Second Order SMO Improves SVM Online and Active Learning

Tobias Glasmachers and Christian Igel
Institut für Neuroinformatik, Ruhr-Universität Bochum
44780 Bochum, Germany

Abstract

Iterative learning algorithms that approximate the solution of support vector machines (SVMs) have two potential advantages. First, they allow for online and active learning. Second, for large datasets computing the exact SVM solution may be too time consuming and an efficient approximation can be preferable. The powerful LASVM iteratively approaches the exact SVM solution using sequential minimal optimization (SMO). It allows for efficient online and active learning. Here, this algorithm is considerably improved in speed and accuracy by replacing the working set selection in the SMO steps. A second order working set selection strategy, which greedily aims at maximizing the progress in each single step, is incorporated.

1 Introduction

The applicability of support vector machines (SVMs, Cortes and Vapnik 1995) to large scale problems is limited by the training time, which scales at least quadratically with the number of input patterns. Although progress has been made during the last decade to significantly speed up SVM learning (Joachims, 1999; Platt, 1999; Fan et al., 2005; Glasmachers and Igel, 2006), there is an increasing interest in approximation methods (Vishwanathan et al., 2003; Bordes et al., 2005; Tsang et al., 2005; Keerthi et al., 2006). Iteratively approximating the SVM solution may not only save computation time, it also allows for online and active learning.

We consider the LASVM algorithm proposed by Bordes et al. (2005), which performs *sequential minimal optimization* (SMO, Platt 1999) during learning. The algorithm can be used for online learning, where the machine is presented a continuous stream of fresh training examples and the task is to update the classifier quickly after every observed pattern. In active learning mode, this continuous stream of examples is produced by a heuristics picking informative training patterns.

Recently, progress has been made in speeding up SMO-like algorithms (Fan et al., 2005; Glasmachers and Igel, 2006). We transfer these results to online and active learning by incorporating them into the LASVM algorithm.

2 SVM Training with SMO

We consider training data $(x_i, y_i) \in X \times \{\pm 1\}$, a Mercer kernel function $k : X \times X \to \mathbb{R}$ with corresponding feature space F and feature map $\Phi : X \to F$ obeying $\langle \Phi(x), \Phi(z) \rangle = k(x, z)$ for all $x, z \in X$, and a fixed regularization parameter $C > 0$. Training a C-SVM means solving the primal optimization problem

$$\text{minimize } \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{\ell} \max\{0, 1 - y_i \cdot (\langle w, \Phi(x_i) \rangle + b)\} \ . \tag{1}$$

In practice, the corresponding dual problem is solved, which can be stated as

$$\left.\begin{array}{lll} \text{maximize} & W(\alpha) = y^T\alpha - \frac{1}{2}\alpha^T K \alpha & \\ \text{s.t.} & \sum_{i=1}^{\ell} \alpha_i = 0 & \text{(equality constraint)} \\ & m_i \le \alpha_i \le M_i \ \ \forall i \in \{1, \ldots, \ell\} & \text{(inequality constraint)} \ , \end{array}\right\} \tag{2}$$

where $K \in \mathbb{R}^{\ell \times \ell}$ denotes the kernel Gram matrix with $K_{ij} = k(x_i, x_j)$ and the vector $y \in \mathbb{R}^{\ell}$ is composed of the training data labels. The lower and upper bounds are defined as $m_i = \min\{0, y_i \cdot C\}$ and $M_i = \max\{0, y_i \cdot C\}$, respectively. An input pattern x is classified according to the sign of

$$c(x) = \langle w, \Phi(x) \rangle + b = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b = \sum_{i \in S} \alpha_i k(x_i, x) + b \ ,$$

where S contains the indices of all support vectors (i.e., all input patterns x_i with $\alpha_i \neq 0$).

