2016 IEEE Conference on Computer Vision and Pattern Recognition

Fast Algorithms for Linear and Kernel SVM+

Wen Li[1] Dengxin Dai[1] Mingkui Tan[2] Dong Xu[3] Luc Van Gool[1,4]
[1]Computer Vision Laboratory, ETH Zürich, Switzerland
[2]School of Computer Science, University of Adelaide, Australia
[3]School of Electrical and Information Engineering, University of Sydney, Australia
[4]VISICS, ESAT/PSI, KU Leuven, Belgium

Abstract

The SVM+ approach has shown excellent performance in visual recognition tasks for exploiting privileged information in the training data. In this paper, we propose two efficient algorithms for solving the linear and kernel SVM+, respectively. For linear SVM+, we absorb the bias term into the weight vector, and formulate a new optimization problem with simpler constraints in the dual form. Then, we develop an efficient dual coordinate descent algorithm to solve the new optimization problem. For kernel SVM+, we further apply the ℓ_2-loss, which leads to a simpler optimization problem in the dual form with only half of dual variables when compared with the dual form of the original SVM+ method. More interestingly, we show that our new dual problem can be efficiently solved by using the SMO algorithm of the one-class SVM problem. Comprehensive experiments on three datasets clearly demonstrate that our proposed algorithms achieve significant speed-up than the state-of-the-art solvers for linear and kernel SVM+.

1. Introduction

Many computer vision tasks contain privileged information that only exists in the training data, and not available during the test stage. For example, the training images of many datasets for image recognition are annotated with privileged information such as *attributes*, *object bounding boxes*, *textual descriptions*, *depth information*. Although the raw test images in the real-world applications are not associated with such information, it has been demonstrated that such information is useful for learning classifiers with better recognition performance [5, 11, 20, 29, 32, 33, 35].

This problem is known as the *Learning Using Privileged Information (LUPI)* problem [32]. Different from the traditional learning paradigm, where the training data and the test data have the same representation, LUPI leverages training data containing additional information that is only available during the training process, not in the testing process. Such additional information in the training data is referred to as privileged information or hidden information.

Following the LUPI paradigm, Vapnik and Vashist [32] proposed an SVM-like algorithm called *SVM+*, in which they replace the slack variables in the standard SVM with a slack function defined in the privileged feature space. Through the slack function, the additional privileged information is used to model the loss function, which guides the hyperplane learning in the main feature space. In contrast, the slack variables in the standard SVM are only constrained to non-negative values, which is often less effective than the slack function in SVM+.

Although SVM+ can be formulated as a quadratic programming problem in the dual form similarly as the standard SVM, it is still non-trivial to efficiently solve it, because the introduction of the slack function leads to more constraints, and makes the number of dual variables doubled. While an SMO-style algorithm was developed in [26], the working set selection method is complicated, and the algorithm is also slow in practice. Moreover, it is unclear how to apply it to linear SVM+ without calculating the kernel matrix, which is becoming more crucial, due to rapidly increasing data in real-world applications.

In this paper, we propose two efficient algorithms for solving linear SVM+ and kernel SVM+, respectively. In particular, inspired by linear SVM [16], we augment the feature vector with an additional constant element, and absorb the bias term in the decision function into the weight vector, which leads to a dual form with simpler constraints. The new linear SVM+ formulation can be efficiently solved by using the dual coordinate descent method, in a similar way to linear SVM. For kernel SVM+, we further propose to apply the ℓ_2-loss, which leads to a simpler optimization problem in the dual form with only half of dual variables when compared with the original SVM+. More interestingly, we show that the resultant dual form is in analogy to one-class SVM, which can thus be efficiently solved using the efficient sequential minimal optimization (SMO) algorithm [28] with the existing state-of-the-art SVM solvers

1063-6919/16 $31.00 © 2016 IEEE
DOI 10.1109/CVPR.2016.248

such as LIBSVM [10, 1].

We implement our algorithms for linear and kernel SVM+ based on LIBLINEAR [9] and LIBSVM [1], respectively. We conduct extensive experiments on three tasks: digit recognition on the MNIST+ dataset [32], scene recognition on the Scene-15 dataset [22], and web image retrieval on the NUS-WIDE dataset [4]. The results demonstrate that our proposed algorithms achieve significant speed-up than the state-of-the-art solvers for solving the linear and kernel SVM+ problems.

2. Related Work

Learning Using Privileged Information (LUPI), as a new learning paradigm, was first proposed by Vapnik and Vashist [32], therein with the SVM+ algorithm. After that, many variants of SVM+ have been proposed for solving different tasks [24, 12, 34, 29, 33, 23]. In [24], Liang and Cherkassky developed a multi-task learning approach based on SVM+. In [17], a multi-task multi-class extension of SVM+ was proposed. Fouad *et al.* [12] designed a two-step approach for metric learning, and Xu *et al.* [34] formulated a convex formulation for metric learning using privileged information based on the information theory metric learning (ITML) method. Sharmanska *et al.* [29] proposed the Rank Transfer method for utilizing privileged information, and demonstrated the effectiveness of privileged information in various computer vision tasks. In [23], Li *et al.* extended SVM+ to the multi-instance learning scenario for image retrieval and object recognition by learning using web data. In [33], Wang *et al.* proposed a classifier learning algorithm for utilizing privileged information. In [11], Feyereisl *et al.* extended structure SVM to exploiting privileged information for object localization.

Vapnik and Vashist also showed that SVM+ has a faster convergence rate than the standard SVM method under certain conditions [32]. A more thorough theoretic study of SVM+ can be found in [27]. In [31], two mechanisms for LUPI are further explained. Lapin *et al.* [21] also discovered the relationship between SVM+ and weighted SVM, while Li *et al.* [23] discussed the connection of the unsupervised domain adaptation method and SVM+.

