OpenCL based machine learning labeling of biomedical datasets

Oscar Amorós, Anna Puig, and Sergio Escalera

Dept. Matemàtica Aplicada i Anàlisis, UB, Gran Via de les Corts Catalanes 585, Barcelona
Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Barcelona
E-mail: oamorohu7@alumnes.ub.edu,{anna,sergio}@maia.ub.es

Abstract

In this paper, we propose a two-stage labeling method of large biomedical datasets through a parallel approach in a single GPU. Diagnostic methods, structures volume measurements, and visualization systems are of major importance for surgery planning, intra-operative imaging and image-guided surgery. In all cases, to provide an automatic and interactive method to label or to tag different structures contained into input data becomes imperative. Several approaches to label or segment biomedical datasets has been proposed to discriminate different anatomical structures in an output tagged dataset. Among existing methods, supervised learning methods for segmentation have been devised to easily analyze biomedical datasets by a non-expert user. However, they still have some problems concerning practical application, such as slow learning and testing speeds. In addition, recent technological developments have led to widespread availability of multi-core CPUs and GPUs, as well as new software languages, such as NVIDIA's CUDA, and OpenCL, allowing to apply parallel programming paradigms in conventional personal computers.

Adaboost[1] classifier is one of the most widely applied methods for labeling in the Machine Learning community. In a first stage, Adaboost trains a binary classifier from a set of pre-labeled samples described by a set of features. This binary classifier is defined as a weighted combination of weak classifiers. Each weak classifier is a simple decision function estimated on a single feature value. Then, at the testing stage, each weak classifier is independently applied on the features of a set of unlabeled samples.

In this work, we propose an alternative representation of the Adaboost binary classifier. We use this proposed representation to define a new GPU-based parallelized Adaboost testing stage using the OpenCL architecture.

We provide numerical experiments based on large available data sets and we compare our results to CPU-based strategies in terms of time and labeling speeds.

1 Description of purpose

There are several biomedical applications that need to extract different objects of interest, such as tissues and organs, contained into a volume dataset from MRI, CT, fMRI and PET input captions. Diagnostic methods, structures volume measurements, and visualization systems require to specify to which anatomical structure each sample/voxel[3][4] belongs. In the bibliography, Transfer Functions has been used in order to directly associate optical properties or labels to the different data samples according their belonging to a particular structure in the underlying data. However, in general, the anatomical structures are complex, and relationships between them do not allow

to separate sufficiently the different structures. In these cases, Transfer Functions alone do not suffice in order to separate different objects and it becomes necessary to use labeling, or segmentation methods. In this sense, several approaches to label biomedical datasets has been proposed to discriminate different anatomical structures in an output tagged dataset depending on the used imaging modality. Among existing methods, supervised learning methods for segmentation have been devised to easily analyze biomedical datasets by a non-expert user. In a pre-process, an expert user, such a radiologist, should to identify a subset of samples of each anatomical structure. Then, during the learning step, the supervised method defines a classifier to be automatically used in the testing or classification stage. Since learning phase is carried on a pre-process, before labeling, the classifier can be used in different classifications several times by an inexpert users. Thus, to optimize the classification stage becomes imperative.

Thanks to the recent emerging technologies of multi-core CPUs and GPUs, as well as new software languages, such as NVIDIA's CUDA, and OpenCL[2], we propose to parallelize the classification step of a well-know supervised method, called Adaboost, in the GPU.

2 Method(s)

First of all, we revise of the state-of-the-art of the published methods on classification and their GPU-based implementations and we justify our proposed approach, that is three-fold:

- We propose an alternative representation of the Adaboost binary classifier.
- We use this proposed representation to define a new GPU-based parallelized Adaboost testing stage using the OpenCL architecture.
- We provide numerical experiments based on large available data sets and we compare our results to CPU-based strategies in terms of time and labeling speeds.

