Journal of Machine Learning Research 9 (2008) 203-233 Submitted 5/07; Published 2/08

Optimization Techniques for Semi-Supervised Support Vector Machines

Olivier Chapelle[*] CHAP@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 95054

Vikas Sindhwani VIKASS@CS.UCHICAGO.EDU
University of Chicago, Department of Computer Science
1100 E 58th Street
Chicago, IL 60637

Sathiya S. Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 95054

Editor: Nello Cristianini

Abstract

Due to its wide applicability, the problem of semi-supervised classification is attracting increasing attention in machine learning. Semi-Supervised Support Vector Machines (S^{3}VMs) are based on applying the margin maximization principle to both labeled and unlabeled examples. Unlike SVMs, their formulation leads to a non-convex optimization problem. A suite of algorithms have recently been proposed for solving S^{3}VMs. This paper reviews key ideas in this literature. The performance and behavior of various S^{3}VM algorithms is studied together, under a common experimental setting.

Keywords: semi-supervised learning, support vector machines, non-convex optimization, transductive learning

1. Introduction

In many applications of machine learning, abundant amounts of data can be cheaply and automatically collected. However, manual labeling for the purposes of training learning algorithms is often a slow, expensive, and error-prone process. The goal of semi-supervised learning is to employ the large collection of unlabeled data jointly with a few labeled examples for improving generalization performance.

The design of Support Vector Machines (SVMs) that can handle partially labeled data sets has naturally been a vigorously active subject. A major body of work is based on the following idea: solve the standard SVM problem while treating the unknown labels as additional optimization variables. By maximizing the margin in the presence of unlabeled data, one learns a decision boundary that traverses through low data-density regions while respecting labels in the input space. In other words, this approach implements the *cluster assumption* for semi-supervised learning—that is,

*. Most of the work was done while at MPI for Biological Cybernetics,Tübingen, Germany.

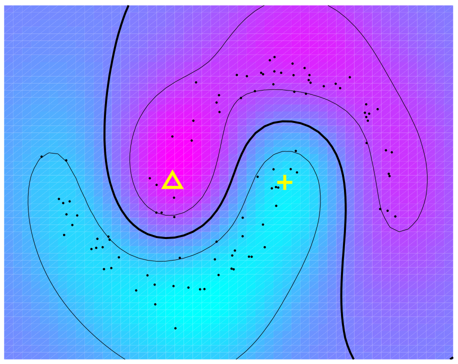

Figure 1: Two moons. There are 2 labeled points (the triangle and the cross) and 100 unlabeled points. The global optimum of S^{3}VM correctly identifies the decision boundary (black line).

points in a data cluster have similar labels (Seeger, 2006; Chapelle and Zien, 2005). Figure 1 illustrates a low-density decision surface implementing the cluster assumption on a toy two-dimensional data set. This idea was first introduced by Vapnik and Sterin (1977) under the name *Transductive SVM*, but since it learns an inductive rule defined over the entire input space, we refer to this approach as *Semi-Supervised SVM* (S^{3}VM).

Since its first implementation by Joachims (1999), a wide spectrum of techniques have been applied to solve the non-convex optimization problem associated with S^{3}VMs, for example, local combinatorial search (Joachims, 1999), gradient descent (Chapelle and Zien, 2005), continuation techniques (Chapelle et al., 2006a), convex-concave procedures (Fung and Mangasarian, 2001; Collobert et al., 2006), semi-definite programming (Bie and Cristianini, 2006; Xu et al., 2004), non-differentiable methods (Astorino and Fuduli, 2007), deterministic annealing (Sindhwani et al., 2006), and branch-and-bound algorithms (Bennett and Demiriz, 1998; Chapelle et al., 2006c).

While non-convexity is partly responsible for this diversity of methods, it is also a departure from one of the nicest aspects of SVMs. Table 1 benchmarks the empirical performance of various S^{3}VM implementations against the globally optimal solution obtained by a Branch-and-Bound algorithm. These empirical observations strengthen the conjecture that the performance variability of S^{3}VM implementations is closely tied to their susceptibility to sub-optimal local minima. Together with several subtle implementation differences, this makes it challenging to cross-compare different S^{3}VM algorithms.

The aim of this paper is to provide a review of optimization techniques for semi-supervised SVMs and to bring different implementations, and various aspects of their empirical performance, under a common experimental setting.

In Section 2 we discuss the general formulation of S^{3}VMs. In Sections 3 and 4 we provide an overview of various methods. We present a detailed empirical study in Section 5 and present a discussion on complexity in Section 6.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	BB
`Coil3`	61.6	61	56.6	56.7	61.6	61.5	0
`2moons`	61	37.7	63.1	68.8	22.5	11	0

Table 1: Generalization performance (error rates) of different S^{3}VM algorithms on two (small) data sets `Coil3` and `2moons`. Branch and Bound (BB) yields the globally optimal solution which gives perfect separation. BB can only be applied to small data sets due to its high computational costs. See Section 5 for experimental details.

2. Semi-Supervised Support Vector Machines

We consider the problem of binary classification. The training set consists of l labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$, and u unlabeled examples $\{\mathbf{x}_i\}_{i=l+1}^{n}$, with $n = l + u$. In the linear S^{3}VM classification setting, the following minimization problem is solved over *both* the hyperplane parameters $(\mathbf{w}, b)$ and the label vector $\mathbf{y}_u := [y_{l+1} \dots y_n]^\top$,

$$\min_{(\mathbf{w},b),\ \mathbf{y}_u} I(\mathbf{w}, b, \mathbf{y}_u) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star \sum_{i=l+1}^{n} V(y_i, o_i) \tag{1}$$

where $o_i = \mathbf{w}^\top \mathbf{x}_i + b$ and V is a loss function. The Hinge loss is a popular choice for V,

$$V(y_i, o_i) = \max(0, 1 - y_i o_i)^p. \tag{2}$$

It is common to penalize the Hinge loss either linearly ($p = 1$) or quadratically ($p = 2$). In the rest of the paper, we will consider $p = 2$. Non-linear decision boundaries can be constructed using the *kernel trick* (Vapnik, 1998).

The first two terms in the objective function I in (1) define a standard SVM. The third term incorporates unlabeled data. The loss over labeled and unlabeled examples is weighted by two hyperparameters, C and $C^\star$, which reflect confidence in labels and in the cluster assumption respectively. In general, C and $C^\star$ need to be set at different values for optimal generalization performance.

The minimization problem (1) is solved under the following class balancing constraint,

$$\frac{1}{u}\sum_{i=l+1}^{n} \max(y_i, 0) = r \quad \text{or equivalently} \quad \frac{1}{u}\sum_{i=l+1}^{n} y_i = 2r - 1. \tag{3}$$

This constraint helps in avoiding unbalanced solutions by enforcing that a certain user-specified fraction, r, of the unlabeled data should be assigned to the positive class. It was introduced with the first S^{3}VM implementation (Joachims, 1999). Since the true class ratio is unknown for the unlabeled data, r is estimated from the class ratio on the labeled set, or from prior knowledge about the classification problem.

There are two broad strategies for minimizing I:

1. **Combinatorial Optimization:** For a given fixed $\mathbf{y}_u$, the optimization over $(\mathbf{w}, b)$ is a standard SVM training.[1] Let us define:

$$\mathcal{J}(\mathbf{y}_u) = \min_{\mathbf{w},b} \ I(\mathbf{w}, b, \mathbf{y}_u). \tag{4}$$

1. The SVM training is slightly modified to take into account different values for C and $C^\star$.

The goal now is to minimize $\mathcal{J}$ over a set of binary variables. This combinatorial view of the optimization problem is adopted by Joachims (1999), Bie and Cristianini (2006), Xu et al. (2004), Sindhwani et al. (2006), Bennett and Demiriz (1998), and Chapelle et al. (2006c). There is no known algorithm that finds the global optimum efficiently. In Section 3 we review this class of techniques.

2. **Continuous Optimization:** For a fixed $(\mathbf{w}, b)$, $\arg\min_y V(y, o) = \operatorname{sign}(o)$. Therefore, the optimal $\mathbf{y}_u$ is simply given by the signs of $o_i = \mathbf{w}^\top \mathbf{x}_i + b$. Eliminating $\mathbf{y}_u$ in this manner gives a continuous objective function over $(\mathbf{w}, b)$:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i o_i)^2 + C^\star \sum_{i=l+1}^{n} \max(0, 1 - |o_i|)^2. \tag{5}$$

This form of the optimization problem illustrates how S^{3}VMs implement the cluster assumption. The first two terms in (5) correspond to a standard SVM. The last term (see Figure 2) drives the decision boundary, that is, the zero output contour, away from unlabeled points. From Figure 2, it is clear that the objective function is non-convex.

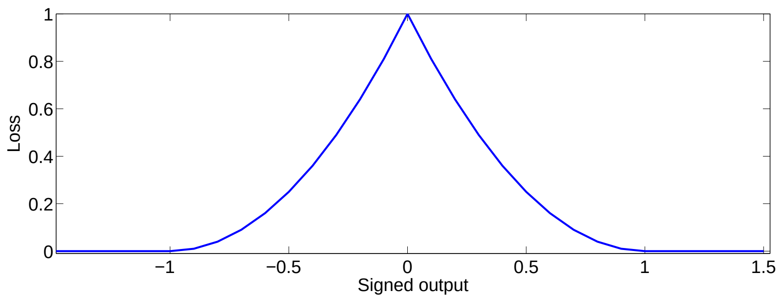

Figure 2: The effective loss $\max(0, 1 - |o|)^2$ is shown above as a function of $o = (\mathbf{w}^\top \mathbf{x} + b)$, the real-valued output at an unlabeled point $\mathbf{x}$.

Note that in this form, the balance constraint becomes $\frac{1}{u}\sum_{i=l+1}^{n} \operatorname{sign}(\mathbf{w}^\top \mathbf{x}_i + b) = 2r - 1$ which is non-linear in $(\mathbf{w}, b)$ and not straightforward to enforce. In Section 4 we review this class of methods (Chapelle and Zien, 2005; Chapelle et al., 2006a; Fung and Mangasarian, 2001; Collobert et al., 2006).

3. Combinatorial Optimization

We now discuss combinatorial techniques in which the labels $\mathbf{y}_u$ of the unlabeled points are explicit optimization variables. Many of the techniques discussed in this section call a standard (or slightly modified) supervised SVM as a subroutine to perform the minimization over $(\mathbf{w}, b)$ for a fixed $\mathbf{y}_u$ (see 4).

