Large Scale Semi-supervised Linear SVMs

V. Sindhwani* and S.S. Keerthi †

March 29, 2006

Abstract

Large scale learning is often realistic only in a semi-supervised setting where a small set of labeled examples is available together with a large collection of unlabeled data. In many Information retrieval and data mining applications, linear methods are strongly preferred because of their ease of implementation, interpretability and empirical performance. In this work, we present a family of semi-supervised linear support vector classifiers that are designed to handle partially-labeled sparse datasets with possibly very large number of examples and features. At their core, our algorithms employ modified finite Newton techniques recently developed in [15]. Our contributions in this paper are as follows: (a) We provide an implementation of Linear Transductive SVM (TSVM) that is significantly more efficient and scalable than currently used dual techniques, for linear classification problems involving large, sparse datasets. (b) We propose a variant of TSVM that involves multiple switching of labels. Experimental results show that this variant provides an order of magnitude further improvement in training efficiency. (c) We present a new algorithm for semi-supervised learning based on a mean field annealing (MFA) approach. This algorithm alleviates the problem of local minimum in the TSVM optimization procedure while being computationally attractive. We conduct an empirical study on several document classification tasks that confirms the value of these approaches in providing scalable tools for semi-supervised learning in large scale settings. Finally, we also note various extensions and applications of our methods.

*Department of Computer Science, University of Chicago, 1100 E 58th Street, Chicago, IL 60637 (work done while visiting Yahoo! Research). EMAIL: *vikass@cs.uchicago.edu*

†Yahoo! Research, Yahoo! Search Marketing, Media Studios North, 3333 Empire Avenue, Bldg. 4 Burbank, CA 91504. EMAIL: *selvarak@yahoo-inc.com*

1 Introduction

Consider the following situation: In a single web-crawl, search engines like Yahoo! and Google index billions of webpages. Only a very small fraction of these web-pages can possibly be hand-labeled by human editorial teams and assembled into topic directories. The remaining web-pages form a massive collection of unlabeled documents. Each document is a sparse collection of words and hyperlinks, highly structured by grammatical constraints, in a very high dimensional linguistic space. It is clear that the development of computational tools to organize large amounts of high-dimensional data with very little human supervision is central to the goal of effective information management in many applications.

Despite its natural and pervasive need, solutions to the problem of utilizing unlabeled data with labeled examples have only recently emerged in machine learning literature. Whereas the abundance of unlabeled data is frequently acknowledged as a motivation in most papers, the true potential of semi-supervised learning in large scale settings is yet to be systematically explored. This appears to be partly due to the lack of scalable tools to handle large volumes of data, and partly due to the common research practice of demonstrating semi-supervised learning on small datasets with extremely few labels.

Our motivation for this work is two-fold: (a) We seek to develop classification algorithms for applications involving large, possibly very high-dimensional but sparse, partially labeled datasets. Being the rule rather than the exception, such datasets arise routinely in numerous applications e.g in document classification, bioinformatics and scientific computing. (b) We seek to employ these techniques for studying semi-supervised learning in realistic large scale settings. With these tools, one can investigate how the geometric structure of data in a high dimensional input space interacts with the distribution of labels and the hypothesis space for learning on real-world problems.

In this paper, we propose extensions of linear Support Vector Machine (SVM) for semi-supervised classification. Linear techniques are often the method of choice in many applications due to their simplicity and interpretability. When data appears in a rich high-dimensional representation, linear functions often provide a sufficiently complex hypothesis space for learning high-quality classifiers. This has been established e.g for document classification with Linear SVMs in numerous studies (see e.g [5]).

Our methods are based on transductive extensions of SVM first proposed in [19] and implemented with different variations in [10, 1, 9, 6]. The key idea is to bias the classification hyperplane to pass through a low data density region keeping points in each data cluster on the same side of the hyperplane while respecting labels. This algorithm uses an extended SVM objective function with a non-convex loss term over the unlabeled examples to implement the cluster assumption in semi-supervised learning[1]. This idea is of historical importance as one of the first concrete proposals for learning from unlabeled data; its popular implementation in [10] is considered state-of-the-art in text categorization, even in the face of increasing recent competition.

We highlight the contributions of this paper.

1. We outline an implementation for a variant of Transductive SVM [10] designed for linear semi-supervised classification on large, sparse datasets. As compared to currently used dual techniques (e.g in the SVM-Light implementation of TSVM), our method exploits data sparsity and linearity of the problem to provide superior scalability. Additionally, we propose a multiple switching heuristic that further improves TSVM training by an order of magnitude. These speed enhancements turn TSVM into a feasible tool for large scale applications.

2. We propose a new algorithm for semi-supervised SVMs utilizing well-established information theoretic ideas for global optimization using mean field methods. This algorithm generates a family of objective functions whose non-convexity is controlled by an annealing parameter. The global minimizer is tracked with respect to this parameter. This approach alleviates the problem of local minima in the TSVM optimization procedure which results in significantly better solutions on many problems, while also being computationally attractive.

3. We conduct an experimental study on many document classification tasks with several thousands of examples and features. This study clearly shows the utility of our tools for large scale problems.

The modified finite Newton algorithm (abbreviated L_2-SVM-MFN) of Keerthi and Decoste [15] for fast training of linear SVMs is a key subroutine for our algorithms.

This paper is arranged as follows. In section 2 we describe a slightly modified version of L_2-SVM-MFN algorithm to suit the description of the semi-supervised methods (TSVM and MFA) in section 3. Experimental results are presented in section 4. In section 5, we discuss some extensions and applications of our methods.

2 Modified Finite Newton Linear L_2-SVM

The modified finite Newton L_2-SVM method [15] (L_2-SVM-MFN) is a recently developed training algorithm for Linear SVMs that is ideally suited

[1] The assumption that points in a cluster should have similar labels. The role of unlabeled data is to identify clusters and high density regions in the input space.

to sparse datasets with large number of examples and possibly large number of features. In a typical application like document classification, many training documents are collected and processed into a format that is convenient for mathematical manipulations. For example, each document may be represented as a collection of d features associated with a vocabulary of d words. Theses feature may simply indicate the presence or absence of a word (binary features), or measure the frequency of a word suitably normalized by its importance (TFIDF features) (see e.g [18] for more details and other representations). Even though the vocabulary might be large, only a very small number of words appear in any document relative to the vocabulary size. Thus, each document is sparsely represented as a bag of words. A label is then manually assigned to each document identifying a particular category to which it belongs (e.g "commercial" or not). The task of a classification algorithm (e.g SVM) is produce a classifier that can reliably identify the category of new documents based on information extracted from training documents .

Given a binary classification problem with l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ where the input patterns $x_i \in \mathbb{R}^d$ (e.g a document) and the labels $y_i \in \{+1, -1\}$, L_2-SVM-MFN provides an efficient primal solution to the following SVM optimization problem:

$$w^\star = \operatorname*{argmin}_{w \in \mathbb{R}^d} \quad \frac{1}{2} \sum_{i=1}^{l} \max \left[0, 1 - y_i \ (w^T x_i)\right]^2 + \frac{\lambda}{2} \|w\|^2 \tag{1}$$

Here, λ is a real-valued regularization parameter and $\operatorname{sign}({w^\star}^T x)$ is the final classifier.

This objective function differs from the standard SVM problem in some respects. First, instead of using the hinge loss as the data fitting term, the square of the hinge loss (or the so-called quadratic soft margin loss function) is used. This makes the objective function continuously differentiable, allowing easier applicability of gradient techniques. Secondly, the bias term ("b") is also regularized. In the problem formulation of Eqn. 1, it is implicitly assumed that an additional component in the weight vector and a constant feature in the example vectors have been added to indirectly incorporate the bias. This formulation combines the simplicity of a least squares aspect with algorithmic advantages associated with SVMs. We also note that all the discussion in this paper can be applied to other loss functions such as Huber's Loss and rounded Hinge loss using the modifications outlined in [15].

We will consider a version of L_2-SVM-MFN where a weighted quadratic soft margin loss function is used.

$$w^\star = \operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) = \operatorname*{argmin}_{w \in \mathbb{R}^d} \quad \frac{1}{2} \sum_{i \in \jmath(w)} c_i \ d_i^2(w) + \frac{\lambda}{2} \|w\|^2 \tag{2}$$

Here we exactly rewrite Eqn. 1 in terms of a partial summation of $d_i(w) = w^T x_i - y_i$ over an index set $\jmath(w) = \{i : y_i\ (w^T x_i) < 1\}$. Additionally, the loss associated with the i^{th} example has a cost c_i. $f(w)$ refers to the objective function being minimized, evaluated at a candidate solution w. Note that if the index set $\jmath(w)$ were independent of w and ran over all data points, this would simply be the objective function for weighted linear regularized least squares (RLS).

Following [15], we observe that f is a strictly convex, piecewise quadratic, continuously differentiable function having a unique minimizer. The gradient of f at w is given by:

$$\nabla\ f(w) = \lambda\ w + \sum_{i \in \jmath(w)} c_i\ d_i(w)\ x_i = \lambda\ w + X^T_{\jmath(w)} C_{\jmath(w)} \left[X_{\jmath(w)} w - Y_{\jmath(w)}\right]$$

where $X_{\jmath(w)}$ is a matrix whose rows are the feature vectors of training points corresponding to the index set $\jmath(w)$, $Y_{\jmath(w)}$ is a column vector containing labels for these points, and $C_{\jmath(w)}$ is a diagonal matrix that contains the costs c_i for these points along its diagonal.

L_2-SVM-MFN is a primal algorithm that uses the Newton's Method for unconstrained minimization of a convex function. The classical Newton's method is based on a second order approximation of the objective function, and involves updates of the following kind:

$$w^{(k+1)} = w^{(k)} + \delta^{(k)}\ n^{(k)} \tag{3}$$

where the step size $\delta^k \in \mathbb{R}$, and the Newton direction $n^k \in \mathbb{R}^d$ is given by:

$$n^{(k)} = -\left[\nabla^2\ f\left(w^{(k)}\right)\right]^{-1} \nabla\ f\left(w^{(k)}\right)$$

Here, $\nabla\ f\left(w^{(k)}\right)$ is the gradient vector and $\nabla^2\ f\left(w^{(k)}\right)$ is the Hessian matrix of f at $w^{(k)}$. However, the Hessian does not exist everywhere, since f is not twice differentiable at those weight vectors w where $w^T x_i = y_i$ for some index i.[2] For this reason, a finite Newton method designed by Mangasarian [12] works around this issue through a generalized definition of the Hessian matrix. On the other hand, the modified finite Newton procedure [15] proceeds as follows. The step $\bar{w}^{(k)} = w^{(k)} + n^{(k)}$ in the Newton direction can be seen to be given by solving the following linear system associated with a weighted linear regularized least squares problem over the data subset defined by the indices $\jmath(w^{(k)})$:

$$\left[\lambda I + X^T_{\jmath(w^{(k)})} C_{\jmath(w^{(k)})} X_{\jmath(w^{(k)})}\right] \bar{w}^{(k)} = X^T_{\jmath(w^{(k)})} C_{\jmath(w^{(k)})} Y_{\jmath(w^{(k)})} \tag{4}$$

[2] In the neighborhood of such a w, the index i leaves or enters $\jmath(w)$. However, at $w, d_i(w) = 0$. So f is continuously differentiable inspite of these index jumps.

where I is the identity matrix. Once $\bar{w}^{(k)}$ is obtained, $w^{(k+1)}$ is obtained from Eqn. 3 by setting $w^{(k+1)} = w^{(k)} + \delta^k(\bar{w}^{(k)} - w^{(k)})$ after performing an exact line search for δ^k, i.e by exactly solving a one-dimensional minimization problem:

$$\delta^{(k)} = \operatorname*{argmin}_{\delta \geq 0} f\left(w^{(k)} + \delta(\bar{w}^{(k)} - w^{(k)})\right)$$

The modified finite Newton procedure has the property of finite convergence to the optimal solution. The key features that bring scalability and numerical robustness to L_2-SVM-MFN are: (a) Solving the regularized least squares system of Eqn. 4 by a numerically well-behaved Conjugate Gradient scheme [8] referred to as CGLS, which is designed for large, sparse data matrices X. The benefit of the least squares aspect of the loss function comes in here to provide access to a powerful set of tools in numerical computation. (b) Due to the one-sided nature of margin loss functions, these systems are required to be solved over only restricted index sets $\jmath(w)$ which can be much smaller than the whole dataset. This also allows additional heuristics to be developed such as terminating CGLS early when working with a crude starting guess like 0, and allowing the following line search step to yield a point where the index set $\jmath(w)$ is small. Subsequent optimization steps then work on smaller subsets of the data[3].

