Journal of Systems Architecture 59 (2013) 155–164

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

K-means clustering algorithm for multimedia applications with flexible HW/SW co-design

Fengwei An *, Hans Jürgen Mattausch

Research Institute for Nanodevice and Bio Systems, Hiroshima University, Japan

ARTICLE INFO

Article history:
Received 13 June 2012
Received in revised form 25 September 2012
Accepted 20 November 2012
Available online 12 December 2012

Keywords:
Hardware/software co-design
K-means clustering algorithm
Nearest neighbor searching
Handwritten digit recognition
Face recognition
Image segmentation

ABSTRACT

In this paper, we report a hardware/software (HW/SW) co-designed K-means clustering algorithm with high flexibility and high performance for machine learning, pattern recognition and multimedia applications. The contributions of this work can be attributed to two aspects. The first is the hardware architecture for nearest neighbor searching, which is used to overcome the main computational cost of a K-means clustering algorithm. The second aspect is the high flexibility for different applications which comes from not only the software but also the hardware. High flexibility with respect to the number of training data samples, the dimensionality of each sample vector, the number of clusters, and the target application, is one of the major shortcomings of dedicated hardware implementations for the K-means algorithm. In particular, the HW/SW K-means algorithm is extendable to embedded systems and mobile devices. We benchmark our multi-purpose K-means system against the application of handwritten digit recognition, face recognition and image segmentation to demonstrate its excellent performance, high flexibility, fast clustering speed, short recognition time, good recognition rate and versatile functionality.

1. Introduction

Clustering algorithms have many applications, such as data mining and knowledge discovery, vector quantization, pattern recognition, or image segmentation. Clustering is the computational task to partition a set of data samples into a certain number of clusters. These clusters consist of similar data samples that are dissimilar to samples in other clusters.

Among clustering algorithms, the most popular and most widely studied is the K-means clustering algorithm [1–3]. It is known that this algorithm converges to a local optimum by a measure which is based on minimizing the mean squared distance from each data sample to its nearest centriod. This measure yields a large number of nearest neighbor queries. Many advanced nearest neighbor searching techniques such as the KD-tree and hash table structure [4,5] were proposed to reduce the time needed to compute the nearest neighbor because it is the most time consuming procedure in the K-means algorithm. Unlike the sequential processing of conventional implementations, the graphics processing unit (GPU) provides a more powerful processing platform which is specialized for parallel programming. Recently, the K-means algorithm was parallelized by GPU implementation [6–8].

Besides the software implementations, K-means hardware architectures were developed in [9,10] for image segmentation. Such hardware implementations have achieved efficient performance provided that the dimensionality or the number of training data samples could be kept low enough, since not only large adders but also large dividers are required in the centroids updating step. Even though the K-means algorithm is simple and therefore seems suitable for the requirements of hardware-friendly algorithms, the centroids updating is still the bottleneck in digital technologies. As a result, the hardware implementation of the K-means clustering algorithm is seldom flexible enough for different applications. Particularly, the flexibility in regards to the dimensionality of each training sample, the number of samples, and the number of clusters restrict the functionalities of hardware-implemented K-means. Rather than a dedicated circuit for nearest distance searching and centroids updating, a hybrid processor included NOIS and ARM solution was proposed in [11]. In [12], the HW/SW communications with PCI bus in the area of the K-means implementation for hyperspectral images was proposed by Dominique Lavenier. However, the usage of the software side is more restricted and mainly prepares the data samples for the K-means clustering.

The K Nearest Neighbors (KNN) classifier is one of the simplest learning algorithms for classifying unknown samples by means of nearest-neighbor searching. Although KNN-based systems have been studied for several decades [13], the main limitation of KNN in practice is again the high computational demand of the minimum distance searching, which emerges as the complicated task with the K-means. In spite of the increasing speed of general-purpose CPUs, the dedicated hardware and GPU implementations

* Corresponding author.
E-mail address: anfengwei@hiroshima-u.ac.jp (F. An).

1383-7621/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2012.11.004

of KNN [14,15] have an obvious superiority in performance. However, finding the k minimum distances to the test sample among the entire training samples, which causes large storage requirements and high computational costs even by hardware, seldom receives the deserved practical consideration.

In order to minimize the computational efforts in the classification stage of the nearest-distance rule based learning algorithm, the cluster-based prototype learning algorithm, e.g. K-means clustering, has been proposed to reduce the number of references of nearest-neighbor classifiers, e.g. KNN. Instead of recruiting all training samples, the prototypes (references) are determined as the centroids of clusters derived from the average of training samples. These prototypes are vectors that reside in the same vector space as the feature vectors of the training samples. The number of prototypes is sufficiently compressed to a much lower order than that of the training samples by the clustering algorithm. The prototype learning algorithm has obviously lower training speed than KNN. However, it is worthwhile to point out that the training is often executed only once in the lifetime of the learning system, as it is also the case for artificial neural networks.

The hardware implementation for realizing machine learning algorithms for pattern recognition offers significant efficiency. However, hardware implementations became unattractive due to the failure of commercializing VLSI implementations since the market has not accepted any chip implementation of learning algorithms owing to their very limited flexibility. Such flexibility is particularly important whenever a learning system has to be applied to a set of different applications.

