A Continuation Method for Semi-Supervised SVMs

Olivier Chapelle OLIVIER.CHAPELLE@TUEBINGEN.MPG.DE
Mingmin Chi MINGMIN.CHI@TUEBINGEN.MPG.DE
Alexander Zien ALEXANDER.ZIEN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Abstract

Semi-Supervised Support Vector Machines (S^3VMs) are an appealing method for using unlabeled data in classification: their objective function favors decision boundaries which do not cut clusters. However their main problem is that the optimization problem is non-convex and has many *local minima*, which often results in suboptimal performances. In this paper we propose to use a global optimization technique known as *continuation* to alleviate this problem. Compared to other algorithms minimizing the same objective function, our continuation method often leads to lower test errors.

1. Introduction

While classical Support Vector Machines (SVMs) are usually a powerful tool in supervised classification tasks, in many problems labeled data are rather scarce, and it is desirable to additionally take advantage of training examples without labels. It has early been proposed (Vapnik & Sterin, 1977) to extend the margin maximizing property of SVMs to unlabeled data points. This favors classification boundaries in low-density areas, which makes a lot of sense if one believes in the "cluster assumption" (Seeger, 2006; Chapelle & Zien, 2005): that usually each cluster of data points belongs to a single class.

Algorithms implementing this principle have been introduced in (Vapnik, 1998; Joachims, 1999) under the name *Transductive Support Vector Machine* (TSVM), and in (Bennett & Demiriz, 1998; Fung & Mangasarian, 2001) under the name *Semi-Supervised Support Vector Machine* (S^3VM). In this paper we use the latter name since, according to our understanding, this algorithm is actually *inductive*. Further implementations include (Chapelle & Zien, 2005; Astorino & Fuduli, 2005; Collobert et al., 2006; Sindhwani et al., 2006).

One reason for the large number of proposed algorithms is that the resulting optimization problem is non-convex. This raises the problem of local minima. As we demonstrate, better local minima indeed tend to lead to higher prediction accuracy. The purpose of this paper is to present an optimization method that can usually find better local minima than competing methods. By doing so, we answer two of the open questions that Thorsten Joachims stated in the first S^3VM implementation (Joachims, 1999),

> [...] more research in algorithms for training TSVMs is needed. How well does the algorithm presented here approximate the global solution? Will the results get even better, if we invest more time into search ?

Section 2 starts with a formal exposition of the S^3VMs. Then Section 3 presents the continuation approach and its instantiation for S^3VMs. We give experimental support in Section 4. While the focus of this paper is the improvement of S^3VM accuracy, we briefly compare it to graph-based semi-supervised learning and consider computational complexity (Section 5).

2. Semi-Supervised SVMs

We first introduce the cost function of S^3VM in the linear case, and then extend it to the non-linear case.

2.1. The Linear Case

We consider here the problem of binary classification. The training set consists of n labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $y_i = \pm 1$, and of m the unlabeled examples $\{\mathbf{x}_i\}_{i=n+1}^{n+m}$.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

Standard linear supervised SVMs (Cortes & Vapnik, 1995) output a decision function of the form $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$, by minimizing the following objective function

$$\frac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} \ell\left(y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right), \tag{1}$$

where $\ell(t) = \max(0, 1-t)^p$ is a loss function penalizing the training errors either linearly ($p = 1$) or quadratically ($p = 2$), and C is a trade-off constant. In the rest of the paper, we consider $p = 1$. The corresponding loss function is depicted in Figure 1(a).

This objective function is usually optimized in the dual, but can also be optimized directly in the primal, e.g. by gradient descent (Keerthi & DeCoste, 2005; Chapelle, 2006; Chapelle & Zien, 2005).

In the semi-supervised case, an additional term is added to the objective function that drives the outputs $\mathbf{w}^\top \mathbf{x}_i + b$ of the unlabeled points $\mathbf{x}_i$ away from 0 (thereby implementing the cluster assumption):

$$\begin{aligned} L(\mathbf{w}) \quad = \quad & \tfrac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} \ell\left(y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right) \\ & + C^* \sum_{i=1+n}^{n+m} \ell\left(|\mathbf{w}^\top \mathbf{x}_i + b|\right). \end{aligned} \tag{2}$$

The main problem is that this additional term in the objective function (2) is *non-convex* and gives rise to local minima (see Figure 1(b)).