We now outline the details of the CGLS and Line search procedures.

2.1 CGLS

The CGLS procedure solves large, sparse, weighted regularized least squares problems of the following form:

$$\left[\lambda I + X^T C X\right] \; \beta = X^T C Y \tag{5}$$

The key computational issue here is to avoid the construction of the large and dense matrix X^TCX, and work only with the sparse matrix X and the diagonal cost matrix (stored as a vector) C.

Starting with a guess solution, β_0, Conjugate Gradient performs iterations of the form:

$$\beta^{(j+1)} = \beta^{(j)} + \gamma^{(j)} p^{(j)}$$

where $p^{(j)}$ is a search direction and $\gamma^{(j)} \in \mathbb{R}$ gives the step in that direction. The residual vector (the difference vector between LHS and RHS of Eqn. 5 for a candidate β, which is also the gradient of the associated quadratic form evaluated at β) is therefore updated as:

$$r^{(j+1)} = X^T C Y - \left[\lambda I + X^T C X\right] \; \beta^{(j+1)} = X^T z^{(j+1)} - \lambda \beta^{(j+1)}$$

[3] An implementation would also include heuristics 1 and 2, and some exception handling steps as described in [15].

Here, we introduce the following intermediate vectors:

$$\begin{aligned} z^{(j+1)} &= C\left(Y - X\beta^{(j+1)}\right) = C\left(Y - \left[X\beta^{(j)} + \gamma^{(j)}Xp^{(j)}\right]\right) \\ &= z^{(j)} - \gamma^{(j)}Cq^{(j)} \\ \text{where } q^{(j)} &= Xp^{(j)} \end{aligned}$$

The optimal step-size $\gamma^{(j)}$ is given by:

$$\gamma^{(j)} = \frac{\|r^{(j)}\|^2}{p^{(j)T}(\gamma I + X^T CX)p^{(j)}} = \frac{\|r^{(j)}\|^2}{\lambda\|p^{(j)}\|^2 + q^{(j)T}Cq^{(j)}}$$

Finally, the search directions are updated as:

$$p^{(j+1)} = r^{(j+1)} + \omega^{(j)}p^{(j)} \qquad \text{where } \omega^{(j)} = \frac{\|r^{(j+1)}\|^2}{\|r^{(j)}\|^2}$$

It can be shown that these updates implicitly and incrementally construct a conjugate basis[4] $p^0, p^1 \ldots$ of $\mathbb{R}^d$ on the fly; each candidate solution β^j is optimal in the subspace defined by the current basis elements, converging to the desired solution in no more (and typically much fewer) than d steps.

The CGLS iterations are terminated when the norm of the gradient $r^{(j+1)}$ becomes small enough relative to the norm of the iterate $z^{(j+1)}$ or if the number of iterations exceed a certain maximum allowable number.

The CGLS iterations are listed in Table 2. The data matrix X is only involved in the computations through matrix vector multiplication for computing the iterates $q^{(j)}$ and $r^{(j)}$. This forms the dominant expense in each iteration (the product with C simply scales each element of a vector). If there are n_0 non-zero elements in the data matrix, this has $O(n_0)$ cost. As a subroutine of L_2-SVM-MFN, CGLS is typically called on a small subset of the full data set. The total cost of CGLS is $O(t_{cgls}n_0)$ where t_{cgls} is the number of iterations, which depends on the practical rank of X and is typically found to be very small relative to the dimensions of X (number of examples and features). The memory requirements are also minimal: only five vectors need to be maintained, including the outputs over the currently active set of data points. For more details on Conjugate gradient, see [2].

Finally, an important feature of CGLS is worth emphasizing. Suppose the solution β of a regularizer least squares problem is available, i.e the linear system in Eqn. 5 has been solved using CGLS. If there is a need to solve a perturbed linear system, it is greatly advantageous in many settings to start the CG iterations for the new system with β as the initial guess. This is often called *seeding*. If the starting residual is small, CGLS can converge much faster than with a guess of 0 vector. The utility of this feature

[4] A basis with mutually orthogonal elements in the sense $v^T\left[\lambda I + X^T CX\right]u = 0$ for any pair of elements u, v.

depends on the nature and degree of perturbation. In L_2-SVM-MFN, the candidate solution $w^{(k)}$ obtained after line search in iteration k is seeded for the CGLS computation of $\bar{w}^k$. Also, in tuning λ over a range of values, it is computationally valuable to seed the solution for a particular λ onto the next value. For the semi-supervised SVM implementations with L_2-SVM-MFN, we will seed solutions across linear systems with slightly perturbed label vectors, data matrices and costs.

2.2 Line Search

Given the vectors $w, \bar{w}$ in some iteration of L_2-SVM-MFN, the line search step requires us to solve:

$$\delta^\star = \underset{\delta \geq 0}{\operatorname{argmin}}\, \phi(\delta) = f\,(w_\delta)$$

where $w_\delta = w + \delta(\bar{w} - w)$.

The one-dimensional function $\phi(\delta)$ is the restriction of the objective function f on the ray from w onto $\bar{w}$. Hence, like f, $\phi(\delta)$ is also a continuously differentiable, strictly convex, piecewise quadratic function with a unique minimizer $\delta^\star$ given by $\phi'(\delta^\star) = 0$. Thus, one needs to find the root of the piecewise linear function

$$\phi'(\delta) = \lambda w_\delta^T(\bar{w} - w) + \sum_{i \in \jmath(w_\delta)} c_i d_i(w_\delta)(\bar{o}_i - o_i) \tag{6}$$

where $o = Xw, \bar{o} = X\bar{w}$.

The linear pieces of ϕ' are defined over those intervals where $\jmath(w_\delta)$ remains constant. Thus, the break points occur at a certain set of values δ_i where $w_{\delta_i}^T x_i = y_i$ for some data point indexed by i, i.e $\delta_i = \frac{y_i - o_i}{\bar{o}_i - o_i} = \frac{1 - y_i o_i}{y_i(\bar{o}_i - o_i)}$. Among these values, one needs to only consider those indices i where $\delta_i \geq 0$ i.e if $i \in \jmath(w)$ (then $y_i o_i < 1$), so $y_i(\bar{o}_i - o_i) > 0$ or if $i \notin \jmath(w)$ (then $y_i o_i > 1$), so $y_i(\bar{o}_i - o_i) < 0$. When δ is increased past a δ_i, in the former case the index i leaves $\jmath(w)$ and in the latter case it enters $\jmath(w)$. Reordering the indices so that δ_i are sorted in a non-decreasing order as $\delta_{j_1}, \delta_{j_2} \ldots$, the root is then easily checked in each interval $(\delta_{j_k}, \delta_{j_{k+1}})$, $k = 1, 2 \ldots$ by keeping track of the slope of the linear piece in that interval. The slope is constant for each interval and non-decreasing as the search progresses through these ordered intervals. The interval in which the slope becomes non-negative for the first time brackets the root. Defining the extension of the linear piece in the interval $(\delta_{j_k}, \delta_{j_{k+1}})$ as $\phi'_k(\delta) = \lambda w_\delta^T(\bar{w} - w) + \sum_{i \in \jmath(w_{\delta_{j_k}})} c_i d_i(w_\delta)(\bar{o}_i - o_i)$, the slope and the root computations are conveniently done by keeping track of $L = \phi'_k(0) = \lambda w^T(\bar{w} - w) + \sum_{i \in \jmath(w_{\delta_{j_k}})} c_i(o_i - y_i)(\bar{o}_i - o_i)$ and $R = \phi'_k(1) = \lambda \bar{w}^T(\bar{w} - w) + \sum_{i \in \jmath(w_{\delta_{j_k}})} c_i(\bar{o}_i - y_i)(\bar{o}_i - o_i)$. The full line search routine is outlined in Table 3.

Table 1 provides an abridged pseudo-code for L_2-SVM-MFN. See [15] for numerous other details. L_2-SVM-MFN alternates between calls to CGLS and line searches. Its computational complexity therefore is $O(t_{mfn}\bar{t}_{cgls}n_0)$ where t_{mfn} is the number of outer iterations of CGLS calls and line search, and $\bar{t}_{cgls}$ is the average number of CGLS iterations. These depend on the data set and the tolerance desired in the stopping criterion, but are typically very small[5]. Therefore, the complexity is found to be effectively linear in the number of entries in the data matrix.

3 Semi-supervised Linear SVMs

We now assume we have l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ and u unlabeled examples $\{x'_j\}_{j=1}^{u}$ with $x_i, x'_j \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Our goal is to construct a linear classifier $\mathrm{sign}(w^T x)$ that utilizes unlabeled data, typically in situations where $l \ll u$. We present semi-supervised algorithms that provide L_2-SVM-MFN the capability of dealing with unlabeled data.

3.1 Transductive SVM

Transductive SVM, originally proposed in [19], appends an additional term in the SVM objective function whose role is to drive the classification hyperplane towards low data density regions. Variations of this idea have appeared in the literature [10, 1, 9]. Since [10] appears to be the most natural extension of standard SVMs among these methods, and is popularly used in Text classification applications, we will focus on developing its large scale implementation.

The following optimization problem is setup for standard TSVM[6]:

$$
\begin{aligned}
w^\star = \operatorname*{argmin}_{w\in\mathbb{R}^d, \{y'_j\in\{-1,+1\}\}_{j=1}^{u}} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l}\max\left[0, 1 - y_i\ (w^T x_i)\right] \\
& + \frac{\lambda'}{2u}\sum_{j=1}^{u}\max\left[0, 1 - y'_j\ (w^T x'_j)\right] \\
\text{subject to:} \quad & \frac{1}{u}\sum_{j=1}^{u}\max\left[0, \mathrm{sign}(w^T x'_j)\right] = r
\end{aligned}
$$

The labels on the unlabeled data, $y'_1 \ldots y'_u$, are $\{+1, -1\}$-valued variables in the optimization problem. In other words, TSVM seeks a hyperplane w and a labeling of the unlabeled examples, so that the SVM objective

[5]For example, [15] reports a text classification experiment involving 198788 examples and 252472 features where $t_{mfn} = 11, \bar{t}_{cgls} = 102$.

[6]The bias term is typically excluded from the regularizer, but this factor is not expected to make any significant difference.

function is minimized, subject to the constraint that a fraction r of the unlabeled data be classified positive. SVM margin maximization in the presence of unlabeled examples can be interpreted as an implementation of the cluster assumption. In the optimization problem above, λ' is a user-provided parameter that provides control over the influence of unlabeled data [7]. If there is enough labeled data, λ, λ', r can be tuned by cross-validation. An initial estimate of r can be made from the fraction of labeled examples that belong to the positive class and subsequent fine tuning can be done based on performance on a validation set.

This optimization is implemented in [10] by first using an inductive SVM to label the unlabeled data and then iteratively switching labels and retraining SVMs to improve the objective function. The TSVM algorithm wraps around an SVM training procedure. The original (and widely popular) implementation of TSVM uses the SVM-Light software. There, the training of SVMs in the inner loops of TSVM uses dual decomposition techniques. As shown by experiments in [15], in sparse, linear settings one can obtain significant speed improvements with L_2-SVM-MFN over SVM-Light. Thus, by implementing TSVM with L_2-SVM-MFN, we expect similar improvements for semi-supervised learning on large, sparse datasets. As we will see, the L_2-SVM-MFN retraining steps in the inner loop of TSVM are typically executed extremely fast by using seeding techniques. Additionally, we also propose a version of TSVM where more than one pair of labels may be switched in each iteration. These speed-enhancement details are discussed in the following subsections.

