# 1 Newton Methods for Fast Solution of Semi-supervised Linear SVMs

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

**Sathiya Keerthi** keerthi@yahoo-inc.com
*Yahoo! Research*
*3333 Empire Avenue, Burbank, CA 91504, USA*

*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 chapter, 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 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 classification tasks which confirms the value of our methods in large scale semi-supervised settings. Our algorithms are implemented in* $\text{SVM}_{lin}$*, a public domain software package.*

## 1.1 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 chapter, 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 (Vapnik, 1998; Joachims, 1998; Bennett and Demirez, 1998; Fung and Mangasarian, 2001; Chapelle and Zien, 2005; Collobert et al., 2006). 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 (Joachims, 1998) is considered state-of-the-art in text categorization, even in the face of increasing recent competition.

We highlight the main contributions of our work.

1. We outline an implementation for a variant of Transductive SVM (Joachims, 1998) designed for linear semi-supervised classification on large, sparse datasets.

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.

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. This study clearly shows the utility of our tools for large scale problems. This scale that has not been explored in semi-supervised learning literature till now.

The modified finite Newton algorithm of Keerthi and DeCoste (2005) for fast training of linear SVMs is a key subroutine for our algorithms. In section 1.2 we describe this algorithm. Its semi-supervised extensions are presented in section 1.3 and 1.4. Experimental results are reported in section 1.5. Section 1.6 contains some concluding comments.

All the algorithms described in this chapter are implemented in a public domain software, $\text{SVM}_{\text{lin}}$ (see section 1.5) which can be used for fast training of linear SVMs for supervised and semi-supervised classification problems.

## 1.2 Modified Finite Newton Linear $l_2$-SVM

The modified finite Newton $l_2$-SVM method (Keerthi and DeCoste, 2005) (abbreviated $l_2$-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:

$$w^\star = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} l_2(y_i w^T x_i) + \frac{\lambda}{2} \|w\|^2 \tag{1.1}$$

where $l_2$ is the l$_2$-SVM loss given by $l_2(z) = \max(0, 1-z)^2$, $\lambda$ is a real-valued regularization parameter[2] and the final classifier is given by $\text{sign}(w^{\star T} x)$.

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. 1.1, 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 consider a version of l$_2$-SVM-MFN where a weighted quadratic soft margin loss function is used.

$$\min_w f(w) = \frac{1}{2} \sum_{i \in \jmath(w)} c_i l_2(y_i w^T x_i) + \frac{\lambda}{2} \|w\|^2 \tag{1.2}$$

Here we have rewritten Eqn. 1.1 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 Keerthi and DeCoste (2005), 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

2. $\lambda = 1/C$ where C is the standard SVM parameter.

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:

$$w^{k+1} = w^k + \delta^k \; n^k \tag{1.3}$$

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 $i$.[3] Thus a generalized definition of the Hessian matrix is used. The modified finite Newton procedure 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 $\jmath(w^k)$:

$$\left[\lambda I + X^T_{\jmath(w^k)} C_{\jmath(w^k)} X_{\jmath(w^k)}\right] \bar{w}^k = X^T_{\jmath(w^k)} C_{\jmath(w^k)} Y_{\jmath(w^k)} \tag{1.4}$$

where $I$ is the identity matrix. Once $\bar{w}^k$ is obtained, $w^{k+1}$ is obtained from Eqn. 1.3 by setting $w^{k+1} = w^k + \delta^k(\bar{w}^k - w^k)$ after performing an exact line search for $\delta^k$, i.e by exactly solving a one-dimensional minimization problem:

$$\delta^k = \underset{\delta \geq 0}{\operatorname{argmin}} \, \phi(\delta) = f\left(w^k + \delta(\bar{w}^k - w^k)\right) \tag{1.5}$$

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. 1.4 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

3. 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.

Line search procedures. We refer the reader to Keerthi and DeCoste (2005) for full details.

### 1.2.1 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. 1.4. Starting with a guess solution, several specialized conjugate-gradient iterations are applied to get $\bar{w}^k$ that solves Eqn. 1.4. The major expense in each iteration consists of two operations of the form $X_{j(w^k)}p$ and $X^T_{j(w^k)}q$. 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, $X_{j(w^k)}$ of the full data set. To compute the *exact solution* of Eqn. 1.4, $r$ iterations are needed, where $r$ is the rank of $X_{j(w^k)}$. But, in practice, such an exact solution is unnecessary. CGLS uses an effective stopping criterion based on gradient norm for early termination (involving a tolerance parameter $\epsilon$). The total cost of CGLS is $O(t_{cgls}n_0)$ where $t_{cgls}$ is the number of iterations, which depends on $\epsilon$ and the condition number of $X_{j(w^k)}$, and is typically found to be very small relative to the dimensions of $X_{j(w^k)}$ (number of examples and features). Apart from the storage of $X_{j(w^k)}$, 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. 1.4 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 $\lambda$ over a range of values, it is valuable to seed the solution for a particular $\lambda$ 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.

