Neurocomputing 235 (2017) 199–209

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Large-scale robust transductive support vector machines

Hakan Cevikalp[a,*], Vojtech Franc[b]

[a] Electrical and Electronics Engineering, Department of Eskisehir Osmangazi University, Meselik, 26480 Eskisehir, Turkey
[b] Department of Cybernetics, Czech Technical University, Prague, Czech Republic

ARTICLE INFO

Communicated by Zidong Wang

Keywords:
Transductive support vector machines
Classification
Large-margin classifier
Optimization

ABSTRACT

In this paper, we propose a robust and fast transductive support vector machine (RTSVM) classifier that can be applied to large-scale data. To this end, we use the robust Ramp loss instead of Hinge loss for labeled data samples. The resulting optimization problem is non-convex but it can be decomposed to a convex and concave parts. Therefore, the optimization is accomplished iteratively by solving a sequence of convex problems known as concave-convex procedure. Stochastic gradient (SG) is used to solve the convex problem at each iteration, thus the proposed method scales well with large training set size for the linear case (to the best of our knowledge, it is the second transductive classification method that is practical for more than a million data). To extend the proposed method to the nonlinear case, we proposed two alternatives where one uses the primal optimization problem and the other uses the dual. But in contrast to the linear case, both alternatives do not scale well with large-scale data. Experimental results show that the proposed method achieves comparable results to other related transductive SVM methods, but it is faster than other transductive learning methods and it is more robust to the noisy data.

1. Introduction

Supervised learning techniques, e.g., large margin classifiers [1,2,3,4], use training data with class labels being associated to the data samples to find a prediction function to estimate labels of new test data samples. However, in many applications, there is a lack of labeled data since obtaining labels is a costly procedure as it often requires human effort. Furthermore, manual labeling is a slow and error-prone process. As a result, in many applications, only a small fraction of data samples can be labeled although we have access to a massive collection of unlabeled data. For instance, in web and text categorization applications, there are abundant unlabeled data that can be collected easily by a user. Similarly, in genomics applications, functions of many genes in sequenced genomes remain unknown and there is only a limited amount of available labeled biological information. In most cases, unlabeled data carry additional information that helps us to find a more accurate prediction function. Transductive (or semi-supervised) learning aims to find a better prediction function on the basis of information coming from both labeled and unlabeled data in contrast to the supervised learning which only uses labeled samples and ignores any information potentially conveyed in unlabeled data samples.

Using unlabeled data samples for learning was first proposed by Vapnik and Sterin [5] under the name transductive support vector machine (TSVM), and the first implementation of TSVM was introduced by Bennett and Demiriz [6]. Bennett and Demiriz [6] formulated the learning problem as a mixed integer program. This formulation requires the introduction of a binary variable for each unlabeled sample in the training set, thus the method is not suitable for large-scale data. Then, Joachims [7] proposed a local combinatorial search approach, called as SVMLight-TSVM (TSVM^{Light}), that is practical for about 10 thousand examples. In this method, a supervised SVM is trained first and it is used for labeling the unlabeled data. Then, current solution is improved by switching labels of a pair of unlabeled data samples selected based on some heuristic techniques. Chapelle and Zien [8] proposed a gradient-descent based margin maximization algorithm to solve the transductive SVM learning problem and combined it with a graph-based embedding to improve the results. Debie and Cristianini [9] and Xu et al. [10] relax the TSVM training problem and formulate it as a convex semi-definite programming. Although, these convex methods do not suffer from local optimal solutions, they do not scale well with large-scale data. Collobert et al. [11] and Wang et al. [12] solve TSVM problem using an approximation optimization procedure known as concave-convex procedure [13], which decomposed the TSVM optimization problem into convex and concave parts. The optimization is accomplished iteratively by solving a sequence of convex problems obtained by linearly approximating the concave lost function. Although these methods suffer from locally optimal solutions, they scale well with the data set size compared to other TSVM

* Corresponding author.
E-mail addresses: hakan.cevikalp@gmail.com (H. Cevikalp), xfrancv@cmp.felk.cvut.cz (V. Franc).

http://dx.doi.org/10.1016/j.neucom.2017.01.012
Received 17 February 2016; Received in revised form 19 December 2016; Accepted 1 January 2017
Available online 06 January 2017
0925-2312/

algorithms. More recently, Li et al. [14] introduced WELLSVM (Weakly Labeled SVM) that maximizes the margin by generating the most violated label vectors iteratively and then combines them by using efficient multiple kernel learning techniques. This method uses the cutting plane algorithm thus it is suitable for large-scale datasets with sizes up to a few millions. There are also other methods that use deterministic annealing [15,16], branch-and-bound algorithms [17], non-smooth optimization method [18], continuation method [19], maximum entropy [20], active learning [21], random-vector functional network [22], multiple kernel learning [23] and others [24] to solve transductive learning problems. Lastly, in addition to these margin-based transductive approaches there are also methods [25,26,27,28,29] that use limited amount of labeled data to estimate labels of unlabeled data by using graph-based (spectral clustering) techniques, but we do not consider these in our study. A more comprehensive survey of transductive optimization techniques can be found in [30].

In this paper, we propose a robust and fast TSVM method based on TSVM method of [11]. More precisely, we replace the Hinge loss that is used for labeled data with a more robust Ramp loss. Moreover, we solve the optimization problem in the primal space by using a stochastic gradient algorithm as opposed to [11] that solves the TSVM problem in the dual space. Solving the optimization problem in the primal space using stochastic gradient enables us to use the method in large-scale datasets including millions of data. In contrast, the dual solver of [11] does not scale with large-scale data well since it uses a quadratic programming solver based on generalized sequential minimal optimization (SMO) [31] algorithm.

2. Method

2.1. Preliminaries and TSVM formulation

Consider that we are given a set of L labeled training samples $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x} \in \mathrm{IR}^d$, $y \in \{+1, -1\}$ and an unlabeled set of U vectors $\mathcal{U} = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+U}\}$. Our goal is to find the best separating hyperplane characterized by $\theta = (\mathbf{w}, b)$, where $\mathbf{w}$ is the normal of the hyperplane and b is the bias term. To label new samples, we use the sign of the following decision function

$$f_\theta(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b. \tag{1}$$

The main idea of TSVM learning is to find an hyperplane that separates the labeled samples with a large margin at the same time ensures that the unlabeled samples will be as far as possible from the margin as illustrated in Fig. 1. To this end, earlier methods [6,7,8] used the following optimization formulation

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+U}\xi_i \text{ s. t. } y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 - \xi_i,$$
$$i = 1, \ldots, L, \ |\mathbf{w}^\top\mathbf{x}_i + b| \geq 1 - \xi_i, \ \ i = L+1, \ldots, L+U. \tag{2}$$

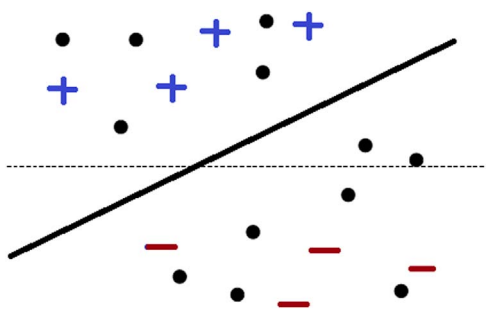


Fig. 1. Finding the best separating hyperplane by transductive learning: The supervised SVM which uses only labeled data (shown with '+' and '–') returns the dashed line as separating hyperplane. However, when the unlabeled data (shown with black circles) is also incorporated in learning, a better separating hyperplane shown with solid line can be found (figure is adopted from [7])

This minimization problem can also be written as unconstrained optimization problem in the form

