Scalable Learning Through Error-correcting Codes based Clustering in Autonomous Systems

Ganapathy Mani*, Bharat Bhargava†
Department of Computer Science & CERIAS
Purdue University
West Lafayette, USA
manig@purdue.edu, bbshail@purdue.edu*†

***Abstract*—Intelligent Autonomous systems (IAS) continuously receive large streams of diverse data from numerous entities operating and interacting in their environment. It is vital that the learning models in IAS to scale up to the new and unknown data items that were not present in the training or testing datasets. Scalable learning is nothing but a method to achieve maximum classification without rejecting any unknown data item as anomalies. In this paper, we present Perfect Error-correcting Codes (PEC) clustering technique to approximate the classes of multi-feature data items by reversing standard forward error correction coding. Approximating classes problems generally arise in information systems that are processing fuzzily cataloged data items. These data items can be classified by applying binary vectors to their corresponding features (1: feature is present or 1: feature is absent) to obtain message words. These codewords can be used as cluster centers. In PEC clustering, binary vectors of 23 bits are mapped into codewords (labels or indices) of 12 bits. Two binary vectors with the Hamming distance of 2 will have a few common labels thus classified accordingly. PEC clustering has 2^{23} codeword space, which makes it ideal for scalability in clustering of thousands of categories. With reasonable redundancy, the clustering can be accomplished in $\mathcal{O}(N)$ time. In addition, we present an information processing model for on-the-fly processing of data streams with multi-processor pipeline: Read, Analyze, and Toggle (RAT) model.**

***Keywords*-error-correcting codes; clustering; scalable learning; deep learning; fuzzy logic; autonomous systems;**

I. Introduction

Classification problems generally arise in dynamic environments with many classes [1], in which IAS operate. As Figure 1 shows, the large data streams can only be analyzed in indirect ways where they can be either sampled or classified into clusters. But sometimes, sampling may produce very skewed results [2]. Hence, clustering can be a right option to process big data. One well-known example is recognizing thousands of visual data items—one of the biggest challenges in computer vision and big data processing. The vast number of classes make the conventional one-verses-all multi-class paradigm to be very expensive in terms of time and space. The time complexity grows linearly with number of categories, which makes training and testing prohibitive for real-time practical applications such autonomous robots. These autonomous systems require high throughput with low latency.

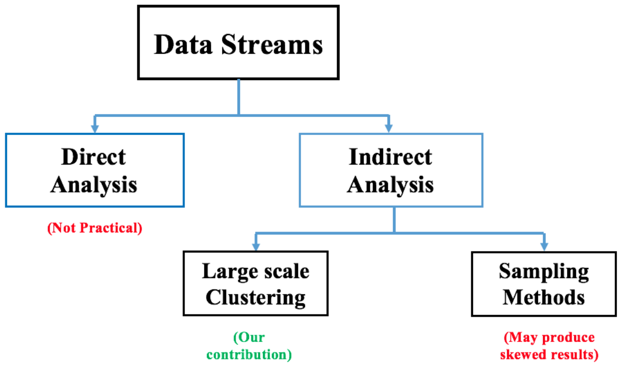

Figure 1. Categorizing large data streams

Most classification methods that are used for classifying data with thousands of diverse categories merge into models built based on trees [3] [4]. In [4], the authors propose a state of the art *label tree*-based model that performs significantly well during testing. Each node in a label tree is associated with a few classes and a classifier (linear) that decides which branch to follow. But this model has several limitations in terms of learning. The technique involves one-vs-all classifiers for training and the classifier is costly. The label trees are allowed to overlap, which increases the complexity. Similar to this technique, other tree-based models [5] [6] utilize expensive multi-class classifiers.