One closely related work is [26], in which an SMO-style algorithm was developed for the SVM+ problem. However, as the two sets of dual variables introduced for the constraints related to the main and privileged features in the primal form are tangled together in the constraints of dual problem, it is non-trivial to design the working set selection method for the SMO algorithm, and leads to a complicated algorithm that is less efficient in practice. Moreover, it is also unclear that how to apply the the SMO-style algorithm for solving the linear SVM+ problem without calculating the kernel matrix. In contrast, in this paper, by absorbing the bias term into the weight vector of the SVM+ classifier, we arrive at an optimization problem with simpler constraints in the dual form. Thus, the conventional dual coordinate descent algorithm can be applied to efficiently solve linear SVM+. Moreover, for the SVM+ form, we further apply the ℓ_2-loss and obtain a smaller dual problem with only a half of dual variables of the original SVM+ formulation. The new dual form shares a similar formulation with one-class SVM, and thus can be solved by using the SMO algorithm implemented in the existing SVM solvers such as LIBSVM [1] without developing any new specific SMO algorithm. We demonstrate that our proposed two algorithms are more efficient than the SMO-style algorithm in [26].

We are also aware that there are some works proposed for solving the tasks in which training and test data contains different information, which are less related to SVM+ [2, 3, 5, 7, 13, 14, 15, 20, 30, 35].

3. Learning Using Privileged Information

In the following, we denote a vector (*resp.*, matrix) with a lower (*resp.*, upper) case letter in boldface. For example, $\mathbf{a}$ represents a vector, and $\mathbf{A}$ a matrix. The transpose of a vector or matrix is denoted by using the superscript $'$. We use $\mathbf{0}_n, \mathbf{1}_n \in \mathbb{R}^n$ to represent the column vector with n zeros and ones, respectively. We also simply use $\mathbf{0}$ and $\mathbf{1}$ instead of $\mathbf{0}_n$ and $\mathbf{1}_n$, when the dimensionality is obvious. Moreover, $\mathbf{a} \circ \mathbf{b}$ (*resp.*, $\mathbf{A} \circ \mathbf{B}$) denotes the element-wise product between two vectors (*resp.*, matrices).

In the Learning Using Privileged Information (LUPI) paradigm [32], the training samples contain additional information that is not available in the test stage. Formally, we represent the training data as $\{(\mathbf{x}_i, \tilde{\mathbf{x}}_i, y_i) | i = 1, \ldots, n\}$, where n is the total number of training samples, $\mathbf{x}_i \in \mathbb{R}^D$ is the main feature vector of the i-th training sample with D being the feature dimensionality, $\tilde{\mathbf{x}} \in \mathbb{R}^{\tilde{D}}$ is the corresponding privileged feature vector with $\tilde{D}$ being its dimension. The goal of LUPI is to learn a decision function $f(\mathbf{x})$ for classifying any test sample $\mathbf{x} \in \mathbb{R}^D$ in the main feature space.

In [32], an SVM based approach called *SVM+* was proposed. Similar to SVM, the decision function in SVM+ is represented as $f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}) + b$, where $\phi(\cdot)$ is a feature mapping induced by the kernel on training data, $\mathbf{w}$ is the weight vector, and b is the bias term. The objective of SVM+ can be written as,

$$\min_{\tilde{\mathbf{w}}, \tilde{b}, \mathbf{w}, b} \ \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + C\sum_{i=1}^{n} \xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) \quad (1)$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\phi(\mathbf{x}_i) + b) \geq 1 - \xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)), \quad (2)$$

$$\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) \geq 0, \quad (3)$$

where $\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) = \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}) + \tilde{b}$ is the slack function defined in the privileged feature space, $\psi(\cdot)$ is the feature mapping function for the privileged features, and $\tilde{\mathbf{w}}$ and $\tilde{b}$

are respectively the weight vector and bias term for the slack function. It can be observed that SVM+ replaces the slack variables in the standard SVM formulation with the slack function $\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i))$. As a result, the loss occurred by the training samples $\mathbf{x}_i$ can be regularized by the privileged information $\tilde{\mathbf{x}}_i$, therefore the hyperplane of SVM+ classifier can be tuned by using the privileged information during the training process.

Let us introduce two sets of dual valuables $\{\alpha_i|_{i=1}^n\}$ and $\{\zeta_i|_{i=1}^n\}$ for the constraints in (2) and (3), respectively, and also denote two vectors $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]' \in \mathbb{R}^n$ and $\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_n]' \in \mathbb{R}^n$. We arrive at the dual form of SVM+,

$$\begin{aligned} \min_{(\boldsymbol{\zeta},\boldsymbol{\alpha})\in\mathcal{A}} \quad & \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})'\mathbf{K}(\boldsymbol{\alpha}\circ\mathbf{y}) - \mathbf{1}'\boldsymbol{\alpha} \qquad (4) \\ & + \frac{1}{2\gamma}(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1})'\tilde{\mathbf{K}}(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1}), \end{aligned}$$

where $\mathcal{A} = \{(\boldsymbol{\zeta}, \boldsymbol{\alpha})|\mathbf{y}'\boldsymbol{\alpha} = 0, \mathbf{1}'(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1}) = 0, \boldsymbol{\alpha} \geq 0, \boldsymbol{\zeta} \geq 0\}$ is the feasible set of $(\boldsymbol{\alpha}, \boldsymbol{\zeta})$, $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the kernel matrix based on the main training features with each element being $K_{ij} = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$, and $\tilde{\mathbf{K}} \in \mathbb{R}^{n\times n}$ is the kernel matrix based on the privileged features with each element being $\tilde{K}_{ij} = \psi(\tilde{\mathbf{x}}_i)'\psi(\tilde{\mathbf{x}}_j)$. After solving the above problem, the weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be reconstructed based on the Karush-Kuhn-Tucker (KKT) conditions as,

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i), \qquad (5) \\ \tilde{\mathbf{w}} &= \frac{1}{\gamma}\sum_{i=1}^n (\alpha_i + \zeta_i - C)\psi(\tilde{\mathbf{x}}_i). \qquad (6) \end{aligned}$$

Although the dual problem in (4) is a quadratic programming problem, solving it with the existing optimization toolboxes is certainly undesirable for real-world visual recognition tasks. However, it is non-trivial to develop efficient algorithm like the SMO algorithm for solving the standard SVM. The main difficulty comes from the constraint $\mathbf{1}'(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1}) = 0$, in which two sets of dual variables are tangled together, which makes the working set selection method become complicated [26].

