Journal of Machine Learning Research 14 (2013) 3813-3817 Submitted 4/13; Revised 9/13; Published 12/13

# BudgetedSVM: A Toolbox for Scalable SVM Approximations

**Nemanja Djuric** NEMANJA@TEMPLE.EDU
**Liang Lan** LANLIANG@TEMPLE.EDU
**Slobodan Vucetic** VUCETIC@TEMPLE.EDU
*304 Wachman Hall, Temple University*
*1805 North Broad Street*
*Philadelphia, PA 19122, USA*

**Zhuang Wang** ZJWANG@US.IBM.COM
*Global Business Services, IBM*
*1475 Phoenixville Pike*
*West Chester, PA 19380, USA*

**Editor:** Antti Honkela

## Abstract

We present BudgetedSVM, an open-source C++ toolbox comprising highly-optimized implementations of recently proposed algorithms for scalable training of Support Vector Machine (SVM) approximators: Adaptive Multi-hyperplane Machines, Low-rank Linearization SVM, and Budgeted Stochastic Gradient Descent. BudgetedSVM trains models with accuracy comparable to LibSVM in time comparable to LibLinear, solving non-linear problems with millions of high-dimensional examples within minutes on a regular computer. We provide command-line and Matlab interfaces to BudgetedSVM, an efficient API for handling large-scale, high-dimensional data sets, as well as detailed documentation to help developers use and further extend the toolbox.

**Keywords:** non-linear classification, large-scale learning, SVM, machine learning toolbox

## 1. Introduction

Support Vector Machines (SVMs) are among the most popular and best performing classification algorithms. Kernel SVMs deliver state-of-the-art accuracies on non-linear problems, but are characterized by linear growth in the number of support vectors with data size, which may prevent learning from truly large data. In contrast, linear SVMs cannot capture non-linear concepts, but are very scalable and allow learning from large data with limited resources. Aimed at bridging the representability and scalability gap between linear and non-linear SVMs, recent advances in large-scale learning resulted in powerful algorithms that enable scalable training of non-linear SVMs, such as Adaptive Multi-hyperplane Machines (AMM) (Wang et al., 2011), Low-rank Linearization SVM (Zhang et al., 2012), and Budgeted Stochastic Gradient Descent (BSGD) (Wang et al., 2012). With accuracies comparable to kernel SVM, the algorithms are scalable to millions of examples, having training and inference times comparable to linear and orders of magnitude shorter than kernel SVM.

We present BudgetedSVM, an open-source C++ toolbox for scalable non-linear classification. The toolbox provides an Application Programming Interface (API) for efficient training and testing of non-linear classifiers, supported by data structures designed for handling data which cannot fit in memory. BudgetedSVM can be seen as a missing link between LibLinear and LibSVM (Hsieh et al., 2008; Chang and Lin, 2011), combining the efficiency of linear with the accuracy of kernel SVM.

We also provide command-line and Matlab interfaces, providing users with an efficient, easy-to-use tool for large-scale non-linear classification.

## 2. Non-linear Classifiers for Large-scale Data

Before taking a closer look at the implementation and usage details of the BudgetedSVM toolbox, in this section we give a brief description of the implemented algorithms.

### 2.1 Adaptive Multi-hyperplane Machines (AMM)

Wang et al. (2011) proposed a classifier that captures non-linearity by assigning a number of linear hyperplanes to each of $C$ classes from a set $\mathcal{Y}$. Given a $D$-dimensional example $\mathbf{x}$, the AMM multi-class classifier has the following form,

$$f(\mathbf{x}) = \arg\max_{i \in \mathcal{Y}} g(i,\mathbf{x}), \text{ where } g(i,\mathbf{x}) = \max_{j=1,\ldots,b_i} \mathbf{w}_{ij}^{\mathrm{T}}\mathbf{x}, \tag{1}$$

where the $i^{\text{th}}$ class is assigned $b_i$ weight vectors with the total budget $B = \sum_i b_i$. AMM is learned via Stochastic Gradient Descent (SGD). The hyper-parameters include a regularization parameter $\lambda$, the number of training epochs $e$, the maximum number of non-zero weights per class $B_{lim}$, $b_i \leq B_{lim}$, and weight pruning parameters $k$ (pruning frequency) and $c$ (pruning aggressiveness). As an initial guideline to the users, we experimentally found that for most data sets the values $e = 5$ (or $e = 1$ for very large data), $B_{lim} = 50$, $k = 10{,}000$, and $c = 10$ are appropriate choices, leaving only $\lambda$ to be determined by cross-validation.

When $b_1, \ldots, b_C$ are fixed to 1, the AMM model reduces to linear multi-class SVM (Crammer and Singer, 2002), and the learning algorithm is equivalent to Pegasos, a popular linear SVM solver (Shalev-Shwartz et al., 2007). As it is a widely-used linear SVM solver, we also provide the Pegasos algorithm directly as a shortcut in the BudgetedSVM toolbox.