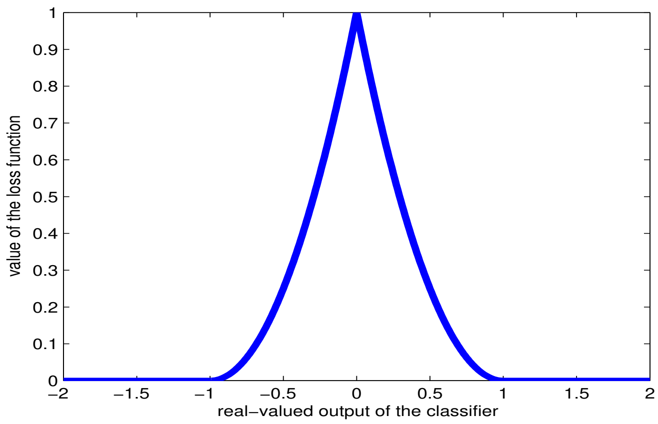
3.1.1 Implementing TSVM Using L_2-SVM-MFN

To develop the TSVM implementation with L_2-SVM-MFN, we need to consider the objective function corresponding to Eqn. 7 but with the L_2 loss function:

$$\begin{aligned} w^\star = \operatorname*{argmin}_{w \in \mathbb{R}^d, \{y'_j \in \{-1,+1\}\}_{j=1}^u} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} \max\left[0, 1 - y_i\ (w^T x_i)\right]^2 \\ & + \frac{\lambda'}{2u}\sum_{j=1}^{u} \max\left[0, 1 - y'_j\ (w^T x'_j)\right]^2 \\ \text{subject to:} \quad & \frac{1}{u}\sum_{j=1}^{u} \max\left[0, \operatorname{sign}(w^T x'_j)\right] = r \end{aligned} \tag{7}$$

[7]Suppose, the data has distinct clusters with a large margin, but the cluster assumption does not hold i.e the labeling given by the supervised classifier is actually the true labeling even though it cuts the clusters. In such cases, λ' can be set to 0 and standard SVM is retrieved. In general, λ' needs to be tuned for each data set.

Figure 1: L_2 Loss function over unlabeled examples for Transductive SVM



Note that this objective function above can also be equivalently written in terms of the following loss over each unlabeled example x:

$$\min\left(\max\left[0, 1-(w^Tx)\right]^2, \max\left[0, 1+(w^Tx)\right]^2\right) = \max\left[0, 1-|w^Tx|\right]^2$$

Here, we pick the value of the label variable y that minimizes the loss on the unlabeled example x, and rewrite in terms of the absolute value of the output of the classifier on x. This loss function is shown in Fig. 1. We note in passing that, L_1 and L_2 loss terms over unlabeled examples are very similar on the interval $[-1, +1]$. The non-convexity of this loss function implies that the TSVM training procedure is susceptible to local optima issues. In the next section, we will outline a mean field annealing procedure that can overcome this problem.

The TSVM algorithm with L_2-SVM-MFN is outlined in Table 4 and closely follows the presentation in [10][8]. A classifier is obtained by first running L_2-SVM-MFN on just the labeled examples. Temporary labels are assigned to the unlabeled data by thresholding the soft outputs of this classifier so that the fraction of the total number of unlabeled examples that are temporarily labeled positive equals the parameter r.

Then starting from a small value of λ', the unlabeled data is gradually brought in by increasing λ' by a factor of 2 in the outer loop. This gradual increase of the influence of the unlabeled data is a way to protect TSVM from being immediately trapped in a local minimum. An inner loop identifies pairs of unlabeled examples with positive and a negative temporary labels such that switching these labels would decrease the objective function. L_2-SVM-MFN is then retrained with the switched labels.

[8]A minor difference is that in our implementation, we did not use separate cost factors for balancing loss terms for the positive and negative class

3.1.2 Multiple Switching

The TSVM algorithm presented in [10] involves switching a single pair of labels. We propose a variant where upto S pairs are switched such that the objective function improves. Here, S is a user controlled parameter. Setting $S = 1$ recovers the original TSVM algorithm, whereas setting $S = u/2$ switches as many pairs as possible in the inner loop (Loop 2 in Table 4) of TSVM. The implementation is conveniently done as follows:

1. Identify unlabeled examples with active indices and currently positive labels. Sort corresponding outputs in ascending order. Let the sorted list be L^+.

2. Identify unlabeled examples with active indices and currently negative labels. Sort corresponding outputs in descending order. Let the sorted list be L^-.

3. Pick pairs of elements, one from each list, from the top of these lists until either a pair is found such that the output from L^+ is greater than the output from L^-, or if S pairs have been picked.

4. Switch the current labels of these pairs.

Using arguments similar to Theorem 2 in [10] we can show that Transductive L_2-SVM-MFN with multiple-pair switching converges in a finite number of steps.

Proposition: Transductive L_2-SVM-MFN with multiple-pair switching converges in finite number of steps.

Proof: The outer loop (marked Loop 1 in Table 4) clearly terminates in finite number of steps. Each call to L_2-SVM-MFN terminates in finite number of iterations due to Theorem 1 in [15]. We only need to show that Loop 2 also has finite termination. Let $J(w, Y')$ be the value of the TSVM objective function for some candidate weight vector w and candidate label vector $Y' = [y'_1 \dots y'_u]$ over the unlabeled data. Let $w(Y'), Y'$ be the operating variables at the end of an iteration of loop 2 where $w(Y') = \mathrm{argmin}_{w \in \mathbb{R}^d} J(w, Y')$. After switching labels, let the new operating label vector be Y''. It is easy to see that:

$$J(w(Y'), Y') > J(w(Y'), Y'') \geq J(w(Y''), Y'')$$

The second inequality follows since $w(Y'')$ minimizes $J(w, Y'')$ over all w. To see the first inequality observe that for any pair of data points (say with indices i, j) whose labels are switched, the following conditions are satisfied: $y'_i = 1, y'_j = -1, w(Y')^T x'_i < 1, -w(Y')^T x'_j < 1, w(Y')^T x'_i < w(Y')^T x'_j$.

The terms contributed by this pair to the objective function decrease after switching labels since the switching conditions imply the following:

$$\max[0, 1 - w^T x_i']^2 + \max[0, 1 + w^T x_j']^2 = (1 - w^T x_i')^2 + (1 + w^T x_j')^2$$
$$> (1 + w^T x_i')^2 + (1 - w^T x_j')^2 \geq \max[0, 1 + w^T x_i']^2 + \max[0, 1 - w^T x_j']^2$$

Thus, swapping the labels of multiple pairs that satisfy the switching conditions reduces the objective function.

Since at the end of consecutive iterations $J(w(Y'), Y') > J(w(Y''), Y'')$, Loop 2 must terminate in finite number of steps because there are only a finite number of possible label vectors. □

We are unaware of any prior work that suggests and evaluates this simple heuristic of switching more than one label. Our experimental results in section 4 establish that this heuristic is very effective in speeding up TSVM training while maintaining generalization performance on textual problems.

3.1.3 Seeding

The effectiveness of L_2-SVM-MFN on large sparse datasets combined with the efficiency gained from seeding w in the re-training steps (after switching labels or after increasing λ') make this algorithm quite attractive. Consider an iteration in Loop 2 of TSVM where a new pair of labels has been switched, and the solution w from the last retraining of L_2-SVM-MFN (marked as Retraining 2 in Table 4) is available for seeding. According to Theorem 1 in [15], when the last L_2-SVM-MFN converged, its solution w is given by the linear system[9]:

$$\left[\lambda I + X_{I(w)}^T C_{I(w)} X_{I(w)}\right] w = X_{I(w)}^T C_{I(w)} Y$$

where Y is the current label vector. When labels Y_i, Y_j are switched, back at the top of loop 2, the label vector is updated as:

$$Y = Y + 2e_{ij}$$

where e_{ij} is a vector whose elements zero everywhere except in the i^{th} and the j^{th} position which are +1 and -1 or -1 and +1 respectively. Note also that if $i, j \in \jmath(w)$ the re-training of L_2-SVM-MFN with w as the starting guess immediately encounters a call CGLS to solve the following perturbed system:

$$\left[\lambda I + X_{\jmath(w)}^T C_{\jmath(w)} X_{\jmath(w)}\right] \tilde{w} = X_{\jmath(w)}^T C_{\jmath(w)} \left[Y + 2e_{ij}\right]$$

[9] The subsequent line search does not change this w; therefore, the optimality conditions are checked immediately after the CGLS step

The starting residual vector r^0 is given by:

$$
\begin{aligned}
r^0 &= X^T_{\jmath(w)} C_{\jmath(w)} \left[Y + 2e_{ij}\right] - \left[\lambda I + X^T_{\jmath(w)} C_{\jmath(w)} X_{\jmath(w)}\right] w \\
&= r(w) + 2X^T_{\jmath(w)} C_{\jmath(w)} e_{ij} \\
&\leq \epsilon + 2\lambda' \|x_i - x_j\|
\end{aligned}
$$

where $r(w)$ in the second step is the final residual of w which fell below ϵ at the convergence of the last re-training. In applications like Text categorization, TFIDF feature vectors are often length normalized and have positive entries. Therefore, $\|x_i - x_j\| \leq \sqrt{2}$. This gives the following bound on the starting residual:

$$
r^0 \leq \epsilon + 2\sqrt{2}\lambda'
$$

which is much smaller than a bound of $n\sqrt{n}\lambda'$ with a zero starting vector. Seeding is quite effective for Loop 1 as well, where λ' is changed, as demonstrated by experiments in [15]. With the two additional loops, the complexity of Transductive L_2-TSVM-MFN becomes $O(n_{switches} \bar{t}_{mfn} \bar{t}_{cgls} n_0)$, where $n_{switches}$ is the number of label switches. The outer loop executes a fixed number of times; the inner loop calls L_2-TSVM-MFN $n_{switches}$ times. Typically, $n_{switches}$ is expected to strongly depend on the data set and also on the number of labeled examples. Since it is difficult to apriori estimate the number of switches, this is an issue that is best understood from empirical observations.

3.2 Mean Field Annealing

The transductive SVM loss function over the unlabeled examples can be seen from Fig. 1 to be non-convex. This makes the TSVM optimization procedure susceptible to local minimum issues causing a loss in its performance in many situations, e.g as recorded in [6]. We now present a new algorithm based on mean field annealing that can potentially overcome this problem while also being computationally very attractive for large scale applications.

Mean Field Annealing [14, 3, 4] (MFA) is an established tool for combinatorial optimization that approaches the problem from information theoretic principles. The discrete variables in the optimization problem are relaxed to continuous probability variables and a non-negative temperature parameter T is used to track the global optimum.

We begin by re-writing the TSVM objective function as follows:

$$
w^\star = \operatorname*{argmin}_{w \in \mathbb{R}^d, \{\mu_j \in \{0,1\}\}_{j=1}^u} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} \max\left[0, 1 - y_i\ (w^T x_i)\right]^2
$$
$$
+\frac{\lambda'}{2u}\sum_{j=1}^{u}\left(\mu_j \max\left[0, 1 - (w^T x'_j)\right]^2 + (1-\mu_j)\max\left[0, 1 + (w^T x'_j)\right]^2\right)
$$

Here, we introduce binary valued variables $\mu_j = (1 + y_j)/2$. Let $p_j \in [0, 1]$ denote the belief probability that the unlabeled example x'_j belongs to the positive class. The Ising model [10] of Mean field annealing motivates the following objective function, where we relax the binary variables μ_j to probability variables p_j, and include entropy terms for the distributions defined by p_j:

$$
\begin{aligned}
w^\star_T = \operatorname*{argmin}_{w \in \mathbb{R}^d, \{p_j \in [0 \;\; 1]\}_{j=1}^u} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} \max\left[0, 1 - y_i \; (w^T x_i)\right]^2 \\
+ \frac{\lambda'}{2u}\sum_{j=1}^{u} \Big(p_j \max\left[0, 1 - (w^T x'_j)\right]^2 & + (1 - p_j) \max\left[0, 1 + (w^T x'_j)\right]^2 \Big) \\
& + \frac{T}{2u}\sum_{j=1}^{u} \left(p_j \;\log\; p_j + (1 - p_j) \;\log\; (1 - p_j)\right)
\end{aligned}
\tag{8}
$$

Here, the "temperature" T parameterizes a family of objective functions. The objective function for a fixed T is minimized under the following class balancing constraints:

$$\frac{1}{u}\sum_{j=1}^{u} p_j = r \tag{9}$$

where r is the fraction of the number of unlabeled examples belonging to the positive class. As in TSVM, r is treated as a user-provided parameter. It may also be estimated from the labeled examples.

