1 Trading Convexity for Scalability

Ronan Collobert
Fabian Sinz
Jason Weston
Léon Bottou

Convex learning algorithms, such as Support Vector Machines (SVMs), are often seen as highly desirable because they offer strong practical properties and are amenable to theoretical analysis. However, in this work we show how nonconvexity can provide scalability advantages over convexity. We show how concave-convex programming can be applied to produce (i) faster SVMs where training errors are no longer support vectors, and (ii) much faster Transductive SVMs.

1.1 Introduction

The machine learning renaissance in the 80s was fostered by multilayer models. They were non-convex and surprisingly efficient. Convexity rose in the 90s with the growing importance of mathematical analysis, and the successes of convex models such as SVMs (Vapnik, 1995). These methods are popular because they span two worlds: the world of applications (they have good empirical performance, ease-of-use and computational efficiency) and the world of theory (they can be analysed and bounds can be produced). Convexity is largely responsible for these factors.

Many researchers have suspected that convex models do not cover all the ground previously addressed with non-convex models. In the case of pattern recognition, one argument was that the misclassification rate is poorly approximated by convex losses such as the SVM Hinge Loss or the Boosting exponential loss (Freund and Schapire, 1996). Various authors proposed non-convex alternatives (Mason et al., 2000; Perez-Cruz et al., 2002; Elisseeff, 2000), sometimes using the same concave-convex programming methods as this chapter (Krause and Singer, 2004; Liu et al., 2005).

The ψ-learning paper (Shen et al., 2003) stands out because it proposes a theoretical result indicating that non-convex loss functions yield fast convergence rates to the Bayes limit. This result depends on a specific assumption about the probability distribution of the examples. Such an assumption is necessary because there are no probability independent bounds on the rate of convergence to Bayes (Devroye et al., 1996). However, under substantially similar assumptions, it has recently been shown that SVMs achieve comparable convergence rates using the convex Hinge Loss (Steinwart and Scovel, 2005). In short, the theoretical accuracy advantage of non-convex losses no longer looks certain.

On real-life datasets, these previous works only report modest accuracy improvements comparable to those reported here. None mention the potential computational advantages of non-convex optimization, simply because everyone assumes that convex optimization is easier. On the contrary, most authors warn the reader about the potentially high cost of non-convex optimization.

This chapter proposes two examples where the *optimization of a non-convex loss functions brings considerable computational benefits* over the convex alternative[1].

Both examples leverage a modern concave-convex programming method (Le Thi, 1994). Section 1.2 shows how the ConCave Convex Procedure (CCCP) (Yuille and Rangarajan, 2002) solves a sequence of convex problems and has no difficult parameters to tune. Section 1.3 proposes an SVM where training errors are no longer support vectors. The increased sparsity leads to better scaling properties for SVMs. Section 1.5 describes what we believe is the best known method for implementing Transductive SVMs with a quadratic empirical complexity. This is in stark constrast to convex versions whose complexity grows with degree four or more.

1.2 The Concave-Convex Procedure

Minimizing a non-convex cost function is usually difficult. Gradient descent techniques, such as conjugate gradient descent or stochastic gradient descent, often involve delicate hyper-parameters (Le Cun et al., 1998). In contrast, convex optimization seems much more straight-forward. For instance, the SMO (Platt, 1999) algorithm locates the SVM solution efficiently and reliably.

We propose instead to optimize non-convex problems 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 for implementing ψ-learning (Shen et al., 2003; Liu et al., 2005).

1. Conversely, Bengio et al. (2006) proposes a convex formulation of multilayer networks which has considerably higher computational costs.

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 1.1) approximates the concave part by its tangent and minimizes the resulting convex function.

