J. Parallel Distrib. Comput. 76 (2015) 16–31

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Scaling Support Vector Machines on modern HPC platforms

Yang You[a,b], Haohuan Fu[a,*], Shuaiwen Leon Song[c], Amanda Randles[d], Darren Kerbyson[c], Andres Marquez[c], Guangwen Yang[b], Adolfy Hoisie[c]

[a] *Ministry of Education Key Laboratory for Earth System Modeling, and Center for Earth System Science, Tsinghua University, Beijing, China*
[b] *Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[c] *Performance and Architecture Lab, Pacific Northwest National Lab, Richland, WA, United States*
[d] *Lawrence Livermore National Lab, Livermore, CA, United States*

HIGHLIGHTS

- An efficient parallel support vector machine for x86 based multi-core platforms.
- The novel optimization techniques to fully utilize the multi-level parallelism.
- The improvement for the deficiencies of the current SVM tools.
- Select the best architectures for input data patterns to achieve best performance.
- The large-scale distributed algorithm and power-efficient approach.

ARTICLE INFO

Article history:
Received 22 March 2014
Received in revised form
31 August 2014
Accepted 6 September 2014
Available online 21 September 2014

Keywords:
Machine learning models
Dynamic modeling
Support Vector Machine
Multi- & many-core architectures
Optimization techniques
Performance analysis

ABSTRACT

Support Vector Machines (SVM) have been widely used in data-mining and Big Data applications as modern commercial databases start to attach an increasing importance to the analytic capabilities. In recent years, SVM was adapted to the field of High Performance Computing for power/performance prediction, auto-tuning, and runtime scheduling. However, even at the risk of losing prediction accuracy due to insufficient runtime information, researchers can only afford to apply offline model training to avoid significant runtime training overhead. Advanced multi- and many-core architectures offer massive parallelism with complex memory hierarchies which can make runtime training possible, but form a barrier to efficient parallel SVM design.

To address the challenges above, we designed and implemented MIC-SVM, a highly efficient parallel SVM for x86 based multi-core and many-core architectures, such as the Intel Ivy Bridge CPUs and Intel Xeon Phi co-processor (MIC). We propose various novel analysis methods and optimization techniques to fully utilize the multilevel parallelism provided by these architectures and serve as general optimization methods for other machine learning tools.

MIC-SVM achieves 4.4–84× and 18–47× speedups against the popular LIBSVM, on MIC and Ivy Bridge CPUs respectively, for several real-world data-mining datasets. Even compared with GPUSVM, running on the NVIDIA k20x GPU, the performance of our MIC-SVM is competitive. We also conduct a cross-platform performance comparison analysis, focusing on Ivy Bridge CPUs, MIC and GPUs, and provide insights on how to select the most suitable advanced architectures for specific algorithms and input data patterns.

1. Introduction

Support Vector Machine (SVM) [10] (Figs. 1–6) is a classification method that has been widely used in text categorization [22], financial analysis [42], bioinformatics [25] and many other fields. SVM has been employed by the advanced databases like Oracle (10g, 11g, 12c) [30] as a major data mining technique as companies increasingly rely on their analytic capabilities. However, the time-consuming training process greatly limits the efficiency of SVM. This concern is likely to be magnified by the increasing volume of data in the Big Data era. Meanwhile, this issue is also exacerbated by the loss of increase in clock frequency and rise of multi- and many-core architectures, whose massive parallelism

[*] Corresponding author.
E-mail addresses: chinayouyang1991@gmail.com, you-y12@mails.tsinghua.edu.cn (Y. You).

http://dx.doi.org/10.1016/j.jpdc.2014.09.005
0743-7315/© 2014 Elsevier Inc. All rights reserved.

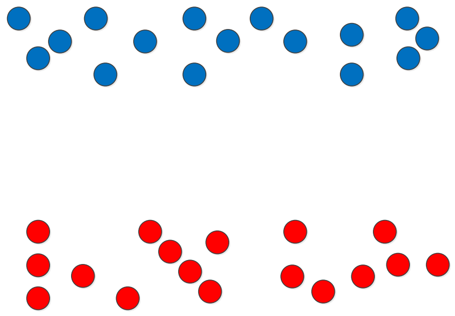
Fig. 1. Initial Data. The blue and red circles are the training points. Different colors means they belong to different classes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

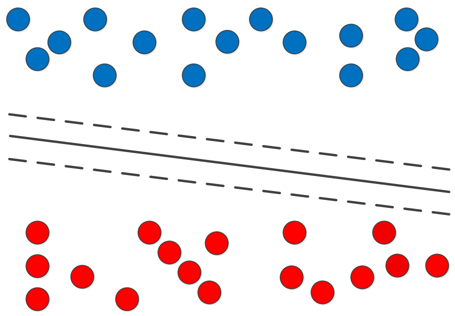
Fig. 2. First Decision Boundary. The program got an initial boundary by classifying the points into two parts.