The solution to the optimization problem above is tracked as the temperature parameter T is lowered to 0. The final solution is given as:

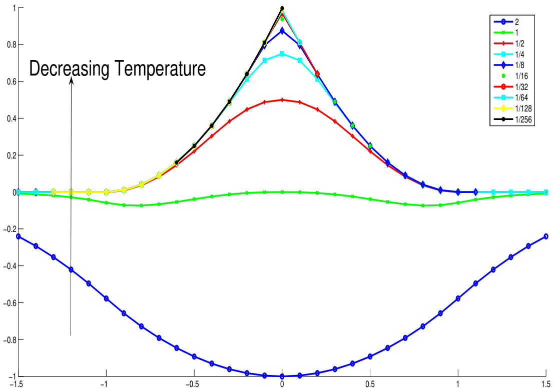
$$w^\star = \lim_{T \to 0} w^\star_T \tag{10}$$

In practice we monitor the value of the objective function in the optimization path and return the solution corresponding to the minimum value achieved.

To develop an intuition for the working on this method, we consider the loss term in the objective function associated with an unlabeled example as a function of the output of the classifier. This loss term is based on calculations to be described below. Fig. 2 plots this loss term for various values of T. As the temperature is decreased, the loss function deforms from a squared-loss shape where a global optimum is easier to achieve, to the TSVM loss function in Fig. 1. At high temperatures a global optimum is easier to obtain. The global minimizer is then slowly tracked as the temperature is lowered towards zero.

[10] A multiclass extension would use the Potts glass model. There, one would have to append the entropy of the distribution over multiple classes to a multi-class objective function.

Figure 2: L_2 Loss function over unlabeled examples for Transductive SVM



The optimization is done in stages, starting with high values of T and then gradually decreasing T towards 0. For each T, the problem in Eqns. 8,9 is optimized by alternating the minimization over w and $p = [p_1 \dots p_u]$ respectively. Fixing p, the optimization over w is done by L_2-SVM-MFN. Fixing w, the optimization over p can also be done easily as described below. Both these problems involve convex optimization and can be done exactly and efficiently. We now provide the details of these optimization steps.

3.2.1 Optimizing w

We describe the steps to efficiently implement the L_2-SVM-MFN loop for optimizing w keeping p fixed. The call to L_2-SVM-MFN is made on the data $\hat{X} = \left[X^T \; X'^T \; X'^T\right]^T$ whose first l rows are formed by the labeled examples, and the next $2u$ rows are formed by the unlabeled examples appearing as two repeated blocks. The associated label vector and cost matrix are given by

$$\hat{Y} = [y_1, y_2 ... y_l, \overbrace{1, 1, ...1,}^{u} \overbrace{-1, -1 ... -1}^{u}]$$

$$C = diag\left[\overbrace{\frac{1}{l}...\frac{1}{l}}^{l}, \; \overbrace{\frac{\lambda'\, p_1}{u}...\frac{\lambda'\, p_u}{u}}^{u} \; \overbrace{\frac{\lambda'(1-p_1)}{u}...\frac{\lambda'(1-p_u)}{u}}^{u}\right] \qquad (11)$$

Even though each unlabeled data contributes two terms to the objective function, effectively only one term contributes to the complexity. This is because matrix-vector products, which form the dominant expense in L_2-SVM-MFN, are performed only on unique rows of a matrix. The output may be duplicated for duplicate rows. Infact, we can re-write the CGLS calls in L_2-SVM-MFN so that the unlabeled examples appear only once in the data matrix. Consider the CGLS call at some iteration where the active index set is $\jmath = \jmath(w)$ for some current candidate weight vector w:

$$\left[\lambda I + \hat{X}_\jmath^T C_\jmath \hat{X}_\jmath\right] \ \bar{w} = \hat{X}^T C_\jmath \hat{Y}_\jmath \tag{12}$$

Note that if $|w^T x'_j| \geq 1$, the unlabeled example x'_j appears as one row in the data matrix $\hat{X}_\jmath$ with label given by $-\text{sign}(w^T x'_j)$. If $|w^T x'_j| < 1$, the unlabeled example x'_j appears as two identical rows $\hat{X}_\jmath$ with both labels. Let $\jmath_l \in 1 \ldots l$ be the indices of the labeled examples in the active set, $\jmath'_1 \in 1 \ldots u$ be the indices of unlabeled examples with $|w^T x'_j| \geq 1$ and $\jmath'_2 \in 1 \ldots u$ be the indices of unlabeled examples with $|w^T x'_j| < 1$. Note that the index of every unlabeled example appears in one of these sets i.e, $\jmath'_1 \cup \jmath'_2 = 1 \ldots u$. Eqn. 12 may be re-written as:

$$\left[\lambda I + \frac{1}{l}\sum_{i \in \jmath_l} x_i^T x_i + \frac{\lambda'}{u}\sum_{j \in \jmath'_1} c_j x_j'^T x_j + \frac{\lambda'}{u}\sum_{j \in \jmath'_2} x_j'^T x_j\right] \ \bar{w} =$$
$$\frac{1}{l}\sum_{i \in \jmath_l} y_i x_i - \frac{\lambda'}{u}\sum_{j \in \jmath'_1} c_j \text{sign}(w^T x_j) x_j + \frac{\lambda'}{u}\sum_{j \in \jmath'_2} (2p_j - 1) x_j$$

where $c_j = p_j$ if $\text{sign}(w^T x'_j) = -1$ and $c_j = 1 - p_j$ if $\text{sign}(w^T x'_j) = 1$. Re-writing in matrix notation, we obtain an equivalent linear system that can be solved by CGLS:

$$\left[\lambda I + \tilde{X}^T \tilde{C} \tilde{X}\right] \ \bar{w} = \tilde{X}^T \tilde{C} \tilde{Y} \tag{13}$$

where $\tilde{X} = [X_{\jmath_l}^T \ \ X']$, $\tilde{C}$ is a diagonal matrix and $\tilde{Y}$ is the vector of effectively active labels. These are defined by:

$$\begin{aligned}
\tilde{C}_{jj} &= \frac{1}{l} \ , \ \tilde{Y}_j = y_i \quad \ j \in 1 \ldots |\jmath_l| \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} &= \frac{\lambda' p_j}{u} \ , \ \tilde{Y}_{j+|\jmath_l|} = 1 \ \text{ if } j \in 1 \ldots u \ , j \in \jmath'_1, \ \text{sign}(w^T x'_j) = -1 \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} &= \frac{\lambda'(1 - p_j)}{u} \ , \ \tilde{Y}_{j+|\jmath_l|} = -1 \ \text{ if } \ j \in 1 \ldots u \ \ j \in \jmath'_1, \quad \text{sign}(w^T x'_j) = 1 \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} &= \frac{\lambda'}{u} \ , \ \tilde{Y}_{j+|\jmath_l|} = (2p_j - 1) \ \text{ if } \ j \in 1 \ldots u \ \ j \in \jmath'_2
\end{aligned} \tag{14}$$

Thus, CGLS needs to only operate on data matrices with one instance of each unlabeled example using a suitably modified cost matrix and label vector.

After the CGLS step, one needs to check the optimality conditions. The optimality conditions can be re-written as:

$$\begin{array}{ll} \forall\, i \in \jmath_l & y_i \bar{o}_i \leq 1+\epsilon \\ \forall\, i \in \jmath_l^c & y_i \bar{o}_i \geq 1-\epsilon \\ \forall\, j \in \jmath_1' & |\bar{o}_j'| \geq 1-\epsilon \\ \forall\, j \in \jmath_2' & |\bar{o}_j'| \leq 1+\epsilon \end{array}$$

For the subsequent line search step, we reassemble appropriate output and label vectors to call the routine in Table 3. The steps for optmizing w are outlined in Table 7.

3.2.2 Optimizing p

For the latter problem of optimizing p for a fixed w, we construct the Lagrangian:

$$\begin{aligned} \mathcal{L} &= \frac{\lambda'}{2u}\sum_{j=1}^{u}\left(p_j \max\left[0, 1-(w^T x_j')\right]^2 + (1-p_j)\max\left[0, 1+(w^T x_j')\right]^2\right) \\ &\quad + \frac{T}{2u}\sum_{j=1}^{u}\left(p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)\right) - \nu\left[\frac{1}{u}\sum_{j=1}^{u} p_j - r\right] \end{aligned}$$

Differentiating the Lagrangian with respect to p_j, we get:

$$\frac{\partial\, \mathcal{L}}{\partial p_j} = \frac{\lambda'}{2u}\left(\max\left[0, 1-(w^T x_j')\right]^2 - \max\left[0, 1+(w^T x_j')\right]^2\right) + \frac{T}{2u}\log\frac{p_j}{1-p_j} - \frac{\nu}{u} = 0$$

Define:

$$g_j = \lambda'\left(\max\left[0, 1-(w^T x_j')\right]^2 - \max\left[0, 1+(w^T x_j')\right]^2\right)$$

Then, the expression for p_j is given by:

$$p_j = \frac{1}{1+e^{\frac{g_j - 2\nu}{T}}} \tag{15}$$

Substituting this expressing in the balance constraint in Eqn. 9, we get a one-dimensional non-linear equation in 2ν:

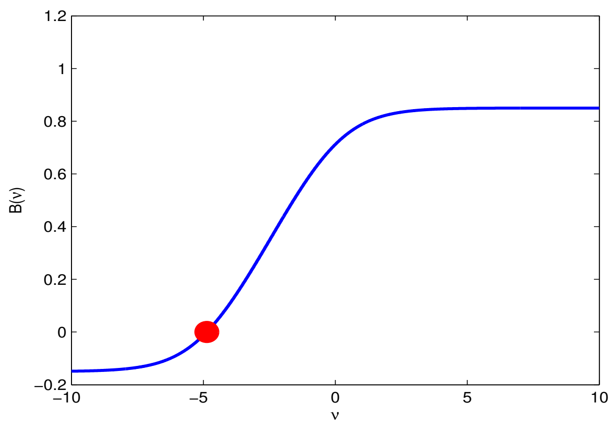
$$\frac{1}{u}\sum_{j=1}^{u}\frac{1}{1+e^{\frac{g_i - 2\nu}{T}}} = r$$

The value of 2ν is computed exactly by using a hybrid combination of Newton-Raphson iterations and the bisection method to find the root of the function $B(\nu) = \frac{1}{u}\sum_{i=l+1}^{l+u} \frac{1}{1+e^{\frac{g_i-\nu}{T}}} - r$. This method is rapid due to the quadratic convergence properties of Newton-Raphson iterations and fail-safe due to bisection steps. Note that the root exists and is unique, since one can see that $B(\nu) \to 1-r > 0$ when $\nu \to \infty$, $B(\nu) \to -r < 0$ when $\nu \to -\infty$, and $B(\nu)$ is a continuous, monotonically increasing function (see Fig.3). The root finding begins by bracketing the root in the interval $[\nu_-, \nu_+]$ so that $B(\nu_-) < 0, B(\nu_+) > 0$ where ν_-, ν_+ are given by:

$$\nu_- = \min(g_1 \ldots g_u) - T \log \frac{1-r}{r}$$
$$\nu_+ = \max(g_1 \ldots g_u) - T \log \frac{1-r}{r}$$

The hybrid root finding algorithm performs Newton-Raphspon iterations with a starting guess of $\frac{\nu_- + \nu_+}{2}$ and invokes the bisection method whenever an iterate goes outside the brackets. The steps for optimizing p are outlined in Table 6.

