

SMO-style algorithms for learning using privileged information

Dmitry Pechyony*

Rauf Izmailov†

Akshay Vashist‡

Vladimir Vapnik§

Abstract

Recently Vapnik et al. [11, 12, 13] introduced a new learning model, called *Learning Using Privileged Information (LUPI)*. In this model, along with standard training data, the teacher supplies the student with additional (privileged) information. In the optimistic case, the LUPI model can improve the bound for the probability of test error from $O(1/\sqrt{n})$ to $O(1/n)$, where n is the number of training examples. Since semi-supervised learning model with n labeled and N unlabeled examples can only achieve the bound $O(1/\sqrt{n+N})$ in the optimistic case, the LUPI model can thus significantly outperform it.

To implement LUPI model, Vapnik et al. [11, 12, 13] suggested to use an SVM-type algorithm called SVM+, which requires, however, to solve a more difficult optimization problem than the one that is traditionally used to solve SVM. In this paper we develop two new algorithms for solving the optimization problem of SVM+. Our algorithms have the structure similar to the empirically successful SMO algorithm for solving SVM. Our experiments show that in terms of the generalization error/running time tradeoff, one of our algorithms is superior over the widely used interior point optimizer.

1 Introduction

Recently Vapnik et al. [11, 12, 13] introduced a new learning model, called *Learning Using Privileged Information (LUPI)*. In this paradigm, in addition to the standard training data, $\mathbf{x} \in X$ and $y \in \{-1, 1\}$, a teacher supplies student with the *privileged information* $\mathbf{x}^* \in X^*$. The privileged information is only available for the training examples and is never available for the test examples. LUPI paradigm requires, given triplets $(\mathbf{x}_i, \mathbf{x}_i^*, y_i)$, $i = 1, \dots, n$, to find a function $h : X \rightarrow \{-1, 1\}$ with small generalization error for the unknown test data $\mathbf{x} \in X$. The privileged information appears in several domains [12, 13]. In time series prediction the privileged information is the behavior of the time series in the future (see example in Section 7) and in protein classification the privileged information is the 3-dimensional structure of the protein. Vapnik et al. [12, 13] showed that in these domains the LUPI paradigm can significantly improve over SVM solution.

LUPI paradigm can be implemented based on the well-known SVM algorithm [4]. The decision function

of SVM is $h(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} + b$, where \mathbf{z} is a feature map of \mathbf{x} , and \mathbf{w} and b are the solution of

$$(1.1) \quad \min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i$$

$$\text{s.t. } \forall 1 \leq i \leq n, y_i (\mathbf{w} \cdot \mathbf{z}_i + b) \geq 1 - \xi_i,$$

$$\forall 1 \leq i \leq n, \xi_i \geq 0.$$

Let \mathbf{z}_i^* be a feature map of \mathbf{x}_i^* . One way of realizing LUPI paradigm for SVM type of algorithms is to use the triplets $(\mathbf{x}_i, \mathbf{x}_i^*, y_i)$ to estimate two functions simultaneously: the decision function h and the *correcting function* (or *slack function*) $\phi(\mathbf{x}_i^*) = \mathbf{w}^* \cdot \mathbf{x}_i^* + d$, and to replace then the slack variables in SVM with the corresponding values of the slack function. This leads to the modification of SVM, called SVM+ [11]:

$$(1.2) \quad \min_{\mathbf{w}, b, \mathbf{w}^*, d} \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{\gamma}{2} \|\mathbf{w}^*\|_2^2 + C \sum_{i=1}^n (\mathbf{w}^* \cdot \mathbf{x}_i^* + d)$$

$$\text{s.t. } \forall 1 \leq i \leq n, y_i (\mathbf{w} \cdot \mathbf{z}_i + b) \geq 1 - (\mathbf{w}^* \cdot \mathbf{x}_i^* + d),$$

$$\forall 1 \leq i \leq n, \mathbf{w}^* \cdot \mathbf{x}_i^* + d \geq 0,$$

where $C > 0$ and $\gamma > 0$ are hyperparameters, \mathbf{z}_i^* is a feature map of \mathbf{x}_i^* . The term $\gamma \|\mathbf{w}^*\|_2^2 / 2$ is intended to restrict the capacity (or VC-dimension) of the function space containing ϕ that is found by (1.2).

In SVM+, the slacks $\phi(\mathbf{z}_i^*)$ are the values of the functions from a set Φ with a limited capacity, while in SVM the slacks ξ_i take arbitrary non-negative values. [13] and [12] show both theoretically and empirically that such a restriction of the capacity of the slack values can significantly improve the performance of SVM.

In this paper, we consider the algorithmic aspect of solving the optimization problem (1.2) of SVM+. A common approach to solve (1.1) is to consider its dual problem. We also apply this approach for (1.2). The dual optimization problem of (1.1) is

$$(1.3) \quad \max_{\alpha} D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_{ij}$$

$$(1.4) \quad \text{s.t. } \sum_{i=1}^n y_i \alpha_i = 0,$$

$$(1.5) \quad \forall 1 \leq i \leq n, 0 \leq \alpha_i \leq C,$$

*NEC Labs, Princeton, USA. pechyony@nec-labs.com

†NEC Corporation of America, New York, USA. rauf.izmailov@necam.com

‡NEC Labs, Princeton, USA. vashist@nec-labs.com

§NEC Labs, Princeton, USA. vlad@nec-labs.com

and the dual optimization problem of (1.2) is

$$(1.6) \quad \max_{\alpha, \beta} D(\alpha, \beta) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_{ij} \\ - \frac{1}{2\gamma} \sum_{i,j=1}^n (\alpha_i + \beta_i - C)(\alpha_j + \beta_j - C) K_{ij}^*$$

$$(1.7) \quad \text{s.t.} \quad \sum_{i=1}^n (\alpha_i + \beta_i - C) = 0, \quad \sum_{i=1}^n y_i \alpha_i = 0,$$

$$(1.8) \quad \forall 1 \leq i \leq n, \alpha_i \geq 0, \beta_i \geq 0,$$

where $K_{ij} \triangleq K(\mathbf{z}_i, \mathbf{z}_j)$ and $K_{ij}^* \triangleq K^*(\mathbf{z}_i^*, \mathbf{z}_j^*)$ are kernels in the decision and the correcting space.

