Journal of Machine Learning Research 1 (2005) 0-0 Submitted 12/05; Published ?

Large Scale Transductive SVMs

Ronan Collobert RONAN@COLLOBERT.COM
NEC Laboratories America,
4 Independence Way, Princeton, NJ08540, USA.

Fabian Sinz FABEE@TUEBINGEN.MPG.DE
NEC Laboratories America,
4 Independence Way, Princeton, NJ08540, USA,
and Max Planck Insitute for Biological Cybernetics,
Spemannstrasse 38, 72076 Tuebingen, Germany.

Jason Weston JASEWESTON@GMAIL.COM
NEC Laboratories America,
4 Independence Way, Princeton, NJ08540, USA.

Léon Bottou LEON@BOTTOU.ORG
NEC Laboratories America,
4 Independence Way, Princeton, NJ08540, USA.

Editor: Unknown

Abstract

We show how the Concave-Convex Procedure can be applied to Transductive SVMs, which traditionally requires solving a combinatorial search problem. This provides for the first time a highly scalable algorithm in the nonlinear case. Detailed experiments verify the utility of our approach. Software is available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.

Keywords: transduction, transductive SVMs, semi-supervised learning, CCCP

1. Introduction

Transductive support vector machines (TSVMs) (Vapnik, 1995) are a method of improving the generalization accuracy of SVMs (Boser et al., 1992) by using unlabeled data. TSVMs, like SVMs, learn a large margin hyperplane classifier using labeled training data, but simultaneously force this hyperplane to be far away from the unlabeled data.

One way of justifying this algorithm, in the context of *semi-supervised learning* is that one is finding a decision boundary that lies in a region of low density, implementing the so-called cluster assumption (see e.g. Chapelle and Zien, 2005). In this framework, if you believe the underlying distribution of the two classes is such that there is a "gap" or low density region between them, then TSVMs can help because it selects a rule with exactly those properties. Vapnik (1995) has a different interpretation for the success of TSVMs, rooted in the idea that transduction (labeling a test set) is inherently easier than induction (learning a general rule). In either case, experimentally it seems clear that algorithms such as TSVM can give considerable improvement in generalization over SVMs, if the number of labeled points is small and the number of unlabeled points is large.

Unfortunately, TSVM algorithms (like other semi-supervised approaches) are often unable to deal with a large number of unlabeled examples. The first implementation of TSVM appeared in (Bennett and Demiriz, 1998), using an integer programming method, intractable for large problems. Joachims (1999b) then proposed a combinatorial approach, known as SVMLight-TSVM, that is practical for a few thousand examples. Fung and Mangasarian (2001) introduced a sequential optimization procedure that could potentially scale well, although their largest experiment used only 1000 examples. However, their method was for the linear case only, and used a special kind of SVM with a 1-norm regularizer, to retain linearity. Finally, Chapelle and Zien (2005) proposed a primal method, which turned out to show improved generalization performance over the previous approaches, but still scales as $(L+U)^3$, where L and U are the numbers of labeled and unlabeled examples. This method also stores the entire $(L+U)\times(L+U)$ kernel matrix in memory. Other methods (Bie and Cristianini, 2004; Xu et al., 2005) transform the non-convex transductive problem into a convex semi-definite programming problem that scales as $(L+U)^4$ or worse.

In this article we introduce a large scale training method for TSVMs using the Concave-Convex Procedure (CCCP) (Yuille and Rangarajan, 2002; Le Thi, 1994). CCCP iteratively optimizes non-convex cost functions that can be expressed as the sum of a convex function and a concave function. The optimization is carried out iteratively by solving a sequence of convex problems obtained by linearly approximating the concave function in the vicinity of the solution of the previous convex problem. This method is guaranteed to find a local minimum and has no difficult parameters to tune. This provides what we believe is the best known method for implementing Transductive SVM with an empirical scaling of $(L+U)^2$, which involves training a sequence of typically 1-10 conventional convex SVM optimization problems. As each of these problems is trained in the dual we retain the SVMs linear scaling with problem dimensionality, in contrast to the techniques of Fung and Mangasarian (2001).

2. The Concave-Convex Procedure for TSVMs

Notation We consider a set of L training pairs $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_L, y_L)\}$, $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \{1, -1\}$ and a (unlabeled) set of U test vectors $\mathcal{U} = \{x_{L+1}, \ldots, x_{L+U}\}$. SVMs have a decision function $f_\theta(.)$ of the form

$$f_\theta(x) = w \cdot \Phi(\boldsymbol{x}) + b\,,$$

where $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ are the parameters of the model, and $\Phi(\cdot)$ is the chosen feature map, often implemented implicitly using the kernel trick (Vapnik, 1995).

TSVM Formulation The original TSVM optimization problem is the following (Vapnik, 1995; Joachims, 1999b; Bennett and Demiriz, 1998). Given a training set $\mathcal{L}$ and a test set $\mathcal{U}$, find among the possible binary vectors

$$\{\mathcal{Y} = (y_{L+1}, \ldots, y_{L+U})\}$$

the one such that an SVM trained on $\mathcal{L} \cup (\mathcal{U} \times \mathcal{Y})$ yields the largest margin.

This is a combinatorial problem, but one can approximate it (see Vapnik, 1995) as finding a SVM separating the training set under constraints which force the unlabeled examples to

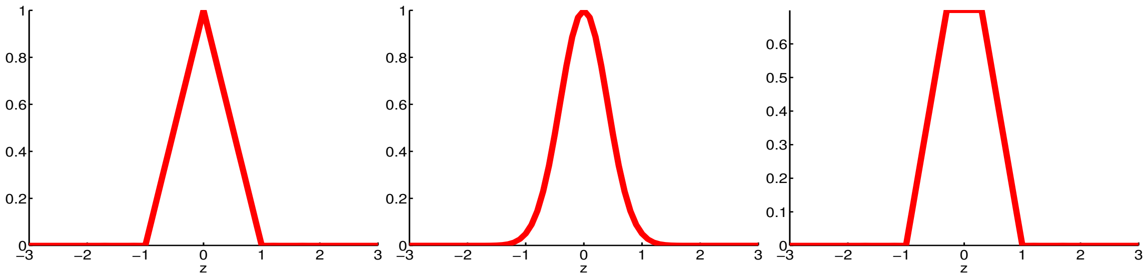

Figure 1: Three loss functions for unlabeled examples, from left to right (i) the Symmetric Hinge $H_1(|t|) = \max(0,\, 1 - |t|)$, (ii) Symmetric Sigmoid $S(t) = \exp(-3t^2)$; and (iii) Symmetric Ramp loss, $R_s(|t|) = \min(1 + s, \max(0, 1 - |t|))$. The last loss function has a plateau of width $2|s|$ where $s \in (-1, 0]$ is a tunable parameter, in this case $s = -0.3$.

be as far as possible from the margin. This can be written as minimizing

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+U}\xi_i$$

subject to

$$y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 - \xi_i, \quad i = 1, \ldots, L$$
$$|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)| \geq 1 - \xi_i, \quad i = L+1, \ldots, L+U$$

This minimization problem is equivalent to minimizing

$$J(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+U} H_1(|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|), \tag{1}$$

where the function $H_1(\cdot) = \max(0,\, 1 - \cdot)$ is the classical Hinge Loss (Figure 2, center). The loss function $H_1(|\cdot|)$ for the unlabeled examples can be seen in Figure 1, left. For $C^* = 0$ in (1) we obtain the standard SVM optimization problem. For $C^* > 0$ we penalize unlabeled data that is inside the margin. This is equivalent to using the hinge loss on the unlabeled data as well, but where we assume the label for the unlabeled example is $y_i = \text{sign}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$.

Losses for transduction TSVMs implementing formulation (1) were first introduced in SVMLight (Joachims, 1999b). As shown above, it assigns a Hinge Loss $H_1(\cdot)$ on the labeled examples (Figure 2, center) and a "Symmetric Hinge Loss" $H_1(|\cdot|)$ on the unlabeled examples (Figure 1, left). More recently, Chapelle and Zien (2005) proposed to handle unlabeled examples with a smooth version of this loss (Figure 1, center). While we also use the Hinge Loss for labeled examples, we use for unlabeled examples a slightly more general form of the Symmetric Hinge Loss, that we allow to be "non-peaky" (Figure 1, right). Given an unlabeled example $\boldsymbol{x}$ and using the notation $z = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, this loss can be written as

$$z \mapsto R_s(z) + R_s(-z) + const.^1\,, \tag{2}$$

where $-1 < s \leq 0$ is a hyper-parameter to be chosen and $R_s = \min(1 - s, \max(0, 1 - t))$ is what we call the "Ramp Loss", a "clipped" version of the Hinge Loss (Figure 2, left).

