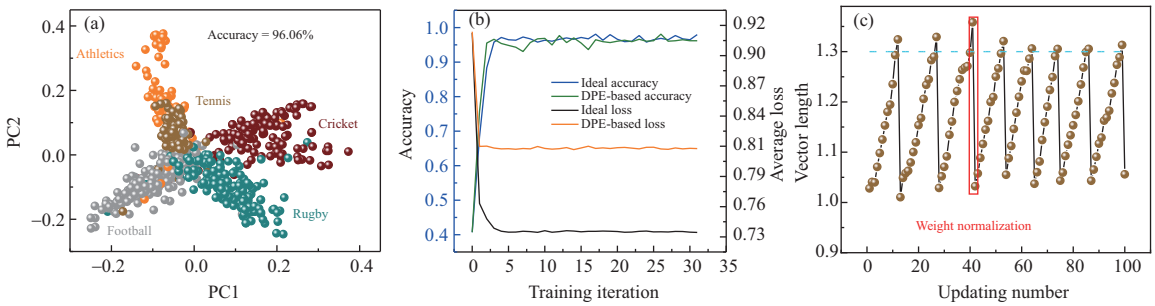

Figure 5 (Color online) Clustering results of the BBCSport dataset with the simulated memristors. (a) Demonstration of well-trained classes painted with various colors based on PCA method; (b) evolution traces of the accuracy as well as the average loss during the training; (c) the length updating during the training process for a weight vector.

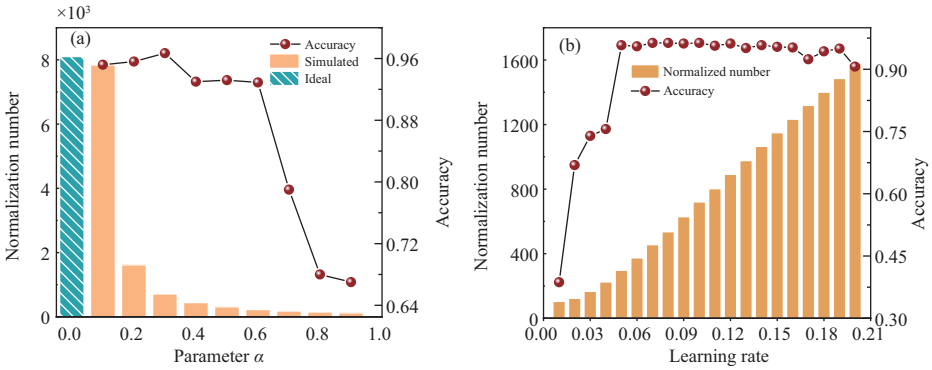

Figure 6 (Color online) Simulation results of the influential factors to the DPE-based document clustering. (a) The influence of the control parameter α on the clustering process; (b) simulation results of the impact of learning rate on the clustering process.

DPE. As we have discussed earlier, the control parameter α, as well as the learning rate η, will affect the similarity results according to (3). Figure 6(a) shows the simulation results of the number of normalization steps as well as the clustering accuracy with the control parameter α. Clearly, as α increases, the number of the normalization steps is greatly reduced for the reason that the accumulation of the numerical updating of the winner vector takes much more time to break through the limitation of $L_{\rm th}$. This will lessen the operations in the memristor crossbar array and bring much more hardware-friendly. What is much more essential is that when α is smaller than 0.3, the success rate for data clustering makes little difference to the software-based one. Even if α gets much bigger (smaller than 0.6), the document clustering task still obtains acceptable results (around a 90% success rate). Yet once α gets too large, it is not suitable to apply Taylor expansion to (2b) and the updating progress with (3) will create much inaccuracy and result in a dramatic decline in the success rate. Meanwhile, Figure 6(b) is the influence of the learning rate on the normalization number and the accuracy. When η is small (less than 0.05), data quantization in the memristor will lead to invalid numerical updating determined by (2a) and result in a reduced success rate. But a too-large learning rate (larger than 0.15 here) will make the L_p much larger than $L_{\rm th}$ before the data normalization step and make the assumption $L_{\rm th} \approx L_p$ invalid and finally degrade the success rate. Apart from that, η will determine the number of the normalization step linearly under the same α for the linear increment of vector length. Thus, the normalization steps can be partly reduced by increasing the learning rate appropriately.