Artificial neural networks and fuzzy systems are the most popular learning algorithms in hardware implementations. The survey paper [16] about the hardware neural networks pointed out their very limited use in real-world applications due to the fact that either the number of input and output layers, the size of the hidden layer, or the weight values vary from case to case. Unlike neural-network approaches, Support Vector Machines (SVMs) use structural risk minimization so that SVMs often outperform neural networks in practice. For this reason, the hardware implementation of SVMs also seemed attractive in the past decade [17–19]. A hardware-friendly learning algorithm is the common requirement of all hardware implementations. In order to reduce the cost, power consumption, etc., hardware-friendly neural networks and SVMs must be simple enough to adapt to the constraints of the hardware. For instance, the mathematical operations, which are easy to be implemented by the software, are often difficult by the hardware. The limited flexibility, simplicity and insufficient technical supports lead to few known commercial neural networks, fuzzy systems or SVMs chips for practical applications.

The contributions of this work can be attributed to two aspects. The first is the hardware architecture for nearest neighbor searching. A large number of nearest neighbor queries arises when the K-means algorithm converges to the local optimum and when a prediction is made by the nearest-neighbor classifier. Consequently, the main computational cost is overcome by means of hardware acceleration with parallelizing and pipelining. The second aspect is the high flexibility to cope with different applications such as machine learning, pattern recognition, and image segmentation. The flexibility comes not only from the software but also from the hardware. It is well known that the software implementation is by far the most flexible while the hardware implementation seems less flexibility in most cases. However, the relatively high flexibility of the hardware implementation for nearest distance searching derives from the fact that the dimensionality of samples and the number of clusters are scalable so as to further improve the searching speed when the application has a low dimensional vector. The K-means algorithm with hardware/software (HW/SW) architecture, which is versatile in machine learning, pattern recognition, or multimedia processing, is extendable to embedded systems and mobile devices.

The contents of this paper are organized as follows: Section 2 describes the K-means algorithm with HW/SW co-design architecture where the hardware is used for the nearest Euclidean distance searching and the software is used to update the centroids and to control the hardware. In Section 3, the complexity of the hardware is described. The applications of handwritten digit recognition, face recognition, and image segmentation are used to validate the K-means algorithm with HW/SW co-design architecture in Section 4. The comparison to GPU-based and full-hardware K-means implementations is carried out in Section 5. The possibility of improving this work is discussed in Section 6. Finally, we conclude in Section 7.

2. HW/SW co-designed K-means clustering

The K-means algorithm is widely used in machine learning and data compression. The goal of this algorithm is to partition a set of data samples into k clusters by considering the boundary information of confusing classes. Suppose that a set of n training samples ($X \in \mathbb{R}^d$) and the clusters number k ($0 < k < n$) are given. The k optimal cluster centriods $(\vec{c_1}, \vec{c_2}, \cdots, \vec{c_k})$ with each centroid $\vec{c_k} \in \mathbb{R}^d$ can be obtained by minimizing the objective function (1)

$$\Phi_{K-means} = \min \sum_{i=1}^{k} \sum_{\vec{x_j} \in \vec{c_i}}^{j<n} ||\vec{x_j} - \vec{c_i}|| \tag{1}$$

$||\vec{x_j} - \vec{c_i}|| = \sqrt{(\vec{x_j} - \vec{c_i})^T (\vec{x_j} - \vec{c_i})}$ is defined as the Euclidean distance metric in $\Re^d$.

The k cluster centroids are initialized by randomly choosing k samples from the training data set. Then each sample $\vec{x_j}$ in the data set is labeled and assigned to the cluster $\vec{c_i}$ which is the nearest neighbor of the input sample according to the Euclidean distance. After processing the entire data set, each centroid is updated by calculating the mean of all data samples which are labeled to this centroid $\vec{x_j} \in \vec{c_i}$ according to (2). The iteration of finding the nearest neighbors and updating the centroids will not be terminated as long as the k cluster centroids change. This is the procedure of convergence to a local minimum in (1)

$$\vec{c_i} = \frac{1}{|\vec{c_i}|} \sum_{\vec{x_j} \in \vec{c_i}} \vec{x_j} \tag{2}$$

The K-means algorithm can be described as follows:

Step 1 Randomly choose k cluster centroids.
Step 2 For each training sample, find the nearest cluster centroid.
Step 3 Assign the sample to its nearest cluster centroid.
Step 4 Update the cluster centroids.
Step 5 Repeat steps 2–4 until the convergence.

In this paper, the critical path of the K-means algorithm with the most computational cost, which arises in step 2, is significantly improved by the hardware implementation. The step of updating the cluster centroid is relatively simple in software but becomes increasingly complex in hardware due to the requirement of large dividers. Therefore, the K-means algorithm can exploit the strengths of both software and hardware, which means that it is very suitable for HW/SW co-design.

2.1. FPGA implementation of nearest-distance searching

The conventional methods for nearest-distance searching are often based on a general purpose computer with sequential pro-