3.1 Branch-and-Bound (BB) for Global Optimization

The objective function (4) can be globally optimized using Branch-and-Bound techniques. This was noted in the context of S^{3}VM by Wapnik and Tscherwonenkis (1979) but no details were presented there. In general, global optimization can be computationally very demanding. The technique described in this section is impractical for large data sets. However, with effective heuristics it can produce globally optimal solutions for small-sized problems. This is useful for benchmarking practical S^{3}VM implementations. Indeed, as Table 1 suggests, the exact solution can return excellent generalization performance in situations where other implementations fail completely. Branch-and-Bound was first applied by Bennett and Demiriz (1998) in association with integer programming for solving linear S^{3}VMs. More recently Chapelle et al. (2006c) presented a Branch-and-Bound (BB) algorithm which we outline in this section. The main ideas are illustrated in Figure 3.

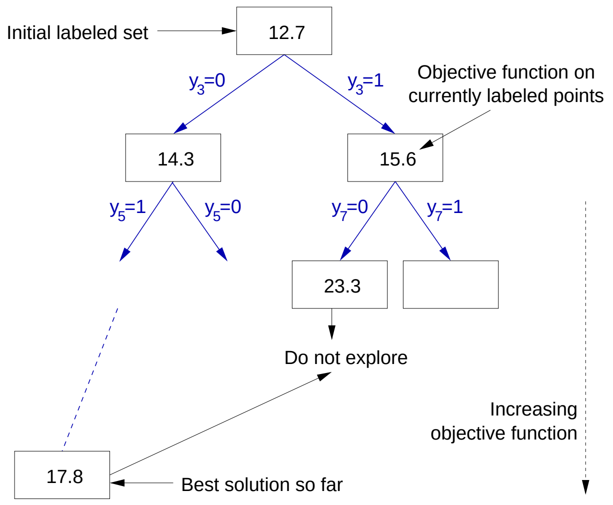

Figure 3: Branch-and-Bound Tree

Branch-and-Bound effectively performs an exhaustive search over $\mathbf{y}_u$, pruning large parts of the solution space based on the following simple observation: suppose that a lower bound on $\min_{\mathbf{y}_u \in A} \mathcal{J}(\mathbf{y}_u)$, for some subset A of candidate solutions, is greater than $\mathcal{J}(\tilde{\mathbf{y}}_u)$ for some $\tilde{\mathbf{y}}_u$, then A can be safely discarded from exhaustive search. BB organizes subsets of solutions into a binary tree (Figure 3) where nodes are associated with a fixed partial labeling of the unlabeled data set and the two children correspond to the labeling of some new unlabeled point. Thus, the root corresponds to the initial set of labeled examples and the leaves correspond to a complete labeling of the data. Any node is then associated with the subset of candidate solutions that appear at the leaves of the subtree rooted at that node (all possible ways of completing the labeling, given the partial labeling at that node). This subset can potentially be pruned from the search by the Branch-and-Bound procedure if a lower bound over corresponding objective values turns out to be worse than an available solution.

The effectiveness of BB depends on the following design issues: (1) the lower *bound* at a node and (2) the sequence of unlabeled examples to *branch* on. For the lower bound, Chapelle et al. (2006c) use the objective value of a standard SVM trained on the associated (extended) labeled set.[2] As one goes down the tree, this objective value increases as additional loss terms are added, and eventually equals $\mathcal{J}$ at the leaves. Note that once a node violates the balance constraint, it can be immediately pruned by resetting its lower bound to ∞. Chapelle et al. (2006c) use a labeling-confidence criterion to choose an unlabeled example and a label on which to branch. The tree is explored on the fly by depth-first search. This confidence-based tree exploration is also intuitively linked to Label Propagation methods (Zhu and Ghahramani, 2002) for graph-transduction. On many small data sets (e.g., Table 1 data sets have up to 200 examples) BB is able to return the globally optimal solution in reasonable amount of time. We point the reader to Chapelle et al. (2006c) for pseudocode.

3.2 S^{3}VM^{light}

S^{3}VM^{light} (Joachims, 1999) refers to the first S^{3}VM algorithm implemented in the popular SVM^{light} software.[3] It is based on local combinatorial search guided by a label switching procedure. The vector $\mathbf{y}_u$ is initialized as the labeling given by an SVM trained on the labeled set, thresholding outputs so that $u \times r$ unlabeled examples are positive. Subsequent steps in the algorithm comprise of switching labels of two examples in opposite classes, thus always maintaining the balance constraint. Consider an iteration of the algorithm where $\mathbf{y}_u$ is the temporary labeling of the unlabeled data and let $(\tilde{\mathbf{w}}, \tilde{b}) = \operatorname{argmin}_{\mathbf{w},b} \; I(\mathbf{w}, b, \mathbf{y}_u)$ and $\mathcal{J}(\mathbf{y}_u) = I(\tilde{\mathbf{w}}, \tilde{b}, \mathbf{y}_u)$. Suppose a pair of unlabeled examples indexed by (i, j) satisfies the following condition,[4]

$$y_i = 1, y_j = -1, V(1, o_i) + V(-1, o_j) > V(-1, o_i) + V(1, o_j) \tag{6}$$

where o_i, o_j are outputs of $(\tilde{\mathbf{w}}, \tilde{b})$ on the examples $\mathbf{x}_i, \mathbf{x}_j$. Then after switching labels for this pair of examples and retraining, the objective function $\mathcal{J}$ can be easily shown to strictly decrease. S^{3}VM^{light} alternates between label-switching and retraining. Since the number of possible $\mathbf{y}_u$ is finite, the procedure is guaranteed to terminate in a finite number of steps at a local minima of (4), that is, no further improvements are possible by interchanging two labels.

In an outer loop, S^{3}VM^{light} gradually increases the value of $C^\star$ from a small value to the final value. Since $C^\star$ controls the non-convex part of the objective function (4), this annealing loop can be interpreted as implementing a "smoothing" heuristic as a means to protect the algorithm from sub-optimal local minima. The pseudocode is provided in Algorithm 1.

3.3 Deterministic Annealing S^{3}VM

Deterministic annealing (DA) is a global optimization heuristic that has been used to approach hard combinatorial or non-convex problems. In the context of S^{3}VMs (Sindhwani et al., 2006), it consists of relaxing the discrete label variables $\mathbf{y}_u$ to real-valued variables $\mathbf{p}_u = (p_{l+1}, \ldots, p_{l+u})$ where p_i is interpreted as the probability that $y_i = 1$. The following objective function is now considered:

$$\begin{aligned} I'(\mathbf{w}, b, \mathbf{p}_u) &= E\left[I(\mathbf{w}, b, \mathbf{y}_u)\right] \qquad (7) \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star \sum_{i=l+1}^{n} p_i V(1, o_i) + (1 - p_i) V(-1, o_i) \end{aligned}$$

2. Note that in this SVM training, the loss terms associated with (originally) labeled and (currently labeled) unlabeled examples are weighted by C and $C^\star$ respectively.
3. Note that in the S^{3}VM literature, this particular implementation is often referred as "TSVM" or "Transductive SVM".
4. This switching condition is slightly weaker than that proposed by Joachims (1999).

Algorithm 1 S^{3}VM^{light}

```
Train an SVM with the labeled points. $o_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i + b$.
Assign $y_i \leftarrow 1$ to the $ur$ largest $o_i$, -1 to the others.
$\tilde{C} \leftarrow 10^{-5} C^\star$
while $\tilde{C} < C^\star$ do
   repeat
      Minimize (1) with $\{y_i\}$ fixed and $C^\star$ replaced by $\tilde{C}$.
      if $\exists (i, j)$ satisfying (6) then
         Swap the labels $y_i$ and $y_j$
      end if
   until No labels have been swapped
   $\tilde{C} \leftarrow \min(1.5C, C^\star)$
end while
```

where E denotes expectation under the probabilities $\mathbf{p}_u$. Note that at optimality with respect to $\mathbf{p}_u$, p_i must concentrate all its mass on $y_i = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$ which leads to the smaller of the two losses $V(1, o_i)$ and $V(-1, o_i)$. Hence, this relaxation step does not lead to loss of optimality and is simply a reformulation of the original objective in terms of continuous variables. In DA, an additional entropy term $-H(\mathbf{p}_u)$ is added to the objective,

$$I''(\mathbf{w}, b, \mathbf{p}_u; T) = I'(\mathbf{w}, b, \mathbf{p}_u) - TH(\mathbf{p}_u)$$
$$\text{where } \; H(\mathbf{p}_u) = -\sum_i p_i \log \; p_i + (1 - p_i) \; \log \; (1 - p_i),$$

and $T \geq 0$ is usually referred to as 'temperature'. Instead of (3), the following class balance constraint is used,

$$\frac{1}{u} \sum_{i=l+1}^{n} p_i = r.$$

Note that when $T = 0$, I'' reduces to (7) and the optimal $\mathbf{p}_u$ identifies the optimal $\mathbf{y}_u$. When $T = \infty$, I'' is dominated by the entropy term resulting in the maximum entropy solution ($p_i = r$ for all i). T parameterizes a family of objective functions with increasing degrees of non-convexity (see Figure 4 and further discussion below).

At any T, let $(\mathbf{w}_T, b_T, \mathbf{p}_{uT}) = \text{argmin}_{(\mathbf{w},b),\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T)$. This minimization can be performed in different ways:

1. *Alternating Minimization*: We sketch here the procedure proposed by Sindhwani et al. (2006). Keeping $\mathbf{p}_u$ fixed, the minimization over $(\mathbf{w}, b)$ is standard SVM training—each unlabeled example contributes two loss terms weighted by $C^\star p_i$ and $C^\star(1 - p_i)$. Keeping $(\mathbf{w}, b)$ fixed, I'' is minimized subject to the balance constraint $\frac{1}{u} \sum_{i=l+1}^{n} p_i = r$ using standard Lagrangian techniques. This leads to:
$$p_i = \frac{1}{1 + e^{(g_i - \nu)/T}} \tag{8}$$
where $g_i = C^\star[V(1, o_i) - V(-1, o_i)]$ and ν, the Lagrange multiplier associated with the balance constraint, is obtained by solving the root finding problem that arises by plugging (8) back in the balance constraint. The alternating optimization proceeds until $\mathbf{p}_u$ stabilizes in a KL-divergence sense. This method will be referred to as DA in the rest of the paper.