One of the widely used algorithms for solving (1.3) is SMO algorithm of John Platt [9]. This algorithm works in the iterative way. At each iteration it optimizes the working set of two dual variables, α_s and α_t , while keeping all other variables fixed. Note that we cannot optimize the proper subset of such working set, say α_s : due to the constraint (1.4), if we fix $n-1$ variables then the last variable is also fixed. Hence the working sets selected by SMO are *irreducible*. The decision which pair of variables should be optimized at the current iteration is done by some working set selection rule (see [5] for the examples).

Motivated by the empirical success of SMO algorithm, we develop two SMO-type algorithms for solving (1.2). The first algorithm, called *alternating SMO* (**aSMO**), solves (1.6) by choosing at each iteration the irreducible working set, consisting of two or three dual variables. The possible working sets of **aSMO** are $\{\beta_s, \beta_t\}$, $\{\alpha_s, \alpha_t\}$ such that $y_s = y_t$, and $\{\alpha_r, \alpha_s, \beta_t\}$, such that $y_r \neq y_s$. In the last case, t may be one of $\{r, s\}$. We prove that, within a finite number of iterations, **aSMO** finds a solution which is arbitrary close to the global maximum of (1.6). The second algorithm, called *generalized SMO* (**gSMO**), solves (1.6) by choosing at each iteration the working set of four variables, $\alpha_s, \alpha_t, \beta_s$ and β_t . This previously unpublished algorithm was used to perform the experiments in [13, 12].

We compared of **aSMO**, **gSMO** and the generic optimizer LOQO, which is based on the interior point methods. For small training sets (up to 100 examples), LOQO is the fastest one, **aSMO** is the runner-up and **gSMO** is the slowest one. However, for larger training sets (e.g., of size 500) **aSMO** is much faster than the other two optimizers. We also found that, in terms of the generalization error of the solution, **aSMO** is better than the other two algorithms. Based on these observations we conclude that in terms of the tradeoff between the generalization error and the running time, **aSMO** is currently the preferable method for solving SVM+.

The paper has the following structure. In Sec. 2, we give the definitions that are used throughout the paper. In Sec. 3, we review the SMO algorithm for solving SVM. Based on this review, we develop in Sec. 4 **aSMO**. We prove the convergence of **aSMO** in Sec. 5. In Sec. 6 we present **gSMO**. We report our experiments in Sec. 7 and give future research directions in Sec. 8.

2 Definitions

The decision function and the correcting functions, expressed in the terms of the dual variables, are $h(\mathbf{x}) = \sum_{j=1}^n \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + b$ and $\phi(\mathbf{x}_i^*) = \frac{1}{\gamma} \sum_{j=1}^n (\alpha_j + \beta_j - C) K_{ij}^* + d$. In this paper, we consider algorithms solving the optimization problem

$$(2.9) \quad \max_{\mathbf{x} \in \mathcal{F}} f(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^m$, $f: \mathbb{R}^m \rightarrow \mathbb{R}$ is a concave function and \mathcal{F} is a convex compact domain. In SVM and SVM+, the function f is a quadratic function and \mathcal{F} is defined by a set of linear equalities and inequalities.

DEFINITION 2.1. A direction $\mathbf{u} \in \mathbb{R}^m$ is *feasible* at the point $\mathbf{x} \in \mathcal{F}$ if $\exists \lambda > 0$ such that $\mathbf{x} + \lambda \mathbf{u} \in \mathcal{F}$.

DEFINITION 2.2. Let $NZ(\mathbf{u}) \subseteq \{1, 2, \dots, n\}$ be a set of non-zero indices of $\mathbf{u} \in \mathbb{R}^n$. A direction \mathbf{u} is *maximally sparse feasible* if it is feasible and any $\mathbf{u}' \in \mathbb{R}^n$, such that $NZ(\mathbf{u}') \subset NZ(\mathbf{u})$, is not feasible.

3 Review of SMO Algorithm

Following [2], we present SMO as a line search optimization algorithm that is outlined in Algorithm 1. Let $I_1(\alpha) = \{i : (\alpha_i > 0 \text{ and } y_i = -1) \text{ or } (\alpha_i < C \text{ and } y_i = 1)\}$, $I_2(\alpha) = \{i : (\alpha_i > 0 \text{ and } y_i = 1) \text{ or } (\alpha_i < C \text{ and } y_i = -1)\}$. At each iteration, SMO chooses a direction $\mathbf{u}_s \in \mathbb{R}^n$ such that $\mathbf{s} = (i, j)$, $i \in I_1$, $j \in I_2$, $u_i = y_i$, $u_j = -y_j$ and for any $r \neq i, j$, $u_r = 0$. If we move from α in the direction \mathbf{u}_s then (1.4) is satisfied. It follows from the definition of I_1 , I_2 and \mathbf{u}_s , that $\exists \lambda > 0$ such that $\alpha + \lambda \mathbf{u}_s \in [0, C]^n$. Thus \mathbf{u}_s is a feasible direction. The direction \mathbf{u}_s is also maximally sparse. Indeed, any direction \mathbf{u}' with strictly more zero coordinates than in \mathbf{u}_s has a single nonzero coordinate. But if we move along such \mathbf{u}' then we will not satisfy (1.4).

Let $\mathbf{g}(\alpha)$ be a gradient of (1.3) at point α . By Taylor expansion of $\psi(\lambda) \triangleq D(\alpha^{\text{old}} + \lambda \mathbf{u}_s)$ at $\lambda = 0$, the maximum of $\psi(\lambda)$ along the half-line $\lambda \geq 0$ is at

$$(3.10) \quad \lambda'(\mathbf{s}) = - \frac{\frac{\partial D(\alpha^{\text{old}} + \lambda \mathbf{u}_s)}{\partial \lambda} |_{\lambda=0}}{\frac{\partial^2 D(\alpha^{\text{old}} + \lambda \mathbf{u}_s)}{\partial \lambda^2} |_{\lambda=0}} = - \frac{\mathbf{g}^T(\alpha^{\text{old}}) \mathbf{u}_s}{\mathbf{u}_s^T H \mathbf{u}_s},$$

Algorithm 1 High-level structure of SMO and alternating SMO algorithms.