4. Dual Coordinate Descent Algorithm for Solving Linear SVM+

In this section, we present the dual coordinate descent method for solving the linear SVM+. In particular, we absorb the bias term into the weight vector $\mathbf{w}$ by appending a constant entry of 1 to each feature vector, *i.e.*, $\mathbf{x} \leftarrow [\mathbf{x}', 1]'$, and $\mathbf{w} \leftarrow [\mathbf{w}', b]'$ for the main features, and $\tilde{\mathbf{x}} \leftarrow [\tilde{\mathbf{x}}', 1]'$, and $\tilde{\mathbf{w}} \leftarrow [\tilde{\mathbf{w}}', \tilde{b}]'$ for privileged information. For ease of presentation, we still use $\mathbf{w}$ and $\mathbf{x}$ (*resp.*, $\tilde{\mathbf{w}}$, $\tilde{\mathbf{x}}$) to represent the weight vector and feature vector for the main (*resp.*, privileged) features.

Now, the objective of linear SVM+ can be reformulated as follows,

$$\begin{aligned} \min_{\tilde{\mathbf{w}},\mathbf{w}} \quad & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + C\sum_{i=1}^n \tilde{\mathbf{w}}'\tilde{\mathbf{x}} \qquad (7) \\ \text{s.t.} \quad & y_i\mathbf{w}'\mathbf{x}_i \geq 1 - \tilde{\mathbf{w}}'\tilde{\mathbf{x}}, \\ & \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i \geq 0, \end{aligned}$$

and its dual form can be written as,

$$\begin{aligned} \min_{(\boldsymbol{\zeta},\boldsymbol{\alpha})\in\mathcal{A}} \quad & \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})'\mathbf{K}(\boldsymbol{\alpha}\circ\mathbf{y}) - \mathbf{1}'\boldsymbol{\alpha} \qquad (8) \\ & + \frac{1}{2\gamma}(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1})'\tilde{\mathbf{K}}(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1}), \end{aligned}$$

where the feasible set becomes $\mathcal{A} = \{(\boldsymbol{\zeta}, \boldsymbol{\alpha})|\boldsymbol{\alpha} \geq 0, \boldsymbol{\zeta} \geq 0\}$, $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the linear kernel matrix of main features defined as $K_{ij} = \mathbf{x}_i'\mathbf{x}_j$, and $\tilde{\mathbf{K}} \in \mathbb{R}^{n\times n}$ is the linear kernel matrix of privileged features defined as $\tilde{K}_{ij} = \tilde{\mathbf{x}}_i'\tilde{\mathbf{x}}_j'$. Compared to the dual form of the original SVM+ formulation (4), the objective function remains the same, but we do not have the constraints $\mathbf{y}'\boldsymbol{\alpha} = 0, \mathbf{1}'(\boldsymbol{\alpha}+\boldsymbol{\zeta}-C\mathbf{1}) = 0$ in (8), after absorbing of the bias terms in (7). The weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be represented using the dual variables in the same way as in (5) and (6) without using the feature mapping functions (*i.e.*, we can set $\phi(\mathbf{x}_i) = \mathbf{x}_i$ and $\psi(\tilde{\mathbf{x}}_i) = \tilde{\mathbf{x}}_i$.

Let us define $\mathbf{Q} = \begin{bmatrix} \mathbf{K}\circ(\mathbf{y}\mathbf{y}') + \frac{1}{\gamma}\tilde{\mathbf{K}} & \frac{1}{\gamma}\tilde{\mathbf{K}} \\ \frac{1}{\gamma}\tilde{\mathbf{K}} & \frac{1}{\gamma}\tilde{\mathbf{K}} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$, $\boldsymbol{\beta} = [\boldsymbol{\alpha}', \boldsymbol{\zeta}']' \in \mathbb{R}^{2n}$, $\mathbf{e} = [(\mathbf{1} + \frac{C}{\gamma}\tilde{\mathbf{K}}\mathbf{1})', (\frac{C}{\gamma}\tilde{\mathbf{K}}\mathbf{1})']' \in \mathbb{R}^{2n}$, the problem in (8) can be rewritten in a more concise form as,

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}'\mathbf{Q}\boldsymbol{\beta} - \mathbf{e}'\boldsymbol{\beta} \qquad (9) \\ \text{s.t.} \quad & \boldsymbol{\beta} \geq 0. \end{aligned}$$

Now we develop a coordinate descent algorithm for the problem in (9) similarly as that for linear SVM [16]. In the dual coordinate descent algorithm, we iteratively update one entry of $\boldsymbol{\beta}$ each time. This process is repeated until the stopping criterion is reached.

In particular, let us denote $f(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}'\mathbf{Q}\boldsymbol{\beta} - \mathbf{e}'\boldsymbol{\beta}$, and we propose to update the i-th entry as $\beta_i \leftarrow \beta_i + d$, where d is a scalar variable that needs to be solved. By defining $\boldsymbol{\delta}^{(i)} = [0, \ldots, 0, 1, 0, \ldots, 0]$ as the vector with all zeros except the i-th entry being one, the subproblem for solving d can be formulated as

$$\min_d f(\boldsymbol{\beta} + d\boldsymbol{\delta}^{(i)}) \quad \text{s.t. } \boldsymbol{\beta} + d\boldsymbol{\delta}^{(i)} \geq 0, \qquad (10)$$

which has an analytic solution as follows,

$$d = \max(-\beta_i, -\frac{\nabla_i f(\boldsymbol{\beta})}{Q_{ii}}) \qquad (11)$$

where Q_{ii} is the (i,i)-th entry of the matrix $\mathbf{Q}$ in (9), and $\nabla_i f(\boldsymbol{\beta})$ is the gradient of $f(\boldsymbol{\beta})$ *w.r.t.* β_i.

Now we discuss how to efficiently calculate the right-hand side of (11). The scalar Q_{ii} can be pre-computed efficiently, so the major problem is the calculation of $\nabla_i f(\boldsymbol{\beta})$. In particular, the gradient of $f(\boldsymbol{\beta})$ can be written as $\nabla f(\boldsymbol{\beta}) = \mathbf{Q}\boldsymbol{\beta} - \mathbf{e}$. Then, for any given index i, the gradient of $f(\boldsymbol{\beta})$ *w.r.t.* β_i can be calculate as,

$$\nabla_i f(\boldsymbol{\beta}) = (\mathbf{Q}\boldsymbol{\beta})_i - e_i = \sum_{j=1}^{2n} Q_{ij}\beta_j - e_i, \tag{12}$$

where Q_{ij} is the (i,j)-th element of the matrix $\mathbf{Q}$, and e_i is the i-th element of the vector $\mathbf{e}$.