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top\mathbf{x}_i + b)) + C^*\sum_{i=L+1}^{L+U} H_1(|\mathbf{w}^\top\mathbf{x}_i + b|), \tag{3}$$

where the function $H_1(t) = \max(0, 1 - t)$ is the classical Hinge loss plotted in Fig. 2, and $C(C^*)$ is a user defined parameter that controls the weight of errors associated to the labeled (unlabeled) data samples. The loss function for unlabeled data is shown in Fig. 3 (a). Note that the loss function on the unlabeled data is not differentiable, thus it is replaced by symmetric sigmoid function $\exp(-3t^2)$ in [8]. It turned out the TSVM formulation given in (3) has the potential to assign all unlabeled samples to only one of the classes with a very large margin, which yields a poor classification accuracy. In order to solve this problem, a balancing constraint that enforces the unlabeled data to be assigned to both classes based on the same fraction of labeled data samples is introduced in [7]. Chapelle and Zien [8] used the following relaxed balancing constraint, which we also use in this study

$$\frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top\mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{4}$$

It should be noted that TSVM optimization problem given in (3) is not convex. Therefore, Collobert et al. [11] and Wang et al. [12] used concave-convex procedure (CCCP) to solve this non-convex problem. CCCP basically decomposes a non-convex function into a convex and a concave part and it solves the problem by an iterative procedure where each iteration approximates the concave part by its tangent and minimizes the resulting convex function as given in Algorithm 1. The convergence of CCCP algorithm has been proved in [13]. It should be noted that the convex optimization problem that constitutes the core of the CCCP algorithm can be solved by using efficient convex algorithms. Collobert et al. [11] replaced the symmetric Hinge loss of unlabeled points with the symmetric Ramp loss defined as

$$SR_s(t) = R_s(t) + R_s(-t), \tag{5}$$

where $R_s(t) = \min(1 - s, \max(0, 1 - t))$ is the Ramp Loss function illustrated in Fig. 2. Here $-1 < s \leq 0$ is a parameter that must be set by the user. As shown in the figure, the Ramp Loss function can be written as the sum of the convex Hinge loss and a concave loss function (or as the difference between two convex Hinge losses), i.e., $R_s(t) = H_1(t) - H_s(t)$. The Ramp Loss function can be seen as a clipped version of the Hinge loss, and the parameter s controls where we clip the Ramp loss. For the symmetric Ramp loss function s parameter controls the wideness of the flat part of the symmetric component of the symmetric Ramp loss plotted in Fig. 3. After some algebra, the final transductive SVM method is converted to an iterative procedure where a dual convex quadratic programming problem is solved at each iteration [11]. It should be noted that this method uses dual formulation of SVM to solve the convex QP problem and it is based on SMO algorithm. Therefore, it is better suited for moderate sized data rather than large-scale data.

Algorithm 1. The Concave-Convex Procedure

Initialize θ^0
repeat
$\theta^{t+1} = \arg\min_\theta(J_{convex}(\theta) + J'_{concave}(\theta^t)\theta)$
until convergence of θ^t.

2.2. Robust Transductive Support Vector Machine (RTSVM) classifier

Collobert et al. [11] use the Hinge loss for labeled samples and the symmetric Ramp loss for unlabeled samples. These loss functions are not in the same range as shown in Figs. 2 and 3. For the symmetric

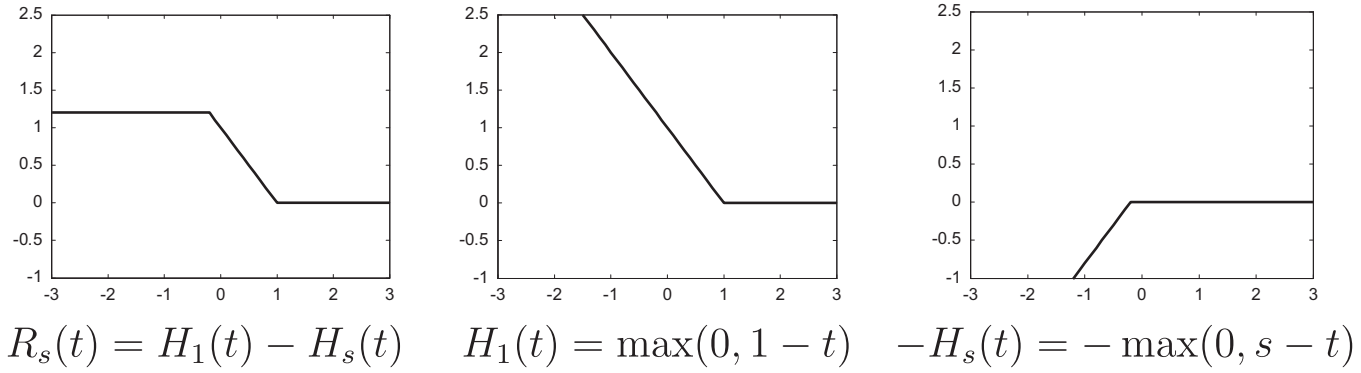


Fig. 2. The illustration of the Ramp loss function, $R_s(t) = H_1(t) - H_s(t)$, where $H_a(t) = \max(0, a - t)$ is the classical Hinge loss. Here, we set $s = -0.20$.

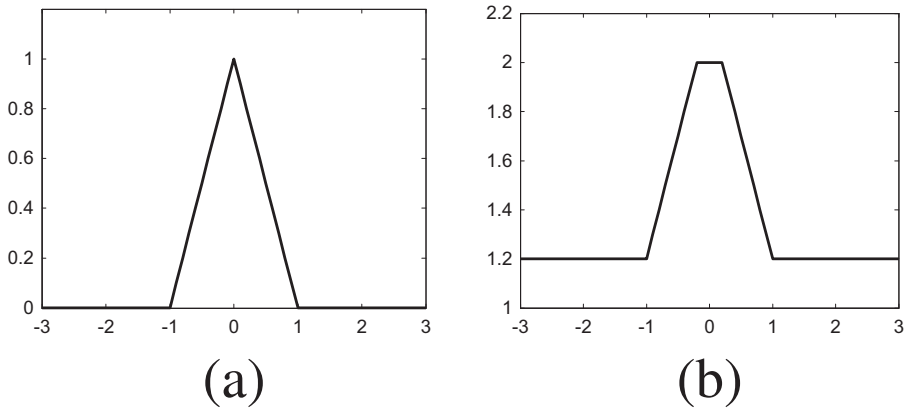


Fig. 3. Loss functions used for unlabeled data: (a) $H_1(|t|) = \max(0, 1 - |t|)$, (b) The symmetric Ramp loss, $SR_s(t) = R_s(t) + R_s(-t)$. Here, we set $s = -0.20$.

Ramp loss, a sample can introduce at most a limited amount of cost value no matter of its position with respect to margin in the input space (the loss can be maximum 0.8 when s is set to −0.2). However, there is no bound for the Hinge loss, e.g., a single outlying point farther from the margin can yield to a large loss. Therefore, the labeled outlying points – the samples that are misclassified outside the margin – start to play a dominant role in determining the separating hyperplane since they tend to have the largest margin according to the Hinge loss. Also, weight parameter of labeled samples C is usually set to higher values compared to weight parameter of unlabeled samples C^*, which aggravates the problem. To ameliorate this drawback, we interchange the convex Hinge loss with a more robust non-convex Ramp loss function. The Ramp loss also bounds the maximum amount of loss similar to the symmetric Ramp loss function and this helps to suppress the influence of misclassified examples. The superiority of the Ramp loss over the Hinge loss for supervised SVM training is well-proven and demonstrated in [32,33,34,35], so we adopt it to transductive learning here.