### 2.2 Low-rank Linearization SVM (LLSVM)

Zhang et al. (2012) proposed to approximate kernel SVM optimization by a linear SVM using low-rank decomposition of the kernel matrix. The approximated optimization is solved via Dual Coordinate-Descent (DCD) (Hsieh et al., 2008). The binary classifier has the form

$$f(\mathbf{x}) = \operatorname{sign}\big(\mathbf{w}^{\mathrm{T}}(\mathbf{M} \cdot g(\mathbf{x}))\big),$$

where $g(\mathbf{x}) = [k(\mathbf{x},\mathbf{z}_1), \ldots, k(\mathbf{x},\mathbf{z}_B)]^{\mathrm{T}}$, $\{\mathbf{z}_i\}_{i=1,\ldots,B}$ is a set of landmark points of size $B$, $k(\mathbf{x},\mathbf{z}_i)$ is a kernel function, $\mathbf{w}$ defines a separating hyperplane in the linearized kernel space (found using the DCD method), and $\mathbf{M}$ is a $B \times B$ mapping matrix. The hyper-parameters include kernel parameters, regularization parameter $\lambda$, and the number of landmark points $B$. Parameter $B$ controls a trade-off between speed and accuracy, while kernel parameters and $\lambda$ are best determined by cross-validation.

### 2.3 Budgeted Stochastic Gradient Descent (BSGD)

Wang et al. (2012) proposed a budgeted algorithm which maintains a fixed number of support vectors in the model, and incrementally updates them during the SGD training. The multi-class BSGD

| | Pegasos | AMM | LLSVM | BSGD | RBF-SVM |
|---|---|---|---|---|---|
| Training time | $O(NCS)$ | $O(NSB)$ | $O(NSB^2+NSB)$ | $O\big(N(C+S)B\big)$ | $O(INCS)$ |
| Prediction time | $O(CS)$ | $O(SB)$ | $O(SB^2+SB)$ | $O\big((C+S)B\big)$ | $O(NCS)$ |
| Model size | $O(CD)$ | $O(DB)$ | $O(DB+B^2)$ | $O\big((C+D)B\big)$ | $O(NCS)$ |

Table 1: Time and space complexities of the classification algorithms

classifier has the same form as (1), but with $g(i,\mathbf{x})$ defined as

$$g(i,\mathbf{x}) = \sum_{j=1}^{B} \alpha_{ij} k(\mathbf{x},\mathbf{z}_j),$$

where $\{\mathbf{z}_j\}_{j=1,\ldots,B}$ is the support vector set, and $\alpha_{ij}$ is a class-specific parameter associated with the $j^{\text{th}}$ support vector. We implemented Pegasos-style training, where the budget is maintained through either merging (where RBF kernel is used) or random removal of support vectors. The hyper-parameters include the number of epochs $e$, kernel parameters, regularization parameter $\lambda$, and budget size $B$. Parameters $B$ and $e$ control a speed-accuracy trade-off, while kernel parameters and $\lambda$ are best determined by cross-validation.

### 2.4 Time and Space Complexity

Time and space complexities of the algorithms are summarized in Table 1, where $N$ is the number of training examples, $C$ is the number of classes, $D$ is the data dimensionality, data sparsity $S$ is the average number of non-zero features, and $B$ is the model size for AMM, BSGD, and LLSVM. Parameter $I$ for SVM with RBF kernel (RBF-SVM) denotes a number of training iterations, empirically shown to be super-linear in $N$ (Chang and Lin, 2011).

## 3. The Software Package

BudgetedSVM can be found at `http://www.dabi.temple.edu/budgetedsvm/`. The software package provides a C++ API, comprising functions for training and testing of non-linear models described in Section 2. Each model can be easily trained and tested by calling the corresponding *train/predict* function, defined in `mm_algs.h`, `bsgd.h`, and `llsvm.h` header files. The API also provides functions for handling large-scale, high-dimensional data, defined in `budgetedSVM.h` file.

BudgetedSVM sequentially loads data chunks into memory to allow large-scale training, storing to memory only indices and values of non-zero features as a linked list. Furthermore, implementation of sparse vectors is optimized for high-dimensional data, allowing faster kernel computations and faster updates of hyperplanes and support vectors than linked list (e.g., as in LibSVM) or array implementation of vectors (e.g., as in MSVMpack by Lauer and Guermeur, 2011) used for regular-scale problems, where either time or memory costs can become prohibitively large during training in a large-scale setting. In particular, vectors are split into disjoint chunks where pointers to each chunk are stored in an array, and memory for a chunk is allocated only if one of its elements is non-zero. While significantly reducing time costs, we empirically found that this approach incurs very limited memory overhead even for data with millions of features. Consequently, BudgetedSVM vector reads and writes are performed memory-efficiently in constant time. Moreover, by storing and incrementally updating support vector $\ell_2$-norms after each training step, time to compute popular kernels (e.g., linear, Gaussian, polynomial) scales only linearly with sparsity $S$. Further implementation details can be found in a comprehensive developer's guide.