We discuss the detailed calculations in two cases. Let us denote a matrix $\mathbf{H} = \mathbf{K} \circ \mathbf{y}\mathbf{y}' \in \mathbb{R}^{n \times n}$. For $\forall i = 1, \ldots, n$, we have

$$\sum_{j=1}^{2n} Q_{ij}\beta_j = \sum_{j=1}^{n} H_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\zeta_j$$

where H_{ij} is the (i,j)-th element of the matrix $\mathbf{H}$. We also have $e_i = 1 + \frac{C}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}$. By applying the KKT conditions *w.r.t.* $\mathbf{w}$ and $\tilde{\mathbf{w}}$ in (5) and (6), the gradient of f *w.r.t.* β_i in (12) can be written as,

$$\nabla_i f(\boldsymbol{\beta})_i = y_i \mathbf{w}'\mathbf{x}_i - 1 + \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i, \quad \forall 1 \leq i \leq n \tag{13}$$

which takes $O(D + \tilde{D})$ time complexity, and it can be further speed-up if the feature vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are sparse.

Similarly, for $\forall i = n+1, \ldots, 2n$, we have

$$\sum_{j=1}^{2n} Q_{ij}\beta_j \quad = \quad \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\zeta_j,$$

and $e_i = \frac{C}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}$. By using the KKT condition for $\tilde{\mathbf{w}}$ in (6), the gradient in (12) can be written as,

$$\nabla_i f(\boldsymbol{\beta})_i = \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i, \quad \forall n+1 \leq i \leq 2n, \tag{14}$$

which takes $O(\tilde{D})$ time complexity only, and can also be further speed-up if the feature vector $\tilde{\mathbf{x}}$ is sparse. This completes the calculation of (11).

After solving d, we update the i-th dual variable β_i by using $\beta_i \leftarrow \beta_i + d$. Considering only one dual variable is modified each time, the weight vectors can be updated efficiently at each iteration. Note that we have $\alpha_i = \beta_i$ when $1 \leq i \leq n$, and $\zeta_{i-n} = \beta_i$, when $n+1 \leq i \leq n$. Based on (5) and (6), the updating rules for $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be written as

$$\mathbf{w} \quad \leftarrow \quad \mathbf{w} + d y_i \mathbf{x}_i, \quad \text{if } 1 \leq i \leq n \tag{15}$$

$$\tilde{\mathbf{w}} \quad \leftarrow \quad \tilde{\mathbf{w}} + \frac{1}{\gamma} d\tilde{\mathbf{x}}_i, \quad \text{if } 1 \leq i \leq 2n \tag{16}$$

Algorithm 1 Dual coordinate descent algorithm for solving the linear SVM+ problem in (7)

```
Input: $\{(\mathbf{x}_i, \tilde{\mathbf{x}}_i, y_i)|_{i=1}^n\}$, $C$, and $\gamma$.
Initialize $\mathbf{w} = \mathbf{0}$, and $\tilde{\mathbf{w}} = -\frac{C}{\gamma}\sum_{i=1}^n \tilde{\mathbf{x}}_i$.
Set $Q_{ii} = \mathbf{x}_i'\mathbf{x}_i$ for $1 \leq i \leq n$, and $Q_{ii} = \tilde{\mathbf{x}}_i'\tilde{\mathbf{x}}_i$ for $n+1 \leq i \leq 2n$.
repeat
    Randomly pick an index $i$.
    if $1 \leq i \leq n$ then
        Calculate $\nabla_i f(\boldsymbol{\beta})$ using (13).
    else
        Calculate $\nabla_i f(\boldsymbol{\beta})$ using (14).
    end if
    Calculate $d$ using (11) based on $Q_{ii}$ and $\nabla_i f(\boldsymbol{\beta})$.
    if $1 \leq i \leq n$ then
        Update $\mathbf{w}$ using (15).
    end if
    Update $\tilde{\mathbf{w}}$ using (16).
until The convergence criterion is reached.
Output: Weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$.
```

We summarize our dual coordinate descent algorithm for solving linear SVM+ in Algorithm 1. We first initialize the weight vectors as $\mathbf{w} = \mathbf{0}_D$, and $\tilde{\mathbf{w}} = -\frac{C}{\gamma}\sum_{i=1}^n \tilde{\mathbf{x}}_i$ (*i.e.*, the solution of $\mathbf{w}$ and $\tilde{\mathbf{w}}$ when $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\zeta} = \mathbf{0}$). Then, each time, we randomly pick up an index i, and solve d (*i.e.*, the change of β_i) using (11). In particular, when $1 \leq i \leq n$, we calculate the gradient $\nabla_i f(\boldsymbol{\beta})$ using (13), and update the weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ using (15) and (16), respectively; when $n+1 \leq i \leq 2n$, we calculate the gradient $\nabla_i f(\boldsymbol{\beta})$ using (14) and update the weight vector $\tilde{\mathbf{w}}$ only using (16). The above process is repeated until the objective converges. Actually, the problem in (9) can be treated as a special form of the linear SVM problem discussed in [16], so the convergence of our algorithm follows that of the dual coordinate descent algorithm for linear SVM. We implement our algorithm based on the LIBLINEAR software [9].

5. SMO Algorithm for Solving Kernel SVM+

In this section, we develop an efficient algorithm for solving kernel SVM+ based on the ρ-SVM formulation. Similar to linear SVM+, we also augment the feature vector in the nonlinear feature space, so we can absorb the bias term into the weight vector, *i.e.*, $\phi(\mathbf{x}_i) \leftarrow [\phi(\mathbf{x}_i)', 1]'$ and $\mathbf{w} \leftarrow [\mathbf{w}', b]'$ for the main features (*resp.*, $\psi(\tilde{\mathbf{x}}_i) \leftarrow [\psi(\tilde{\mathbf{x}}_i)', 1]'$ and $\tilde{\mathbf{w}} \leftarrow [\tilde{\mathbf{w}}', \tilde{b}]'$ for the privileged features)[1]. For ease of presentation, we still use $\phi(\mathbf{x}_i)$ and $\mathbf{w}$ to de-

[1] Usually we do not know the explicit form of $\phi(\cdot)$, but the inner product of the augmented features can be simply calculated by adding one, *i.e.*, $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) \leftarrow \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) + 1$, so the kernel matrix based on the augmented features can be calculated by $\mathbf{K} \leftarrow \mathbf{K} + \mathbf{1}\mathbf{1}'$. The same approach can be applied to the privileged features.

note the nonlinear feature and weight vector for the main features (*resp.*, $\psi(\tilde{\mathbf{x}}_i)$ and $\tilde{\mathbf{w}}$ for privileged features) below.