2.2. Balancing Constraint

In addition to the local minima, the objective function (2) has another problem: it tends to give unbalanced solutions which classify all the unlabeled points in the same class. In (Joachims, 1999) it was proposed to add the additional constraint that the class ratio on the unlabeled set should be the same as on the labeled set. However, if the true class ratio is not well estimated from the labeled set, this constraint can be harmful. We thus adopt the following more flexible constraint from (Chapelle & Zien, 2005)

$$\frac{1}{m} \sum_{i=n+1}^{n+m} \mathbf{w}^\top \mathbf{x}_i + b = \frac{1}{n} \sum_{i=1}^{n} y_i \ . \tag{3}$$

This is in analogy to the treatment of the min-cut problem in spectral clustering, which is usually replaced by the normalized cut to enforce balanced solutions (Joachims, 2003).

One easy way to enforce the constraint (3) is to translate all the points such that the mean of the unlabeled points is the origin, $\sum_{i=n+1}^{n+m} \mathbf{x}_i = 0$. Then, by fixing $b = \frac{1}{n}\sum_{i=1}^{n} y_i$, we can have an unconstrained optimization on $\mathbf{w}$.

2.3. The Non-Linear Case

Support Vector Machines are often used to find non-linear decision boundaries by applying the "kernel trick": the data are first mapped in a high dimensional Hilbert space $\mathcal{H}$ through a mapping $\Phi : \mathcal{X} \mapsto \mathcal{H}$, and then a linear decision boundary is constructed in that space. The mapping Φ is never explicitly computed, but only implicitly through the use of a kernel function k such that $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^\top \Phi(\mathbf{y})$. We will use K to denote the kernel matrix, which is the $n + m$ square matrix of elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Instead of expressing the optimization of (2) in terms of dot products, we will perform it directly in feature space. The idea why this is feasible is that even in the case of a high dimensional feature space $\mathcal{H}$, a training set $(\mathbf{x}_1, \ldots, \mathbf{x}_q)$ of size q (for S^{3}VM, $q = n + m$), when mapped to this feature space, spans a vectorial subspace $E \subset \mathcal{H}$ whose dimension is at most q. By choosing a basis for E and expressing the coordinates of all the points in that basis, we can then directly work in E. The representer theorem (Kimeldorf & Wahba, 1971) also ensures that the optimal solution is in E.

Let $(\mathbf{v}_1, \ldots, \mathbf{v}_q) \in E^q$ be an orthonormal basis of E,[1] where $\mathbf{v}_p$ is expressed as

$$\mathbf{v}_p = \sum_{i=1}^{q} A_{ip} \Phi(\mathbf{x}_i) \ . \tag{4}$$

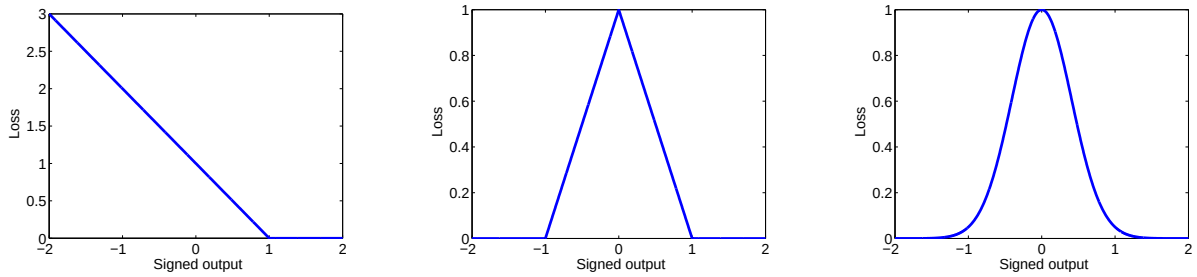
The orthonormality of the basis thus translates into the requirement $A^\top K A = I$, which is equivalent to $K = (AA^\top)^{-1}$. Because of rotations, there is an infinite number of matrices A satisfying this condition. One of them can be found by inverting the Cholesky decomposition of K. Another possibility is to perform the eigendecomposition of K as $K = U\Lambda U^\top$ and set $A = U\Lambda^{-1/2}$. This letter case corresponds to the *kernel PCA map* introduced in (Schölkopf & Smola, 2002, section 14.2).

Now we can use the following mapping $\psi : \mathcal{X} \mapsto \mathbb{R}^q$:

$$\psi_p(\mathbf{x}) = \Phi(\mathbf{x})^\top \mathbf{v}_p = \sum_{i=1}^{n} A_{ip} k(\mathbf{x}, \mathbf{x}_i), \quad 1 \le p \le q \ .$$

It can be easily checked that $\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$. Thus applying the linear approach presented in the

[1] We suppose that $dim(E) = q$, i.e. that the points are linearly independent in feature space or that K has full rank. The same analysis can be followed when $dim(E) < q$.