Algorithm 1.1 : The Concave-Convex Procedure (CCCP)

```
Initialize $\boldsymbol{\theta}^0$ with a best guess.
**repeat**
    $\boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \left( J_{vex}(\boldsymbol{\theta}) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta} \right)$    (1.1)
**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 (1.1) and from the concavity of $J_{cav}(\boldsymbol{\theta})$.

$$J_{vex}(\boldsymbol{\theta}^{t+1}) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta^{t+1}} \leq J_{vex}(\boldsymbol{\theta}^t) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta^t} \tag{1.2}$$

$$J_{cav}(\boldsymbol{\theta}^{t+1}) \leq J_{cav}(\boldsymbol{\theta}^t) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \left(\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t\right) \tag{1.3}$$

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 (1.1) is a convex minimization problem and can be solved using classical and efficient convex algorithms.

1.3 Non-Convex SVMs

This section describes the "curse of dual variables", that the number of support vectors increases in classical SVMs linearly with the number of training examples. The curse can be exorcised by replacing the classical Hinge Loss by a non-convex loss function, the Ramp Loss. The optimization of the new dual problem can be solved using CCCP.

1.3.1 Notation

In this section we consider two-class classification problems. We are given a training set $(\boldsymbol{x}_i, y_i)_{i=1\ldots L}$ with $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$. SVMs have a decision function $f_\theta(.)$ of the form $f_\theta(x) = w \cdot \Phi(x) + b$, where $\boldsymbol{\theta} = (\boldsymbol{w},\, b)$ are the parameters of the model, and $\Phi(\cdot)$ is the chosen feature map, often implicitly defined by a Mercer kernel (Vapnik, 1995). We also write the Hinge Loss $H_s(z) = \max(0,\, s - z)$ (Figure 1.1, center) where the subscript s indicates the position of the Hinge point.

1.3.2 SVM Formulation

The standard SVM criterion relies on the convex Hinge Loss to penalize examples classified with an insufficient margin:

$$\boldsymbol{\theta} \quad \mapsto \quad \frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{1.4}$$

The solution $\boldsymbol{w}$ is a sparse linear combination of the training examples $\Phi(\boldsymbol{x}_i)$, called support vectors (SVs). Recent results (Steinwart, 2003) show that the number of SVs k scales linearly with the number of examples. More specifically

$$k/L \to 2\,\mathcal{B}_\Phi \tag{1.5}$$

where $\mathcal{B}_\Phi$ is the best possible error achievable linearly in the chosen feature space $\Phi(\cdot)$. Since the SVM training and recognition times grow quickly with the number of SVs, it appears obvious that SVMs cannot deal with very large datasets. In the following we show how changing the cost function in SVMs cures the problem and leads to a non-convex problem solvable by CCCP.

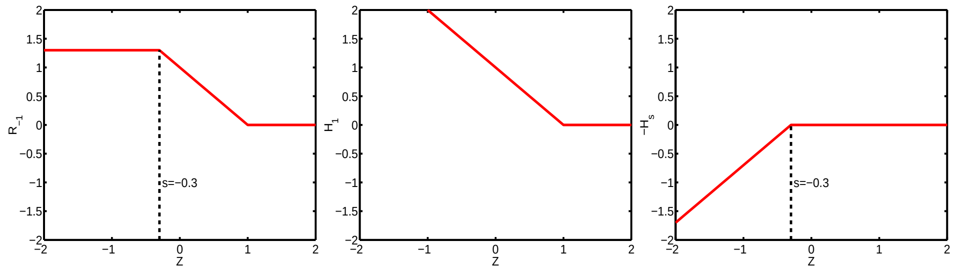

Figure 1.1 The Ramp Loss function $R_s(t) = \min(1-s, \max(0, 1-t)) = H_1(t) - H_s(t)$ (left) can be decomposed into the sum of the convex Hinge Loss (center) and a concave loss (right), where $H_s(t) = \max(0, s-t)$. The parameter s controls the cutoff point of the usual Hinge loss.

1.3.3 Losses for SVMs

The SV scaling property (1.5) is not surprising because all misclassified training examples become SVs. This is in fact a property of the Hinge Loss function. Assume for simplicity that the Hinge Loss is made differentiable with a smooth approximation on a small interval $z \in [1-\varepsilon, 1+\varepsilon]$ near the hinge point. Differentiating (1.4) shows that the minimum $\boldsymbol{w}$ must satisfy

$$\boldsymbol{w} = -C\sum_{i=1}^{L} y_i\, H_1'(y_i)\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\ \Phi(\boldsymbol{x}_i)\,. \tag{1.6}$$

Examples located in the flat area ($z > 1 + \varepsilon$) cannot become SVs because $H_1'(z)$ is 0. Similarly, all examples located in the SVM margin ($z < 1 - \varepsilon$) or simply misclassified ($z < 0$) become SVs because the derivative $H_1'(z)$ is 1.

We propose to avoid converting some of these examples into SVs by making the loss function flat for scores z smaller than a predefined value $s < 1$. We thus introduce the Ramp Loss (Figure 1.1):

$$R_s(z) = H_1(z) - H_s(z) \tag{1.7}$$

Replacing H_1 by R_s in (1.6) guarantees that examples with score $z < s$ do not become SVs.

Rigorous proofs can be written without assuming that the loss function is differentiable. Similar to SVMs, we write (1.4) as a constrained minimization problem with slack variables and consider the Karush-Kuhn-Tucker conditions. Although the resulting problem is non-convex, these conditions remain necessary (but not sufficient) optimality conditions (Ciarlet, 1990). We omit this easy but tedious derivation.

This sparsity argument provides a *new motivation for using a non-convex loss function.* Unlike previous works (see Introduction), our setup is designed to test and exploit this new motivation.

1.3.4 CCCP for Non-Convex SVMs

Decomposing (1.7) makes Ramp Loss SVMs amenable to CCCP optimization. The new cost $J^s(\boldsymbol{\theta})$ then reads:

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \tag{1.8}$$

$$= \underbrace{\frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{vex}(\boldsymbol{\theta})} \underbrace{- C\sum_{i=1}^{L} H_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{cav}(\boldsymbol{\theta})}$$

For simplification purposes, we introduce the notation

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

The convex optimization problem (1.1) that constitutes the core of the CCCP algorithm is easily reformulated into dual variables using the standard SVM technique. This yields the following algorithm[2].

2. Note that $R_s(z)$ is non-differentiable at $z = s$. It can be shown that the CCCP remains valid when using any super-derivative of the concave function. Alternatively, function

Algorithm 1.2 : CCCP for Ramp Loss SVMs

