Nonconvex Online Support Vector Machines

Şeyda Ertekin, Léon Bottou, and C. Lee Giles, *Fellow*, *IEEE*

Abstract—In this paper, we propose a nonconvex online Support Vector Machine (SVM) algorithm (LASVM-NC) based on the *Ramp Loss*, which has the strong ability of suppressing the influence of outliers. Then, again in the online learning setting, we propose an outlier filtering mechanism (LASVM-I) based on approximating nonconvex behavior in convex optimization. These two algorithms are built upon another novel SVM algorithm (LASVM-G) that is capable of generating accurate intermediate models in its iterative steps by leveraging the duality gap. We present experimental results that demonstrate the merit of our frameworks in achieving significant robustness to outliers in noisy data classification where mislabeled training instances are in abundance. Experimental evaluation shows that the proposed approaches yield a more scalable online SVM algorithm with sparser models and less computational running time, both in the training and recognition phases, without sacrificing generalization performance. We also point out the relation between nonconvex optimization and min-margin active learning.

Index Terms—Online learning, nonconvex optimization, support vector machines, active learning.

1 Introduction

In supervised learning systems, the generalization performance of classification algorithms is known to be greatly improved with large margin training. Large margin classifiers find the maximal margin hyperplane that separates the training data in the appropriately chosen kernel-induced feature space. It is well established that if a large margin is obtained, the separating hyperplane is likely to have a small misclassification rate during recognition (or prediction) [1], [2], [3]. However, requiring that all instances be correctly classified with the specified margin often leads to overfitting, especially when the data set is noisy. Support Vector Machines [4] address this problem by using a *soft margin* criterion, which allows some examples to appear on the wrong side of the hyperplane (i.e., misclassified examples) in the training phase to achieve higher generalization accuracy. With the soft margin criterion, patterns are allowed to be misclassified for a certain cost, and consequently, the outliers—the instances that are misclassified outside of the margin—start to play a dominant role in determining the decision hyperplane,since they tend to have the largest margin loss according to the Hinge Loss. Nonetheless, due to its convex property and practicality, Hinge Loss has become a commonly used loss function in SVMs.

Convexity is viewed as a virtue in the machine learning literature both from a theoretical and experimental point of view. Convex methods can be easily analyzed mathematically and bounds can be produced. Additionally, convex solutions are guaranteed to reach global optima and are not sensitive to initial conditions. The popularity of convexity further increased after the success of convex algorithms, particularly with SVMs, which yield good generalization performance and have strong theoretical foundations. However, in convex SVM solvers, all misclassified examples become support vectors, which may limit the scalability of the algorithm to learning from large-scale data sets. In this paper, we show that nonconvexity can be very effective for achieving sparse and scalable solutions, particularly when the data consist of abundant label noise. We present herein experimental results that show how a nonconvex loss function, *Ramp Loss*, can be integrated into an online SVM algorithm in order to suppress the influence of misclassified examples.

Various works in the history of machine learning research focused on using nonconvex loss functions as an alternate to convex Hinge Loss in large margin classifiers. While Mason et al. [5] and Krause and Singer [6] applied it to Boosting, Perez-Cruz et al. [7] and Xu and Cramer [8] proposed training algorithms for SVMs with the Ramp Loss and solved the nonconvex optimization by utilizing semidefinite programming and convex relaxation techniques. On the other hand, some previous works of Liu et al. [9] and Wang et al. [10] used the Concave-Convex Procedure (CCCP) [11] for nonconvex optimization as the work presented here. Those studies are worthwhile in the endeavor of achieving sparse models or competitive generalization performance; nevertheless, none of them are efficient in terms of computational running time and scalability for real-world data mining applications, and yet the improvement in classification accuracy is only marginal. Collobert et al. [12] pointed out the scalability advantages of nonconvex approaches and used CCCP for nonconvex optimization in order to achieve faster batch SVMs and Transductive SVMs. In this paper, we focus on bringing the scalability advantages of nonconvexity to the online learning setting by using an online SVM algorithm, LASVM [13]. We also highlight and discuss the connection between the nonconvex loss and traditional min-margin active learners.

- *Ş. Ertekin is with the Massachusetts Institute of Technology, Sloan School of Management, Cambridge, MA 02139. E-mail: seyda@mit.edu.*
- *L. Bottou is with the NEC Laboratories America, 4 Independence Way, Suite 200, Princeton, NJ 08540. E-mail: leon@bottou.org.*
- *C.L. Giles is with the College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802. E-mail: giles@ist.psu.edu.*

Manuscript received 15 Mar. 2009; revised 11 Dec. 2009; accepted 30 Mar. 2010; published online 24 May 2010.
Recommended for acceptance by O. Chapelle.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-03-0168.
Digital Object Identifier no. 10.1109/TPAMI.2010.109.

0162-8828/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

Online learning offers significant computational advantages over batch learning algorithms and the benefits of online learning become more evident when dealing with streaming or very large-scale data. Online learners incorporate the information of recently observed training data into the model via incremental model updates and without the need for retraining it with previously seen entire training data. Since these learners process the data one at a time in the training phase, selective sampling can be applied and evaluation of the informativeness of the data prior to the processing by the learner becomes possible. The computational benefits of avoiding periodic batch optimizations, however, necessitate the online learner fulfilling two critical requirements, namely, the intermediate models need to be well enough trained in order to capture the characteristics of the training data, but, on the other hand, should not be overoptimized since only part of the entire training data is seen at that point in time. In this paper, we present an online SVM algorithm, LASVM-G, that maintains a balance between these conditions by leveraging the *duality gap* between the primal and dual functions throughout the online optimization steps. Based on the online training scheme of LASVM-G, we then present LASVM-NC, an online SVM algorithm with nonconvex loss function, which yields a significant speed improvement in training and builds a sparser model, hence resulting in faster recognition than its convex version as well. Finally, we propose an SVM algorithm (LASVM-I) that utilizes the selective sampling heuristic by ignoring the instances that lie in the flat region of the Ramp Loss in advance, before they are processed by the learner. Although this approach may appear like an overaggressive training sample elimination process, we point out that these instances do not play a large role in determining the decision hyperplane according to the Ramp Loss anyway. We show that for a particular case of sample elimination scenario, misclassified instances according to the most recent model are not taken into account in the training process. For another case, only the instances in the margin pass the barrier of elimination and are processed in the training, hence leading to an extreme case of *small pool active learning* framework [14] in online SVMs. The proposed nonconvex implementation and selective sample ignoring policy yields sparser models with fewer support vectors and faster training with less computational time and kernel computations, which overall leads to a more scalable online SVM algorithm. The benefits of the proposed methods are fully realized for kernel SVMs and their advantages become more pronounced in noisy data classification, where mislabeled samples are in abundance.

In the next section, we present a background on Support Vector Machines. Section 3 gives a brief overview of the online SVM solver, LASVM [13]. We then present the proposed online SVM algorithms, LASVM-G, LASVM-NC, and LASVM-I. The paper continues with the experimental analysis presented in Section 8, followed by concluding remarks.

2 Support Vector Machines

Support Vector Machines [4] are well known for their strong theoretical foundations, generalization performance, and ability to handle high-dimensional data. In the binary classification setting, let $((x_1, y_1) \cdots (x_n, y_n))$ be the training data set where x_i are the feature vectors representing the instances and $y_i \in \{-1, +1\}$ are the labels of those instances. Using the training set, SVM builds an optimum hyperplane—a linear discriminant in a higher dimensional feature space—that separates the two classes by the largest margin. The SVM solution is obtained by minimizing the following *primal* objective function:

$$\min_{\boldsymbol{w},b} J(w,b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \tag{1}$$

$$\text{with} \quad \forall i \begin{cases} y_i(\boldsymbol{w}\cdot\Phi(x_i)+b) \geq 1-\xi_i, \\ \xi_i \geq 0, \end{cases}$$

where w is the normal vector of the hyperplane, b is the offset, y_i are the labels, $\Phi(\cdot)$ is the mapping from input space to feature space, and ξ_i are the slack variables that permit the nonseparable case by allowing misclassification of training instances.

In practice, the convex quadratic programming (QP) problem in (1) is solved by optimizing the dual cost function:

$$\max_{\boldsymbol{\alpha}} G(\boldsymbol{\alpha}) \equiv \sum_{i=1}^{N}\alpha_i y_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(x_i,x_j), \tag{2}$$

$$\text{subject to} \quad \begin{cases} \sum_i \alpha_i = 0, \\ A_i \leq \alpha_i \leq B_i, \\ A_i = \min(0, Cy_i), \\ B_i = \max(0, Cy_i), \end{cases} \tag{3}$$

where $K(x_i,x_j) = \langle\Phi(x_i),\Phi(x_j)\rangle$ is the kernel matrix representing the dot products $\Phi(x_i)\cdot\Phi(x_j)$ in feature space. We adopt a slight deviation of the coefficients α_i from the standard representation and let them inherit the signs of the labels y_i, permitting the α_i to take on negative values. After solving the QP problem, the norm of the hyperplane $\boldsymbol{w}$ can be represented as a linear combination of the vectors in the training set

$$\boldsymbol{w} = \sum_i \alpha_i \Phi(x_i). \tag{4}$$

Once a model is trained, a soft margin SVM classifies a pattern x according to the sign of a decision function, which can be represented as a *kernel expansion*

$$\hat{y}(x) = \sum_{i=1}^{n}\alpha_i K(x,x_i) + b, \tag{5}$$

where the sign of $\hat{y}(x)$ represents the predicted classification of x.