(a) (b) (c)

Figure 1. Losses in the S^{3}VM objective function (2): (a) hinge loss for the labeled points, i.e., $\ell(t) = \max(0, 1-t)$, (b) symmetric hinge loss for the unlabeled points, i.e., $\ell(|t|) = \max(0, 1-|t|)$ and (c) differentiable surrogate for (b), $\tilde{\ell}(t) = \exp(-3t^2)$.

previous section to this representation is equivalent to solving the non-linear problem.

3. The Continuation Approach

The *continuation* method belongs to the field of global optimization techniques (Wu, 1996). The idea is similar to deterministic annealing: a smoothed version of the objective function is first minimized. With enough smoothing, the optimization will be convex, and the global minimum can be found. Then the smoothing decreases and the new minimum is computed, where the solution found in the previous step serves as a starting point. The algorithm iterates until there is no smoothing (cf Algorithm 1 and Figure 2).

Algorithm 1 Continuation method

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_{γ_i}.
end for

Note that in general, other smoothing functions than the Gaussian can be used.

3.1. Smoothing of the S^{3}VM Objective Function

We now present the results of the convolution of the objective function (2) with a Gaussian $(\pi\gamma)^{-d/2} \exp(-t^2/\gamma)$ of variance $\gamma/2$. The convolution is done only on $\mathbf{w}$ and not on b, since the latter is fixed (see Section 2.2). Finally, since it is easy to convolve two Gaussians together, we decided to replace the loss of an unlabeled point, $\max(0, 1-|t|)$, by

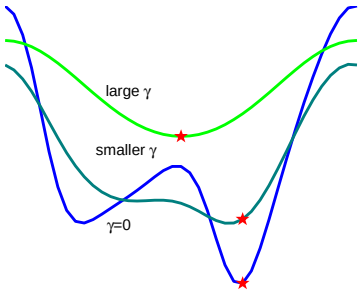


Figure 2. Illustration of the continuation method: the original objective function in blue has two local minima. By smoothing it, we find one global minimum (the red star on the green curve). And by reducing the smoothing, the minimum moves toward the global minimum of the original function (cf the blue-green curve).

$\exp(-st^2)$ with $s = 3$ as in (Chapelle & Zien, 2005), cf Figure 1(c).

The calculations are lengthy but straightforward, and we just present the final results. The convolution of the loss associated with one training point $\mathbf{x}_i$ turns out to be a one-dimensional convolution, since the function $\mathbf{w}^\top \mathbf{x}_i$ is constant on the subspace orthogonal to $\mathbf{x}_i$.

The convolved loss is

$$L_\gamma(\mathbf{w}) := \frac{1}{(\pi\gamma)^{\frac{d}{2}}} \int L(\mathbf{w}-\mathbf{t}) \exp(-\|\mathbf{t}\|^2/\gamma) d\mathbf{t} \quad (5)$$

$$= \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{2}\gamma d \quad (6)$$

$$+ C \sum_{i=1}^{n} \frac{\gamma\|\mathbf{x}_i\|}{\sqrt{2}} \left[\frac{\exp(-e_i^2)}{\sqrt{\pi}} - e_i \mathrm{erfc}(e_i) \right] \quad (7)$$

$$+ C^* \sum_{i=n+1}^{n+m} \frac{1}{\sqrt{a_i}} \exp\left(\frac{-s(\mathbf{w}^\top\mathbf{x}_i + b)^2}{a_i} \right) \quad (8)$$

$$\text{with} \quad \begin{cases} a_i = 1 + 2\gamma s \|\mathbf{x}_i\|^2 \\ e_i = \frac{y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1}{\sqrt{2\gamma}\|\mathbf{x}_i\|} \end{cases},$$

where $\operatorname{erfc}(\cdot)$ denotes the complementary error function, $\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, dt$.

