Ensembles of Budgeted Kernel Support Vector Machines for Parallel Large Scale Learning

Julien-Charles Lévesque*, **Christian Gagné**
Laboratoire de vision et systèmes numériques
Dép. de génie électrique et génie informatique
Université Laval, Québec, QC, Canada

Robert Sabourin
Lab. imagerie, vision et intelligence artificielle
Dép. de génie de la production automatisée
École tech. supérieure, Montréal, QC, Canada

Abstract

In this work, we propose to combine multiple budgeted kernel support vector machines (SVMs) trained with stochastic gradient descent (SGD) in order to exploit large databases and parallel computing resources. The variance induced by budget restrictions of the kernel SVMs is reduced through the averaging of predictions, resulting in greater generalization performance. The variance of the trainings results in a diversity of predictions, which can help explain the better performance. Finally, the proposed method is intrinsically parallel, which means that parallel computing resources can be exploited in a straightforward manner.

1 Introduction

The problem of learning from large databases is of great importance today due to the ever increasing amount of data available. When in the presence of such datasets, one must resort to simpler models since complex learning algorithms usually require prohibitive amounts of time to train. Recent advances have resulted in multiple new methods for the training of Support Vector Machines (SVMs) using large databases [1], many of which exploit Stochastic Gradient Descent (SGD) [2–4].

Most of the recent research about large scale learning has been focused on linear SVMs, largely ignoring kernel SVMs. It is often simply stated that the methods are directly transposable to kernel SVMs, which leaves some important questions unanswered. A few existing algorithms directly aim at training large scale kernel SVMs, including LASVM [5], CVM [6] and parallel SVMs [4], but they have high complexity in the number of training examples due to an unbounded model size. Some algorithms aim at sparsifying SVMs [7], but they first require a complete unbounded optimization to be run. Applying SGD to learn kernel SVMs can also lead to an indefinitely growing set of support vectors, significantly hindering training and application to large datasets. This problem has been called *the curse of kernelization* by Wang et al. [8].

Training a kernel SVM with SGD results in an algorithm similar to a kernel Perceptron [9]. In order to deal with the curse of kernelization, it is possible to impose restrictions on the number of support vectors allowed – in other words fixing a budget. The most prominent strategies for budget maintenance include the removal of the support vector with the smallest weight [10], the projection of a support vector onto the others [11], and the merging of support vectors [8].

Variance is observed between replications of the same training based on SGD, especially when budget constraints are applied. To exploit the variance inherent to these learning algorithms, we will study the combination of several budgeted SVMs trained with SGD. This will allow a straightforward parallelization of the training algorithm, in that each SVM can be trained independently of the others on a single processor. We benchmark the proposed method against other popular methods and show that training multiple budgeted SGD SVMs in parallel can increase performance reliably.

*julien-charles.levesque.1@ulaval.ca

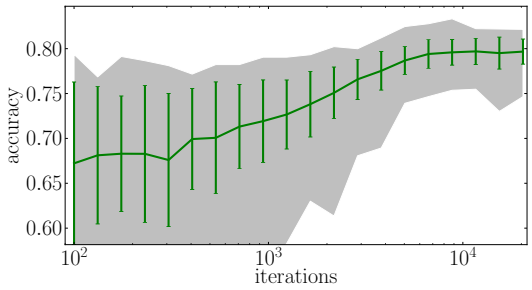

Figure 1: Testing accuracy for a single budgeted SVM trained by SGD on the Magic dataset (UCI), error bars show the standard deviation over 100 repetitions, the gray area shows the envelope of all obtained accuracies. The budget is 1000 SVs, maintained with removal of smallest SV, and others parameters were selected with a grid search.

2 Combining budgeted SVMs

We consider budgeted kernel SVMs as a base learner to construct ensembles, where each SVM has a separate budget and set of support vectors. Given an example x, we will express the prediction of SVM j as $h_j(x) = \langle \mathbf{w}_j, \phi(x) \rangle = \sum_{i \in \mathcal{S}_j} \alpha_{ji} k(x, x_{ji})$, where k is a Mercer kernel[1] and $S_j = \{(\alpha_{ji}, x_{ji})\}_{i=1}^{k}$ the set of support vectors. These SVMs are trained by SGD on the primal objective

$$P(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{t=1}^{N} l(\mathbf{w}; (x_t, y_t)), \tag{1}$$

where l is the hinge loss, $l(\mathbf{w}; (x_t, y_t)) = \max(0, 1 - y_t h(x))$. A gradient is computed at every sample and the weight vector is updated with a learning rate $\eta_t = 1/(\lambda t)$, as in [3].

