Large Scale Semi-supervised Linear SVMs

Vikas Sindhwani
Department of Computer Science
University of Chicago
Chicago, IL 60637, USA
vikass@cs.uchicago.edu

S. Sathiya Keerthi
Yahoo! Research
Media Studios North
Burbank, CA 91504, USA
selvarak@yahoo-inc.com

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 classifiers 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 recently developed modified finite Newton techniques. Our contributions in this paper are as follows: (a) We provide an implementation of 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 Deterministic Annealing (DA) approach. This algorithm alleviates the problem of local minimum in the TSVM optimization procedure while also being computationally attractive. We conduct an empirical study on several document classification tasks which confirms the value of our methods in large scale semi-supervised settings.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Performance, Experimentation

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

Keywords

Support Vector Machines, Unlabeled data, Global optimization, Text Categorization

1. INTRODUCTION

Consider the following situation: In a single web-crawl, search engines like Yahoo! and Google index billions of documents. Only a very small fraction of these documents can possibly be hand-labeled by human editorial teams and assembled into topic directories. In information retrieval relevance feedback, a user labels a small number of documents returned by an initial query as being relevant or not. The remaining documents form a massive collection of unlabeled data. 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.

In this paper, we propose extensions of linear Support Vector Machines (SVMs) 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, for example, for document classification with Linear SVMs in numerous studies.

Our methods are motivated by the intuition of margin maximization for semi-supervised SVMs [13, 5, 1, 6, 3, 4]. 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 [5] is considered state-of-the-art in text categorization, even in the face of increasing recent competition.

[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.

We highlight the main contributions of this paper.

(1) We outline an implementation for a variant of Transductive SVM [5] designed for linear semi-supervised classification on large, sparse datasets. As compared to currently used dual techniques (e.g in the SVM$^{\text{light}}$ implementation of TSVM), our method effectively 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 novel algorithm for semi-supervised SVMs inspired from Deterministic Annealing (DA) techniques. This approach generates a family of objective functions whose non-convexity is controlled by an annealing parameter. The global minimizer is parametrically tracked in this family. This approach alleviates the problem of local minima in the TSVM optimization procedure which results in better solutions on some problems. A computationally attractive training algorithm is presented that involves a sequence of alternating convex optimizations.

(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 [8] for fast training of linear SVMs is a key subroutine for our algorithms.

This paper is arranged as follows. In section 2 we describe the l_2-SVM-MFN algorithm and present its semi-supervised extensions in section 3. Experimental results are reported in section 4. Section 5 contains some concluding comments. A more detailed version of this paper is available at [11].

2. MODIFIED FINITE NEWTON LINEAR L$_2$-SVM

The modified finite Newton l_2-SVM method [8] (l_2-SVM-MFN) is a recently developed training algorithm for Linear SVMs that is ideally suited to sparse datasets with large number of examples and possibly large number of features.

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 documents) and the labels $y_i \in \{+1, -1\}$, l_2-SVM-MFN provides an efficient primal solution to the following SVM optimization problem:

$$w^\star = \operatorname*{argmin}_{w\in\mathbb{R}^d} \frac{1}{2}\sum_{i=1}^{l} l_2(y_i w^T x_i) + \frac{\lambda}{2}\|w\|^2 \qquad (1)$$

where l_2 is the l_2-SVM loss given by $l_2(z) = \max(0, 1-z)^2$, λ is a real-valued regularization parameter and the final classifier is given by $\operatorname{sign}(w^{\star T} x)$.

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 [8].