In order to provide scalability to the ever growing number of classes in dynamic and unpredictable environment, labeling categories and creating clusters based on those categories must be (a) fast and efficient to process data streams and (b) accurate to be used for practical applications. In this paper, we present a clustering scheme—*PEC* clustering—that is based on reversing the standard forward error-correcting codes. With the combination of 23 bits in a binary vectors 2^{23} clusters (categories) can be created. We also propose a data stream processing structure—*RAT* multi-processor pipeline—that can process data items in parallel.

The rest of the paper is organized as follows. Section II presents the related work, section III summarizes the PEC clustering technique with the explanation of perfect codes and parallel processing of data streams, section IV evaluates the scheme and compares the time complexities with other clustering techniques, we finally conclude along with our future work in section V.

II. Related Work

Categorizing large number of image data items has received significant attention in computer vision after datasets of very large object classes such as ImageNet [7] became publicly available. One category of work focuses on efficient feature categorization and achieving significant performance increases [8] [9]. Another category focuses on optimizations using tree-based models [10] [11]. Recent advances in deep learning has lead to the proposal of state-of-the-art performance challenges [12]. These models assume that there is always a prior probability available for entire training data. These mechanisms work relatively well if the goal is just to store maximum number of classes.

Error-correcting codes have played vital part in developing machine learning tools [13]. A fault-tolerant indexing scheme has been proposed in [14] that takes advantage of the perfect codes and our current work is directly inspired from their project. These codes provide a robust classification mechanisms when the data can only be categorized in a fuzzy manner.

III. Perfect Error-correcting Code (PEC) Clustering

In conventional forward error correction techniques, redundant parity bits are joined with a data word to create a codeword. In case of any discrepancies during transmission of the codeword, parity bits are used to restore the initial data word. In PEC clustering scheme, we use a perfect (23, 12, 7) error correction scheme. It has the minimum hamming distance of 7. So up to 3 bits of errors can be corrected if there are discrepancies. Codewords become spheres (clusters) with unique 23-bit binary vector as hash index. The explanation of mapping data word space to codewords can be found in [15].

A. Perfect Error-correcting Code

Hamming bound for error correcting codes is defined as, for any code E,

$$E = (M_n, D_k, H_d) \qquad (1)$$

Here M_n is the length of codeword, D_k is the dimension (length of the data word), and H_d is the minimum Hamming distance between two codewords and $H_d \leq 2e + 1$. It satisfies,

$$|E| \sum_{i=0}^{e} \binom{M_n}{i} \leq 2^{M_n} \qquad (2)$$

PEC with (23, 12, 7) code satisfies the Hamming bound equality as follows (M_n = 23, D_k = 12, and H_d = 7),

$$2^{12} \cdot \sum_{i=0}^{3} \binom{23}{i} = 2^{12} \cdot 2^{11} = 2^{23}$$

Here,

$$2^{11} = \binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}$$

B. PEC Clustering

(23, 12, 7) has one-to-one relationship between codeword and data word. There are 2^{23} codewords and 2^{12} data words. If the codeword space assumed to be a binary cube with 23 dimensions, it can be split equally into 2^{12} spheres. Thus each codeword sphere will consists of 2^{11} = 2048 binary vectors of the size of 23 bits. Since PEC can correct mistakes up to three bits, 23-bit vectors inside the spheres are within H_d = 3 Hamming distance from the centroid 23-bit vector. Since PEC focuses on H_d = 3, the clustering scheme will be interested in $\binom{23}{3}$ = 1771 23-bit binary vectors that are 3 Hamming distance away from center. Other vectors with even just 1 more bit variation will be sent to a different sphere (cluster). $\frac{1771}{2048} \approx 86\%$ of the vectors will stay close to the center where as $\frac{277}{2058} \approx 14$ will be assigned to new cluster's hash index.