Input: Function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, convex domain $\mathcal{F} \subset \mathbb{R}^n$, initial point $\mathbf{x}_0 \in \mathcal{F}$, constant $\tau > 0$.

- 1: Set $\mathbf{x} = \mathbf{x}_0$.
- 2: **while** exists a maximally sparse feasible direction \mathbf{u} such that $\nabla f(\mathbf{x})^T \mathbf{u} > \tau$ **do**
- 3: Choose a maximally sparse feasible direction \mathbf{u} such that $\nabla f(\mathbf{x})^T \mathbf{u} > \tau$.
- 4: Let $\lambda^* = \arg \max_{\lambda: \mathbf{x} + \lambda \mathbf{u} \in \mathcal{F}} f(\mathbf{x} + \lambda \mathbf{u})$.
- 5: Set $\mathbf{x} = \mathbf{x} + \lambda^* \mathbf{u}$.
- 6: **end while**
- 7: Output \mathbf{x} .

where H is a Hessian of $D(\boldsymbol{\alpha}^{\text{old}})$. The clipped value of $\lambda'(\mathbf{s})$ is

$$(3.11) \quad \lambda^*(\mathbf{s}) = \min_{i \in \mathbf{s}} \left(\frac{C - \alpha_i^{\text{old}}}{u_i}, \max_{i \in \mathbf{s}} \left(\lambda'(\mathbf{s}), -\frac{\alpha_i^{\text{old}}}{u_i} \right) \right)$$

so that the new value $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\text{old}} + \lambda^*(\mathbf{s}) \mathbf{u}_{\mathbf{s}}$ is always in the domain $[0, C]^n$.

Let $\tau > 0$ be a small constant and $I = \{\mathbf{u}_{\mathbf{s}} \mid \mathbf{s} = (i, j), i \in I_1, j \in I_2, \mathbf{g}(\boldsymbol{\alpha}^{\text{old}})^T \mathbf{u}_{\mathbf{s}} > \tau\}$. If $I = \emptyset$ then SMO stops. It can be shown that in this case Karush-Kuhn-Tucker (KKT) optimality conditions are almost satisfied (up to accuracy τ).

Let $I \neq \emptyset$. We describe a procedure for choosing the next direction. This procedure is currently implemented in LIBSVM and has complexity of $O(n)$. Let

$$(3.12) \quad \mathbf{s} = \arg \max_{\mathbf{t}: \mathbf{u}_{\mathbf{t}} \in I} \mathbf{g}(\boldsymbol{\alpha}^{\text{old}})^T \mathbf{u}_{\mathbf{t}}$$

be the nonzero components of the direction \mathbf{u} that is maximally aligned with the gradient. Earlier versions of LIBSVM chose $\mathbf{u}_{\mathbf{s}}$ as the next direction. As observed by [5], empirically faster convergence is obtained if instead of $\mathbf{s} = (i, j)$ we choose $\mathbf{s}' = (i, j')$ such that $\mathbf{u}_{\mathbf{s}'} \in I$ and the move in the direction $\mathbf{u}_{\mathbf{s}'}$ maximally increases (1.3):

$$(3.13) \quad \mathbf{s}' = \arg \max_{\substack{\mathbf{t}=(t_1, t_2) \\ t_1=i, \mathbf{u}_{\mathbf{t}} \in I}} D(\boldsymbol{\alpha}^{\text{old}} + \lambda'(\mathbf{t}) \mathbf{u}_{\mathbf{t}}) - D(\boldsymbol{\alpha}).$$

Alternatively, the clipped value of λ' can be used:

$$(3.14) \quad \mathbf{s}' = \arg \max_{\substack{\mathbf{t}=(t_1, t_2) \\ t_1=i, \mathbf{u}_{\mathbf{t}} \in I}} D(\boldsymbol{\alpha}^{\text{old}} + \lambda^*(\mathbf{t}) \mathbf{u}_{\mathbf{t}}) - D(\boldsymbol{\alpha}^{\text{old}}).$$

As shown by [5], with both choices the overall optimization time is about the same.

4 Alternating SMO for solving SVM+

Alternating SMO (aSMO) is an instantiation of Alg. 1 to the dual optimization problem (1.6) of SVM+. We now describe technical details of this instantiation.

We start with the identification of the maximally sparse feasible directions. Since (1.6) has $2n$ variables, $\{\alpha_i\}_{i=1}^n$ and $\{\beta_i\}_{i=1}^n$, the search direction $\mathbf{u} \in \mathbb{R}^{2n}$. We designate the first n coordinates of \mathbf{u} and \mathbf{g} for $\boldsymbol{\alpha}$'s and the last n coordinates for $\boldsymbol{\beta}$'s. It can be verified that (1.6) has 3 types of sets of maximally sparse feasible directions, $I_1, I_2, I_3 \subset \mathbb{R}^{2n}$, defined as follows:

1. $I_1 \triangleq \{\mathbf{u}_{\mathbf{s}} \mid \mathbf{s} = (s_1, s_2), n+1 \leq s_1, s_2 \leq 2n, s_1 \neq s_2; u_{s_1} = 1, u_{s_2} = -1, \beta_{s_2} > 0, \forall i \notin \mathbf{s} u_i = 0\}$. A move in the direction $\mathbf{u}_{\mathbf{s}} \in I_1$ increases β_{s_1} and decreases $\beta_{s_2} > 0$ by the same quantity. The rest of the variables remain fixed.
2. $I_2 \triangleq \{\mathbf{u}_{\mathbf{s}} \mid \mathbf{s} = (s_1, s_2), 0 \leq s_1, s_2 \leq n, s_1 \neq s_2, y_{s_1} = y_{s_2}; u_{s_1} = 1, u_{s_2} = -1, \alpha_{s_2} > 0, \forall i \notin \mathbf{s} u_i = 0\}$. A move in the direction $\mathbf{u}_{\mathbf{s}} \in I_2$ increases α_{s_1} and decreases $\alpha_{s_2} > 0$ by the same quantity. The rest of the variables remain fixed.
3. $I_3 \triangleq \{\mathbf{u}_{\mathbf{s}} \mid \mathbf{s} = (s_1, s_2, s_3), 0 \leq s_1, s_2 \leq n, n+1 \leq s_3 \leq 2n, s_1 \neq s_2; \forall i \notin \mathbf{s} u_i = 0; y_{s_1} \neq y_{s_2}; u_{s_1} = u_{s_2} = 1, u_{s_3} = -2, \beta_{s_3} > 0 \text{ or } u_{s_1} = u_{s_2} = -1, u_{s_3} = 2, \alpha_{s_1} > 0, \alpha_{s_2} > 0\}$. A move in the direction $\mathbf{u}_{\mathbf{s}} \in I_3$ either increases $\alpha_{s_1}, \alpha_{s_2}$ and decreases $\beta_{s_3} > 0$, or decreases $\alpha_{s_1} > 0, \alpha_{s_2} > 0$ and increases β_{s_3} . The absolute value of the change in β_{s_3} is twice the absolute value of the change in α_{s_1} and in α_{s_2} . The rest of the variables remain fixed.