We consider a version of l_2-SVM-MFN where a weighted quadratic soft margin loss function is used.

$$\min_w f(w) = \frac{1}{2}\sum_{i\in\jmath(w)} c_i l_2(y_i w^T x_i) + \frac{\lambda}{2}\|w\|^2 \qquad (2)$$

Here we have rewritten Eqn. 1 in terms of the support vector 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 [8], 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:

$$\nabla\ f(w) = \lambda\ w + X_{\jmath(w)}^T C_{\jmath(w)} \left[X_{\jmath(w)} w - Y_{\jmath(w)}\right]$$

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:

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

where the step size $\delta^k \in \mathbb{R}$, and the Newton direction $n^k \in \mathbb{R}^d$ is given by: $n^k = -[\nabla^2\ f(w^k)]^{-1}\nabla\ f(w^k)$. Here, $\nabla\ f(w^k)$ is the gradient vector and $\nabla^2\ f(w^k)$ 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] Thus a generalized definition of the Hessian matrix is used. The modified finite Newton procedure [8] 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 $\jmath(w^k)$:

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

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:

$$\delta^k = \operatorname*{argmin}_{\delta\geq 0} \phi(\delta) = f\left(w^k + \delta(\bar{w}^k - w^k)\right) \qquad (5)$$

The modified finite Newton procedure has the property of finite convergence to the optimal solution. The key features

[2] In the neighborhood of such a w, the index i leaves or enters $\jmath(w)$. However, at w, $y_i w^T x_i = 1$. So f is continuously differentiable inspite of these index jumps.

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 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 Below, we briefly discuss the CGLS and Line search procedures. We refer the reader to [8] for full details.

2.1 CGLS

CGLS is a special conjugate-gradient solver that is designed to solve, in a numerically robust way, large, sparse, weighted regularized least squares problems such as the one in Eqn. 4. Starting with a guess solution, several specialized conjugate-gradient iterations are applied to get $\bar{w}^k$ that solves Eqn. 4. The major expense in each iteration consists of two operations of the form $X_{\jmath(w^k)}p$ and $X^T_{\jmath(w^k)}q$. If there are n_0 non-zero elements in the data matrix, these involve $O(n_0)$ cost. It is worth noting that, as a subroutine of l$_2$-SVM-MFN, CGLS is typically called on a small subset, $X_{\jmath(w^k)}$ of the full data set. To compute the *exact solution* of Eqn. 4, r iterations are needed, where r is the rank of $X_{\jmath(w^k)}$. But, in practice, such an exact solution is unnecessary. CGLS uses an effective stopping criterion for early termination. 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 $X_{\jmath(w^k)}$ and is typically found to be very small relative to the dimensions of $X_{\jmath(w^k)}$ (number of examples and features). The memory requirements of CGLS are also minimal: only five vectors need to be maintained, including the outputs over the currently active set of data points.

Finally, an important feature of CGLS is worth emphasizing. Suppose the solution w of a regularized least squares problem is available, i.e the linear system in Eqn. 4 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 w as the initial guess. This is 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 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 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^k,$\bar{w}^k$ in some iteration of l$_2$-SVM-MFN, the line search step requires us to solve Eqn. 5. The one-dimensional function $\phi(\delta)$ is the restriction of the objective function f on the ray from w^k onto $\bar{w}^k$. Hence, like f, $\phi(\delta)$ is also a continuously differentiable, strictly convex, piecewise quadratic function with a unique minimizer. ϕ' is a continuous piecewise linear function whose root, δ^k, can be easily found by sorting the break points where its slope changes and then performing a sequential search on that sorted list. The cost of this operation is negligible compared to the cost of the CGLS iterations.

l$_2$-SVM-MFN alternates between calls to CGLS and line searches. Its computational complexity 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. For example, on a text classification experiment involving 198788 examples and 252472 features, $t_{mfn} = 11$ and $\bar{t}_{cgls} = 102$. Therefore, the complexity of l$_2$-SVM-MFN is 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 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 [5, 1, 6]. Since [5] 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[3]:

$$\min_{w,\{y'_j\}_{j=1}^u} \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l(y_i\ w^T x_i) + \frac{\lambda'}{2u}\sum_{j=1}^{u} l(y'_j\ w^T x'_j)$$

$$\text{subject to: } \frac{1}{u}\sum_{j=1}^{u}\max[0, \mathrm{sign}(w^T x'_j)] = r$$

where the hinge loss function, $l(z) = l_1(z) = max(0, 1-z)$ is normally used. The labels on the unlabeled data, $y'_1 \dots y'_u$, 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 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. For example, if the data has distinct clusters with a large margin, but the cluster assumption does not

