Project Proposal
Parallel Algorithm for Sparse Coding and Dictionary Learning on GPUs

Manh Huynh
Parallel Distributed System - Fall 2016
Department of Computer Science and Engineering
University of Colorado Denver

November 15, 2016

1 Introduction

While the goal of sparse coding is to find the sparse representation of a set of input signals by given a learned dictionary, the goal of K-SVD for dictionary learning is to create a set of compact and meaningful atoms that is better fit the data. These two algorithms often closely work together to create the best sparse coding and dictionary. Then, a given input signal can be described as the linear combination of the learned atoms. Since a high dimensional signal now can be represented as a very sparse signal, the dictionary learning method becomes very useful in a wide range of applications including image compression, image denoising, and recently be adapted to face recognition and human tracking.

The aim of dictionary learning is to learn an dictionary $D \in R^{nxK}$, where K is the number of atoms and n is the dimension of each signal. Given the input signal x, we want to find the sparse vector α such that $x \approx D\alpha$, or $||x - D\alpha||_p \leq \epsilon$ for some small value ϵ and L_p norm. The sparse vector α contains the representation coefficient of atoms in D, or we can think of each of element in sparse representation α is the contribution of each of atoms to construct x. Normally, the norm L_p is L_1,L_2, or L_∞. In this project, the L_2 is used for its simplicity. The table 1 shows the notation we use in this project.

In this project, we aim to exploit fully parallel algorithm for sparse coding and dictionary learning for the purpose of multi-target tracking application. Specifically, dictionary is learned to classify the target's appearance, which is an important component and also highly computational in tracking. Since real-time tracking requires the processing time within the frame speed, we expect to speed-up the current serial algorithm to match this requirement. We note that the tracking algorithms may consist of many other components such as human detection, moition prediction, Hungarian algorithm, which also consume lots of computation. However, in the scope of this project, we mainly focus on dictionary learning and leave the other parts as future works. In the next section, we will discuss more about the algorithm for sparse coding and dictionary learning. We will also point out the some challenges for each algorithm that we may encounter when designing parallel algorithms.

Table 1: Table of Notation

n	$\triangleq$	Dimensions of input signal
K	$\triangleq$	Size of Dictionary, the number of atoms
N	$\triangleq$	The number of input signals
x	$\triangleq$	Single input signal
α	$\triangleq$	Sparse representation of x
$D[n \times K]$	$\triangleq$	Dictionary of size $n \times K$
$X[n \times N]$	$\triangleq$	Set of input signals of size $n \times N$
$A[K \times N]$	$\triangleq$	Sparse representation of matrix X

2 Sparse Coding Algorithm

The objective function of sparse representation is:

$$\min_{\alpha} ||\alpha||_0 \quad \text{subject to} \quad ||D\alpha - x||_p < \epsilon \tag{1}$$

The $||\alpha||_0$ is the L_0 norm, which denotes the number of zero element in sparse vector. In overall, the goal of sparse coding algorithm is to find the sparsest vector α that satisfy the condition $D\alpha$ is close to x. Solving the equation 1 is an NP-hard problem. There is several research try to relaxed the constraint by using L_1 norm instead of L_0. Others using greedy matching algorithm (algorithm 1), and that is the algorithm of our focus in this proposal. The idea of gready matching algorithm is searching through all atoms in dictionary to find the best match in the input x, calculating the error, and continue to find another atom in dictionary to fit the error. The process is recursive until there is smaller the error threshold (ϵ). From my perspective, we can totally do calculation for each of input x individually, but it is still difficult to exploit fully parallel within the process of one vector x. This is because there is data dependency, that is we have to wait until the first best match atom is found, to find the second best to match the error. The second thing which is challenging is different input vector x may be processed in different time. If we allocate some of PEs unit dedicated to that processing of x, all these PEs may finish sooner that other. That is what we want to improve to increase the load balance and then efficiency.