Figure 3: The value of $B(\nu) = \frac{1}{u}\sum_{i=l+1}^{l+u} \frac{1}{1+e^{\frac{g_i-\nu}{T}}} - r$ as a function of ν. Here, $T = 1, r = 0.15, g_i \in [-5, 5]$. The root is marked on the plot.



3.2.3 Stopping Criteria

For a fixed T, this alternate minimization proceeds until some stopping criterion is satisfied. A natural criterion is the mean Kullback-Liebler divergence (relative entropy)[11] $KL(p, q)$ between current values of p_i and the values, say q_i, at the end of last iteration. Thus the stopping criterion for fixed T

[11] Other distance measures e.g euclidean distance may also be used instead.

is:

$$KL(p, q) = \sum_{j=1}^{u} p_j \ \log \ \frac{p_j}{q_j} + (1 - p_j) \ \log \ \frac{1 - p_j}{1 - q_j} < u\epsilon \tag{16}$$

A good value for ϵ is 10^{-6}. The temperature may be decreased in the outer loop until the total entropy falls below a threshold, which we take to be $\epsilon = 10^{-6}$ as above.

$$H(p) = -\sum_{j=l}^{u} (p_j \ \log \ p_j + (1 - p_j) \ \log \ (1 - p_j)) < u\epsilon \tag{17}$$

Note that we set upper limits on the total number of iterations in both loops. Often the entropy may converge to a value larger than the stopping threshold and the outer loop is exited by exeeding the maximum iterations allowed. In such cases, due to seeding, the extra iterations add insignificant cost.

The TSVM objective function is monitored as the optimization proceeds.

$$\begin{aligned} J(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} \max\left[0, 1 - y_i \ (w^T x_i)\right]^2 \\ + \frac{\lambda'}{2u}\sum_{j=1}^{u} \max\left[0, 1 - |w^T x'_j|\right]^2 \end{aligned} \tag{18}$$

The weight vector corresponding to the minimum transductive cost in the optimization path is returned as the solution.

The steps of mean field annealing with L_2-SVM-MFN are outlined in Table 5.

4 Empirical Study

Semi-supervised learning experiments were conducted to test these algorithms on six text binary classification problems. These are listed in Table 8.

Table 8: Two-class datasets used in the experiments : d is the data dimensionality, $\bar{n}_0$ is the average number of non-zero entries per example vector, $l + u$ is the number of labeled and unlabeled examples, t is the number of test examples.

Dataset	d	$\bar{n}_0$	$l + u$	t
auto-vs-aviation	20707	51.32	35588	35587
real-vs-simulated	20958	51.32	36155	36154
ccat	47236	75.93	17332	5787
gcat	47236	75.93	17332	5787
33-vs-36	59072	26.56	41346	41346
pcmac	7511	54.58	1460	486

The auto-vs-aviation and real-vs-simulated binary classification datasets come from a collection of UseNet articles [12] from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation[13]. The ccat and gcat data sets pose the problem of separating corporate and governement related articles respectively; these are the top-level categories in the RCV1 training data set [11]. These data sets create an interesting situation where semi-supervised learning is required to learn different low density separators respecting different classification tasks in the same input space. The 33-vs-36 data set is a subset of a multiclass Yahoo shopping data set. Finally, the pcmac data set is a small subset of the 20-newsgroups data popularly used in semi-supervised learning literature (e.g in [6, 16]). The results below are averaged over 10 random[14] splits of training (labeled and unlabeled) and test sets. The amount of labeled data in the training set was gradually varied to generate learning curves. We use a default value of $\lambda' = 1$ for all datasets except[15] for auto-vs-aviation and ccat where $\lambda' = 10$. The default value of $\lambda = 0.001$ was used for all datasets.

Minimization of Objective Function

We first examine the effectiveness of TSVM and MFA in optimizing the TSVM objective function. In Figure 4, we plot the minimum value of the objective function achieved by TSVM and MFA with respect to varying number of labels. As compared to TSVM, we see that MFA performs significantly better optimization on auto-vs-aviation, ccat, and pcmac datasets and slightly better optimization on the other datasets.

Transduction Learning Curves

In Figures 5, 6, 7 we plot error rates over unlabeled examples for SVM, TSVM and MFA with respect to varying amounts of labeled data.

The following observations can be made:

1. Comparing the learning curves for SVM against the semi-supervised algorithms, the benefit of unlabeled data is evident on all datasets, and is particularly striking on auto-vs-aviation, real-vs-simulated, gcat, and pcmac.

2. On auto-vs-aviation and pcmac , MFA outperforms TSVM significantly. On ccat, MFA performs a much better optimization but this only translates into slight error rate improvements. MFA and

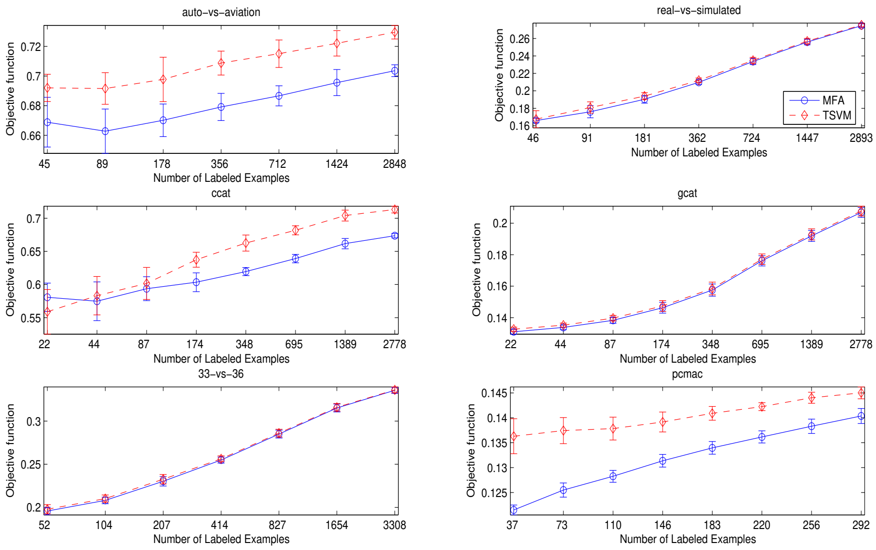
[12] available at http://www.cs.umass.edu/~mccallum/data/sraa.tar.gz

[13] We used the RAINBOW software [13] to generate feature vectors after removing words that only appear in 10 or fewer documents.

[14] The class ratios are maintained in these splits.

[15] This produced better results for both TSVM and MFA. A careful optimization of λ' was not attempted.

Figure 4: Minimum value of objective function achieved by MFA and Transductive L_2-SVM-MFN with respect to number of labels.



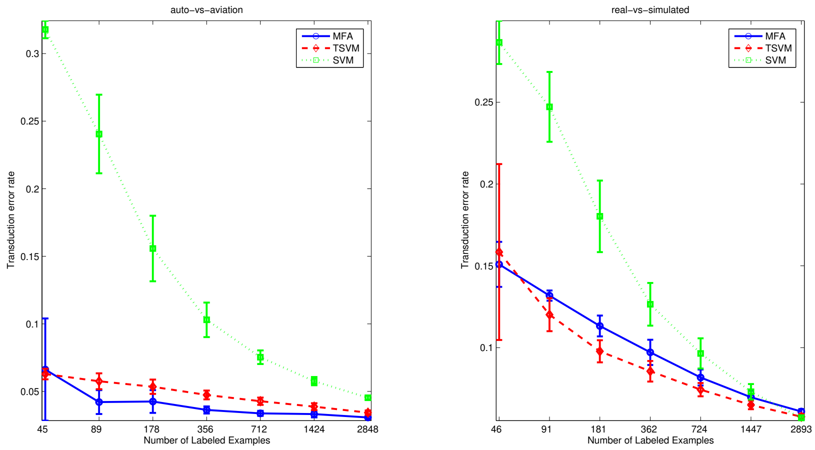
TSVM are very closely matched on gcat and 33-vs-36. On real-vs-simulated dataset, TSVM and MFA perform very similar optimization of the transduction objective function (see Figure 4), but appear to return very different solutions. The TSVM solution returns lower error rates as compared to MFA on this dataset.

3. For TSVM, on all datasets we found that multiple switching returned nearly identical performance as single switching. Since it saves significant computation time, our study establishes multiple switching as a valuable heuristic for applications of TSVM.

Out-of-Sample Learning Curves

In Figures 8, 9, 10 we plot error rates over unseen test examples for SVM, TSVM and MFA with respect to varying amounts of labeled data. Comparing with learning curves for transduction, we see the observations in the previous section are also true for out-of-sample performance. Both TSVM and MFA provide high quality extension to unseen test data.

Figure 5: Transduction error rate with respect to number of labels



Computational Timings

In Figure 11, we plot the average computation time for MFA and TSVM with 1, 10, 100 and maximum switching. The following observations can be made:

1. The standard single switch TSVM is an order of magnitude slower than the multiple switching variant.

2. MFA takes a moderate amount of time. It is significantly faster than single switch TSVM and typically slower than TSVM with maximum switching.

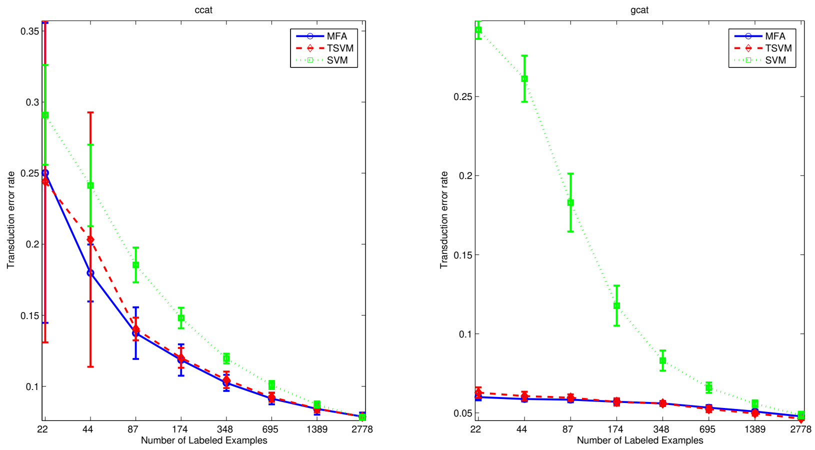
3. The TSVM computation time is more strongly dependent on the number of labels than MFA.

Comparison with SVM$^{\text{light}}$

In Table 9, we compare our implementations with SVM$^{\text{light}}$ at its default optimization settings on the first split at the lowest end of the learning curves. These comparisons demonstrate massive speedups with our methods over the dual techniques used in SVM$^{\text{light}}$.

Note that the results presented in this section were obtained with a MATLAB implementation with a C interface to the core CGLS routine. In our experiments, the computational difference between single and multiple switching may have been somewhat exaggerated due to implementation in

Figure 6: Transduction error rate with respect to number of labels

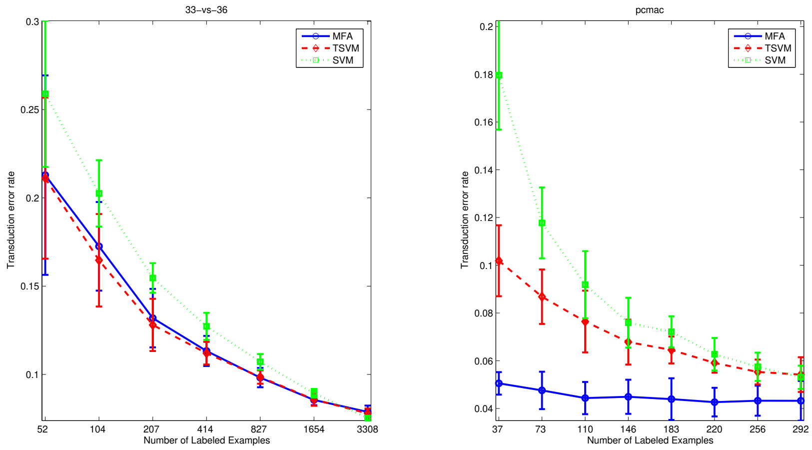


MATLAB (since single switching involves more non-vectorized loops which are not optimized in MATLAB). We expect significantly faster computation times with a full C or a fortran implementation, especially with parallel computation of matrix vector products.