Our robust TSVM method solves the following problem

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} R_s(y_i(\mathbf{w}^\top\mathbf{x}_i + b)) + C^*\sum_{i=L+1}^{L+U} SR_s(\mathbf{w}^\top\mathbf{x}_i + b)$$
$$\text{s. t. } \frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top\mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{6}$$

This optimization problem is identical to the one studied by Collobert et al. [11] with the exception that the more robust non-convex Ramp loss is used for labeled samples instead of the Hinge loss. To use the symmetric Ramp loss function defined for unlabeled data samples, each unlabeled sample appears as two examples labeled with both negative and positive classes. More precisely, we create the new samples as follows

$$y_i = +1,\ \ i \in [L+1, \ldots, L+U],\ y_i = -1,$$
$$i \in [L+U+1, \ldots, L+2U],\ \mathbf{x}_i = \mathbf{x}_{i-U},\ \ i \in [L+U+1, \ldots, L+2U]. \tag{7}$$

Then, by using the equations $R_s(t) = H_1(t) - H_s(t)$ and $SR_s(t) = R_s(t) + R_s(-t)$, the above cost function without constraint can be written as

$$J(\theta) = J_{convex}(\theta) + J_{concave}(\theta), \tag{8}$$

where

$$J_{convex}(\theta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top\mathbf{x}_i + b)) \quad + C^*\sum_{i=L+1}^{L+2U} H_1(y_i(\mathbf{w}^\top\mathbf{x}_i + b)), \tag{9}$$

and

$$J_{concave}(\theta) = -C\sum_{i=1}^{L} H_s(y_i(\mathbf{w}^\top\mathbf{x}_i + b)) \quad - C^*\sum_{i=L+1}^{L+2U} H_s(y_i(\mathbf{w}^\top\mathbf{x}_i + b)). \tag{10}$$

Because the cost function (6) can be decomposed into a convex and concave part, we can apply concave-convex procedure (CCCP) to solve the problem. By employing CCCP, the minimization of $J(\theta)$ with respect to $\theta = (\mathbf{w}, b)$ can be achieved by iteratively updating the parameter θ by the following rule

$$\theta^{t+1} = \arg\min_{\theta}(J_{convex}(\theta) + J'_{concave}(\theta^t)\theta), \tag{11}$$

subject to the constraint $\frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top\mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i$. To this end, we need to find the derivative of the concave part with respect to θ,

$$\frac{\partial J_{concave}(\theta)}{\partial\theta} = -C\sum_{i=1}^{L}\frac{\partial H_s(\theta)}{\partial f_\theta(\mathbf{x}_i)}\frac{\partial f_\theta(\mathbf{x}_i)}{\partial\theta} - C^*\sum_{i=L+1}^{L+2U}\frac{\partial H_s(\theta)}{\partial f_\theta(\mathbf{x}_i)}\frac{\partial f_\theta(\mathbf{x}_i)}{\partial\theta}.$$

To simplify this, let us define

$$\beta_i = y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} = \begin{cases} C, & \text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) < s \quad \text{and} \quad 1 \le i \le L \\ C^*, & \text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) < s \quad \text{and} \quad L+1 \le i \le L+2U \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

By using the definition $f_\theta(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ and $\partial f_\theta(\mathbf{x}_i)/\partial\theta = (\mathbf{x}_i, 1)$, each update of CCCP applied to the minimization problem requires the minimization of the following cost

$$J(\theta) = J_{convex}(\theta) + \frac{\partial J_{concave}(\theta)}{\partial \theta} = J_{convex}(\theta) + \left(\sum_{i=1}^{L+2U} y_i \beta_i \frac{\partial f_\theta(\mathbf{x}_i)}{\partial \theta} \right) \theta$$
$$= J_{convex}(\theta) + \sum_{i=1}^{L+2U} \beta_i y_i (\mathbf{w}^\top \mathbf{x}_i + b).$$

subject to $\frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i$. By inserting the Hinge losses in the convex cost function, the overall optimization problem becomes

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + \sum_{i=1}^{L+2U} \beta_i y_i (\mathbf{w}^\top \mathbf{x}_i + b) \quad \text{s. t.} \ \frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{13}$$

The above cost function includes sum of convex functions, thus the resulting cost function is also convex. Instead of taking dual of the problem and solving it with a dual QP solver as in [11], we consider the primal problem and solve it by using stochastic gradient (SG). We simply prefer SG here since it scales better with data set size compared to the dual QP solver. Without the balancing constraint, application of the SG algorithm is pretty straightforward, but the balancing constraint complicates the problem. To handle the constraint, we first simplify the balancing constraint as follows

$$\mathbf{v}^\top \mathbf{c} = \gamma, \quad \text{where} \quad \mathbf{v} = \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}, \ \mathbf{c} = \begin{pmatrix} \frac{1}{U}\sum_{i=L+1}^{L+U} \mathbf{x}_i \\ 1 \end{pmatrix}, \ \gamma = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{14}$$

There are basically two approaches to force the equality constraint: In the first approach, we initialize the algorithm with the hyperplane parameters satisfying the constraint and then select directions that will both decrease the objective function and satisfy the constraint at the same time. The following descent direction satisfies both

$$\mathbf{d} = -\left(\mathbf{I} - \frac{\mathbf{c}^\top \mathbf{c}}{\|\mathbf{c}\|^2} \right) \mathbf{g} \tag{15}$$

where $\mathbf{g}$ is the sub-gradient of the objective function with respect to $\theta = (\mathbf{w}, b)$, and $\mathbf{I}$ is the identity matrix. The TSVM is generally initialized by using the hyperplane returned by the supervised SVM classifier and there is no guarantee that the constraint is satisfied initially. As a second more convenient approach that was adopted in this study, we can project the updated hyperplane parameters on the balancing constraint after each update. We define the orthogonal projection $\mathcal{P}$: $\mathrm{IR}^{d+1} \to \mathrm{IR}^{d+1}$ onto the affine sub-space $\{\mathbf{v} \in \mathrm{IR}^{d+1} | \mathbf{v}^\top \mathbf{c} = \gamma\}$ as

$$\mathcal{P}(\mathbf{v}) = \arg\min_{\hat{\mathbf{v}}} \|\hat{\mathbf{v}} - \mathbf{v}\|, \ \text{s. t.} \ \mathbf{c}^\top \hat{\mathbf{v}} = \gamma, \tag{16}$$

which has the closed form solution

$$\mathcal{P}(\mathbf{v}) = \frac{\mathbf{c}}{\|\mathbf{c}\|^2}(\gamma - \mathbf{c}^\top \mathbf{v}) + \mathbf{v}. \tag{17}$$

So this is very straightforward procedure that can work with any initialization. The resulting final robust TSVM method can be summarized as in Algorithm 2. We use SG algorithm given in Algorithm 3 to solve the convex minimization problem that constitutes the core of the CCCP. Solving the optimization problem in the primal space using SG is the key for the speed and scalability. In [36], it was shown that total run time complexity of SG algorithm is given by $O(d/(\lambda\epsilon))$, where d is a bound on the number of non-zero features, λ is the regularization parameter. Therefore, the run-time does not directly depend on the size of the training set, which makes SG algorithm ideal for large-scale datasets. As a result, the proposed method also scales well with large-scale data. On the other hand, solvers using dual space are not applicable for large-scale data. The reason is that a runtime of any genuine dual-space solver (i.e solver optimizing the dual variables only) cannot be better than $O(n^2)$ for noisy data (which is the time needed just to evaluate the objective), where n stands for the number of examples. As a result, the dual-space solvers, like the SMO algorithm, are not applicable for truly large-scale problems. Finally, to initialize the method, we use supervised linear SVM with labeled data samples only as in [11]. CCCP converges very quickly, e.g., in our experiments CCCP converged to a solution after at most five iterations.