1. The constant does not affect the optimization problem we will later describe.

Losses similar to the Ramp Loss have been already used for different purposes, like in the Doom II algorithm (Mason et al., 2000) or in the context of "Ψ-learning" (Shen et al., 2003). The s parameter controls where we clip the Ramp Loss, and as a consequence it also controls the wideness of the flat part of the loss (2) we use for transduction: when $s = 0$, this reverts to the Symmetric Hinge $H_1(|\cdot|)$. When $s \neq 0$, we obtain a non-peaked loss function (Figure 1, right) which can be viewed as a simplification of Chapelle's loss function. We call this loss function (2) the "Symmetric Ramp Loss".

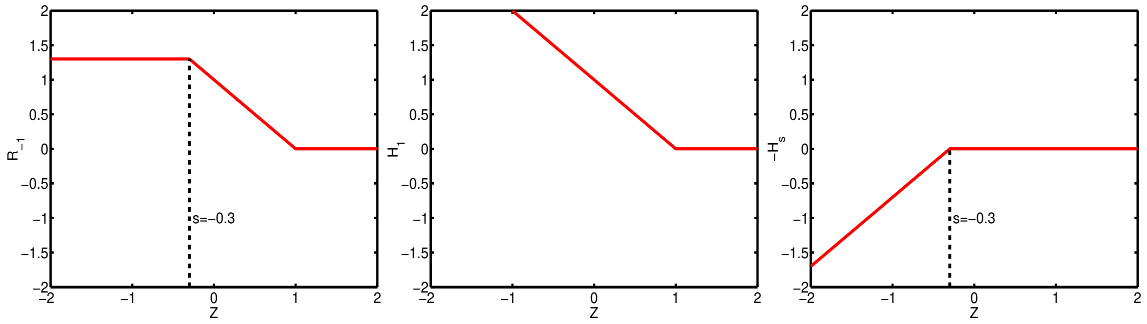

Figure 2: The Ramp Loss function $R_s(t) = \min(1-s, \max(0, 1-t)) = H_1(t) - H_s(t)$ (left) can be decomposed into the sum of the convex Hinge Loss (center) and a concave loss (right), where $H_s(t) = \max(0, s-t)$. The parameter s controls the cutoff point of the usual Hinge loss.

Training a TSVM using the loss function (2) is equivalent to training a SVM using the Hinge loss $H_1(\cdot)$ for labeled examples, and using the Ramp loss $R_s(\cdot)$ for unlabeled examples, where each unlabeled example appears as two examples labeled with both possible classes. More formally, after introducing

$$\begin{array}{rclr} y_i & = & 1 & i \in [L+1 \ldots L+U] \\ y_i & = & -1 & i \in [L+U+1 \ldots L+2U] \\ \boldsymbol{x}_i & = & \boldsymbol{x}_{i-U} & i \in [L+U+1 \ldots L+2U]\,, \end{array}$$

we can rewrite (1) as

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{3}$$

This is the minimization problem we now consider in the rest of the paper.

Balancing constraint One problem with TSVM as stated above is that in high dimensions with few training data, it is possible to classify all the unlabeled examples as belonging to only one of the classes with a very large margin, which leads to poor performance. To cure this problem, one further constrains the solution by introducing a balancing constraint that ensures the unlabeled data are assigned to both classes. Joachims (1999b) directly enforces that the fraction of positive and negatives assigned to the unlabeled data should be the same fraction as found in the labeled data. Chapelle and Zien (2005) use a similar but slightly relaxed constraint, which we also use in this work:

$$\frac{1}{U} \sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \;=\; \frac{1}{L} \sum_{i=1}^{L} y_i\,. \tag{4}$$

Concave-Convex Procedure (CCCP) Unfortunately, the TSVM optimization problem as given above is not convex, and minimizing a non-convex cost function is often considered difficult. Gradient descent techniques, such as conjugate gradient descent or stochastic gradient descent, often involve delicate hyper-parameters (LeCun et al., 1998). In contrast, convex optimization seems much more straight-forward. For instance, the SMO algorithm (Platt, 1999) locates the SVM solution efficiently and reliably.

We propose to solve this non convex problem using the "Concave-Convex Procedure" (CCCP) (Yuille and Rangarajan, 2002). The CCCP procedure is closely related to the "Difference of Convex" (DC) methods that have been developed by the optimization community during the last two decades (Le Thi, 1994). Such techniques have already been applied for dealing with missing values in SVMs (Smola et al., 2005), for improving boosting algorithms (Krause and Singer, 2004), and in the "Ψ-learning" framework (Shen et al., 2003).

Assume that a cost function $J(\boldsymbol{\theta})$ can be rewritten as the sum of a convex part $J_{vex}(\boldsymbol{\theta})$ and a concave part $J_{cav}(\boldsymbol{\theta})$. Each iteration of the CCCP procedure (Algorithm 1) approximates the concave part by its tangent and minimizes the resulting convex function.

Algorithm 1 : The Concave-Convex Procedure (CCCP)

Initialize $\boldsymbol{\theta}^0$ with a best guess.
repeat

$$\boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \left(J_{vex}(\boldsymbol{\theta}) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta} \right) \tag{5}$$

until convergence of $\boldsymbol{\theta}^t$

One can easily see that the cost $J(\boldsymbol{\theta}^t)$ decreases after each iteration by summing two inequalities resulting from (5) and from the concavity of $J_{cav}(\boldsymbol{\theta})$.

$$J_{vex}(\boldsymbol{\theta}^{t+1}) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta}^{\boldsymbol{t+1}} \leq J_{vex}(\boldsymbol{\theta}^t) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta}^{\boldsymbol{t}} \tag{6}$$

$$J_{cav}(\boldsymbol{\theta}^{t+1}) \leq J_{cav}(\boldsymbol{\theta}^t) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \left(\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \right) \tag{7}$$

The convergence of CCCP has been shown by Yuille and Rangarajan (2002) by refining this argument. The authors also showed that the CCCP procedure remains valid if $\boldsymbol{\theta}$ is required to satisfy some linear constraints. Note that no additional hyper-parameters are needed by CCCP. Furthermore, each update (5) is a convex minimization problem and can be solved using classical and efficient convex algorithms.

CCCP for TSVMs Interestingly, the Ramp Loss can be rewritten as the difference between two Hinge losses (see Figure 2):

$$R_s(z) = H_1(z) - H_s(z)\,. \tag{8}$$

Because of this decomposition, the TSVM minimization problem as stated in (3) is amenable to CCCP optimization. The cost $J^s(\boldsymbol{\theta})$ can indeed be decomposed into a convex $J^s_{vex}(\boldsymbol{\theta})$

and concave $J^s_{cav}(\boldsymbol{\theta})$ part as follows:

$$\begin{aligned} J^s(\boldsymbol{\theta}) &= \frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \\ &= \underbrace{\frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{vex}(\boldsymbol{\theta})} \\ &\quad \underbrace{-C^* \sum_{i=L+1}^{L+2U} H_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{cav}(\boldsymbol{\theta})} . \end{aligned} \tag{9}$$

This decomposition allows us to apply the CCCP procedure as stated in Algorithm 1. The convex optimization problem (5) that constitutes the core of the CCCP algorithm is easily reformulated into dual variables $\boldsymbol{\alpha}$ using the standard SVM technique.

After some algebra, we show in Appendix A that enforcing the balancing constraint (4) can be achieved by introducing an extra Lagrangian variable α_0 and an example $\boldsymbol{x}_0$ implicitely defined by

$$\Phi(\boldsymbol{x}_0) = \frac{1}{U} \sum_{i=L+1}^{L+U} \Phi(\boldsymbol{x}_i)\,,$$

with label $y_0 = 1$. Thus, if we note K the kernel matrix such that

$$K_{ij} = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)\,,$$

the column corresponding to the example $\boldsymbol{x}_0$ is computed as follow:

$$K_{i0} = K_{0i} = \frac{1}{U} \sum_{j=L+1}^{L+U} \Phi(\boldsymbol{x}_j) \cdot \Phi(\boldsymbol{x}_i) \quad \forall i\,. \tag{10}$$

The computation of this special column can be achieved very efficiently by computing it only once, or by approximating the sum (10) using an appropriate sampling method.

Given the decomposition of the cost (9) and the trick of the special extra example (10) to enforce the balancing constraint, we can easily apply Algorithm 1 to TSVMs. To simplifiy the first order approximation of the concave part in the CCCP procedure (5), we denote[2]

$$\beta_i \quad = \quad y_i \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)} \quad = \quad \begin{cases} C^* & \text{if } \; y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) < s \\ 0 & \text{otherwise} \end{cases} \quad , \tag{11}$$

for unlabeled examples (that is $i \geq L{+}1$). The concave part J^s_{cav} does not depend on labeled examples ($i \leq L$) so we obviously have $\beta_i = 0$ for all $i \leq L$. This yields Algorithm 2, after some standard derivations detailed in Appendix A.