Table 9: Speed (in seconds) and error rate comparisons with $\text{SVM}^{\text{light}}$. S=1 and S=max denote single and multiple-switching implementations of TSVM with the modified finite newton procedure.

Dataset		$\text{SVM}^{\text{light}}$	S=1	S=max	MFA
auto-vs-aviation	time	101759	5849	390	1446
	error rate	0.0664	0.0576	0.0575	0.0595
real-vs-simulated	time	498313	6244	373	1129
	error rate	0.1589	0.1430	0.1426	0.1460
ccat	time	13540	2352	390	1185
	error rate	0.4071	0.2098	0.2081	0.1861
gcat	time	243840	1267	358	159
	error rate	0.0665	0.0646	0.0639	0.0591
33-vs-36	time	48390	7406	309	393
	error rate	0.2290	0.2140	0.2136	0.2206
pcmac	time	167	4	2	12
	error rate	0.0597	0.0782	0.0802	0.0556

Figure 7: Transduction error rate with respect to number of labels



Importance of Annealing

To confirm the necessity of an annealing component (tracking the minimizer with respect to T) in the optimization, we compare MFA with the alternating w,p optimization procedure where the temperature parameter is held fixed at $T = 1$ and $T = 0.1$. In Figure 12 we plot the minimum value of the objective function achieved with and without annealing. The corresponding learning curves are plotted in Figure 13. We see that annealing provides higher quality solutions as compared to fixed temperature optimization.

It is important to note that the gradual increase of λ' to the user-set value in TSVM is also a mechanism to avoid local optima. The non-convex part of the objective function is gradually increased to a desired value. In this sense, λ' simultaneously plays the role of an annealing parameter and also controls the strength of the cluster assumption. This dual role has the advantage that a suitable λ' can be chosen by monitoring performance on a validation set as the algorithm proceeds. In MFA, however, we directly use an established framework for global optimization of a non-convex objective function, decoupling annealing from the implementation of the cluster assumption. As our experiments show, this can lead to significantly better solutions on many problems.

Figure 8: Test error rate with respect to number of labels

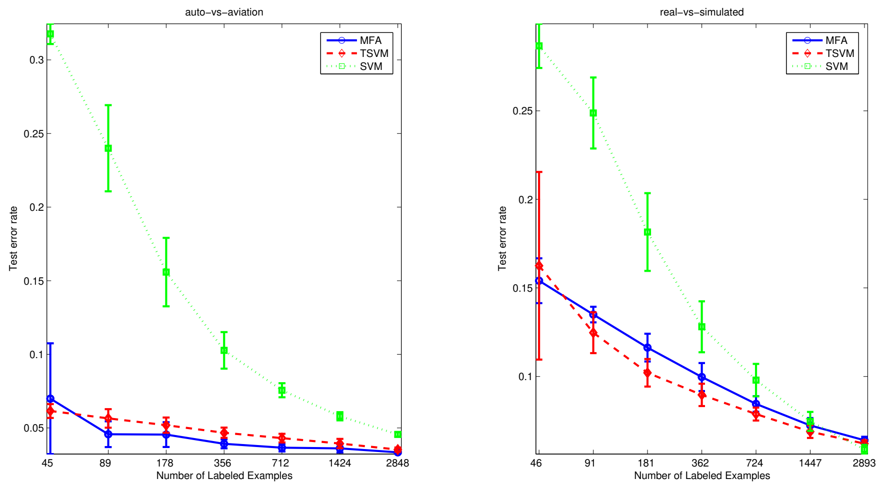


Figure 9: Test error rate with respect to number of labels

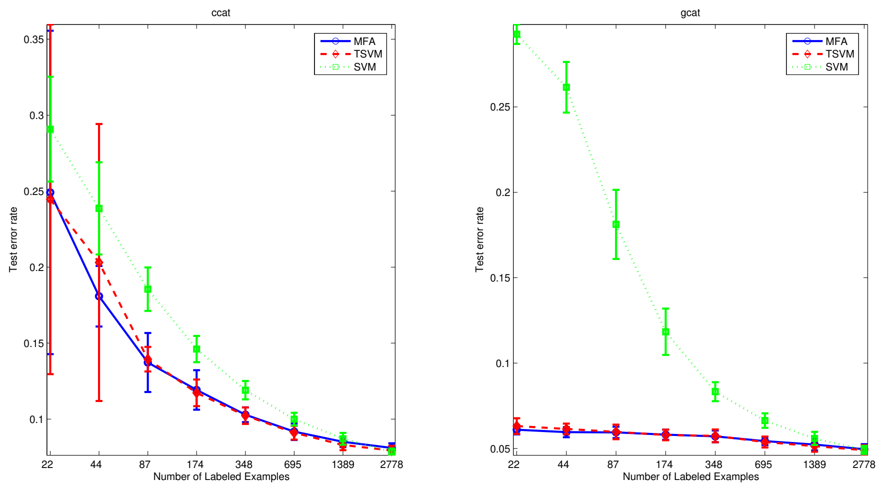


Figure 10: Test error rate with respect to number of labels

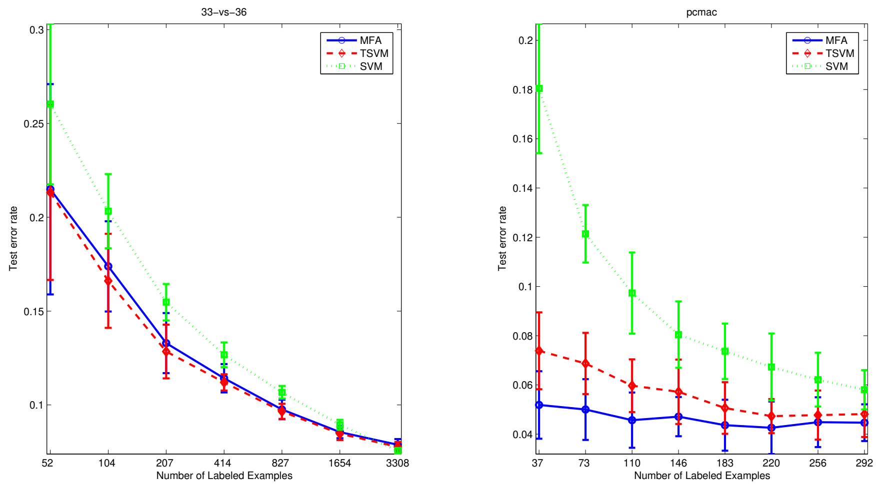


Figure 11: Computation time with respect to number of labels for MFA and Transductive L_2-SVM-MFN with single and multiple switches.

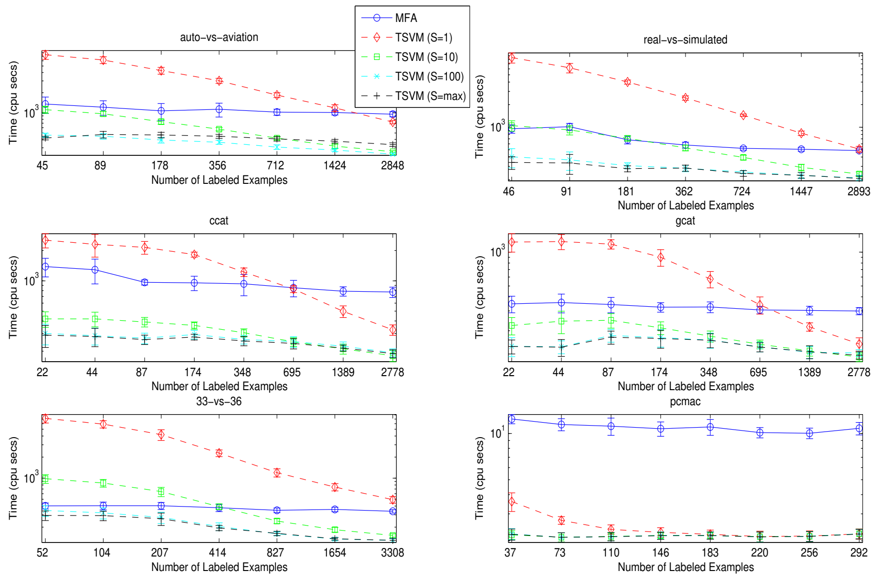


Figure 12: Minimum value of objective function achieved by MFA and a fixed temperature optimization with respect to number of labels.

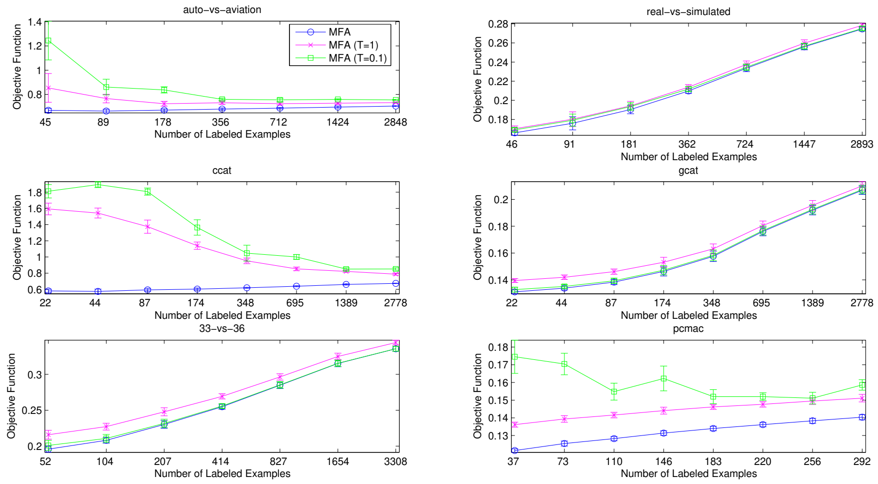
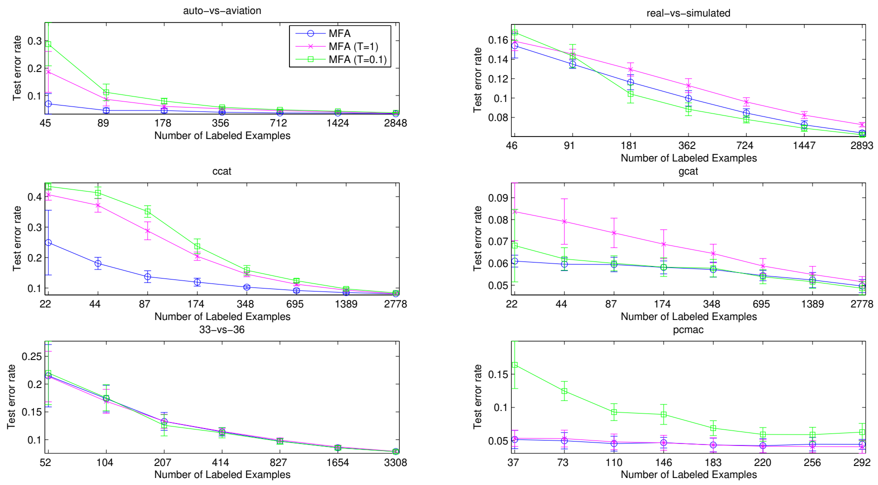


Figure 13: Error Rates achieved by MFA and a fixed temperature optimization with respect to number of labels.



5 Extensions and Applications

In this section, we collect our thoughts on various extensions and applications of the methods proposed in this paper, and outline some directions for future work.

Non-linear Mean Field Annealing

In many problems of interest one needs to construct complex, non-linear classification boundaries. The MFA algorithm can be extended to operate in a non-linear Reproducing Kernel Hilbert Space (RKHS) of functions $\mathcal{H}_K$ defined by a kernel function K. Specifically, we solve an optimization problem of the following kind:

$$
\begin{aligned}
f_T^\star = \operatorname*{argmin}_{f \in \mathcal{H}_K, \{p_j \in [0\ \ 1]\}_{j=1}^u} \quad & \frac{\lambda}{2}\|f\|_K^2 + \frac{1}{l}\sum_{i=1}^{l} V(y_i, x_i, f) \\
& + \frac{\lambda'}{u}\sum_{j=1}^{u}\left[p_j V(1, x_j', f) + (1-p_j) V(-1, x_j', f)\right] \\
+ \frac{T}{u}\sum_{j=1}^{u}\left(p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)\right) \quad & \text{subject to: } \frac{1}{u}\sum_{j=1}^{u} p_j = r
\end{aligned}
$$

Here, $V(y, x, f)$ is a loss function and $\|f\|_K$ is the norm of f in the RKHS $\mathcal{H}_K$. Annealing is performed by taking the limit $f^\star = \lim_{T \to 0} f_T^\star$.