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

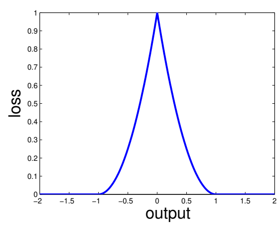
hold, then λ' can be set to 0 and the standard SVM is retrieved. If there is enough labeled data, λ, λ' can be tuned by cross-validation. An initial estimate of r can be made from the fraction of labeled examples that belong to the positive class and subsequent fine tuning can be done based on validation performance.

This optimization is implemented in [5] 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 $\mathrm{SVM}^{\mathrm{light}}$ software. There, the training of SVMs in the inner loops of TSVM uses dual decomposition techniques. As shown by experiments in [8], in sparse, linear settings one can obtain significant speed improvements with l_2-SVM-MFN over $\mathrm{SVM}^{\mathrm{light}}$. Thus, by implementing TSVM with l_2-SVM-MFN, we expect similar improvements for semi-supervised learning on large, sparse datasets. Note that l_2-SVM-MFN can also be used to speedup other TSVM formulations e.g [4] in such cases. 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 consider the TSVM objective function but with the L_2-SVM loss function, $l = l_2$.

Figure 1: l_2 loss function for TSVM



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

$$\min[l_2(w^T x), l_2(-w^T x)] = \max[0, 1 - |w^T x|]^2$$

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 subsection, we will outline a deterministic annealing procedure that can overcome this problem.

The TSVM algorithm with l_2-SVM-MFN closely follows the presentation in [5]. 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 certain factor 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 negative temporary labels such that switching these labels would decrease the objective function. l_2-SVM-MFN is then retrained with the switched labels.

3.1.2 Multiple Switching

The TSVM algorithm presented in [5] involves switching a single pair of labels at a time. 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 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 [5] we can show that Transductive l_2-SVM-MFN with multiple-pair switching converges in a finite number of steps.

We are unaware of any prior work that suggests and evaluates this simple multiple-pair switching heuristic. Our experimental results in section 4 establish that this heuristic is remarkably effective in speeding up TSVM training while maintaining generalization performance.

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. The complexity of Transductive L_2-TSVM-MFN is $O(n_{switches}\bar{t}_{mfn}\bar{t}_{cgls}n_0)$, where $n_{switches}$ is the number of label switches. 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 Deterministic 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 [3]. We now present a new algorithm based on deterministic annealing that can potentially overcome this problem while also being computationally very attractive for large scale applications. Deterministic Annealing [2, 9] (DA) 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:

$$w^\star = \underset{w, \{\mu_j\}_{j=1}^{u}}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(w^T x_i)$$
$$+\frac{\lambda'}{2u}\sum_{j=1}^{u}\Big(\mu_j l_2(w^T x_j') + (1-\mu_j) l_2(-w^T x_j')\Big)$$

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 [4] 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:

$$w_T^\star = \underset{w, \{p_j\}_{j=1}^{u}}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(y_i w^T x_i)$$
$$+\frac{\lambda'}{2u}\sum_{j=1}^{u}\Big(p_j l_2(w^T x_j') + (1-p_j) l_2(-w^T x_j')\Big)$$
$$+\frac{T}{2u}\sum_{j=1}^{u}[p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)] \qquad (6)$$

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

$$\frac{1}{u}\sum_{j=1}^{u} p_j = r \qquad (7)$$

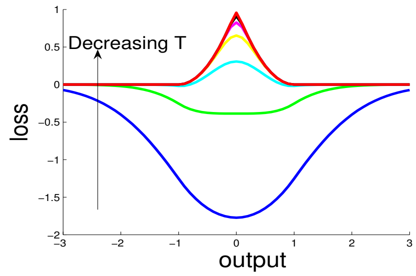
where r is the fraction of the number of unlabeled examples belonging to the positive class. As in TSVM, r is treated as a user-provided parameter. It may also be estimated from the labeled examples.