2.1 Adaboost binary classifier

Adaboost classifier is one of the most widely applied methods for labeling in the Machine Learning community. In this paper, we focus on the Discrete version of Adaboost, which has shown robust results in real applications. Given a set of N training samples $(x_1, y_1), .., (x_N, y_N)$, with x_i a vector valued feature and $y_i = -1$ or 1. We define $F(x) = \sum_1^M c_f f_m(x)$ where each $f_m(x)$ is a classifier producing values ± 1 and c_m are constants; the corresponding prediction is sign$(F(x))$. The Adaboost procedure trains the classifiers $f_m(x)$ on weighed versions of the training sample, giving higher weights to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear combination of the classifiers from each stage. E_w represents expectation over the training data with weights $w = (w_1, w_2, .., w_N)$, and $1_(S)$ is the indicator of the set S. For a good generalization of $F(x)$, each $f_m(x)$ is required to obtain a classification prediction just better than random. Thus, the most common "weak classifier" f_m is the "decision stump". For each $f_m(x)$ we just need to compute a threshold value and a polarity to take a binary decision, selecting that one that minimizes the error based on the assigned weights. This simple combination of classifiers has demonstrated to reduce the variance error term of the final classifier $F(x)$.

In Algorithm 1, we show the *testing* of the final decision function $F(x) = \sum_1^M c_f f_m(x)$ using the Discrete Adaboost algorithm with Decision Stump "weak classifier". Each Decision Stump f_m fits a threshold T_m and a polarity P_m over the selected m-th feature. In testing time, x^m corresponds to the value of the feature selected by $f_m(x)$ on a test sample x. Note that c_m value is subtracted from $F(x)$ if the hypothesis $f_m(x)$ is not satisfied on the test sample. Otherwise, positive values of c_m

are accumulated. Finally decision on x is obtained by sign$(F(x))$.

1: Given a test sample x
2: $F(x) = 0$
3: Repeat for $m = 1, 2, .., M$:
(a) $F(x) = F(x) + c_m(P_m \cdot x^m < P_m \cdot T_m)$;
4: Output sign$(F(x))$

Algorithm 1: Discrete Adaboost testing algorithm.

We propose to define a new and equivalent representation of c_m and $|x|$ that facilitate the parallelization of the testing. We define the matrix $V_{f_m(x)}$ of size $3 \times (|x| \cdot M)$, where $|x|$ corresponds to the dimensionality of the feature space. First row of $V_{f_m(x)}$ codifies the values c_m for the corresponding features that have been considered during training. In this sense, each position i of the first row of $V_{f_m(x)}$ contains the value c_m for the feature $mod(i, |x|)$ if $mod(i, |x|) \neq 0$ or $|x|$, otherwise. The next value of c_m for that feature is found in position $i + |x|$. The positions corresponding to features not considered during training are set to zero. The second and third rows of $V_{f_m(x)}$ for column i contains the values of P_m and T_m for the corresponding Decision Stump.

Note that in the representation of $V_{f_m(x)}$ we loss the information of the order in which the Decision Stumps were fitted during the training step. However, though in different order, all trained "weak classifiers" are codified, and thus, the final additive decision model $F(x)$ is equivalent.

2.2 OpenCL architecture

Our proposed OpenCL's architecture is based primarily on the GPU-PCIe accelerator concept, taking into account the PCIe bottleneck and the GPU's main memory or Global Memory. As we deal with large datasets of voxels, we use out-of-core techniques to subdivide the initial dataset into subset which can fit into GPU main memory. Thus, PCIe bandwidth becomes an essential factor in the overall execution.

We overview our proposed parallel architecture of OpenCL kernels in Figure 2. The eight features considered at each sample by our binary classifier are: the spatial location (x, y, z), the sampled value (v), and its associated gradient value (gx, gy, gz, $|g|$). Our binary classifier, for each feature, has $N = 3 * |numberofweigths|$

We create a matrix of Work-Groups that covers the x and y size of the dataset fitted into GPU global memory, whereas the component z is computed in a inner loop at eack kernel. Each Work-Group classifies one voxel. Insides each Workgroup, we define $N * 8$ threads, or WorkItems. Each thread compute a single operation with the 3 channels or weights of the weak classifier. These $N * 8$ values will be reduced at the end of the execution and compared to a reduced addition. The final label at each voxel is directly computed by this comparison.

3 Results

In order to present the results, we considerate different three-dimensional data from CT and MRI image modalities with different sizes. For each data set we trained a Discrete Adaboost classifiers with 32 Decision Stumps. For each voxel we considered eight features: three spatial coordinates, voxel value, three spatial derivates, and gradient magnitude. Finally, we codified the Discrete Adaboost testing classifier as described in previous sections in Matlab, C++, OpenMP, GSGL, and OpenCL codes. Our preliminary results confirm our expectations of time consuming and interactivity of the classification stage.