In order to restrain the computational complexity of each SVM, a constraint is fixed on the size of the support vector set, so that $|S_j| \leq b$. Each time a support vector is added, the size of the support vector ensemble is checked, and when the budget is exceeded, a budget maintenance algorithm is executed to bring back the budget to the maximum size. In this work, we will reuse two existing budget maintenance strategies, which are 1) the removal of the Support Vector (SV) with the smallest α value [10], and 2) the merging of this same smallest-α-SV with another SV from the set [8].

The proposed method can be analyzed in the setting of bagging classifiers. Bagging represents the combination of several similar learners trained on subsets generated by bootstrapping from the training data (sampling with replacement) [12]. This is very similar to SGD with shuffling of the data followed by sequential access, which is sampling without replacement. The bagging method is able to reduce variance for learners with a high variance and low bias [13]. It turns out that a budgeted SVM trained with SGD is a low bias learner with a certain amount of variance, and this variance only increases when adding a budget constraint. Figure 1 illustrates this point by showing the average testing performance of one hundred budgeted SVM training repetitions on the Magic dataset.

For these reasons, combining budgeted SVMs through a simple majority vote should reduce their variance, and thus their generalization error. This performance increase comes at practically no cost since we can directly launch multiple optimizations in parallel on distributed processors without any synchronization required. The training complexity of the given model stays the same, that is linear in the number of samples, with varying costs per update depending on the budget maintenance strategy (e.g., $O(B)$ for both SV removal and merging given a budget size B). Memory and prediction complexities are multiplied by the number of members in the ensemble, so they both become $O(mB)$. The intuition for parallelization is similar to [14], although the models must be kept separately after training due to the use of a kernel with a corresponding Hilbert space of infinite dimensionality.

[1] In this work, we use the Gaussian kernel $k(x, x') = \exp\left(-\frac{\gamma}{2}\|x - x'\|^2\right)$.

3 Experiments

We evaluate our method on four two-class problems with sizes ranging from medium to large: Adult (a8a), Magic, Covtype (binary version) and Mnist8M (binary version where we compare 7s to the other numbers). Our goal with these experiments is to show a reliable performance increase for the combination of identical but independently trained SVMs. Since the training takes place in parallel, the training complexity remains the same as training a single budgeted SVM.

We evaluate four methods consisting of a single classifier: a budgeted SVM with budget maintenance by removal of the SV with smallest α trained by SGD (BSGD-R), a budgeted SVM with budget maintenance by merging of two support vectors trained by SGD (BSGD-M), a linear SVM trained by SGD (pegasos), and a kernel SVM trained with sequential minimization optimization (libsvm). We also evaluate ensembles of m budgeted SVMs with both budget maintenance strategies for various values of m. Each method is given a single pass over the dataset (apart from libsvm).

In each case, hyper-parameter selection is achieved by a cross-validation technique, using 5 folds for validation. A grid search is conducted on each parameter of each method :

- BSGD-R and BSGD-M (budget is fixed to 1000) : regularization $\lambda = [10^{-6}, 10^{-5}, ..., 10^{-2}]$, kernel width $\gamma = [10^{-2}, 10^{-1}, ..., 10^{2}]$
- Pegasos : regularization $\lambda = [10^{-6}, 10^{-5}, ..., 10^{-2}]$
- Libsvm : regularization $C = [10^{-2}, 10^{-1}, ..., 10^{2}]$, kernel width $\gamma = [10^{-2}, 10^{-1}, ..., 10^{2}]$

For each grid search, a final set of hyper-parameters was selected and the models trained during the grid search were used to evaluate the testing performance. If a pre-existing testing dataset existed it was used, otherwise a hold-out testing split was generated with 1/3 of the available data.

For Mnist8M, since it is so large, a grid search was conducted on a subset of the dataset of size 250K, then models were retrained on the full dataset with selected hyper-parameters. Furthermore, libsvm could not be trained on the complete Mnist8M dataset, so it was trained on five smaller randomly selected subsets (size=200K) and then evaluated on the same test set as the other methods.

