SVM Classification

David Eliahu Shaddi Hasan Omer Spillinger

May 14, 2013

1 Introduction

Computer scientists develop algorithms—it's just what we do, we can't help it. Algorithm designers produce an ever-increasingly set of techniques for solving an ever-wider set of problems. Indeed, algorithm selection itself has become a challenging problem, particularly in fields where performance is critical such as high-performance computing and machine learning. The factors that influence algorithm performance are extremely complicated, and understanding how an algorithm will perform given an input and a particular machine can require extensive domain knowledge. Performance can even vary significantly from system to system due to differences in factors such as memory hierarchy, machine architecture, utilization, and so on. This problem is even more pronounced on virtualized infrastructure: the application developer really has no way of knowing how the underlying hardware will perform, or even if her application will always be running on the same hardware (e.g., due to virtual machine migration).

All this made us ask ourselves, how can we *predict* the optimal algorithm to use in a given situation? Being able to do so would be a significant benefit to developers: such a technique could be incorporated into libraries so that developers could write software oblivious of the underlying algorithmic implementation, confident that the library would choose the proper algorithm.

One solution to this problem would be to develop a model of machine performance and analytically determine the best algorithm to use for a given input. Clearly, this approach wouldn't scale—the model would be enormously complex, and would need to be developed for every combination of machine and algorithm.

One could also take an empirical approach: simply measure how algorithms perform on a given machine under different inputs. A naïve solution would be to exhaustively search the space of possible inputs; this is intractable in the general case, and even in situations where the range of possible inputs is finite exhaustive search is time consuming.

The complexity of performance classification lends itself to a machine learning approach. Rather than building a complex model of machine performance, we can train a classifier on a set of training examples to estimate the best algorithm to use. In contrast to exhaustive search, a classifier-based approach need not explore a large portion of the search space. The user can make an explicit tradeoff between classifier accuracy and training time.

We take this classifier-based approach in our project. In this work, we focus on dense matrix multiplication (DMM). We chose this problem for several reasons. First, DMM is an important step in many HPC and machine learning tasks, and it often represents the computational performance bottleneck for those tasks. Thus, even relatively small performance gains for DMM can translate to real savings for large tasks. Secondly, a plethora of algorithms for DMM exist in the literature. Two of the authors have developed a communication-avoiding algorithm for DMM called CARMA; CARMA's approach is substantially different than the industry-standard algorithm MKL, making it likely that it would have different performance properties. Finally, DMM is easily parameterizable—in general, performance is dictated by the dimensions of the input matrices.

We perform classification using a support vector machine. We evaluate our classifier on a set of multiplications of matrices ranging in size from 64 x 64 to 3250 x 3250—a state space of over 32 billion possible inputs. We find that our classifier achieves good results across multiple machines, with F1 scores ranging from 0.75 to 0.88, and that training on only a few hundred points is sufficient to achieve this level of accuracy. Finally, using our approach we observe performance increases of up to 28% versus choosing an algorithm a priori.

The remainder of the paper is organized as follows. We first discuss our dataset and its generation. We then describe our approach towards evaluation and analysis. In section **??** we describe our results before considering future directions and concluding.

2 Data

To perform our analysis