More simulations evaluate the device properties of the memristor and provide a guideline for future hardware implementation. During the in-situ training, the instability of the memristor array may cause device failures which lead to write errors and affect the results, especially for the hyper-dimensional data storage. Here, the stuck-at-fault (STA) defect model is used to evaluate the influence of the device failure of memristors. Figure 7(a) shows the influence of the memristor array under different ratios of stuck at 0 (the maximum conductance). The result shows that less than 30% stuck at 0 failure of the memristor nodes can still ensure the success rate of clustering higher than 80%. That is, the zero values in the sparse hyper vectors match the stuck at 0 failure and degrade the influence of cell failure. But if the values are stuck at other states of the memristor cells which map non-zero values, the length of vectors

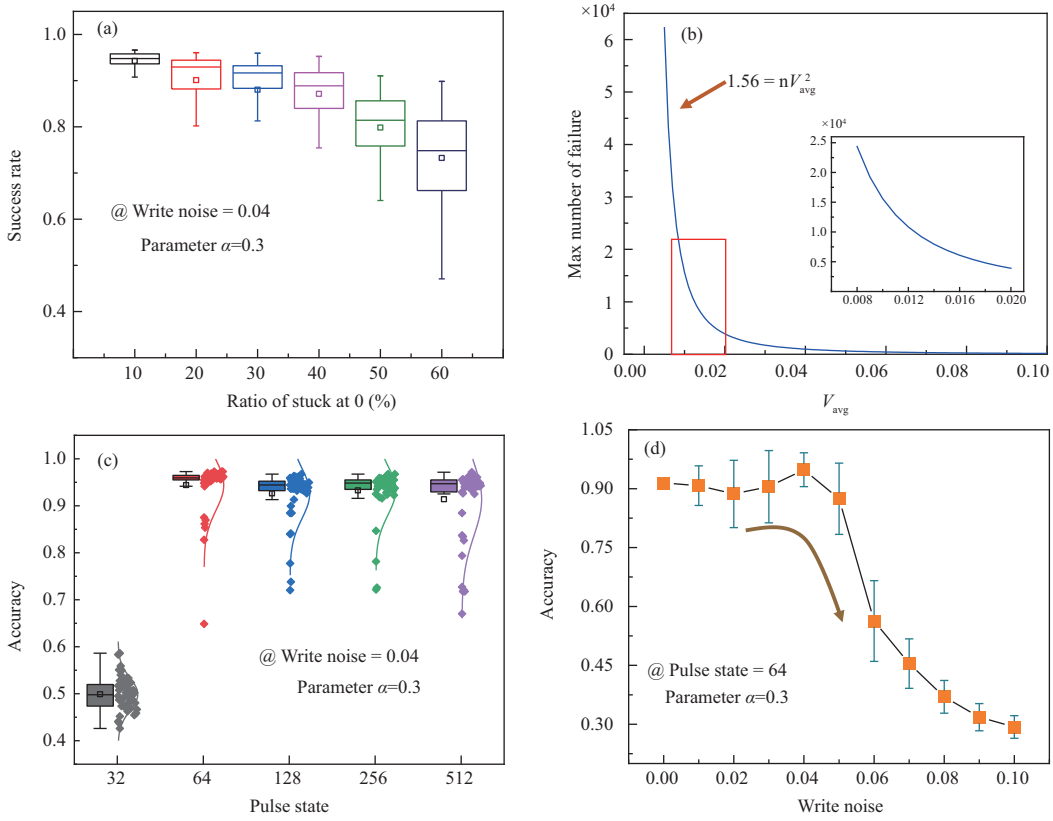

Figure 7 (Color online) Simulation results of the influential factors to the DPE-based document clustering. (a) Influence of the memristor array under different ratios of stuck at 0; (b) relationship between the maximum number of failure cells and the average value V_{avg} of the failure cells; (c) accuracy distribution under different pulse states; (d) influential discussion of the write noise to the clustering results.

will be easily larger than the unit one. Assume the ratio of the failure memristors is r and the average mapped values of the failure memristor is V_{avg}. Under parameter $\alpha = 0.6$, the relationship between r and V_{avg} obeys $(1+0.6)^2 - 1 \geqslant N \cdot r \cdot V_{\text{avg}}^2 = nV_{\text{avg}}^2$. Where N is the total number of the memristor cells and n is the number of failure cells. Figure 7(b) illustrates the acceptable maximum number of failure cells under different V_{avg} with the BBCSport dataset. Results show that the maximum number of failure cells degrades dramatically when V_{avg} has a slight increment. This indicates that the performance of SKM is sensitive to the defect of stuck at non-zero values of device failure. In general, to avoid the influence of the data failure, the failure memristor cells are suggested to be set to high resistance states (low mapped values).