Specifically, after using the augmented features, the decision function can be represented as $f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x})$. We use the squared hinge loss for ρ-SVM, which leads to the ℓ_2 loss ρ-SVM formulation as follows,

$$\begin{aligned} \min_{\mathbf{w},b,\rho} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{n} \xi_i^2 - \rho \qquad (17) \\ \text{s.t.} \quad & y_i(\mathbf{w}'\phi(\mathbf{x}_i)) \geq \rho - \xi_i. \end{aligned}$$

By replacing each slack variable ξ_i with the slack function $\xi(\tilde{\mathbf{x}}_i) = \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)$, we arrive at the primal form of ℓ_2-SVM+ as follows,

$$\begin{aligned} \min_{\tilde{\mathbf{w}},\mathbf{w},\rho} \quad & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + \frac{1}{2}C\sum_{i=1}^{n} \left(\tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right)^2 - \rho \\ \text{s.t.} \quad & y_i(\mathbf{w}'\phi(\mathbf{x}_i)) \geq \rho - \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i). \qquad (18) \end{aligned}$$

Now we derive the dual form of our ℓ_2-SVM+ formulation in (18). By introducing dual variables $\alpha_1, \ldots, \alpha_n$ for the constraints in (18), we write its Lagrangian as follows,

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + \frac{1}{2}C\sum_{i=1}^{n} \left(\tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right)^2 - \rho \quad (19) \\ & - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}'\phi(\mathbf{x}_i)) - \rho + \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right), \end{aligned}$$

Let us denote a vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]'$. By setting the derivatives of (19) *w.r.t.* to the primal variables $\mathbf{w}, \tilde{\mathbf{w}}, \rho$ to zeros, we obtain the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ as well as two KKT conditions for $\mathbf{w}$ and $\tilde{\mathbf{w}}$ as,

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i), \qquad (20) \\ \tilde{\mathbf{w}} &= \sum_{i=1}^{n} \alpha_i \left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1} \psi(\tilde{\mathbf{x}}_i), \qquad (21) \end{aligned}$$

where $\mathbf{P} = [\psi(\tilde{\mathbf{x}}_1), \ldots, \psi(\tilde{\mathbf{x}}_n)]$ is the data matrix of privileged features in the nonlinear feature space.

Substituting the two equations in (20) and (21) into the Lagrangian in (19), we arrive at the dual form of (18),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}'(\mathbf{H}+\mathbf{G})\boldsymbol{\alpha} \qquad (22) \\ \text{s.t.} \quad & \mathbf{1}'\boldsymbol{\alpha} = 1, \quad \boldsymbol{\alpha} \geq 0, \end{aligned}$$

where $\mathbf{G} = \mathbf{P}'\left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1}\mathbf{P}$, $\mathbf{H} = \mathbf{K} \circ (\mathbf{y}\mathbf{y}')$, and $\mathbf{K}$ is the kernel matrix of augmented main features with $K_{ij} = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ being its (i,j)-th element. Based on the equation $\left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1}\mathbf{P} = \mathbf{P}\left(\gamma\mathbf{I} + C\mathbf{P}'\mathbf{P}\right)^{-1}$, we have $\mathbf{G} = \mathbf{P}'\mathbf{P}\left(\gamma\mathbf{I} + C\mathbf{P}'\mathbf{P}\right)^{-1} = \tilde{\mathbf{K}}\left(\gamma\mathbf{I} + C\tilde{\mathbf{K}}\right)^{-1}$, where $\tilde{\mathbf{K}}$ is the kernel matrix of augmented privileged features with $\tilde{K}_{ij} = \psi(\mathbf{x}_i)'\psi(\mathbf{x}_j)$ being its (i,j)-th element. By further applying the Woodbury Identity, we obtain $\mathbf{G} = \frac{1}{C}\mathbf{I} - \frac{1}{C}\left(\mathbf{I} + \frac{C}{\gamma}\tilde{\mathbf{K}}\right)^{-1}$.

Algorithm 2 Algorithm for solving the ℓ_2-SVM+ problem in (18)

```
Input: $\mathbf{K}, \tilde{\mathbf{K}} \in \mathbb{R}^{n \times n}$, $C$, and $\gamma$.
Calculate $\mathbf{G} = \frac{1}{C}\mathbf{I} - \frac{1}{C}\left(\mathbf{I} + \frac{C}{\gamma}\tilde{\mathbf{K}}\right)^{-1}$.
Set $\mathbf{Q} = \mathbf{H} + \mathbf{G}$, and $\nu = 1/n$.
Obtain $\boldsymbol{\alpha}$ by solving the problem in (23) with the one-class SVM solver in LIBSVM.
Output: Dual variable vector $\boldsymbol{\alpha}$.
```

Note the problem in (22) is a quadratic programming problem with the number of dual variables being n, which is only half number of variables in the dual form of the original SVM+ method in (4). Moreover, the constraints in (22) are also simpler, so it can be solved by using the efficient SMO algorithm. Actually, the problem in (22) shares a similar form with one-class SVM in the LIBSVM software [1]. In particular, we write the dual form of one-class SVM as follows (Eqn. (8) in [1]),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}'\mathbf{Q}\boldsymbol{\alpha} \qquad (23) \\ \text{s.t.} \quad & \mathbf{1}'\boldsymbol{\alpha} = \nu n, \quad 0 \leq \boldsymbol{\alpha} \leq 1, \end{aligned}$$

where $\mathbf{Q}$ is the kernel matrix in one-class SVM, ν is a predefined parameter, n is total the number of training samples. Note the constraints $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\mathbf{1}'\boldsymbol{\alpha} = 1$ in (22) imply $0 \leq \boldsymbol{\alpha} \leq 1$. By setting $\mathbf{Q} = (\mathbf{H} + \mathbf{G})$ and $\nu = \frac{1}{n}$, the dual problem of ℓ_2-SVM+ in (22) can be converted to the optimization problem in (23). Therefore, we ignore the details of the SMO algorithm here, and employ the SMO implementation in LIBSVM to solve our ℓ_2-SVM+ problem.