2. Note that $J^s_{cav}(\cdot)$ is non-differentiable at $z = s$, because $H_s(\cdot)$ is not. It can be shown that the CCCP remains valid when using any super-derivative of the concave function. Alternatively, the function $H_s(\cdot)$ could be made smooth in a small interval $[s - \varepsilon, s + \varepsilon]$.

Algorithm 2 : CCCP for TSVMs

Initialize $\boldsymbol{\theta}^0 = (\boldsymbol{w}^0, b^0)$ with a standard SVM solution on the labeled points.

Compute $\beta_i^0 = \begin{cases} C^* & \text{if } \; y_i\, f_{\boldsymbol{\theta}^0}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases}$

Set $\zeta_i = y_i$ for $1 \leq i \leq L+2U$ and $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$

repeat

- **Solve** the following convex problem (with $K_{ij} = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$)

$$\max_{\boldsymbol{\alpha}} \left(\boldsymbol{\alpha} \cdot \boldsymbol{\zeta} - \frac{1}{2}\, \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{K}\, \boldsymbol{\alpha} \right) \text{ subject to } \begin{cases} \boldsymbol{\alpha} \cdot \mathbf{1} = 0 \\ 0 \leq y_i\, \alpha_i \leq C \quad \forall 1 \leq i \leq L \\ -\beta_i \leq y_i\, \alpha_i \leq C^* - \beta_i \quad \forall i \geq L+1 \end{cases}$$

- **Compute** b^{t+1} using $f_{\theta^{t+1}}(x_i) = \sum_{j=0}^{L+2U} \alpha_j^{t+1}\, K_{ij} \; + \; b^{t+1}$ and

$$\begin{array}{rcrcl} \forall i \leq L & : & 0 < y_i\, \alpha_i < C & \Longrightarrow & y_i\, f_{\theta^{t+1}}(x_i) = 1 \\ \forall i > L & : & -\beta_i < y_i\, \alpha_i < C^* - \beta_i & \Longrightarrow & y_i\, f_{\theta^{t+1}}(x_i) = 1 \end{array}$$

- **Compute** $\beta_i^{t+1} = \begin{cases} C^* & \text{if } \; y_i\, f_{\boldsymbol{\theta}^{t+1}}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases}$

until $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t$

Convergence of Algorithm 2 in finite time t^* is guaranteed because variable $\boldsymbol{\beta}$ can only take a finite number of distinct values, because $J(\boldsymbol{\theta}^t)$ is decreasing, and because inequality (7) is strict unless $\boldsymbol{\beta}$ remains unchanged.

Complexity The main point we want to emphasize in this paper is the advantage in terms of training time of our method compared to existing approaches. Training a CCCP-TSVM amounts to solve a series of SVM optimization problems with $L + 2U$ variables. Although SVM training has a worst case complexity of $\mathcal{O}((L + 2U)^3)$ it typically scales quadratically (see Joachims, 1999a; Platt, 1999), and we find this is the case for our TSVM subproblems as well. Assuming a constant number of iteration steps, the whole optimization of TSVMs with CCCP should scale quadratically in most practical cases (see Figure 3, Figure 8 and Figure 9). From our experience, around five iteration steps are usually sufficient to reach the minimum, as shown in the experimental section of this paper, Figure 4.

3. Previous Work

SVMLight-TSVM Like our work, the heuristic optimization algorithm implemented in SVMLight (Joachims, 1999b) solves successive SVM optimization problems, but on $L + U$ instead of $L + 2U$ data points. It improves the objective function by iteratively switching the labels of two unlabeled points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\xi_i + \xi_j > 2$. It uses two nested loops to optimize a TSVM which solves a quadratic program in each step. The convergence proof of the inner loop relies on the fact that there is only a finite number 2^U of labelings of U

unlabeled points, even though it is unlikely that all of them are examined. But since the heuristic only swaps the labels of two unlabeled examples at each step in order to enforce the balancing constraint it might need many iterations to reach a minimum, which makes it intractable for big dataset sizes in practice (cf. Figure 3).

SVMLight uses annealing heuristics for the selection of C^*. It begins with a small value of C^* ($C^* = 1e-5$), and multiplies C^* by 1.5 on each iteration until it reaches C. The numbers $1e-5$ and 1.5 are hard coded into the implementation. On each iteration the tolerance on the gradients is also changed so as to give more approximate (but faster) solutions on earlier iterations. Again, several heuristics parameters are hard coded into the implementation.

∇TSVM The ∇TSVM of Chapelle and Zien (2005) is optimized by performing gradient descent in the primal space: minimize

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H^2(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+U} H^*(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)),$$

where $H^2(t) = \max(0, 1-t)^2$ and $H^*(t) = \exp(-3t^2)$ (cf. Figure 1, center). This optimization problem can be considered a smoothed version of (1). ∇TSVM also has similar heuristics for C^* as SVMLight-TSVM. It begins with a small value of C^* ($C^* = bC$), and iteratively increases C^* over l iterations until it finally reaches C. The values $b = 0.01$ and $l = 10$ are default parameters in the code available at: `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds`.

Since the gradient descent is carried out in the primal, to learn nonlinear functions it is necessary to perform kernel PCA (Schölkopf et al., 1997). The overall algorithm has a time complexity equal to the square of the number of variables times the complexity of evaluating the cost function. In this case, evaluating the objective scales linearly in the number of examples, so the overall worst case complexity of solving the optimization problem for ∇TSVM is $\mathcal{O}((U+L)^3)$. The KPCA calculation alone also has a time complexity of $\mathcal{O}((U+L)^3)$. This method also requires to store the entire kernel matrix in memory, which clearly becomes infeasible for large datasets.

CS^{3}VM The work of Fung and Mangasarian (2001) is algorithmically the closest TSVM approach to our proposal. Following the formulation of transductive SVMs found in Bennett and Demiriz (1998), the authors consider transductive linear SVMs with a 1-norm regularizer, which allow them to decompose the corresponding loss function as a sum of a linear function and a concave function. Bennett proposed the following formulation which is similar to (1): minimize

$$||\boldsymbol{w}||_1 + C\sum_{i=1}^{L}\xi_i + C^* \sum_{i=L+1}^{U} \min(\xi_i, \xi_i^*)$$

subject to

$$y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 - \xi_i, \quad i = 1, \ldots, L$$

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 - \xi_i, \quad i = L+1, \dots, L+U$$
$$-(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i^*, \quad i = L+1, \dots, L+U$$
$$\xi_i \geq 0, \xi_i^* \geq 0.$$

The last term of the objective function is nonlinear and corresponds to the loss function given in Figure 1, left. To deal with this, the authors suggest to iteratively approximate the concave part as a linear function, leading to a series of linear programming problems. This can be viewed as a simplified subcase of CCCP (a linear function being convex) applied to a special kind of SVM. Note also that the algorithm presented in their paper did not implement a balancing constraint for the labeling of the unlabeled examples as in (4). Our transduction algorithm is nonlinear and the use of kernels, solving the optimization in the dual, allows for large scale training with high dimensionality and number of examples.

4. Small scale experiments

This section presents small scale experiments appropriate for comparing our algorithm with existing TSVM approaches. In order to provide a direct comparison with published results, these experiments use the same setup as (Chapelle and Zien, 2005). All methods use the standard RBF kernel, $\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}') = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2)$.

data set	classes	dims	points	labeled
g50c	2	50	500	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50

Table 1: Small-Scale Datasets. We used the same datasets and experimental setup in these experiments as found in Chapelle and Zien (2005).

Table 1 lists the datasets we have used. The `g50c` dataset is an artificial dataset where the labels correspond to two Gaussians in a 50-dimensional space. The means of those Gaussians are placed in such a way that the Bayes error is 5%. The `coil20` data is a set of gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees (S.A.Nene and H.Murase, 1996). The `text` dataset consists of the classes `mswindows` and `mac` of the `Newsgroup20` dataset preprocessed as in Szummer and Jaakkola (2001a). The `uspst` dataset is the test part of the `USPS` hand written digit data. All datasets are split into ten parts with each part having a small amount of labeled examples and using the remaining part as unlabeled data.