Typical implementations solve the dual problem up to a fixed accuracy, for example until the Karush-Kuhn-Tucker (KKT) optimality conditions are violated by no more than, say, $\varepsilon = 0.001$. Iterative decomposition algorithms such as SMO find an (approximate) solution of problem (2) without the need to store the $\ell \times \ell$ matrix K in working memory. An iteration of a SMO-like algorithm consists of the selection of a working set $B = \{i, j\}$, the solution of the induced sub-problem changing only the dual variables α_i and α_j (see equation (4)), the update of the gradient and an optimality check. The algorithm has to compute the gradient $g(\alpha) = \nabla W(\alpha) = y - K\alpha$ of the dual objective function needed (at least) for the optimality check. All these steps require at most $O(\ell)$ operations using $O(\ell)$ working memory.

Experiments show that the optimization speed crucially depends on the working set selection strategy (Fan et al., 2005; Glasmachers and Igel, 2006). As the decomposition algorithm has to keep track of $g(\alpha)$, it is straightforward to use this information for working set selection. This leads to the most prominent selection method, *most violating pair* (MVP) working set selection (e.g., see Joachims, 1999). The index pair (i, j) fulfilling

$$i = \operatorname*{argmin}_{i | m_i < \alpha_i} g_i(\alpha) \qquad \text{and} \qquad j = \operatorname*{argmax}_{j | \alpha_j < M_j} g_i(\alpha) \tag{3}$$

is called most violating pair (MVP), as it most strongly violates the KKT conditions. The MVP can be obtained in $O(\ell)$ operations.

3 The LASVM Algorithm

The LASVM algorithm (Bordes et al., 2005) holds a set $\tilde{S}$ of support vectors on which the current classifier is based. It uses SMO to modify both the dual variables α_i, $i \in \tilde{S}$, and the set $\tilde{S}$ itself. There are two types of update steps, called `process` and `reprocess`, using different kinds of working sets B. In `process` the first element of B comes from the stream of online or active learning examples, while in `reprocess` it is selected from $\tilde{S}$. The second element is always chosen from $\tilde{S}$. If the coefficient of a new example ends up non-zero after `process` the example is added to $\tilde{S}$. After `reprocess` all examples with zero coefficients not violating the KKT conditions are removed from $\tilde{S}$.

The active learning LASVM algorithm starts with initializing $\tilde{S}$ with $\ell_{\text{init}} \ll \ell$ elements of each class using `process` repeatedly. A predefined number of epochs follows, consisting of ℓ iterations each. Every iteration is divided into three steps: An active selection step, a `process` step, and a `reprocess` step. After the last epoch an optional finishing step similar to standard SMO optimization can be conducted. The duration of this finishing step cannot be controlled in advance, as it repeats `reprocess` steps until the KKT conditions are fulfilled with a predefined accuracy ε on the support vector set $\tilde{S}$.

On many datasets, the resulting classifier performs already sufficiently well in terms of generalization error after only a single epoch of training (Bordes et al., 2005). In addition, the resulting kernel expansion after the finishing step is usually more sparse than the exact SVM solution.

Bordes et al. (2005) have shown that in the limit of arbitrary many epochs LASVM converges to the exact SVM solution. This statement holds for the online learning case and for the active learning scheme with slight modifications.

4 Second Order Working Set Selection for LASVM

The information computed in the SMO update step can be used for highly efficient working set selection (Fan et al., 2005; Glasmachers and Igel, 2006). For a pair $B = \{i, j\}$ of variable indices the progress in the dual objective achieved in a potential SMO step can be computed as follows. Let α be fixed as the current solution at the start of the SMO iteration. We parametrize the feasible line segment with a variable $\mu_{_B}$ as $\alpha + \mu_{_B} w_{_B}$ with $w_{_B} \in \mathbb{R}^\ell$ being the vector of all zeros except $(w_{_B})_i = 1$ and $(w_{_B})_j = -1$. Because the equality constraint is already incorporated into the parametrization, the SMO sub-problem (see Platt, 1999) is given by

$$\begin{array}{ll} \text{maximize} & W(\alpha + \mu_{_B} w_{_B}) \\ \text{s.t.} & \max\{m_i - \alpha_i, \alpha_j - M_j\} \le \mu_{_B} \le \min\{M_i - \alpha_i, \alpha_j - m_j\} \ . \end{array} \tag{4}$$