```
Initialize $\boldsymbol{\theta}^0 = (\boldsymbol{w}^0, b^0)$ and $\boldsymbol{\beta}^0$ as described in the text.
repeat
    • Solve the following convex problem ( with $K_{ij} = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$ ) and get $\boldsymbol{\alpha}^{t+1}$
        $$\max_{\boldsymbol{\alpha}} \left( \boldsymbol{\alpha} \cdot \boldsymbol{y} - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \right) \text{ subject to } \begin{cases} \boldsymbol{\alpha} \cdot \mathbf{1} = 0 \\ -\beta_i^t \le y_i \alpha_i \le C - \beta_i^t \quad \forall 1 \le i \le L \end{cases} \qquad (1.9)$$
    • Compute $b^{t+1}$ using $f_{\theta^{t+1}}(x_i) = \sum_{j=0}^{L} \alpha_j^{t+1} K_{ij} + b^{t+1}$ and
        $$\forall i \le L \;:\quad 0 < y_i \alpha_i < C \implies y_i f_{\theta^{t+1}}(x_i) = 1$$
    • Compute $\beta_i^{t+1} = \begin{cases} C & \text{if } y_i f_{\boldsymbol{\theta}^{t+1}}(\boldsymbol{x}_i) < s \\ 0 & \text{otherwise} \end{cases}$
until $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t$
```

Convergence in finite number of iterations 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 (1.3) is strict unless $\boldsymbol{\beta}$ remains unchanged.

1.3.5 Complexity

Although SVM training has a worst case complexity of $O(L^3)$ it typically scales quadratically (see Joachims, 1999a; Platt, 1999). Setting the initial $\boldsymbol{\beta}^0$ to zero in Algorithm 1.2 makes the first convex optimization identical to the Hinge Loss SVM optimization. Although useless support vectors are eliminated during the following iterations, the whole optimization process remains quadratic, assuming a constant number of iterations (see Figure 1.3).

Interestingly, we can easily give a *lower* bound on the SVM training complexity: simply verifying that a vector $\boldsymbol{\alpha}$ is a solution of the SVM quadratic problem involves computing the gradients of (1.9) with respect to $\boldsymbol{\alpha}$ and checking the Karush-Kuhn-Tucker optimality conditions (Vapnik, 1995). With L examples and S support vectors, this requires a number of operations proportional to $L\,S$. With convex SVMs the number of support vectors S tends to increase linearly with L, as shown in theory in Section 1.3.2 and in practice in Section 1.4.1, Figure 1.2: it is thus not surprising that convex SVMs are *at least* quadratic with respect to the number of training examples L.

We can reduce the number of support vectors S even in the first iteration of Algorithm 1.2 by simply choosing a better initial $\boldsymbol{\beta}^0$. We propose to initialize

$R_s(z)$ could be made smooth in a small interval $[s - \varepsilon, s + \varepsilon]$ as in our previous argument.

$\boldsymbol{\beta}^0$ according to the output $f_{\boldsymbol{\theta}^0}(\boldsymbol{x})$ of a Hinge Loss SVM trained on a subset of examples.

$$\beta_i^0 \quad = \quad \begin{cases} C & \text{if } \; y_i\, f_{\boldsymbol{\theta}^0}(\boldsymbol{x}_i) < s \\ 0 & \text{otherwise} \end{cases}$$

The successive convex optimizations are much faster because their solutions have roughly the same number of SVs as the final solution (see Figure 1.2). We show in the experimental Section 1.4.2 that this procedure is robust in practice, and its overall training time can be significantly smaller than the standard SVM training time.

1.3.6 Previous Work

The excessive number of SVs in SVMs has long been recognized as one of the main flaws of this otherwise elegant algorithm. Many methods to reduce the number of SVs are after-training methods that only improve efficiency during the test phase (see §18, Schölkopf and Smola, 2002).

Both Elisseeff (2000) and Perez-Cruz et al. (2002) proposed a sigmoid loss for SVMs. Their motivation was to approximate the $0-1$ Loss and they were not concerned with speed. Similarly, motivated by the theoretical promises of ψ-learning, Liu et al. (2005) proposes a special case of Algorithm 1.2 with $s = 0$ as the Ramp Loss parameter, and $\boldsymbol{\beta}^0 = 0$ as the algorithm initialization. In the experiment section, we show that our algorithm provides *sparse solutions*, thanks to the s parameter, and *accelerated training times*, thanks to a better choice of the $\boldsymbol{\beta}^0$ initialization.

More recently Xu et al. (2006) proposed a variant of SVMs where the Hinge Loss is also replaced by a non-convex loss. Unfortunately the resulting cost function is minimized after "convexification" of the problem yielding to an extremely time-consuming semi-definite minimization problem.

1.4 Experiments with Non-Convex SVMs