By the representer theorem, the solution to the above problem, for fixed p, is given as:

$$
f_T^\star(p) = \sum_{i=1}^{l+u} \alpha_i K(x, x_i)
$$

where the expansion coefficients α_i depend on both T and p. Non-linear MFA can be easily implemented for a variety of loss functions. For example, if the loss function V is continously differentiable, one can use a non-linear version of the modified finite newton procedure to optimize the $l+u$ primal variables α. Alternatively, for the hinge loss $V(y, x, f) = \max\left[0, 1 - yf(x)\right]$, one can use a standard implementation of sequential minimal optimization (SMO). This optimization is then alternated with optimization of p given by a derivation similar to that in section 3.2.2. In an outer loop, the temperature is gradually reduced.

An interesting family of hybrid algorithms can be developed that run non-linear MFA with a class of kernels [16] that are adapted for semi-supervised learning, based on geometric structures of the data estimated using a data-adjacency graph. The benefits of a hybrid approach that combines graph-based semi-supervised learning with TSVM-style optimization has been noted in [6]. The former techniques provide data representations and complexity notions that are better suited for semi-supervised learning, and the latter techniques complement these with explicit optimization over the unknown labels.

One-Class Problems, Clustering and Fully Supervised Learning

TSVM and MFA can be deployed in a variety of settings involving different amounts of labeled and unlabeled data. Many real world settings present the task of identifying members of a certain class as opposed to distinguishing between well-specified classes. For example, in order to identify web documents concerning sports, it is much easier to label sports documents than to label the diverse and ill-characterized set of non-sports documents. In such problems, labeled examples come from a single class, very often with large amounts of unlabeled data containing some instances of the class of interest and many instances of the "others" class.

Being a special case of semi-supervised learning, the problem of one-class learning with unlabeled data can be addressed by the algorithms developed in this paper. Recall that these algorithms implement the cluster assumption under constraints on class ratios. For one-class problems, unlabeled data is expected to be helpful in biasing the classification hyperplane to pass through a low density region keeping clusters of the relevant class on one side and instances of the "others" class on the other side.

A novel class of linear clustering algorithms may arise by adapting our algorithms for the extreme case where no labels are available. At the other extreme of fully supervised learning, one may utilize MFA ideas for optimizing non-convex loss functions closer to the missclassification (zero-one) loss (as compared to e.g the hinge loss), yielding algorithms with possibly superior generalization performance and better sparsity properties. See [7] for some recent work on regularization algorithms with non-convex loss functions.

References

[1] K. Bennett and A. Demirez (1998),*Semi-Supervised Support Vector Machines* NIPS 1998

[2] D.Bertsekas ,*Nonlinear Programming*, Athena Scientific, 1995, (2nd Edition, 1999),

[3] G Bilbro, R Mann, TK Miller, WE Snyder, DE Van den, *Optimization by Mean Field Annealing*, NIPS 1989

[4] GL Bilbro, WE Snyder & RC Mann, *Mean-field approximation minimizes relative entropy*, Journal of Optical Society of America A, Vol 8, No2, Feb. 1991

[5] S. T. Dumais, J. Platt, D. Heckerman and M. Sahami, *Inductive learning algorithms and representations for text categorization* In Proceedings of ACM-CIKM98, Nov. 1998, pp. 148-155.

[6] O. Chapelle & A. Zien, *Semi-Supervised Classification by Low Density Separation*, AI & Statistics

[7] R. Collobert, J. Weston, and L. Bottou,*Trading Convexity for Scalability*, NIPS 2005

[8] A. Frommer & P. Maas, *Fast CG-based methods for Tikhonov-Phillips regularization.*, SIAM Journal of Scientific Computing, 20(5), 1831-1850, 1999

[9] G. Fung & O. Mangasarian , *Semi-Supervised Support Vector Machines for Unlabeled Data Classification* Optimization Methods and Software 15, 2001, 29-44

[10] T. Joachims, *Transductive Inference for Text Classification using Support Vector Machines*, ICML 1999. HTTP://JOACHIMS.SVMLIGHT.ORG

[11] Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. *RCV1: A New Benchmark Collection for Text Categorization Research.* Journal of Machine Learning Research, 5:361-397, 2004. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf.

[12] O. Mangasarian, *A Finite Newton Method for Classification*,Optimization Methods and Software, 17:913-929, 2002

[13] A. McCallum, *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.* http://www.cs.cmu.edu/∼mccallum/bow. 1996.

[14] C. Peterson & B. Soderberg, *Neural Optimization*, Handobook of Brain theory and Neural Networks,M. A. Arbib (Ed.),Bradford Books, MIT Press 2002.

[15] S. S. Keerthi & D. DeCoste, *A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs*, JMLR 6(Mar):341–361, 2005. (Patent Pending).

[16] V. Sindhwani, P. Niyogi, & M. Belkin, *Beyond the Point Cloud: from Transductive to Semi-supervised Learning*, Proc. of International Conference on Machine Learning, 2005

[17] V. Sindhwani, *Kernel Machines for Semi-supervised Learning*, Masters Thesis, University of Chicago, 2004

[18] S. Chakrabarty, *Mining the Web: Analysis of Hypertext and Semi Structured Data*, Morgan Kaufmann, 2002

[19] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

Table 1: L_2-*SVM-MFN*

Problem	Given l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ where $x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$ and a cost for each example $\{c_i\}_{i=1}^{l}$, Solve: $$w^\star = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \quad \frac{1}{2}\sum_{i=1}^{l} c_i \max\left[0, 1 - y_i\ (w^T x_i)\right]^2 + \frac{\lambda}{2}\|w\|^2$$
Define	$X = [x_1 \dots x_l]^T \in \mathbb{R}^{l \times d}, \quad Y = [y_1 \dots y_l]^T \in \mathbb{R}^{l \times 1}$ $C \in \mathbb{R}^{l \times l}$: a diagonal matrix with $C_{ii} = c_i$ $w \in \mathbb{R}^d$: (a guess for the solution) If a guess is available, it is also convenient to pass: $o = Xw \qquad \jmath = \{i : y_i o_i < 1\} \qquad \jmath^c = \{i : i \in \jmath\}$
Inputs	X, Y, C, λ and $w, o, \jmath, \jmath^c$ (if available)
Initialize	if w, o unavailable (or set as zero vectors) set $w = 0 \in \mathbb{R}^d,\ o = 0 \in \mathbb{R}^l,\ \epsilon = 10^{-2}$, cgitermax $= 10$, $\jmath = 1 \dots l,\ \jmath^c = \phi$ if $w, o, \jmath, \jmath^c$ are available, set $\epsilon = 10^{-6}$, cgitermax $= 10000$ $\tau = 10^{-6}$ iter $= 0$ itermax $= 50$

Iterate	while (iter < itermax)
	iter=iter+1
	
	$(\bar{w}, \bar{o}_\jmath, \text{opt}) = \text{CGLS}(X_\jmath, Y_\jmath, C_\jmath, w, o_\jmath, \epsilon, \text{cgitermax})$ (see table 2)
	$\bar{o}_{\jmath^c} = X_{\jmath^c}\bar{w}$
	If cgitermax=10 reset cgitermax = 10000
	if $(\text{opt} = 1, \forall\ i \in \jmath\ \ y_i\bar{o}_i \leq 1 + \tau \ \ ,\ \ \forall\ i \in \jmath^c\ \ y_i\bar{o}_i \geq 1 - \tau)$
	If $\epsilon = 10^{-2}$ reset $\epsilon = 10^{-6}$ and
	continue the while loop iterations.
	Else set $w = \bar{w} \qquad o = \bar{o}$
	and exit the while loop.
	end if
	
	$\delta = \text{LINE-SEARCH}(w, \bar{w}, o, \bar{o}, Y, C)$ (see table 3)
	
	$w = w + \delta(\bar{w} - w)$
	$o = o + \delta(\bar{o} - o)$
	$\jmath = \{i \in 1 \dots l : y_i o_i < 1\} \qquad \jmath^c = \{i \in 1 \dots l : i \notin \jmath\}$
	end while

Outputs	$w, o, \jmath, \jmath^c$

Table 2: *CGLS*

Problem	Given l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ where $x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$ and a cost for each example $\{c_i\}_{i=1}^{l}$, Solve: $\beta^\star = \underset{\beta \in \mathbb{R}^d}{\text{argmin}} \quad \frac{1}{2}\sum_{i=1}^{l} c_i \left[y_i - \beta^T x_i\right]^2 + \frac{\lambda}{2}\|\beta\|^2$ Equivalently Solve: $\left[\lambda I + X^T C X\right] \beta^\star = X^T C Y$
Define	$X = [x_1 \dots x_l]^T \in \mathbb{R}^{l \times d}, \quad Y = [y_1 \dots y_l]^T \in \mathbb{R}^{l \times 1}$ $C \in \mathbb{R}^{l \times l}$: a diagonal matrix with $C_{ii} = c_i$ $\beta \in \mathbb{R}^d$: a guess for the solution (set $\beta = 0 \in \mathbb{R}^d$ if unavailable) $o = X\beta$
Inputs	$X, Y, C, \lambda, \beta, o, \epsilon$, cgitermax
Initialization	$z = C(Y - o) \quad r = X^T z - \lambda\beta \quad p = r \quad \omega_1 = \|r\|^2$ cgiter $= 0$ optimality $= 0$

Iterate	while (cgiter $<$ cgitermax)
	cgiter=cgiter $+ 1$
	$q = Xp$
	$\gamma = \frac{\omega_1}{\lambda\|p\|^2 + q^T C q}$
	$\beta = \beta + \gamma p$
	$o = o + \gamma q$
	$z = z - \gamma C q$
	$\omega_2 = \omega_1$
	$r = X^T z - \lambda\beta$
	$\omega_1 = \|r\|^2$
	if ($\omega_1 < \epsilon^2 \|z\|^2$)
	Set optimality $= 1$ and Exit while loop.
	end if
	$\omega = \frac{\omega_1}{\omega_2}$
	$p = r + \omega p$
	end while

Outputs	β, o, optimality

Table 3: *LINE-SEARCH*

Inputs	$w, \bar{w}, o, \bar{o}, Y, C$ as defined in Table 1
Initialize	$\jmath = \{i : y_i o_i < 1\}$ $L = \lambda w^T(\bar{w} - w) + \sum_{i \in \jmath} C_{ii}(o_i - y_i)(\bar{o}_i - o_i)$ $R = \lambda \bar{w}^T(\bar{w} - w) + \sum_{i \in \jmath} C_{ii}(\bar{o}_i - y_i)(\bar{o}_i - o_i)$ Define $\delta_i = \frac{(y_i - o_i)}{\bar{o}_i - o_i}$ for all i $\Delta_1 = \{\delta_i : i \in \jmath, \quad y_i(\bar{o}_i - o_i) > 0\}$ $\Delta_2 = \{\delta_i : i \notin \jmath, \quad y_i(\bar{o}_i - o_i) < 0\}$ $\Delta = \Delta_1 \cup \Delta_2$ $j = 0$ Reorder indices so that $\delta_i \in \Delta$ are sorted in non-decreasing order $\delta_{i_1}, \delta_{i_2} \ldots$.