### 1.2.2 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. 1.5. 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. $\phi'$ is a continuous piecewise linear function whose root, $\delta^k$, 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.

### 1.2.3 Complexity

$l_2$-SVM-MFN alternates between calls to CGLS and line searches until the support vector set $\jmath(w_k)$ stabilizes upto a tolerance parameter $\tau$. 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. The number of CGLS iterations to reach a relative error of $\epsilon$ can be bounded in terms of $\epsilon$ and the condition number of the left-hand-side matrix in Eqn 1.4 (Bjork, 1996). Thus, the CGLS calls have linear complexity in the number of non-zeros in the data matrix.

In practice, $t_{mfn}, \bar{t}_{cgls}$ depends on the data set and the tolerances desired in the stopping criterion, but are typically very small. As an example of typical behavior: on a Reuters (Lewis et al., 2004) text classification problem (top level category CCAT versus rest) involving 804414 examples and 47236 features, $t_{mfn} = 7$ with a maximum of $t_{cgls} = 28$ CGLS iterations; on this dataset $l_2$-SVM-MFN converges in about 100 seconds on an Intel 3GHz, 2GB RAM machine[4]. The practical scaling of $l_2$-SVM-MFN is found to be linear in the number of non-zero entries in the data matrix (Keerthi and DeCoste, 2005).

### 1.2.4 Other Loss functions

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 Keerthi and DeCoste (2005).

We also note a recently proposed linear time training algorithm for hinge loss (Joachims, 2006). While detailed comparative studies are yet to be

4. For this experiment, $\lambda$ is chosen as in (Joachims, 2006); $\epsilon, \tau = 10^{-6}$.

conducted, preliminary experiments have shown that $l_2$-SVM-MFN and the methods of (Joachims, 2006) are competitive with each other (at their default tolerance parameters).

In the following section, we develop semi-supervised algorithms that provide $l_2$-SVM-MFN the capability of dealing with unlabeled data. 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 $\text{sign}(w^T x)$ that utilizes unlabeled data, typically in situations where $l \ll u$.

## 1.3 Fast Multi-switch Transductive SVMs

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 (Joachims, 1998; Bennett and Demirez, 1998; Fung and Mangasarian, 2001). Since (Joachims, 1998) describes what 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[5]:

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

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

where the hinge loss function, $l(z) = l_1(z) = max(0, 1-z)$ is normally used. The labels on the unlabeled data, $y'_1 \ldots y'_u$, 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 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, $\lambda'$ is a user-provided parameter that provides control over the influence of unlabeled data. For example, if the

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

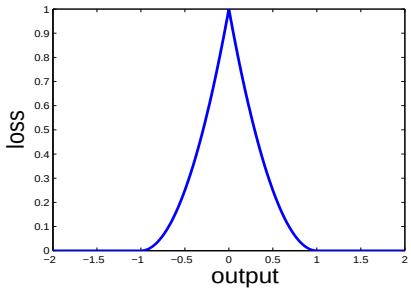
data has distinct clusters with a large margin, but the cluster assumption does not hold, then $\lambda'$ can be set to 0 and the standard SVM is retrieved. If there is enough labeled data, $\lambda, \lambda'$ can be tuned by cross-validation. An initial estimate of $r$ can be made from the fraction 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 (Joachims, 1998) 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$^{\text{light}}$ software. There, the training of SVMs in the inner loops of TSVM uses dual decomposition techniques. As shown by experiments in (Keerthi and DeCoste, 2005), in sparse, linear settings one can obtain significant speed improvements with $l_2$-SVM-MFN over SVM$^{\text{light}}$. 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. that of Collobert et al. (2006) 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.

### 1.3.1 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 1.1**: $l_2$ 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. 1.1. 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 help overcome this problem.

The TSVM algorithm with $l_2$-SVM-MFN closely follows the presentation in (Joachims, 1998). 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 $\lambda'$, the unlabeled data is gradually brought in by increasing $\lambda'$ 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, starting the CGLS/line-search iterations with the current classifier.

### 1.3.2 Multiple Switching

The TSVM algorithm presented in Joachims (1998) 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 Joachims (1998) 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 1.5 establish that this heuristic is remarkably effective in speeding up TSVM training while maintaining generalization performance.

### 1.3.3 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 $\lambda'$) 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.

## 1.4 Semi-supervised SVMs based on Deterministic Annealing

The transductive SVM loss function over the unlabeled examples can be seen from Fig. 1.1 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 by Chapelle and Zien (2005). We now present a new algorithm based on Deterministic Annealing (DA) that can potentially overcome this problem while also being computationally very attractive for large scale applications. Deterministic Annealing (Bilbro et al., 1989; Soderberg, 1989) 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:

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

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 [6] motivates the following objective function, where we relax the binary variables $\mu_j$ to probability-like variables $p_j$, and include entropy terms for the distributions defined by $p_j$:

$$w^\star_T = \underset{w,\{p_j\}_{j=1}^u}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(y_i w^T x_i)$$
$$+\frac{\lambda'}{2u}\sum_{j=1}^{u}\left(p_j l_2(w^T x'_j) + (1-p_j) l_2(-w^T x'_j)\right)$$
$$+\frac{T}{2u}\sum_{j=1}^{u}[p_j \ \log \ p_j + (1-p_j) \ \log \ (1-p_j)] \tag{1.6}$$

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:

$$\frac{1}{u}\sum_{j=1}^{u} p_j = r \tag{1.7}$$

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.