The solution to the optimization problem above is tracked as the temperature parameter T is lowered to 0. 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. 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 minimizer is then slowly tracked as the temperature is lowered towards zero.

[4] 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: DA loss function parameterized by T.



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. 6,7 is optimized by alternating the minimization over w and $p = [p_1 \ldots p_u]$ respectively. Fixing p, the optimization over w is done by l_2-SVM-MFN with seeding. 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 some details.

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

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

$$C = diag\left[\overbrace{\frac{1}{l} ... \frac{1}{l}}^{l}, \ \overbrace{\frac{\lambda' \, p_1}{u} ... \frac{\lambda' \, p_u}{u}}^{u} \ \overbrace{\frac{\lambda'(1-p_1)}{u} ... \frac{\lambda'(1-p_u)}{u}}^{u}\right] \qquad (8)$$

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.

3.2.2 Optimizing p

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

$$\mathcal{L} = \frac{\lambda'}{2u}\sum_{j=1}^{u}\Big(p_j l_2(w^T x_j') + (1-p_j) l_2(-w^T x_j')\Big) +$$
$$\frac{T}{2u}\sum_{j=1}^{u}(p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)) - \nu\left[\frac{1}{u}\sum_{j=1}^{u} p_j - r\right]$$

Solving $\partial\mathcal{L}/\partial p_j = 0$, we get:

$$p_j = \frac{1}{1+e^{\frac{g_j - 2\nu}{T}}} \qquad (9)$$

where $g_j = \lambda'[l_2(w^T x'_j) - l_2(-w^T x'_j)]$. Substituting this expression in the balance constraint in Eqn. 7, 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 root is computed by using a hybrid combination of Newton-Raphson iterations and the bisection method together with a carefully set initial value.

3.2.3 *Stopping Criteria*

For a fixed T, the alternate minimization of w and p proceeds until some stopping criterion is satisfied. A natural criterion is the mean Kullback-Liebler divergence (relative entropy) $KL(p,q)$ 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 is:

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

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, i.e.,

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

The TSVM objective function,

$$\frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(y_i \ (w^T x_i) + \frac{\lambda'}{2u}\sum_{j=1}^{u}\max\left[0, 1-|w^T x'_j|\right]^2$$

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

4. EMPIRICAL STUDY

Semi-supervised learning experiments were conducted to test these algorithms on six text binary classification problems. These are listed in Table 1.

The aut-avn and real-sim binary classification datasets come from a collection of UseNet articles[5] from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation. The ccat and gcat data sets pose the problem of separating corporate and government related articles respectively; these are the top-level categories in the RCV1 training data set [7]. 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 33-36 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 [3]). The results below are averaged over 10 random stratified splits of training (labeled and unlabeled) and test sets. The performance of SVM, DA and TSVM is studied as a function of the amount of labeled data in the training set. Since the two classes are fairly well represented in these datasets, we report error rates, but expect our conclusions to also hold for other performance measures such as F-measure. We use a default value of $\lambda' = 1$ for all datasets except[6] for aut-avn and ccat where $\lambda' = 10$. The default value of $\lambda = 0.001$ was used for all datasets.

For datasets and software implementation, we point the reader to [12].

[5] Available at: http://www.cs.umass.edu/∼mccallum/data/sraa.tar.gz

Table 1: Two-class datasets. d : data dimensionality, $\bar{n}_0$: average sparsity, $l+u$: number of labeled and unlabeled examples, t : number of test examples, r : positive class ratio.

Dataset	d	$\bar{n}_0$	$l+u$	t	r
aut-avn	20707	51.32	35588	35587	0.65
real-sim	20958	51.32	36155	36154	0.31
ccat	47236	75.93	17332	5787	0.46
gcat	47236	75.93	17332	5787	0.30
33-36	59072	26.56	41346	41346	0.49
pcmac	7511	54.58	1460	486	0.51

Minimization of Objective Function

We first examine the effectiveness of TSVM and DA in optimizing the TSVM objective function. In Table 2, we report the minimum value of the objective function achieved by TSVM and DA with respect to varying number of labels. As compared to TSVM, we see that DA performs significantly better optimization on aut-avn, ccat, and pcmac datasets and slightly better optimization on the other datasets.