The experiments in this section use a modified version of SVMTorch,[3] coded in C. Unless otherwise mentioned, the hyper-parameters of all the models were chosen using a cross-validation technique, for best generalization performance. All results were averaged on 10 train-test splits.

3. Experiments can be reproduced using the source code available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.

Table 1.1 Comparison of SVMs using the Hinge Loss (H_1) and the Ramp Loss (R_s). Test error rates (Error) and number of SVs. All hyper-parameters including s were chosen using a validation set.

Dataset	Train	Test	Notes
Waveform[1]	4000	1000	Artificial data, 21 dims.
Banana[1]	4000	1300	Artificial data, 2 dims.
USPS+N[2]	7329	2000	0 vs rest + 10% label noise.
Adult[2]	32562	16282	As in (Platt, 1999).

[1] `http://mlg.anu.edu.au/~raetsch/data/index.html`
[2] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases`

	SVM H_1		SVM R_s	
Dataset	**Error**	**SV**	**Error**	**SV**
Waveform	8.8%	983	8.8%	**865**
Banana	9.5%	1029	9.5%	**891**
USPS+N	0.5%	3317	0.5%	**601**
Adult	15.1%	11347	**15.0%**	**4588**

1.4.1 Accuracy and sparsity

We first study the accuracy and the sparsity of the solution obtained by the algorithm. For that purpose, all the experiments in this section were performed by initializing Algorithm 1.2 using $\boldsymbol{\beta}^0 = 0$ which corresponds to initializing CCCP with the classical SVM solution. Table 1.1 presents an experimental comparison of the Hinge Loss and the Ramp Loss using the RBF kernel $\Phi(\boldsymbol{x}_l) \cdot \Phi(\boldsymbol{x}_m) = \exp(-\gamma \|\boldsymbol{x}_l - \boldsymbol{x}_m\|^2)$. The results of Table 1.1 clearly show that the Ramp Loss achieves similar generalization performance with much fewer SVs. This increased sparsity observed in practice follows our mathematical expectations, as exposed in the Ramp Loss section.

Figure 1.2 (left) shows how the s parameter of the Ramp Loss R_s controls the sparsity of the solution. We report the evolution of the number of SVs as a function of the number of training examples. Whereas the number of SVs in classical SVMs increases linearly, the number of SVs in Ramp Loss SVMs strongly depends on s. If $s \to -\infty$ then $R_s \to H_1$; in other words, if s takes large negative values, the Ramp Loss will not help to remove outliers from the SVM expansion, and the increase in SVs will be linear with respect to the number of training examples, as for classical SVMs. As reported on the left graph, for $s = -1$ on Adult the increase is already almost linear. As $s \to 0$, the Ramp Loss will prevent misclassified examples from becoming SVs. For $s = 0$ the number of SVs appears to increase like the square root of the number of examples, both for Adult and USPS+N.

Figure 1.2 (right) shows the impact of the s parameter on the generalization performance. Note that selecting $0 < s < 1$ is possible, which allows the Ramp Loss to remove well classified examples from the set of SVs. Doing so degrades the

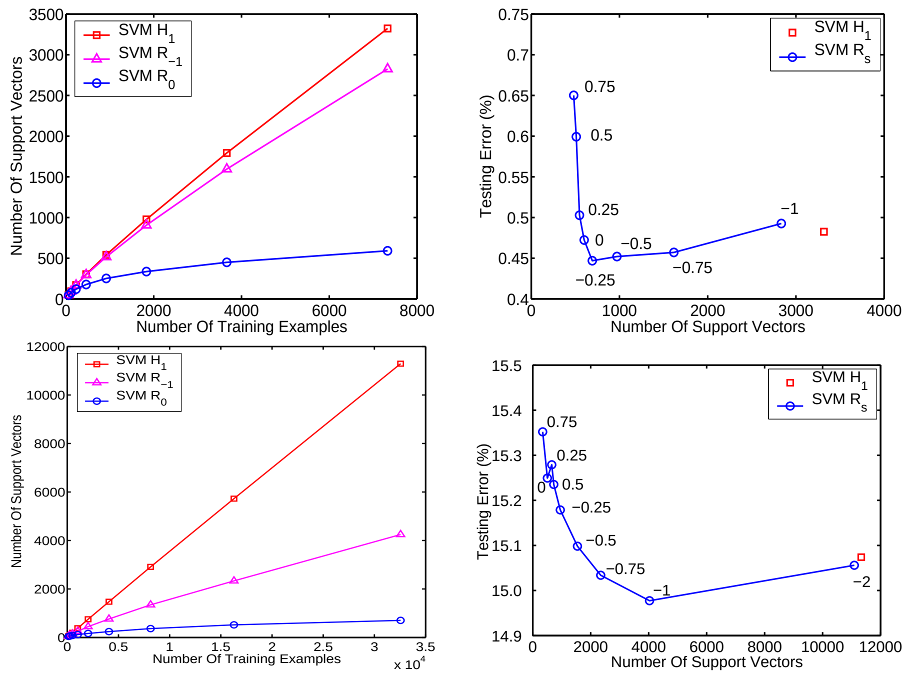

Figure 1.2 Number of SVs vs. training size (left) and training error (right) for USPS+N (top) and Adult (bottom). We compare the Hinge Loss H_1 and Ramp Losses R_{-1} and R_0 (left) and R_s for different values of s, printed along the curve (right).

generalization performance on both datasets.

Clearly s should be considered as a hyperparameter and selected for speed and accuracy considerations.

1.4.2 Speedup

The experiments we have detailed so far are not faster to train than a normal SVM because the first iteration ($\boldsymbol{\beta}^0 = 0$) corresponds to initializing CCCP with the classical SVM solution. Interestingly, few additional iterations are necessary (Figure 1.3), and they run much faster because of the smaller number of SVs. The resulting training time was less than twice the SVM training time.

We thus propose to initialize the CCCP procedure with a subset of the training set (let's say $1/P^{th}$), as described in Section 1.3.5. The first convex optimization is then going to be at least P^2 times faster than when initializing with $\boldsymbol{\beta}^0 = 0$ (SVMs training time being known to scale at least quadratically with the number of examples). We then train the remaining iterations with all the data, however, compared to standard SVMs fewer of these examples become SVs. Since the subsequent iterations involve a smaller number of SVs, we expect accelerated

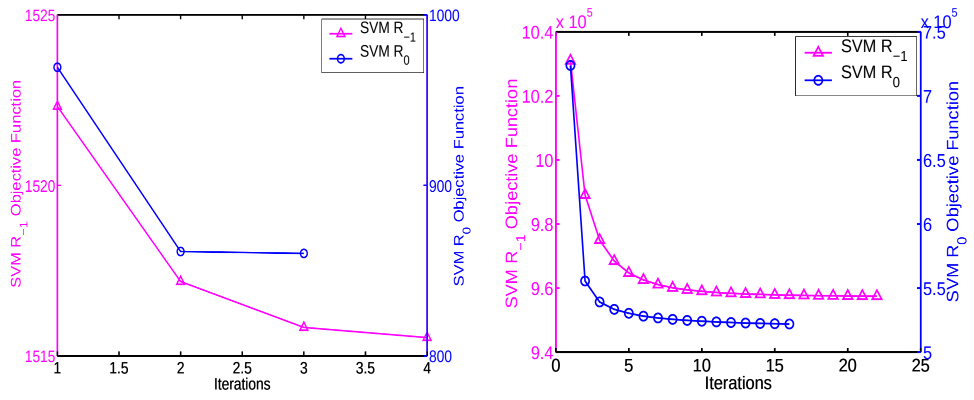

Figure 1.3 The objective function with respect to the number of iterations in the outer-loop of the CCCP for USPS+N (left) and Adult (right).

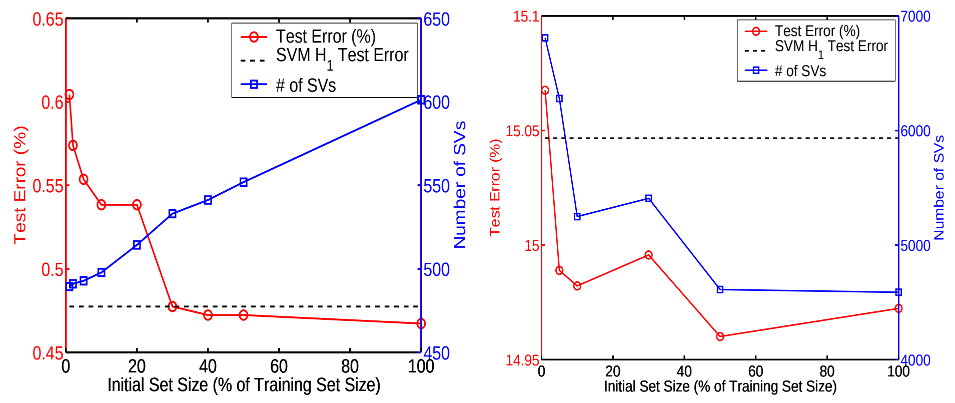