A widely popular methodology for solving the SVM QP problem is Sequential Minimal Optimization (SMO) [15]. SMO works by making successive direction searches, which involves finding a pair of instances that violate the KKT conditions and taking an optimization step along that feasible direction. The $\boldsymbol{\alpha}$ coefficients of these instances are modified by opposite amounts, so SMO makes sure that the constraint $\sum_i \alpha_i = 0$ is not violated. Practical implementations of SMO select working sets based on finding a pair of instances that violate the KKT conditions more than τ-precision, also known as *τ-violating pairs* [13]:

$$(i,j) \text{ is a } \tau\text{-violating pair} \Longleftrightarrow \begin{cases} \alpha_i < B_i, \\ \alpha_j > A_j, \\ g_i - g_j > \tau, \end{cases}$$

where g denotes the gradient of an instance and τ is a small positive threshold. The algorithm terminates when all KKT violations are below the desired precision.

The effect of the bias term. Note that the equality constraint on the sum of α_i in (3) appears in the SVM formulation only when we allow the offset (bias) term b to be nonzero. While there is a single "optimal" b, different SVM implementations may choose separate ways of adjusting the offset. For instance, it is sometimes beneficial to change b in order to adjust the number of false positives and false negatives [2, page 203], or even disallow the bias term completely (i.e., $b = 0$) [16] for computational simplicity. In SVM implementations that disallow the offset term, the constraint $\sum_i \alpha_i = 0$ is removed from the SVM problem. The online algorithms proposed in this paper also adopt the strategy of setting $b = 0$. This strategy gives the algorithms the flexibility to update a single α_i at a time at each optimization step, bringing computational simplicity and efficiency to the solution of the SVM problem without adversely affecting the classification accuracy.

3 LASVM

LASVM [13] is an efficient online SVM solver that uses less memory resources and trains significantly faster than other state-of-the-art SVM solvers while yielding competitive misclassification rates after a single pass over the training examples. LASVM realizes these benefits due to its novel optimization steps that have been inspired by SMO. LASVM applies the same pairwise optimization principle to online learning by defining two direction search operations. The first operation, PROCESS, attempts to insert a new example into the set of current support vectors (SVs) by searching for an existing SV that forms a τ-violating pair with maximal gradient. Once such an SV is found, LASVM performs a direction search that can potentially change the coefficient of the new example and make it a support vector. The second operation, REPROCESS, attempts to reduce the current number of SVs by finding two SVs that are τ-violating SVs with maximal gradient. A direction search can zero the coefficient of one or both SVs, removing them from the set of current support vectors of the model. In short, PROCESS adds new instances to the working set and REPROCESS removes the ones that the learner does not benefit from anymore. In the online iterations, LASVM alternates between running single PROCESS and REPROCESS operations. Finally, LASVM simplifies the kernel expansion by running REPROCESS to remove all τ-violating pairs from the kernel expansion, a step known as FINISHING. The optimizations performed in the FINISHING step reduce the number of support vectors in the SVM model.

4 LASVM with Gap-Based Optimization—LASVM-G

In this section, we present LASVM-G—an efficient online SVM algorithm that brings performance enhancements to LASVM. Instead of running a single REPROCESS operation after each PROCESS step, LASVM-G adjusts the number of REPROCESS operations at each online iteration by leveraging the gap between the primal and the dual functions. Further, LASVM-G replaces LASVM's one time FINISHING optimization and cleaning stage with the optimizations performed in each REPROCESS cycle at each iteration and the periodic non-SV removal steps. These improvements enable LASVM-G to generate more reliable intermediate models than LASVM, which lead to sparser SVM solutions that can potentially have better generalization performance. For further computational efficiency, the algorithms that we present in the rest of the paper use the SVM formulation with $b = 0$. As we pointed out in Section 2, the bias term b acts as a hyperparameter that can be used to adjust the number of false positives and false negatives for varying settings of b, or to achieve algorithmic efficiency due to computational simplicity when $b = 0$. In the rest of the paper, all formulations are based on setting the bias $b = 0$ and thus optimizing a single α at a time.

4.1 Leveraging the Duality Gap

One question regarding the optimization scheme in the original LASVM formulation is the rate at which to perform REPROCESS operations. A straightforward approach would be to perform one REPROCESS operation after each PROCESS step, which is the default behavior of LASVM. However, this heuristic approach may result in underoptimization of the objective function in the intermediate steps if this rate is smaller than the optimal proportion. Another option would be to run REPROCESS until a small predefined threshold ε exceeds the L_∞ norm of the projection of the gradient $(\partial G(\alpha)/\partial \alpha_i)$, but little work has been done to determine the correct value of the threshold ε. A geometrical argument relates this norm to the position of the support vectors relative to the margins [17]. As a consequence, one usually chooses a relatively small threshold, typically in the range 10^{-4}-10^{-2}. Using such a small threshold to determine the rate of REPROCESS operations results in many REPROCESS steps after each PROCESS operation. This will not only increase the training time and computational complexity, but can potentially overoptimize the objective function at each iteration. Since nonconvex iterations work toward suppressing some training instances (outliers), the intermediate learned models should be well enough trained in order to capture the characteristics of the training data, but, on the other hand, should not be overoptimized since only part of the entire training data is seen at that point in time. Therefore, it is necessary to employ a criteria to determine an accurate rate of REPROCESS operations after each PROCESS. We define this policy as *the minimization of the gap between the primal and the dual* [2].

Optimization of the duality gap. From the formulations of the primal and dual functions in (1) and (2), respectively, it can be shown that the optimal values of the primal and dual are same [18]. Furthermore, at any nonoptimal point, the primal function is *guaranteed* to lie above the dual curve. In formal terms, let $\hat{\theta}$ and $\hat{\alpha}$ be solutions of problems (1) and (2), respectively. The strong duality asserts that for any feasible θ and α,

$$G(\alpha) \leq G(\hat{\alpha}) = J(\hat{\theta}) \leq J(\theta) \; with \; \hat{\theta} = \sum_i \hat{\alpha}_i \Phi(x_i). \qquad (6)$$

That is, at any time during the optimization, the value of the primal $J(\cdot)$ is higher than the dual $G(\cdot)$. Using the equality $w = \sum_l \alpha_l \Phi(x_l)$ and $b = 0$, we show that this holds as follows:

$$\begin{aligned} J(\theta) - G(\alpha) &= \frac{1}{2}\|w\|^2 + C\sum_l |1 - y_l(w \cdot \Phi(x_l))|_+ \\ &\quad - \sum_l \alpha_l y_l + \frac{1}{2}\|w\|^2 \\ &= \|w\|^2 - \sum_l \alpha_l y_l + C\sum_l |1 - y_l(w \cdot \Phi(x_l))|_+ \\ &= w\sum_l \alpha_l \Phi(x_l) - \sum_l \alpha_l y_l \\ &\quad + C\sum_l |1 - y_l(w \cdot \Phi(x_l))|_+ \\ &= -\sum_l y_l \alpha_l |1 - y_l(w \cdot \Phi(x_l) + b)|_+ \\ &\quad + C\sum_l |1 - y_l(w \cdot \Phi(x_l) + b)|_+ \\ &= \sum_l \underbrace{(C - \alpha_l y_l)}_{\geq 0} \underbrace{|1 - y_l(w \cdot \Phi(x_l))|_+}_{\geq 0} \\ &\geq 0, \end{aligned}$$

where $C - \alpha_l y_l \geq 0$ is satisfied by the constraint of the dual function in (3). Then, the SVM solution is obtained when one reaches $\bar{\theta}, \bar{\alpha}$ such that

$$\varepsilon > J(\bar{\theta}) - G(\bar{\alpha}) \qquad where \qquad \bar{\theta} = \sum_i \bar{\alpha}_i \Phi(x_i). \tag{7}$$

The strong duality in (6) then guarantees that $J(\bar{\theta}) < J(\hat{\theta}) + \varepsilon$. Few solvers implement this criterion since it requires the additional calculation of the gap $J(\theta) - G(\alpha)$. In this paper, we advocate using criterion (7) using a threshold value ε that grows sublinearly with the number of examples. Letting ε grow makes the optimization coarser when the number of examples increases. As a consequence, the asymptotic complexity of optimizations in online setting can be smaller than that of the exact optimization.

Most SVM solvers use the dual formulation of the QP problem. However, increasing the dual does not necessarily reduce the duality gap. The dual function follows a nice monotonically increasing pattern at each optimization step, whereas the primal shows significant up and down fluctuations. In order to keep the size of the duality gap in check, before each PROCESS operation, we compute the standard deviation of the primal, which we call the *Gap Target* $\hat{\mathbb{G}}$:

$$\hat{\mathbb{G}} = \sqrt{\sum_{i=1}^{n} h_i^2 - \frac{\left(\sum_{i=1}^{n} h_i\right)^2}{l}}, \tag{8}$$

where l is the number of support vectors and $h_i = C y_i g_i$ with C and g_i denoting the misclassification penalty and the gradient of instance i, respectively. After computing the gap target, we run a PROCESS step and check the new Gap $\mathcal{G}$ between the primal and the dual. After an easy derivation, the gap is computed as

$$\mathcal{G} = -\sum_{i=1}^{n} (\alpha_i g_i + \max(0, C g_i)). \tag{9}$$

Note that, as we have indicated earlier, the bias term b is set to zero in all of the formulations. In the online iterations, we cycle between running REPROCESS and computing the gap $\mathcal{G}$ until the termination criteria $\mathcal{G} \leq \max(C, \hat{\mathbb{G}})$ is reached. That is, we require the duality gap after the REPROCESS operations to be not greater than the initial gap target $\hat{\mathbb{G}}$. The C parameter is part of the equation in order to prevent the algorithm from specifying a too narrow gap target and therefore prevent making excessive number of optimization steps. The heuristic upper bound on the gap is developed based on the oscillating characteristics of the primal function during the optimization steps and these oscillations are related to the successive choice of examples to REPROCESS. Viewing these oscillations as noise, the gap target enables us to stop the REPROCESS operations when the difference is within the noise. After this point, the learner continues with computing the new Gap Target and running PROCESS and REPROCESS operation on the next fresh instance from the unseen example pool.