2. *Gradient Methods*: An alternative possibility[5] is to substitute the optimal $\mathbf{p}_u$ (8) as a function of $(\mathbf{w}, b)$ and obtain an objective function over $(\mathbf{w}, b)$ for which gradient techniques can be used:

$$\mathcal{S}(\mathbf{w}, b) := \min_{\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T). \tag{9}$$

   $\mathcal{S}(\mathbf{w}, b)$ can be minimized by conjugate gradient descent. The gradient of $\mathcal{S}$ is easy to compute. Indeed, let us denote by $\mathbf{p}_u^*(\mathbf{w}, b)$ the argmin of (9). Then,

$$\frac{\partial \mathcal{S}}{\partial w_i} = \frac{\partial I''}{\partial w_i} + \sum_{j=l+1}^{n} \underbrace{\left.\frac{\partial I''}{\partial p_j}\right|_{\mathbf{p}_u = \mathbf{p}_u^*(\mathbf{w}, b)}}_{0} \frac{\partial p_j^*(\mathbf{w})}{\partial w_i} = \frac{\partial I'(\mathbf{w}, b, \mathbf{p}_u^\star(\mathbf{w}))}{\partial w_i}.$$

   The partial derivative of I'' with respect to p_j is 0 by the definition of $\mathbf{p}_u^*(\mathbf{w}, b)$. The argument goes through even in the presence of the constraint $\frac{1}{u}\sum p_i = r$; see Chapelle et al. (2002, Lemma 2) for a formal proof. In other words, we can compute the gradient of (7) with respect to $\mathbf{w}$ and consider $\mathbf{p}_u$ fixed. The same holds for b. This method will be referred to as ∇DA in the rest of the paper.

Figure 4 shows the effective loss terms in $\mathcal{S}$ associated with an unlabeled example for various values of T. In an outer loop, starting from a high value, T is decreased geometrically by a constant factor. The vector $\mathbf{p}_u$ is then tightened back close to discrete values (its entropy falls below some threshold), thus identifying a solution to the original problem. The pseudocode is provided in Algorithm 2.

Table 2 compares the DA and ∇DA solutions as $T \to 0$ at two different hyperparameter settings.[6] Because DA does alternate minimization and ∇DA does direct minimization, the solutions returned by them can be quite different. Since ∇DA is faster than DA, we only report ∇DA results in the Section 5.

Algorithm 2 DA/∇DA

```
Initialize $p_i = r \quad i = l+1, \ldots, n$
Set $T = 10C^\star$, $R = 1.5$, $\varepsilon = 10^{-6}$.
while $H(\mathbf{p}_{uT}) > \varepsilon$ do
    Solve $(\mathbf{w}_T, b_T, \mathbf{p}_{uT}) = \mathrm{argmin}_{(\mathbf{w},b),\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T)$ subject to: $\frac{1}{u}\sum_{i=l+1}^{n} p_i = r$
    (find local minima starting from previous solution—alternating optimization or gradient methods can be used.)
    $T = T/R$
end while
Return $\mathbf{w}_T, b_T$
```

3.4 Convex Relaxation

We follow Bie and Cristianini (2006) in this section, but outline the details for the squared Hinge loss (see also Xu et al., 2004, for a similar derivation). Rewriting (1) as the familiar constrained

5. Strictly speaking, this approach is more along the lines of methods discussed in Section 4.
6. In Sindhwani et al. (2006), the best solution in the optimization path is returned.

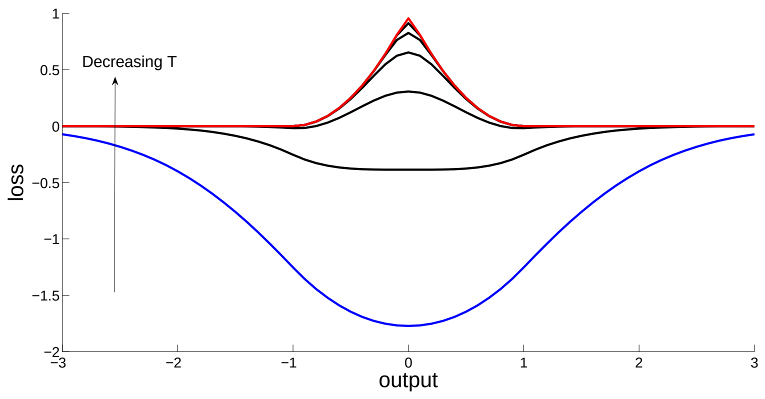

Figure 4: DA parameterizes a family of loss functions (over unlabeled examples) where the degree of non-convexity is controlled by T. As $T \to 0$, the original loss function (Figure 2) is recovered.

	DA	∇DA	DA	∇DA
g50c	6.5	7	8.3	6.7
Text	13.6	5.7	6.2	6.5
Uspst	22.5	27.2	11	11
Isolet	38.3	39.8	28.6	26.9
Coil20	3	12.3	19.2	18.9
Coil3	49.1	61.6	60.3	60.6
2moons	36.8	22.5	62.1	30

Table 2: Generalization performance (error rates) of the two DA algorithms. DA is the original algorithm (alternate optimization on $\mathbf{p}_u$ and $\mathbf{w}$). ∇DA is a direct gradient optimization on $(\mathbf{w}, b)$, where $\mathbf{p}_u$ should be considered as a function of $(\mathbf{w}, b)$. The first two columns report results when $C^\star = C$ and the last columns report results when $C^\star = C/100$. See Section 5 for more experimental details.

optimization problem of SVMs:

$$\min_{(\mathbf{w},b),\mathbf{y}_u} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i^2 + C^\star\sum_{i=l+1}^{n}\xi_i^2 \ \text{ subject to: } \ y_i o_i \geq 1-\xi_i \ \ i=1,\ldots,n.$$

Consider the associated dual problem:

$$\min_{\{y_i\}}\max_{\alpha} \ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K_{ij} \ \text{ subject to: } \ \sum_{i=1}^{n}\alpha_i y_i = 0, \ \ \alpha_i \geq 0$$

where $K_{ij} = \mathbf{x}_i^\top \mathbf{x}_j + D_{ij}$ and D is a diagonal matrix given by $D_{ii} = \frac{1}{2C}, \ i=1,\ldots,l$ and $D_{ii} = \frac{1}{2C^\star}, \ i = l+1,\ldots,n$.

Introducing an $n \times n$ matrix Γ, the optimization problem can be reformulated as:

$$\min_{\Gamma} \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \Gamma_{ij} K_{ij} \tag{10}$$

$$\text{under constraints } \sum \alpha_i y_i = 0, \ \alpha_i \geq 0, \ \Gamma = yy^\top. \tag{11}$$

The objective function (10) is now convex since it is the pointwise supremum of linear functions. However the constraint (11) is not. The idea of the relaxation is to replace the constraint $\Gamma = yy^\top$ by the following set of convex constraints:

$$\begin{aligned} &\Gamma \succeq 0, \\ &\Gamma_{ij} = y_i y_j, \ 1 \leq i,j \leq l, \\ &\Gamma_{ii} = 1, \ l+1 \leq i \leq n. \end{aligned}$$

Though the original method of Bie and Cristianini (2006) does not incorporate the class balancing constraint (3), one can additionally enforce it as $\frac{1}{u^2} \sum_{i,j=l+1}^{n} \Gamma_{ij} = (2r-1)^2$. Such a soft constraint is also used in the continuous S^{3}VM optimization methods of Section 4.

The convex problem above can be solved through Semi-Definite Programming. The labels of the unlabeled points are estimated from Γ (through one of its columns or its largest eigenvector).

This method is very expensive and scales as $O((l+u^2)^2(l+u)^{2.5})$. It is possible to try to optimize a low rank version of Γ, but the training remains slow even in that case. We therefore do not conduct empirical studies with this method.

4. Continuous Optimization

In this section we consider methods which do not include $\mathbf{y}_u$, the labels of unlabeled examples, as optimization variables, but instead solve suitably modified versions of (5) by continuous optimization techniques. We begin by discussing two issues that are common to these methods.

Balancing Constraint The balancing constraint (3) is relatively easy to enforce for all algorithms presented in Section 3. It is more difficult for algorithms covered in this section. The proposed workaround, first introduced by Chapelle and Zien (2005), is to instead enforce a linear constraint:

$$\frac{1}{u} \sum_{i=l+1}^{n} \mathbf{w}^\top \mathbf{x}_i + b = 2\tilde{r} - 1, \tag{12}$$

where $\tilde{r} = r$. The above constraint may be viewed as a "relaxation" of (3). For a given $\tilde{r}$, an easy way of enforcing (12) is to translate all the points such that the mean of the unlabeled points is the origin, that is, $\sum_{i=l+1}^{n} \mathbf{x}_i = 0$. Then, by fixing $b = 2\tilde{r} - 1$, we have an unconstrained optimization problem on $\mathbf{w}$. We will assume that the $\mathbf{x}_i$ are translated and b is fixed in this manner; so the discussion will focus on unconstrained optimization procedures for the rest of this section. In addition to being easy to implement, this linear constraint may also add some robustness against uncertainty about the true unknown class ratio in the unlabeled set (see also Chen et al., 2003, for related discussion).

However, since (12) relaxes (3),[7] the solutions found by algorithms in this section cannot strictly be compared with those in Section 3. In order to admit comparisons (this is particularly important for the empirical study in Section 5), we vary $\tilde{r}$ in an outer loop and do a dichotomic search on this value such that (3) is satisfied.

7. Note that simply setting $\tilde{r} = r$ in (12) will not enforce (3) exactly.

Primal Optimization For linear classification, the variables in $\mathbf{w}$ can be directly optimized. Non-linear decision boundaries require the use of the "kernel trick" (Boser et al., 1992) using a kernel function $k(\mathbf{x},\mathbf{x}')$. While most of the methods of Section 3 can use a standard SVM solver as a sub-routine, the methods of this section need to solve (5) with a non-convex loss function over unlabeled examples. Therefore, they cannot directly use off-the-shelf dual-based SVM software. We use one of the following primal methods to implement the techniques in this section.

Method 1 We find $\mathbf{z}_i$ such that $\mathbf{z}_i \cdot \mathbf{z}_j = k(\mathbf{x}_i,\mathbf{x}_j)$. If B is a matrix having columns $\mathbf{z}_i$, this can be written in matrix form as $B^\top B = K$. The Cholesky factor of K provides one such B. This decomposition was used for ∇S^{3}VM (Chapelle and Zien, 2005). Another possibility is to perform the eigendecomposition of K as $K = U\Lambda U^\top$ and set $B = \Lambda^{1/2}U^\top$. This latter case corresponds to the *kernel PCA map* introduced by Schölkopf and Smola (2002, Section 14.2). Once the $\mathbf{z}_i$ are found, we can simply replace $\mathbf{x}_i$ in (5) by $\mathbf{z}_i$ and solve a linear classification problem. For more details, see Chapelle et al. (2006a).