Figure 1.4 Results on USPS+N (left) and Adult (right) using an SVM with the Ramp Loss R_s. We show the test error and the number of SVs as a function of the percentage r of training examples used to initialize CCCP, and compare to standard SVMs trained with the Hinge Loss. For each r, all hyper-parameters were chosen using cross-validation.

training times.

Figure 1.4 shows the robustness of the CCCP procedure when initialized with a subset of the training set. On USPS+N and Adult, using respectively only $1/3^{th}$ and $1/10^{th}$ of the training examples is sufficient to match the generalization performance of classical SVMs.

Using this scheme, and tuning the CCCP procedure for speed and similar accuracy to SVMs, we obtain more than a two-fold and four-fold speedup over SVMs on USPS+N and Adult respectively, together with a large increase in sparsity (see Figure 1.5).

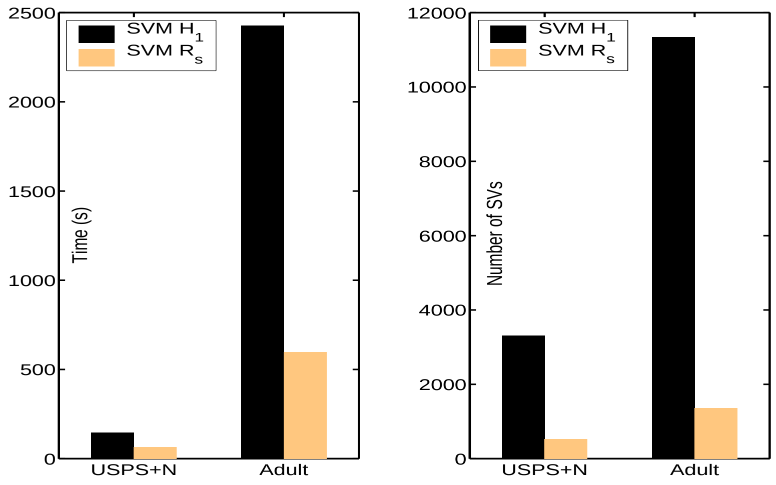

Figure 1.5 Results on USPS+N and Adult comparing Hinge Loss and Ramp Loss SVMs. The hyper-parameters for the Ramp Loss SVMs were chosen such that their test error would be at least as good as the Hinge Loss SVM ones.

1.5 Non-Convex Transductive SVMs

The same non-convex optimization techniques can be used to perform large scale semi-supervised learning. Transductive SVMs (Vapnik, 1995) seek large margins for both the labeled and unlabeled examples, in the hope that the true decision boundary lies in a region of low density, implementing the so-called cluster assumption (see Chapelle and Zien, 2005). When there are relatively few labeled examples and relatively abundant unlabeled examples, TSVMs can leverage unlabeled examples and give considerably better generalization performance than standard SVMs. Unfortunately, the TSVM implementations are rarely able to handle a large number of unlabeled examples.

Early TSVM implementations perform a *combinatorial* search of the best labels for the unlabeled examples. Bennett and Demiriz (1998) use an integer programming method, intractable for large problems. The SVM^{light}-TSVM (Joachims, 1999b) prunes the search tree using a non-convex objective function. This is practical for a few thousand unlabeled examples.

More recent proposals (De Bie and Cristianini, 2004; Xu et al., 2005) transform the transductive problem into a larger *convex semi-definite programming problem.* The complexity of these algorithms grows like $(L+U)^4$ or worse, where L and U are numbers of labeled and unlabeled examples. This is only practical for a few hundred examples.

We advocate instead the *direct optimization of the non-convex objective function.* This direct approach has been used before. The sequential optimization procedure of Fung and Mangasarian (2001) is the most similar to our proposal. This method potentially could scale well, although they only use 1000 examples in their largest experiment. However, it is restricted to the linear case, does not implement a balancing constraint (see below), and uses a special kind of SVM with a 1-norm regularizer to maintain linearity. The primal approach of Chapelle and Zien (2005) shows improved generalization performance, but still scales as $(L+U)^3$ and requires storing the entire $(L+U) \times (L+U)$ kernel matrix in memory.

1.5.1 Notation

We consider a set of L training pairs $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x} \in \mathbb{R}^n$, $y \in \{1, -1\}$ and an (unlabeled) set of U test vectors $\mathcal{U} = \{x_{L+1}, \ldots, x_{L+U}\}$.

1.5.2 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 an SVM separating the training set under constraints which force the unlabeled examples to be as far as possible from the margin. This can be written as minimizing

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

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

$$J(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+U} H_1(|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|), \tag{1.10}$$

where the function $H_1(\cdot) = \max(0, 1 - \cdot)$ is the classical Hinge Loss (Figure 1.1, center). The loss function $H_1(|\cdot|)$ for the unlabeled examples can be seen in Figure 1.6, left. For $C^* = 0$ in (1.10) 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

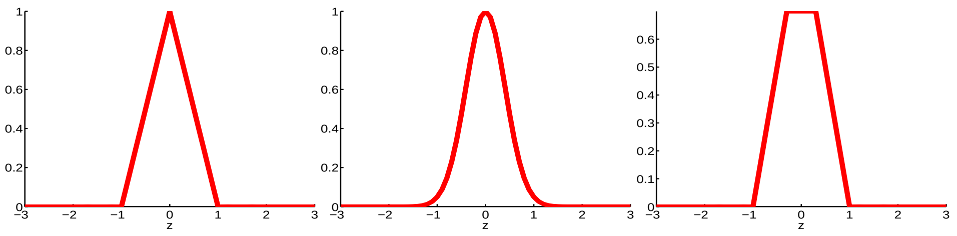

Figure 1.6 Three loss functions for unlabeled examples, from left to right (i) the Symmetric Hinge $H_1(|t|) = \max(0, 1 - |t|)$, (ii) Symmetric Sigmoid $S(t) = \exp(-3t^2)$; and (iii) Symmetric Ramp loss, $R_s(|t|) = \min(1 + s, \max(0, 1 - |t|))$. The last loss function has a plateau of width $2|s|$ where $s \in (-1, 0]$ is a tunable parameter, in this case $s = -0.3$.

assume the label for the unlabeled example is $y_i = \text{sign}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$.

1.5.3 Losses for transduction