4.1 Accuracies

Consistent with (Chapelle and Zien, 2005), all hyperparameters are tuned on the test set. Chapelle and Zien (2005) argue that, in order to perform algorithm comparisons, it is sufficient to be "*interested in the best performance and simply select the parameter values minimizing the test error*". However we should be more cautious when comparing algorithms

	Coil20	g50c	Text	Uspst	(number of hyperparameters)
SVM	24.64	8.32	18.86	23.18	2
SVMLight-TSVM	26.26	6.87	7.44	26.46	2
∇TSVM	17.56	5.80	5.71	17.61	2
CCCP-TSVM$\|_{UC^*=LC}^{s=0}$	16.69	5.62	7.97	16.57	2
CCCP-TSVM$\|_{UC^*=LC}$	16.06	5.04	5.59	16.47	3
CCCP-TSVM	15.92	3.92	4.92	16.45	4

Table 2: Results on Small-Scale Datasets. We report the best test error over the hyperparameters of the algorithms, as in the methodology of Chapelle and Zien (2005). SVMLight-TSVM is the implementation in SVMLight. ∇TSVM is the primal gradient descent method of Chapelle and Zien (2005). CCCP-TSVM$|_{UC^*=LC}^{s=0}$ reports the results of our method using the heuristic $UC^* = LC$ with the Symmetric Hinge Loss, i.e. with $s = 0$. We also report CCCP-TSVM$|_{UC^*=LC}$ where we allow the optimization of s, and CCCP-TSVM where we allow the optimization of both C^* and s.

that have different sets of hyperparameters. For CCCP-TSVMs we have two additional parameters, C^* and s. Therefore we report the CCCP-TSVM error rates for three different scenarios:

- CCCP-TSVM, where all four parameters are tuned on the test set.
- CCCP-TSVM$|_{UC^*=LC}$ where we choose C^* using a heuristic method. We use heuristic $UC^* = LC$ because it decreases C^* when the number of unlabeled data increases. Otherwise, for large enough U no attention will be paid to minimizing the training error. Further details on this choice are given in Section 4.3.
- CCCP-TSVM$|_{UC^*=LC}^{s=0}$ where we choose $s = 0$ and C^* using heuristic $UC^* = LC$. This setup has the same free parameters (C and γ) as the competing TSVM implementations, and therefore provides the most fair comparison.

The results are reported in Table 1. CCCP-TSVM in all three scenarios achieves approximately the same error rates as ∇TSVM and appears to be superior to SVMLight-TSVM. Section 4.3 provides additional results using different hyperparameter selection strategies and discusses more precisely the impact of each hyperparameter.

4.2 Training times

At this point we ask the reader to simply assume that all authors have chosen their hyperparameter selection method as well as they could. We now compare the computation times of these three algorithms.

The CCCP-TSVM algorithm was implemented[3] in C++. The successive convex optimizations are performed using a state-of-the-art SMO implementation. Without further

3. Source code available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.

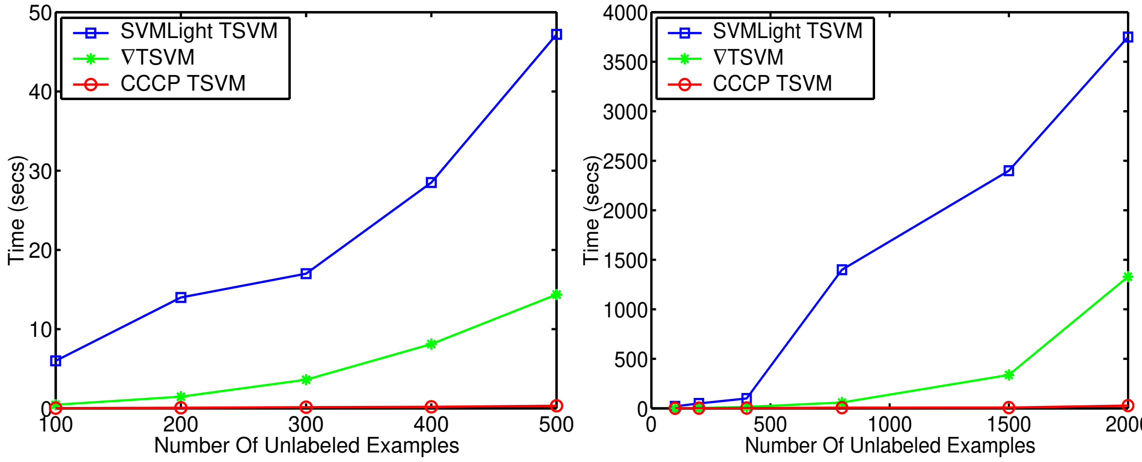

Figure 3: Training times for `g50c` (left) and `text` (right) for SVMLight-TSVMs, ∇TSVMs and CCCP-TSVMs using the best parameters for each algorithm as measured on the test set in a single trial. For the Text dataset, using 2000 unlabeled examples CCCP-TSVMs are **133x** faster than SVMLight-TSVMs, and **50x** faster than ∇TSVMs.

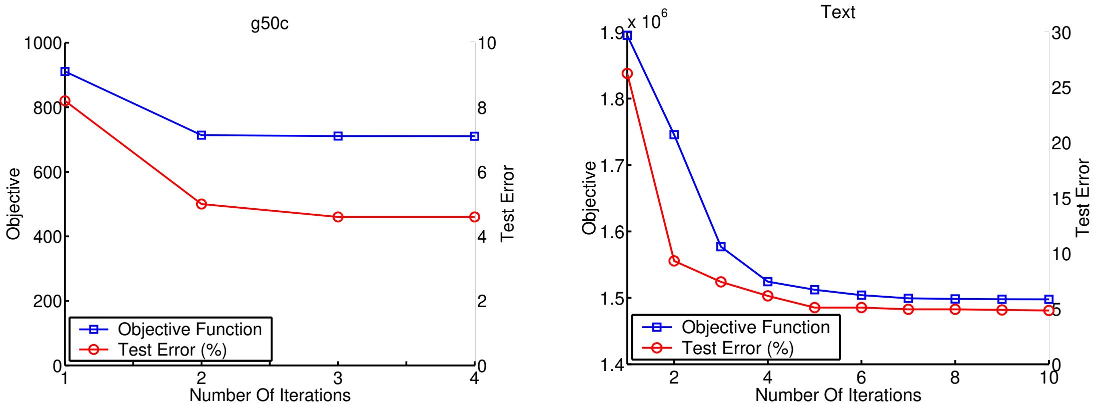

Figure 4: Value of the objective function and test error during the CCCP iterations of training TSVM on two datasets (single trial), `g50c` (left) and `text` (right). CCCP-TSVM tends to converge after only a few iterations.

optimization, CCCP-TSVMs run orders of magnitude faster than SVMLight-TSVMs and ∇TSVM[4]. Figure 3 shows the training time on `g50c` and `text` for the three methods as we vary the number of unlabeled examples. For each method we report the training times for the hyperparameters that give optimal performance as measured on the test set on the first split of the data (we use CCCP-TSVM$|_{UC^*=LC}^{s=0}$ in these experiments). Using all 2000 unlabeled data on Text, CCCP-TSVMs are approximately *133 times faster* than SVMLight-TSVM and *50 times faster* than ∇TSVM.

We expect these differences to increase as the number of unlabeled examples increases further. In particular, ∇TSVM requires the storage of the entire kernel matrix in memory,

4. ∇TSVM was implemented by adapting the Matlab LDS code of Chapelle and Zien (2005) available at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds`.

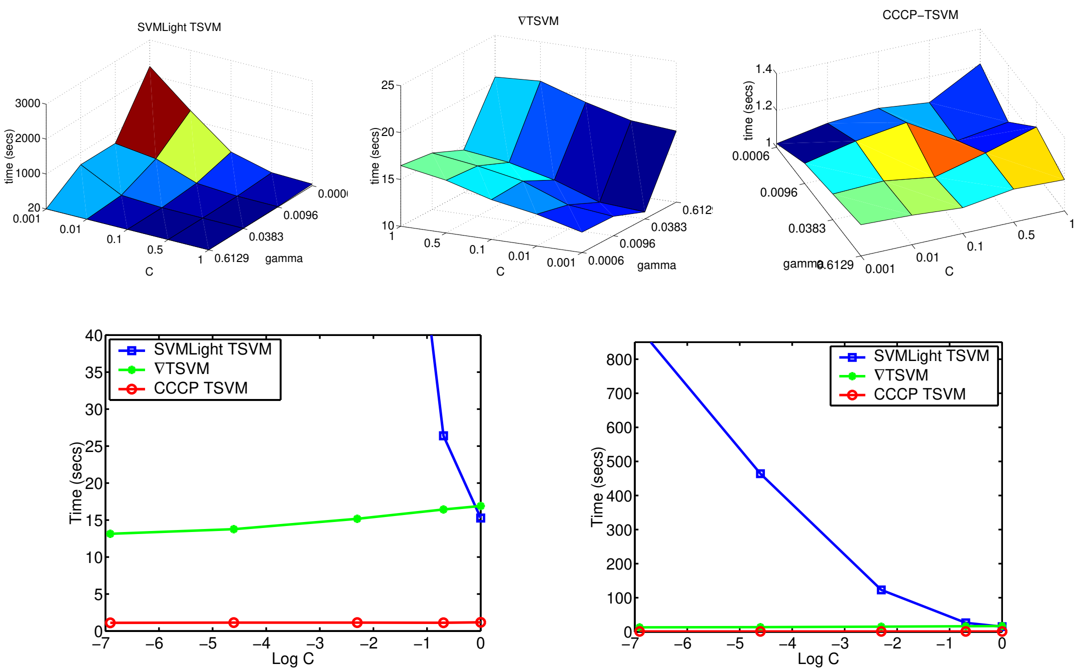

Figure 5: Computation times for different choices of hyperparameters on dataset `g50c` (first split only) for the three TSVM implementations tested (top three figures). The bottom two figures show the computation time for all three algorithms with respect to the parameter C only, where the time is the mean training time taken over the possible values of γ. The bottom right figure is a scale up of the bottom left figure, as SVMLight-TSVM is so slow it hardly appears on the left figure. In general, SMLight-TSVM computation time appears very sensitive to parameter choices, with small values of C and γ resulting in computation times around 2250 seconds, whereas large values of C and γ are much faster. ∇ TSVM has almost the opposite trend on this dataset: it is slower for large values of C or γ, although even the slowest time is still only around 20 seconds. Our CCCP-TSVM takes only around 1 second for all parameter choices.

and is therefore clearly infeasible for some of the large scale experiments we attempt in Section 5.