We rewrite

$$W(\alpha + \mu_{_B} w_{_B}) = W(\alpha) + (g_i(\alpha) - g_j(\alpha))\mu_{_B} - \frac{1}{2}(K_{ii} - 2K_{ij} + K_{jj})(\mu_{_B})^2 \ .$$

Ignoring the inequality constraint we obtain the solution

$$\mu^*_{B} = \frac{g_i(\alpha) - g_j(\alpha)}{K_{ii} - 2K_{ij} + K_{jj}}$$

resulting in the dual objective progress

$$\begin{aligned} P(B) &= W(\alpha + \mu^*_{B} w_{B}) - W(\alpha) = \frac{1}{2}(K_{ii} - 2K_{ij} + K_{jj})(\mu^*_{B})^2 \\ &= \frac{(g_i(\alpha) - g_j(\alpha))^2}{2(K_{ii} - 2K_{ij} + K_{jj})} \ . \end{aligned} \tag{5}$$

Fan et al. (2005) propose to use this quantity as a fast approximation of the true progress.

Because it would take $O(\ell^2)$ operations to check all possible index pairs, Fan et al. (2005) suggested to fix the first index i as in (3) and just to select $j = \operatorname{argmax}_{j \,|\, \alpha_j < M_j} P(\{i, j\})$ maximizing the dual objective progress. This strategy requires only linear time. In LASVM, the choice of the indices i and j is restricted to $\tilde{S}$ and the working set selection is linear in $|\tilde{S}|$.

Given that the diagonal entries K_{nn} of the kernel Gram matrix can be cached we still need the i-th matrix row of K to carry out the computation. If this row is not available from the kernel cache, expensive kernel evaluations are needed. But these computations are required for the gradient update at the end of the SMO step anyway. Thus, the second order information can be used without the need for additional kernel evaluations.

In the following, we call this algorithm second order (SO) working set selection strategy, because it makes use of kernel values defining the second order terms of the dual objective.

For incorporation of the SO working set selection in the iterative LASVM we have to cache the diagonal of the kernel Gram matrix K before the optimization loop starts. This can be done in space and time linear in ℓ. Then, in both the `process` and `reprocess` steps we replace MVP by SO when choosing the second element of the working set from $\tilde{S}$.

It turns out to be sufficient to consider the approximation of the true progress as defined in equation (5). The neglected inequality constraint could be respected by just a few additional computations (Glasmachers and Igel, 2006). However, experiments indicated that this does not improve the learning speed of LASVM when using the SO working set selection strategy.

5 Experiments and Results

In our experiments we compared the performance of LASVM using either MVP or SO working set selection.[1] In the work of Bordes et al. (2005) the original LASVM was compared to an old version of LIBSVM using MVP working set selection. Thus, we had to clarify whether LASVM is still faster after the new working set selection is incorporated

[1]Source code is available from http://www.neuroinformatik.rub.de/PEOPLE/igel/solasvm.

into both machines. Therefore, we conducted the experiments also with LIBSVM 2.8 using SO working set selection (Fan et al., 2005).

We considered four benchmark datasets, namely `Adult`, `Banana`, `USPS+N`, and `Chessboard`. The first three benchmarks were also used by Bordes et al. (2005). These are difficult datasets in the sense that it takes several learning epochs for LASVM to converge. On datasets where LASVM basically converges in a single epoch we observed no significant differences between the two working set selection strategies in LASVM. In addition, the `chessboard` dataset (Glasmachers and Igel, 2005) was considered because this simple dataset leads to dual SVM problems that are very difficult to solve by SMO.

Gaussian kernels $k(x, z) = \exp(-\gamma\|x-z\|^2)$ with parameter $\gamma > 0$ were used. The cache for the kernel matrix rows was set to the default size of 256 MB if not stated otherwise. Some datasets consist of 10 different partitionings into training and test sets. In these cases all our results refer to mean values over the partitions (Bordes et al., 2005).