4.2 Building Blocks

The implementation of LASVM-G maintains the following pieces of information as its key building blocks: the coefficients α_i of the current kernel expansion $\mathcal{S}$, the bounds for each α, and the partial derivatives of the instances in the expansion, given as

$$g_k = \frac{\partial W(\alpha)}{\partial \alpha_k} = y_k - \sum_i \alpha_i K(x_i, x_k) = y_k - \hat{y}(x_k). \tag{10}$$

The kernel expansion here maintains all of the training instances in the learner's active set, both the support vectors and the instances with $\alpha = 0$.

In the online iterations of LASVM-G, the optimization is driven by two kinds of direction searches. The first operation, PROCESS, inserts an instance into the kernel expansion and initializes the α_i and gradient g_i of this instance (Step 1). After computing the step size (Step 2), it performs a direction search (Step 3). We set the offset term for kernel expansion b to zero for computational simplicity. As discussed in the SVM section regarding the offset term, disallowing b removes the necessity of satisfying the constraint $\sum_{i \in \mathcal{S}} \alpha_i = 0$, enabling the algorithm to update a single α at a time, both in PROCESS and REPROCESS operations.

LASVM-G PROCESS(i)

1) $\alpha_i \leftarrow 0, \qquad g_i \leftarrow y_k - \sum_{s \in S} \alpha_s K_{is}$
2) **If** $g_i < 0$ then
   $\lambda = \max\{A_i - \alpha_i, \frac{g_i}{K_{ii}}\}$
   Else
   $\lambda = \max\{B_i - \alpha_i, \frac{g_i}{K_{ii}}\}$
3) $\alpha_i \leftarrow \alpha_i + \lambda$
   $g_s \leftarrow g_s - \lambda K_{is} \quad \forall s$ *in kernel expansion*

The second operation, REPROCESS, searches all of the instances in the kernel expansion and selects the instance with the maximal gradient (Steps 1-3). Once an instance is selected, LASVM-G computes a step size (Step 4) and performs a direction search (Step 5).

LASVM-G REPROCESS()

1) $i \leftarrow \arg\min_{s \in S} g_s$ with $\alpha_s > A_s$
   $j \leftarrow \arg\max_{s \in S} g_s$ with $\alpha_s < B_s$

2) Bail out if (i, j) is not a τ-violating pair.
3) **If** $g_i + g_j < 0$ **then** $g \leftarrow g_i, t \leftarrow i$
   Else $g \leftarrow g_j, t \leftarrow j$
4) **If** $g < 0$ **then**
   $\lambda = \max\{A_t - \alpha_t, \frac{g}{K_{tt}}\}$
   Else
   $\lambda = \min\{B_t - \alpha_t, \frac{g}{K_{tt}}\}$
5) $\alpha_k \leftarrow \alpha_k + \lambda$
   $g_s \leftarrow g_s - \lambda K_{ts} \quad \forall s$ *in kernel expansion*

Both PROCESS and REPROCESS operate on the instances in the kernel expansion, but neither of them remove any instances from it. A removal step is necessary for improved efficiency because, as the learner evolves, the instances that were admitted to the kernel expansion in earlier iterations as support vectors may not serve as support vectors anymore. Keeping such instances in the kernel expansion slows down the optimization steps without serving much benefit to the learner and increases the application's requirement for computational resources. A straightforward approach to address this inefficiency would be to remove all of the instances with $\alpha_i = 0$, namely, all nonsupport vectors in the kernel expansion. One concern with this approach is that once an instance is removed, it will not be seen by the learner again, and thus it will no longer be eligible to become a support vector in the later stages of training. It is important to find a balance between maintaining the efficiency of a small-sized kernel expansion and not aggressively removing instances from the kernel expansion. Therefore, the cleaning policy needs to preserve the instances that can potentially become SVs at a later stage of training while removing instances that have the lowest possibility of becoming SVs in the future.

Our cleaning procedure periodically checks the number of non-SVs in the kernel expansion. If the number of non-SVs n is more than the number of instances that is permitted in the expansion m by the algorithm, CLEAN selects the extra non-SV instances with the highest gradients for removal. It follows from (10) that this heuristic predominantly selects the most misclassified instances that are farther away from the hyperplane for deletion from the kernel expansion. This policy suppresses the influence of the outliers on the model to yield sparse and scalable SVM solutions.

CLEAN
n: *number of non-SVs in the kernel expansion.*
m: *maximum number of allowed non-SVs.*
$\vec{v}$: *Array of partial derivatives.*
1) **If** $n < m$ **return**
2) $\vec{v} \leftarrow \vec{v} \cup |g_i|_+, \forall i$ with $\alpha_i = 0$
3) Sort the gradients in $\vec{v}$ in ascending order.
   $g_{threshold} \leftarrow v[m]$
4) **If** $|g_i|_+ \geq g_{threshold}$ **then** remove $x_i, \forall i$ with $\alpha_i = 0$

We want to point out that it is immaterial to distinguish whether an instance has not been an SV for many iterations or it has just become a non-SV. In either case, these examples do not currently contribute to the classifier and are treated equally from a cleaning point of view.

Note that these algorithmic components are geared toward designing and online SVM solver with nonlinear kernels. Even though it is possible to apply these principles to linear SVMs as well, the algorithmic aspects for linear SVMs will be different. For instance, for linear SVMs, it is possible to store the normal vector w of the hyperplane directly instead of manipulating the support vector expansions. In this regard, the focus of the paper is more about kernel SVMs.

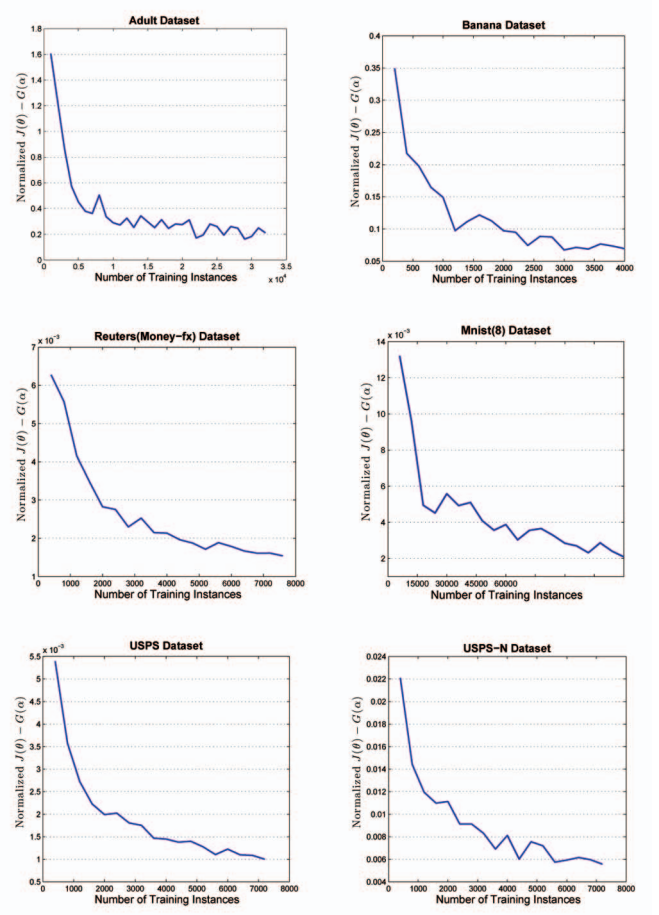

Fig. 1. The duality gap $(J(\theta) - G(\alpha))$, normalized by the number of training instances.

4.3 Online Iterations in LASVM-G

LASVM-G exhibits the same learning principle as LASVM, but in a more systematic way. Both algorithms make one pass (epoch) over the training set. Empirical evidence suggests that a single epoch over the entire training set yields a classifier as good as the SVM solution.

Upon initialization, LASVM-G alternates between PROCESS and REPROCESS steps during the epoch like LASVM, but distributes LASVM's one time FINISHING step to the optimizations performed in each REPROCESS cycle at each iteration and the periodic CLEAN operations. Another important property of LASVM-G is that it leverages the gap between the primal and the dual functions to determine the number of REPROCESS steps after each PROCESS (the -G suffix emphasizes this distinction). Reducing the duality gap too fast can cause overoptimization in early stages without yet observing sufficient training data. Conversely, reducing the gap too slowly can result in underoptimization in the intermediate iterations. Fig. 1 shows that as the learner sees more training examples, the duality gap gets smaller.

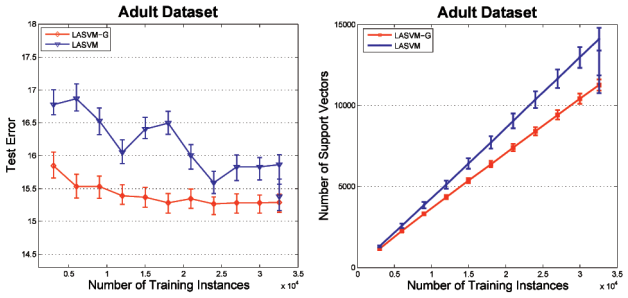

Fig. 2. LASVM versus LASVM-G for the Adult data set. (a) Test error convergence. (b) Growth of number of SVs.

The major enhancements that are introduced to LASVM enable LASVM-G to achieve higher prediction accuracies than LASVM in the intermediate stages of training. Fig. 2 presents a comparative analysis of LASVM-G versus LASVM for the Adult data set (Table 2). Results on other data sets are provided in Section 8.