Finally, Figure 4 shows the value of the objective function and test error during the CCCP iterations of training TSVM on two datasets. The CCCP-TSVM objective function converges after five to ten iterations.

4.3 Hyperparameters

We now discuss in detail how the hyperparameters γ, C, C^* and s affect the performance of the TSVM algorithms.

Effect of the parameters γ and C. The parameters γ and C have similar effects on generalization as in the purely supervised SVM approach (see Keerthi and Lin. (2003) for an empirical study). However, during model selection, one has to try many choices of parameters. Some algorithms have different computational behaviour across different parameter choices. Therefore we have studied how different choices of C and γ affect the computation times of all three TSVM algorithms. Figure 5 compares these computation times for the `g50c` dataset. SVMLight-TSVM is particularly slow for small γ and C, taking up to 2250 seconds, whereas the other two algorithms are relatively more stable. In particular, CCCP-TSVM takes only around 1 second for every possible parameter choice. This means that during cross validation the CCCP-TSVM speedup over SVMLight-TSVM is even larger than the 133x speedup observed for the relatively benign choice of hyperparameters in Figure 3.

Effect of the parameter C^* As mentioned before, both SVMLight-TSVM and ∇TSVM use an annealing heuristic for hyperparameter C^*. They start their optimization using a small value of C^* and slowly increase C^* until it reaches the final desired value $C^* = C$. CCCP-TSVM solves the optimization problem for the desired value of C^* without an annealing heuristic. When one wishes to avoid optimizing C^*, we suggest the heuristic $UC^* = LC$.

Comparing the heuristics $C^* = C$ and $UC^* = LC$ Table 3 compares the $C^* = C$ and $UC^* = LC$ heuristics on the small scale datasets. Results are provided for the case $s = 0$ and the case where we allow the optimization of s. Although the heuristic $C^* = C$ gives reasonable results for small amounts of unlabeled data, we prefer the heuristic $UC^* = LC$. When the number of unlabeled examples U becomes large setting $C^* = C$ will mean the third term in the objective function (1) will dominate, resulting in almost no attention being paid to minimizing the training error. In these experiments the heuristic $UC^* = LC$ is close to the best possible choice of C^*, whereas $C^* = C$ is a little worse.

We also conducted an experiment to compare these two heuristics for larger unlabeled data sizes U. We took the same `uspst` dataset (that is, the test part of the USPS dataset) and we increased the number of unlabeled examples by adding up to 6000 additional unlabeled examples taken from the original USPS training set. Figure 6 reports the best test error for both heuristics over possible of choices of γ and C, taking the mean of the same 10 training splits with 50 labeled examples as before. The results indicte that $C^* = C$ works poorly for large U.

	Coil20	g50c	Text	Uspst	(number of hyperparameters)
CCCP-TSVM$\|^{s=0}_{C^*=C}$	22.33	4.68	7.76	20.09	2
CCCP-TSVM$\|^{s=0}_{UC^*=LC}$	16.69	5.62	7.97	16.57	2
CCCP-TSVM$\|^{s=0}$	16.67	4.56	7.76	16.55	3
CCCP-TSVM$\|_{C^*=C}$	19.02	4.28	5.22	18.33	3
CCCP-TSVM$\|_{UC^*=LC}$	16.06	5.04	5.59	16.47	3
CCCP-TSVM	15.92	3.92	4.92	16.45	4

Table 3: Comparison of $C^* = C$ and $C^* = \frac{L}{U}C$ heuristics on on Small-Scale Datasets with the best optimized value of C^* (CCCP-TSVM$|^{s=0}$ or CCCP-TSVM, depending on whether s is fixed). The heuristic $C^* = \frac{L}{U}C$ maintains the balance between unlabeled points U and labeled points L as U and L change, and is close to the best possible choice of C^*. The $C^* = C$ heuristic also works for relatively small values of U as in this case. We report all methods with and without the optimization of s.

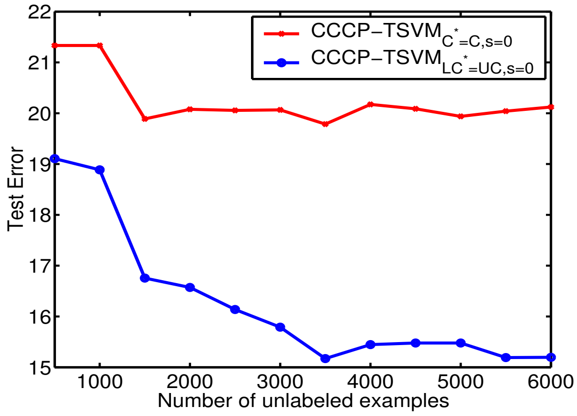

Figure 6: Comparison of the $C^* = C$ and $UC^* = LC$ heuristics on the `uspst` dataset as we increase the number of unlabeled examples, by adding extra unlabeled data from the original usps training set. We report the best test error over for both heuristics over possible of choices of γ and C, taking the mean of the same 10 training splits with 50 labeled examples as before. As the number of unlabeled examples increases the $C^* = C$ heuristic gives too much weight to the unlabeled data, resulting in no improvement in performance. Intuitively, the $UC^* = LC$ heuristic balances the weight of the unlabeled and labeled data and empirically performs better.

Iteratively increasing C^* — Iteratively increasing C^* during the optimization can be interpreted as starting from a convex problem ($C^* = 0$) and gradually making it more non-convex, which may be a good strategy to solve such non-convex problems. However, we believe that the annealing procedure has also a regularizing effect. The optimization is more likely to get stuck in a local minimum that appears when C^* has a value much smaller than C. This may be why the $C^* = C$ heuristic works well for algorithms that also use the annealing trick.

We conducted an experiment to see the performance of SVMLight-TSVM with and without the annealing heuristic. On `g50c`, we chose a linear kernel and compute the optimal value of C on the test set using $C^* = C$. With the annealing heuristic, we obtain a test error of 7.6%. For the same parameters without the annealing procedure, we obtain 12.4%. Clearly the annealing heuristic has a strong effect on the results of SVMLight-TSVM. CCCP-TSVM has no such heuristic.

Effect of the parameter s The parameter s in CCCP-TSVM controls the choice of loss function to minimize over. It controls the size of the plateau of the Symmetric Ramp function (Figure 1, right). Training our algorithm with a tuned value of s appears to give slightly improved results over using the Symmetric Hinge loss ($s = 0$, see Figure 1, left), especially on the `text` dataset, as can be seen in Tables 2 and 3. Furthermore, Figure 7 highlights the importance of the parameter s of the loss function (2) by showing the best test error over different choices of s for two datasets, `text` and `g50c`.

We conjecture that the peaked loss of the Symmetric Hinge function forces early decisions for the $\boldsymbol{\beta}$ variables and might lead to a poor local optimum. This effect then disappears as soon as we clip the loss. That is, the flat part of the loss far inside the margin prevents our

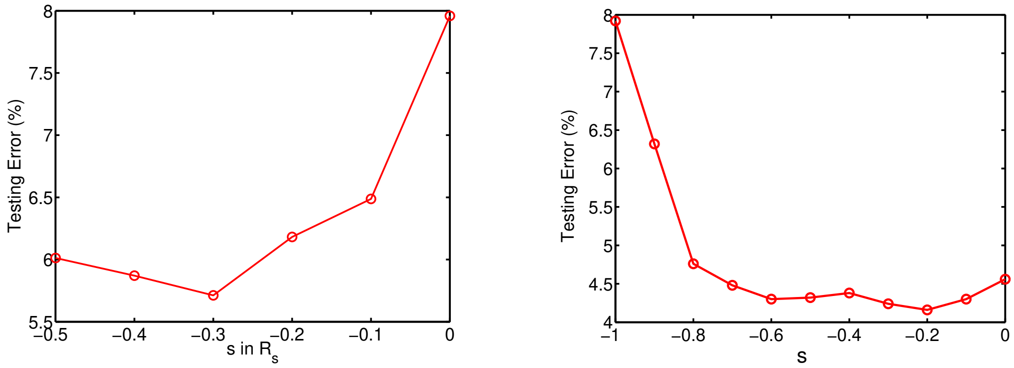

Figure 7: Effect of the parameter s of the Symmetric Ramp loss (see Figure 1 and equation (2)) on the `text` dataset (left) and the `g50c` dataset (right). The peaked loss of the Symmetric Hinge function ($s = 0$) forces early decisions for the $\boldsymbol{\beta}$ variables and might lead to a poor local optimum. This effect then disappears as soon as we clip the loss.

algorithm making erroneous early decisions regarding the labels of the unlabeled data that may be hard to undo later in the optimization.