| | Pegasos | | AMM | | | LLSVM | | | BSGD | | | RBF-SVM | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data set | e.r. | t.t. | e.r. | *B* | t.t. | e.r. | *B* | t.t. | e.r. | *B* | t.t. | e.r. | t.t. |
| *webspam* | | | | | | 3.46 | 500 | 2.5m | 2.04 | 500 | 2.0m | | |
| $N = 280{,}000$ | 7.94 | 0.5s | 4.74 | 9 | 3s | 2.60 | 1,000 | 6.1m | 1.72 | 1,000 | 3.9m | 0.77 | 4.0h |
| $D = 254$ | | | | | | 1.99 | 3,000 | 0.5h | 1.49 | 3,000 | 0.2h | (#SV: 26,447) | |
| *rcv1* | | | | | | 4.97 | 500 | 0.2h | 3.33 | 500 | 0.8h | | |
| $N = 677{,}399$ | 2.73 | 1.5s | 2.39 | 19 | 9s | 4.23 | 1,000 | 0.5h | 2.92 | 1,000 | 1.5h | 2.17 | 20.2h |
| $D = 47{,}236$ | | | | | | 3.05 | 3,000 | 2.2h | 2.53 | 3,000 | 4.4h | (#SV: 50,641) | |
| *mnist8m-bin* | | | | | | 6.84 | 500 | 1.6h | 2.23 | 500 | 2.3h | | |
| $N = 8{,}000{,}000$ | 22.71 | 1.1m | 3.16 | 18 | 4m | 4.59 | 1,000 | 3.8h | 1.92 | 1,000 | 4.9h | 0.43 | N/A[1] |
| $D = 784$ | | | | | | 2.59 | 3,000 | 15h | 1.62 | 3,000 | 16.1h | | |

Table 2: Error rates (e.r.; in %) and training times[2](t.t.) on benchmark data sets

We also provide command-line and Matlab interfaces for easier use of the toolbox, which follow the user-friendly format of LibSVM and LibLinear. For example, we can type `budgetedsvm-train -A 1 a9a_train.txt a9a_model.txt` in the command prompt to train a classifier on the *adult9a* data set. The `-A 1` option specifies that we use the AMM algorithm, while the data is loaded from `a9a_train.txt` file and the trained model is stored to the `a9a_model.txt` file. Similarly, we type `budgetedsvm-predict a9a_test.txt a9a_model.txt a9a_output.txt` to evaluate the trained model, which loads the testing data from `a9a_test.txt` file, the model from `a9a_model.txt` file, and stores the predictions to `a9a_output.txt` file. We also provide a short tutorial which outlines the basic steps for using the BudgetedSVM interfaces.

### 3.1 Performance Comparison

The BudgetedSVM toolbox can learn an accurate model even for data with millions of examples and features, with training times orders of magnitude faster than RBF-SVM trained using LibSVM. For illustration, in Table 2 we give comparison of error rates and training times on binary classification tasks using several large-scale data sets (Wang et al., 2011). On *webspam* and *rcv1* it took LibSVM hours to train RBF-SVM, while BudgetedSVM algorithms with much smaller budgets achieved high accuracy within minutes, and even seconds in the case of AMM. Similarly, RBF-SVM training on large-scale *mnist8m-bin* could not be completed in a reasonable time on our test machine, while the implemented algorithms were trained within a few hours on extremely limited budgets to achieve low error rates. More detailed analysis of the BudgetedSVM algorithms can be found in their respective papers.

## 4. Conclusion and Future Work

BudgetedSVM implements four large-scale learners. Using optimized functions and data structures as a basis, through our and community efforts we plan to add more classifiers, such as Tighter Perceptron (Wang and Vucetic, 2009) and BPA (Wang and Vucetic, 2010), to make BudgetedSVM a more inclusive toolbox of budgeted SVM approximations.

## Acknowledgments

This work was supported by NSF grants IIS-0546155 and IIS-1117433.

1. Listed accuracy was obtained after 2 days of P-packSVM training on 512 processors (Zhu et al., 2009).
2. We excluded data loading time (evaluated on Intel® E7400 with 2.80GHz processor, 4GB RAM).

## References

C.-C. Chang and C.-J. 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.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *International Conference on Machine Learning*, pages 408–415, 2008.

F. Lauer and Y. Guermeur. MSVMpack: A multi-class support vector machine package. *Journal of Machine Learning Research*, 12:2293–2296, 2011.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *International Conference on Machine Learning*, pages 807–814, 2007.

Z. Wang and S. Vucetic. Tighter perceptron with improved dual use of cached data for model representation and validation. In *International Joint Conference on Neural Networks*, pages 3297–3302, 2009.

Z. Wang and S. Vucetic. Online passive-aggressive algorithms on a budget. In *International Conference on Artificial Intelligence and Statistics*, pages 908–915, 2010.

Z. Wang, N. Djuric, K. Crammer, and S. Vucetic. Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training. *Journal of Machine Learning Research*, 13:3103–3131, 2012.

K. Zhang, L. Lan, Z. Wang, and F. Moerchen. Scaling up kernel SVM on limited resources: A low-rank linearization approach. In *International Conference on Artificial Intelligence and Statistics*, 2012.

Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen. P-packSVM: Parallel primal gradient descent kernel SVM. In *IEEE International Conference on Data Mining*, 2009.