While both algorithms report the same generalization performance in the end of training, LASVM-G reaches a better classification accuracy at an earlier point in training than LASVM and is able to maintain its performance relatively stable with a more reliable model over the course of training. Furthermore, LASVM-G maintains fewer number of support vectors in the intermediate training steps, as evidenced in Fig. 2b.

LASVM-G

1) **Initialization**:
   Set $\alpha \leftarrow 0$
2) **Online Iterations**:
   Pick an example x_i
   Compute Gap Target $\hat{\mathbb{G}}$
   $Threshold \leftarrow \max(C, \hat{\mathbb{G}})$
   Run PROCESS(x_i)
   while Gap $\mathcal{G} > Threshold$
   Run REPROCESS
   end
   Periodically run CLEAN

In the next sections, we further introduce three SVM algorithms that are implemented based on LASVM-G, namely, LASVM-NC, LASVM-I, and FULL SVM. While these SVM algorithms share the main building blocks of LASVM-G, each algorithm exhibits a distinct learning principle. LASVM-NC uses the LASVM-G methodology in a nonconvex learner setting. LASVM-I is a learning scheme that we propose as a convex variant of LASVM-NC that employs selective sampling. FULL SVM does not take advantage of the nonconvexity or the efficiency of the CLEAN operation, and acts as a traditional online SVM solver in our experimental evaluation.

5 Nonconvex Online SVM Solver—LASVM-NC

In this section, we present LASVM-NC, a nonconvex online SVM solver that achieves sparser SVM solutions in less time than online convex SVMs and batch SVM solvers. We first introduce the nonconvex Ramp Loss function and discuss how nonconvexity can overcome the scalability problems of convex SVM solvers. We then present the methodology to optimize the nonconvex objective function, followed by the description of the online iterations of LASVM-NC.

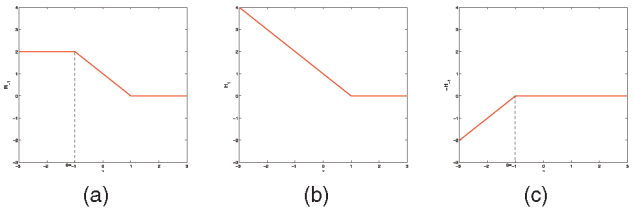

Fig. 3. (a) The ramp loss can be decomposed into (b) a convex hinge loss and (c) a concave loss.

5.1 Ramp Loss

Traditional convex SVM solvers rely on the Hinge Loss H_1 (as shown in Fig. 3b) to solve the QP problem, which can be represented in Primal form as

$$\min_{\boldsymbol{w},b} J(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} H_1(y_i f(x_i)). \tag{11}$$

In the Hinge Loss formulation $H_s(z) = \max(0, s - z)$, s indicates the Hinge point and the elbow at $s = 1$ indicates the point at which $y_l f_\theta(x_l) = y_l(w \cdot \Phi(x_l) + b) = 1$. Assume, for simplicity, that the Hinge Loss is made differentiable with a smooth approximation on a small interval $z \in [1 - \epsilon, 1 + \epsilon]$ near the hinge point. Differentiating (11) shows that the minimum w must satisfy

$$w = -C\sum_{l=1}^{L} y_l H_1^{'}(y_l) f_\theta(\boldsymbol{x}_l)\Phi(\boldsymbol{x}_i). \tag{12}$$

In this setting, correctly classified instances outside of the margin ($z \geq 1$) cannot become SVs because $H_1'(z) = 0$. On the other hand, for the training examples with ($z < 1$), $H_1'(z)$ is 1, so they cost a penalty term at the rate of misclassification of those instances. One problem with Hinge Loss-based optimization is that it imposes no limit on the influences of the outliers, that is, the misclassification penalty is unbounded. Furthermore, in Hinge Loss-based optimization, all misclassified training instances become support vectors. Consequently, the number of support vectors scales linearly with the number of training examples [19]. Specifically,

$$\frac{\#SV}{\#Examples} \rightarrow 2\mathcal{B}_\Phi, \tag{13}$$

where $\mathcal{B}_\Phi$ is the best possible error achievable linearly in the feature space $\Phi(\cdot)$. Such a fast pace of growth of the number of support vectors becomes prohibitive for training SVMs in large-scale data sets.

In practice, not all misclassified training examples are necessarily informative to the learner. For instance, in noisy data sets, many instances with label noise become support vectors due to misclassification, even though they are not informative about the correct classification of new instances in recognition. Thus, it is reasonable to limit the influence of

the outliers and allow the real informative training instances to define the model. Since Hinge Loss admits all outliers into the SVM solution, we need to select an alternative loss function that enables selectively ignoring the instances that are misclassified according to the current model. For this purpose, we propose using the Ramp Loss

$$R_s(z) = H_1(z) - H_s(z), \tag{14}$$

which allows us to control the score window for z at which we are willing to convert instances into support vectors. Replacing $H_1(z)$ with $R_s(z)$ in (12), we see that the Ramp Loss suppresses the influence of the instances with score $z < s$ by not converting them into support vectors. However, since Ramp Loss is nonconvex, it prohibits us from using widely popular optimization schemes devised for convex functions.

While convexity has many advantages and nice mathematical properties, the SV scaling property in (13) may be prohibitive for large-scale learning because all misclassified examples become support vectors. Since the nonconvex solvers are not necessarily bounded by this constraint, nonconvexity has the potential to generate sparser solutions [12]. In this work, our aim is to achieve the best of both worlds: generate a reliable and robust SVM solution that is faster and sparser than traditional convex optimizers. This can be achieved by employing the CCCP, and thus reducing the complexity of nonconvex loss function by transforming the problem into a difference of convex parts. The Ramp Loss is amenable to CCCP optimization since it can be decomposed into a difference of convex parts (as shown in Fig. 3 and (14)). The cost function $J^s(\boldsymbol{\theta})$ for the Ramp Loss can then be represented as the sum of a convex part $J_{vex}(\boldsymbol{\theta}^t)$ and a concave part $J_{cav}(\boldsymbol{\theta}^t)$:

$$\begin{aligned}\min_{\boldsymbol{\theta}} \; J^s(\boldsymbol{\theta}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} R_s(y_i f(x_i)) \\ &= \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} H_1(y_i f(x_i))}_{J^s_{vex}(\boldsymbol{\theta})} \underbrace{- C\sum_{l=1}^{n} H_s(y_i f(x_i))}_{J^s_{cav}(\boldsymbol{\theta})}.\end{aligned} \tag{15}$$

For simplification purposes, we use the notation

$$\beta_l = y_l \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(x_l)} = \begin{cases} C, & \text{if } y_l f_{\boldsymbol{\theta}}(x_l) < s, \\ 0, & \text{otherwise,} \end{cases} \tag{16}$$

where C is the misclassification penalty and $f_{\boldsymbol{\theta}}(x_l)$ is the kernel expansion defined as in (5) with the offset term $b = 0$. The cost function in (15), along with the notation introduced in (16), is then reformulated as the following dual optimization problem:

$$\begin{aligned}&\max_{\alpha} G(\alpha) = \sum_i y_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j K_{i,j}, \\ &\text{with} \begin{cases} A_i \le \alpha_i \le B_i, \\ A_i = \min(0, Cy_i) - \beta_i y_i, \\ B_i = \max(0, Cy_i) - \beta_i y_i, \\ \beta_i \text{ from (16)}. \end{cases}\end{aligned} \tag{17}$$

Collobert et al. [12] use a similar formulation for the CCCP-based nonconvex optimization of batch SVMs, but there are fundamental differences between optimization of batch SVMs and the online algorithms presented here. In particular, the batch SVM needs a convex initialization step prior to nonconvex iterations to go over the entire or part of the training data in order to initialize the CCCP parameters to avoid getting stuck in a poor local optima. Furthermore, batch nonconvex SVMs alternate between solving (17) and updating the βs of *all* training instances. On the other hand, LASVM-NC runs a few online convex iterations as the initialization stage, and adjusts the β of only the new fresh instance based on the current model and solves (17) while the online algorithm is progressing. Additionally, due to the nature of online learning, our learning scheme also permits selective sampling, which will be further discussed in the LASVM-I section.

We would also like to point out that if the βs of all of the training instances are initialized to zero and left unchanged in the online iterations, the algorithm becomes traditional Hinge Loss SVM. From another viewpoint, if $s \ll 0$, then the βs will remain zero and the effect of Ramp Loss will not be realized. Therefore, (17) can be viewed as a generic algorithm that can act as both Hinge Loss SVM and Ramp Loss SVM with CCCP that enables nonconvex optimization.

5.2 Online Iterations in LASVM-NC

The online iterations in LASVM-NC are similar to LASVM-G in the sense that they are also based on alternating PROCESS and REPROCESS steps, with the distinction of replacing the Hinge Loss with the Ramp Loss. LASVM-NC extends the LASVM-G algorithm with the computation of the $\boldsymbol{\beta}$, followed by updating the α bounds A and B as shown in (17). Note that while the $\boldsymbol{\beta}$ do not explicitly appear in the PROCESS and REPROCESS algorithm blocks, they do, in fact, affect these optimization steps through the new definition of the bounds A and B.

When a new example x_i is encountered, LASVM-NC first computes the β_i for this instance as presented in the algorithm block, where y_i is the class label, $f_{\boldsymbol{\theta}}(x_i)$ is the decision score for x_i, and s is the score threshold for permitting instances to become support vectors.

We would like to point out that CCCP has convergence guarantees (c.f. [12]), but it is necessary to initialize the CCCP algorithm appropriately in order to avoid getting trapped in poor local optima. In batch SVMs, this corresponds to running classical SVM on the entire set or on a subset of training instances in the first iteration to initialize CCCP, followed by the nonconvex optimization in the subsequent iterations. In the online setting, we initially allow convex optimization for the first few instances by setting their $\beta_i = 0$ (i.e., use Hinge Loss), and then switch to nonconvex behavior in the remainder of online iterations.