The detailed algorithm for solving ℓ_2-SVM+ is described in Algorithm 2. We first calculate the matrix $\mathbf{G}$ using the kernel matrix of privileged features. Then we call the one-SVM solver in LIBSVM to obtain the dual variable vector $\boldsymbol{\alpha}$. Finally, the decision function can be written as $\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)'\phi(\mathbf{x})$, where $\mathbf{x}$ is the test sample, and $\phi(\cdot)$ is the augmented nonlinear feature vector.

Although in our algorithm a matrix inverse operator is involved when calculating the matrix $\mathbf{G}$, the size of $\mathbf{G}$ is only $n \times n$, and the subsequent QP problem is also only with n dual variables. With the excellent implementation of matrix inverse function such as that in MATLAB, and the SMO implementation in LIBSVM, our algorithm is much faster than the existing SVM+ solver [26] in practice.

Moreover, in some scenarios such as multi-instance learning using privileged information [25], it often needs to iteratively solve the SVM+ problem. In this case, after calculating the matrix inverse, we only need to iteratively solve the one-class SVM, which is much more efficient than the traditional method which needs to iteratively optimize a QP problem for solving the SVM+ problem (see Section 6.2).

6. Experiments

In this section, we evaluate the efficiency of our proposed two algorithms for linear and kernel SVM+, and compare them with the existing SVM+ solvers for the image classification and web image retrieval tasks. Considering we solve two variants of SVM+ instead of the original SVM+ formulation, we also report the classification/retrieval performance for comparisons.

6.1. Image Classification

We conduct the experiments for two image recognition tasks: digit recognition and scene recognition.

Datasets: In the digit classification task, Vapnik and Vashist [32] have shown that the additional textual descriptions in training data are helpful for learning a better classifier. We employ the benchmark MNIST+ dataset used in [32, 26] for classifying the images as two digits "5" and "8". The dataset is constructed by using a subset of the MNIST dataset containing the images of the digits "5" and "8". It is split into a training set of 100 images (50 images with digit "5", and 50 images with digit "8"), a validation set of $4,002$ images, and a test set of $1,866$ images. All the images are resized into 10×10 pixels, and the 100-d vector of raw pixels is used as the main feature vector for each image. Each training image is additionally supplied with a poetic description (see [32] for the examples), which is converted into a 21-d textual feature vector, and used as the privileged information in SVM+.

For the scene recognition task, we employ the Scene-15 dataset[22], which contains $4,485$ images of 15 different scenes. We split the data into three sets: 300 images as the training set, 40% images as the test set, and the rest as the validation set. We downsample all the images by factor 3, because small images are preferred in real-world applications for saving the storage of visual data at test time. The CNN features [19] are extracted with the CAFFE framework [18], which has shown good performance in various visual recognition tasks. During training, the CNN features extracted from the original images are treated as privileged information, and the CNN features extracted from the downsampled images as the main features. The experimental setting is inspired by [6], where high-resolution images yield superior performance than low-resolution images for standard vision tasks. PCA is applied on two types of features to obtain 100-d compact representations.

Baselines: In our experiments, we consider two settings, the linear case and the nonlinear case. In the linear case, there is no solver specifically designed for linear SVM+. So we mainly compare our method with the following two baselines,

- LIBLINEAR: The standard linear SVM without using privileged information implemented in LIBLINEAR [9]. We include it as a baseline for investigating the effectiveness of our linear SVM+ formulation.
- gSMO: The SMO-style algorithm for solving SVM+ in the dual form, which is proposed in [26]. We use the released C++ implementation from the authors.

Note that the gSMO method was designed for solving kernel SVM+, but it can still take the feature vectors as the input. So we also include it as a baseline for the linear case.

For the nonlinear case, the Gaussian kernel is used for all the methods, in which the bandwidth parameter is set as the mean of distances between all training samples by default. Besides the aforementioned gSMO method, we also compare our ℓ_2-SVM+ method with the existing state-of-the-art solvers of SVM+, and also include the standard SVM solved by LIBSVM as a baseline for investigating the effectiveness of SVM+.

- SVM: The standard SVM without using privileged information, whihc is solved by using LIBSVM [9]. We include it as a baseline for investigating the effectiveness of our ℓ_2-loss SVM+ formulation.
- CVX-SVM+: An implementation based on the CVX optimization toolbox provided in [24]. It directly solves the quadratic programming problem in (4) using the QP solver in CVX.
- MAT-SVM+: We additionally include the QP solver implemented in MATLAB R2014b as a baseline, and employ it to solve the QP problem in (4).

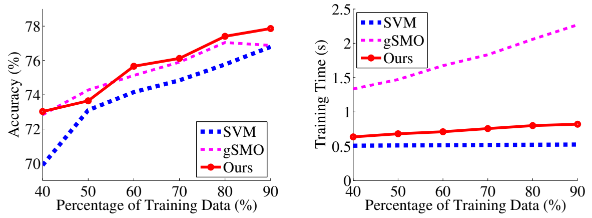
For all methods, the tradeoff parameters are determined based on the validation set. One-vs-all strategy is used on the Scene-15 dataset, which contains 15 classes. The classification accuracy on the test set is reported for performance evaluation. The training time of all methods is measured on a workstation with Intel i7-3770K CPU@3.50GHz.

6.1.1 Experiments on Linear SVM+

Experimental results: The classification accuracies and training time of all methods on two datasets are summarized in Table 1. In terms of the classification accuracy, we observe that both our linear SVM+ algorithm and gSMO achieve better results than the baseline linear SVM method, which demonstrates the effectiveness of exploiting the poetic description as privileged information for improving the image based digit recognition task. Our method achieves slight better results than the gSMO method, possibly because we use a new variant of SVM+, and our dual coordinate descent algorithm guarantees the optimal solution.

Table 1. Accuracies (%) and training time of all methods on the MNIST+ and Scene-15 datasets in linear case. Our results are highlighted in boldface.