We measured runtimes for different cache sizes, classification rate on a test set not used for training, and primal (1) as well as dual (2) objective value. First, we conducted 50 epochs of training to investigate the convergence behavior of the two LASVM variants. Second, we conducted one single epoch of training. This corresponds to a realistic application scenario. These results are compared to the baseline batch learning results produced by LIBSVM trained until convergence.

The learning trajectories over 50 epochs shown at the top of Figure 1 demonstrate that the new working set selection dramatically speeds up LASVM. The SO algorithm leads to much faster learning than MVP. The longer LASVM needs to converge the more pronounced are the differences. The performance indicators listed in the table in Figure 1 clearly show that the solution quality considerably increased at the same time.

It seems that the selection of better working sets leads to a faster convergence of the set $\tilde{S}$. Wrongly adding a training example to $\tilde{S}$ can decrease the overall speed as subsequent working set selection scales linear in $|\tilde{S}|$. Even worse, this may lead to other wrong choices in future decisions. Therefore a well founded working set selection can play a key role in speeding up the entire algorithm. This is in accordance with the number of support vectors used by both methods, see the table in Figure 1. After a single learning epoch without finishing, the SO solutions are extremely sparse compared to machines produced using MVP. Besides improved training speed and accuracy we consider this feature as a decisive difference between MVP and SO working set selection. In three of four cases the solutions of LASVM with SO working set selection were even more sparse than the exact SVM solutions, whereas LASVM with MVP can lead to machines with almost four times more support vectors compared to LIBSVM when considering a single learning epoch.

Thus, the main result of this paper is non-ambiguous: The SO method is better than the original LASVM in all measured performance criteria. The comparison of LASVM with LIBSVM 2.8, both using SO working set selection, is more difficult. Our selection of difficult datasets as discussed above clearly biases this comparison as we can not expect LASVM to reach the performance of LIBSVM on these problems. On simple datasets where LASVM needs only a few epochs to reach the same objective value as LIBSVM the LASVM algorithm has the same classification error (Bordes et al., 2005). When comparing

the classification rates in the table, we have to keep in mind that LASVM was only trained for one epoch and had usually not yet converged, whereas LIBSVM was trained until convergence. Of course, the LASVM solution will approach the exact SVM solution found by LIBSVM (i.e., having approximately the same classification rate and number of support vectors) in subsequent epochs. Still, even for some of our difficult datasets we only see small differences in classification rate between LIBSVM and LASVM with SO working set selection after a single epoch, whereas LASVM with MVP performs considerably worse.

On the `Adult` dataset the LIBSVM algorithm outperforms LASVM in terms of training time although it computes a more accurate solution. This result indicates that LIBSVM profits more than LASVM from the SO working set selection in this example. This is possible because the number of LASVM iterations is fixed by the epoch length while LIBSVM may stop after less iterations. On the other three datasets LASVM is clearly faster than LIBSVM, especially for small cache sizes, see bottom of Figure 1. The small cache behavior is an indicator of how an algorithm performs on large scale datasets. By design it is one of LASVM's advantages to work well with comparatively small cache sizes. Therefore it is not surprising that LASVM is much more robust to a cache size reduction than LIBSVM. This is a general property of LASVM independent of the working set selection algorithm.

6 Conclusion

The powerful LASVM online and active learning algorithm (Bordes et al., 2005) can be combined in a natural and elegant way with second order working set selection. Theoretical considerations support the expectation that the combined algorithm shows superior learning speed. The experiments show that we indeed achieve a considerable speed up. At the same time we gain increased accuracy and sparseness. For datasets that are difficult to learn this effect is very pronounced. We conclude that second order working set selection should become the default in iterative SVM learning algorithms relying on sequential minimal optimization.

References

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 5:1579–1619, 2005. http://leon.bottou.com/projects/lasvm.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Glasmachers and C. Igel. Gradient-Based Adaptation of General Gaussian Kernels. *Neural Computation*, 17:2099–2105, 2005.