The gradient of $L_\gamma(\cdot)$ at $\mathbf{w}$ is

$$\nabla L_\gamma(\mathbf{w}) = \mathbf{w}^\top - \frac{C}{2} \sum_{i=1}^{n} \operatorname{erfc}(e_i) y_i \mathbf{x}_i^\top - C^* \sum_{i=n+1}^{n+m} \frac{2s(\mathbf{w}^\top \mathbf{x}_i + b)}{a_i^{3/2}} \exp\left(\frac{-s(\mathbf{w}^\top \mathbf{x}_i + b)^2}{a_i}\right) \mathbf{x}_i^\top .$$

3.2. Choice of the Smoothing Sequence

We decide to choose γ_0 large enough such that L_{γ_0} is convex and thus easy to optimize, but not too large to avoid wasting time in this non-interesting regime. More precisely, we compute a lower bound on the smallest eigenvalue of the Hessian and set γ_0 such that this lower bound is 0.

The Hessian of (7), the term of L_γ corresponding to the labeled loss, is positive definite (since the convolution of a convex function with a positive function is convex). To control the term (8), we consider any unlabeled point $\mathbf{x}_i$, $n+1 \le i \le n+m$, and lower bound the associated term of the Hessian,

$$-\frac{2sC^*}{a_i^{3/2}} \underbrace{\left(1 - \frac{s}{a_i}\left(\mathbf{w}^\top \mathbf{x}_i - b\right)^2\right)}_{\le 1} \underbrace{e^{-\frac{s}{a_i}\left(\mathbf{w}^\top \mathbf{x}_i - b\right)^2}}_{\le 1} \mathbf{x}_i \mathbf{x}_i^\top \succ -\frac{C^*}{\sqrt{2s}\gamma^{3/2}} \frac{\mathbf{x}_i \mathbf{x}_i^\top}{\|\mathbf{x}_i\|^3} , \qquad (9)$$

using $\sqrt{a_i} > \sqrt{2\gamma s}\|\mathbf{x}_i\|$. We want the sum of all lower bound terms plus the Hessian for the regularizer (6) (which is just the identity matrix) to be positive definite. This is the case for $\gamma \ge \frac{(C^* \lambda_{\max})^{2/3}}{(2s)^{1/3}} =: \gamma_0$, where $\lambda_{\max}$ is the maximum eigenvalue of the matrix $\sum_{i=n+1}^{n+m} \frac{\mathbf{x}_i \mathbf{x}_i^\top}{\|\mathbf{x}_i\|^3}$.

The final value of γ is chosen such that L_γ and L_0 (the original objective function) are close. Again, we look at the unlabeled part of the objective function (8). When $\gamma \to 0$, $a_i \to 1$ and, as expected, (8) converges to the original loss. So we choose the final value of γ such that $a_i \approx 1$, or $2\gamma s\|\mathbf{x}_i\|^2 \ll 1$. In practice, we fix it to $\frac{1}{10} \frac{1}{2s \max_i \|\mathbf{x}_i\|^2} =: \gamma_{\text{end}}$.

Finally, we decide to take eleven steps on a geometric scale between γ_0 and γ_{end}, yielding the sequence

$$\gamma_i = \left(\frac{\gamma_{\text{end}}}{\gamma_0}\right)^{\frac{i}{10}} \gamma_0, \quad 0 \le i \le 10 \ .$$

3.3. Link with Kernel Feature Extraction

By construction of γ_0, for a value of $\gamma = \tilde{\gamma}_0$ which is smaller than γ_0, the smallest eigenvalue of the Hessian of (6) will be zero. The corresponding eigenvector is related to the leading eigenvector $\mathbf{v}^*$ of the matrix $M := \sum_{i=n+1}^{n+m} \frac{\mathbf{x}_i \mathbf{x}_i^\top}{\|\mathbf{x}_i\|^3}$ (it would be the same if (9) were an equality). Thus for $\gamma = \tilde{\gamma}_0$, (6)+(8) do not penalize $\mathbf{w}$ in the direction of $\mathbf{v}^*$, which means that the algorithms encourages the use of this direction in the task of minimizing (7).

Ignoring the denominator in M and recalling that the points are in feature space, $\mathbf{v}^*$ is the *principal kernel component* (Schölkopf & Smola, 2002, chapter 14). This renders the behavior observed above very satisfying, as it has been argued that the first principal kernel components capture the cluster structure in the data (Ng et al., 2001; Schölkopf & Smola, 2002; Chapelle et al., 2002). So the term in the objective function corresponding to the unlabeled points can be seen as a data dependent regularization term, encouraging solutions that are smooth with respect to the cluster structure contained in the unlabeled data.

4. Experiments

To validate the performance of our method, we conduct numerical experiments. First we investigate the relationship between objective function and test error. Then we compare the described continuation method, which we call cS^{3}VM, to the ∇S^{3}VM described in (Chapelle & Zien, 2005) and the S^{3}VM implementation in SVM^{light} (Joachims, 1999), here denoted by S^{3}VM^{light}. For simplicity we start with the hyperparameters being fixed to reasonable values, and afterwards verify the robustness of the findings with respect to changes of the hyperparameters.

