Vectorized Implementation of K-means

Tomoki Otsuka and Norishige Fukushima*
Nagoya Institute of Technology, Japan
Email: *fukushima@nitech.ac.jp

ABSTRACT

K-means is a widely used clustering technique that seeks to minimize the squared distance among points in the same cluster. K-means is an appealing clustering method in terms of computational speed. Also, K-means is the simplest clustering; thus, researchers select K-means as a first choice. Therefore, accelerating K-means is essential. In this study, we transform data structure from the array of structures to the structure of arrays for accelerating K-means by SIMD vectorization. Experimental results show that our implementation is faster than OpenCV's implementation.

Keywords: K-means, K-means++, SIMD, AVX

INTRODUCTION

The unsupervised classification of patterns into groups, called clustering, is one of the most important tasks in exploring and analyzing data [1]. K-means [2] is a well-known clustering technique that is used in various image processing methods, e.g., high-dimensional searching [3] and biomedical processing [4]. Also, K-means is used in a part of the acceleration method for image filtering [5, 6], such as bilateral filtering [7] and non-local means [8]. The goal of K-means is to determine K clustering centers that minimize the squared distance among points in the same cluster. Usually, the K-means algorithm begins with K arbitrary centers chosen randomly from input data points. Each data point is then assigned to the nearest center. Then, each center is recomputed as the center of the cluster of all points assigned to it. When the assigned clusters for each data point stabilizes, the process terminates.

K-means is an appealing clustering method in terms of processing speed. Also, K-means is the simplest clustering; thus, researchers select K-means as a first choice. Moreover, K-means is used for an initial estimation of more sophisticated clustering [9, 10]. Therefore K-means is essential for clustering.

The main drawback of K-means is initial value dependence. K-means++ [11] is a solution for this problem. In K-means++, the initial clustering centers are determined by the probability distribution of the squared distance; thus, K-means++ can determine more unbiased initial clustering centers than K-means. K-means++ is widely used, e.g., IBM SPSS Statistics [12], scikit-learn [13], and OpenCV [14]. However, the implementation of it is not matured. K-means++ requires more processes to determine initial clustering centers than K-means; thus, it requires more processing cost than K-means. Therefore, the acceleration of K-means++ is essential.

In our study, we accelerate K-means and K-means++ by SIMD vectorization. The standard format of image data is unsuitable for vectorization; thus, we first transform the image's data structure from the array of structures (AoS) to the structure of arrays (SoA). The transformation is realized by matrix transform. Then, we vectorize K-means and K-means++ for acceleration.

K-MEANS AND K-MEANS++

K-means

We introduce K-mean clustering in this section. Let $\boldsymbol{N} \in \mathbb{R}^{\boldsymbol{d}}$ be the number of data points and be $\boldsymbol{K} \in \mathbb{R}^{\boldsymbol{d}}$ be the number of clustering points. K-means aims to minimize the following function:

$$\arg\min_{\mathcal{C}} \sum_{x_i \in \mathcal{X}} \min_{c_j \in \mathcal{C}} ||x_i - c_j||^2 ,$$

where $x_i \in \mathcal{X}$ $(i = 1,2, \dots, N) \subset \mathbb{R}^d$ is a d dimensional data point. $c_i \in \mathcal{C}$ $(i = 1,2, \dots, K) \subset \mathbb{R}^d$ is a clustering center, called centroid. The problem is NP-hard, and K-means is a heuristic solution. Let $D(x^2)$ be the squared distance between $x_i \in \mathcal{X}$ and a target centroid. K-means of Forgy's method [15], which is a traditional initialization for K-means, is as follows:

1. Randomly select K data points as initial centroids.
2. For all pixels $\forall x_i \in \mathcal{X}$, compute $D(x)^2$ from each centroid, and then reassign to the cluster that minimizes $D(x)^2$.
3. If c_i with no data points in its cluster exists, change the clustering by following process.

    3-a. Find the biggest cluster, i.e., the cluster with most data points.

    3-b. Find the farthest data points $\hat{x}_i$ in the biggest cluster.

    3-c. Exclude $\hat{x}_i$ from the biggest cluster and include $\hat{x}_i$ in the cluster with c_i as the centroid.
4. Repeat step 2 and 3 until the assigned cluster for each x no longer changes.

Step 2 and 3 guarantees to decrease $\sum_{x_i \in \mathcal{X}} \min_{c_j \in \mathcal{C}} ||x_i - c_j||^2$; thus, this algorithm makes local improvements to an arbitrary clustering until it is no longer possible to do so. Additionally, Lloyd's algorithm [16] is also a well-known K-means implementation. This algorithm starts with K arbitrary centers, typically chosen uniformly at random from the data points instead of using step 1 and 2.