TSVMs implementing formulation (1.10) were first introduced in SVM^{light} (Joachims, 1999b). As shown above, it assigns a Hinge Loss $H_1(\cdot)$ on the labeled examples (Figure 1.1, center) and a "Symmetric Hinge Loss" $H_1(|\cdot|)$ on the unlabeled examples (Figure 1.6, left). More recently, Chapelle and Zien (2005) proposed to handle unlabeled examples with a smooth version of this loss (Figure 1.6, 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 1.6, right). Given an unlabeled example $\boldsymbol{x}$ and using the notation $z = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, this loss can be written as[4]

$$z \mapsto R_s(z) + R_s(-z) + const. \tag{1.11}$$

where $-1 < s \leq 0$ is a hyper-parameter to be chosen and $R_s = \min(1-s, \max(0, 1-t))$ is the "Ramp Loss" we already introduced in Section 1.3.3, Figure 1.1. 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 (1.11) 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 1.6, right) which can be viewed as a simplification of Chapelle's loss function. We call this loss function (1.11) the "Symmetric Ramp Loss".

Training a TSVM using the loss function (1.11) is equivalent to training an 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

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

labeled with both possible classes. More formally, after introducing

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

we can rewrite (1.10) as

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{1.12}$$

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

1.5.4 Balancing constraint

One problem with TSVM as stated above is that in high dimensions with few training examples, 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:

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \;=\; \frac{1}{L}\sum_{i=1}^{L} y_i\,. \tag{1.13}$$

1.5.5 CCCP for TSVMs

As for non-convex SVMs in Section 1.3 the decomposition (1.7) of the Ramp Loss makes the TSVM minimization problem as stated in (1.12) 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:

$$\begin{aligned} J^s(\boldsymbol{\theta}) &= \frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \\ &= \underbrace{\frac{1}{2}\,\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{vex}(\boldsymbol{\theta})} \\ &\quad \underbrace{-C^* \sum_{i=L+1}^{L+2U} H_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))}_{J^s_{cav}(\boldsymbol{\theta})}\;. \end{aligned} \tag{1.14}$$

This decomposition allows us to apply the CCCP procedure as stated in Algorithm 1.1. The convex optimization problem (1.1) 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 (Collobert et al., 2006) that enforcing the balancing constraint (1.13) 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 follows:

$$K_{i0} = K_{0i} = \frac{1}{U}\sum_{j=L+1}^{L+U} \Phi(\boldsymbol{x}_j)\cdot\Phi(\boldsymbol{x}_i) \quad \forall i\,. \tag{1.15}$$

The computation of this special column can be achieved very efficiently by computing it only once, or by approximating the sum (1.15) using an appropriate sampling method.

Given the decomposition of the cost (1.14) and the trick of the special extra example (1.15) to enforce the balancing constraint, we can easily apply Algorithm 1.1 to TSVMs. To simplify the first order approximation of the concave part in the CCCP procedure (1.1), we denote as in Section 1.3.4

$$\beta_i \quad = \quad y_i\,\frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)} \quad = \quad \begin{cases} C^* & \text{if } \; y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) < s \\ 0 & \text{otherwise} \end{cases}\,, \tag{1.16}$$

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 1.3, after some standard derivations detailed in (Collobert et al., 2006). This algorithm is very similar to Algorithm 1.2 which we obtained in the case of non-convex SVMs.

Algorithm 1.3 : 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 $K_{ij} = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)$ ) and get $\boldsymbol{\alpha}^{t+1}$
      $\max_{\boldsymbol{\alpha}} \left(\boldsymbol{\alpha}\cdot\boldsymbol{\zeta} - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{K}\,\boldsymbol{\alpha}\right)$ 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 $f_{\boldsymbol{\theta}^{t+1}}(x_i) = \sum_{j=0}^{L+2U} \alpha_j^{t+1} K_{ij} + b^{t+1}$ and
      $\forall i \leq L\ :\quad 0 < y_i\,\alpha_i < C \implies y_i\, f_{\boldsymbol{\theta}^{t+1}}(x_i) = 1$
      $\forall i > L\ :\quad -\beta_i^t < y_i\,\alpha_i < C^* - \beta_i^t \implies y_i\, f_{\boldsymbol{\theta}^{t+1}}(x_i) = 1$
  • 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 $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t$
```