We abbreviate $\mathbf{x} \triangleq (\boldsymbol{\alpha}, \boldsymbol{\beta})^T$ and $\mathbf{x}^{\text{old}} \triangleq (\boldsymbol{\alpha}^{\text{old}}, \boldsymbol{\beta}^{\text{old}})^T$. It can be verified that if we move from any feasible point \mathbf{x} in the direction $\mathbf{u}_{\mathbf{s}} \in I_1 \cup I_2 \cup I_3$ then (1.7) is satisfied. The optimization step in aSMO is $\mathbf{x} = \mathbf{x}^{\text{old}} + \lambda^*(\mathbf{s}) \mathbf{u}_{\mathbf{s}}$, where $\mathbf{u}_{\mathbf{s}} \in I_1 \cup I_2 \cup I_3$ and $\lambda^*(\mathbf{s}) \in \mathbb{R}$ maximizes $\psi(\lambda) = D(\mathbf{x}^{\text{old}} + \lambda \mathbf{u}_{\mathbf{s}})$ such that $\mathbf{x}^{\text{old}} + \lambda \mathbf{u}_{\mathbf{s}}$ satisfies (1.8). Let $\mathbf{g}(\mathbf{x}^{\text{old}})$ and H be respectively gradient and Hessian of (1.6) at the point \mathbf{x}^{old} and, similarly to (3.10),

$$(4.15) \quad \lambda'(\mathbf{s}) = \arg \max_{\lambda: \lambda \geq 0} \psi(\lambda) = -\frac{\mathbf{g}(\mathbf{x}^{\text{old}})^T \mathbf{u}_{\mathbf{s}}}{\mathbf{u}_{\mathbf{s}}^T H \mathbf{u}_{\mathbf{s}}}.$$

By (1.8) the clipped value of $\lambda'(\mathbf{s})$ is

$$(4.16) \quad \lambda^*(\mathbf{s}) = \max_{i \in \mathbf{s}} \left(\lambda'(\mathbf{s}), -\frac{x_i^{\text{old}}}{u_i} \right).$$

Let $\tau > 0$ be a small constant and $I = \{\mathbf{u}_{\mathbf{s}} \mid \mathbf{u}_{\mathbf{s}} \in I_1 \cup I_2 \cup I_3 \text{ and } \mathbf{g}(\mathbf{x}^{\text{old}})^T \mathbf{u}_{\mathbf{s}} > \tau\}$. If $I = \emptyset$ then, similarly to SMO, aSMO stops.

Suppose that $I \neq \emptyset$. We choose the next direction in 3 steps. The first 2 steps are similar to the ones done by LIBSVM and are described in Sec. 3. At the first step, for each $1 \leq i \leq 3$ we find a vector $\mathbf{u}_{\mathbf{s}} \in I_i$ that has a minimal angle with $\mathbf{g}(\mathbf{x}^{\text{old}})$ among all vectors in I_i . For $1 \leq i \leq 3$, if $I_i \neq \emptyset$ then $\mathbf{s}^{(i)} = \arg \max_{\mathbf{s}: \mathbf{u}_{\mathbf{s}} \in I_i} \mathbf{g}(\mathbf{x}^{\text{old}})^T \mathbf{u}_{\mathbf{s}}$, otherwise $\mathbf{s}^{(i)}$ is empty.

For $1 \leq i \leq 3$, let $\tilde{I}_i \triangleq \{\mathbf{u}_s \mid \mathbf{u}_s \in I_i \text{ and } \mathbf{g}(\mathbf{x}^{\text{old}})^T \mathbf{u}_s > \tau\}$. At the second step for $1 \leq i \leq 2$, if the pair $\mathbf{s}^{(i)} = (s_1^{(i)}, s_2^{(i)})$ is not empty then we fix $s_1^{(i)}$ and find $\mathbf{s}'^{(i)} = (s_1^{(i)}, s_2'^{(i)})$ such that $\mathbf{u}_{\mathbf{s}'^{(i)}} \in \tilde{I}_i$ and the move in the direction $\mathbf{u}_{\mathbf{s}'^{(i)}}$ maximally increases (1.6):

$$\mathbf{s}'^{(i)} = \arg \max_{\substack{\mathbf{t}: \mathbf{t}=(t_1, t_2) \\ t_1=s_1^{(i)}, \mathbf{u}_t \in \tilde{I}_i}} \underbrace{D(\mathbf{x}^{\text{old}} + \lambda'(\mathbf{t})\mathbf{u}_t) - D(\mathbf{x}^{\text{old}})}_{\triangleq \Delta_i(\mathbf{t})}$$

Similarly, for $i = 3$ we fix $s_1^{(3)}$ and $s_3^{(3)}$ and find $\mathbf{s}'^{(3)} = (s_1^{(3)}, s_2'^{(3)}, s_3^{(3)})$ such that $\mathbf{u}_{\mathbf{s}'^{(3)}} \in \tilde{I}_3$ and the move in the direction $\mathbf{u}_{\mathbf{s}'^{(3)}}$ maximally increases (1.6). At the third step, we choose $j = \arg \max_{1 \leq i \leq 3, \mathbf{s}'^{(i)} \neq \emptyset} \Delta_i(\mathbf{s}'^{(i)})$. The final search direction is $\mathbf{s}'^{(j)}$.