Iterate	for $j = 1, 2 \ldots$
	    $\delta' = L + \delta_{i_j}(R - L)$
	
	    if $(\delta' \geq 0)$
	        Exit for loop
	    end if
	
	    Set $s = -1$ if $\delta_{i_j} \in \Delta_1$ or $s = 1$ if $\delta_{i_j} \in \Delta_2$
	    $L = L + sC_{i_j i_j}(o_{i_j} - y_{i_j})(\bar{o}_{i_j} - o_{i_j})$
	    $R = R + sC_{i_j i_j}(\bar{o}_{i_j} - y_{i_j})(\bar{o}_{i_j} - o_{i_j})$
	end for

Output	$\delta^\star = \frac{-L}{R - L}$

Table 4: *Algorithm: Transductive L_2-SVM-MFN* Solves Eqn. 7

Problem	Given l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ where $x_i \in \mathbb{R}^d, y_i \in \{-1,+1\}$ and u unlabeled examples $\{x_j{}'\}_{j=1}^{u}$, Solve problem in Eqn. 7.
Define	$X = [x_1 \ldots x_l]^T \in \mathbb{R}^{l\times d}$ $\quad Y = [y_1 \ldots y_l]^T \in \mathbb{R}^{l\times 1}$ $X' = [x'_1 \ldots x'_u]^T \in \mathbb{R}^{l\times u}$ $o = Xw \quad o' = X'w$
Inputs	$X, Y, X', \lambda, \lambda'$ r, S (maximum number of label pairs to switch, default S=1)
Initialization	$C \in \mathbb{R}^{l\times l}$: a diagonal matrix with $C_{ii} = \frac{1}{l}$ R=2 w_0= L_2-SVM-MFN(X, Y, C) Compute $o' = X'w_0$. Assign positive and negative labels to the unlabeled data in the ratio $r : (1-r)$ respectively by thresholding o'. Put these labels in a vector Y'. Set $\hat{\lambda}' = 10^{-5}$ $\quad$ Define: $\hat{X} = \begin{pmatrix} X \\ X' \end{pmatrix}$ $\quad \hat{Y} = \begin{pmatrix} Y \\ Y' \end{pmatrix}$ Define $\hat{C} \in \mathbb{R}^{(l+u)\times(l+u)}$: a diagonal matrix with: $\hat{C}_{ii} = \frac{1}{l}$ $(1 \le i \le l)$ $\quad \hat{C}_{ii} = \frac{\hat{\lambda}'}{u}$ $(l+1 \le i \le l+u)$ Set $w = 0 \in \mathbb{R}^d$ $\quad o = 0 \in \mathbb{R}^l$ $\quad o' \in \mathbb{R}^u$ $\quad \jmath = 1 \ldots (l+u)$ $\quad \jmath^c = \phi$
Iterate (Loop 1)	while $\hat{\lambda}' < \lambda'$
Re-training 1	$(w, [o\ o'], \jmath, \jmath^c)$= L_2-SVM-MFN$(\hat{X}, \hat{Y}, \hat{C}, w, [o\ o'], \jmath, \jmath^c)$
Iterate (Loop 2)	while ($\exists$ s index pairs $(i_k, j_k)_{k=1}^{s} : 1 \le i_k, j_k \le u$ with $s \le S$ such that: $Y'_{i_k} = +1$, $Y'_{j_k} = -1$, $o'_{i_k} < 1$, $-o'_{j_k} < 1$, $o'_{i_k} < o'_{j_k}$)
Switch Labels	$Y'_{i_k} = -1$ $\quad Y'_{j_k} = +1$ $\quad$ for $k = 1, 2 \ldots s$ $\hat{Y} = \begin{pmatrix} Y \\ Y' \end{pmatrix}$
Re-training 2	$(w, [o\ o'], \jmath, \jmath^c)$= L_2-SVM-MFN$(\hat{X}, \hat{Y}, \hat{C}, w, [o\ o'], \jmath, \jmath^c)$ end while (loop 2)
Increase $\hat{\lambda}'$	$\hat{\lambda}' = R\hat{\lambda}'$ $\hat{C}_{ii} = \frac{1}{l}$ $(1 \le i \le l)$ $\quad \hat{C}_{ii} = \frac{\hat{\lambda}'}{u}$ $(l+1 \le i \le l+u)$ end while (loop 1)
Output	w

Table 5: *Algorithm: Mean Field Annealing L_2-SVM-MFN.* Solves Eqn. 10,8,9

Problem	Given l labeled examples $\{x_i, y_i\}_{i=1}^{l}$ where $x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$ and u unlabeled examples $\{x'_j\}_{j=1}^{u}$, Solve for $w^\star$ in Eqns 10,8,9.
Define	$X = [x_1 \dots x_l]^T \in \mathbb{R}^{l \times d}$ $\quad Y = [y_1 \dots y_l]^T \in \mathbb{R}^{l \times 1}$ $X' = [x'_1 \dots x'_u]^T \in \mathbb{R}^{l \times u}$
Inputs	X, Y, X', λ, λ', r
Initialization	$T = 10 \quad R = 1.5 \quad \epsilon = 10^{-6}$ iter1=0 itermax1 = 30 itermax = 100 $\tilde{X} = [X^T \quad X'^T]^T$ $p = [r \dots r]^T \in \mathbb{R}^u$ $h = H(p)$ (Eqn. 16) $(w, [o\ o'], \jmath_l, \jmath_1, \jmath_2) =$ OPTIMIZE-W$(X, Y, p, \lambda, \lambda')$ (Table 7) $F = J(w)$ (Eqn. 18) $F_{\min} = F \quad w_{\min} = w \quad o'_{\min} = o'$

Loop 1	while (iter1 $<$ itermax1) AND ($h > \epsilon$)
	    iter1 = iter1 + 1 iter2 = 0
	    kl=1

Loop 2	while (iter2 $<$ itermax2) AND (kl $> \epsilon$)
	        iter2 = iter2 +1
	        $q = p$
	
	        $p =$ OPTIMIZE-P(o', λ', T, r) (Table 6)
	        $(w, [o\ o'], \jmath_l, \jmath_1, \jmath_2) =$
	          OPTIMIZE-W$(X, Y, p, \lambda, \lambda', w, [o\ o'], \jmath_l, \jmath_1, \jmath_2)$ (Table 7)
	
	        kl = $KL(p, q)$ (Eqn. 16)
	        $F = J(w)$ (Eqn. 18)
	        if $F < F_{\min}$
	            $F_{\min} = F \quad w_{\min} = w \quad o'_{\min} = o'$
	        end if
	    end while (loop 2)
	    $h = H(p)$
	    $T = T/R$
	end while (loop 1)

Output	$w_{\min}$

Table 6: OPTIMIZE-P Subroutine for mean field annealing to optimizing p. See Table 5

Inputs	o', λ', T, r Compute $g = [g_1 \ldots g_u]$ where $g_j = \lambda' \left(\max \left[0, 1 - o'_j\right]^2 - \max \left[0, 1 + o'_j\right]^2 \right)$
Initialize	$\epsilon = 10^{-10}$ iter $= 0$ maxiter $= 500$ $\nu_- = \min(g_1 \ldots g_u) - T \log \frac{1-r}{r}$ $\nu_+ = \max(g_1 \ldots g_u) - T \log \frac{1-r}{r}$ $\nu = (\nu_+ + \nu_-)/2$ Initial guess $s = [e^{-\frac{(g_1-\nu)}{T}} \ldots\ldots e^{-\frac{(g_u-\nu)}{T}}]$ $B(\nu) = \frac{1}{u} \sum_{i=1}^{u} \frac{1}{1+s_i} - r$ $B'(\nu) = \frac{1}{Tu} \sum_{i=1}^{u} \frac{s_i}{(1+s_i)^2}$ (if $s_i \to \infty$, i.e larger than some upper limit, set corresponding term to 0)

Iterate	while ($\lvert B(\nu)\rvert > \epsilon$) AND (iter $<$ maxiter)
	iter=iter+1
	if $\lvert B'(\nu)\rvert > 0$
	$\hat{\nu} = \nu - \frac{B(\nu)}{B'(\nu)}$
	end if
	if ($\hat{\nu} < \nu_-$) OR ($\hat{\nu} > \nu_+$) OR $B'(\nu) = 0$

Bisection	$\nu = \frac{\nu_- + \nu_+}{2}$ else
Newton-Raphson	$\nu = \hat{\nu}$ end if
Update	$s = [e^{-\frac{(g_1-\nu)}{T}} \ldots\ldots e^{-\frac{(g_u-\nu)}{T}}]$ $B(\nu) = \frac{1}{u} \sum_{i=1}^{u} \frac{1}{1+s_i} - r$ $B'(\nu) = \frac{1}{Tu} \sum_{i=1}^{u} \frac{s_i}{(1+s_i)^2}$ (if $s_i \to \infty$, i.e larger than some upper limit, set corresponding term to 0) if $B(\nu) < 0$ set $\nu_- = \nu$ else $\nu_+ = \nu$ if $\lvert \nu_+ - \nu_- \rvert < \epsilon$ exit while loop end while
Output	$p = [\frac{1}{1+s_1} \ldots \frac{1}{1+s_u}]$

Table 7: OPTIMIZE-W (specialized L_2-SVM-MFN routine for mean field annealing. See Table 5

Inputs	$\tilde{X}, Y, p, \lambda, \lambda'$ and $w, [o\ o'], \jmath_l, \jmath_1, \jmath_2$ (if available)
Initialize	if w unavailable (or set as zero vectors) set $w = 0 \in \mathbb{R}^d$, $[o\ o'] = 0 \in \mathbb{R}^{l+u}$, $\epsilon = 10^{-2}$, cgitermax $= 10$, $\jmath_l = 1 \ldots l$, $\jmath_2 = 1 \ldots u$, $\jmath_1 = \jmath_l^c = \phi$ if $w, [o\ o'], \jmath_l, \jmath_1, \jmath_2$ are available, set $\epsilon = 10^{-6}$, cgitermax $= 10000$, $\jmath_l^c = \{i \in 1 \ldots l : i \notin \jmath_l\}$ $\tau = 10^{-8}$ iter $= 0$ itermax $= 50$ Set $\hat{Y}, C$ according to Eqn. 11 Set $\tilde{Y}, \tilde{C}$ according to Eqn. 14

Iterate	while (iter $<$ itermax)
	iter=iter+1
	
	Define the active index set: $\jmath = \jmath_l \bigcup \{j\}_{j=l+1}^{l+u}$
	$(\bar{w}, [\bar{o}_{\jmath_l}\ \bar{o}'], \text{opt}) = \text{CGLS}\Big(X_\jmath, \tilde{Y}_\jmath, C_\jmath, w, [o_{\jmath_l}\ \bar{o}'], \epsilon, \text{cgitermax}\Big)$
	$\bar{o}_{\jmath_l^c} = X_{\jmath_l^c} \bar{w}$
	
	if cgitermax=10 reset cgitermax $= 10000$
	
	if opt $= 1$, $\forall\ i \in \jmath_l\ \ y_i \bar{o}_i \leq 1 + \tau$, $\forall\ i \in \jmath_l^c\ \ y_i \bar{o}_i \geq 1 - \tau$
	$\forall\ j \in \jmath_l\ \ \lvert o'_j \rvert >= 1 - \tau\ \ \forall\ j \in \jmath_2\ \ \lvert o'_j \rvert < 1 + \tau$
	If $\epsilon = 10^{-2}$ reset $\epsilon = 10^{-6}$ and
	continue the while loop iterations.
	Else set $w = \bar{w}$ $o = \bar{o}$
	and exit the while loop.
	end if
	$\delta = \text{LINE-SEARCH}\Big(w, \bar{w}, [o\ o'\ o'], [\bar{o}\ \bar{o}'\ \bar{o}'], \hat{Y}, C\Big)$
	$w = w + \delta(\bar{w} - w)$
	$o = o + \delta(\bar{o} - o)$ $o' = o' + \delta(\bar{o}' - o')$
	$\jmath = \{i : y_i o_i < 1\}$ $\jmath_l^c = \{i : i \notin \jmath_l\}$
	$\jmath_1 = \{j \in 1 \ldots u : \lvert o'_j \rvert >= 1\}$ $\jmath_2 = \{j \in 1 \ldots u : \lvert o'_j \rvert < 1\}$
	Recompute $\tilde{Y}, \tilde{C}$ according to Eqn. 14
	end while

Output	$w, [o\ o'], \jmath_l, \jmath_1, \jmath_2$