Algorithm 2. The Robust Transductive Support Vector Machines (RTSVM)

Initialize $\theta^0 = (\mathbf{w}^0, b^0)$, t=0, $\epsilon_1 > 0$, $\epsilon_2 > 0$
Compute

$$\beta_i^0 = y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} = \begin{cases} C, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \quad \text{and} \quad 1 \le i \le L \\ C^*, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \quad \text{and} \quad L+1 \le i \le L+2U \\ 0, & \text{otherwise.} \end{cases}$$

while $\| \mathbf{w}_{t+1} - \mathbf{w}_t \| \ge \epsilon_1$ or $\| \boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t \| \ge \epsilon_2$ **do**
– Solve the following convex minimization problem by using SG algorithm given in Algorithm 3
$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + \sum_{i=1}^{L+2U} \beta_i y_i (\mathbf{w}^\top \mathbf{x}_i + b)$
such that $\frac{1}{U}\sum_{i=1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i$;
– Set $\mathbf{w}^{t+1} = \mathbf{w}$, $b^{t+1} = b$;
– Compute

$$\beta_i^{t+1} = \begin{cases} C, & \text{if } y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \quad \text{and} \quad 1 \le i \le L \\ C^*, & \text{if } y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \quad \text{and} \quad L+1 \le i \le L+2U \\ 0, & \text{otherwise.} \end{cases}$$

– Set $t = t + 1$;
end while

Algorithm 3. Stochastic Gradient Based Solver with Projection

Initialize
$\mathbf{w}_1$, b_1, $T > 0$, $\lambda_0 > 0$, $\epsilon > 0$
Description:
for $t \in 1, \ldots, T$ **do**
$\lambda_t \leftarrow \lambda_0/t$;
for $i \in$ randperm$(L + 2U)$ **do**
– Compute sub-gradients

$$\mathrm{g}_t = \begin{cases} -y_i C(C^*)\mathbf{x}_i + \beta_i y_i \mathbf{x}_i, & \text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \le 1 \\ \beta_i y_i \mathbf{x}_i, & y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1. \end{cases} h_t$$
$$= \begin{cases} -y_i C(C^*) + \beta_i y_i, & \text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \le 1 \\ \beta_i y_i, & y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1. \end{cases}$$

– Update hyperplane parameters
$\widetilde{\mathbf{w}}_t \leftarrow \mathbf{w}_t - \frac{\lambda_t}{L+2U}(\mathbf{w}_t + \mathrm{g}_t)$
$\widetilde{b}_t \leftarrow b_t - \frac{\lambda_t}{L+2U} h_t$
– Project parameters onto the feasible set imposed by the constraint

$(\mathbf{w}_t, b_t) = \mathcal{P}(\widetilde{\mathbf{w}}_t, \widetilde{b}_t)$
end for
if $(t \geq 2)\&(\|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \epsilon)$, **break**
end for

Algorithm 4. The Kernel Robust Transductive Support Vector Machines (Kernel RTSVM)

Initialize $\theta^0 = (\mathbf{w}^0, b^0)$, t=0, $\epsilon_1 > 0$, $\epsilon_2 > 0$
Compute

$$\beta_i^0 = y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} = \begin{cases} C, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \quad \text{and} \quad 1 \leq i \leq L \\ C^*, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \quad \text{and} \quad L+1 \leq i \leq L+2U \\ 0, & \text{otherwise.} \end{cases}$$

Set $\zeta_i = y_i$ for $1 \leq i \leq L + 2U$ and $\zeta_0 = (1/L)\sum_{i=1}^{L} y_i$.
while $\| \mathbf{w}_{t+1} - \mathbf{w}_t \| \geq \epsilon_1$ or $\| \boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t \| \geq \epsilon_2$ **do**
– Solve the following convex minimization problem by using SMO

$$\arg\min_{\widetilde{\boldsymbol{\alpha}}} \frac{1}{2}\widetilde{\boldsymbol{\alpha}}^\top \mathbf{K}\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\zeta}^\top \widetilde{\boldsymbol{\alpha}} \qquad \text{s. t. } 0 \leq y_i\widetilde{\alpha}_i \leq C, \forall\, i \in [1, \ldots, L]$$
$$, -\beta_i \leq y_i\widetilde{\alpha}_i \leq C^* - \beta_i, \forall$$
$$i \in [L+1, \ldots, L+2U], \sum_{i=0}^{L+2U} \widetilde{\alpha}_i = 0.$$

– Compute **w**and b by using (25) and (30) and set $\mathbf{w}^{t+1} = \mathbf{w}$, $b^{t+1} = b$;
– Compute

$$\beta_i^{t+1} = \begin{cases} C, & \text{if} y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \quad \text{and} \quad 1 \leq i \leq L \\ C^*, & \text{if} y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \quad \text{and} \quad L+1 \leq i \leq L+2U \\ 0, & \text{otherwise.} \end{cases}$$

– Set $t = t + 1$;
end while

2.3. Using kernel functions with RTSVM

Let $\Phi(.)$ be the implicit feature space embedding and $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ be the corresponding kernel function, where $\langle . \rangle$ denotes the feature space inner product. The proposed method can be used with kernels without explicitly mapping the data samples since the solution can be expressed as a linear combination of the training samples. To kernelize the method we use two approaches: The first approach is only suitable for moderate sized data and it uses the dual of the problem to write the solution in terms of the inner products data samples as in [11]. The derivations are given in Appendix and the final resulting algorithm is summarized in Algorithm 4. In the second approach, we directly minimize the primal problem by using kernels as in [36,37]. The main trick is that, at each iteration of the SG algorithm the separating hyperplane normal can be written as $\mathbf{w} = \sum_{i \in S} \alpha_i \Phi(\mathbf{x}_i)$, where S is the subset of $\{1, \ldots, L + 2U\}$ that includes the indices corresponding to the non-zero α_i coefficients (the coefficients that correspond to the support vectors). Therefore, we can store the set S and the non-zero coefficients α_i instead of storing **w**. The inner product with any mapped test sample $\Phi(\mathbf{x}_{test})$ can be computed as

$$\langle \mathbf{w}, \Phi(\mathbf{x}_{test}) \rangle = \sum_{i \in S} \alpha_i k(\mathbf{x}_i, \mathbf{x}_{test}). \tag{18}$$

Note that only one element of $\boldsymbol{\alpha}$ is changed at each iteration and the method uses kernel evaluations instead of explicit mapping. It is worthwhile pointing out that even though the solution is represented in terms of α_i, we calculate the sub-gradient with respect to the hyperplane normal **w**.

It turns out that both methods are computationally expensive and they can be used for moderate sized data only. Moreover, stochastic gradient based kernel RTSVM is even computationally more expensive than SMO based algorithm given in Algorithm 4 since the solution is not sparse anymore (almost all α_i coefficients are nonzero) and we have to make kernel evaluations with respect to all training data at each iteration. So, we recommend to use SMO based kernel method.

3. Experiments

We provide illustrative tests of transductive learning methods on both synthetic and real-world datasets with different characteristics. Our Robust Transductive SVM (RTSVM) method[1] is compared to classical SVM and other transductive learning methods including TSVM^{Light} [7], TSVM^{CCCP} [11], WELLSVM [14] and TSVM^{LDS} [8]. For multi-class classification problems, we used one-against-rest (OAR) regime for all methods. For classical SVM, we trained the classifier with labeled data and used unlabeled data as a test set whereas the transductive learning methods used both training and test data to learn the classifier. Finally, for small datasets, we used LIBSVM package[2] for training whereas LIBOCAS[3] is used for large-scale data experiments.