K-means minimizes the local solution; therefore, the method has a high dependence on initial centroids. K-means++ solves the issue of initiation.

K-means++

We also introduce K-means++. In K-means++, the initial centers are determined by the following procedure instead of using random assignment.

1. Randomly select a data point x_i as an initial centroid c_0.
2. For all pixels $\forall x_i \in \mathcal{X}$, compute $D(x)^2$ from the nearest centering point.
3. Determine a new cluster center by following process.

    3-a. Randomly select a data point x_i as a candidate centroid c_i.

    3-b. For all pixels $\forall x_i \in \mathcal{X}$, compute $D(x)^2$ from the nearest centering point.

    3-c. If $\sum_{x \in \mathcal{X}} D(x)^2$ is smaller than before, determine c_i as a new cluster center.

    3-d. repeat step 3-a, -b and -c several times.
4. Repeat step 3 until finding the K-th centroid.

Then, we start step. 2 in K-means. The concept of K-means++ is to position the initial clustering points away from each other.

VECTORIZATION OF K-MEANS

Computing $D(x)^2$ has the highest processing cost that is in K-means step-3 and K-means++ step-2 and 3. The processing order is $O(NKd)$, where N is the number of input data, K is the number of clusters, and d the number of dimensions of input data. Accelerating the part is essential. We accelerate the processing by SIMD vectorization, e.g., SSE, AVX, AVX512, and NEON. In SIMD operation, we load multiple data sequentially with vector length, i.e., SSE (4), AVX (8), AVX512 (16), and NEON (4), and then we load the other data for operation is loaded. The two data units of a vector are computed once. Therefore, the computation time becomes faster by the SIMD vector length, ideally.

Usually, we prepare input data in the AoS format. For example, the image format is interleaved, i.e., RGBRGB…RGB. However, AoS is unsuitable for vectorization, because computing d dimensional norm in D requires a horizontal operation in vectorization for inner-product operations. Also, n should be multiple of SIMD vector length, i.e., 4, 8, or 16. If not, we need padding to ceil the nearest vector length that is redundant processing or we need to abandon vectorization for the overhanging part. The horizontal operation and padding data are inefficient solutions for vectorization, and these cannot extract the maximum performance of computers. Note that OpenCV's implementation uses AoS format, and vectorization is realized by "hal" module in its library.

By contrast, the SoA structure has no restriction for horizontal operation. Also, SoA structure moderates the residual processing issue. In SoA, we need the residual processing of padding or no-vectorization for the d dimension's loop. In padding for AVX case, data size becomes $Nd', d' = ceil(d, 8)$. AoS also requires residual processing but need for the N-loop. The size is $N'd,\ N' = ceil(N, 8)$. Usually, $N \gg d$ and N is a large number, so $N'/N \cong 1$. d is not a large number $d'/d > 1$, e.g., 3-dimensional case, the ratio is 8/3=2.667.

The transformation is realized by matrix transpose. A *N*-rows-*d*-cols matrix becomes *d*-rows-*N*-cols matrix. For example, when input data are a color image, the data are rearranged from RGB format to RRR…GGG…BBB format by $N \times 3$ matrix transpose.

EXPERIMENTAL RESULTS

We demonstrate the effectiveness of our implementation. We used the output image of $cv::kmeans$ in OpenCV as the answer image of K-means and K-means++. The code was written in C++ and vectorized by AVX/AVX2. The test CPU was Intel Core i9-9900K 3.60 GHz, 8-cores/16-threads. The test image size was 512×512.

Figure 1 shows the processing time of our implementation, scikit-learn's and OpenCV's one. The proposed implementation can suppress the processing time than scikit-learn and OpenCV.

Figure 2 shows the average of squared distance $D(x^2)$ in our implementation and OpenCV's one. Moreover, $D(x^2)$ of the proposed method is nearly equal to K-means and K-means++ of OpenCV, respectively.

In these figures, our proposed method can cluster more properly with suppressed computational time.

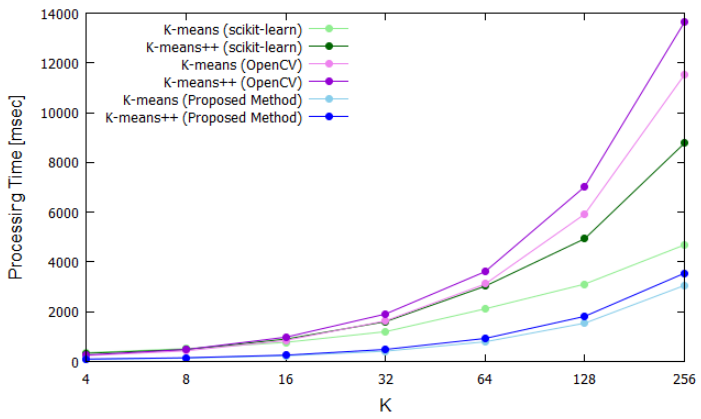

Figure 1. Processing time of K-means++ implemented by ours and OpenCV ($N = 512 \times 512, d = 3$). x-scale is $\log_2$.