Another two main aspects including the conductance states of the memristor and the write noise for conductance modulation are further discussed in detail. Figure 7(c) shows the distribution of the success rate of the clustering tasks under different pulse states of the memristor. Being larger than 6 bit, the success rates remain at a similar level and comparable with the software obtained (96.7%). This indicates that the 6-bit conductance states can provide distinguishable conductance levels to store the TF-IDF values of the documents. The high-bit requirement of the memristors has been reported achievable in large-scale crossbar and fabrication which will support our implementation [15, 32]. In addition, low-bit memristor cells can also achieve high-precision calculations through the cooperation of multiple chips [26, 37]. Figure 7(d) shows the inevitable write noise of the device modulation. When the write noise is small, the conductance can be modulated much closer to the ideal one. But once it gets too large (larger than 0.04), the in-situ modulation determined by (2a) and (3) will deviate from the original updating direction and make the in-situ training inaccurate.

Three more open-accessed document datasets, including the BBC news, Google snippets, and the 20 Newsgroups, are used to discuss the complexity of the algorithm. The detailed information of the datasets is shown in Table 1. The factor N_s/N_d^2 is used to measure the time complexity in the CPU-based systems where N_s denotes the number of operations in the computer and N_d is the number of the documents.

Table 1 The comparison of the required operations both in the software based system and in the memristive document clustering[a)]

Dataset	N of classes (N_c)	N of documents (N_d)	N of terms (N_t)	Operations in software (N_s)	N_s/N_d^2 ($O(n^2)$)	Operations in DPE (N_m)
BBCSport	5	737	13050	∼48.1 M	∼88.5	$3N_d + N_r$
BBC news	5	2225	29395	∼327 M	∼66.0	
Google Snippets	8	12340	30642	∼3.02 G	∼19.9	
20 Newsgroups	20	18846	34118	∼12.9 G	∼36.2	

a) N_r denotes the normalization numbers.

The results show that the N_s/N_d^2 keeps the tens of levels for different datasets which indicate that the analytical temporal complexity $O(N \cdot d)$ of the original algorithm approximates $O(n^2)$ and exhibits high temporal complexity. For the memristive document clustering method, the number of the operations keeps $3N_d + N_r$ which shows a linear time complexity. Here, N_r denotes the normalization numbers and is always smaller than N_d according to the former simulation. $3N_d$ indicates the operations in the crossbar array, including the cosine similarity calculation (one step) and the vector length computation (two steps) for each sample. Thus the memristive document clustering method provides a much higher time efficiency than the conventional one, which will facilitate its usages for data analysis in the era of data explosion.

5 Conclusion

In conclusion, focused on the cosine similarity acceleration, a memristive document clustering method utilizing the spherical K-means is proposed, which lowers the clustering complexity from $O(N \cdot d)$ to $O(N)$. Moreover, an in-situ vector normalization method in the memristor array is introduced to provide a simple and efficient hardware training scheme for clustering. Simulation results show that this method not only provides a reliable clustering success rate but also leads to a more hardware-friendly feature by reducing the normalization steps. The impacts of non-ideal factors of memristive devices, including the device failure, the quantized states, and the write noise, are discussed in detail, which provides the guidelines for future hardware implementations. In short, this clustering method demonstrates a much more efficient way for text data analysis and will benefit the massive data processing tasks.

Acknowledgements This work was supported by National Key Research and Development Plan of MOST of China (Grant No. 2019YFB2205100), National Natural Science Foundation of China (Grant Nos. 61874164, 92064012, 61841404), and the support of Hubei Key Laboratory for Advanced Memories, Hubei Engineering Research Center on Microelectronics, and Chua Memristor Institute.