3.1. Experiments on synthetic data

We first test our algorithms on a synthetic data. To this end, we created normally distributed data shown in Fig. 4. Both classes have the same covariance structure but their centers are different, and each training class has 50 samples. For testing, we also created 50 samples per class by using the same distribution. The classes are close to being linearly separable, so we test only linear kernels here. To demonstrate the robustness against to outliers, we gradually corrupted data switching the randomly chosen examples of positive and negative classes. Table 1 shows the classification accuracies of test set for different fraction of outliers (e.g., 20% rate is obtained by switching the labels of 10 samples from both classes). For clean data, SVM, TSVM^{CCCP}, and TSVM^{Light} implementation slightly beat others. As the number of outlying samples is increased, the accuracies of all other methods drop except RTSVM and TSVM^{LDS} methods where the proposed RTSVM yields 93% and TSVM^{LDS} yields 93% or 92% until outliers ratio is 40%. Accuracies of other methods are still satisfactory even though the fraction of outliers is quite high owing to the fact that we have chosen a small C parameter to reduce the effects of outliers. But, when the fraction of outliers is set to 50% there is a significant drop in accuracies of all methods. But our proposed RTSVM is less affected compared to others and it significantly outperforms other classifiers by achieving 73% accuracy even though the half of the data samples are outliers. It should be noted WELLSVM yields the worst accuracy. It is due to the lack of the bias term. The provided software by the authors [14] only learns hyperplane normal without the bias term (as a result all hyperplanes are restricted to pass through the origin), thus the returned hyperplane sometimes fails to separate the data well as in this example.

3.2. Experiments on small datasets

Here we test our algorithms on the four well-known datasets used for testing transductive learning methods, another three datasets from UCI Repository[4] and Volatile Organic Compound (VOC) dataset.[5] The

[1] Our code and some tested datasets are available at http://mlcv.ogu.edu.tr/softwarertsvm.html.
[2] available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[3] available at http://cmp.felk.cvut.cz/~xfrancv/ocas/html/
[4] http://archive.ics.uci.edu/ml
[5] http://users.rowan.edu/~polikar/RESEARCH/vocdb.html

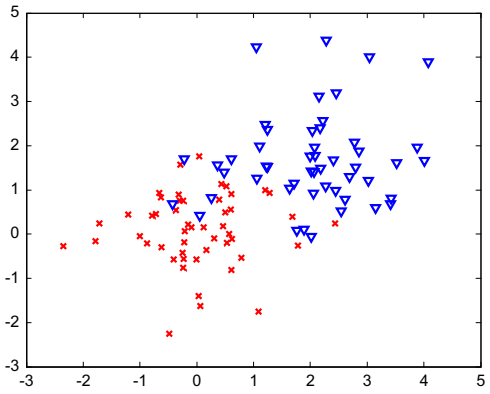


Fig. 4. Normally distributed synthetic data.

Table 1
Classification Rates (%) on the Synthetic Dataset.

Method	Clean Data	10%	20%	30%	40%	50%
SVM	**94**	90	88	83	85	49
TSVM^{Light}	**94**	91	91	89	90	37
TSVM^{LDS}	93	**93**	92	92	**93**	28
TSVM^{CCCP}	**94**	91	86	83	86	44
WELLSVM	80	72	72	72	71	41
RTSVM	93	**93**	**93**	**93**	**93**	**73**

Table 2
Small Sized Datasets.

Data Set	# Classes	Dimensionality	# Samples	# Labels
g50c	2	50	550	50
g10n	2	10	550	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50
Iris	3	4	150	45
VOC	5	6	384	65
Wine	3	13	178	18
WDBC	2	30	569	50

sizes and dimensionality of the datasets are given in Table 2. The first four datasets are used to test transductive learning algorithms in [8,30] and 10 different splits for each dataset into labeled and unlabeled points are provided by the authors. We use the same splits and therefore our results are directly comparable to other tested methods in [8,30]. For the remaining datasets, we used the same setup and create 10 different splits by choosing random training and test samples. Results are given in Table 3. Except Coil20 and Uspst datasets, the transductive learning algorithms typically beat SVM results or yield to similar accuracies. Among all tested transductive learning algorithms, TSVM^{LDS} is clearly the best performing method and it achieves the best accuracies for four of the tested datasets. The performance difference is very significant especially on Coil20, Uspst and Iris datasets. Our proposed method slightly beats others on two datasets. In general, TSVM^{LDS} first applies a nonlinear embedding method based on local neighborhood and then applies TSVM to the embedded data. So, we apply the same setting for the proposed method and denote the resulting method by RTSVM^{LDS}. Our results also significantly improve after this procedure and we get the best accuracies on Coil20 and Uspst datasets by using RTSVM^{LDS}. These results show that the significant improvement on those 3 datasets is due to the preprocessing of the data not due to the type of the transductive SVM solver. Note that we use a very limited number of labeled samples for

Table 3
Classification Rates (%) on the Small Datasets.

Method	G50	Coil20	Text	Uspst	Iris	VOC	Wine	WDBC
SVM	86.56	73.66	81.20	75.07	90.90	70.22	69.06	93.58
TSVM^{Light}	90.14	71.93	**92.53**	72.28	90.19	71.47	69.69	93.58
TSVM^{LDS}	94.60	90.07	85.52	81.57	**96.55**	**80.62**	70.56	95.32
TSVM^{CCCP}	87.86	71.41	84.79	70.11	90.65	79.28	**70.69**	**95.59**
WELLSVM	90.04	67.81	88.64	72.14	92.49	73.89	67.56	93.55
RTSVM	**94.72**	71.41	83.67	74.83	91.93	73.98	69.44	95.32
RTSVM^{LDS}	92.22	**91.17**	82.55	**82.16**	95.35	79.44	69.37	94.95

all datasets and there is very unlikely that there are many outliers. For such cases, RTSVM and TSVM^{CCCP} must yield to identical results, but we obtain different accuracies since we use solvers (SG and SMO) with very different characteristics for those methods.

3.3. Experiments on large scale data

Here we test our algorithm on five large-scale datasets: CIFAR10, a face dataset which is created by using three public datasets, Epsilon, SensIT Vehicle, and MNIST. The proposed RTSVM, TSVM^{CCCP}, and WELLSVM are the only methods that can be applied in this setting, so we compare only these methods. For some datasets, TSVM^{CCCP} failed to converge since it uses SMO. If the result of SVM^{CCCP} is not given in a table, it indicates that it failed to converge for that particular dataset in 10 days.

3.3.1. Cifar10 experiments

Cifar10 dataset includes 32×32 color images of 10 object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Some images from this dataset are shown in Fig. 5. There are 6000 images per class, thus the total number of samples is 60,000. Fisher vector (FV) representation is used to represent images and we used a similar setup as in [38]. The dimensionality of the tested descriptors is reduced to 64 by using Principal Component Analysis. We used 1.2×10^6 descriptors to learn PCA projections and 128-component Gaussian mixture model (GMM) components. The final dimension of the image FVs is 16,384. So both the dimensionality and the training set size are large.

In our experiments we randomly selected 1000 labeled data samples from each class and used the remaining 5000 as unlabeled data samples. This procedure is repeated 5 times and the final accuracies are averages of the results obtained in each trial. Accuracies are given in Table 4. WELLSVM achieves the best accuracy followed by the proposed method.

3.3.2. Gender recognition experiments

We used a collection of three public datasets: Labeled Faces in the Wild [39], PubFig [40] and PAL [41]. In addition, we also downloaded around 14 thousand face images from the Internet. The images are annotated by several independent persons. We selected a subset of near-frontal images (yaw angle in [−30°, 30°]) containing 34,259 faces in total. The database contains challenging "in-the-wild" images exhibiting a large variation in the resolution, illumination changes, race and background clutter. Fig. 6 shows a sample of male and female images.

The faces were split randomly three times into training, validation and testing part in the ratio 60/20/20. We made sure that images of the same identity never appear in different parts simultaneously.