Note from (17) that the α bounds for instances with $\beta = 0$ follow the formulation for the traditional convex

LASVM-NC

$\mathbb{S}$: *min. number of SVs to start nonconvex behavior.*

1) **Initialization**:
   Set $\boldsymbol{\beta} \leftarrow \mathbf{0}$, $\alpha \leftarrow \mathbf{0}$
2) **Online Iterations**:
   Pick an example x_i

Set $\beta_i = \begin{cases} C & \text{if } y_i f_{\boldsymbol{\theta}}(x_i) < s \text{ and } \#\text{SV} > \mathbb{S} \\ 0 & \text{otherwise} \end{cases}$

Set α_i bounds for x_i to $(\min(0, Cy_i) - \beta_i y_i \leq \alpha_i \leq \max(0, Cy_i) - \beta_i y_i)$
Compute Gap Target $\hat{\mathbb{G}}$
$Threshold \leftarrow \max(C, \hat{\mathbb{G}})$
Run PROCESS(x_i)
while Gap $\mathcal{G} > Threshold$
  Run REPROCESS
end
Periodically run CLEAN

setting. On the other hand, the bounds for the instances with $\beta = C$, that is, the outliers with score ($z < s$) are assigned new bounds based on the Ramp Loss criteria. Once LASVM-NC establishes the α bounds for the new instance, it computes the Gap Target $\hat{\mathbb{G}}$ and takes a PROCESS step. Then, it makes optimizations of the REPROCESS kind until the size of the duality gap comes down to the Gap Threshold. Finally, LASVM-NC periodically runs the CLEAN operation to keep the size of the kernel expansion under control and maintain its efficiency throughout the training stage.

6 LASVM and Ignoring Instances—LASVM-I

This SVM algorithm employs the Ramp function in Fig. 3a as a filter to the learner *prior* to the PROCESS step. That is, once the learner is presented with a new instance, it first checks if the instance is on the ramp region of the function ($1 > y_i \sum_j \alpha_j K_{ij} > s$). The instances that are outside of the ramp region are not eligible to participate in the optimization steps and they are immediately discarded without further action. The rationale is that the instances that lie on the flat regions of the Ramp function will have derivative $H'(z) = 0$, and based on (12), these instances will not play role in determining the decision hyperplane $\boldsymbol{w}$.

The LASVM-I algorithm is based on the following record keeping that we conducted when running LASVM-G experiments. In LASVM-G and LASVM-NC, we kept track of two important data points. First, we intentionally permitted every new coming training instance into the kernel expansion in online iterations and recorded the position of all instances on the Ramp Loss curve just before inserting the instances into the expansion. Second, we kept track of the number of instances that were removed from the expansion which were on the flat region of the Ramp Loss curve when they were admitted. The numeric breakdown is presented in Table 1. Based on the distribution of these cleaned instances, it is evident that most of the cleaned examples that were initially admitted from ($z > 1$) region were removed from the kernel expansion with CLEAN at a later point in time. This is expected, since the instances with ($z > 1$) are already correctly classified by the current model with a certain confidence, and hence do not become SVs.

On the other hand, Table 1 shows that almost all of the instances inserted from the left flat region (misclassified examples due to $z < s$) became SVs in LASVM-G, and therefore were never removed from the kernel expansion. In contrast, almost all of the training instances that were admitted from the left flat region in LASVM-NC were removed from the kernel expansion, leading to a much larger reduction of the number of support vectors overall.

TABLE 1
Analysis of Adult Data Set at the End of the Training Stage

	Expansion		Admitted		Cleaned	
	# SV	# Non-SV	Ramp(L)	Ramp(R)	Ramp(L)	Ramp(R)
FULL SVM	11831	20731	32562		0	
LASVM-G	11265	0	1340	20252	1	19562
LASVM-NC	4609	0	1920	26032	1916	23823

Intuitively, the examples that are misclassified by a wide margin should not become support vectors. Ideally, the support vectors should be the instances that are within the margin of the hyperplane. As studies on Active Learning show [14], [20], the most informative instances to determine the hyperplane lie within the margin. Thus, LASVM-I ignores the instances that are misclassified by a margin ($z < s$) up front and prevents them from becoming support vectors.

LASVM-I

1) **Initialization**:
   Set $\alpha \leftarrow 0$
2) **Online Iterations**:
   Pick an example x_i
   Compute $z = y_i \sum_{j=0}^{n} \alpha_j K(x_i, x_j)$
   if ($z > 1$ or $z < s$)
    Skip x_i and bail out
   else
    Compute Gap Target $\hat{\mathbb{G}}$
   $Threshold \leftarrow \max(C, \hat{\mathbb{G}})$
   Run PROCESS(x_i)
   while Gap $\mathcal{G} > Threshold$
    Run REPROCESS
   end
   Periodically run CLEAN

Note that LASVM-I cannot be regarded as a nonconvex SVM solver since the instances with $\beta = C$ (which corresponds to $z < s$) are already being filtered out up front before the optimization steps. Consequently, all of the instances visible to the optimization steps have $\beta = 0$, which converts the objective function in (17) into the convex Hinge Loss from an optimization standpoint. Thus, combining these two filtering criteria ($z > 1$ and $z < s$), LASVM-I trades nonconvexity with a filtering Ramp function to determine whether to ignore an instance or proceed with optimization steps. Our goal with designing LASVM-I is that, based on this initial filtering step, it is possible to achieve further speedups in training times while maintaining competitive generalization performance. The experimental results validate this claim.

7 LASVM-G without Clean—FULL SVM

This algorithm serves as a baseline case for comparisons in our experimental evaluation. The learning principle of FULL SVM is based on alternating between LASVM-G's PROCESS and REPROCESS steps throughout the training iterations.

FULL SVM

1) **Initialization**:
   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:
Pick an example x_i
Compute Gap Target $\hat{\mathbb{G}}$
$Threshold \leftarrow \max(C, \hat{\mathbb{G}})$
Run PROCESS(x_i)
while Gap $\mathcal{G} > Threshold$
Run REPROCESS
end

When a new example is encountered, FULL SVM computes the Gap Target (given in (8)) and takes a PROCESS step. Then, it makes optimizations of the REPROCESS kind until the size of the duality gap comes down to the Gap Threshold. In this learning scheme, FULL SVM admits every new training example into the kernel expansion without any removal step (i.e., no CLEAN operation). This behavior mimics the behavior of traditional SVM solvers by providing that the learner has constant access to all training instances that it has seen during training and can make any of them a support vector any time if necessary. The SMO-like optimization in the online iterations of FULL SVM enables it to converge to the batch SVM solution.

Each PROCESS operation introduces a new instance to the learner, updates its α coefficient, and optimizes the objective function. This is followed by potentially multiple REPROCESS steps which exploit τ-violating pairs in the kernel expansion. Within each pair, REPROCESS selects the instance with maximal gradient, and potentially can zero the α coefficient of the selected instance. After sufficient iterations, as soon as a τ-approximate solution is reached, the algorithm stops updating the α coefficients. For full convergence to the batch SVM solution, running FULL SVM usually consists of performing a number of epochs where each epoch performs n online iterations by sequentially visiting the randomly shuffled training examples. Empirical evidence suggests that a single epoch yields a classifier almost as good as the SVM solution. For the theoretical explanation of the convergence properties of the online iterations, refer to [13].

The freedom to maintain and access the whole pool of seen examples during training in FULL SVM does come with a price though. The kernel expansion needs to constantly grow as new training instances are introduced to the learner, and it needs to hold all non-SVs in addition to the SVs of the current model. Furthermore, the learner still needs to include those non-SVs in the optimization steps and this additional processing becomes a significant drag on the training time of the learner.

8 Experiments

The experimental evaluation involves evaluating these outlined SVM algorithms on various data sets in terms of both their classification performances and algorithmic efficiencies leading to scalability. We also compare these algorithms against the reported metrics of LASVM and LIBSVM on the same data sets. In the experiments presented below, we run a single epoch over the training examples, all experiments use RBF kernels and the results averaged over 10 runs for each data set. Table 2 presents the characteristics of the data sets and the SVM parameters that were determined via 10-fold cross validation. *Adult* is a hard to classify census data set to predict if the income of the person is greater than 50K based on several census parameters, such as age, education, marital status, etc. *Mnist*, *USPS*, and *USPS-N* are optical character recognition data sets. USPS-N contains artificial noise which we generated by changing the labels of 10 percent of the training examples in the USPS data set. The Reuters-21578 is a popular text mining benchmark data set of 21,578 news stories that appeared on the Reuters newswire in 1987, and we test the algorithms with the Money-fx category of the Reuters-21578 data set. The Banana data set is a synthetic two-dimensional data set that has 4,000 patterns consisting of two banana-shaped clusters that have around 10 percent noise.