Since the direction vectors $\mathbf{u}_s \in I_1 \cup I_2 \cup I_3$ have a constant number of nonzero components, it can be shown that our procedure for choosing the next direction has complexity of $O(n)$.

5 Convergence results for alternating SMO

For any feasible point $\mathbf{x} \in \mathcal{F}$, let $\psi(\mathbf{x}, \mathbf{u}) = \max\{\lambda \geq 0 \mid \mathbf{x} + \lambda \mathbf{u} \in \mathcal{F}\}$. We have that $\psi(\mathbf{x}, \mathbf{u}) > 0$ iff \mathbf{u} is a feasible direction at \mathbf{x} . Bordes et al. [1] presented a modification of Alg. 1. Instead of maximally sparse vectors \mathbf{u} , at each iteration they consider directions \mathbf{u} from a set \mathcal{U} such that $\psi(\mathbf{x}, \mathbf{u}) \geq \kappa > 0$. This inequality ensures that the optimization step, made by the algorithm, is sufficiently large. This requirement allows to prove finite-time convergence (see also the discussion below). The optimization algorithm of [1] is formalized in Alg. 2.

aSMO is obtained from Alg. 2 by setting $\mathcal{U} = I_1 \cup I_2 \cup I_3$ and $\kappa = 0$. Since Alg. 2 requires that $\kappa > 0$, aSMO is different from it. In our experiments, we used a modified version of aSMO by replacing the conditions $\alpha_t > 0$ and $\beta_t > 0$ in the definition of $I_1 - I_3$ with $\alpha_t > \kappa > 0$ and $\beta_t > \kappa > 0$. We use the results of [1] to demonstrate convergence properties of this modified version of aSMO. Our results are based on the following definition and theorem of [1].

DEFINITION 5.1. ([1]) *A set of directions $\mathcal{U} \subset \mathbb{R}^n$ is a **finite witness family** for a convex set \mathcal{F} if \mathcal{U} is finite and for any $\mathbf{x} \in \mathcal{F}$, any feasible direction \mathbf{u} at the point \mathbf{x} is a positive linear combination of a finite number of feasible directions $\mathbf{v}_i \in \mathcal{U}$ at the point \mathbf{x} .*

THEOREM 5.1. ([1]) *Let \mathcal{U} be a finite witness family for \mathcal{F} . Then Alg. 2 stops after a finite number of iterations. Let $\mathbf{x}^*(\tau, \kappa)$ be the solution found by Alg. 2 for a given τ and κ and let \mathbf{x}^* be a solution of (2.9). Then $\lim_{\tau \rightarrow 0, \kappa \rightarrow 0} \mathbf{x}^*(\tau, \kappa) = \mathbf{x}^*$.*

Algorithm 2 Optimization procedure of [1].

Input: Function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, domain $\mathcal{F} \subset \mathbb{R}^n$, set of directions $\mathcal{U} \subseteq \mathbb{R}^n$, initial point $\mathbf{x}_0 \in \mathcal{F}$, constants $\tau > 0, \kappa > 0$.

- 1: Set $\mathbf{x} = \mathbf{x}_0$.
 - 2: **while** exists $\mathbf{u} \in \mathcal{U}$ such that $\nabla f(\mathbf{x})^T \mathbf{u} > \tau$ and $\psi(\mathbf{x}, \mathbf{u}) > \kappa$ **do**
 - 3: Choose any $\mathbf{u} \in \mathcal{U}$ such that $\nabla f(\mathbf{x})^T \mathbf{u} > \tau$ and $\psi(\mathbf{x}, \mathbf{u}) > \kappa$.
 - 4: Let $\lambda^* = \arg \max_{\lambda: \mathbf{x} + \lambda \mathbf{u} \in \mathcal{F}} f(\mathbf{x} + \lambda \mathbf{u})$.
 - 5: Set $\mathbf{x} = \mathbf{x} + \lambda^* \mathbf{u}$.
 - 6: **end while**
 - 7: Output \mathbf{x} .
-

SMO is obtained from Alg. 2 by setting $\kappa = 0$ and $\mathcal{U} = \{\mathbf{e}_i - \mathbf{e}_j, i \neq j\}$, where $(\mathbf{e}_1, \dots, \mathbf{e}_n)$ is a canonical basis of \mathbb{R}^n . Hence SMO is different from Alg. 2. Keerthi and [7] and [10] prove finite time convergence of SMO with $\kappa = 0$. Their proof is different from the proof of Theorem 5.1, which requires $\kappa > 0$. The adaptation of the finite-time convergence proof of [7] and [10] to aSMO with $\kappa = 0$ is an open problem.

The following Proposition¹, combined with Theorem 5.1, ensures that aSMO converges after a finite number of iterations.

PROPOSITION 5.1. *The set $I_1 \cup I_2 \cup I_3$ is a finite witness family for the set \mathcal{F} defined by (1.7)-(1.8).*

6 Generalized SMO (gSMO)

As alternating aSMO, gSMO works in the iterative way. At each iteration, it chooses the working set of 2 indices, s and t , and maximizes the part of (1.6), denoted by $D(\alpha_s, \alpha_t, \beta_s, \beta_t)$, that depends on the variables corresponding to the s th and the t th examples.

Let $\rho = y_s y_t = \pm 1$, $\Delta = a_s^{\text{old}} + \rho \alpha_t^{\text{old}}$, $\delta_i = \alpha_i + \beta_i - C$ and $\mu = - \sum_{i=1; i \neq s, t}^n (\alpha_i^{\text{old}} + \beta_i^{\text{old}} - C)$. It follows from (1.7) that

$$(6.17) \quad \alpha_s = \Delta - \rho \alpha_t, \quad \beta_s = \mu - \delta_t + C - \Delta + \rho \alpha_t$$

We substitute (6.17) into $D(\alpha_s, \alpha_t, \beta_s, \beta_t)$ and obtain a function of 2 variables $D(\alpha_t, \beta_t)$. Let $(\alpha_t^*, \beta_t^*) = \arg \min_{\alpha_t \in \mathbb{R}, \beta_t \in \mathbb{R}} D(\alpha_t, \beta_t)$.