The faces were localized by a commercial face detector[6] and consequently processed by a landmark detector [42]. The detected landmarks were used to transform the face into its canonical pose of

[6] Courtesy of Eydea Recognition Ltd, www.eyedea.cz

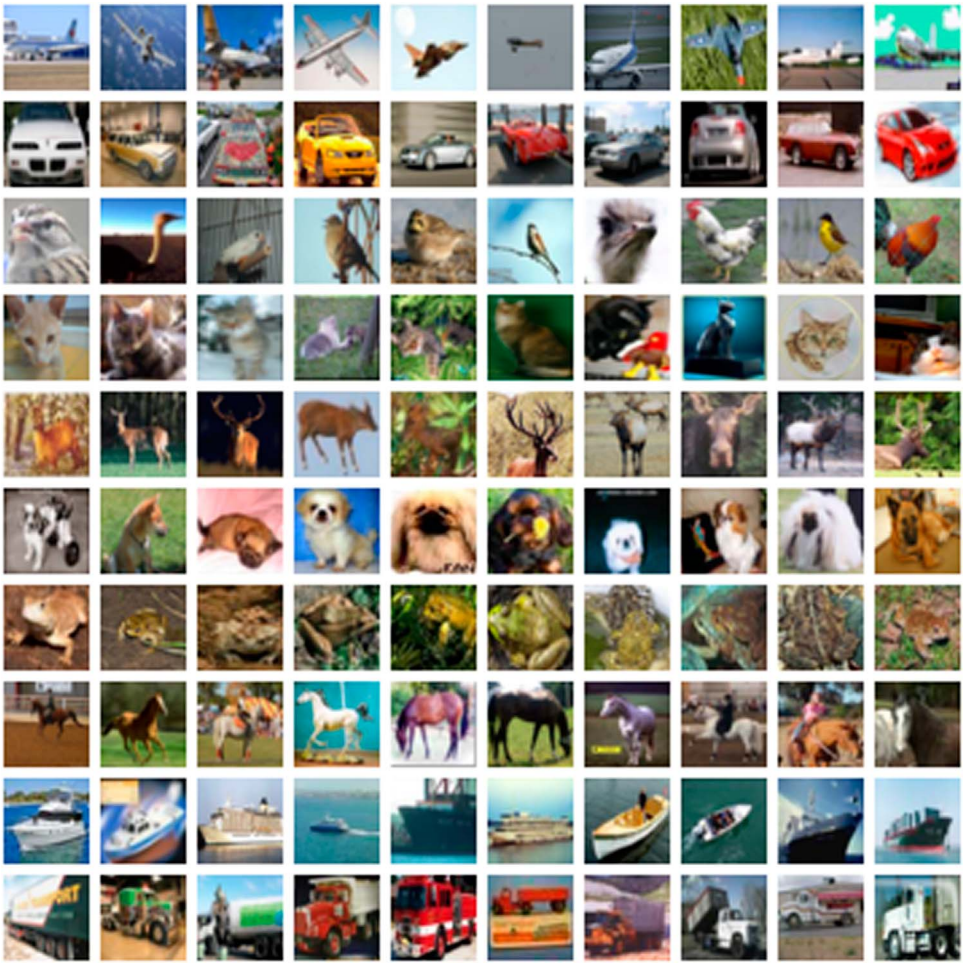


Fig. 5. Some selected images from Cifar10 dataset.

Table 4
Classification Rates on Cifar10 Dataset.

Method	Accuracy (%)
SVM	52.31 ± 0.4
TSVM^{CCCP}	–
WELLSVM	**54.82** ± 0.7
RTSVM	53.13 ± 0.6

size 60×40 pixels. We trained 2048-dimensional feature descriptor using a Convolutional Neural Network (CNN) with 7 convolution and 2 fully connected layers. The CNN was trained by MatConvNet toolbox [43] to classify the face into 12 age-gender categories (males/0–10 years, female/0–10 years, male/11–20 years, ...). The last layer was then replaced by a specialized linear classifier trained to recognize only the gender.

As a baseline we trained linear SVM from 3000 labeled examples. Then, we used additional ≈18,000 unlabeled images to train the transductive learning methods. The results are summarized in Table 5. The proposed RTSVM achieves the best result followed by TSVM^{CCCP}. WELLSVM is the worst performing method, its accuracy is even lower than accuracy of SVM.

3.3.3. Epsilon dataset experiments

Epsilon dataset (downloaded from https://www.csie.ntu.edu.tw~cjlinlibsvmtoolsdatasets) includes 2000-dimensional samples of two classes. There are 400 K samples in the training set and 100 K samples in the test set. We used training set as labeled data and test set unlabeled data in our experiments, so the transductive methods were run on a data with size 0.5 million which is very large. Therefore, we could not get results with TSVM^{CCCP} method since it did not return any solution in 10 days. The accuracies are given in Table 6. The proposed method achieves the best accuracy followed by WELLSVM. The accuracies of all tested methods are very close.

3.3.4. SensIT vehicle dataset experiments

SensIT Vehicle dataset (downloaded from https://www.csie.ntu.edu.tw~cjlinlibsvmtoolsdatasets) includes 100-dimensional samples of three classes. There are 78,823 samples in the training set and 19,705 samples in the test set. We used the training set as labeled data and test set unlabeled data in our experiments as before. The accuracies are given in Table 7. TSVM^{CCCP} achieved the best accuracy followed by RTSVM. In terms of the speed, the training time of the proposed method is 27.7 min whereas the training times of WELLSVM and TSVM^{CCCP} are respectively 35.3 min and 256.3 min. So, the proposed method is the most efficient one.

3.3.5. MNIST experiments

MNIST dataset includes hand-written images of 10-digit classes. There are 60 K samples in the training set and 10 K samples in the test set. By using a similar setting as in [11], we subsampled 1000 labeled samples per class from the training and used the remaining 50 K samples as unlabeled data. Accuracies are computed by using the test samples that are not used during training. The results are given in Table 8. The proposed RTSVM method achieves the best accuracy. It is also the most efficient one in terms of training time: it is 1.6 times

Fig. 6. Samples of male (left) and female (right) images from "in-the-wild" dataset used for gender estimation experiments.

Table 5
Classification accuracy of the gender estimation on a collection of "in-the-wild" images.

Method	**Accuracy** (%)
SVM	89.61 ± 0.86
TSVM^{CCCP}	90.14 ± 0.64
WELLSVM	82.37 ± 2.38
RTSVM	**91.09** ± 1.54

Table 6
Classification Rates on Epsilon Dataset.

Method	**Accuracy** (%)
SVM	89.75
TSVM^{CCCP}	– –
WELLSVM	89.78
RTSVM	**89.83**

Table 7
Classification Rates on SensIT Vehicle Dataset.

Method	**Accuracy** (%)
SVM	80.19
TSVM^{CCCP}	**82.04**
WELLSVM	81.01
RTSVM	81.85

Table 8
Classification Rates on MNIST Digit Dataset.

Method	**Accuracy** (%)
SVM	90.92
TSVM^{CCCP}	91.03
WELLSVM	84.37
RTSVM	91.23

faster than WELLSVM and 258 times faster than TSVM^{CCCP}. WELLSVM is the worst performing method.

3.4. Comparison of training time

Here we conducted experiments on a synthetic binary dataset to compare the training times of the transductive learning algorithms. To this end, we created 100-dimensional data samples drawn from a Gaussian distribution with the axis-aligned standard deviations, where the standard deviations are chosen from a uniform distribution between 0 and 1. Both classes have the same covariance structure but the class means are chosen to be different to give a slight overlap between these two classes. We used 2 K labeled samples and 8 K unlabeled samples since some of the tested transductive methods are not suitable for data sets with more than 10 K samples. Both accuracies

Table 9
Classification Rates on Synthetic Gaussian Dataset.