4.1. Experimental Setup

We conduct experiments on three datasets. As a classical semi-supervised learning task, we use the two-class text dataset `Text` from (Chapelle & Zien, 2005) where the goal is to classify the newsgroups `mac` and `mswindows` from the `Newsgroup20` dataset. However, S^{3}VMs are known to perform well on text data (Joachims, 1999; Chapelle & Zien, 2005). Analyzing the experimental results from (Chapelle & Zien, 2005), it appears that ∇S^{3}VM and S^{3}VM^{light} do not perform well for the recognition of handwritten digits and objects in images. As more challenging datasets, we thus use `Uspst` and `Coil` from (Chapelle & Zien, 2005). `Uspst` contains 2007 images of hand-written digits (the test set of the original USPS data set), hence is consists

of ten classes. `Coil` contains 1440 images of twenty different objects, each of which corresponds to a class.

For each of these datasets, we use the same ten splits into labeled and unlabeled points as (Chapelle & Zien, 2005). For the multiclass datasets, we focus on a given pair of classes by removing the other classes from both the labeled set and the unlabeled set. For `Uspst`, this results in 450 experiments: 45 one-vs-one tasks with ten repetitions (splits) for each. For `Coil`, we end up with 1900 experiments.

We note that all three datasets seem to comply to some extent with the cluster assumption: S^3VMs can reliably outperform SVMs that use just the labeled data. The behavior of S^3VMs on a dataset (`g10n`) that violates this assumption has been investigated by (Chapelle & Zien, 2005); there, using the unlabeled data does not yield an accuracy improvement.

4.2. Objective versus Test Error

For this first investigation we pick `Uspst`, the most challenging of our datasets. We choose the Gaussian RBF kernel, which is a good general purpose kernel, and known to work well for USPS. We fix the kernel width (standard deviation) σ to 8. Then we fix the regularization parameter C to 100: this a high value, well suited for the low noise present in that dataset. Finally, we set C^* in (2) to C.

To verify that lower values of the objective function indeed correspond to lower test errors, we compare the local minima found by cS^3VM and ∇S^3VM. As Figure 3 shows, there is indeed a clear correlation: when the objective function is higher (stuck in a worse local minimum), the test error is frequently higher as well. This justifies putting effort into the optimization.

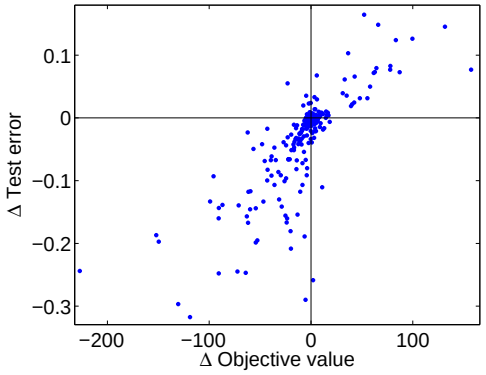


Figure 3. Difference of test error versus difference in objective function value. Each dot corresponds to the two local minima found with with cS^3VM and ∇S^3VM for one of the 450 learning tasks of `Uspst`.

4.3. Comparing S^3VM Optimization Methods

Next, we compare the prediction errors on the 450 experiments described above by scatter plots shown in Figure 4. cS^3VM is better than each of the two competitors in the majority of cases. It is also apparent that the average test error of cS^3VM is lower than those of the other two methods. These averages are provided in Table 1 for for all three data sets. As a reference point, we include the results of a plain (supervised) SVM.