Table 2: TSVM,DA: Minimum value of objective function achieved with respect to number of labels. Statistically significantly better minimizations are shown in bold.

aut-avn	l=45	89	178	356	712	1424
TSVM	0.692	0.692	0.698	0.709	0.715	0.722
DA	**0.669**	**0.663**	**0.670**	**0.679**	**0.687**	**0.696**
real-sim	l=46	91	181	362	724	1447
TSVM	0.168	0.181	0.194	0.212	0.235	0.257
DA	0.166	0.176	0.190	0.210	0.234	0.256
ccat	l=44	87	174	348	695	1389
TSVM	0.583	0.602	0.637	0.663	0.682	0.704
DA	0.575	0.594	**0.603**	**0.619**	**0.639**	**0.662**
gcat	l=44	87	174	348	695	1389
TSVM	0.135	0.140	0.147	0.159	0.177	0.193
DA	**0.134**	0.138	0.146	0.158	0.176	0.192
33-36	l=52	104	207	414	827	1654
TSVM	0.198	0.210	0.233	0.256	0.286	0.316
DA	0.196	0.208	0.230	0.255	0.285	0.315
pcmac	l=37	73	110	146	183	220
TSVM	0.136	0.137	0.138	0.139	0.141	0.142
DA	**0.122**	**0.125**	**0.128**	**0.131**	**0.134**	**0.136**

Generalization Performance

In Table 3 we report error rates over unseen test examples for SVM (which only uses labeled examples), TSVM and DA with respect to varying amounts of labeled data. The following observations are made.

(1) Comparing the performance of SVM against the semi-supervised algorithms, the benefit of unlabeled data for boost-

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

Table 3: SVM, TSVM, DA: test error rate comparison. TSVM (S=max) are results for multiple (maximum possible) switching TSVM. Bold numbers indicate a significant performance difference between TSVM and DA.

aut-avn	l= 45	89	178	356	712	1424
SVM	31.8	24.0	15.6	10.3	7.6	5.8
DA	7.0	**4.6**	4.5	**3.9**	**3.7**	**3.6**
TSVM	6.2	5.6	5.2	4.7	4.3	3.9
TSVM (S=max)	6.1	5.6	5.2	4.7	4.3	3.9
real-sim	l= 46	91	181	362	724	1447
SVM	28.7	24.9	18.2	12.8	9.8	7.5
DA	15.4	13.5	11.6	10.0	8.4	7.2
TSVM	16.3	12.5	10.2	9.0	7.9	6.9
TSVM (S=max)	15.9	**12.4**	**10.2**	**8.9**	**7.9**	6.9
ccat	l=44	87	174	348	695	1389
SVM	23.9	18.6	14.6	11.9	10.0	8.7
DA	18.1	13.7	11.9	10.3	9.2	8.5
TSVM	20.3	13.9	11.7	10.2	9.1	8.3
TSVM (S=max)	20.2	14.0	11.7	10.2	9.1	8.3
gcat	l=44	87	174	348	695	1389
SVM	26.1	18.1	11.8	8.3	6.6	5.6
DA	6.0	5.9	5.8	5.7	5.4	5.2
TSVM	6.1	6.0	5.8	5.7	5.4	5.1
TSVM (S=max)	6.2	6.0	5.8	5.8	5.4	5.1
33-36	l= 52	104	207	414	827	1654
SVM	26.0	20.3	15.5	12.7	10.7	8.9
DA	21.5	17.4	13.3	11.4	9.8	8.6
TSVM	21.3	16.6	12.8	11.2	9.7	8.5
TSVM (S=max)	21.4	16.6	12.8	11.2	9.7	8.5
pcmac	l= 37	73	110	146	183	220
SVM	18.0	12.1	9.7	8.0	7.4	6.7
DA	**5.2**	**5.0**	**4.6**	4.7	4.4	4.3
TSVM	7.4	6.9	6.0	5.7	5.1	4.7
TSVM (S=max)	7.3	6.8	6.0	5.7	5.1	4.7

ing generalization performance is evident on all datasets. This is true even for moderate number of labels, though it is particularly striking towards the lower end.