TABLE 2
Data Sets Used in the Experimental Evaluations and the SVM Parameters C and γ for the RBF Kernel

	Train Ex.	Test Ex.	# Features	C	$K(x, \bar{x})$
Adult (Census)	32562	16282	122	100	$e^{-0.005\|x-\bar{x}\|^2}$
Banana	4000	1300	2	10	$e^{-\|x-\bar{x}\|^2}$
Mnist (Digit 8)	60000	10000	784	100	$e^{-0.001\|x-\bar{x}\|^2}$
Reuters(Money-fx)	7770	3299	8315	1	$e^{-0.5\|x-\bar{x}\|^2}$
USPS	7329	1969	256	1	$e^{-2\|x-\bar{x}\|^2}$
USPS-N	7329	1969	256	1	$e^{-2\|x-\bar{x}\|^2}$

8.1 Generalization Performances

One of the metrics that we used in the evaluation of the generalization performances is Precision-Recall Breakeven Point (PRBEP) (see, e.g., [21]). Given the definition of precision as the number of correct positive class predictions among all positive class predictions and recall as the number of correct positive class predictions among all positive class instances, PRBEP is a widely used metric that measures the accuracy of the positive class where precision equals recall. In particular, PRBEP measures the trade-off between high precision and high recall. Fig. 4 shows the growth of PRBEP curves sampled over the course of training for the data sets. Compared to the baseline case FULL SVM, all algorithms are able to maintain competitive generalization performances at the end of training on all examples and show a more homogeneous growth compared to LASVM, especially for the Adult and Banana data sets. Furthermore, as shown in Table 3, LASVM-NC and LASVM-I actually yield higher classification accuracy for USPS-N compared to FULL SVM. This can be attributed to their ability to filter *bad* observations (i.e., noise) from training data. In noisy data sets, most of the noisy instances are misclassified and become support vectors in FULL SVM, LASVM-G, and LASVM due to Hinge Loss. This increase in the number of support vectors (see Fig. 6) causes SVM to learn complex classification boundaries that can overfit to noise, which can adversely affect their generalization performances. LASVM-NC and LASVM-I are less sensitive to noise, and they learn simpler models that are able to yield better generalization performances under noisy conditions.

For the evaluation of classification performances, we report three other metrics, namely, prediction accuracy (in Table 3), and AUC, and g-means (in Table 4). Prediction

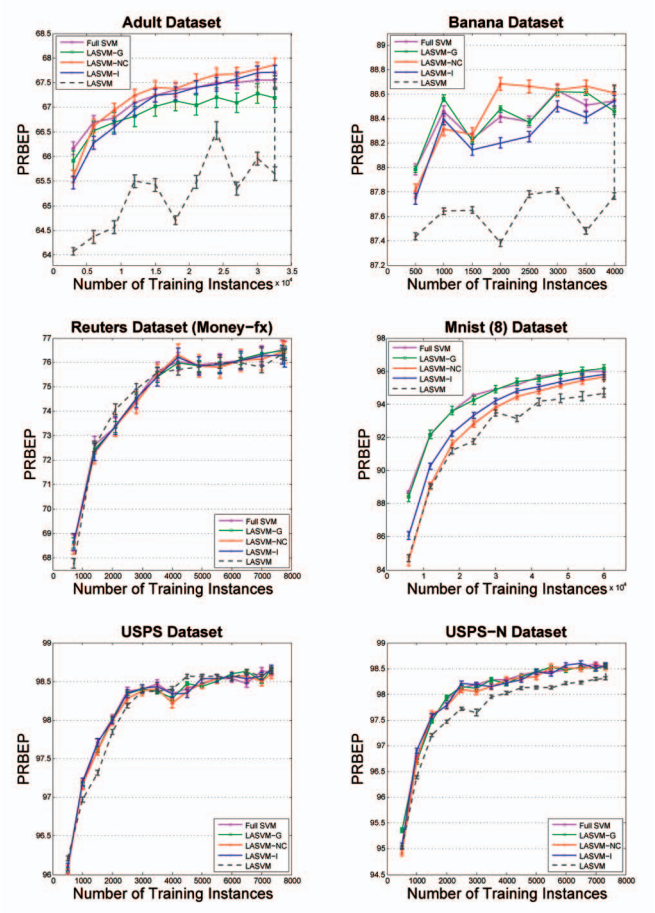

Fig. 4. PRBEP versus number of training instances. We used $s = -1$ for the ramp loss for LASVM-NC.

accuracy measures a model's ability to correctly predict the class labels of unseen observations. The *area under the ROC curve* (AUC) is a numerical measure of a model's discrimination performance and shows how correctly the model separates the positive and negative observations and ranks them. The Receiver operating characteristic (ROC) curve is the plot of sensitivity versus $1 - \text{specificity}$ and the AUC represents the area below the ROC curve. g-means is the geometric mean of *sensitivity* and *specificity* where sensitivity and specificity represent the accuracy on positive and negative instances, respectively. We report that all algorithms presented in this paper yield as good results for these performance metrics as FULL SVM and comparable classification accuracy to LASVM and LIBSVM. Furthermore, LASVM-NC yields the highest g-means for the Adult, Reuters, and USPS-N data sets compared to the rest of the algorithms.

We study the impact of the s parameter on the generalization performances of LASVM-NC and LASVM-I, and present our findings in Fig. 5. Since FULL SVM, LASVM-G, LASVM, and LIBSVM do not use Ramp Loss, they are represented with their testing errors and total number of support vectors achieved in the end of training. These plots depict that LASVM-NC and LASVM-I algorithms achieve competitive generalization performance with much fewer support vectors, especially for Adult, Banana, and USPS-N data sets. In all data sets, increasing the value of s into the positive territory actually has the effect of preventing

TABLE 3
Comparison of All Four SVM Algorithms with LASVM and LIBSVM for All Data Sets

		Datasets					
		Adult	Mnist(8)	Banana	Reuters	USPS	USPS-N
Accuracy	FULL SVM	84.87	99.25	90.03	97.19	99.54	98.43
	LASVM-G	84.81	99.27	89.81	97.19	99.54	99.42
	LASVM-NC	85.01	99.15	89.97	97.16	99.52	99.51
	LASVM-I	84.82	99.18	89.84	97.16	99.57	99.49
	LASVM	84.76	99.07	89.80	97.13	99.52	99.56
	LIBSVM	84.91	99.36	90.07	97.19	99.54	99.44
# SV	FULL SVM	11831	3412	947	1122	242	2455
	LASVM-G	11266	3157	941	1120	200	2288
	LASVM-NC	4609	2653	551	1086	194	752
	LASVM-I	5776	2722	669	1093	192	937
	LASVM	11296	2889	880	1140	201	2134
	LIBSVM	11420	3146	915	1090	237	2307
Train Time (sec)	FULL SVM	1186	4757.4	1	14.4	43.7	36
	LASVM-G	479	547.1	0.4	5.3	13.3	17
	LASVM-NC	129	526.0	0.4	4.7	8	8.3
	LASVM-I	92	491.4	0.3	3.6	5.8	6
	LASVM	376	640	0.5	7	9.5	52
	LIBSVM	318	1372.2	0.6	7.2	10.2	52

correctly classified instances that are within the margin from becoming SVs. This becomes detrimental to the generalization performance of LASVM-NC and LASVM-I since those instances are among the most informative instances to the learner. Likewise, moving s further down to the negative territory diminishes the effect of the Ramp Loss on the outliers. If $s \to -\infty$, then $R_s \to H_1$; in other words, if s takes large negative values, the Ramp Loss will not help to remove outliers from the SVM kernel expansion.

TABLE 4
Experimental Results That Assess the Generalization Performance and Computational Efficiency

		Datasets					
		Adult	Mnist(8)	Banana	Reuters	USPS	USPSN
PRBEP	FULL SVM	67.55	95.98	88.54	76.48	98.62	98.53
	LASVM-G	67.18	96.18	88.46	76.42	98.66	98.50
	LASVM-NC	67.85	95.64	88.61	76.42	98.59	98.53
	LASVM-I	67.71	95.80	88.54	76.20	98.65	98.56
	LASVM	67.50	95.72	88.59	76.31	98.63	95.48
AUC	FULL SVM	0.897	0.998	0.965	0.987	0.999	0.998
	LASVM-G	0.893	0.998	0.966	0.987	0.999	0.998
	LASVM-NC	0.901	0.998	0.965	0.987	0.999	0.998
	LASVM-I	0.899	0.998	0.964	0.987	0.999	0.998
	LASVM	0.896	0.998	0.965	0.987	0.999	0.998
Gmeans	FULL SVM	73.13	97.29	89.51	81.00	98.93	98.42
	LASVM-G	72.87	97.42	89.30	81.17	98.92	98.42
	LASVM-NC	75.03	96.86	89.47	81.19	98.99	98.74
	LASVM-I	73.72	97.06	89.38	81.06	98.89	98.72
	LASVM	73.35	97.33	89.28	81.04	98.99	98.43
# Kernel ($\times 10^6$)	FULL SVM	709.0	2269.7	8.51	33.2	27	30.2
	LASVM-G	233.0	182.6	3.8	9.9	5.2	12.3
	LASVM-NC	116.9	153.5	3.1	9.9	5.2	7.3
	LASVM-I	105.9	121.2	2.0	8	3.2	6.8
	LASVM	566	221	4.1	14.2	6.6	14.3

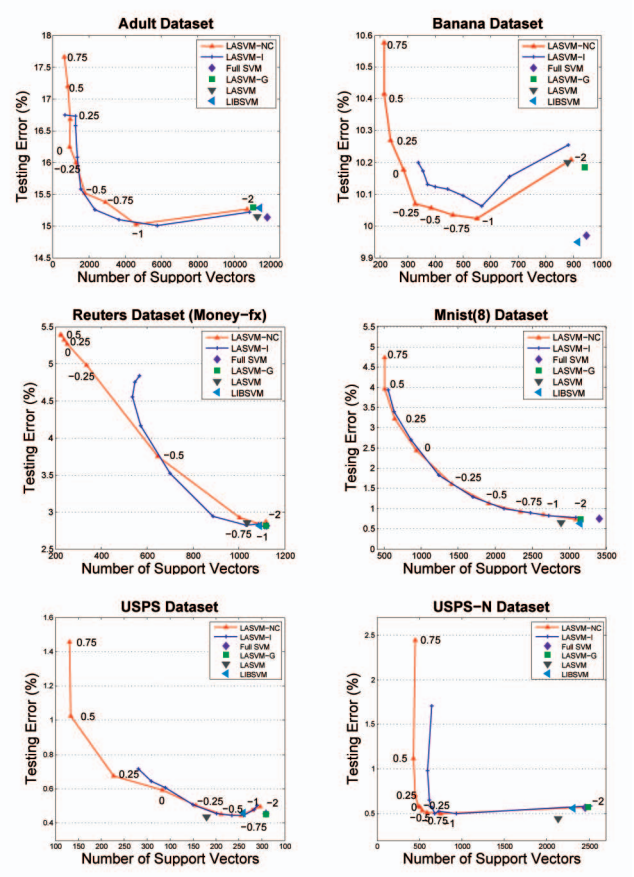

Fig. 5. Testing error versus number of support vectors for various settings of the s parameter of the ramp loss.