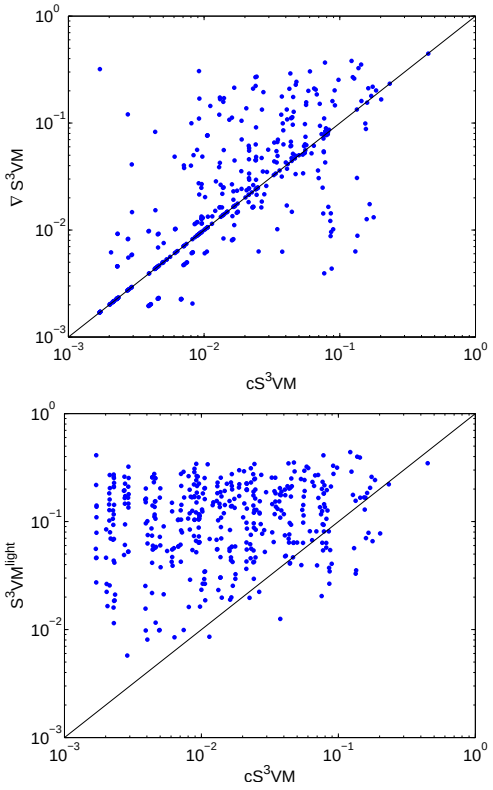


Figure 4. Comparison of test errors achieved in the 450 `Uspst` experiments. Top, cS^3VM vs ∇S^3VM; bottom, cS^3VM vs S^3VM^{light}.

Table 1. Test errors on different datasets averaged over splits (and, for `Uspst` and `Coil`, over pairs of classes) with fixed hyperparameters. `Uspst`, Gaussian RBF kernel with $\sigma = 8$; `Coil`, Gaussian RBF kernel with $\sigma = 1000$; `Text`, linear kernel. In all cases, $C^* = C = 100$ (which is close to hard margin).

dataset	SVM	cS^3VM	∇S^3VM	S^3VM^{light}
`Uspst`	9.2%	2.9%	4.6%	13.0%
`Coil`	6.80%	2.07%	2.57%	2.46%
`Text`	18.8%	5.3%	6.2%	7.4%

For `Coil` the difference between cS^3VM and ∇S^3VM is significant at 95% confidence, but not between cS^3VM

and S^{3}VM^{light}. This dataset is much easier than the `Uspst`, and all the S^{3}VM implementations achieve a good test error, making it more difficult to compare them. Note that for `Coil` S^{3}VM^{light} has a slight advantage because of its hard balancing constraint: here the fraction of positive points in the unlabeled set exactly equals that in the labeled data (50%).

Finally, it should be verified that the improvement over ∇S^{3}VM is not due to a too coarse annealing sequence. We repeat the `Uspst` experiment with a long and slow annealing sequence for ∇S^{3}VM: start C^* at very small value (10^{-6} its final value) and took 100 steps. This did not result in an improvement (test error of 4.9%). As for S^{3}VM^{light}, it has a very conservative annealing schedule built in.

4.4. Influence of the Hyperparameters

Since all three compared algorithms minimize (almost) the same objective function,[2] the fixed hyperparameters should be (almost) equally good or bad for all of them. However, there is still a small possibility that some regime of hyperparameter settings is more suitable for some algorithm than for another.

To be sure that the hyperparameters do not unduly influence the conclusions drawn above, we try different values of the hyperparameters. All the results reported in this section are again averages over splits and (except for `Text`) pairs of classes.

First we vary the kernel width σ on an exponential scale while keeping C (and C^*). Figure 5 shows that cS^{3}VM is consistently better than the other methods (exemplified for `Uspst`).

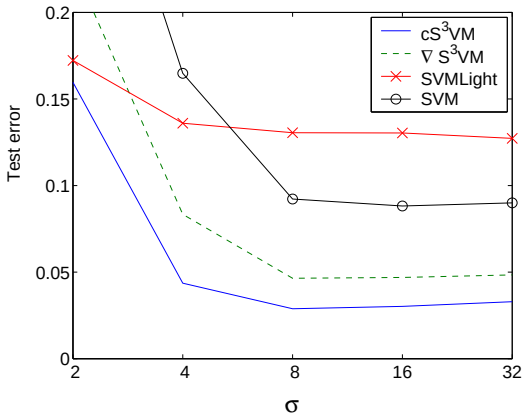


Figure 5. Test errors on `Uspst` of the different implementations of S^{3}VM with various values of σ (for $C = C^* = 100$).

[2]The objective function of cS^{3}VM and ∇S^{3}VM is the same. S^{3}VM^{light} uses the loss (b) instead of (c) in Figure 1 and has a different balancing constraint.

We also try a smaller values of C, namely $C = 10$ for `Uspst` and $C = 1$ for `Text`. Again, cS^{3}VM turns out best in all cases (cf Table 2).

Table 2. Test errors with varying C (with $C^* = C$). Top, `Uspst` with $\sigma = 8$; bottom, `Text`.

