Libsvm using GPU

Anirudhan J. Rajagopalan

N18824115

ajr619

December 15, 2016

1 Abstract

This project is an attempt to implement the Support Vector Machine [3] classifier using GPUs. We tried two approaches: The first one is to try to port the Sequential Minimal Optimization part to the GPU. The second one is to implement the Gram matrix [5] calculation to GPUs. We end up using the second approach and we show show performance improvements over the conventional approach used by libsvm.

2 Introduction

The most popular implementation of SVM is libsvm [2]. It is used in almost all the traditional machine learning applications. Since it projects data into a higher dimensional space, we were able to solve problems that are not easily classifiable in the lower dimensional space.

Porting a SVM to GPUs will help us use this classifier easily in deep learning models too. Though deep learning already uses various non-linearity to support higher dimensional data, the use of SVM might help us learn the underlying representation without using deeper layers in a deep learning model.

In this project, we explore various ways to parallelize the SVM problem and implement it in GPU.

3 Background Information

SVM is a linear classifier and is mainly successful due to its use of Hilbert space. SVM uses kernel functions which are functions in Reproducing Kernel Hilbert Space (RKHS). Implementation wise, these kernel functions can be as simple as computing a dot product between two set of features.o For the purpose of this project, we will be worrying about these simple RKHS functions. The idea is that, a non-linear lower dimensional data might be separable in a higher dimensional space.

Once, the higher dimensional features are calculated, SVM uses this higher dimensional features to find a classifier that has maximal distance from any of the points from the dataset.

The other part of SVM is the Quadratic Programming part. This part involves actually finding the plane that classifies the points in the dataset while also having a maximum margin from both the data points.

Given l examples $(\bar{x}_1, y_1), (\bar{x}_2, y_2), \ldots, (\bar{x}_l, y_l)$, with $\bar{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ $\forall i$ where the regularization is controlled by C.

$$\min_{f \in H} C \sum_{i=1}^{l} V(y_i, f(\bar{x}_i)) + \frac{1}{2}||f||_k^2 \qquad (1)$$

Here V is the loss function which is typically Hinge loss. Usage of Hinge loss or any approximation of Hing loss such as Huber-Hinge loss helps us find a maximal margin plane that classifies the samples.

Typically we solve for the dual of this actual problem, which can be given as

$$\max_{\alpha \in \mathbb{R}^l} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \alpha^T K \alpha \tag{2}$$

Subject to: $\sum_{i=1}^{l} y_i \alpha_i = 0$ and $0 \leq \alpha_i \leq C$, $i = 1, \ldots, l$ where $K_{ij} = y_i y_j k(\bar{x}i, \bar{x}_j)$ is a kernel function. Solving this requires quadratic programming approaches which gives us the classification function.

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i k(\bar{x}, \bar{x}_i) + b \tag{3}$$

With all the theory mentioned as briefly as possible, we can proceed to find a way to implement this interesting problem using GPUs.

4 Literature Survey

There are a couple of previous work in this area. We mainly use

1. Parallel Multiclass Classification Using SVMs on GPUs by Herrero-Lopez, Sergio and Williams, John R. and Sanchez, Abel [4].

2. GPU acceleration for support vector machines by Athanasopoulos, Andreas and Dimou, Anastasios and Mezaris, Vasileios and Kompatsiaris, Ioannis [1]

There are chapters in Nvidia GPU Gems book which also describe a way to parallelize the SVM implementation. It is almost similar to the method published by Herrero-Lopez.

Our implementation of SVM is going to be loosely based on libsvm's implementation. As such, it became important to understand libsvm's codebase.

Libsvm expects data in svm-light format. Each line of the input file has the structure defined in Fig 1

```
<line> .=. <target> <feature>:<value> <feature>:<value> ... <feature>:<value> # <info>
<target> .=. +1 | -1 | 0 | <float>
<feature> .=. <integer> | "qid"
<value> .=. <float>
<info> .=. <string>
```

Figure 1: Svm light format.

Libsvm uses internal structures such as *svm_node* and *svm_problem* to read the input file and store it internally. After reading the input file, it calculates the Kernel matrix which is then used for finding the maximum margin classifier.

An input file consists of samples. Each samples consists of target and features.

Each and every feature is stored in a svm_node with index and value. All these are stored in an array per sample. This structure doesn't easily translate to vector representation and hence plugging in the GPU based gram matrix calculation or SMO implementation.

5   Proposed solution

We propose two different approaches for implementing SVMs using GPU.

1. Implement the Sequential Minimal Optimization method for solving the dual problem using GPUs and OpenMpi.

2. Implement the Gram matrix calculation using GPU and use libsvm to solve the actual SMO problem.

We started out by understanding the libsvm package. The code doesn't have much documentation and understanding the code base proved to be really hard.

6   Experimental Setup

Specs of the machine, problem size, etc.

7   Experimental Result & Analysis

What are your findings. I expect much deeper description that "As we can see X is better than Y".

8   Conclusions

References

[1] Andreas Athanasopoulos, Anastasios Dimou, Vasileios Mezaris, and Ioannis Kompatsiaris. Gpu acceleration for support vector machines. In *WIAMIS 2011: 12th International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011*. TU Delft; EWI; MM; PRB, 2011.

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] Sergio Herrero-Lopez, John R. Williams, and Abel Sanchez. Parallel multi-class classification using svms on gpus. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, GPGPU-3, pages 2–11, New York, NY, USA, 2010. ACM.

[5] Wikipedia. Gramian matrix — wikipedia, the free encyclopedia, 2016. [Online; accessed 16-November-2016].