(2) On aut-avn and pcmac, DA outperforms TSVM significantly. On ccat, DA performs a much better optimization (Table 2) but this does not translate into major error rate improvements. DA and TSVM are very closely matched on gcat and 33-36. On real-sim, TSVM and DA perform very similar optimization of the transduction objective function (Table 2), but appear to return very different solutions. The TSVM solution returns lower error rates as compared to DA on this dataset.

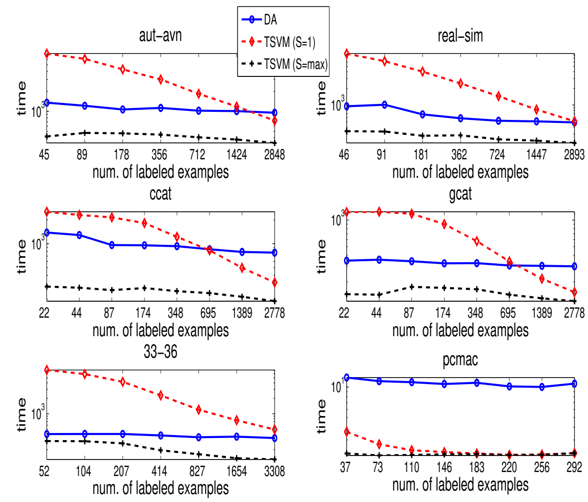
(3) On all datasets we found that multiple switching (see rows corresponding to TSVM (S=max) in Table 3) returned nearly identical performance as single switching. Since it saves significant computation time, our study establishes multiple switching as a valuable heuristic for training TSVM.

(4) These observations are also true for in-sample transductive performance. Both TSVM and DA are found to provide high quality extension to unseen test data.

Computational Timings

In Figure 3, we plot the average computation time for DA and TSVM with single and maximum switching. We make the following observations. The single switch TSVM is an order of magnitude slower than the maximum switching variant. DA is significantly faster than single switch TSVM but slower than TSVM with maximum switching.

Figure 3: Computation time with respect to number of labels for DA and Transductive l_2-SVM-MFN with single and multiple switches.



In Table 4, we compare our TSVM and DA implementations with SVM$^{\text{light}}$ at its default optimization settings on the first split with fewest labeled examples. These comparisons clearly demonstrate massive speedups with our methods. Note that the results presented in this section were obtained with a MATLAB implementation. We expect significantly faster computation times with a C or a fortran implementation, especially with parallel computation of matrix vector products[7].

Table 4: Speed comparisons (in seconds) with SVM$^{\text{light}}$. S=1 and S=max denotes our single and multiple maximum implementations.

Dataset	SVM$^{\text{light}}$	S=1	S=max	DA
aut-avn	101759	5849	390	1446
real-sim	498313	6244	373	1129
ccat	13540	2352	390	1185
gcat	243840	1267	358	159
33-36	48390	7406	309	393
pcmac	167	4	2	12