At the next step we project (α_t^*, β_t^*) on the feasible domain \mathcal{F} defined by the following four constraints (derived from (1.8)):

$$\begin{cases} \alpha_t \geq 0, & \alpha_s = \Delta - \alpha_t \rho \geq 0 \\ \beta_t \geq 0, & \beta_s = \mu - (\Delta - \alpha_t \rho - C) - \delta_t \geq 0 \end{cases}$$

¹The proof, omitted because of the space constraints, can be found at www.nec-labs.com/~pechyony/svplus.smo.pdf.

The domain \mathcal{F} is a rectangle or a triangle, depending on the value of ρ :

Case 1: $\rho = +1$. In this case \mathcal{F} is a rectangle:

$$\begin{cases} 0 \leq \alpha_t \leq \Delta \\ 0 \leq \beta_t \leq \mu - \Delta + 2C \end{cases}$$

The domain \mathcal{F} is non-empty, because $\Delta \geq 0$ and $\mu - \Delta + 2C \geq 0$. Indeed, since $\rho = 1$, then $\Delta = \alpha_s + \alpha_t \geq 0$. Also, $\beta_s + \beta_t = \mu - (\alpha_s + \alpha_t) + 2C = \mu - \Delta + 2C \geq 0$.

Case 2: $\rho = -1$. In this case \mathcal{F} is a triangle:

$$\begin{cases} \alpha_t \geq L = \max(0, -\Delta) \\ \beta_t \geq 0 \\ 2\alpha_t + \beta_t \leq \mu - \Delta + 2C \end{cases}$$

We denote $H = (\mu - \Delta + 2C)/2$; then $L \leq \alpha_t \leq H$ and $0 \leq \beta_t \leq 2(H - L)$. It can be verified that the inequality $H \geq L$; always holds and thus $\mathcal{F} \neq \emptyset$.

If $(\alpha_t^*, \beta_t^*) \in \mathcal{F}$ then (α_t^*, β_t^*) is the maximum point of $D(\alpha_t, \beta_t)$ over \mathcal{F} . Otherwise, since $D(\alpha_t, \beta_t)$ is a quadratic function, its maximum point over \mathcal{F} belongs to one of its boundaries. We now consider each of the boundaries of \mathcal{F} separately and execute the following operations for each of them:

1. Restrict $D(\alpha_t, \beta_t)$ on the corresponding boundary. The result would be a quadratic function of a single argument, α_t or β_t .
2. Maximize the restricted function $D(\alpha_t, \beta_t)$ on the corresponding boundary.

Let $B_i \subset \mathcal{F}$ be the i -th boundary of \mathcal{F} , $1 \leq i \leq 4$ if $\rho = 1$ and $1 \leq i \leq 3$ if $\rho = -1$. Also, let $(\alpha_t^{(i)}, \beta_t^{(i)}) = \arg \max_{(\alpha_t, \beta_t) \in B_i} D(\alpha_t, \beta_t)$. The maximal point of $D(\alpha_t, \beta_t)$ over \mathcal{F} is $(\alpha_t^*, \beta_t^*) = \arg \max_i D(\alpha_t^{(i)}, \beta_t^{(i)})$, where $1 \leq i \leq 4$ if $\rho = 1$ and $1 \leq i \leq 3$ if $\rho = -1$.

Similarly to LIBSVM, we choose the working set based on the Karush-Kuhn-Tucker (KKT) conditions. Let $F_i \triangleq \sum_{j=1}^n y_j \alpha_j^{\text{old}} K_{ij}$ and $f_i \triangleq \sum_{j=1}^n (\alpha_j^{\text{old}} + \beta_j^{\text{old}} - C) K_{ij}$. The Lagrangian of the dual (1.6) is

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\nu}, \kappa, \eta) &= -\frac{1}{2} \sum_{i,j=1}^n y_i y_j K_{ij} \alpha_i \alpha_j - \\ &-\frac{1}{2\gamma} \sum_{i,j=1}^n K_{ij}^* \delta_i \delta_j + \sum_{i=1}^n \alpha_i - \\ &-\kappa \sum_{i=1}^n \alpha_i y_i - \eta \sum_{i=1}^n \delta_i - \nu_i \alpha_i - \tau_i \beta_i, \end{aligned}$$

where $\boldsymbol{\tau}, \boldsymbol{\nu} \in \mathbb{R}^n$ and $\kappa, \eta \in \mathbb{R}$ are the Lagrangian multipliers. It can be shown that at the optimal solution of (1.6) $\kappa = b$ and $\eta = d$. The KKT conditions for the dual of (1.6) are

$$\begin{aligned} \frac{\partial L}{\partial \alpha_i} &= 1 - y_i F_i - b y_i - \nu_i - f_i / \gamma - d = 0, \\ \frac{\partial L}{\partial \beta_i} &= -f_i / \gamma - d - \tau_i = 0, \end{aligned}$$

$$\forall 1 \leq i \leq n, \nu_i \alpha_i = 0, \tau_i \beta_i = 0, \nu_i \geq 0, \tau_i \geq 0.$$

Depending on the values of α_i and β_i , we have the following cases:

$$\begin{aligned} \text{Case } A_{>+} & [\alpha_i > 0, y_i = 1] \quad 1 - y_i F_i - f_i / \gamma - d = b \\ \text{Case } A_{>-} & [\alpha_i > 0, y_i = -1] \quad -1 + y_i F_i + f_i / \gamma + d = b \\ \text{Case } A_{=+} & [\alpha_i = 0, y_i = 1] \quad 1 - y_i F_i - f_i / \gamma - d \geq b \\ \text{Case } A_{=-} & [\alpha_i = 0, y_i = -1] \quad -1 + y_i F_i + f_i / \gamma + d \leq b \\ \text{Case } B_{>} & [\beta_i > 0] \quad -f_i \gamma = d \\ \text{Case } B_{=} & [\beta_i = 0] \quad -f_i \gamma \geq d \end{aligned}$$