It is important to note that at the point $s = -1$, the Ramp-loss-based algorithms LASVM-NC and LASVM-I behave as an *Active Learning* [20], [22] framework. Active Learning is widely known as a querying technique for selecting the most informative instances from a pool of unlabeled instances to acquire their labels. Even in cases where the labels for all training instances are available beforehand, active learning can still be leveraged to select the most informative instances from training sets. In SVMs, the informativeness of an instance is synonymous with its distance to the hyperplane and the instances closer to the hyperplane are the most informative. For this reason, traditional min-margin active learners focus on the instances that are within the margin of the hyperplane and pick an example from this region to process next by searching the entire training set. However, such an exhaustive search is impossible in the online setup and computationally expensive in the offline setup. Ertekin et al. [14] suggest that querying for the most informative example does not need to be done from the entire training set, but instead, querying from randomly picked small pools can work equally well in a more efficient way. *Small pool active learning* first samples M random training examples from the entire training set and selects the best one among those M examples based on the condition that the selected instance among the top η percent closest instances in the entire training set with probability $(1-\eta)$, where $0 \leq \eta \leq 1$. With probability $1-\eta^M$, the value of the criterion for this example exceeds the η-quantile of the criterion for all training examples regardless of the size of the training set. In practice, this means that the best example among 59 random training examples has a 95 percent chance of belonging to the best 5 percent of examples in the training set.

In the extreme case of small pool active learning, setting the size of the pool to 1 corresponds to investigating whether that instance is within the margin or not. In this regard, setting $s = -1$ for the Ramp Loss in LASVM-NC and LASVM-I constrains the learner's focus only on the instances within the margin. Empirical evidence suggests that LASVM-NC and LASVM-I algorithms exhibit the benefits of active learning at $s = -1$ point, which yields the best results in most of our experiments. However, the exact setting for the s hyperparameter should be determined by the requirements of the classification task and the characteristics of the data set.

8.2 Computational Efficiency

A significant time-consuming operation of SVMs is the computation of kernel products $K(i,j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. For each new example, its kernel product with every instance in the kernel expansion needs to be computed. By reducing the number of kernel computations, it is possible to achieve significant computational efficiency improvements over traditional SVM solvers. In Fig. 7, we report the number of kernel calculations performed over the course of training iterations. FULL SVM suffers from uncontrolled growth of the kernel expansion, which results in a steep increase in the number of kernel products. This also shows why SVMs cannot handle large-scale data sets efficiently. In comparison, LASVM-G requires fewer kernel products than FULL SVM since LASVM-G keeps the number of instances in the kernel expansion under control by periodically removing uninformative instances through CLEAN operations.

LASVM-NC and LASVM-I yield significant reduction in the number of kernel computations and their benefit is most pronounced in the noisy data sets Adult, Banana, and USPS-N. LASVM-I achieves better reduction of kernel computations than LASVM-NC. This is due to the aggressive filtering done in LASVM-I where no kernel computation is performed for the instances on the flat regions of the Ramp Loss. On the other hand, LASVM-NC admits the instances that lie on the left flat region into the kernel expansion but achieves sparsity through the nonconvex optimization steps. The β values of those admitted instances are set to C in LASVM-NC so that they have new α bounds. Resulting from the new α bounds, these instances are more likely to be picked in the REPROCESS steps due to violating KKT conditions, and consequently removed from the set of support vectors. The reason for the low number of kernel products in LASVM-NC is due to its ability to create sparser models than other three algorithms. A comparison of the growth of the number of support vectors during the course of training is shown in Fig. 6. LASVM-NC and LASVM-I end up with smaller number of support vectors than FULL SVM, LASVM-G, and LIBSVM. Furthermore, compared to LASVM-I, LASVM-NC builds noticeably sparser models with less support vectors in noisy Adult, Banana, and USPS-N data sets. LASVM-I, on the other hand, makes fewer kernel calculations in training stage than LASVM-NC for those data sets. This is a key distinction of these two algorithms: The computational efficiency of LASVM-NC is the result of its ability to build sparse models. Conversely, LASVM-I creates

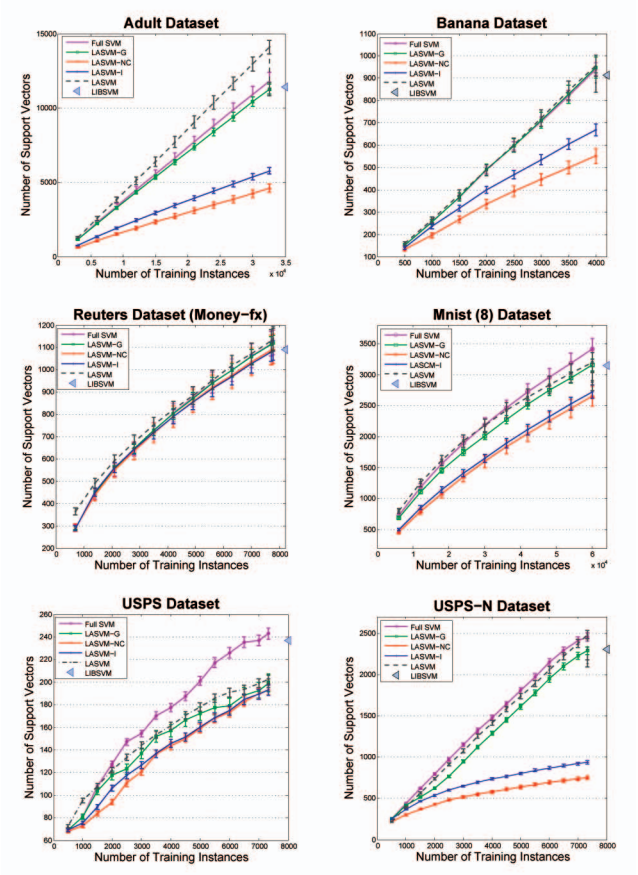

Fig. 6. Number of SVs versus number of training instances.

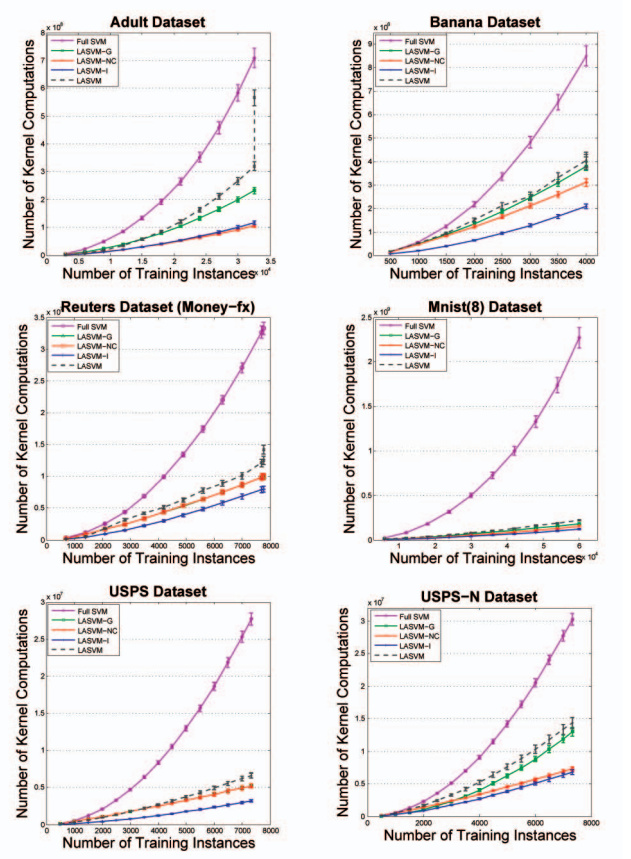

Fig. 7. Number of kernel computations versus number of training instances.

comparably more support vectors than LASVM-NC, but makes fewer kernel calculations due to early filtering. The overall training times for all data sets and all algorithms are presented both in Fig. 8 and Table 3. LASVM-G, LASVM-NC, and LASVM-I are all significantly more efficient than FULL SVM. LASVM-NC and LASVM-I also yield faster training than LIBSVM and LASVM. Note that the LIBSVM algorithm here uses a second order active set selection. Although second order selection can also be applied to LASVM-like algorithms to achieve improved speed and accuracy [23], we did not implement it in the algorithms discussed in this paper. Nevertheless, in Fig. 8, the fastest training times belong to LASVM-I where LASVM-NC comes close second. The sparsest solutions are achieved by LASVM-NC and this time LASVM-I comes close second. These two algorithms represent a compromise between training time versus sparsity and recognition time, and the appropriate algorithm should be chosen based on the requirements of the classification task.