Method 2 We set $\mathbf{w} = \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i)$ where ϕ denotes a higher dimensional feature map associated with the nonlinear kernel. By the Representer theorem (Schölkopf and Smola, 2002), we indeed know that the optimal solution has this form. Substituting this form in (5) and using the kernel function yields an optimization problem with β as the variables.

Note that the centering mentioned above to implement (12) corresponds to using the modified kernel Schölkopf and Smola (2002, page 431) defined by:

$$k(\mathbf{x},\mathbf{x}') := k(\mathbf{x},\mathbf{x}') - \frac{1}{u}\sum_{i=l+1}^{n} k(\mathbf{x},\mathbf{x}_i) - \frac{1}{u}\sum_{i=l+1}^{n} k(\mathbf{x}',\mathbf{x}_i) + \frac{1}{u^2}\sum_{i,j=l+1}^{n} k(\mathbf{x}_i,\mathbf{x}_j). \tag{13}$$

All the shifted kernel elements can be computed in $O(n^2)$ operations.

Finally, note that these methods are very general and can also be applied to algorithms of Section 3.

4.1 Concave Convex Procedure (CCCP)

The CCCP method (Yuille and Rangarajan, 2003) has been applied to S^{3}VMs by Fung and Mangasarian (2001), Collobert et al. (2006), and Wang et al. (2007). The description given here is close to that in Collobert et al. (2006).

CCCP essentially decomposes a non-convex function f into a convex component f_{vex} and a concave component f_{cave}. At each iteration, the concave part is replaced by a linear function (namely, the tangential approximation at the current point) and the sum of this linear function and the convex part is minimized to get the next iterate. The pseudocode is shown in Algorithm 3.

Algorithm 3 CCCP for minimizing $f = f_{vex} + f_{cave}$

```
Require: Starting point $\mathbf{x}_0$
  $t \leftarrow 0$
  while $\nabla f(\mathbf{x}_t) \neq 0$ do
    $\mathbf{x}_{t+1} \leftarrow \arg\min_{\mathbf{x}} f_{vex}(\mathbf{x}) + \nabla f_{cave}(\mathbf{x}_t) \cdot \mathbf{x}$
    $t \leftarrow t + 1$
  end while
```

In the case of S^{3}VM, the first two terms in (5) are convex. Splitting the last non-convex term corresponding to the unlabeled part as the sum of a convex and a concave function, we have:

$$\max(0,1-|t|)^2 = \underbrace{\max(0,1-|t|)^2+2|t|}_{\text{convex}} \underbrace{-2|t|}_{\text{concave}} .$$

If an unlabeled point is currently classified positive, then at the next iteration, the effective (convex) loss on this point will be

$$\tilde{L}(t) = \begin{cases} 0 & \text{if } t \geq 1, \\ (1-t)^2 & \text{if } |t| < 1, \\ -4t & \text{if } t \leq -1. \end{cases}$$

A corresponding $\tilde{L}$ can be defined for the case of an unlabeled point being classified negative. The CCCP algorithm specialized to S^{3}VMs is given in Algorithm 4. For optimization variables we employ method 1 given at the beginning of this section.

Algorithm 4 CCCP for S^{3}VMs

```
Starting point: Use the w obtained from the supervised SVM solution.
repeat
    $y_i \leftarrow \text{sign}(\mathbf{w}\cdot\mathbf{x}_i+b),\ \ l+1 \leq i \leq n.$
    $(\mathbf{w},b) = \arg\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\max(0,1-y_i(\mathbf{w}\cdot\mathbf{x}_i+b))^2 + C^\star\sum_{i=l+1}^{n}\tilde{L}(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)).$
until convergence of $y_i,\ l+1 \leq i \leq n.$
```

The CCCP method given in Collobert et al. (2006) does not use annealing, that is, increasing $C^\star$ slowly in steps as in S^{3}VM^{light} to help reduce local minima problems. We have however found performance improvements with annealing (see Table 12).

4.2 ∇S^{3}VM

This method is proposed by Chapelle and Zien (2005) to minimize directly the objective function (5) by gradient descent. For optimization variables, method 1 given at the beginning of this section is used. Since the function $t \mapsto \max(0,1-|t|)^2$ is not differentiable, it is replaced by $t \mapsto \exp(-st^2)$, with $s = 5$ (see Figure 5), to get the following smooth optimization problem:[8]

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\max(0,1-y_i(\mathbf{w}\cdot\mathbf{x}_i+b))^2 + C^\star\sum_{i=l+1}^{n}\exp(-s(\mathbf{w}\cdot\mathbf{x}_i+b)^2). \tag{14}$$

As for S^{3}VM^{light}, ∇S^{3}VM performs annealing in an outer loop on $C^\star$. In the experiments we followed the same annealing schedule as in Chapelle and Zien (2005): $C^\star$ is increased in 10 steps to its final value. More precisely, at the ith step, $C^\star$ is set to $2^{i-10}C^\star_{final}$.

4.3 Continuation S^{3}VM (cS^{3}VM)

Closely related to ∇S^{3}VM, Chapelle et al. (2006a) proposes a *continuation* method for minimizing (14). Gradient descent is performed on the same objective function (with the same loss for the

8. Chapelle and Zien (2005) used $s = 3$ with hinge loss, $p = 1$ in (2), but $s = 5$ seems to be a better choice for quadratic hinge loss ($p = 2$).

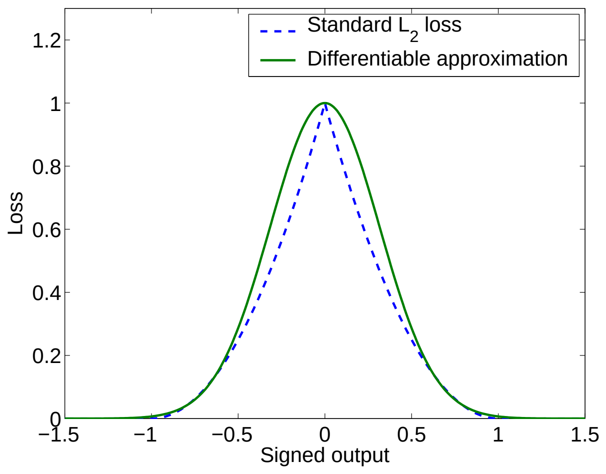

Figure 5: The loss function on the unlabeled points $t \mapsto \max(0, 1-|t|)^2$ is replaced by a differentiable approximation $t \mapsto \exp(-5t^2)$.

unlabeled points as shown in Figure 5), but the method used for annealing is different. Instead of slowly increasing $C^\star$, it is kept fixed, and a continuation technique is used to transform the objective function.

This kind of method belongs to the field of global optimization techniques (Wu, 1996). The idea is similar to deterministic annealing (see Figure 6). A smoothed version of the objective function is first minimized. With enough smoothing the global minimum can hopefully be easily found. Then the smoothing is decreased in steps and the minimum is tracked—the solution found in one step serves as the starting point for the next step. The method is continued until there is no smoothing and so we get back to the solution of (14). Algorithm 5 gives an instantiation of the method in which smoothing is achieved by convolution with a Gaussian, but other smoothing functions can also be used.

Algorithm 5 Continuation method for solving $\min_{\mathbf{x}} f(\mathbf{x})$

```
Require: Function $f : \mathbb{R}^d \mapsto \mathbb{R}$, initial point $\mathbf{x}_0 \in \mathbb{R}^d$
Require: Sequence $\gamma_0 > \gamma_1 > \ldots \gamma_{p-1} > \gamma_p = 0$.
  Let $f_\gamma(\mathbf{x}) = (\pi\gamma)^{-d/2} \int f(\mathbf{x}-\mathbf{t}) \exp(-\|\mathbf{t}\|^2/\gamma) d\mathbf{t}$.
  for $i = 0$ to $p$ do
    Starting from $\mathbf{x}_i$, find local minimizer $\mathbf{x}_{i+1}$ of $f_{\gamma_i}$.
  end for
```

It is not clear if one should only smooth the last non-convex term of (14) (the first two terms are convex) or the whole objective as in Chapelle et al. (2006a). It is noteworthy that since the loss for the unlabeled points is bounded, its convolution with a Gaussian of infinite width tends to the zero function. In other words, with enough smoothing, the unlabeled part of the objective function vanishes and the optimization is identical to a standard SVM.

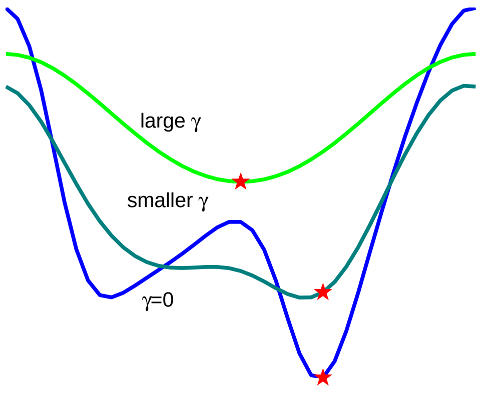

Figure 6: Illustration of the continuation method: the original objective function in blue has two local minima. By smoothing it, we find one global minimum (the red star on the green curve). By reducing the smoothing, the minimum moves toward the global minimum of the original function.

4.4 Newton S^{3}VM

One difficulty with the methods described in sections 4.2 and 4.3 is that their complexity scales as $O(n^3)$ because they employ an unlabeled loss function that does not have a linear part, for example, see (14). Compared to a method like S^{3}VM^{light} (see Section 3.2), which typically scales as $O(n_{\mathrm{sv}}^3 + n^2)$, this can make a large difference in efficiency when n_{sv} (the number of support vectors) is small. In this subsection we propose a new loss function for unlabeled points and an associated Newton method (along the lines of Chapelle, 2007) which brings down the $O(n^3)$ complexity of the ∇S^{3}VM method.

