TOP: A Compiler-Based Framework for Optimizing Machine Learning Algorithms through Generalized Triangle Inequality

Yufei Ding
UC Santa Barbara
yufeiding@cs.ucsb.edu

Lin Ning, Hui Guang, Xipeng Shen
North Carolina State University
{lning,hguan2,xshen5}@ncsu.edu

Madanlal Musuvathi, Todd Mytkowicz
Microsoft Research
{madanm,toddm}@microsoft.com

ABSTRACT

This paper describes our recent research progress on generalizing triangle inequality (TI) to optimize Machine Learning algorithms that involve either vector dot products (e.g., Neural Networks) or distance calculations (e.g., KNN, K-Means). The progress includes a new form of TI named Angular Triangular Inequality, abstractions to enable unified treatment to various ML algorithms, and TOP, a compiler-based optimizer for effectively applying TI to optimize machine learning algorithms. Experiments show that TOP is able to automatically produce optimized algorithms that either matches or outperforms manually designed algorithms, giving up to 237x speedups and 2.5X on average[1].

1. INTRODUCTION

Vector dot products and point-to-point distance calculations are essential to many important algorithms across various domains. Vector dot products, for instance, are the core operations in artificial neural networks, as well as algorithms used in document clustering and document retrieval. Distance calculations are the essential computations in many othe the machine learning algorithms. The commonly used clustering algorithm K-Means [15], for example, is an iterative algorithm which computes the distances between every data point and each of a set of K cluster centers in order to decide which center is closest to each point. Other examples include K-Nearest Neighbor (KNN) [12], point-to-point shortest path in graphs [4], 3D image construction [3], and so on. In each of these algorithms, vector dot products or distance calculations over a large number of data points form the typical performance bottleneck.

Our research was initially inspired by previous manual efforts by domain experts [8, 10] to apply the Triangle Inequality (TI) theorem to optimize distance calculations for some of those algorithms. TI says that the sum of two edges of a triangle must be greater than the third edge. With TI, one may easily get the lower and upper bounds of the distances between two points. The bounds could be safely used (and reused) in place of the actual distances in suitable scenarios to save some distance calculations.

Domain experts have tried to leverage TI to manually create variations of some machine learning algorithms to reduce the number of distance calculations. Creating these efficient variants is often difficult and domain specific—the variants often differ in distance definition (i.e., what defines distance), calculation constraints (i.e., how they are employed), context of usage (i.e., when to use optimized implementations of distance definitions), and other aspects. Thus, these previous efforts are often *problem-specific*. Coming up with such a solution usually takes the domain experts deep and nontrivial insights, theoretical analysis, and empirical measurements. This point is empirically backed up by the large number of research papers published in the premium venues in the domains. For instance, in the recent 10 years of top machine learning or data mining conferences, there are more than 20 papers on developing algorithms to optimize distance calculations for K-Means (e.g., [8, 11, 13, 16, 19]).

Our work is motivated by an observation that, despite the many differences among those problems, the underlying mechanisms in which distance calculations have been optimized share commonality. This observation prompts us to examine the algorithm design process from the perspective of programming systems: If we can generalize the various distance-related problems into a single abstract form, we can develop an optimizing framework as a unified solution to such algorithmic optimization problems. As a consequence, such a framework saves the significant manual effort previously required to optimize distance-related algorithms. Also the framework could provide a more systematic treatment to existing distance-related problems than previous manually optimized algorithms do, with even better performance.

We further expend the thought to vector dot products based on the connections between vector dot products and point-to-point distances. Moreover, we generalize the theory of TI by developing a new type of TI, named *Angle Triangular Inequality* (ATI) [5]. ATI considers angles among vectors rather than edges; it significantly expands the applicability of TI-based optimizations, and at the same time, enhances the tightness of the bounds.