9 Conclusion

In traditional convex SVM optimization, the number of support vectors scales linearly with the number of training examples, which unreasonably increases the training time and computational resource requirements. This fact has hindered widespread adoption of SVMs for classification tasks in large-scale data sets. In this work, we have studied the ways in which the computational efficiency of an online SVM solver can be improved without sacrificing the generalization performance. This paper is concerned with suppressing the influences of the outliers, which particularly becomes problematic in noisy data classification. For this purpose, we first present a systematic optimization approach for an online learning framework to generate more reliable and trustworthy learning models in intermediate iterations (LASVM-G). We then propose two online algorithms, LASVM-NC and LASVM-I, which leverage the Ramp function to avoid the outliers to become support vectors. LASVM-NC replaces the traditional Hinge Loss with the Ramp Loss and brings the benefits of nonconvex optimization using CCCP to an online learning setting. LASVM-I uses the Ramp function as a filtering mechanism to discard the outliers during online iterations. In online learning settings, we can discard new coming training examples accurately enough only when the intermediate models are reliable as much as possible. In LASVM-G, the increased stability of intermediate models is achieved by the duality gap policy. This increased stability in the model significantly reduces the number of wrongly discarded instances in online iterations of LASVM-NC and LASVM-I. Empirical evidence suggests that the algorithms provide efficient and scalable learning with noisy data sets in two respects: 1) *computational*: there is a significant decrease in the number of computations and running time during training and recognition, and 2) *statistical*: there is a significant decrease in the number of examples required for good generalization. Our findings also reveal that discarding the outliers by leveraging the Ramp function is closely related to the working principles of margin-based Active Learning.

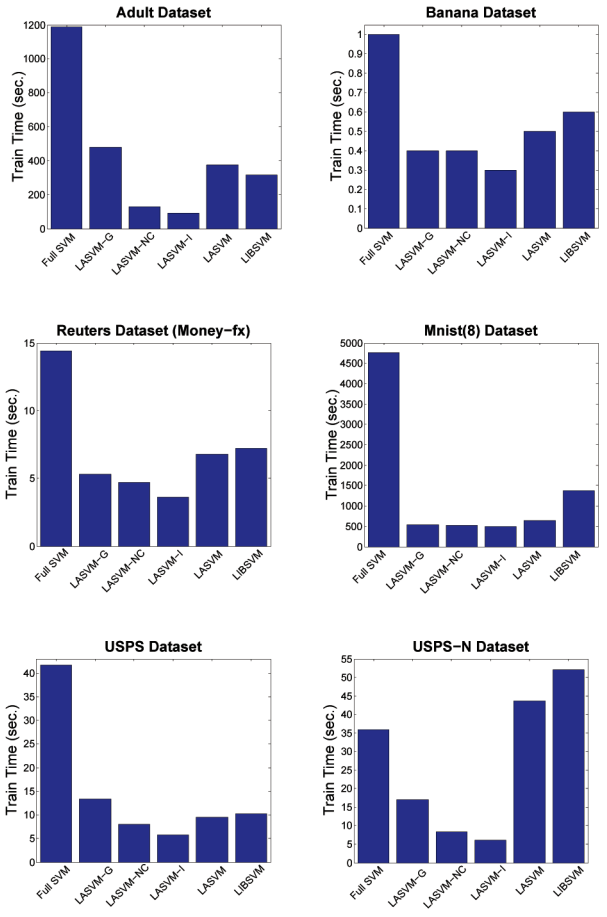

Fig. 8. Training times of the algorithms for all data sets after one pass over the training instances.

ACKNOWLEDGMENTS

This work was done while Şeyda Ertekin was with the Department of Computer Science and Engineering at the Pennsylvania State University and NEC Laboratories, America.

REFERENCES

[1] O. Bousquet and A. Elisseeff, "Stability and Generalization," *J. Machine Learning,* vol. 2, pp. 499-526, 2002.
[2] B. Schölkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.
[3] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.
[4] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.
[5] L. Mason, P.L. Bartlett, and J. Baxter, "Improved Generalization through Explicit Optimization of Margins," *Machine Learning,* vol. 38, pp. 243-255, 2000.
[6] N. Krause and Y. Singer, "Leveraging the Margin More Carefully," *Proc. Int'l Conf. Machine Learning,* p. 63, 2004.
[7] F. Perez-Cruz, A. Navia-Vazquez, and A.R. Figueiras-Vidal, "Empirical Risk Minimization for Support Vector Classifiers," *IEEE Trans. Neural Networks,* vol. 14, no. 2, pp. 296-303, Mar. 2002.
[8] D.S.L. Xu and K. Cramer, "Robust Support Vector Machine Training via Convex Outlier Ablation," *Proc. 21st Nat'l Conf. Artificial Intelligence,* 2006.
[9] Y. Liu, X. Shen, and H. Doss, "Multicategory ψ Learning and Support Vector Machine: Computational Tools," *J. Computational and Graphical Statistics,* vol. 14, pp. 219-236, 2005.
[10] L. Wang, H. Jia, and J. Li, "Training Robust Support Vector Machine with Smooth Ramp Loss in the Primal Space," *Neurocomputing,* vol. 71, pp. 3020-3025, 2008.
[11] A.L. Yuille and A. Rangarajan, "The Concave-Convex Procedure (CCCP)," *Advances in Neural Information Processing Systems.* MIT Press, 2002.
[12] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Trading Convexity for Scalability," *Proc. Int'l Conf. Machine Learning,* pp. 201-208, 2006.
[13] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast Kernel Classifiers with Online and Active Learning," *J. Machine Learning Research,* vol. 6, pp. 1579-1619, 2005.
[14] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the Border: Active Learning in Imbalanced Data Classification," *Proc. ACM Conf. Information and Knowledge Management,* pp. 127-136, 2007.
[15] J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods: Support Vector Learning,* pp. 185-208, MIT Press, 1999.
[16] S. Shalev-Shwartz and N. Srebro, "SVM Optimization: Inverse Dependence on Training Set Size," *Proc. Int'l Conf. Machine Learning,* pp. 928-935, 2008.
[17] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy, "Improvements to Platt's smo Algorithm for svm Classifier Design," *Neural Computation,* vol. 13, no. 3, pp. 637-649, 2001.
[18] O. Chapelle, "Training a Support Vector Machine in the Primal," *Neural Computation,* vol. 19, no. 5, pp. 1155-1178, 2007.
[19] I. Steinwart, "Sparseness of Support Vector Machines," *J. Machine Learning Research,* vol. 4, pp. 1071-1105, 2003.
[20] G. Schohn and D. Cohn, "Less Is More: Active Learning with Support Vector Machines," *Proc. Int'l Conf. Machine Learning,* pp. 839-846, 2000.
[21] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," Technical Report 23, Univ. Dortmund, 1997.
[22] S. Tong and D. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," *J. Machine Learning Research,* vol. 2, pp. 45-66, 2001.
[23] T. Glasmachers and C. Igel, "Second-Order SMO Improves SVM Online and Active Learning," *Neural Computation,* vol. 20, no. 2, pp. 374-382, 2008.

Şeyda Ertekin received the BSc degree in electrical and electronics engineering from Orta Dogu Teknik Universitesi (ODTU) in Ankara, Turkey, the MSc degree in computer science from the University of Louisiana at Lafayette, and the PhD degree in computer science and engineering from Pennsylvania State University–University Park in 2009. She is currently a postdoctoral research associate at the Massachusetts Institute of Technology (MIT). Her research interests focus on the design, analysis, and implementation of machine learning algorithms for large-scale data sets to solve real-world problems in the fields of data mining, information retrieval, and knowledge discovery. She is mainly known for her research that spans online and active learning for efficient and scalable machine learning algorithms. At Penn State, she was a member of the technical team of CiteSeerX. Throughout her PhD studies, she also worked as a researcher in the Machine Learning Group at NEC Research Laboratories in Princeton, New Jersey. Prior to that, she had worked at Aselsan, Inc., in Ankara, and worked on digital wireless telecommunication systems. She has also worked as a consultant to several companies in the US on the design of data mining infrastructures and algorithms. She is the recipient of numerous awards from the ACM, US National Science foundation (NSF), and Google.

Léon Bottou received the diplôme de l'Ecole Polytechnique, Paris, in 1987, the Magistère en mathématiques fondamentales et appliquées et informatiques from the Ecole Normale Supérieure, Paris, in 1988, and the PhD degree in computer science from the Université de Paris-Sud in 1991. He joined AT&T Bell Labs from 1991 to 1992 and AT&T Labs from 1995 to 2002. Between 1992 and 1995, he was the chairman of Neuristique in Paris, a small company pioneering machine learning for data mining applications. He joined NEC Labs America in Princeton, New Jersey in 2002. His primary research interest is machine learning. His contributions to this field address theory, algorithms, and large-scale applications. His secondary research interest is data compression and coding. His best known contribution in this field is the DjVu document compression technology (http://www.djvu.org). He is serving on the boards of the *Journal of Machine Learning Research* and the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He also serves on the scientific advisory board of Kxen, Inc. (http://www.kxen.com). He won the New York Academy of Sciences Blavatnik Award for Young Scientists in 2007.

C. Lee Giles is the David Reese professor of information sciences and technology at the Pennsylvania State University, University Park. He has appointments in the Departments of Computer Science and Engineering and Supply Chain and Information Systems. Previously, he was at NEC Research Institute, Princeton, New Jersey, and the Air Force Office of Scientific Research, Washington, District of Columbia. His research interests are in intelligent cyberinfrastructure, Web tools, search engines and information retrieval, digital libraries, Web services, knowledge and information extraction, data mining, name matching and disambiguation, and social networks. He was a cocreator of the popular search engines and tools: SeerSuite, CiteSeer (now CiteSeerX) for computer science, and ChemXSeer, for chemistry. He also was a cocreater of an early metasearch engine, Inquirus, the first search engine for robots.txt, BotSeer, and the first for academic business, SmealSearch. He is a fellow of the ACM, IEEE, and INNS.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.