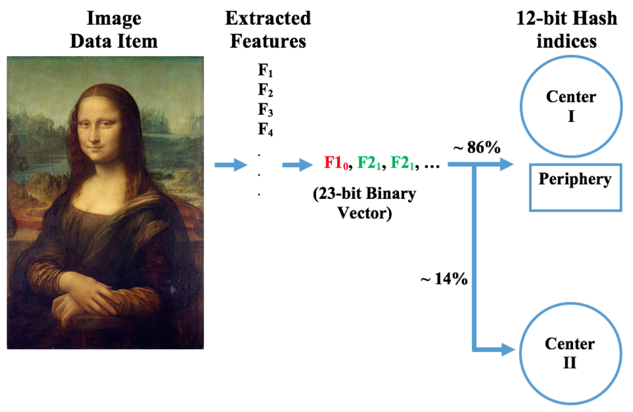

Figure 2. PEC Clustering Technique

PEC clustering technique (Figure 2) applies 23-bit binary vector to extracted features (1 for presence of feature and 0 for the absence) or predefined features and clusters them in appropriate hash indexes (cluster labels). Predefined feature extraction works as follows: assume a system that is classifying images of the city Washington D.C. The user can set the features and program the software to look for those features and apply the 23-bit vector. For example, F_1: Does the image has sharp triangle shaped white pillar?, $F2$ Is there a street sign(s) with an alphabet and number? etc. Based on the detected features, the 23-bit binary vector (e.g. 0, 1, 0, 0, 1, ...) will be created and will be assigned to a specific cluster in the hash indices. The vector does not need to be 23 bits long since the scheme supports even fewer parameters.

Another advantage of PEC clustering is that the 23-bit binary vectors can be used for autonomously generating 23-bit labels to be applied for a specific type of data item. If unknown data (that was not present in either training or testing dataset) appears then a new template (label) can be generated autonomously, creating a new cluster. The newly generated template can be applied to similar data in the future.

C. Parallel Processing of Data Streams

Information processing required for PEC clustering scheme to classify data streams requires multiprocessor pipeline. The PEC clustering technique must be capable of on-the-fly processing of data streams: distributed data processing can accommodate simultaneous processing of sequential/parallel data streams: the key idea behind the parallel processing is to host distributed data processing units (DDPU) that can (a) read (R) to load the data, (b) Analyze (A) to process and classify the data, and (c) toggle (T) to shift to/from read or analyze.

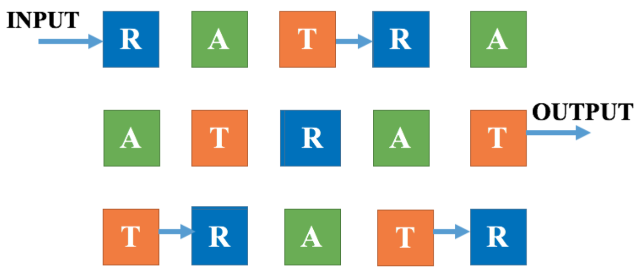

Figure 3. RAT processing of intensive data streams

The RAT processing model (Figure 3) allows any random algorithm to be performed on a data item on-the-fly with sufficient number of data processors. The processing model provides an efficient structure to transfer the states from one state to another in different processors (e.g. CPU processing cores). The system uses overlapping of data items through segmentation constraints (memory or hardware) imposed by processing module. The data streams can be autonomously partitioned and scheduled to move through the pipeline and to be added to the corresponding clusters. One of the unique features of RAT pipeline structure is that it does not require any conventional parallel processing techniques but can be accomplished by sequential processors.

IV. Evaluation

One of the main disadvantages of popular clustering algorithms is their time complexity. Conventional clustering algorithms with rich functionalities operate in exponential time. Hence they are prohibitive in preprocessing intense real-time data streams. PEC clustering operates on hash indices for cluster assignment. Once the 23-bit binary vector is applied to a data item, the algorithm needs to search the related cluster in the hash and store it. In the best case scenario, hash search can be completed in constant time ($\mathcal{O}(1)$) and in the worst case scenario the hash search can take $\mathcal{O}$(n) time. Thus the PEC clustering takes $\mathcal{O}$(n) time. There are other clustering mechanisms such as Fuzzy c-means clustering [19] and BIRCH clustering [20] that can complete the task in $\mathcal{O}$(n) time but they are not equipped to deal with large continuous stream of data. Comparison with existing clustering methods is given in Table I.