To make efficiency gains we employ method 2 described at the beginning of this section (note that method 1 requires an $O(n^3)$ preprocessing step) and perform the minimization on β, where $\mathbf{w} = \sum_{i=1}^n \beta_i \phi(\mathbf{x}_i)$. Note that the β_i are expansion coefficients and not the Lagrange multipliers α_i in standard SVMs. Let us consider general loss functions, ℓ_L for the labeled points, ℓ_U for the unlabeled points, replace $\mathbf{w}$ by β as the variables in (5), and get the following optimization problem,

$$\min_{\beta} \frac{1}{2}\beta^\top K\beta + C\sum_{i=1}^{l} \ell_L(y_i(K_i^\top\beta + b)) + C^\star \sum_{i=l+1}^{n} \ell_U(K_i^\top\beta + b), \tag{15}$$

where K is the kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and K_i is the i^{th} column of K.[9]

As we will see in detail below, computational time is dictated by n_{sv}, the number of points that lie in the domain of the loss function where curvature is non-zero. With this motivation we choose the differentiable loss function plotted in Figure 7 having several linear and flat parts which are smoothly connected by small quadratic components.[10]

9. Note that, the kernel elements used here correspond to the modified kernel elements in (13).
10. The CCCP method of Collobert et al. (2006) also considers a loss function with a flat middle part, but it was not motivated by computational gains.

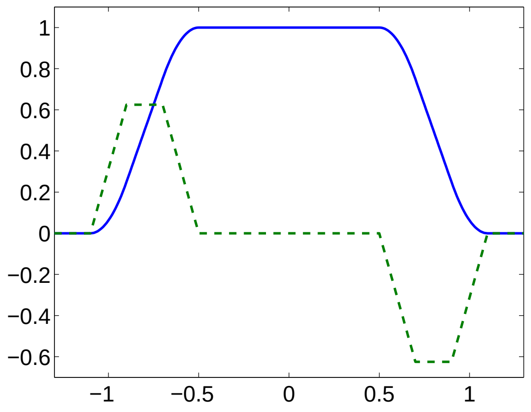

Figure 7: The piecewise quadratic loss function ℓ_U and its derivative (divided by 4; dashed line).

Consider the solution of (15) with this choice of loss function. The gradient of (15) is

$$K\mathbf{g} \;\text{ with }\; g_i = \begin{cases} \beta_i + C\ell'_L(y_i(K_i^\top\beta + b))y_i & 1 \le i \le l \\ \beta_i + C^\star\ell'_U(K_i^\top\beta + b) & l+1 \le i \le n \end{cases}. \tag{16}$$

Using a gradient based method like nonlinear conjugate gradient would sill be costly because each evaluation of the gradient requires $O(n^2)$ effort. To improve the complexity when n_{sv} is small, one can use Newton's method instead. Let us now go into these details.

The Hessian of (15) is

$$K + KDK, \;\text{ with } D \text{ diagonal, }\; D_{ii} = \begin{cases} C\ell''_L(y_i(K_i^\top\beta + b)) & 1 \le i \le l \\ C^\star\ell''_U(K_i^\top\beta + b) & l+1 \le i \le n \end{cases}. \tag{17}$$

The corresponding Newton update is $\beta \leftarrow \beta - (K + KDK)^{-1}K\mathbf{g}$. The advantage of Newton optimization on this problem is that the step can be computed in $O(n_{\mathsf{sv}}^3 + n^2)$ time[11] as we will see below (see also Chapelle, 2007, for a similar derivation). The number of Newton steps required is usually a small finite constant.

A problem in performing a Newton optimization with a non-convex optimization function is that the step might not be a descent direction because the Hessian is not necessarily positive definite. To avoid this problem, we use the Levenberg-Marquardt algorithm (Fletcher, 1987, Algorithm 5.2.7). Roughly speaking, this algorithm is the same as Newton minimization, but a large enough ridge is added to the Hessian such that it becomes positive definite. For computational reasons, instead of adding a ridge to the Hessian, we will add a constant times K.

The goal is to choose a $\lambda \ge 1$ such that $\lambda K + KDK$ is positive definite and solve $(\lambda K + KDK)^{-1}K\mathbf{g}$ efficiently. For this purpose, we reorder the points such that $D_{ii} \ne 0,\ i \le n_{\mathsf{sv}}$ and $D_{ii} = 0,\ i > n_{\mathsf{sv}}$. Let A be the Cholesky decomposition of K:[12] A is the upper triangular matrix satisfying $A^\top A = K$. We suppose that K (and thus A) is invertible. Let us write

$$\lambda K + KDK = A^\top(\lambda I_n + ADA^\top)A.$$

11. Consistent with the way we defined earlier, note here that n_{sv} is the number of "support vectors" where a support vector is defined as a point $\mathbf{x}_i$ such that $D_{ii} \ne 0$.

12. As we will see below, we will not need the Cholesky decomposition of K but only that of of K_{sv}.

The structure of K and D implies that

$$\lambda K + KDK \succ 0 \Leftrightarrow B := \lambda I_{n_{\mathsf{sv}}} + A_{\mathsf{sv}} D_{\mathsf{sv}} A_{\mathsf{sv}}^\top \succ 0,$$

where A_{sv} is the Cholesky decomposition of K_{sv} and K_{sv} is the matrix formed using the first n_{sv} rows and columns of K. After some block matrix algebra, we can also get the step as

$$-(\lambda K + KDK)^{-1} K\mathbf{g} = \begin{pmatrix} A_{\mathsf{sv}}^{-1} B^{-1} A_{\mathsf{sv}} (\mathbf{g}_{\mathsf{sv}} - \frac{1}{\lambda} D_{\mathsf{sv}} K_{\mathsf{sv},\mathsf{nsv}} \mathbf{g}_{\mathsf{nsv}}) \\ \frac{1}{\lambda} \mathbf{g}_{\mathsf{nsv}} \end{pmatrix}, \tag{18}$$

where nsv refers to the indices of the ”non support vectors”, that is, $\{i,\ D_{ii} = 0\}$. Computing this direction takes $O(n_{\mathsf{sv}}^3 + n^2)$ operations. The checking of the positive definiteness of B can be done by doing Cholesky decomposition of K_{sv}. This decomposition can then be reused to solve the linear system involving B. Full details, following the ideas in Fletcher (1987, Algorithm 5.2.7), are given in Algorithm 6.

Algorithm 6 Levenberg-Marquardt method

```
$\beta \leftarrow 0$.
$\lambda \leftarrow 1$.
**repeat**
    Compute $\mathbf{g}$ and $D$ using (16) and (17)
    $\mathsf{sv} \leftarrow \{i,\ D_{ii} \neq 0\}$ and $\mathsf{nsv} \leftarrow \{i,\ D_{ii} = 0\}$.
    $A_{\mathsf{sv}} \leftarrow$ Cholesky decomposition of $K_{\mathsf{sv}}$.
    Do the Cholesky decomposition of $\lambda I_{n_{\mathsf{sv}}} + A_{\mathsf{sv}} D_{\mathsf{sv}} A_{\mathsf{sv}}^\top$. If it fails, $\lambda \leftarrow 4\lambda$ and try again.
    Compute the step $\mathbf{s}$ as given by (18).
    $\rho \leftarrow \frac{\Omega(\beta+\mathbf{s})-\Omega(\beta)}{\frac{1}{2}\mathbf{s}^\top(K+KDK)\mathbf{s}+\mathbf{s}^\top K\mathbf{g}}$.        % If the obj fun $\Omega$ were quadratic, $\rho$ would be 1.
    If $\rho > 0$, $\beta \leftarrow \beta + s$.
    If $\rho < 0.25$, $\lambda \leftarrow 4\lambda$.
    If $\rho > 0.75$, $\lambda \leftarrow \min(1, \frac{\lambda}{2})$.
**until** Norm$(\mathbf{g}) \leq \varepsilon$
```

As discussed above, the flat part in the loss (cf. Figure 7) provides computational value by reducing n_{sv}. But we feel it may also possibly help in leading to better local minimum. Take for instance a Gaussian kernel and consider an unlabeled point far from the labeled points. At the beginning of the optimization, the output on that point will be 0.[13] This unlabeled point does not contribute to pushing the decision boundary one way or the other. This seems like a satisfactory behavior: it is better to wait to have more information before taking a decision on an unlabeled point for which we are unsure. In Table 3 we compare the performance of the flat loss in Figure 7 and the original quadratic loss used in (5). The flat loss yields a huge gain in performance on `2moons`. On the other data sets the two losses perform somewhat similarly. From a computational point of view, the flat part in the loss can sometimes reduce the training time by a factor 10 as shown in Table 3.

5. Experiments

This section is organized around a set of empirical issues:

13. This is true only for balanced problems; otherwise, the output is b.

	Error rate		Training time	
	Flat	Quadratic	Flat	Quadratic
`g50c`	6.1	5.3	15	39
`Text`	5.4	7.7	2180	2165
`Uspst`	18.6	15.9	233	2152
`Isolet`	32.2	27.1	168	1253
`Coil20`	24.1	24.7	152	1244
`Coil3`	61.5	58.4	6	8
`2moons`	11	66.4	1.7	1.2

Table 3: Comparison of the Newton-S^{3}VM method with two different losses: the one with a flat part in the middle (see Figure 7) and the standard quadratic loss (Figure 2). Left: error rates on the unlabeled set; right: average training time in seconds for one split and one binary classifier training (with annealing and dichotomic search on the threshold as explained in the experimental section). The implementations have not been optimized, so the training times only constitute an estimate of the relative speeds.

1. While S^{3}VMs have been very successful for text classification (Joachims, 1999), there are many data sets where they do not return state-of-the-art empirical performance (Chapelle and Zien, 2005). This performance variability is conjectured to be due to local minima problems. In Section 5.3, we discuss the suitability of the S^{3}VM objective function for semi-supervised learning. In particular, we benchmark current S^{3}VM implementations against the exact, globally optimal solution and we discuss whether one can expect significant improvements in generalization performance by better approaching the global solution.

2. Several factors influence the performance and behavior of S^{3}VM algorithms. In Section 5.4 we study their quality of optimization, generalization performance, sensitivity to hyperparameters, effect of annealing and the robustness to uncertainty in class ratio estimates.

3. S^{3}VMs were originally motivated by Transductive learning, the problem of estimating labels of unlabeled examples without necessarily producing a decision function over the entire input space. However, S^{3}VMs are also semi-supervised learners as they are able to handle unseen test instances. In Section 5.5, we run S^{3}VM algorithms in an inductive mode and analyze performance differences between unlabeled and test examples.

4. There is empirical evidence that S^{3}VMs exhibit poor performances on “manifold” type data (where graph-based methods typically excel) or when the data has many distinct sub-clusters (Chapelle and Zien, 2005). We explore the issue of data geometry and S^{3}VM performance in Section 5.6.