3 Dictionary Learning Algorithm

The dictionary learning stage refines the dictionary, D, after the sparse representation matrix has been calculated. Note that sparse representation matrix A consist of the sparse vector for each input vector x in matrix X. The algorithm of K-SVD for dictionary learning is shown in algorithm 2. The idea of K-SVD algorithm for dictionary learning is that, we will update each of atoms. Because each atom represent a basis of data representation in high dimensional space, it means that we want to find set of basis that better fit the data. The procedure to do that is we want to remove each of atom one at time, finding the errors of data without that atom. Then using SVD to find the largest eigen vector for that data error. Finally, we replace the chosen atom with that

Algorithm 1 Greedy Matching Pursuit Algorithm

```
**Input:** Dictionary D, input signal x
**Output:** Sparse presentation vector $\alpha$
**Initialization:**
$R \leftarrow x$
$n \leftarrow 1$
**while** $R_{n+1} > threshold$ **do**
    Find atom $g_j$ with maximum inner product $| < R_n, g_j > |$
    $\alpha_i \leftarrow < R_n, g_j > / ||g_j||^2$
    $R_{n+1} \leftarrow R_n - \alpha_i g_j$
    $n \leftarrow n + 1$
**end while**
**end**
```

Algorithm 2 Orthogonal Matching Pursuit Algorithm

```
**Input:** Orthogonal dictionary D, input signal x
**Output:** Sparse presentation vector $\alpha$
**Initialization:**
$R \leftarrow x$
$n \leftarrow 1$
**while** $\epsilon \leq threshold$ **do**
    Find atom $g_j$ with maximum inner product $| < R_n, g_j > |$
    $\alpha_i \leftarrow < R_n, g_j > / ||g_j||^2$
    Find orthogonal projection operator P of previous selected atoms (Gram-Schmidt algorithm)
    Apply the orthogonal projection operator to the residual R
    $R_{n+1} \leftarrow R_n - \alpha_i g_j$
    $n \leftarrow n + 1$
**end while**
**end**
```

eigen vector, because eigen vector help better represent the data.
We note that K-SVD algorithm include the greedy algorithm to calculate the sparse representation for input signals in X and the K-SVD to find the better atom. The process is repeated in the number of defined iterations. The complexity of this serial algorithm is $O(n^4)$, which is unacceptable for real-time multitarget tracking. However, this algorithm turns out to be difficult to parallelize since each of atom has to be updated sequentially. The paper[1] consider batching method for similar and different patterns in sparse representation matrix to speed-up. In detail, set of atoms that distribute on the same set of representation vectors can be grouped together for processing. The result shows that it produces a considerable speed-up. However, this paper uses image denoising for the test, in which we believe we may encounter another problem when using this batching method for human tracking.

Algorithm 3 K-SVD Algorithm

```
Input: D,X
Output: Learned D, A
for t = 1 to iterations do
    $A \leftarrow greedymatchingpursuit(D, X)$
    //Dictionary Update Stage
    for i=1 to K do do
        L = indices of non-zero element in $A(i, :)$
        $\hat{X} = X_L$ ; $\hat{A} = A_L$
        $E = \hat{X} - \Sigma_{k \neq i} d_k * \alpha_k^T$
        $U * \Sigma * V^T \approx svd(E)$
        $d_i = U(:, 1)$
    end for
end for
end
```

4 Conclusion

We have briefly introduced the goal and scope of this project. Also, some part of detail about sequential algorithm on sparse coding and dictionary learning are presented. More importantly, fully exploiting to adapt these algorithm into multi-target tracking is a challenging task. We has pointed out some of difficulties of the project. However, we expect that the more challenges will be appeared on the way studying on this topic.

References

[1] Lu He, Timothy Miskell, Rui Liu, Hengyong Yu, Huijuan Xu, and Yan Luo. Scalable 2d k-svd parallel algorithm for dictionary learning on gpus. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 11–18. ACM, 2016.

[2] Mohamed Elbahri, Nasreddine Taleb, Kidiyo Kpalma, and Joseph Ronsin. Parallel algorithm implementation for multi-object tracking and surveillance. *IET Computer Vision*, 10(3):202–211, 2015.