The result of our exploration is *Triangular OPtimizer* (TOP), a compiler and runtime software framework that enables automatic algorithmic optimizations for various distance-based problems and vector dot products involved in various machine learning algorithms. Unlike typical program optimizations that optimize an implementation of an algorithm

[1]This paper is based on the series of results we published at PLDI'2017 [5], VLDB'2015 [6], ICML'2015 [7].

```
/*
Goal: Cluster points in S into K classes with T containing all cluster centers.
S: a set of query points to cluster.
T: a set of target points (i.e., cluster centers).
N: a set of indices of points. |N|=|S|.
*/
... // declarations
TOP_defDistance(Euclidean); // distance definition
T = init();
changedFlag = 1;
while (changedFlag){
    // find the closest target (a point in T) for each point in S
    N = TOP_findClosestTargets(1, S, T);
    TOP_update(T, &changedFlag, N, S); // T gets updated
}
```

Figure 1: K-Means written in TOP API. Prefix "TOP_" indicates calls to TOP API. They will be replaced with low-level function calls by TOP compiler, making the algorithm automatically avoid unnecessary distance calculations.

at an instruction level, the *algorithmic* optimizations that TOP enables change the algorithm used to find out relations between data points, thus generating new algorithms that are up to hundreds of times faster than existing ones.

With TOP, users specify the distance problem using a set of high-level and relatively intuitive APIs. TOP then automatically creates an optimized algorithm that minimizes the distance calculations for that problem. We have found that TOP is applicable to many problems involving vector dot products or distance-based calculations that meet the Triangular Inequality condition, regardless of the domain, definition of distances, distance calculation patterns, usage of the distances, and so on. Its generated algorithm matches or outperforms the algorithms manually designed by the domain experts. With TOP, many manual efforts by domain experts could have been saved; it makes optimizing new problems much easier, and boost the performance of existing algorithms.

Specifically, we propose a simple abstraction, called *abstract distance-related problem*, to formalize various distance-related algorithms across seemingly disparate domains, in a unified manner. The abstraction allows a systematic examination of all kinds of scenarios related with distance computations, which in turn, leads to a spectrum of algorithmic optimization along with some automatic mechanisms for selecting the best optimization to use. For vector dot products, we implement code pattern matching in a source-level compiler to recognize vector dot products whose results are used only for some value comparisons, and then conduct an automatic code transformation to apply TI or ATI to save unnecessary vector dot products.

Through empirical explorations, we attain a set of insights on the suitable ways to apply TI or ATI for various kinds of problems. We turn these findings into a runtime library, the invocations of which in a program would automatically save unnecessary distance calculations or vector dot products.

Along with the library, we equip TOP with a set of APIs and a compilation module. Through the API, programmers can easily specify the problem of interest, as illustrated in Figure 1. The compiler module then derives important properties of the problem, and inserts necessary calls of the library such that at runtime, unnecessary distance calculations can be effectively detected and avoided.

2. EXPERIMENTAL RESULTS

We evaluate the technique on eight benchmarks. The first five benchmarks (KNNN [12], KNNjoin [1], KMeans [15], ICP [3], NBody [9]) are distance-related problems, and the other three, (Document clustering (DC) [14], Top-K Document Retrieval (KDR) [20], Restricted Boltzman Machine (RBM) Neural Networks [18]) involve dot products.

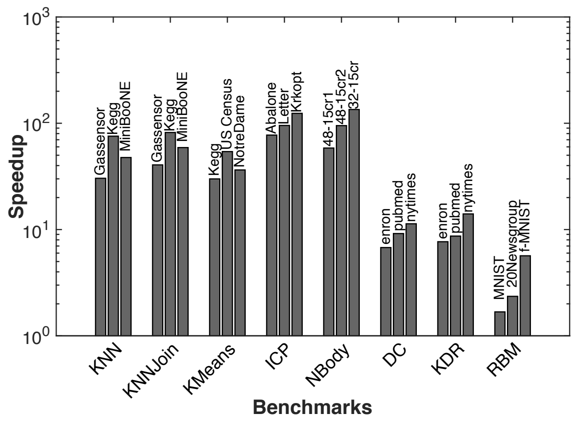

Figure 2: The graph shows the speedup over the original implementation of the default algorithms on Intel i5-4570 CPU and 16G memory; labeled with datasets.