In fact the $\nabla TSVM$ authors make a similar argument to explain why they prefer their algorithm over SVMLight: "*(SVMLight) TSVM might suffer from the combinatorial nature of its approach. By deciding, from the very first step, the putative label of every point (even with low confidence), it may lose important degrees of freedom at an early stage and get trapped in a bad local minimum*".

Here, the authors are refering to the way SVMLight TSVM has a discrete rather than continuous approach of assigning labels to unlabeled data. However, we think that the smoothed loss function of ∇TSVM may help it to outperform the Symmetric Hinge loss of SVMLight TSVM, making it similar to the clipped loss when we use $s < 0$. Indeed the ∇TSVM smoothed loss, $\exp(-3t^2)$, has small gradients when t close to 0.

A potential issue of the Symmetric Ramp loss is the fact that the gradient is exactly 0 for points lying on the plateau. Points are not updated at all in this region. This may be sub-optimal: if we are unlucky enough that all unlabeled points lie in this region, we perform no updates at all. Performing model selection on parameter s eliminates this problem. Alternatively, we could use a piece-wise linear loss with two different slopes for $|f(x)| > s$ and for $|f(x)| < s$. Although it is possible to optimize such a loss function using CCCP, but we have not evaluated this approach.

5. Large Scale Experiments

In this section, we provide experimental results on large scale experiments. Since other methods are intractable on such data sets, we only compare CCCP-TSVM against SVMs.

5.1 RCV1 Experiments

The first large scale experiment that we conducted was to separate the two largest top-level categories CCAT (CORPORATE/INDUSTRIAL) and GCAT (GOVERNMENT/SOCIAL) of the training part of the Reuters dataset as prepared by Lewis et al. (2004). The set of

Method	Train size	Unlabeled size	Parameters	Test Error
SVM	100	0	$C = 252.97,\ \sigma = 15.81$	16.61%
TSVM	100	500	$C = 2.597, C^* = 10,\ s = -0.2,\ \sigma = 3.95$	11.99%
TSVM	100	1000	$C = 2.597, C^* = 10,\ s = -0.2,\ \sigma = 3.95$	11.67%
TSVM	100	2000	$C = 2.597, C^* = 10,\ s = -0.2,\ \sigma = 3.95$	11.47%
TSVM	100	5000	$C = 2.597, C^* = 2.5297,\ s = -0.2,\ \sigma = 3.95$	10.65%
TSVM	100	10000	$C = 2.597, C^* = 2.5297,\ s = -0.4,\ \sigma = 3.95$	10.64%
SVM	1000	0	$C = 25.297,\ \sigma = 7.91$	11.04%
TSVM	1000	500	$C = 2.597, C^* = 10,\ s = -0.4,\ \sigma = 3.95$	11.09%
TSVM	1000	1000	$C = 2.597, C^* = 2.5297,\ s = -0.4,\ \sigma = 3.95$	11.06%
TSVM	1000	2000	$C = 2.597, C^* = 10,\ s - 0.4 =,\ \sigma = 3.95$	10.77%
TSVM	1000	5000	$C = 2.597, C^* = 2.5297,\ s = -0.2,\ \sigma = 3.95$	10.81%
TSVM	1000	10000	$C = 2.597, C^* = 25.2970,\ s = -0.4,\ \sigma = 3.95$	10.72%

Table 4: Comparing CCCP-TSVMs with SVMs on the RCV1 problem for different number of labeled and unlabeled examples. See text for details.

these two categories consists of 17754 documents. The features are constructed using the bag of words technique, weighted with a TF.IDF scheme and normalized to length one. We performed experiments using 100 and 1000 labeled examples. For model selection we use a validation set with 2000 and 4000 labeled examples for the two experiments. The remaining 12754 examples were used as a test set.

We chose the parameter C and the kernel parameter γ (using an RBF kernel) that gave the best performance on the validation set by training a TSVM using the validation set as the unlabeled data. These values were then fixed for every experiment.

We then varied the number of unlabeled examples U, and report the test error for each choice of U. In each case we performed model selection to find the parameters C^* and s. A selection of the results can be seen in Table 4.

The best result we obtained for $1k$ training points was 10.58% test error, using $10.5k$ unlabeled points, and for $0.1k$ training points was 10.42% using $9.5k$ unlabeled points. Compared to the best performance of a SVM of 11.04% for the former and 16.61% for the latter this shows that unlabeled data can improve the results on this problem, especially in the case of few training data, where the improvement in test error is around 5.5%. However, when enough training data is available to the algorithm, the improvement is only in the order of one percent.

Figure 8 shows the training time of CCCP optimization as a function of the number of unlabeled examples. On a 64 bit Opteron processor the optimization time for $12.5k$ unlabeled examples was approximately 18 min using the $1k$ training set and 69 min using $0.1k$ training points. Although the worst case complexity of SVMs is cubic and the optimization time seems to be dependent on the ratio of the number of labeled to unlabeled examples, the training times clearly show a quadratic trend.

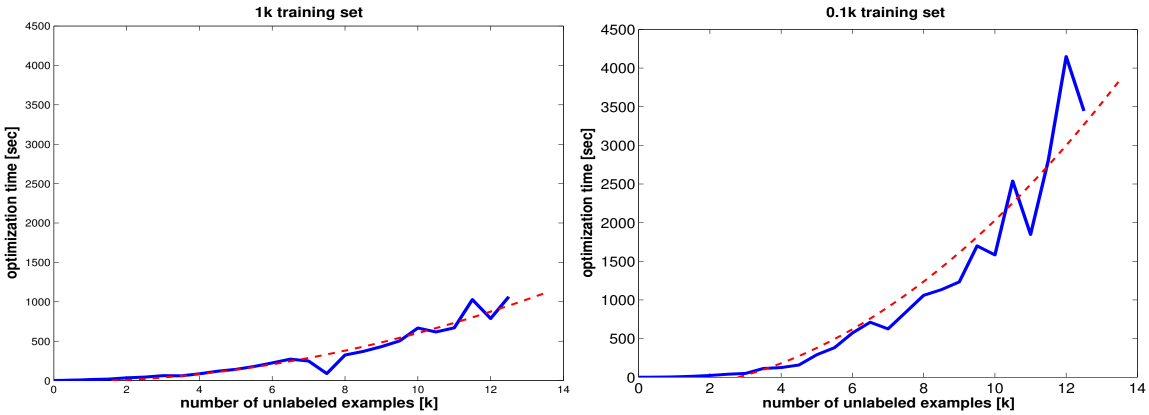

Figure 8: Optimization time for the Reuters dataset as a function of the number of unlabeled data. The algorithm was trained on 1,000 points (left) and on 100 points (right). The dashed lines represent a parabola fitted at the time measurements.

Method	Train size	Unlabeled size	Parameters	Test Error
SVM	100	0	$C = 10,\ \gamma = 0.0128$	23.44%
TSVM	100	2000	$C^* = 1,\ s = -0.1$	16.81%
SVM	1000	0	$C = 10,\ \gamma = 0.0128$	7.77%
TSVM	1000	2000	$C^* = 5,\ s = -0.1$	7.13%
TSVM	1000	5000	$C^* = 1,\ s = -0.1$	6.28%
TSVM	1000	10000	$C^* = 0.5,\ s = -0.1$	5.65%
TSVM	1000	20000	$C^* = 0.3,\ s = -0.1$	5.43%
TSVM	1000	40000	$C^* = 0.2,\ s = -0.1$	5.31%
TSVM	1000	60000	$C^* = 0.1,\ s = -0.1$	5.38%

Table 5: Comparing CCCP-TSVMs with SVMs on the MNIST problem for different number of labeled and unlabeled examples. See text for details.