References

1 Wenskovitch J, Crandell I, Ramakrishnan N, et al. Towards a systematic combination of dimension reduction and clustering in visual analytics. IEEE Trans Visual Comput Graph, 2018, 24: 131–141
2 Abualigah L M, Khader A T, Al-Betar M A, et al. Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Syst Appl, 2017, 84: 24–36
3 Xie X, Wang B. Web page recommendation via twofold clustering: considering user behavior and topic relation. Neural Comput Applic, 2018, 29: 235–243
4 Trappey A J C, Trappey C V, Hsu F, et al. A fuzzy ontological knowledge document clustering methodology. IEEE Trans Syst Man Cybern B, 2009, 39: 806–814
5 Hu M, Graves C E, Li C, et al. Memristor-based analog computation and neural network classification with a dot product engine. Adv Mater, 2018, 30: 1705914
6 Zhang T, Yang K, Xu X, et al. Memristive devices and networks for brain-inspired computing. Phys Status Solidi RRL, 2019, 13: 1900029
7 Ma W, Zidan M A, Lu W D. Neuromorphic computing with memristive devices. Sci China Inf Sci, 2018, 61: 060422
8 Yu S M. Orientation classification by a winner-take-all network with oxide RRAM based synaptic devices. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2014
9 Sheridan P M, Cai F, Du C, et al. Sparse coding with memristor networks. Nat Nanotech, 2017, 12: 784–789
10 Jiang Y, Kang J, Wang X. RRAM-based parallel computing architecture using k-nearest neighbor classification for pattern recognition. Sci Rep, 2017, 7: 45233
11 Jeong Y J, Lee J, Moon J, et al. K-means data clustering with memristor networks. Nano Lett, 2018, 18: 4447–4453
12 Yin S, Kim M, Kadetotad D, et al. A 1.06-μW smart ECG processor in 65-nm CMOS for real-time biometric authentication and personal cardiac monitoring. IEEE J Solid-State Circ, 2019, 54: 2316–2326
13 van Laarhoven T. L2 regularization versus batch and weight normalization. 2017. ArXiv:170605350
14 Wang F, Xiang X, Cheng J, et al. NormFace: L2 hypersphere embedding for face verification. In: Proceedings of the 25th ACM international conference on Multimedia, 2017
15 Yao P, Wu H, Gao B, et al. Face classification using electronic synapses. Nat Commun, 2017, 8: 15199
16 Endo Y, Miyamoto S. Spherical K-means++ clustering. In: Modeling Decisions for Artificial Intelligence. Berlin: Springer, 2015. 9321: 103–114

17 Kim H, Kim H K, Cho S. Improving spherical k-means for document clustering: fast initialization, sparse centroid projection, and efficient cluster labeling. Expert Syst Appl, 2020, 150: 113288
18 Salton G, Wong A, Yang C S. A vector space model for automatic indexing. Commun ACM, 1975, 18: 613–620
19 Ravindran R M, Thanamani D A S. K-means document clustering using vector space model. Bonfring Int J Data Min, 2015, 5: 10–14
20 Curiskis S A, Drake B, Osborn T R, et al. An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit. Inf Process Manage, 2020, 57: 102034
21 Lau J H, Baldwin T. An empirical evaluation of doc2vec with practical insights into document embedding generation. In: Proceedings of the 1st Workshop on Representation Learning for NLP, 2016. 78–86
22 Sadowski C, Levin G. SimHash: Hash-Based Similarity Detection. Techreport, 2007. 1–10
23 Chaidaroon S, Fang Y. Variational deep semantic hashing for text documents. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017. 75–84
24 Xia P, Zhang L, Li F. Learning similarity with cosine similarity ensemble. Inf Sci, 2015, 307: 39–52
25 Duwairi R, Abu-Rahmeh M. A novel approach for initializing the spherical K-means clustering algorithm. Simul Model Practice Theor, 2015, 54: 49–63
26 Zidan M A, Jeong Y J, Lee J, et al. A general memristor-based partial differential equation solver. Nat Electron, 2018, 1: 411–420
27 Peng X, Huang S, Luo Y, et al. DNN+NeuroSim: an end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2019
28 Yao P, Wu H, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
29 Fernando B R, Hasan R, Taha T M. Low power memristor crossbar based winner takes all circuit. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), 2018
30 Lin P, Li C, Wang Z, et al. Three-dimensional memristor circuits as complex neural networks. Nat Electron, 2020, 3: 225–232
31 Wang Z, Li C, Lin P, et al. In situ training of feed-forward and recurrent convolutional memristor networks. Nat Mach Intell, 2019, 1: 434–442
32 Li C, Belkin D, Li Y, et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nat Commun, 2018, 9: 2385
33 Liu X, Zeng Z, Wunsch II D C. Memristor-based LSTM network with in situ training and its applications. Neural Networks, 2020, 131: 300–311
34 Perera D G, Kin F L. On-chip hardware support for similarity measures. In: Prcoeedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2007
35 Ambrogio S, Narayanan P, Tsai H, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature, 2018, 558: 60–67
36 Chen P-Y, Peng X C, Yu S M. NeuroSim+: an integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In: Prcoeedings of IEEE International Electron Devices Meeting, 2018
37 Zhu Z H, Sun H B, Lin Y J, et al. A configurable multi-precision CNN computing framework based on single bit RRAM. In: Proceedings of the 56th Annual Design Automation Conference, 2019. 1–6