We achieved PCIe transfers 64 times faster than the previous GSGL version. The introduction of gradient calculation into the GPU added also a speedup having 0,0005 seconds of execution time for 128^3 voxel models in a NVIDIA Geforce GTX 470.

The execution times for the classification kernel are near half that of the GSGL version. In a

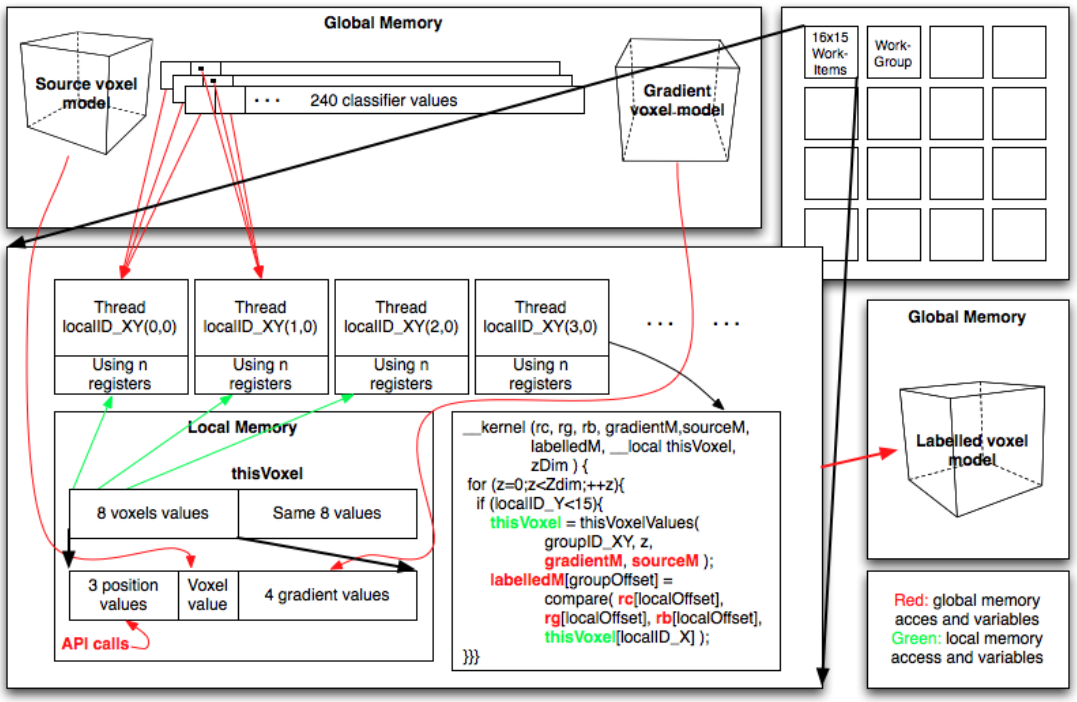

Figure 1: The proposed OpenCL classifier implementation.

future iteration, we will improve global memory reads by reading 64 times the amount of data read in each global memory transfer. That is preserving the work-group design, but using a for loop to make each work-group to classify more voxels.

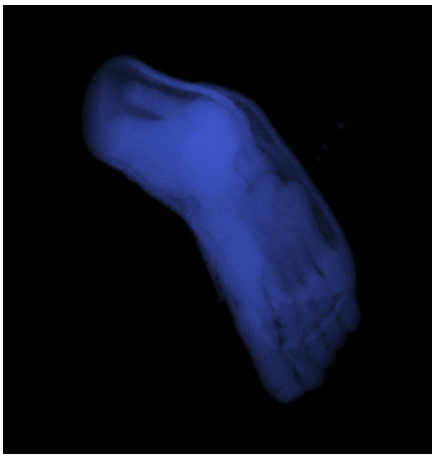

Figure 2: A 3D classified voxel model.

4 Conclusions

We presented an optimization of Adaboost test classifier based on OpenCL using the GPU capabilities. The performed experiments on large available data sets show significant improvements in terms of computational time.

References

[1] Yoav Freund, Robert E. Schapire. "Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

[2] http://www.khronos.org/opencl/

[3] Cohen, D. and Kaufman, A., "Scan Conversion Algorithms for Linear and Quadratic Objects", in Volume Visualization, pp. 280-301, 1990.

[4] Kaufman, A. and Shimony, E., '3D Scan-Conversion Algorithms for Voxel-Based Graphics', Interactive 3D Graphics, 1986.