Journal of Machine Learning Research 19 (2018) 1-5 Submitted 12/17; Revised 6/18; Published 7/18

ThunderSVM: A Fast SVM Library on GPUs and CPUs

Zeyi Wen[†] WENZY@COMP.NUS.EDU.SG
Jiashuai Shi[†‡] SHIJIASHUAI@GMAIL.COM
Qinbin Li[†] LIQINBIN1998@GMAIL.COM
Bingsheng He[†] HEBS@COMP.NUS.EDU.SG
Jian Chen[‡] ELLACHEN@SCUT.EDU.CN
[†] *School of Computing, National University of Singapore, 117418, Singapore*
[‡] *School of Software Engineering, South China University of Technology, Guangzhou, 510006, China*

Editor: Alexandre Gramfort

Abstract

Support Vector Machines (SVMs) are classic supervised learning models for classification, regression and distribution estimation. A survey conducted by Kaggle in 2017 shows that 26% of the data mining and machine learning practitioners are users of SVMs. However, SVM training and prediction are very expensive computationally for large and complex problems. This paper presents an efficient and open source SVM software toolkit called ThunderSVM which exploits the high-performance of Graphics Processing Units (GPUs) and multi-core CPUs. ThunderSVM supports all the functionalities—including classification (SVC), regression (SVR) and one-class SVMs—of LibSVM and uses identical command line options, such that existing LibSVM users can easily apply our toolkit. ThunderSVM can be used through multiple language interfaces including C/C++, Python, R and MATLAB. Our experimental results show that ThunderSVM is generally an order of magnitude faster than LibSVM while producing identical SVMs. In addition to the high efficiency, we design our convex optimization solver in a general way such that SVC, SVR, and one-class SVMs share the same solver for the ease of maintenance. Documentation, examples, and more about ThunderSVM are available at `https://github.com/zeyiwen/thundersvm`.

Keywords: SVMs, GPUs, multi-core CPUs, efficiency, multiple interfaces

1. Introduction

Support Vector Machines (SVMs) have been widely used in many applications including document classification (D'Orazio et al., 2014), image classification (Pasolli et al., 2014), blood pressure estimation (Kachuee et al., 2015), disease detection (Bodnar and Salathé, 2013), and outlier detection (Roth, 2006). A survey conducted by Kaggle in 2017 shows that 26% of the data science practitioners use SVMs to solve their problems (Thomas, 2017). The open-source project LibSVM which supports classification (SVC), regression (SVR) and one-class SVMs has been widely used in many applications. LibSVM was developed in 2000 (Chang and Lin, 2011), and has been maintained since then. Despite the advantages of SVMs, SVM training and prediction are very expensive for large and complex problems.

Graphics Processing Units (GPUs) have been used to accelerate the solutions of many real-world applications (Dittamo and Cisternino, 2008), due to the abundant computing

©2018 Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He and Jian Chen.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v19/17-740.html`.

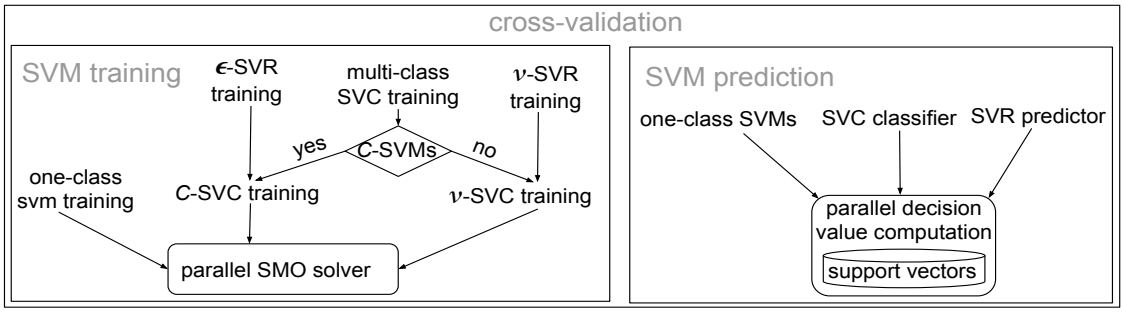

Figure 1: Overview of training and prediction

cores and high memory bandwidth of GPUs. In this paper, we introduce a toolkit named *ThunderSVM* which exploits GPUs and multi-core CPUs. The mission of our toolkit is to help users easily and efficiently apply SVMs to solve problems. It is worthy to point out that one way to train SVMs faster is to use kernel approximations. ThunderSVM aims to find an exact solution. ThunderSVM supports all the functionalities of LibSVM including SVC, SVR and one-class SVMs. We use the same command line input options as LibSVM, such that existing LibSVM users are able to easily switch to ThunderSVM. Moreover, ThunderSVM supports multiple interfaces such as C/C++, Python, R and MATLAB. ThunderSVM can run on Linux, Windows or Macintosh operating systems with or without GPUs. Empirical results show ThunderSVM is generally 10 times faster than LibSVM in all the functionalities. The full version of ThunderSVM, which is released under Apache License 2.0, can be found on GitHub at `https://github.com/zeyiwen/thundersvm`. The GitHub repository of ThunderSVM has attracted 700 stars and 85 forks as of July 24, 2018.