		MNIST+		Scene-15	
		Accuracy	Time (ms)	Accuracy	Time (s)
SVM		81.73	0.6	77.56	0.76
SVM+	gSMO	84.35	72.1	78.05	2.52
	Ours	**84.62**	**9.5**	**78.10**	**0.87**



(a) (b)

Figure 1. The accuracies (Figure (a)) and the training time (Figure (b)) of different methods for solving linear SVM+ when using different number of training samples on the Scene-15 dataset.

In terms of the training time, both SVM+ algorithms are slower than the baseline linear SVM, because that more complicated objective functions are used to incorporate privileged information. Our new dual coordinate descent algorithm for linear SVM+ is much faster than gSMO, which generally takes only a bit more training time than the linear SVM as shown in Table 1.

Results using different number of training samples: We further take the Scene-15 dataset to investigate the accuracies and efficiency of different methods *w.r.t.* different number of training samples. Similarly as in [32, 26], we randomly sample $40\%, 50\%, 60\%, 70\%, 80\%$ and 90% training samples for learning SVM and SVM+ classifiers. The experiments are repeated for 10 times, and we plot the average accuracies and training time of different methods in Figure 1. While the accuracies of all methods become higher when the number of training data increases, our linear SVM+ algorithm consistently outperforms the linear SVM method, and generally achieves slight better performance than the gSMO method. In term of the training time, it can be observed that the training time of our method increases linearly *w.r.t.* the number of training samples, and is several times faster than the baseline gSMO method.

Convergence: As mentioned in Section 4, our algorithm can be treated as a special form of the linear SVM discussed in [16]. So it shares the similar convergence property as the dual coordinate descent algorithm for linear SVM. To verify the convergence of our algorithm, in Figure 2, we take the MNIST+ dataset as an example to plot the objective values of our algorithm when the number of iterations[2] increases. It can be observed that our algorithm converges well. The objective value decreases very fast within the first ten iterations, and continues to decrease as the number of iterations increases.

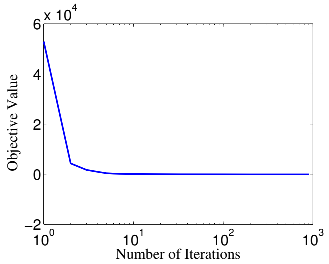


Figure 2. The objective of our coordinate descent algorithm for linear SVM+ on MNIST+ dataset.

6.1.2 Experiments on Kernel SVM+

Experimental results: We report the classification accuracies and training time of all methods on two datasets when using the Gaussian kernel in Table 2. On the MNIST+ dataset, we observe that the SVM+ algorithms again achieve better results than the baseline SVM algorithm, due to the utilizing of poetic descriptions as privileged information. However, on the Scene-15 dataset, gSMO and CVX-SVM+ are worse than the standard SVM method. Our ℓ_2-SVM+ algorithm achieves better result than the baseline algorithms, showing our new formulation for kernel SVM+ is effective for the scene classification problem on this dataset.

In terms of the training time, we observer that our newly proposed algorithm is the most efficient one among all SVM+ algorithms, and generally achieves order-of-magnitude speedup over the second fastest algorithm (MAT-SVM+ on the MNIST+ dataset, and gSMO on the Scene-15 dataset). The results demonstrate the efficiency of our new reformulation of kernel SVM+. With the reformulation, we convert it as a one-class SVM problem, and take advantage of the existing state-of-the-art SMO implementation in LIBSVM to solve it.

Results using different number of training samples: Similarly as for the linear case, in Figure 3, we take the Scene-15 dataset to plot the classification accuracies and training time of different methods by using different percentages of training data. The CVX-SVM+ method is not included when reporting the training time for better visualization. We observe that our ℓ_2-SVM+ algorithm outperforms other SVM+ algorithms in terms of both classification accuracy and efficiency when varying the number of training samples.

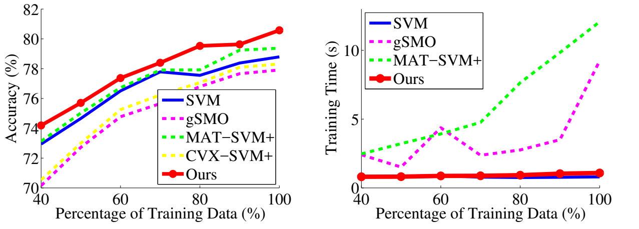
6.2. Web Image Retrieval

In this subsection, we demonstrate the advantage of our ℓ_2-SVM+ algorithm for solving the multiple instance

[2] Similarly as in LIBLINEAR, one iteration refers to that we pass all training samples once.

Table 2. Accuracies (%) and training time of all methods on the MNIST+ and Scene-15 dataset using the Gaussian kernel. Our results are highlighted in boldface.

		MNIST+		Scene-15	
		Acc.	Time (ms)	Acc.	Time (s)
SVM		92.34	0.7	78.79	0.80
SVM+	gSMO	92.77	78.8	77.91	9.23
	CVX-SVM+	93.14	767.7	78.32	47.16
	MAT-SVM+	93.14	54.9	79.38	12.07
	Ours	**93.15**	**1.3**	**80.57**	**1.08**



(a) (b)

Figure 3. The accuracies (Figure (a)) and training time (Figure (b)) of different methods for solving kernel SVM+ when using different number of training samples on the Scene-15 dataset.

learning using privileged information problem. We employ the mi-SVM-PI algorithm proposed in [25] to evaluate different SVM+ solvers. The mi-SVM-PI algorithm needs to iteratively solves the SVM+ problem, and simultaneously infer the labels for training samples under MIL constraints. We compare the mi-SVM-PI method based on our ℓ_2-SVM+ algorithm, with their original implementation based on MATALB.

Experimental setting: It has been shown that the textual descriptions associated with the web images are effective for learning better classifiers using multiple instance learning approaches [23, 25]. Following [23, 25], we employ the NUS-WIDE web image dataset, which contains $269,648$ web images crawled from the image sharing website *Flickr*. It is officially split into a training set of 60% images, and test set of 40% images. All the images are accompanied with textual tags provided by *Flickr* users. The test images are annotated for 81 concepts. Similarly as in [23], we extract the DeCAF$_6$ features [8], which leads to a 4096-dim feature vector for each web image. For the training data, we also extract a 200-dim term frequency feature from the associated textual tags of each image, and use it as the privileged information. 25 positive bags (*resp.*, negative bags) are constructed with each bag containing 15 relevant images (*resp.*, irrelevant images) as the training data for each concept. The Gaussian kernel is used for the visual features, and linear kernel is used for the textual features. The image retrieval performance is evaluated on the test set, in which only the visual features are extracted. The average precision based on the top-ranked 100 test images is used for performance evaluation, and the mean average precision (MAP) over 81 concepts is reported. Moreover, the training time of all methods over 81 concepts is reported for efficiency evaluation.