1.5.6 Complexity

The main point we want to emphasize is the advantage in terms of training time of our method compared to existing approaches. Training a CCCP-TSVM amounts to solving a series of SVM optimization problems with $L + 2U$ variables. As we highlighted in Section 1.3.5 SVM training has a worst case complexity of $O((L + 2U)^3)$ but typically scales quadratically, 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 1.7, Figure 1.9 and Figure 1.10). From our experience, around five iteration steps are usually sufficient to reach the minimum, as shown in Section 1.6 of this chapter, Figure 1.8.

1.5.7 Previous Work

1.5.7.1 SVMLight-TSVM

Like our work, the heuristic optimization algorithm implemented in SVM^{light} (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 $\mathbf{x}_i$ and $\mathbf{x}_j$ 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. However, 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 1.7).

SVM^{light} 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.

1.5.7.2 ∇TSVM

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

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

where $H^2(t) = \max(0, 1-t)^2$ and $H^*(t) = \exp(-3t^2)$ (cf. Figure 1.6, center). This optimization problem can be considered a smoothed version of (1.10). ∇TSVM also has similar heuristics for C^* as SVM^{light}-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 $O((U + L)^3)$. The KPCA calculation alone also has a time complexity of $O((U + L)^3)$. This method also requires one to store the entire kernel matrix in memory, which clearly becomes infeasible for large datasets.

1.5.7.3 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 (1.10): minimize

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

subject to

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

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

$$-(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i^*, \quad i = L+1, \ldots, L+U$$

$$\xi_i \geq 0, \xi_i^* \geq 0.$$

The last term of the objective function is nonlinear and corresponds to the loss function given in Figure 1.6, 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 (1.13). 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.

1.6 Experiments with TSVMs

1.6.1 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, $\Phi(\mathbf{x})\cdot\Phi(\mathbf{x}') = \exp(-\gamma||\mathbf{x}-\mathbf{x}'||^2)$.

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

	Coil20	g50c	Text	Uspst	(number of hyperparameters)
SVM	24.64	8.32	18.86	23.18	2
SVM^{light}-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 1.3 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). SVM^{light}-TSVM is the implementation in SVM^{light}. ∇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, that is 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.

Table 1.2 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 (Nene et al., 1996). The `text` dataset consists of the classes `mswindows` and `mac` of the `Newsgroup20` dataset preprocessed as in (Szummer and Jaakkola, 2002). 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. 5B

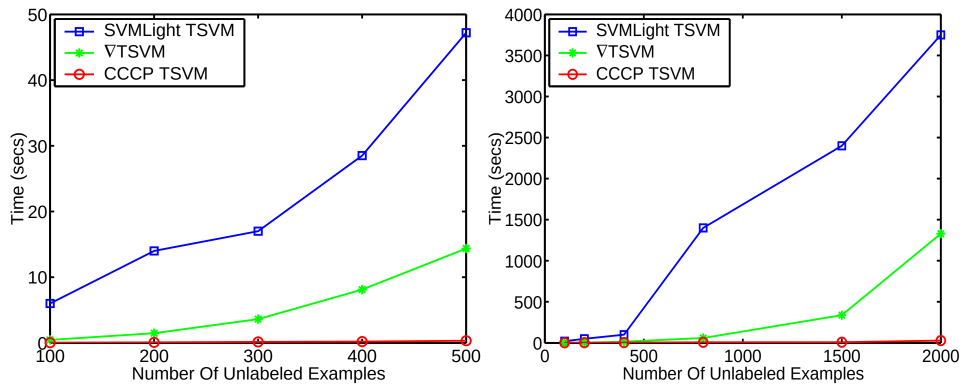

Figure 1.7 Training times for `g50c` (left) and `text` (right) for SVM^{light}-TSVMs, ∇TSVMs and CCCP-TSVMs using the best parameters for each algorithm as measured on the test set in a single trial. For the Text dataset, using 2000 unlabeled examples CCCP-TSVMs are **133x** faster than SVM^{light}-TSVMs, and **50x** faster than ∇TSVMs.

1.6.1.1 Accuracies

Following Chapelle and Zien (2005), all hyperparameters are tuned on the test set (they do this for all methods but one, LDS, for which they perform cross-validation). We should be cautious when comparing our algorithm in this way, especially when the algorithms have different sets of hyperparameters. For CCCP-TSVMs, compared to the other algorithms, 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.
- CCCP-TSVM $|^{s=0}_{UC^*=LC}$ 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 1.3. CCCP-TSVM in all three scenarios achieves approximately the same error rates as ∇TSVM and appears to be superior to SVM^{light}-TSVM.

1.6.1.2 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.

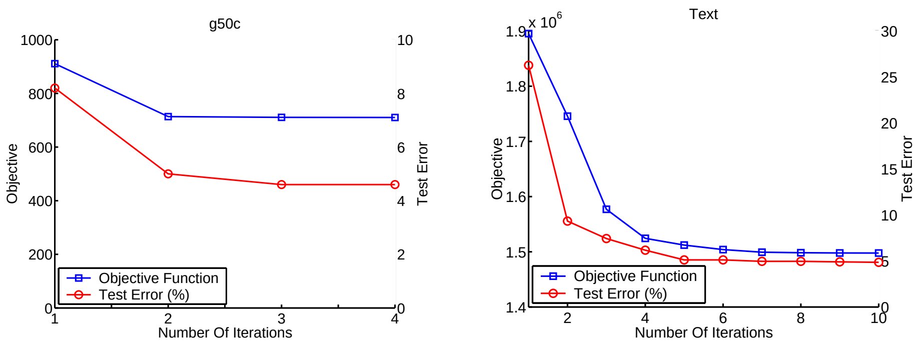

Figure 1.8 Value of the objective function and test error during the CCCP iterations of training TSVM on two datasets (single trial), `g50c` (left) and `text` (right). CCCP-TSVM tends to converge after only a few iterations.

The CCCP-TSVM algorithm was implemented in C++.[5] The successive convex optimizations are performed using a state-of-the-art SMO implementation. Without further optimization, CCCP-TSVMs run orders of magnitude faster than SVM^{light}-TSVMs and ∇TSVM.[6] Figure 1.7 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 SVM^{light}-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, and is therefore clearly infeasible for some of the large scale experiments we attempt in Section 1.6.2.

Finally, Figure 1.8 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.

1.6.2 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. Source code available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.
6. ∇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`.

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

1.6.2.1 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 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. This was done 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 reported 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 1.4. The best result we obtained for 1000 training points was 10.58% test error, when using 10500 unlabeled points, and for 100 training points was 10.42% when using 9500 unlabeled points. Compared to the best performance of an 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. This is especially true in the case of few training examples, 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.

Method	Training 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 1.5 Comparing CCCP-TSVMs with SVMs on the MNIST problem for different number of labeled and unlabeled examples. See text for details.