2. Overview and Design of ThunderSVM

Like LibSVM, ThunderSVM supports one-class SVMs, C-SVMs and ν-SVMs where C represents the regularization constant and ν represents the parameter controlling the training error. Both C-SVMs and ν-SVMs are used for classification and regression.

Figure 1 shows the overview of ThunderSVM which has many functionalities: one-class SVMs for distribution estimation, C-SVC and ν-SVC for SVM classification, and ϵ-SVR and ν-SVR for SVM regression. The training algorithms for those SVMs are built on top of a generic parallel SMO solver which is for solving quadratic optimization problems. Notably, the SVM training for regression (such as ϵ-SVR and ν-SVR) and the multi-class SVM training can be converted into the training of an SVM classifier. The prediction module is relatively simple, because the prediction is the same for one-class SVMs, C-SVMs and ν-SVMs. The prediction is essentially computing predicted values based on support vectors. ThunderSVM also contains the cross-validation functionality.

2.1 Design of Parallel SVM Training Algorithms

We have developed a series of optimizations for the training. First, ThunderSVM computes a number of rows of the kernel matrix in a batch, reuses the rows that are stored in the GPU

memory buffer, and solves multiple subproblems in that batch. Thus, ThunderSVM avoids performing a large number of small read/write operations to the high latency memory and reduces repeated kernel value computation. Moreover, we apply GPU shared memory to accelerate parallel reduction, and use the massive parallelism to update elements of arrays.

For solving each subproblem in the training, we use the SMO algorithm which consists of three key steps. Step (i): Find two extreme training instances which can potentially improve the currently trained SVM the most. Step (ii): Improve the two Lagrange multipliers of the two instances. Step (iii): Update the optimality indicators of all the training instances. We parallelize Step (i) and (iii). Step (ii) is computationally inexpensive, and we simply execute it sequentially. In Step (i), our key idea is to apply the parallel reduction (Merrill, 2015) twice for finding the two extreme training instances. In the parallel reduction, we first load the whole array from the GPU global memory to shared memory in a coalescent way, and then reduce the array size by two at each iteration until only one element left. In Step (iii), we dedicate one thread to update one optimality indicator to use the massive parallelism mechanism of the GPU. The training is terminated when the optimality condition is met or the SVMs cannot be further improved. More details about the training and the termination condition can be found in our supplementary material (Wen et al., 2017).

For solving the batch of subproblems, we propose techniques to exploit the properties of the batch of subproblems. First, to reduce access to the high latency memory, the kernel values needed for the batch are organized together and computed through matrix multiplication. As a result, we reduce a large number of small read/write operations to the high latency memory during kernel value computation. We use matrix operations from the high-performance library cuSparse (Nvidia, 2008) provided by NVIDIA. Second, to reduce repeated kernel value computation, we store the kernel values in a GPU buffer for efficient reuse during the optimization for the batch. Regarding the selection of the batch, we make use of the set of instances with the deepest gradients.

Training SVMs for regression (SVR) or for multi-class classification can be reduced to training SVMs for binary classification (SVC), as discussed in the previous study (Shevade et al., 2000). Two SVC training algorithms (C-SVC and ν-SVC) and training algorithm for one-class SVMs are essentially solving optimization problems using SMO. More details about the relationship of SVR and SVC, and SMO can be found in our supplementary material (Wen et al., 2017). The key task in the SVM training is to parallelize SMO, and the insight has been discussed above. The parallelism principles are applicable to CPUs.

2.2 Design of the Parallel Prediction Algorithm

Although ThunderSVM supports several algorithms (such as one-class SVMs, classification and regression), their underlying prediction algorithm is identical: a function based on the support vectors and their Lagrange multipliers. The function is $v = \sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b$ where $\boldsymbol{x}_j$ is the instance of interest for prediction; y_i and α_i are the label and Lagrange multiplier of the support vector $\boldsymbol{x}_i$, respectively; b is the bias of the SVM hyperplane; $K(\cdot, \cdot)$ is the kernel function. In ThunderSVM, we perform the prediction by evaluating the equation in parallel. First, we conduct a vector to matrix multiplication in parallel to obtain all the needed kernel values, where the vector is $\boldsymbol{x}_j$ and the matrix consists of all the support vectors. Then, the sum of the equation can be performed using a parallel reduction.