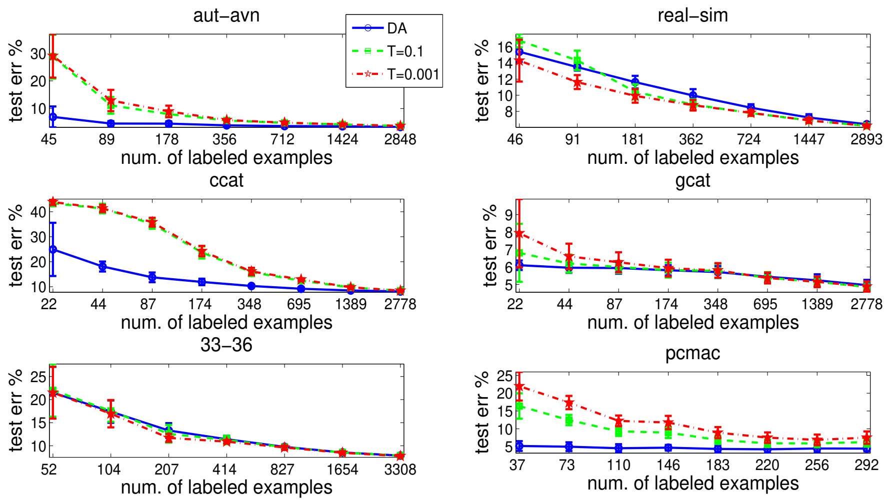
Importance of Annealing

To confirm the necessity of an annealing component (tracking the minimizer with respect to T) in the optimization, we compare DA with the alternating w,p optimization procedure where the temperature parameter is held fixed at $T = 0.1$ and $T = 0.001$. In Figure 4 we plot the error rates achieved with and without annealing. We see that annealing tends to provide 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

[7]preliminary experiments with a C++ implementation (to be made available at [12]) suggested about 5-fold further improvements in speed.

Figure 4: Error Rates achieved by DA and fixed temperature optimization with respect to number of labels.



local optima. The non-convex part of the TSVM objective function is gradually increased to a desired value. In this sense, λ' simultaneously plays the role of an annealing parameter and also provides control over 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 DA, however, we directly apply a framework for global optimization, and decouple annealing from the implementation of the cluster assumption. As our experiments show, this can lead to significantly better solutions on many problems.

5. CONCLUSION

In this paper we have proposed a family of primal SVM algorithms for large scale semi-supervised learning based on the finite Newton technique. Our methods significantly enhance the training speed of TSVM over existing methods such as SVM$^{\text{light}}$ and also include a new effective technique based on deterministic annealing. The new TSVM method with multiple switching is the fastest of all the algorithms considered, and also returns good generalization performance. The DA method is relatively slower but often gives the best accuracy. These algorithms can be very valuable in applied scenarios where sparse classification problems arise frequently, labeled data is scarce and plenty of unlabeled data is easily available. Even in situations where a good number of labeled examples are available, utilizing unlabeled data to obtain a semi-supervised solution using these algorithms can be worthwhile. For one thing, the semi-supervised solutions never lag behind purely supervised solutions in terms of performance. The presence of a mix of labeled and unlabeled data can provide added benefits such as reducing performance variability and stabilizing the linear classifier weights. Our algorithms can be extended to the non-linear setting [10], and may also be developed to handle clustering and one-class classification problems. These are subjects for future work.

6. REFERENCES

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

[2] G. Bilbro, R. Mann, T.K. Miller, W.E. Snyder and D.E. Van den, Optimization by Mean Field Annealing, NIPS 1989.

[3] O. Chapelle and A. Zien, Semi-Supervised Classification by Low Density Separation, AI & Statistics, Barbados, January 2005.

[4] R. Collobert, F. Sinz, J. Weston, and L. Bottou, Large Scale Transductive SVMs, (submitted) 2006.

[5] T. Joachims, Transductive Inference for Text Classification using Support Vector Machines, ICML 1998.

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

[7] D. Lewis, Y. Yang, T. Rose and F. Li, RCV1: A New Benchmark Collection for Text Categorization Research, *Journal of Machine Learning Research* 5:361-397, 2004.

[8] S. S. Keerthi and D. DeCoste, A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs, *Journal of Machine Learning Research* 6:341–361, 2005.

[9] C. Peterson and B. Soderberg, A new method for mapping optimization problems onto neural networks, International Journal of Neural Systems, 1(1):3–22, 1989.

[10] V. Sindhwani, S. S. Keerthi, and O. Chapelle, Deterministic Annealing for Semi-supervised Kernel Machines, ICML 2006.

[11] V. Sindhwani and S.S. Keerthi, Large Scale Semi-supervised Linear SVMs, Technical report, Yahoo research, 2006.

[12] `http://www.cs.uchicago.edu/~vikass/research.html`

[13] V. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998.