At the outset, we point out that this section does not provide an exhaustive cross-comparison between algorithms. Such a comparison would require, say, cross-validation over multiple hyperparameters, randomization over choices of labels, dichotomic search to neutralize balance constraint differences and handling different choices of annealing sequences. This is computationally quite demanding and, more seriously, statistically brittle due to the lack of labeled validation data in semi-supervised tasks. Our goal, therefore, is not so much to establish a ranking of algorithms reviewed in this paper, but rather to observe their behavior under a neutral experimental protocol.

We next describe the data sets used in our experiments. Note that our choice of data sets is biased towards multi-class manifold-like problems which are particularly challenging for S^3VMs. Because of this choice, the experimental results do not show the typically large improvements one might expect over standard SVMs. We caution the reader not to draw the conclusion that S^3VM is a weak algorithm in general, but that it often does not return state-of-the-art performance on problems of this nature.

5.1 Data Sets

Most data sets come from Chapelle and Zien (2005). They are summarized in Table 4.

data set	classes	dims	points	labeled
`g50c`	2	50	550	50
`Text`	2	7511	1946	50
`Uspst`	10	256	2007	50
`Isolet`	9	617	1620	50
`Coil20`	20	1024	1440	40
`Coil3`	3	1024	216	6
`2moons`	2	102	200	2

Table 4: Basic properties of benchmark data sets.

The artificial data set `g50c` is inspired by Bengio and Grandvalet (2004): examples are generated from two standard normal multi-variate Gaussians, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes error is 5%. The real world data sets consist of two-class and multi-class problems. The `Text` data set is defined using the classes `mac` and `mswindows` of the `Newsgroup20` data set preprocessed as in Szummer and Jaakkola (2001). The `Uspst` set contains the test data part of the well-known `USPS` data on handwritten digit recognition. The `Isolet` is a subset of the *ISOLET spoken letter database* (Cole et al., 1990) containing the speaker sets 1, 2 and 3 and 9 confusing letters $\{B,C,D,E,G,P,T,V,Z\}$. In `Coil20` (respectively `Coil3`), the data are gray-scale images of 20 (respectively 3) different objects taken from different angles, in steps of 5 degrees (Nene et al., 1996). The `Coil3` data set has been used first by Chapelle et al. (2006c) and is particularly difficult since the 3 classes are 3 cars which look alike (see Figure 8).

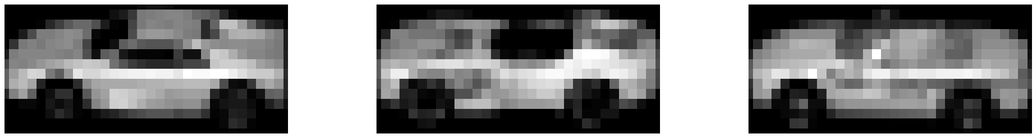

Figure 8: The 3 cars from the COIL data set, subsampled to 32×32

Finally, `2moons` has been used extensively in the semi-supervised learning literature (see for instance Zhu and Ghahramani, 2002, and Figure 1). For this data set, the labeled points are fixed and new unlabeled points are randomly generated for each repetition of the experiment.

5.2 Experimental Setup

To minimize the influence of external factors, unless stated otherwise, the experiments were run in the following normalized way:

Hyperparameters The Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2)$ was used. For simplicity the constant $C^\star$ in (1) was set to C.[14] The same hyperparameters C and σ have been used for all the methods. They are found with cross-validation by training an inductive SVM on the entire data set (the unlabeled points being assigned their real label). These values are reported in Table 5. Even though it seems fair to compare the algorithms with the same hyperparameters, there is a possibility that some regime of hyperparameter settings is more suitable for a particular algorithm than for another. We discuss this issue in section 5.4.3.

Multi-class Data sets with more than two classes are learned with a one-versus-the-rest approach. The reported objective value is the mean of the objective values of the different classifiers. We also conducted experiments on pair-wise binary classification problems (cf. Section 5.6.1) constructed from the multi-class data sets.

Objective value Even though the objective function that we want to minimize is (5), some algorithms like ∇S^{3}VM use another (differentiable) objective function. In order to compare the objective values of the different algorithms, we do the following: after training, we predict the labels of the unlabeled points and train a standard SVM on this augmented labeled set (with $p = 2$ in (2) and C, $C^\star$ weightings on originally labeled and unlabeled terms respectively). The objective value reported is the one of this SVM.

Balancing constraint For the sake of simplicity, we set r in the balancing constraint (3) to the true ratio of the positive points in the unlabeled set. This constraint is relatively easy to enforce for all algorithms presented in Section 3. It is more difficult for algorithms in Section 4 and, for them we used the dichotomic search described at the beginning of Section 4.

For a given data set, we randomly split the data into a labeled set and an unlabeled set. We refer to the error rate on the unlabeled set as the *unlabeled error* to differentiate it from the *test error* which would be computed on an unseen test set. Results are averaged over 10 random splits. The difference between unlabeled and test performance is discussed in Section 5.5.

5.3 Suitability of the S^{3}VM Objective Function

Table 1 shows unlabeled error rates for common S^{3}VM implementations on two small data sets `Coil3` and `2moons`. On these data sets, we are able to also run Branch-and-Bound and get the true globally optimal solution. We see that the global optimum corresponds to a perfect solution, while the local minima found by approximate implementations yield very poor accuracies. From these results, it appears that the minimization of the S^{3}VM objective function makes good sense, even

14. Alternatively, one could set $C^\star = C\frac{l}{u}$ to have equal contribution from labeled and unlabeled points.

	σ	C
g50c	38	19
Text	3.5	31
Uspst	7.4	38
Isolet	15	43
Coil20	2900	37
Coil3	3000	100
2moons	0.5	10

Table 5: Values of the hyperparameters used in the experiments.

though the performance of practical S^{3}VM may not consistently reflect this due to local minima problems.

Table 6 records the rank correlation between unlabeled error and objective function. The rank correlation has been computed in the following way. For each split, we take 10 different solutions and compute the associated unlabeled error and objective value. Ideally, we would like to sample these solutions at random around local minima. But since it is not obvious how to do such a sampling, we simply took the solution given by the different S^{3}VM algorithms as well as 4 “intermediate” solutions obtained as follows. A standard SVM is trained on the original labeled set and a fraction of the unlabeled set (with their true labels). The fraction was either 0, 10, 20 or 30%. The labels of the remaining unlabeled points are assigned through a thresholding of the real value outputs of the SVM. This threshold is such that the balancing constraint (3) is satisfied. Finally, an SVM is retrained using the entire training set. By doing so, we “sample” solutions varying from an inductive SVM trained on only the labeled set to the optimal solution. Table 6 provides evidence that the unlabeled error is correlated with the objective values.

	Coefficient
g50c	0.2
Text	0.67
Uspst	0.24
Isolet	0.23
Coil20	0.4
Coil3	0.17
2moons	0.45

Table 6: Kendall’s rank correlation (Abdi, 2006) between the unlabeled error and the objective function averaged over the 10 splits (see text for details).

5.4 Behavior of S^{3}VM Algorithms

Several factors influence the performance and behavior of S^{3}VM algorithms. We study their quality of optimization, generalization performance, sensitivity to hyperparameters, effect of annealing and the robustness to uncertainty in class ratio estimates.

5.4.1 QUALITY OF MINIMIZATION

In Table 7 we compare different algorithms in terms of minimization of the objective function. ∇S^3VM and cS^3VM appear clearly to be the methods achieving the lowest objective values. However, as we will see in the next section, this does not necessarily translate into lower unlabeled errors.

∇S^3VM	cS^3VM	CCCP	S^3VM^{light}	∇DA	Newton
1.7	1.9	4.5	4.9	4.3	3.7

Table 7: For each data set and each split, the algorithms were ranked according to the objective function value they attained. This table shows the average ranks. These ranks are only about objective function values; error rates are discussed below.

5.4.2 QUALITY OF GENERALIZATION

Table 8 reports the unlabeled errors of the different algorithms on our benchmark data sets. A first observation is that most algorithms perform quite well on `g50c` and `Text`. However, the unlabeled errors on the other data sets, `Uspst`, `Isolet`, `Coil20`, `Coil3`, `2moons` (see also Table 1) are poor and sometimes even worse than a standard SVM. As pointed out earlier, these data sets are particularly challenging for S^3VMs. Moreover, the honors are divided and no algorithm clearly outperforms the others. We therefore cannot give a recommendation on which one to use.

	∇S^3VM	cS^3VM	CCCP	S^3VM^{light}	∇DA	Newton	SVM	SVM-5cv
`g50c`	6.7	6.4	6.3	6.2	7	6.1	8.2	4.9
`Text`	5.1	5.3	8.3	8.1	5.7	5.4	14.8	2.8
`Uspst`	15.6	36.2	16.4	15.5	27.2	18.6	20.7	3.9
`Isolet`	25.8	59.8	26.7	30	39.8	32.2	32.7	6.4
`Coil20`	25.6	30.7	26.6	25.3	12.3	24.1	24.1	0

Table 8: Unlabeled errors of the different S^3VMs implementations. The next to the last column is an SVM trained only on the labeled data, while the last column reports 5 fold cross-validation results of an SVM trained on the whole data set using the labels of the unlabeled points. The values in these two columns can be taken as upper and lower bounds on the best achievable error rates. See Table 1 for results on `Coil3` and `2moons`.

Note that these results may differ from the ones previously reported by Chapelle and Zien (2005), Collobert et al. (2006), Sindhwani et al. (2006), and Chapelle et al. (2006a) on the same data sets. Most of this difference comes from the choice of the hyperparameters. Indeed, as explained below, several algorithms are rather sensitive to the choice of hyperparameters. The exact experimental setting is also different. In results reported elsewhere, r is often estimated from the labeled set and the constraint is often a "soft" one. In Table 8, the hard balance constraint is enforced for all methods assuming r to be known exactly. Finally, in Chapelle et al. (2006a), only pair-wise binary problems were considered for the cS^{3}VM algorithm, while results in Table 8 for multiclass data sets are obtained under a one-versus-the-rest setup.

5.4.3 SENSITIVITY TO HYPERPARAMETERS

As mentioned before, it is possible that different algorithms excel in different hyperparameter regimes. This is more likely to happen due to better local minima at different hyperparameters as opposed to a better global solution (in terms of error rates).