Figure 1.9 shows the training time of CCCP optimization as a function of the

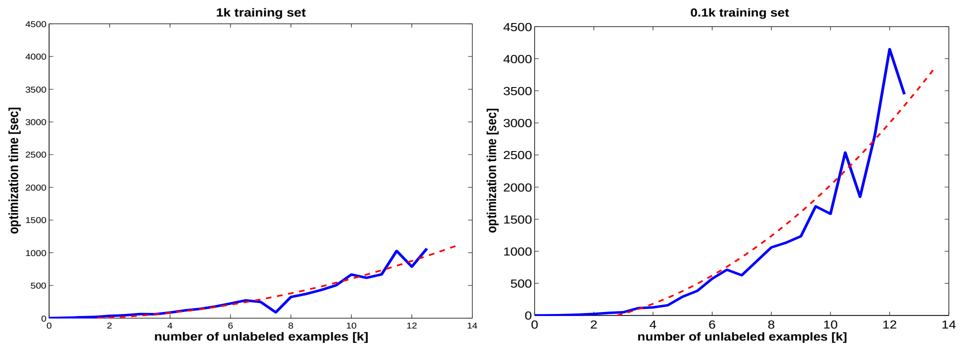

Figure 1.9 Optimization time for the Reuters dataset as a function of the number of unlabeled data. The algorithm was trained on 1,000 points (left) and on 100 points (right). The dashed lines represent a parabola fitted at the time measurements.

number of unlabeled examples. On a 64 bit Opteron processor the optimization time for 12500 unlabeled examples was approximately 18 minutes using the 1000 training examples and 69 minutes using 100 training examples. 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 show a quadratic trend.

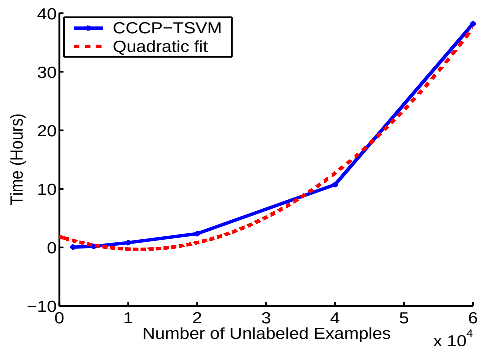

Figure 1.10 Optimization time for the MNIST dataset as a function of the number of unlabeled data. The algorithm was trained on 1,000 labeled examples and up to 60,000 unlabeled examples. The dashed lines represent a polynomial of degree two with a least square fit on the algorithm's time measurements.

1.6.2.2 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 plus 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 by using a separate validation set of size 1000. For TSVMs, for efficiency reasons we fixed the values of 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 to changes in the unlabeled set, and kept the other parameters fixed. For the larger labeled set we took 2000, 5000, 10000, 20000, 40000 and 60000 unlabeled examples. We always measure the error rate on the complete test set. The test error rate and parameter choices for each experiment are given in the Table 1.5, and the training times are given in Figure 1.10.

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, and that gains are still possible with more realistic dataset sizes.

1.7 Conclusion

We described two non-convex algorithms using CCCP that bring marked scalability improvements over the corresponding convex approaches, namely for SVMs and TSVMs. Moreover, any new improvements to standard SVM training could immediately be applied to either of our CCCP algorithms.

In general, we argue that the current popularity of convex approaches should not dissuade researchers from exploring alternative techniques, as they sometimes give clear computational benefits.

Acknowledgements

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.

References

Yoshua Bengio, Nicolas Le Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 123–130. MIT Press, Cambridge, MA, 2006.

Kristin P. Bennett and Ayhan 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.

Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

Philippe G. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation.* Paris, Masson, 1990.

Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

Tijl De Bie and Nello 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.

Luc Devroye, Lázló Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of Mathematics.* New York, Springer Verlag, 1996.

André Elisseeff. *Étude de la complexité et contrôle de la capacité des systèmes d'apprentissage: SVM multi-classe, réseaux de regularization et réseaux de neurones multi-couche.* PhD thesis, ENS-Lyon, France, 2000.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–146, San Francisco, Morgan Kaufmann, 1996.

Glenn Fung and Olvi L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

Thorsten Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support*

Vector Learning, pages 169–184. MIT Press, Cambridge, MA, 1999a.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, San Francisco, Morgan Kaufmann, 1999b.

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

Yann Le Cun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In G.B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer Verlag, New York, 1998.

Hoai An 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, France, 1994.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Yufeng Liu, Xiaotong Shen, and Hani Doss. Multicategory ψ-learning and support vector machine: Computational tools. *Journal of Computational & Graphical Statistics*, 14(1):219–236, 2005.

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

Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image libary (coil-20). Technical Report CUS-005-96, Columbia University, New York, February 1996.

Fernando Perez-Cruz, Angel Navia-Vazquez, Aníbal R. Figueiras-Vidal, and Antonio Artes-Rodriguez. Empirical risk minimization for support vector classifiers. *IEEE Transactions on Neural Networks*, 14(3):296–303, 2002.

John Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, MIT Press, 1999.

Bernhard Schölkopf, Smola Alexander J, and Klaus-Robert Müller. Kernel principal component analysis. In *Proceedings ICANN97*, Springer Lecture Notes in Computer Science, page 583, 1997.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. Cambridge MA, MIT Press, 2002.

Xiaotong Shen, George C. Tseng, Xuegong Zhang, and Wing Hung Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

Alexander J. Smola, S. V. N. Vishwanathan, and Thomas Hofmann. Kernel methods for missing variables. In *Proceedings of the 10th International Workshop* *on Artificial Intelligence and Statistics*, 2005.

Ingo Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003.

Ingo Steinwart and Clint Scovel. Fast rates to bayes for kernel machines. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1345–1352, Cambridge, MA, MIT Press, 2005.

Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, MIT Press, 2002.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* New York, Springer Verlag, 1995.

Linli Xu, Koby Crammer, and Dale Schuurmans. Robust support vector machine training via convex outlier ablation. In *The 21st National Conference on Artificial Intelligence, AAAI*, 2006.

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

Alan L. Yuille and Anand 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, MIT Press, 2002.