Method	Accuracy (%)	Training Time (sec)
TSVM^{Light}	86.29	11192
TSVM^{LDS}	80.00	1521
TSVM^{CCCP}	86.34	2867
WELLSVM	75.33	285
RTSVM	86.55	38

and training times (in terms of seconds) are given in Table 9. All tests are conducted on a workstation using 2.60 GHz Intel Xeon CPU and 256 GB RAM. Our proposed RTSVM method is the most efficient one in terms of both accuracy and training time. WELLSVM yields the worst accuracy, but it is the second fastest method. TSVM^{Light} is the slowest method in terms of the training time. It should be noted that the proposed RTSVM, WELLSVM and TSVM^{CCCP} are the only methods that are practical for larger data sets and the proposed method is approximately 7.5 times faster than WELLSVM and 75.4 times faster than TSVM^{CCCP}.

4. Conclusion

In this work, we introduced a robust and fast transductive SVM classifier that can scale well with large-scale data. To this end, we replaced the Hinge loss with the robust Ramp loss that suppresses the influence of outlying points. The resulting optimization problem is non-convex, but we used concave-convex procedure to solve the problem since it can be decomposed into a concave and convex parts. In contrast to [11], we considered the primal space and the solved the optimization problem with SG method. This enabled us to apply the proposed classifier to large-scale data easily.

The classification accuracies of the proposed method were similar to the best performing tested transductive SVM methods for clean data, but our classifier significantly outperformed all other tested classifiers for noisy data. We also successfully applied our classifier to larger data sets where the application of other transductive SVM classifiers was not feasible. Lastly, we have observed that an embedding of samples using local neighborhood information significantly improves accuracies of transductive SVMs for some data sets. Therefore, including an additional term which forces to preserve local neighborhood relations among samples into the optimization problem of transductive SVMs may yield to better results. As a future work, we are planning to work on this problem.

Acknowledgements

This work was funded by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number EEEAG-113E118. VF was supported by the project ERC-CZ LL1303.

Appendix A. Derivation of the dual RTSVM classifier

Consider that we are given a set of L labeled training samples $L = \{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_L), y_L)\}$, $\mathbf{x} \in \mathrm{IR}^d$, $y \in \{+1, -1\}$ and an unlabeled set of U vectors $U = \{\Phi(\mathbf{x}_{L+1}), \ldots, \Phi(\mathbf{x}_{L+U})\}$. We augment the original unlabeled data as

$$y_i = +1,\ \ i \in [L+1, \ldots, L+U],\ y_i = -1,\ \ i \in [L+U+1, \ldots, L+2U],\ \Phi(\mathbf{x}_i) = \Phi(\mathbf{x}_{i-U}),\ \ i \in [L+U+1, \ldots, L+2U]. \tag{19}$$

By using the definition,

$$H_1(t) = \max(0, 1-t) = \min \xi \text{s. t. } \xi \geq 0, \xi \geq 1-t, \tag{20}$$

the convex optimization problem given in (13) can be re-written as

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=1}^{L+2U}\beta_i y_i(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) \text{s. t. } \frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) = \frac{1}{L}\sum_{i=1}^{L} y_i.y_i(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b)$$
$$\geq 1-\xi_i, \forall\, i \in [1, \ldots, L+2U] \xi_i \geq 0, \forall\, i \in [1, \ldots, L+2U] \tag{21}$$

To derive the dual, we introduce the Lagrangian variables

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\nu}) \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=1}^{L+2U}\beta_i y_i(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) - \alpha_0\left(\frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) - \frac{1}{L}\sum_{i=1}^{L} y_i\right) - \sum_{i=1}^{L+2U}\alpha_i(y_i(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) - 1 + \xi_i)$$
$$- \sum_{i=1}^{L+2U}\nu_i\xi_i, \tag{22}$$

where $\alpha_i, \nu_i \geq 0$ for $i \geq 1$. The optimality conditions yield

$$\frac{\partial \mathrm{L}}{\partial \mathbf{w}} = \mathbf{0} \rightarrow \mathbf{w} = \sum_{i=1}^{L+2U} y_i(\alpha_i - \beta_i)\Phi(\mathbf{x}_i) - \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\Phi(\mathbf{x}_i) \frac{\partial \mathrm{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{L+2U} y_i(\alpha_i - \beta_i) + \alpha_0 = 0, \frac{\partial \mathrm{L}}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq C, \forall\, i \in [1, \ldots, L]$$
$$\frac{\partial \mathrm{L}}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq C^*, \forall\, i \in [L+1, \ldots, L+2U]. \tag{23}$$

For simplification, we define a new sample

$$\Phi(\mathbf{x}_0) = \frac{1}{U}\sum_{i=L+1}^{L+U}\Phi(\mathbf{x}_i) \tag{24}$$

with corresponding label $y_0 = 1$ and $\beta_0 = 0$. Then, by defining

$$\mathbf{w} = \sum_{i=0}^{L+2U} y_i(\alpha_i - \beta_i)\Phi(\mathbf{x}_i), \tag{25}$$

the convex optimization problem in (21) can be written as the following constrained quadratic programming problem similar to the SVM dual optimization problem

$$\arg\min_{\alpha}\frac{1}{2}\sum_{i,j=0}^{L+2U} y_i y_j(\alpha_i-\beta_i)(\alpha_j-\beta_j)\langle \Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle - \sum_{i=1}^{L+2U}\alpha_i - \alpha_0\left(\frac{1}{L}\sum_{i=1}^{L} y_i\right), \text{ s. t. } 0\le\alpha_i\le C,\ \forall\, i\in[1,\ldots,L]$$
$$, 0\le\alpha_i\le C^*,\ \forall\, i\in[L+1,\ldots,L+2U],\ \sum_{i=0}^{L+2U} y_i(\alpha_i-\beta_i)=0, \tag{26}$$

where the operator $\langle . \rangle$ defines the inner product. If we define $\zeta = y_i$ for $1\le i\le L+2U$ and $\zeta_0 = (1/L)\sum_{i=1}^{L} y_i$ and define the kernel matrix $\mathbf{K}$ such that

$$K_{ij} = \langle \Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle \tag{27}$$

and

$$\widetilde{\alpha}_i = y_i(\alpha_i-\beta_i), \tag{28}$$

the optimization problem given in (26) can be re-written in a more compact form as

$$\arg\min_{\alpha}\frac{1}{2}\widetilde{\boldsymbol{\alpha}}^{\top}\mathbf{K}\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\zeta}^{\top}\widetilde{\boldsymbol{\alpha}} \text{ s. t. } 0\le y_i\widetilde{\alpha}_i\le C,\ \forall\, i\in[1,\ldots,L], -\beta_i\le y_i\widetilde{\alpha}_i\le C^*-\beta_i,\ \forall\, i\in[L+1,\ldots,L+2U],\ \sum_{i=0}^{L+2U}\widetilde{\alpha}_i=0. \tag{29}$$

It should be noted that only the bounds in (29) on the $\widetilde{\alpha}_i$ have to be adjusted after each update of $\boldsymbol{\beta}$. To extend the method to the nonlinear case, we just have to replace the inner products with the defined kernel functions, i.e., $k(\mathbf{x}_i,\mathbf{x}_j) = \langle \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)\rangle$.