Due to computational constraints, instead of setting the three hyperparameters ($C, C^\star$ and the Gaussian kernel width, σ) by cross-validation for each of the methods, we explored the influence of these hyperparameters on one split of the data. More specifically, Table 9 shows the *relative* improvement (in %) that one can gain by selecting other hyperparameters. These numbers may be seen as a measure of the *robustness* of a method. Note that these results have to be interpreted carefully because they are only on one split: for instance, it is possible that“by chance” the method did not get stuck in a bad local minimum for one of the hyperparameter settings, leading to a larger number in Table 9.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
g50c	31.2	31.2	27.8	13.3	35	7.7
Text	22	7.1	29.2	19.9	34.4	1.1
Uspst	12	70.2	19.2	0	41	15.6
Isolet	7.6	57.6	8	0	4.9	2.6
Coil20	46.4	42.7	27.9	5.8	5.7	16.9
Coil3	32.6	39.2	15.3	20.2	5.8	24.6
2moons	45.6	50	54.1	13.5	23.5	0
Mean	28.2	42.6	25.9	10.4	21.5	9.8

Table 9: On the 1st split of each data set, 27 set of hyperparameters $(\sigma', C', C^{\star\prime})$ have been tested from $\sigma' \in \{\frac{1}{2}\sigma, \sigma, 2\sigma\}, C' \in \{\frac{1}{10}C, C, 10C\}, C^{\star\prime} \in \{\frac{1}{100}C', \frac{1}{10}C', C'\}$. The table shows the *relative* improvement (in %) by taking the best hyperparameters over default ones.

By measuring the variation of the unlabeled errors with respect to the choice of the hyperparameters, Table 10 records an indication of the sensitivity of the method with respect to that choice. From this point of view S^{3}VM^{light} appears to be the most stable algorithm.

∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
6.8	8.5	6.5	2.7	8.4	8.7

Table 10: The variance of the unlabeled errors have been averaged over the 27 possible hyperparameters (cf. Table 9). The table shows those variance averaged over the data sets. A small number shows that a given method is not too sensitive to the choice of the hyperparameters.

Finally, from the experiments in Table 9, we observed that in some cases a smaller value of $C^\star$ is helpful. We have thus rerun the algorithms on all the splits with $C^\star$ divided by 100: see Table 11. The overall performance does not necessarily improve, but the very poor results become better (see for instance `Uspst,Isolet,Coil20` for cS^{3}VM).

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
`g50c`	8.3	8.3	8.5	8.4	6.7	7.5
`Text`	5.7	5.8	8.5	8.1	6.5	14.5
`Uspst`	14.1	15.6	14.9	14.5	11	19.2
`Isolet`	27.8	28.5	25.3	29.1	26.9	32.1
`Coil20`	23.9	23.6	23.6	21.8	18.9	24.6
`Coil3`	60.8	55.0	56.3	59.2	60.6	60.5
`2moons`	65.0	49.8	66.3	68.7	30	33.5

Table 11: Same as Table 8, but with $C^\star$ divided by 100.

5.4.4 EFFECT OF ANNEALING SCHEDULE

All the algorithms described in this paper use some sort of annealing (e.g., gradually decreasing T in DA or increasing $C^\star$ in S^{3}VM^{light} in an outer loop) where the role of the unlabeled points is progressively increased. The three ingredients to fix are:

1. The starting point which is usually chosen in such a way that the unlabeled points have very little influence and the problem is thus almost convex.

2. The stopping criterion which should be such that the annealed objective function and the original objective function are very close.

3. The number of steps. Ideally, one would like to have as many steps as possible, but for computational reasons the number of steps is limited.

In the experimental results presented in Figure 9, we only varied the number of steps. The starting and final values are as indicated in the description of the algorithms. The original CCCP paper did not have annealing and we used the same scheme as for ∇S^{3}VM: $C^\star$ is increased exponentially from $10^{-3}C$ to C. For DA and ∇DA, the final temperature is fixed at a small constant and the decrease rate R is such that we have the desired number of steps. For all algorithms, one step means that there is no annealing.

One has to be cautious when drawing conclusion from this plot. Indeed, the results are only for one data set, one split and fixed hyperparameters. The goal is to get an idea of whether the annealing for a given method is useful; and if so, how many steps should be taken. From this plot, it seems that:

- All methods, except Newton, seem to benefit from annealing. However, on some other data sets, annealing improved the performances of Newton's method (results not shown).

- Most methods do not require a lot of steps. More precisely, we have noticed that the number of steps does not matter as much as the minimization around a "critical" value of the annealing parameter; if that point is missed, then the results can be bad.

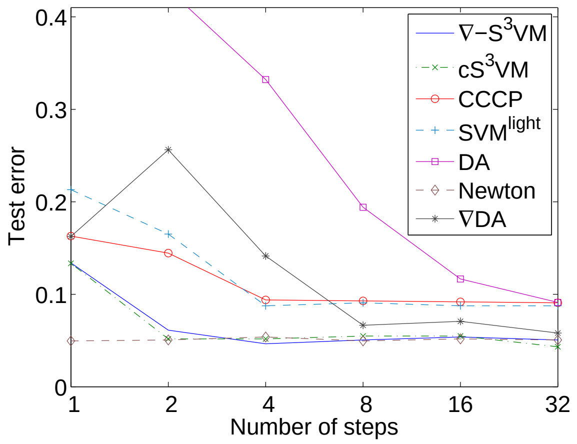

Figure 9: Influence of the number of annealing steps on the first split of `Text`.

	No annealing	Annealing
`g50c`	7.9	6.3
`Text`	14.7	8.3
`Uspst`	21.3	16.4
`Isolet`	32.4	26.7
`Coil20`	26.1	26.6
`Coil3`	49.3	56.6
`2moons`	67.1	63.1

Table 12: Performance of CCCP with and without annealing. The annealing schedule is the same as the one used for ∇S^3VM and Newton: $C^\star$ is increased in 10 steps from its final value divided by 1000.

- DA seems the method which relies the most on a relatively slow annealing schedule, while ∇DA can have a faster annealing schedule.
- CCCP was originally proposed without annealing, but it seems that its performance can be improved by annealing. To confirm this fact, Table 12 compares the results of CCCP with and without annealing on the 10 splits of all the data sets.

The results provided in Table 8 are with annealing for all methods.

5.4.5 Robustness to Class Ratio Estimate

Several results reported in the previous tables are better than, for instance, the results in Chapelle and Zien (2005); Chapelle et al. (2006a). This is because (a) we took into account the knowledge of the true class ratio among the unlabeled examples, and (b) we enforced the constraint (3) exactly (with the dichotomic search described at the beginning of Section 4).

Of course, in practice the true number of positive points is unknown. One can estimate it from the labeled points as:

$$r = \frac{1}{2}\left(\frac{1}{l}\sum_{i=1}^{l} y_i + 1\right).$$

Table 13 presents the generalization performances in this more realistic context where class ratios are estimated as above and original soft balance constraint is used where applicable (recall the discussion in Section 4).

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	SVM
`g50c`	7.2	6.6	6.7	7.5	8.4	5.8	9.1
`Text`	6.8	5	12.8	9.2	8.1	6.1	23.1
`Uspst`	24.1	41.5	24.3	24.4	29.8	25	24.2
`Isolet`	48.4	58.3	43.8	36	46	45.5	38.4
`Coil20`	35.4	51.5	34.5	25.3	12.3	25.4	26.2
`Coil3`	64.4	59.7	59.4	56.7	61.7	62.9	51.8
`2moons`	62.2	33.7	55.6	68.8	22.5	8.9	44.4

Table 13: Constant r estimated from the labeled set. For methods of Section 4, the original constraint (12) is used; there is no dichotomic search (see beginning of Section 4).

5.5 Transductive Versus Semi-supervised Learning

S^{3}VMs were introduced as *Transductive* SVMs, originally designed for the task of directly estimating labels of unlabeled points. However S^{3}VMs provide a decision boundary in the entire input space, and so they can provide labels of unseen test points as well. For this reason, we believe that S^{3}VMs are inductive semi-supervised methods and not strictly transductive. A discussion on the differences between semi-supervised learning and transduction can be found in Chapelle et al. (2006b, Chapter 25).[15]

We expect S^{3}VMs to perform equally well on the unlabeled set and on an unseen test set. To test this hypothesis, we took out 25% of the unlabeled set that we used as a unseen test set. We performed 420 experiments (6 algorithms, 7 data sets and 10 splits). Based on these 420 pairs of error rates, we did not observe a significant difference at the 5% confidence level. Also, for each of the 7 data sets (resp 6 algorithms), there was no statistical significant differences in the 60 (resp 70) pairs of error rates.

Similar experiments were performed by Collobert et al. (2006, Section 6.2). Based on 10 splits, the error rate was found to be better on the unlabeled set than on the test set. The authors made the hypothesis that when the test and training data are not identically distributed, transduction can be helpful. Indeed, because of small sample effect, the unlabeled and test set could appear as not coming from the same distribution (this is even more likely in high dimension).

We considered the `g50c` data set and biased the split between unlabeled and test set such that there is an angle between the principal directions of the two sets. Figure 10 shows a correlation

15. Paper is available at `http://www.kyb.tuebingen.mpg.de/ssl-book/discussion.pdf`.

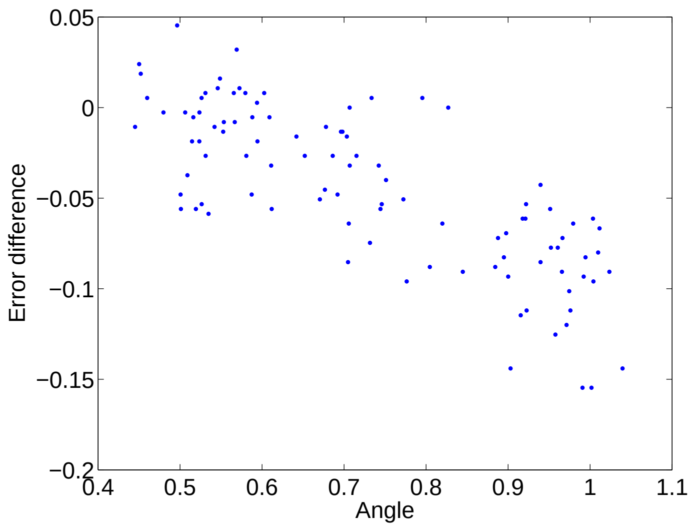

Figure 10: The test set of `g50c` has been chosen with a bias such that there is an angle between the principal directions of the unlabeled and test sets. The figure shows the difference in error rates (negative means better error rate on the unlabeled set) as a function of the angle for different random (biased) splits.

between the difference in error rates and this angle: the error on the test set deteriorates as the angle increases. This confirms the hypothesis stated above.