5.2 MNIST Experiments

In the second large scale experiment, we conducted experiments on the MNIST handwritten digit database, as a 10-class problem. The original data has 60,000 training examples and 10,000 testing examples. We subsampled the training set for labeled points, and used the test set for unlabeled examples (or the test set + remainder of the training set when using more than 10,000 unlabeled examples). We performed experiments using 100 and 1000 labeled examples. We performed model selection for 1-vs-the-rest SVMs by trying a grid of values for σ and C and selecting the best ones using a separate validation set of size 1000. For TSVMs, for efficiency reasons we fixed σ and C to be the same ones as chosen for SVMs. We then performed model selection using 2000 unlabeled examples to find the best choices of C^* and s using the validation set. When using more unlabeled data, we only reperformed model selection on C^* as it appeared that this parameter was the most sensitive

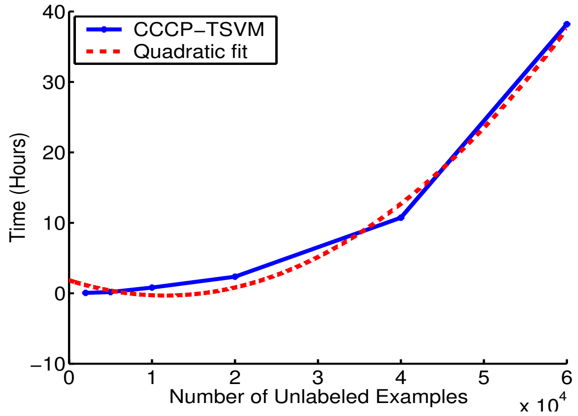

Figure 9: Optimization time for the MNIST dataset as a function of the number of unlabeled data. The algorithm was trained on 1,000 labeled examples and up to 60,000 unlabeled examples. The dashed lines represent a polynomial of degree two with a least square fit on the algorithm's time measurements.

to changes in the unlabeled set, and kept the other parameters fixed. For the larger labeled set we took 2000, 5000 and 10000, 20000, 40000 and 60000 unlabeled examples. We always measure the error rate on the complete test set. The test error and parameter choices for each experiment are given in the Table 5, and the training times are given in Figure 9.

The results show an improvement over SVM for CCCP-TSVMs which increases steadily as the number of unlabeled examples increases. Most experiments in semi-supervised learning only use a few labeled examples and do not use as many unlabeled examples as described here. It is thus reassuring to know that these methods do not apply just to toy examples with around 50 training points, but gains are still possible with more realistic dataset sizes.

6. Discussion and Conclusions

6.1 Cluster kernels and manifold-learning

Transductive SVM is not the only method of leveraging unlabeled data in a supervised learning task. In recent years this has become a popular research topic, and a battery of techniques have been proposed. Some of the main methods, which we refer to as cluster kernels, do not change the learning algorithm at all, but merely the representation of the data as a pre-processing step. In a purely unsupervised fashion, these methods learn cluster or manifold structure from the data, and produce a new representation of it such that distances between points in the new space are small if they are in the same cluster or on the same manifold. Some of the main methods include Chapelle et al. (2002); Chapelle and Zien (2005); Sindhwani et al. (2005); Szummer and Jaakkola (2001b) and Weston et al. (2003).

Other notable methods include generalizations of nearest-neighbor or Parzen window type approaches to learning manifolds given labeled data (Zhu et al., 2003; Belkin and Niyogi, 2002; Zhou et al., 2004). Finally, Bayesian approaches have also been pursued (Graepel et al., 2000; Lawrence and Jordan, 2005).

We note that some cluster kernel methods (Chapelle and Zien, 2005) can perform significantly better than TSVM on some datasets. In fact, Chapelle and Zien (2005) show that, as these methods provide a new representation, one can just as easily run TSVM on the new representation. The combination of TSVM and cluster kernels then provides state-of-the-art results.

6.2 Semi-supervised versus transductive learning

From a theoretical point of view, there is much idealogical debate over which underlying theory that explains TSVM is correct. The argument here is largely about which framework, semi-supervised learning or transductive, is interesting to study theoretically or to apply practically.

Semi-supervised school The majority of researchers appear to be in the *semi-supervised* school of thought, which claims that TSVMs help simply because of a regularizer that reflects prior knowledge, see e.g. (Chapelle and Zien, 2005). That is, one is given a set of unlabeled data, and one uses it to improve an inductive classifier to improve its generalization on an unseen test set.

Transductive school Transduction's aim, as argued by its inventor Vapnik, is to improve the generalization ability on a test set, given the test set as unlabeled data. Vapnik claims that this can mean solving a less complex problem than induction, where one must first learn a general inductive rule and then apply it using deduction. One could potentially label the test data without building an inductive rule. Vapnik (1995) has also obtained a generalization bound for TSVMs that is slightly better than the one for SVMs. There is also a PAC-Bayesian analysis (Derbeko et al., 2004).

Transduction is not possible in those cases where the test set becomes available after training only, so in that sense its applicability is more limited than for semi-supervised learning. On the other hand, when the test and training data are not identically distributed, we believe the concept of transduction could be particularly worthwhile.

6.3 Conclusion

In this article we have described an algorithm for TSVMs using CCCP that brings scalability improvements over existing implementation approaches. It involves the iterative solving of standard dual SVM QP problems, and usually requires just a few iterations. One nice thing about being an extension of standard SVM training is that any improvements in SVM scalability can immediately also be applied to TSVMs. For example in the linear case, one could easily apply fast linear SVM training such as in Keerthi and DeCoste (2005) to produce very fast linear TSVMs. For the nonlinear case, one could apply the online SVM training scheme of Bordes et al. (2005) to give a fast online transductive learning procedure.

Acknowledgments

We thank Hans Peter Graf, Eric Cosatto, and Vladimir Vapnik for their advice and support. Part of this work was funded by NSF grant CCR-0325463.

Appendix A. Derivation of the optimization problem

We consider a set of L training pairs $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_L, y_L)\}$, $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \{1, -1\}$ and a (unlabeled) set of U test vectors $\mathcal{U} = \{x_{L+1}, \dots, x_{L+U}\}$. SVMs have a decision function $f_\theta(.)$ of the form

$$f_\theta(x) = w \cdot \Phi(\boldsymbol{x}) + b\,,$$

where $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ are the parameters of the model, and $\Phi(\cdot)$ is the chosen feature map.

We are interested in minimizing the TSVM cost function (3), under the constraint (4). We rewrite the problem here for convenience: minimizing

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,, \tag{12}$$

under the constraint

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \;=\; \frac{1}{L}\sum_{i=1}^{L} y_i\,. \tag{13}$$

Assume that a cost function $J(\boldsymbol{\theta})$ can be rewritten as the sum of a convex part $J_{vex}(\boldsymbol{\theta})$ and a concave part $J_{cav}(\boldsymbol{\theta})$. As mentioned above in Algorithm 1, the minimization of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ ($\boldsymbol{\theta}$ being possibly restricted to a space $\mathcal{A}$ defined by some linear constraints) can be achieved by iteratively updating the parameters $\boldsymbol{\theta}$ using the following update

$$\boldsymbol{\theta}^{t+1} = \underset{\boldsymbol{\theta}\in\mathcal{A}}{\arg\min}\left(J_{vex}(\boldsymbol{\theta}) + J'_{cav}(\boldsymbol{\theta}^t)\cdot\boldsymbol{\theta}\right)\,. \tag{14}$$

In the case of our cost (12), we showed (see (9)) that $J^s(\boldsymbol{\theta})$ can be decomposed into the sum of $J^s_{vex}(\boldsymbol{\theta})$ and $J^s_{cav}(\boldsymbol{\theta})$ where

$$J^s_{vex}(\boldsymbol{\theta}) = \frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \tag{15}$$

and

$$J^s_{cav}(\boldsymbol{\theta}) = -C^* \sum_{i=L+1}^{L+2U} H_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{16}$$

In order to apply the CCCP update (14) we first have to calculate the derivative of the concave part (16) with respect to $\boldsymbol{\theta}$:

$$\frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -C^* \sum_{i=L+1}^{L+2U} \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}\,\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}$$

We introduce the notation

$$\begin{aligned}
\beta_i &= y_i\,\frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)} \\
&= \begin{cases} C^*\, H'_s[y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)] & \text{if } i \geq L+1 \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} C^* & \text{if } \; y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases} \quad .
\end{aligned}$$

Since $f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i) + b$ with $\theta = (w, b)$, and $\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) / \partial \boldsymbol{\theta} = (\Phi(\boldsymbol{x}_i), 1)$, each update (14) of the CCCP procedure applied to the our minimization problem (12) consists in minimizing the following cost

$$\begin{aligned} J^s_{vex}(\boldsymbol{\theta}) + \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot \boldsymbol{\theta} &= J^s_{vex}(\boldsymbol{\theta}) + \left(\sum_{i=L+1}^{L+2U} y_i \, \beta_i \, \frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}} \right) \cdot \boldsymbol{\theta} \\ &= J^s_{vex}(\boldsymbol{\theta}) + \sum_{i=L+1}^{L+2U} \beta_i \, y_i \left[\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i) + b \right] , \end{aligned} \tag{17}$$

under the linear constraint (13).