At the optimality we have $d_{\max} \leq d_{\min}$, where $d_{\min} = \min\{f_i / \gamma : i \in B_{=}\}$ and $d_{\max} = \max\{f_i / \gamma : 1 \leq i \leq n\}$. Initially we try to choose the working set based on the KKT conditions for d . If $d_{\max} > d_{\min} + \tau$, where τ is a small (10^{-3}) constant, then we choose as a working set a pair of indices $1 \leq s \leq n$ and $t \in B_{=}$ that maximally violate this inequality. Otherwise we compute $d = (d_{\max} + d_{\min})/2$ and try to choose the working set based on KKT condition for b . Let $A_{\text{up}} = A_{=+} \cup A_{>+} \cup A_{>-}$ and $A_{\text{down}} = A_{=-} \cup A_{>-} \cup A_{>+}$. At the optimality we must have $b_{\max} \geq b_{\text{eq}}$, where $b_{\max} = \min\{y_i(1 - y_i F_i - f_i / \gamma - d) : i \in A_{\text{up}}\}$ and $b_{\text{eq}} = \max\{y_i(1 - y_i F_i - f_i / \gamma - d) : i \in A_{\text{down}}\}$. If $b_{\max} < b_{\text{eq}} + \tau$, then we choose a pair of indices $s \in A_{\text{up}}$ and $t \in A_{\text{down}}$ that maximally violate this inequality. Otherwise the optimality conditions are reached and the optimization finishes.

7 Experiments

7.1 Datasets We used two datasets, that were also used in [13, 12]. The first dataset was obtained from the MNIST dataset² for digit recognition. We considered the binary problem “5” vs. “8”. The master training set, of size 100, was formed by the first 50 images of “5” and the first 50 images of “8”. Training sets of the smaller size, from 40 to 90, were generated by randomly sampling the master training set. Each training example was supplied with a poetic (free text) description of the corresponding image (see [13] for the examples). The poetic description was manually translated into 21-dimensional real-valued vector, which served as a privileged information. Each component of the vector of privileged information describes some salient feature of the corresponding picture. In all our experiments with MNIST dataset, we used a fixed validation set of 4002 examples and a fixed test set of 1866 examples.

The second dataset was obtained from Mackey-Glass time series [8], described by the differential equation $\frac{dx(t)}{dt} = -ax(t) + \frac{bx(t-\tau)}{1+x^{10}(t-\tau)}$, $t \geq 0$, where a, b and

²Available at <http://yann.lecun.com/exdb/mnist/>

Table 1: Mean running time (in seconds) on Mackey-Glass dataset

	$\Delta = 1$	$\Delta = 5$	$\Delta = 8$
gSMO, X*SVM+	28.7	29.91	35.54
aSMO, X*SVM+	3.83	5.1	4.28
LOQO, X*SVM+	0.68	2.09	2.14
gSMO, dSVM+	23.57	32.24	37.24
aSMO, dSVM+	3.48	5.36	5.42
LOQO, dSVM+	failed	1.48	2.42

τ are constants. We used the code³ of Roger Jang that generates this time series. We considered the binary problem of predicting if $x(t + \Delta) > x(t)$ based on the feature vector $(x(t - 3), x(t - 2), x(t - 1), x(t))$, where $\Delta > 0$ is a fixed parameter. As in [12], we set $\Delta = 1, 5, 8$ and generated three binary problems. As a privileged information, we took the vicinity of $x(t + \Delta)$, i.e. the vector $(x(t + \Delta - 2), x(t + \Delta - 1), x(t + \Delta + 1), x(t + \Delta + 2))$. In our experiments with Mackey-Glass dataset, we used a random validation and test sets, both of size 2000.

7.2 Empirical Results We implemented aSMO and gSMO on the top of LIBSVM [3]. We also used LIBSVM as a benchmark implementation of SVM. Following [13, 12] we ran SVM+ in two modes. In the first mode, denoted X*SVM+, we solved (1.6) with the training set $\{(x_i, x_i^*, y_i)\}_{i=1}^n$. In the second mode, denoted dSVM+, we ran SVM over the training set $\{x_i^*\}_{i=1}^n$, obtained the decision function f^* in the space X^* , set $d_i = 1 - y_i f^*(z_i^*)$ and solved (1.6) with the training set $\{(x_i, d_i, y_i)\}_{i=1}^n$. This procedure makes the slacks in the space X^* to be a (nonlinear) function of the slacks in the space X . We used RBF kernel for both X and X^* .

We chose C, σ, σ^* and γ by optimizing the validation error over 4-dimensional grid of parameters, G_0 , of size $10 \times 10 \times 6 \times 6 = 3600$. Initially, we found three combinations of hyperparameters in G_0 , denoted by $a_1^{(0)}, a_2^{(0)}$ and $a_3^{(0)}$, with the lowest validation error. Then we performed 3 zooming steps. At the i th step we constructed 3 smaller grids, $G_1^{(i)}, G_2^{(i)}, G_3^{(i)}$, (of size $5 \times 5 \times 5 \times 5$) around $a_j^{(i-1)}, j = 1, 2, 3$, and found three combinations of hyperparameters in $G_1^{(i)} \cup G_2^{(i)} \cup G_3^{(i)}$, denoted by $a_1^{(i)}, a_2^{(i)}, a_3^{(i)}$, with the lowest validation error. We chose among $a_1^{(3)}, a_2^{(3)}, a_3^{(3)}$ the combination of hyperparameters with the lowest validation error.

To demonstrate the need for specialized optimizers for SVM+, we compared aSMO and gSMO with a number of state-of-the-art optimizers, CVXOPT⁴, LOQO⁵ and

Table 2: Errors on Mackey-Glass dataset

	$\Delta = 1$	$\Delta = 5$	$\Delta = 8$
SVM	1.63	4.78	7.13
gSMO, X*SVM+	1.17	4.36	7.09
aSMO, X*SVM+	1.17	3.64	4.95
LOQO, X*SVM+	1.57	3.28	4.47
gSMO, dSVM+	1.19	4.37	7.18
aSMO, dSVM+	1.26	3.27	4.8
LOQO, dSVM+	failed	3.35	4.65

QP solver of Yinyu Ye⁶. These solvers are based on interior point optimization algorithms. We present the comparison with LOQO, which gave the best results among the off-the-shelf optimizers.