Once we find the $\widetilde{\alpha}_i$ coefficients by SMO, we compute the hyperplane normal by using (25). Then, the hyperplane bias parameter b can be obtained by using constraints as follows

$$\forall\, i\in[1,\ldots,L],\ 0<\alpha_i<C \Rightarrow y_i(\mathbf{w}^{\top}\Phi(\mathbf{x}_i)+b)=1, \tag{30}$$

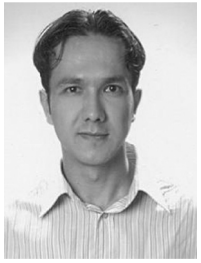
or

$$\forall\, i\in[L+1,\ldots,L+2U],\ 0<\alpha_i<C^* \Rightarrow y_i(\mathbf{w}^{\top}\Phi(\mathbf{x}_i)+b)=1. \tag{31}$$

References

[1] C. Cortes, V. Vapnik, Support vector networks, Mach. Learn. 20 (1995) 273–297.
[2] H. Cevikalp, B. Triggs, Hyperdisk based large margin classifier, Pattern Recognit. 46 (2013) 1523–1531.
[3] H. Cevikalp, B. Triggs, H.S. Yavuz, Y. Kucuk, M. Kucuk, A. Barkana, Large margin classifiers based on affine hulls, Neurocomputing 73 (2010) 3160–3168.
[4] H. Cevikalp, B. Triggs, F. Jurie, R. Polikar, Margin-based discriminant dimensionality reduction for visual recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008.
[5] V. Vapnik, A. Sterin, On structural risk minimization or overall risk in a problem of pattern recognition, Autom. Remote Control 10 (1977) 1495–1503.
[6] K. Bennett, A. Demiriz, Semi-supervised support vector machines, Neural Inf. Process. Syst. (1998).
[7] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the International Conference on Machine Learning, 1999.
[8] O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in: Proceedings of the 10th International Workshop on Artifical Intelligence and Statistics, 2005.
[9] T.D. Bie, N. Cristianini, Convex methods for transduction, in: Neural Information Processing Systems, 2003.
[10] Z. Xu, R. Jin, J. Zhu, I. King, M.R. Lyu, Efficient convex relaxation for transductive support vector machine, in: Neural Information Processing Systems, 2008.
[11] R. Collobert, F. Sinz, J. Weston, L. Bottou, Large scale transductive SVMs, J. Mach. Learn. Res. 7 (2006) 1687–1712.
[12] J. Wang, X. Shen, W. Pan, On efficient large margin semisupervised learning: method and theory, J. Mach. Learn. Res. 10 (2009) 719–742.
[13] A.L. Yullie, A. Rangarajan, The concave-convex procedure (CCCP), in: Neural Information Processing Systems, 2002.
[14] Y.-F. Li, I.W. Tsang, J.T. Kwok, Z.-H. Zhou, Convex and scalable weakly labeled SVMs, J. Mach. Learn. Res. 14 (2013) 2151–2188.
[15] V. Sindhwani, S.S. Keerthi, O. Chapelle, Deterministic annealing for semi-supervised kernel machines, in: Proceedings of the International Conference on Machine Learning, 2006.
[16] V. Sindhwani, S.S. Keerthi, Large scale semi-supervised linear SVMs, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006.
[17] O. Chapelle, V. Sindhwani, S.S. Keerthi, Branch and bound for semi-supervised support vector machines, in: Neural Information Processing Systems, 2007.
[18] A. Astorino, A. Fuduli, Nonsmooth optimization techniques for semisupervised classification, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 2135–2142.
[19] O. Chapelle, M. Chi, A. Zien, A continuation method for semi-supervised SVMs, in: Proceedings of the International Conference on Machine Learning, 2006.
[20] A.N. Erkan, Y. Altun, Semi-supervised learning via generalized maximum entropy, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2010.
[21] X. Wang, J. Wen, S. Alam, Z. Jiang, Y. Wu, Semi-supervised learning combining transductive support vector machine with active learning, Neurocomputing 173 (2016) 1288–1298.
[22] S. Scardapane, D. Comminiello, M. Scarpiniti, A. Uncini, A semi-supervised random vector functional-link network based on the transductive framework, Inf. Sci. 364 (2016) 156–166.
[23] Z. Sun, C. Wang, D. Li, J. Li, Semisupervised classification for hyperspectral imagery with transductive multiple-kernel learning, IEEE Geosci. Remote Sens. Lett. 11 (2014) 1991–1995.
[24] P. Zhang, T. Zhuo, Y. Zhang, D. Tao, J. Cheng, Online tracking based on efficient transductive learning with sample matching costs, Neurocomputing 175 (2016) 166–176.
[25] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Tech. rep., Carnegie Mellon University (2002).
[26] B.S.D. Zhou, J. Huang, Learning from labeled and unlabeled data on a directed graphs, in: Proceedings of the International Conference on Machine Learning, 2005.
[27] P. Cheng, Y. Qiu, K. Zhao, X. Wang, A transductive graphical model for single image super-resolution, Neurocomputing 148 (2015) 376–387.
[28] L. Wang, S. Hao, Q. Wang, Y. Wang, Semi-supervised classification for hyperspectral imagery based on spatial-spectral label propagation, ISPRS J. Photogramm. Remote Sens. 97 (2014) 123–137.
[29] C. Hou, L. Jiao, Y. Hou, A regularization framework in polar coordinates for transductive learning in networked data, Inf. Sci. 221 (2013) 262–273.
[30] O. Chapelle, V. Sindhwani, S.S. Keerthi, Optimization techniques for semi-supervised support vector machines, J. Mach. Learn. Res. 9 (2008) 203–233.
[31] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, 1999, pp. 185–208.
[32] S. Ertekin, L. Bottou, C.L. Giles, Nonconvex online support vector machines, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 368–381.
[33] L. Wang, H. Jia, J. Li, Training robust support vector machine with smooth ramp loss in the primal space, Neurocomputing 71 (2008) 3020–3025.
[34] L. Xu, K. Crammer, D. Schuurmans, Robust support vector machine training via convex outlier ablation, in: Proceedings of the 21 st National Conference on Artifical Intelligence, 2006.
[35] R. Collobert, F. Sinz, J. Weston, L. Bottou, Trading convexity for scalability, in: Proceedings of the International Conference on Machine Learning, 2006.
[36] S. Shalev-Shwartz, Y. Singer, N. Srebro, Pegasos: Primal estimated sub-gradient solver for SVM, in: Neural Information Processing Systems, 2007.
[37] O. Chapell, Training a support vector machine in the primal, Neural Comput. 19 (2007) 1155–1178.
[38] J. Sanchez, F. Perronnin, T. Mensink, J. Verbeek, Image classification with the fisher vector: theory and practice, Int. J. Comput. Vis. 34 (2013) 1704–1716.

[39] G.B. Huang, M. Ramesh, T. Berg, E. Learned-Miller, Labeled faces in the wild: A database for studying face recognition in unconstrained environments, Tech. Rep. 07–49, University of Massachusetts, Amherst (October 2007).
[40] N. Kumar, A.C. Berg, P.N. Belhumeur, S.K. Nayar, Attribute and Simile Classifiers for Face Verification, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2009.
[41] M. Minear, D. Park, A lifespan database of adult facial stimuli, Behav. Res. Methods Instrum. Comput.: J. Psychon. Soc. 36 (2004) 630–633.
[42] M. Uřičář, V. Franc, V. Hlaváč, Detector of facial landmarks learned by the structured output SVM, in: G. Csurka, J. Braz (Eds.), VISAPP '12 in: Proceedings of the 7th International Conference on Computer Vision Theory and Applications, Vol. 1, SciTePress - Science and Technology Publications, Porto, Portugal, 2012, pp. 547–556.
[43] A. Vedaldi, K. Lenc, Matconvnet – convolutional neural networks for matlab, 2015.



Hakan Cevikalp received his M.S. degree from the Department of Electrical and Electronics Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey, in 2001 and the Ph. D. degree from the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, in 2005. He worked as a post-doctoral researcher at LEAR team of INRIA Rhone-Alpes in France in 2007 and Rowan University in USA in 2008. He is currently working as an assistant professor in the Department of Electrical and Electronics Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey. His research interests include pattern recognition, neural networks, image and signal processing, optimization, and computer vision.

Vojtech Franc received the M.S. and the Ph.D. degree from the Faculty of Electrical Engineering at the Czech Technical University in Prague. He is currently working as an assistant professor at the Department of Cybernetics of the CTU Prague. His research interests include machine learning, pattern recognition, optimization and computer vision.