4 Results

Table 1 and Figure 2 present the testing performance of the evaluated methods, including ensembles for $m = 7$ and 15 classifiers. We can see that the generalization performance steadily increases as we add more members to the ensembles (or computing resources to the pool). Merging consistently outperformed the removal of the smallest-α SV for both ensembles and single SVMs.

On the Mnist8M dataset, one point to note is that the performance of the first BSGD-M ensemble is already maxed out – there is a gain from one SVM to three SVMs, but from there adding more SVMs does not improve performance. This is something to be expected for easier problems.

The performance of libsvm is higher for two datasets, namely Magic and Covtype (more so for Covtype). This is somewhat expected as libsvm offers better bounds than an SGD SVM, albeit at a higher cost. The need for SGD arises when the size of datasets becomes too large for libsvm to process.

Table 1: Average testing accuracy for the studied methods. m is the number of classifiers, $m = 1$ represents a single classifier. Standard deviation shown in parentheses.

Algorithm/Data	m	Adult	Magic	Covtype	Mnist8M
libsvm	1	84.92 (0.10)	86.59 (0.07)	95.72 (0.01)	99.76 (0.01)
pegasos	1	84.74 (0.31)	78.66 (0.43)	60.72 (0.03)	97.51 (0.01)
BSGD-R	1	82.87 (0.31)	81.24 (0.20)	67.65 (0.28)	98.76 (0.19)
	7	84.22 (0.30)	83.96 (0.21)	69.48 (0.46)	98.75 (0.00)
	15	84.47 (0.24)	84.71 (0.25)	69.62 (0.25)	99.15 (0.32)
BSGD-M	1	84.49 (0.37)	84.10 (0.41)	72.48 (0.84)	99.60 (0.00)
	7	85.05 (0.17)	85.44 (0.23)	74.05 (0.16)	99.90 (0.00)
	15	85.16 (0.20)	85.71 (0.28)	74.28 (0.11)	99.90 (0.00)

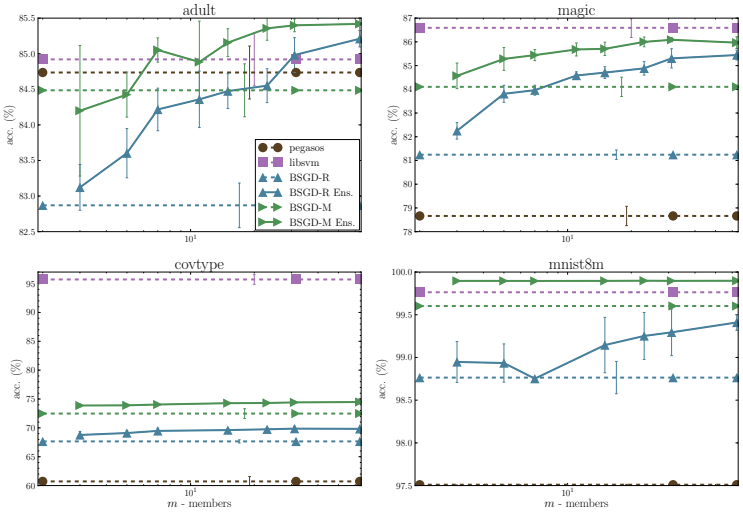

Figure 2: Testing performance for the studied methods. Ensemble methods are represented as straight lines, single classifiers as dotted lines.

Table 2: Time for the training and testing of the studied methods (train / test times, in seconds).

Dataset Size (Dim.)	Adult 22.6K / 9.9K (128)	Magic 12.6K / 6.3K (10)	Covtype 387K / 193K (54)	Mnist8M 8M / 100K (784)
libsvm	42 / 14	5 / 2	65405 / 5735	4774 / 2296*
pegasos	1 / 0	1 / 0	2 / 2	1092 / 14
BSGD-R	6 / 4	1 / 1	85 / 59	12331 / 152
BSGD-M	38 / 4	17 / 1	1287 / 50	13204 / 135

*libsvm is trained only on 200K data subsets, hence the lower training time. It could not train on larger subsets.