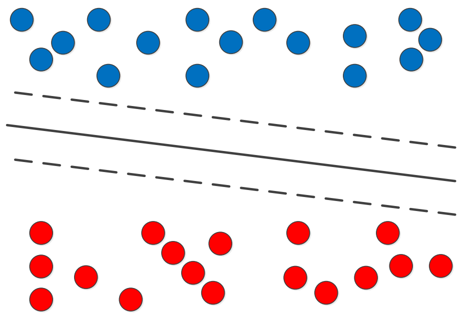
Fig. 3. Second Decision Boundary. The objective of SVM training is to maximize the distance between decision boundaries (dash lines).

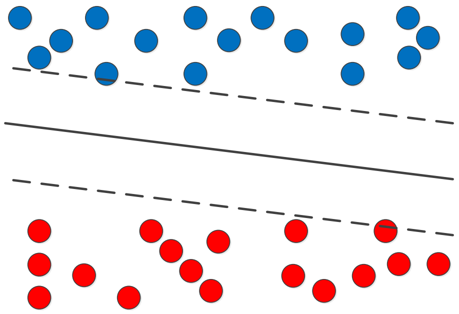
Fig. 4. Third Decision Boundary. The program got a better decision boundary because the distance is larger.

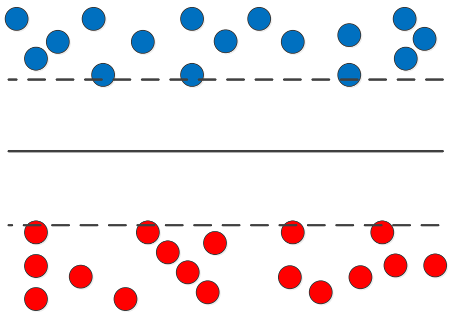
Fig. 5. Best Decision Boundary. The program finally got the best decision boundary by maximizing the distance.

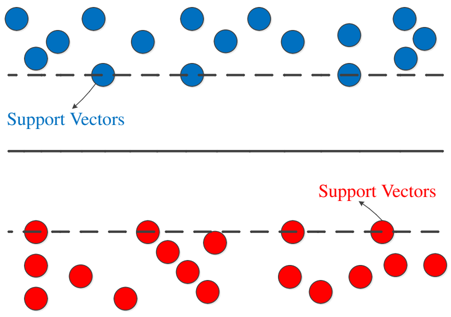

Fig. 6. Mark Support Vectors. The support vector refers to the blue and red circles on the decision boundary. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and complex memory hierarchies form a barrier to efficient parallel SVM design (due to its memory bound and irregular memory access bottlenecks). While programmers in the past could depend on the ready-made performance improvement that comes from a faster clock frequency, now they have to face the challenge of scaling the performance over tens or even hundreds of cores within a single chip often operating at a lower frequency [38]. One of the most representative multicore architectures is Intel Xeon Phi Coprocessor (MIC) [7].

In the field of High Performance Computing (HPC), SVMs have recently been adapted at runtime for performance and power prediction at system and compiler level for design space exploration [41] [45]. The common approach of the current work is to train the SVM model offline and then apply the static model for online prediction in order to avoid significant performance overheads. However, this dramatically reduces the flexibility of the runtime system; it may also increase the chance of model misprediction due to the lack of current runtime behavior information. In order to use SVM at runtime for dynamic modeling and scheduling in HPC, a method is needed to accelerate its training phase on modern advanced multi- and many-core architectures.

Previous work either focused on designing SVM tools for CPUs with relatively few cores and simple memory hierarchies, such as the serial LIBSVM [6] or [19], or creating techniques to accelerate SVM on GPUs such as GPUSVM [4]. However, there has been no efficient SVM tool designed for advanced x86 based multi- and many-core architectures, even though they have already gained popularity on recent TOP500 list [12]. Meanwhile, there are several design deficiencies (i.e. no adaptive runtime support for efficient memory management and data parallelism) within the existing tools such as LIBSVM and GPUSVM, that will ultimately limit the performance improvements for future architectures.

In this paper, we present MIC-SVM, a highly efficient parallel support vector machine designed for x86 based multi-core and many-core architectures such as Ivy Bridge CPUs and Intel Knight Corner (KNC) MIC [13]. Our goal is to design and implement an open-source SVM Library that is not only highly efficient but also can be easily adopted to the existing runtime systems or software. We also want to create methods and techniques that are general enough to be applied to optimize other similar machine learning models.