C	SVM	cS^{3}VM	∇S^{3}VM	S^{3}VM^{light}
10	9.7%	3.0%	4.9%	12.8%
100	9.2%	2.9%	4.6%	13.0%

C	SVM	cS^{3}VM	∇S^{3}VM	S^{3}VM^{light}
1	21.0%	5.0%	6.1%	7.4%
100	18.8%	5.3%	6.2%	7.4%

Finally, we vary C^* (not for S^{3}VM^{light}, since this is not offered by the actual implementation); see Table 3. All results indicate that the superior accuracy of cS^{3}VM is stable with respect to the hyperparameter settings.

Table 3. Test errors on `Uspst` with $\sigma = 8$ and $C = 100$.

	cS^{3}VM	∇S^{3}VM
$C^* = 10$	2.8%	5.0%
$C^* = 100$	2.9%	4.6%

5. Further Considerations

5.1. Relation with Graph-based Approaches

As the last section demonstrates, cS^{3}VM is relatively good and novel method to train S^{3}VMs. But how good is the S^{3}VM itself when compared to other semi-supervised approaches? To answer this question, we compare S^{3}VMs to a modern graph-based approach, namely LapSVM (Sindhwani et al., 2005). In this approach, a supervised SVM is trained on the labeled points, but in addition to the "normal" kernel, the graph Laplacian derived from both labeled and unlabeled points is used for regularization. In their setting, the two resulting regularizers are added to the data fitting term after being weighted with γ_A for the normal kernel, and γ_I for the Laplacian kernel. The standard supervised SVM is recovered as a special case with $\gamma_I = 0$; then γ_A corresponds to $1/C$ in the SVM.

For `Text`, being two-class, we can immediately compare our results to those in (Sindhwani et al., 2005): there, LapSVM achieved 10.4% – significantly worse than any S^{3}VM method (Table 2, bottom). However, it is known that graph-based methods work well on data with manifold structure, but less so on text.

As a more challenging test for the S^{3}VM, we compare

the performance on `Uspst`. We set up the graph Laplacian kernel with the same hyperparameters (σ, degree of the Laplacian, and number of nearest neighbors) as chosen in (Sindhwani et al., 2005). We only change γ_A to $1/C = 0.01$ (10^{-6} in (Sindhwani et al., 2005)); however, both correspond to an almost hard margin. After computing both Gaussian RBF and the graph Laplacian kernel,[3] we vary γ_I and train both an SVM and the cS^{3}VM on top of them. The results are shown in Figure 6.

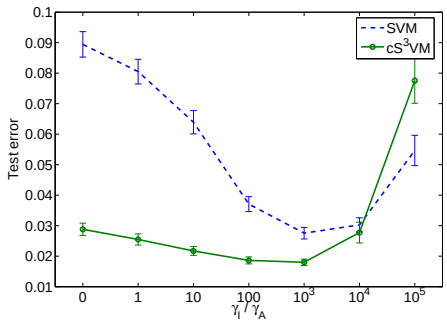


Figure 6. SVM and cS^{3}VM trained with additional graph-based regularization (of strength γ_I).

When comparing cS^{3}VM (with just the normal kernel) and LapSVM (with best γ_I), we have 2.9% vs 2.8% (cf also Table 1). This is very encouraging, since graph based methods were known to be much better than S^{3}VM on manifold datasets. Second, hybrid methods — combining best of both worlds — can further improve the results (here, to 1.8%).

5.2. Computational Considerations

The runtime of the cS^{3}VM algorithm as described above is cubic in the number of total data points, i.e. $\mathcal{O}\left((n+m)^3\right)$. First, the eigendecomposition takes cubic time; then, the continuation method requires a constant number (eleven) of gradient descent minimizations in the kernel PCA space, each of which is again of complexity $\mathcal{O}\left((n+m)^3\right)$. Given that semi-supervised learning is designed for problems with plenty of unlabeled data, cubic time complexity seems impractical.

However, if desired, the computation can be sped up considerably. First, in many cases the data lie close to a lower-dimensional subspace in $\mathcal{H}$. In this case, the leading p eigenvectors of K can provide a sufficient basis for representing the data. These eigenvectors can be approximated by applying the kernel PCA to a subset of $q > p$ points. The total computational cost of the algorithm is then reduced to $\mathcal{O}(q^3 + pq(n+m))$.

To demonstrate this, we classify the digits "5" versus "8" from the MNIST dataset. We randomly draw 10 labeled points of each class and 5000 unlabeled points. The hyperparameters are set to $\sigma = 5 * 256$ and $C = C^* = 100$. Then we vary $p = q$ over $5000 * \{1/64, 1/32, \ldots, 1/2, 1\}$. Running time and results are shown in Figure 7. The training time can be 2 orders of magnitude faster without loosing too much accuracy.