The graph in Figure 2 gives our speedups on each dataset over the *standard* version (i.e., the original implementations of the default algorithms.) Compared with the standard version, which does not use TI-based optimizations, our technique achieves as much as 134X (NBody) speedups and 46X on average. The technique also applies to GPU versions of the code, yielding significant speedups as well; on an Kepler GPU, the TI-optimized KNN outperforms a CUBLAS version by up to 120X (11X on average) [2], and the optimized RBM shows 3X speedups [17].

The accelerations come primarily from the savings of distance or dot product computations. Although the mount of savings vary, depending on many factors, we observe over 91% computation savings for all the datasets tested on these benchmarks other than RBM. In particular, we notice that the savings are often more prominent for larger input and problem settings (e.g., dataset size, data dimensions, and the number of clusters). Dataset size is the most influential factor across all benchmarks.

The overhead of bound computations is always negligible compared to the original computation cost in the standard version without TI optimization. The reason for this is twofold. First, bound computation itself is a scalar operation, while both distance and dot product computation are vector operations. When the data dimension is high, the cost of bound computation is much smaller than that of direct distance and dot production computation. Second, the total number of bound computations carried out is much smaller than that of distance and dot product computations required in a standard version.

3. REFERENCES

[1] C. Böhm and F. Krebs. The k-nearest neighbour join: Turbo charging the kdd process. *Knowledge and Information Systems, Springer*, 6(6):728–749, 2004.

[2] G. Chen, Y. Ding, and X. Shen. Sweet knn: An efficient knn on gpu through reconciliation between redundancy removal and regularity. In *The Proceedings of IEEE International Conference on Data Engineering*, 2017.

[3] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, IEEE*, pages 2724–2729, 1991.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische mathematik*, volume 1, pages 269–271, 1959.

[5] Y. Ding, L. Ning, H. Guan, and X. Shen. Generalizations of the theory and deployment of triangular inequality for compiler-based strength reduction. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2017.

[6] Y. Ding, X. Shen, M. Musuvathi, and T. Mytkowicz. Top: A framework for enabling algorithmic optimizations for distance-related problems. In *Proceedings of the 41st International Conference on Very Large Data Bases*, 2015.

[7] Y. Ding, X. Shen, M. Musuvathi, and T. Mytkowicz. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *ICML*, 2015.

[8] J. Drake and G. Hamerly. Accelerated k-means with adaptive distance bounds. In *5th NIPS Workshop on Optimization for Machine Learning*, 2012.

[9] V. Eijkhout. *Introduction to High Performance Scientific Computing*. Lulu. com, 2010.

[10] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, pages 147–153, 2003.

[11] A. Fahim, A. Salem, F. Torkey, and M. Ramadan. An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A, Springer*, 7(10):1626–1633, 2006.

[12] E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. In *DTIC Document*, 1951.

[13] G. Hamerly. Making k-means even faster. In *SDM, SIAM*, pages 130–140, 2010.

[14] A. Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[15] S. Lloyd. Least squares quantization in pcm. In *Information Theory, IEEE*, volume 28,2, pages 129–137, 1982.

[16] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip. Efficient clustering of uncertain data. In *Data Mining, 2006. ICDM'06, IEEE*, pages 436–445, 2006.

[17] L. Ning, R. Pittman, and X. Shen. LCD: A fast contrastive divergence based algorithm for restricted boltzmann machine. In *The Proceedings of IEEE International Conference on Data Mining*, 2017.

[18] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, New York, NY, USA, 2008. ACM.

[19] J. Wang, J. Wang, Q. Ke, G. Zeng, and S. Li. Fast approximate k-means via cluster closures. In *Computer Vision and Pattern Recognition (CVPR), IEEE*, pages 3037–3044, 2012.

[20] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.