The constructed ensembles have a better performance than individual SGD SVMs because their combination helps to average out the stochastic noise present in the training. The members of the ensemble seem to have a good amount of diversity, in other words, predictions seem to vary between each SVM for a fixed sample. One way to validate this statement is to look at the behaviour of the Oracle model for the same ensembles. The Oracle model predicts the correct label if at least one member of the ensemble was right [15]. For instance, on the Covtype problem, the Oracle performance for $m = 15$ is very high : $96.28(\pm 0.28)$ for BSGD-R and $91.92(\pm 0.33)$ for BSGD-M. This shows that the budget constraints induce more stochasticity and variety in the models (high variance). The same trends are seen in the other datasets.

Table 2 shows the execution time for each method. The key point to note here is that the training time is the same for an ensemble as it is for a single BSGD learner, assuming we have as many processors as we have members. In that case, what we obtain is essentially a free increase in performance. The computational complexity of the proposed method is still linear in the number of samples. The method can thus be applied to very large datasets - one can even distribute each individual training with methods from the litterature (e.g., mini-batches) and then launch multiple trainings in parallel.

5 Conclusion

In this paper, we studied the combination of multiple budgeted SVMs trained by SGD. This method is indeed a good candidate for the averaging of predictions due to its high variability. It is also an inherently parallel method, thus we improve the performance over other SGD SVM methods at little cost. Interesting directions for future work include the use of less data for each SVM (e.g., splitting the dataset), and the construction of heteoregeneous ensembles by using different hyper-parameters for each SVM.

References

[1] A. Bordes, L. Bottou, and P. Gallinari. "SGD-QN: Careful quasi-newton stochastic gradient descent". In: *Journal of Machine Learning Research* 10 (2009), pp. 1737–1754.

[2] J. Langford, A. Smola, and M. Zinkevich. "Slow learners are fast". In: *Journal of Machine Learning Research* 1.2099 (2009), pp. 1–23.

[3] S. Shalev-Shwartz et al. "Pegasos: primal estimated sub-gradient solver for SVM". In: *Mathematical Programming* 127.1 (Oct. 2011), pp. 3–30.

[4] Z. A. Zhu et al. "P-packSVM: Parallel Primal grAdient desCent Kernel SVM". In: *Proceedings of the 9th IEEE International Conference on Data Mining*. 2009, pp. 677–686.

[5] A. Bordes, S. Ertekin, and J. Weston. "Fast kernel classifiers with online and active learning". In: *Journal of Machine Learning Research* 6 (2005), pp. 1579–1619.

[6] I. W. Tsang and J. T. Kwok. "Core Vector Machines : Fast SVM Training on Very Large Data Sets". In: *Journal of Machine Learning Research* 6 (2005), pp. 363–392.

[7] A. Cotter, S. Shalev-Shwartz, and N. Srebro. "Learning Optimally Sparse Support Vector Machines". In: *Proceedings of the 30th International Conference on Machine Learning*. 2013.

[8] Z. Wang, K. Crammer, and S. Vucetic. "Breaking the Curse of Kernelization : Budgeted Stochastic Gradient Descent for Large-Scale SVM Training". In: *Journal of Machine Learning Research* 13 (2012), pp. 3103–3131.

[9] R. Khardon and D. Roth. "Efficiency versus Convergence of Boolean Kernels for On-Line Learning Algorithms". In: *Advances in Neural Information Processing Systems*. 2002, pp. 423–430.

[10] O. Dekel, S. Shalev-Shwartz, and Y. Singer. "The forgetron: A kernel-based perceptron on a budget". In: *SIAM Journal on Computing* 37.5 (2008), pp. 1342–1372.

[11] F. Orabona, J. Keshet, and B. Caputo. "Bounded Kernel-Based Online Learning". In: *Journal of Machine Learning Research* 10 (2009), pp. 2643–2666.

[12] L. Breiman. "Bagging Predictors". In: *Machine Learning* 140 (1996), pp. 123–140.

[13] G. Valentini and T. G. Dietterich. "Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods". In: *Journal of Machine Learning Research* 5 (2004), pp. 725–775.

[14] M. A. Zinkevich et al. "Parallelized stochastic gradient descent". In: *Advances in Neural Information Processing Systems*. 2010.

[15] L. I. Kuncheva and C. J. Whitaker. "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy". In: *Machine Learning* 51.2 (2003), pp. 181–207.