T. Glasmachers and C. Igel. Maximum-gain working set selection for support vector machines. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1999.

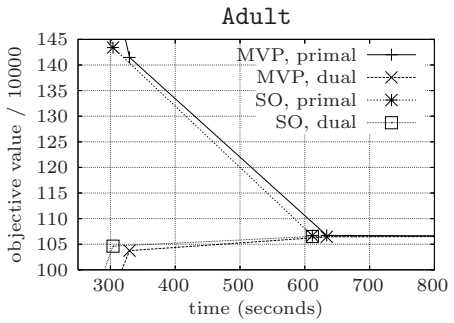
S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 8:1–22, 2006.

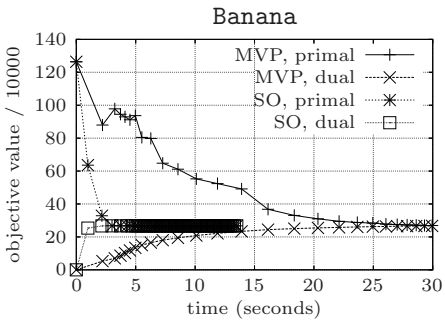
J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

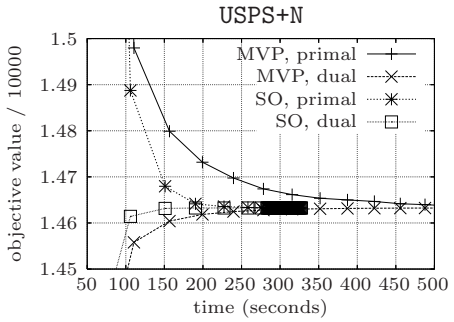
I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

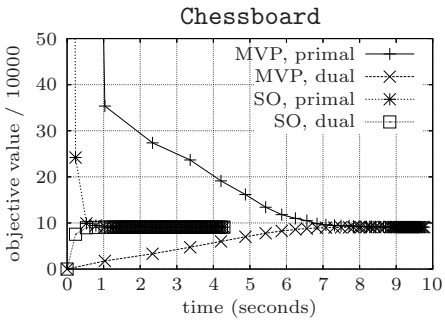
S. V. N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 760–767. AAAI Press, 2003.

Fig. 1: Performance of LASVM with most violating pair (MVP) and second order (SO) working set selection. LIBSVM 2.8 results are reported for comparison. The plots at the top illustrate the evolution of primal and dual objective over 50 epochs of training on the benchmark problems. The upper (decreasing) and lower (increasing) curves represent the primal and dual objective, respectively, converging to the same optimal value. The table lists the classification rate, primal and dual objective value as well as the number of support vectors. The curves beneath show the dependency of the runtime on the cache size for the first learning epoch.









dataset	parameters	algorithm	#SV	cl. rate	primal	dual
`Adult`		MVP	14033	74.84%	1,187,350	1,036,160
$\ell = 32,562$	$C = 100$	SO	11167	82.10%	1,189,174	1,046,430
1 partition	$\gamma = 0.005$	LIBSVM	11345	85.13%	1,065,429	1,065,408
`Banana`		MVP	484	64.79%	915,624	52,984
$\ell = 4,000$	$C = 316$	SO	145	80.01%	375,718	253,562
10 partitions	$\gamma = 0.5$	LIBSVM	162	90.04%	267,278	265,545
`USPS+N`		MVP	3870	99.09%	17,214	14,207
$\ell = 7,329$	$C = 10$	SO	2501	99.36%	15,867	14,454
10 partitions	$\gamma = 2$	LIBSVM	2747	99.51%	14,635	14,633
`Chessboard`		MVP	3395	97.86%	782,969	60,771
(5000)	$C = 1000$	SO	944	98.91%	143,614	83,686
1 partition	$\gamma = 2$	LIBSVM	873	99.40%	90,959	90,946