We performed 12 random draws of the training sets and report in Table 1 and Fig. 1(a)-(b) the average running times of aSMO, gSMO and LOQO on Mackey-Glass and MNIST datasets respectfully. The average was taken over the draws of the training set and over all considered hyperparameters. This comparison is done for very small training set, of size 40-100. For one of the twelve draws in Mackey-Glass with $\Delta = 1$, LOQO has failed to finish the experiment within a week. In the rest of these experiments the fastest algorithm is LOQO, the runner-up is aSMO the slowest is gSMO. We also performed a larger experiment for Mackey-Glass dataset, with training size of 500. aSMO completed this experiment within three days. In contrary, the corresponding runs of gSMO and LOQO were far from being finished after ten days. The obtained running times of LOQO are not surprising, since the interior point algorithms have cubic complexity and thus are feasible only for very small datasets.

Our experiments show that aSMO is consistently faster than gSMO. Since aSMO updates less variables than gSMO, each iteration of the former algorithms is faster than the one of latter algorithm. We observed that the speedup can be up to four times. Also, the working sets chosen by aSMO are more “flexible” than the ones chosen by gSMO. Indeed, each iteration of gSMO that updates the variables $\alpha_s, \alpha_t, \beta_s$ and β_t , can be done with two iterations of aSMO. If instead of updating these four variables we really need to update only three or two of them then aSMO will do that in a single cheap iteration. However, in the latter case, gSMO will spend some time in trying to update the variables that are not really needed to be updated. Also, aSMO is capable of updating three variables with a single cheap iteration. gSMO will do that with two expensive iterations.

We found that the speed of LOQO comes at the cost of the generalization error of its solution. Table 2 and

³See neural.cs.nthu.edu.tw/jang/dataset/mg/mg.c

⁴Available at abel.ee.ucla.edu/cvxopt/

⁵Available at www.princeton.edu/~rvdb/loqo/LOQO.html

⁶Available at www.stanford.edu/~yyye/matlab.html

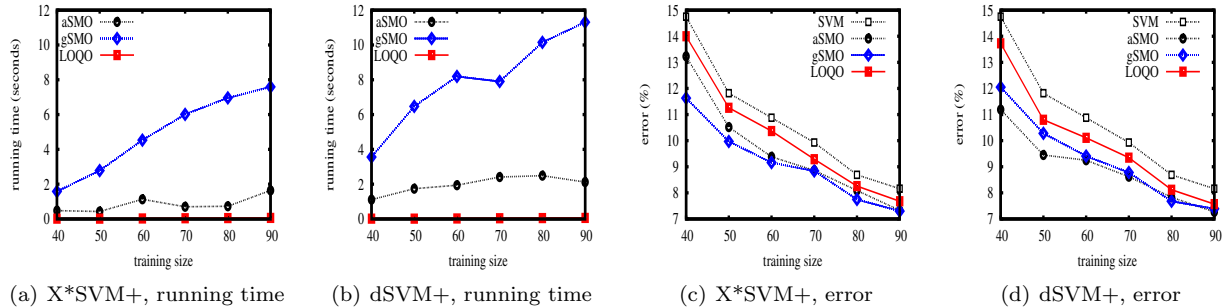


Figure 1: Comparison of SVM, aSMO, gSMO and LOQO on MNIST dataset

Figures 1(c)-(d) compare the errors of SVM, aSMO and LOQO on Mackey-Glass and MNIST datasets. aSMO and LOQO are competitive on Mackey-Glass dataset. However, on MNIST dataset aSMO consistently, and most of the time significantly, outperforms LOQO. We tried to change the stopping criterion of LOQO, to allow it to run more time. But even in this case its final generalization error was about the same as reported in Table 2 and Figures 1(c)-(d).

We allowed aSMO and gSMO to perform at most $2 \cdot 10^6$ iterations, unless they converge earlier. We observed that beyond this number of iterations in most of the experiments the improvement of generalization error is not significant. The difference between the generalization errors of alternating and gSMO shows that within a finite number of iterations aSMO finds better solution than gSMO.

Based on the our experiments we conclude that aSMO is currently the best implementation of SVM+ in terms of the tradeoff between the accuracy and the running time.

8 Conclusions

We presented two algorithms for solving the optimization problem of SVM+. To the best of our knowledge, these are the only currently existing ad-hoc algorithms for solving SVM+. We observed that the generic interior point optimizers are feasible only for very small problems. Moreover, the generalization error of their solution may be significantly worse than the generalization error of the solution generated by our algorithms.

The presented algorithms are valid for any kernel in X and X^* . Similarly to SVM [6], it is probably possible to develop specialized optimization algorithms for SVM+ when one or both kernels are linear. Finally, is it possible to solve SVM+ by directly optimizing the primal problem (1.2)?

Acknowledgements We thank Leon Bottou for pointing us the convergence results of [1].

References

- [1] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *JMLR*, 6:1579–1619, 2005.
- [2] L. Bottou and C.-J. Lin. Support Vector Machine solvers. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 1–27. MIT Press, 2007.
- [3] C.-C. Chang and C.-J. Lin. LIBSVM: a library for Support Vector Machines, 2001. Available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [5] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *JMLR*, 6:243–264, 2005.
- [6] T. Joachims. Training linear SVMs in linear time. In *KDD*, pages 217–226, 2006.
- [7] S.S. Keerthi and E.G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1-3):351–360, 2002.
- [8] M.C. Mackey and J. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.
- [9] 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*, pages 185–208. MIT Press, 1999.
- [10] N. Takahashi and T. Nishi. Rigorous proof of termination of SMO algorithm for Support Vector Machines. *IEEE Trans. on Neural Networks*, 16(3):774–776, 2005.
- [11] V. Vapnik. *Estimation of dependencies based on empirical data*. Springer-Verlag, 2nd edition, 2006.
- [12] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.
- [13] V. Vapnik, A. Vashist, and N. Pavlovich. Learning using hidden information: Master class learning. In *Proceedings of NATO workshop on Mining Massive Data Sets for Security*, pages 3–14. 2008.