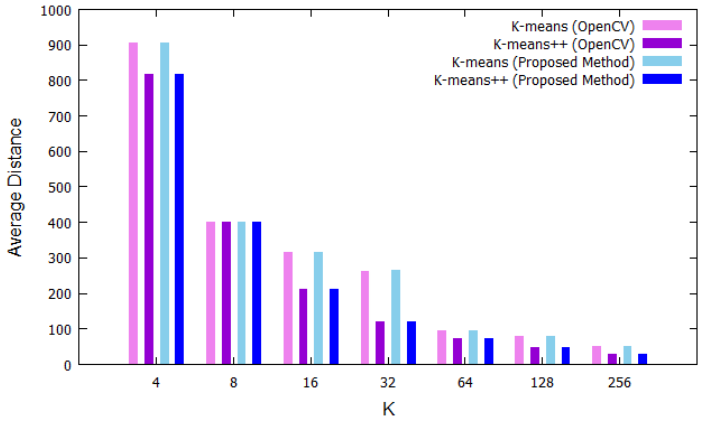

Figure 2. The average of $D(x^2)$ implemented by ours and OpenCV ($N = 512 \times 512,\ d = 3$). x-scale is $\log_2$.

CONCLUSION

In this study, we transform the input data structure from AoS to SoA. Moreover, we effectively implement K-means++ by vectorization. The experimental results show that our implementation of K-means is about 3.03 and 1.53 times faster than OpenCV and scikit-learn in $K = 256$, respectively. Moreover, in K-means++, our implementation is about 3.62 and 2.47 times faster than OpenCV and scikit-learn in $K = 256$, respectively. In terms of clustering accuracy, our implementation is nearly equal to OpenCV's one. Therefore, our method can compute clustering correctly with suppressed processing time.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI (JP17H01764, JP18K19813).

REFERENCES

[1] Jain, A. K., Murty, M. N., and Flynn, P. J., "Data Clustering: A Review," ACM Computing Surveys, 31, 3, 264–323 (1999).

[2] MacQueen, J., "Some methods for classification and analysis of multivariate observations," Proc. Berkeley symposium on mathematical statistics and probability, 1, 14 (1967).

[3] Muja, M., and David, G. L., "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 11, 2227-2240 (2014).

[4] Xu, Rui, and Wunsch, D. C., "Clustering algorithms in biomedical research: a review." *IEEE Reviews in Biomedical Engineering* 3 (2010): 120-154.

[5] Nair, P., and Chaudhury, K. N., "Fast high-dimensional kernel filtering," IEEE Signal Processing Letters, 26, 377– 381 (2019)

[6] Miyamura, T., Fukushima, N., Waqas, M., Sugimoto, K., and Kamata, S., "Image Tiling for Clustering to Improve Stability of Constant-time Color Bilateral Filtering," in Proc. International Conference on Image Processing (ICIP) (2020).

[7] Tomasi, C., and Manduchi, R., "Bilateral filtering for gray and color images," in Proc. IEEE International Conference on Computer Vision (ICCV) (1998).

[8] Buades, A., Coll, B., and Morel, J. -M., “A non-local algorithm for image denoising,” in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 60-65 (2005).
[9] Celebi, M. E., Kingravi, H. A., and Vela, P. A., “A comparative study of efficient initialization methods for the k-means clustering algorithm,” *Expert Systems with Applications*, 40, 1, 200-210 (2013).
[10] Bradley, P. S., Fayyad, U., “Refining Initial Points for K-Means Clustering,” in: Proc. International Conference onMachine Learning, 91–99 (1998).
[11] Arthur, D., and Vassilvitskii, S., "k-means++: The Advantages of Careful Seeding.," Technical Report, Stanford (2006).
[12] Noruˇsis, M. J., “IBM SPSS Statistics 19 Statistical Procedures Companion,” Addison Wesley (2011).
[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É.,; 12, 85, 2825−2830 (2011).
[14] Bradski, G., and Kaehler, A., “Learning OpenCV: Computer vision with the OpenCV library,” O'Reilly Media, Inc., (2008).
[15] Forgy, E., “Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification,” Biometrics, 21, 768 (1965).
[16] Lloyd, S., “Least Squares Quantization in PCM,” IEEE Transactions on Information Theory, 28, 2, 129–136 (1982).