Table 3. MAP (%) and training time (s) of different methods on the NUS-WIDE dataset. Our results are highlighted in boldface.

		MAP	Time (s)
SVM		54.41	9.80
SVM+	MAT-SVM+	55.63	204.10
	Ours	**55.68**	**19.44**
mi-SVM-PI	MAT-SVM+	59.11	765.47
	Ours	**59.43**	**24.26**

Experimental results: We report the MAPs and training time of different methods in Table 3. From the table, we observe that the SVM+ methods using both solvers achieve better results than the standard SVM method, because of using additional textual information in the training process. Moreover, by incorporating the multi-instance learning approach, the MAPs of mi-SVM-PI method based on both solvers are further improved. In both cases, the methods based on our ℓ_2-SVM+ achieve slight better results than the ones based on MAT-SVM+.

In terms of the efficiency, we observe that our ℓ_2-SVM+ again achieves order-of-magnitude speed-up when compared with MAT-SVM+. When incorporating the multi-instance learning approach, the mi-SVM-PI based on our ℓ_2-SVM+ uses only a bit more time than ℓ_2-SVM+, because we only need to calculate the matrix inverse once for each concept, and iteratively solve the one-class SVM problem. In contrast, the mi-SVM-PI based on MAT-SVM+ needs to iteratively solve the QP problem introduced by the SVM+ problem. Therefore, our method is more than 30 times faster than MAT-SVM+ on the NUS-WIDE dataset.

7. Conclusion

In this paper, we have proposed two new algorithms for solving the linear and kernel SVM+. By reformulating the original SVM+ method, we obtain the dual problem with simpler constraints. Then we develop an efficient dual coordinate descent algorithm to solve the linear SVM+ problem. We also show that the kernel SVM+ using the ℓ_2-loss can be converted to the one-class SVM problem, and thus can be efficiently solved by using the SMO algorithm implemented in the existing SVM solvers such as LIBSVM. Comprehensive experiments on three tasks have demonstrated the efficiency of our proposed algorithms for linear and kernel SVM+.

Acknowledgement

The work is supported by the ERC Advanced Grant Varcity (#273940).

References

[1] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2:27:1–27:27, 2011. 2, 5

[2] J. Chen, X. Liu, and S. Lyu. Boosting with side information. In *ACCV*, 2012. 2

[3] L. Chen, W. Li, and D. Xu. Recognizing RGB images by learning from RGB-D data. In *CVPR*, pages 1418–1425, 2014. 2

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: a real-world web image database from National University of Singapore. In *CIVR*, 2009. 2

[5] D. Dai, T. Kroeger, R. Timofte, and L. Van Gool. Metric imitation by manifold transfer for efficient vision applications. In *CVPR*, 2015. 1, 2

[6] D. Dai, Y. Wang, Y. Chen, and L. Van Gool. Is image super-resolution helpful for other vision tasks? In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. 6

[7] J. Ding, M. Shao, and Y. Fu. Latent low-rank transfer subspace learning for missing modality recognition. In *AAAI*, 2014. 2

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 8

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008. 2, 4, 6

[10] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *JMLR*, 6:1889–1918, 2005. 2

[11] J. Feyereisl, S. Kwak, J. Son, and B. Han. Object localization based on structural SVM using privileged information. In *NIPS*, 2015. 1, 2

[12] S. Fouad, P. Tino, S. Raychaudhury, and P. Schneider. Incorporating privileged information through metric learning. *T-NNLS*, 24(7):1086–1098, 2013. 2

[13] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. *arXiv:1507.00448*, 2015. 2

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop, NIPS*, 2014. 2

[15] J. Hoffman, S. Guadarrama, E. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko. LSDA: Large scale detection through adaptation. In *NIPS*, 2014. 2

[16] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008. 1, 3, 4, 7

[17] Y. Ji, S. Sun, and Y. Lu. Multitask multiclass privileged information support vector machines. In *ICPR*, 2012. 2

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 6

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 6

[20] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *T-PAMI*, 2013. 1, 2

[21] M. Lapin, M. Hein, and B. Schiele. Learning using privileged information: SVM+ and weighted SVM. *Neural Networks*, 53:95–108, 2014. 2

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2, 6

[23] W. Li, L. Niu, and D. Xu. Exploiting privileged information from web data for image categorization. In *ECCV*, pages 437–452, 2014. 2, 8

[24] L. Liang and V. Cherkassky. Connection between SVM+ and multi-task learning. In *IJCNN*, 2008. 2, 6

[25] L. Niu, W. Li, and D. Xu. Exploiting privileged information from web data for action and event recognition. *IJCV*, 2015. 5, 8

[26] D. Pechyony, R. Izmailov, A. Vashist, and V. Vapnik. SMO-style algorithms for learning using privileged information. In *DMIN*, 2010. 1, 2, 3, 5, 6, 7

[27] D. Pechyony and V. Vapnik. On the theory of learning with privileged information. In *NIPS*, 2010. 2

[28] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Advances in Kernel Methods - Support Vector Learning, 1998. 1

[29] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to rank using privileged information. In *ICCV*, 2013. 1, 2

[30] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, 2012. 2

[31] V. Vapnik and R. Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *JMLR*, 16:20232049, 2015. 2

[32] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(56):544–557, 2009. 1, 2, 6, 7

[33] Z. Wang and Q. Ji. Classifier learning with hidden information. In *CVPR*, 2015. 1, 2

[34] X. Xu, W. Li, and D. Xu. Distance metric learning using privileged information for face verification and person re-identification. *T-NNLS*, 26:3150–3162, Dec 2015. 2

[35] Q. Zhang, G. Hua, W. Liu, Z. Liu, and Z. Zhang. Can visual recognition benefi from auxiliary information in training? In *ACCV*, 2014. 1, 2