Table I
Comparison of Computational Complexities of Clustering Algorithms

Clustering Algorithms	**Time Complexities**
k-means	$\mathcal{O}(nkd)$
Hierarchical Clustering	$\mathcal{O}(n^2)$
Clustering using REpresentatives (CURE) [16]	$\mathcal{O}(n^2\ log\ n)$
ROCK [17]	$\mathcal{O}(min(n^2, nm_m m_a))$
CLICK [18]	$\mathcal{O}(n\ log\ n)$
PEC	$\mathcal{O}(n)$

Table II
Average cluster size with random 23-bit binary vectors with diverse probabilities for "1" occurring

Position of 1	**Average cluster size**
1	11
5	10
10	8
15	8
20	18
23	14

Table II shows the average cluster sizes for the position of 1 in the 23-bit binary vector. The probability is determined by $Pr(1_i) = \frac{i}{23}$ where i = 1, ..., 22. The result shows the cluster sizes are some what relatively distributed.

V. Conclusion

In this paper, we proposed PEC clustering based on (23, 12, 7) perfect error-correcting codes. Reversal of this error-correction scheme results in a robust clustering technique for stream data processing. The scheme offers 23-bit binary vector label for each data time and producing up to 2^{23} combinations of clusters. Hence the scheme can be used for classifying data with thousands of categories. It can correct up to 3 bits of errors in that 23-bit binary vector thus the scheme offers some fault tolerant. One of the most important qualities of PEC clustering is that it operates in $\mathcal{O}(n)$ time complexity. Compared to traditional and rich functionality clustering algorithms, PEC is fast and fault-tolerant. We also proposed a multi-pipeline processing structure that can efficiently process intensive data streams. We intend to apply this technique to computer vision problems such as large-scale image classifications in dynamic environments, and integrate the PEC clustering technique with deep learning models.

Acknowledgment

This research work is supported by NGC Research Consortium.

References

[1] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "Sun database: Exploring a large collection of scene categories," *International Journal of Computer Vision*, vol. 119, no. 1, pp. 3–22, 2016.

[2] G. W. Imbens and M. Kolesar, "Robust standard errors in small samples: Some practical advice," *Review of Economics and Statistics*, vol. 98, no. 4, pp. 701–712, 2016.

[3] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[4] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems*, 2010, pp. 163–171.

[5] A. Beygelzimer, J. Langford, and P. Ravikumar, "Multiclass classification with filter trees," *Preprint, June*, vol. 2, 2007.

[6] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl, "Conditional probability tree estimation analysis and algorithms," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 51–58.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[8] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: fast feature extraction and svm training," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1689–1696.

[9] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 1215–1222.

[10] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[11] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei, "What does classifying more than 10,000 image categories tell us?" in *European conference on computer vision*. Springer, 2010, pp. 71–84.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[14] G. Mani, N. Bari, D. Liao, and S. Berkovich, "Organization of knowledge extraction from big data systems," in *Computing for Geospatial Research and Application (COM. Geo), 2014 Fifth International Conference on*. IEEE, 2014, pp. 63–69.

[15] S. Y. Berkovich and E. El-Qawasmeh, "Reversing the error-correction scheme for a fault-tolerant indexing," *The Computer Journal*, vol. 43, no. 1, pp. 54–64, 2000.

[16] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM Sigmod Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[17] ——, "Rock: A robust clustering algorithm for categorical attributes," *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[18] M. Peters and M. J. Zaki, "Click: Clustering categorical data using k-partite maximal cliques," *Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY*, vol. 12180, 2004.

[19] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[20] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.