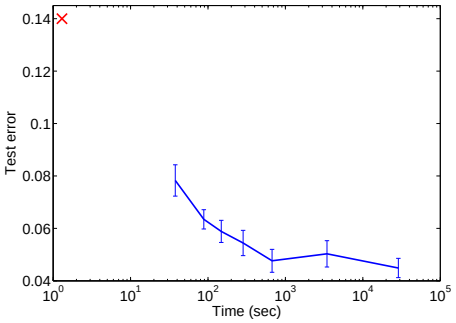


Figure 7. Time vs test error for various values of $p = q$. The bars represent the standard error of the mean. The red cross is an SVM trained on the 20 labeled points.

5.3. One-vs-the Rest

The results reported above for `Uspst` and `Coil` are in a 1-vs-1 setting. For 1-vs-rest, the performance of cS^{3}VM is disappointing (cf Table 4).

Table 4. Performances of different S^{3}VM implementations on the `Uspst` dataset in a 1-vs-rest setting or 5-vs-5 setting (digits 0 to 4 against 5 to 9). The hyperparameters are the same as in Section 4.2.

	SVM	cS^{3}VM	∇S^{3}VM	S^{3}VM^{light}
1-vs-rest	23.4%	34.2%	21.0%	26.7%
5-vs-5	17.8%	19.6%	16.7%	16.9%

One possible explanation why cS^{3}VM does not perform well in 1-vs-rest and 5-vs-5 is the following: unlike 1-vs-1, each class consists several clusters. Our continuation method might have difficulties with this situation, which presents several possibilities for undesirable low density separations. The analysis in section 3.3 suggests that cS^{3}VM favors the split corresponding to the largest principal component.

Also, note that none of the S^{3}VMs methods performs well: they hardly improve over a standard SVM, and a graph-based approach can achieve much better results. Indeed, a 12.7% error on `Uspst` in a 1-vs-rest

[3] using the code from `http://www.cs.uchicago.edu/~vikass/research.html`

setting has been reported in (Sindhwani et al., 2005).[4] This shows that S^{3}VMs are not yet a state-of-the-art method for multiclass problems.

6. Conclusion and Discussion

In (Chapelle & Zien, 2005) it was conjectured that the objective function of S^{3}VM is well designed, since it implements the cluster assumption. Our results (e.g. Figure 3) confirm that a better optimization of this function indeed tends to result in a higher predictive accuracy. Consequently care should be taken not to get trapped in bad local minima.

We use a global optimization technique for non-convex problems to handle this difficulty. Evaluated in a one-vs-one setting, the results compare favorably against two competing approaches suggested in the literature. However, 1-vs-1 is not sufficient for real-world multiclass semi-supervised learning, since the unlabeled data cannot be restricted to the two classes under consideration. Future work should therefore extend to the one-vs-rest scheme or to a true multiclass method.

References

Astorino, A., & Fuduli, A. (2005). *Nonsmooh optimization techniques for semi-supervised classification* (Technical Report). U. Pisa, Dipartimento di Matematica. `www.dm.unipi.it/mat_dia_med/PAPER_Astorino.pdf`.

Bennett, K., & Demiriz, A. (1998). Semi-supervised support vector machines. *Advances in Neural Information processing systems 12*.

Chapelle, O. (2006). *Training a support vector machine in the primal* (Technical Report 147). Max Planck Institute. `www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal`.

Chapelle, O., Weston, J., & Schölkopf, B. (2002). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems 15*.

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

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive SVMs. *Journal of Machine Learning Research*. Submitted, `www.kyb.tuebingen.mpg.de/bs/people/fabee/universvm.html`.

[4] with different hyperparameters as in Table 4.

Cortes, C., & Vapnik, V. (1995). Support vector network. *Machine learning*, *20*, 1–25.

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

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

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *International Conference on Machine Learning*.

Keerthi, S. S., & DeCoste, D. M. (2005). A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, *6*, 341–361.

Kimeldorf, G., & Wahba, G. (1971). Some results on tchebycheffian spline functions. *J. Math. Anal. Applic.*, *33*, 82–95.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Seeger, M. (2006). A taxonomy of semi-supervised learning methods. In O. Chapelle, B. Schölkopf and A. Zien (Eds.), *Semi-supervised lerning*. MIT Press.

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

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

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

Vapnik, V. N. (1998). *Statistical learning theory*. New York: John Wiley & Sons, Inc.

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