The convex part (16) contains Hinge Losses which can be rewritten as

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

It is thus easy to see that the minimization of (17) under the constraint (13) is equivalent to the following quadratic minimization problem under constraints:

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad & \frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{i=1}^{L} \xi_i + C^* \sum_{i=L+1}^{L+2U} \xi_i + \sum_{i=L+1}^{L+2U} \beta_i \, y_i \, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \\ \text{s.t.} \quad & \frac{1}{U} \sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \frac{1}{L} \sum_{i=1}^{L} y_i & (18) \\ & y_i f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 - \xi_i \quad \forall 1 \leq i \leq L + 2U & (19) \\ & \xi_i \geq 0 \quad \forall 1 \leq i \leq L + 2U & (20) \end{aligned}$$

Introducing lagrangian variables α_0, $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$ corresponding respectively to constraints (18), (19) and (20), we can write the Lagrangian of this problem as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta},\, \boldsymbol{\xi},\, \boldsymbol{\alpha}, \boldsymbol{\nu}) = & \frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{i=1}^{L} \xi_i + C^* \sum_{i=L+1}^{L+2U} \xi_i + \sum_{i=L+1}^{L+2U} \beta_i \, y_i \, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \\ & - \alpha_0 \left(\frac{1}{U} \sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \frac{1}{L} \sum_{i=1}^{L} y_i \right) \\ & - \sum_{i=1}^{L+2U} \alpha_i \left(y_i f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - 1 + \xi_i \right) \\ & - \sum_{i=1}^{L+2U} \nu_i \xi_i \,, \end{aligned} \tag{21}$$

where α_0 can be positive or negative (equality constraint) and $\alpha_i, \; i \geq 1$ are non-negative (inequality constraints).

Taking in account that $\beta_i = 0$ for $i \leq L$, calculating the derivatives with respect to the primal variables yields

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{L+2U} y_i\,(\alpha_i - \beta_i)\,\Phi(\boldsymbol{x}_i) - \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\Phi(\boldsymbol{x}_i)\\
\frac{\partial \mathcal{L}}{\partial b} &= -\sum_{i=1}^{L+2U} y_i\,(\alpha_i - \beta_i) - \alpha_0\\
\frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \alpha_i - \nu_i \quad \forall 1 \leq i \leq L\\
\frac{\partial \mathcal{L}}{\partial \xi_i} &= C^* - \alpha_i - \nu_i \quad \forall L+1 \leq i \leq L+2U\,.
\end{aligned}$$

For simplifying the notation, we now define an extra special example $\boldsymbol{x}_0$ in an implicit manner:

$$\Phi(\boldsymbol{x}_0) = \frac{1}{U}\sum_{i=L+1}^{L+U}\Phi(\boldsymbol{x}_i)\,,$$

and we set $y_0 = 1$ and $\beta_0 = 0$. Setting the derivatives to zero gives us

$$\boldsymbol{w} = \sum_{i=0}^{L+2U} y_i\,(\alpha_i - \beta_i)\,\Phi(\boldsymbol{x}_i) \tag{22}$$

and

$$\sum_{i=0}^{L+2U} y_i\,(\alpha_i - \beta_i) = 0 \tag{23}$$

and also

$$C - \alpha_i - \nu_i = 0 \quad \forall 1 \leq i \leq L, \quad C^* - \alpha_i - \nu_i \quad \forall L+1 \leq i \leq L+2U\,. \tag{24}$$

In order to find the minimum of the minimization problem (12) we want to find a saddle point of the Lagrangian (21), as classical SVMs methods. Substituting (22), (23) and (24) into the Lagrangian (21) yields the following maximization problem

$$\begin{aligned}
\arg\max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i,j=0}^{L+2U} y_i\,y_j(\alpha_i - \beta_i)\,(\alpha_j - \beta_j)\Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)\\
& + \sum_{i=1}^{L+2U}\alpha_i + \alpha_0\left(\frac{1}{L}\sum_{i=1}^{L} y_i\right)
\end{aligned} \tag{25}$$

under the constraints

$$\begin{gathered}
0 \leq \alpha_i \leq C \quad \forall 1 \leq i \leq L\\
0 \leq \alpha_i \leq C^* \quad \forall L+1 \leq i \leq L+2U\\
\textstyle\sum_{i=0}^{L+2U} y_i\,(\alpha_i - \beta_i) = 0\,.
\end{gathered} \tag{26}$$

The parameter $\boldsymbol{w}$ is then given by (22) and b is obtained using one of the following Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned}
\alpha_0 \neq 0 &\implies \frac{1}{U}\sum_{i=L+1}^{L+U}[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i)+b] = \frac{1}{L}\sum_{i=1}^{L} y_i \\
\forall 1 \leq i \leq L,\ 0 < \alpha_i < C &\implies y_i[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i)+b] = 1 \\
\forall L+1 \leq i \leq L+2U,\ 0 < \alpha_i < C^* &\implies y_i[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i)+b] = 1
\end{aligned}$$

If we define $\zeta_i = y_i$ for $1 \leq i \leq L+2U$ and $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$, if we consider the kernel matrix K such that

$$K_{ij} = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)\,,$$

and if we perform the substitution

$$\tilde{\alpha}_i = y_i\,(\alpha_i - \beta_i)\,,$$

then we can rewrite the maximization problem (25) under the constraints (26) as the following

$$\arg\max_{\tilde{\boldsymbol{\alpha}}} \quad \boldsymbol{\zeta}\cdot\tilde{\boldsymbol{\alpha}} - \frac{1}{2}\,\tilde{\boldsymbol{\alpha}}^{\mathrm{T}} K \tilde{\boldsymbol{\alpha}}$$

under the constraints

$$\begin{aligned}
0 \leq y_i\,\tilde{\alpha}_i \leq C & \qquad \forall 1 \leq i \leq L \\
-\beta_i \leq y_i\,\tilde{\alpha}_i \leq C^* - \beta_i & \qquad \forall L+1 \leq i \leq L+2U \\
\textstyle\sum_{i=0}^{L+2U} \tilde{\alpha}_i = 0\,. &
\end{aligned} \tag{27}$$

Obviously this optimization problem is very close to a SVM optimization problem. It is thus possible to optimize it with a standard optimizer for SVMs. Note that only the bounds in (27) on the $\tilde{\alpha}_i$ have to be adjusted after each update of $\boldsymbol{\beta}$.

References

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems*. MIT Press, 2002.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 12*, pages 368–374. MIT Press, Cambridge, MA, 1998.

T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, May 2005. http://jmlr.csail.mit.edu/papers/v6/bordes05a.html.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. *Neural Information Processing Systems 15*, 2002.

P. Derbeko, R. El-Yaniv, and R. Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *Journal of Artificial Intelligence Research*, 22:117–142, 2004.

G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. In *Optimisation Methods and Software*, pages 1–14. Kluwer Academic Publishers, Boston, 2001.

T. Graepel, R. Herbrich, and K. Obermayer. Bayesian transduction. In *Advances in Neural Information Processing Systems 12, NIPS*, pages 456–462, 2000.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. The MIT Press, 1999a.

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning, ICML*, 1999b.

S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, 2005.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Comp.*, 15:2667–1689, 2003.

N. Krause and Y. Singer. Leveraging the margin more carefully. In *International Conference on Machine Learning, ICML*, 2004.

N. D. Lawrence and M. I. Jordan. Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems, NIPS*. MIT Press, 2005.

H. A. Le Thi. *Analyse numérique des algorithmes de l'optimisation D.C. Approches locales et globale. Codes et simulations numériques en grande dimension. Applications.* PhD thesis, INSA, Rouen, 1994.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G.B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998.

D. D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004. URL `http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf`.

L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. The MIT Press, 1999.

S.K.Nayar S.A.Nene and H.Murase. Columbia object image libary (coil-20). Technical Report CUS-005-96, Columbia Univ. USA, Febuary 1996.

B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Proceedings ICANN97*, Springer Lecture Notes in Computer Science, page 583, 1997.

X. Shen, G. C. Tseng, X. Zhang, and W. H. Wong. On (psi)-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *International Conference on Machine Learning, ICML*, 2005.

A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *NIPS*, 14, 2001a.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. *Neural Information Processing Systems 14*, 2001b.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 1995.

J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble. Cluster kernels for semi-supervised protein classification. *Advances in Neural Information Processing Systems 17*, 2003.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1537–1544. MIT Press, Cambridge, MA, 2005.

A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems, NIPS*, 2004.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *The Twentieth International Conference on Machine Learning*, pages 912–919, 2003.