5.6 Role of Data Geometry

There is empirical evidence that S^{3}VMs exhibit poor performances on "manifold" type data or when the data has many distinct sub-clusters (Chapelle and Zien, 2005). We now explore this issue and propose an hybrid method combining the S^{3}VM and LapSVM (Sindhwani et al., 2005).

5.6.1 EFFECT OF MULTIPLE CLUSTERS

In Chapelle et al. (2006a), cS^{3}VM exhibited poor performance in multiclass problems with one-versus-the-rest training, but worked well on pairwise binary problems that were constructed (using all the labels) from the same multiclass data sets. Note that in practice it is not possible to do semi-supervised one-vs-one multiclass training because the labels of the unlabeled points are truly unknown.

We compared different algorithms on pairwise binary classification tasks for all the multiclass problems. Results are shown in the first 4 rows of Table 14. Except on `Coil3`, most S^{3}VM algorithms show improved performances. There are two candidate explanations for this behavior:

1. The binary classification problems in one-versus-the-rest are unbalanced and this creates difficulties for S^{3}VMs.

	∇S^3VM	cS^3VM	CCCP	S^3VM^{light}	∇DA	Newton	SVM	SVM-5cv
Uspst	1.9	2	2.8	2.9	4	1.9	5.3	0.9
Isolet	4.8	11.8	5.5	5.7	5	6.3	7.3	1.2
Coil20	2.8	3.9	2.9	3.1	2.7	2.4	3.3	0
Coil3	45.8	48.7	40.3	41.3	44.5	47.2	36.7	0
Uspst2	15.6	25.7	16.6	16.1	20	16	17.7	3

Table 14: Experiments in a pairwise binary setting. `Uspst2` is the same set as `Uspst` but where the task is to classify digits 0 to 4 versus 5 to 9.

2. In a one-versus-the-rest approach, the negative class is the concatenation of several classes and is thus made up of several clusters. This might accentuate the local minimum problem of S^3VMs.

To test these hypothesis, we created a binary version of `Uspst` by classifying digits 0 to 4 versus 5 to 9: this data set (`Uspst2` in Table 14) is balanced but each class is made of several clusters. The fact that the S^3VMs algorithms were not able to perform significantly better than the SVM baseline tends to accredit the second hypothesis: S^3VM results deteriorate when there are several clusters per class.

5.6.2 HYBRID S^3VM-GRAPH METHODS

Recall that the results in Table 8 boost the empirical evidence that S^3VMs do not return state-of-the-art performance on "manifold"-like data sets. On these data sets the cluster assumption is presumably satisfied under an intrinsic "geodesic" distance rather than the original euclidean distance between data points. It is reasonable, therefore, to attempt to combine S^3VMs with choices of kernels that conform to the particular geometry of the data.

LapSVM (Sindhwani et al., 2005) is a popular semi-supervised algorithm in which a kernel function is constructed from the Laplacian of an adjacency graph that models the data geometry; this kernel is then used with a standard SVM. This procedure was shown to be very effective on data sets with a manifold structure. In Table 15 we report results with a hybrid S^3VM-graph method: the kernel is constructed as in LapSVM,[16] but then it is plugged into S^3VM^{light}. Such a hybrid was first described in Chapelle et al. (2006a) where cS^3VM was combined with LapSVM.

The results of this hybrid method is very satisfactory, often outperforming both LapSVM and S^3VM^{light}.

Such a method complements the strengths of both S^3VM and Graph-based approaches: the S^3VM adds robustness to the construction of the graph, while the graph enforces the right cluster structure to alleviate local minima problems in S^3VMs. We believe that this kind of technique is probably one of the most robust and powerful way for a semi-supervised learning problem.

16. Hyperparameters were chosen based on experiments in Sindhwani et al. (2005) without any extensive tuning.

		Exact r	(Table 8 setting)	Estimated r	(Table 13 setting)
	LapSVM	S^{3}VM^{light}	LapSVM-S^{3}VM^{light}	S^{3}VM^{light}	LapSVM-S^{3}VM^{light}
`g50c`	6.4	6.2	4.6	7.5	6.1
`Text`	11	8.1	8.3	9.2	9.0
`Uspst`	11.4	15.5	8.8	24.4	19.6
`Isolet`	41.2	30.0	46.5	36.0	49.0
`Coil20`	11.9	25.3	12.5	25.3	12.5
`Coil3`	20.6	56.7	17.9	56.7	17.9
`2moons`	7.8	68.8	5.1	68.8	5.1

Table 15: Comparison of a Graph-based method, LapSVM (Sindhwani et al., 2005), with S^{3}VM^{light} and hybrid LapSVM-S^{3}VM^{light} results under the settings of Table 8 and 13.

6. A Note on Computational Complexity

Even though a detailed comparison of the computational complexity of the different algorithms is out of the scope of this paper, we can still give the following rough picture.

First, all methods use annealing and so the complexity depends on the number of annealing steps. This dependency is probably sublinear because when the number of steps is large, retraining after each (small) step is less costly. We can divide the methods in two categories:

1. Methods whose complexity is of the same order as that of a standard SVM trained with the predicted labels of the unlabeled points, which is $O(n_{\mathsf{sv}}^3 + n^2)$ where n_{sv} is the number of points which are in a non-linear part of the loss function. This is clearly the case for S^{3}VM^{light} since it relies on an SVM optimization. Note that the training time of this algorithm can be sped up by swapping labels in "batch" rather than one by one (Sindhwani and Keerthi, 2006). CCCP is also in this category as the optimization problem it has to solve at each step is very close to an SVM. Finally, even if the Newton method is not directly solved via SVM, its complexity is also $O(n_{\mathsf{sv}}^3 + n^2)$ and we include it in this category. For both CCCP and Newton, the possibility of having a flat part in the middle of the loss function can reduce n_{sv} and thus the complexity. We have indeed observed with the Newton method that the convergence can be an order of magnitude faster when the loss includes this flat part in the middle (see Table 3).

2. Gradient based methods, namely ∇S^{3}VM, cS^{3}VM and ∇DA do not have any linear part in the objective function and so they scale as $O(n^3)$. Note that it is possible to devise a gradient based method and a loss function that contains some linear parts. The complexity of such an algorithm would be $O(nn_{\mathsf{sv}}^2 + n^2)$.

The original DA algorithm alternates between optimization of $\mathbf{w}$ and $\mathbf{p}_u$ and can be understood as block coordinate optimization. We found that it was significantly slower than the other algorithms; its direct gradient-based optimization counterpart, ∇DA, is usually much faster.

Finally, note that even if the algorithms of the second category have a complexity of $O(n^3)$, one can find an approximate solution by reducing the dimensionality of the problem from n to m and get a complexity of $O(nm^2)$. For instance, Chapelle et al. (2006a) reports a speed-up of 100 times without loss in accuracy for cS^{3}VM.

Ultimately a semi-supervised learning algorithm should be able to handle data sets with millions of unlabeled points. The best way of scaling up S^{3}VMs is still an open question and should be the topic of future research.

7. Conclusion

When practical S^{3}VM implementations fail to give good results on a problem, one might suspect that either: (a) the cluster assumption does not hold; or, (b) the cluster assumption holds but local minima problems are severe; or, (c) the S^{3}VM objective function is unable to implement the cluster assumption. We began our empirical study by benchmarking current S^{3}VM implementations against a global optimizer. Our results (see Section 5.3) narrowed the cause for performance loss down to suboptimal local minima, and established a correlation between generalization performance and the S^{3}VM objective function. For problems where the cluster assumption is true, we expect the S^{3}VM objective to indeed be an appropriate quantity to minimize. Due to non-convexity however, this minimization is not completely straightforward—an assortment of optimization techniques have been brought to bear on this problem with varying degrees of success. In this paper, we have reviewed these techniques and studied them empirically, taking several subtle differences into account. In a neutral experimental protocol, we were unable to identify any single technique as being consistently superior to another in terms of generalization performance. We believe better methods are still needed to optimize the S^{3}VM objective function.

While S^{3}VMs return good performance on textual data sets, they are currently not competitive with graph-methods on domains such as image classification often characterized by multiple, highly non-Gaussian clusters. A particularly promising class of techniques (see Section 5.6.2) is based on combining S^{3}VMs with graph methods.

S^{3}VMs have been sparingly explored in domains other than text and image classification. New application domains may provide additional insight into the behavior of S^{3}VM methods while enhancing their general appeal for semi-supervised learning. Another major theme is the extension of S^{3}VMs for structured output problems, possibly building on one of several recent lines of work for handling complex supervised learning tasks. A first step towards such an extension would require a clear statement of the cluster assumption applicable to the semi-supervised structured output setting. These are fertile areas for future research.

References

H. Abdi. Kendall rank correlation. In N.J. Salkind, editor, *Encyclopedia of Measurement and Statistics*. SAGE, 2006.

A. Astorino and A. Fuduli. Nonsmooth optimization techniques for semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2135–2142, 2007.

Y. Bengio and Y. Grandvalet. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 12*, 1998.

T. De Bie and N. Cristianini. Semi-supervised learning using semi-definite programming. In O. Chapelle, B. Schoëlkopf, and A. Zien, editors, *Semi-supervised Learning*. MIT Press, 2006.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

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

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised SVMs. In *International Conference on Machine Learning*, 2006a.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006b. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, 2006c.

Y. Chen, G. Wang, and S. Dong. Learning with progressive transductive support vector machine. *Pattern Recognition Letter*, 24(12):1845–1855, 2003.

R. Cole, Y. Muthusamy, and M. Fanty. The ISOLET spoken letter database. Technical Report CS/E 90-004, Oregon Graduate Institute, 1990.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 1987.

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

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

H. Liu and S.-T. Huang. Fuzzy transductive support vector machines for hypertext classification. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(1):21–36, 2004.

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

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

M. Seeger. A taxonomy of semi-supervised learning methods. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Lerning*. MIT Press, 2006.

M. Silva, T. Maia, and A. Braga. An evolutionary approach to transduction in support vector machines. In *Fifth International Conference on Hybrid Intelligent Systems*, pages 329–334, 2005.

V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR*, 2006.

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

V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *International Conference on Machine Learning*, 2006.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

V. Vapnik and A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.

L. Wang, X. Shen, and W. Pan. On transductive support vector machines. In J. Verducci, X. Shen, and J. Lafferty, editors, *Prediction and Discovery*. American Mathematical Society, 2007.

W. Wapnik and A. Tscherwonenkis. *Theorie der Zeichenerkennung*. Akademie Verlag, Berlin, 1979.

Z. Wu. The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM Journal on Optimization*, 6(3): 748–768, 1996.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.

A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report 02-107, CMU-CALD, 2002.