data set			elapsed time (sec)			speedup	
name	cardinality	dimension	ThunderSVM		LibSVM		
			gpu	cpu		gpu	cpu
mnist8m (svc)	8.1x10^6	784	7.1x10^3	3.2x10^4	8.1x10^5	114	39.9
rcv1_test (svc)	677399	47236	621	20633	8.3x10^5	1337	40
epsilon (svc)	400000	2000	1251	26042	1 week+	483+	23+
e2006-tfidf (svr)	16087	150360	13.25	343.5	9161	691	25.3
webdata (ocsvm)	49749	300	4.66	16.5	1493	320	90.5

Table 1: Comparison between ThunderSVM with LibSVM

3. Experimental Studies

We compare the efficiency on training SVMs for classification, regression and one-class SVMs (denoted by "OCSVM"). Five representative data sets are listed in Table 1. We conducted our experiments on a workstation running Linux with two Xeon E5-2640 v4 10 core CPUs, 256GB main memory and a Tesla P100 GPU of 12GB memory. ThunderSVM are implemented in CUDA-C and C++ with OpenMP. We used the Gaussian kernel. Five pairs of hyper-parameters (C, γ) for the data sets are (10, 0.125), (100, 0.125), (0.01, 1), (256, 0.125), and (64,7.8125) and are the same as the existing studies (Wen et al., 2014, 2018). More experimental evaluation can be found in our supplementary material. ThunderSVM when using GPUs is over 100 times faster than LibSVM. When running on CPUs, it is over 10 times faster than LibSVM. For prediction, ThunderSVM is also 10 to 100 times faster than LibSVM (Wen et al., 2017). We varied the hyper-parameters C from 0.01 to 100 and γ from 0.03 to 10, and ThunderSVM is 10 to 100 times faster than LibSVM.

4. Conclusion

In this paper, we present our software tool called "ThunderSVM" which supports all the functionalities of LibSVM. For ease of usage, ThunderSVM uses identical input command line options as LibSVM, and supports Python, R and Matlab. Empirical results show that ThunderSVM is generally 100 times faster than LibSVM in all the functionalities when GPUs are used. When running purely on CPUs, ThunderSVM is often 10 times faster than LibSVM. We hope this significant efficiency improvement would help practitioners in the community quickly solve their problems and enable SVMs to solve more complex problems.

Acknowledgments

This work is supported by a MoE AcRF Tier 1 grant (T1 251RES1610) and Tier 2 grant (MOE2017-T2-1-122) in Singapore. Prof. Chen is supported by the Guangdong special branch plans young talent with scientific and technological innovation (No. 2016TQ03X445), Guangzhou science and technology planning project (No. 2019-03-01-06-3002-0003) and Guangzhou Tianhe District science and technology planning project (No. 201702YH112). Bingsheng He and Jian Chen are corresponding authors. We acknowledge NVIDIA for the hardware donations and thank the anonymous reviewers for their insightful comments.

References

Todd Bodnar and Marcel Salathé. Validating models for disease detection using Twitter. In *International Conference on World Wide Web (WWW)*, pages 699–702. ACM, 2013.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Cristian Dittamo and Antonio Cisternino. GPU White paper, 2008.

Vito D'Orazio, Steven T Landis, Glenn Palmer, and Philip Schrodt. Separating the wheat from the chaff: applications of automated document classification using Support Vector Machines. *Political Analysis*, 22(2):224–242, 2014.

Mohamad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1006–1009. IEEE, 2015.

Duane Merrill. CUB v1. 5.3: CUDA unbound, a library of warp-wide, block-wide, and device-wide GPU parallel primitives, 2015.

CUDA Nvidia. Cublas library. *NVIDIA Corporation, Santa Clara, California*, 15(27):31, 2008.

Edoardo Pasolli, Farid Melgani, Devis Tuia, Fabio Pacifici, and William J Emery. SVM active learning approach for image classification using spatial information. *IEEE Transactions on Geoscience and Remote Sensing*, 52(4):2217–2233, 2014.

Volker Roth. Kernel fisher discriminants for outlier detection. *Neural Computation*, 18(4): 942–960, 2006.

Shirish Krishnaj Shevade, S Sathiya Keerthi, Chiranjib Bhattacharyya, and Karaturi Radha Krishna Murthy. Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193, 2000.

Amber Thomas. Kaggle 2017 survey results: `https://www.kaggle.com/amberthomas/kaggle-2017-survey-results`, 2017.

Zeyi Wen, Rui Zhang, Kotagiri Ramamohanarao, Jianzhong Qi, and Kerry Taylor. MASCOT: fast and highly scalable SVM cross-validation using GPUs and SSDs. In *IEEE International Conference on Data Mining (ICDM)*, pages 580–589. IEEE, 2014.

Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. Supplementary material of ThunderSVM: `https://github.com/zeyiwen/thundersvm/blob/master/thundersvm-full.pdf`, 2017.

Zeyi Wen, Rui Zhang, Kotagiri Ramamohanarao, and Li Yang. Scalable and fast SVM regression using modern hardware. *World Wide Web*, 21(2):261–287, 2018.