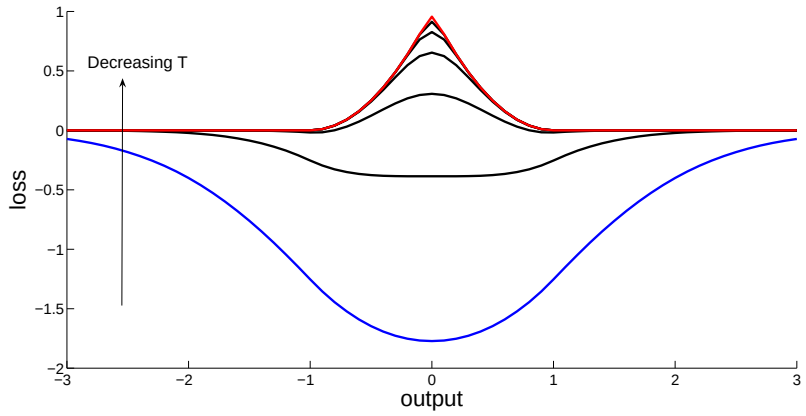
We monitor the value of the objective function in the optimization path and return the solution corresponding to the minimum value achieved.

To develop an intuition for the working on this method, we consider the loss term in the objective function associated with an unlabeled example

6. 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.

as a function of the output of the classifier. Figure 1.2 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. 1.1. The minimizer is slowly tracked as the temperature is lowered towards zero.

**Figure 1.2**: DA loss function parameterized by T.




We note a recently proposed method (Chapelle et al., 2006) with similar motivation.

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. 1.6,1.7 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 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.

### 1.4.1 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

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

$$C = diag\left[\overbrace{\frac{1}{l}...\frac{1}{l}}^{l}, \overbrace{\frac{\lambda' p_1}{u}...\frac{\lambda' p_u}{u}}^{u} \overbrace{\frac{\lambda'(1-p_1)}{u}...\frac{\lambda'(1-p_u)}{u}}^{u}\right] \tag{1.8}$$

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.

### 1.4.2 Optimizing $p$

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

$$\mathcal{L} = \frac{\lambda'}{2u}\sum_{j=1}^{u}\left(p_j l_2(w^T x'_j) + (1-p_j) l_2(-w^T x'_j)\right) +$$

$$\frac{T}{2u}\sum_{j=1}^{u}(p_j \ \log \ p_j + (1-p_j) \ \log \ (1-p_j)) - \nu\left[\frac{1}{u}\sum_{j=1}^{u} p_j - r\right]$$

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

$$p_j = \frac{1}{1+e^{\frac{g_j - 2\nu}{T}}} \tag{1.9}$$

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. 1.7, we get a one-dimensional non-linear equation in $2\nu$:

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

The root is computed by using a hybrid combination of Newton-Raphson iterations and the bisection method together with a carefully set initial value.

### 1.4.3 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 $\epsilon$ 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.

## 1.5 Empirical Study

Semi-supervised learning experiments were conducted to test these algorithms on four medium-scale datasets (aut-avn, real-sim, ccat and gcat) and three large scale (full-ccat,full-gcat,kdd99) datasets. These are listed in Table 1.1. All experiments were performed on Intel Xeon CPU 3GHz, 2GB RAM machines.

#### Software

For software implementation used for benchmarking in this section, we point the reader to the $\text{SVM}_{\text{lin}}$ package available at
`http://www.cs.uchicago.edu/~vikass/svmlin.html`

**Datasets**

The aut-avn and real-sim binary classification datasets come from a collection of UseNet articles[7] from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation. The ccat and gcat datasets pose the problem of separating corporate and government related articles respectively; these are the top-level categories in the RCV1 training data set Lewis et al. (2004). full-ccat and full-gcat are the corresponding datasets containing all the 804414 training and test documents in the RCV1 corpus. 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 kdd99 dataset is from the KDD 1999 competition task to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. This is a relatively low-dimensional dataset containing about 5 million examples.

**Table 1.1**: 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 |
| full-ccat | 47236 | 76.7 | 804414 | - | 0.47 |
| full-gcat | 47236 | 76.7 | 804414 | - | 0.30 |
| kdd99 | 128 | 17.3 | 4898431 | - | 0.80 |

For the medium-scale datasets, the results below are averaged over 10 random stratified splits of training (labeled and unlabeled) and test sets and the detailed performance of SVM, DA and TSVM (single and maximum switching) is studied as a function of the amount of labeled data in the training set. For the large scale datasets full-ccat, full-gcat and kdd99 we are mainly interested in computation times; a transductive setting is used to study performance in predicting the labels of unlabeled data on single splits.

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

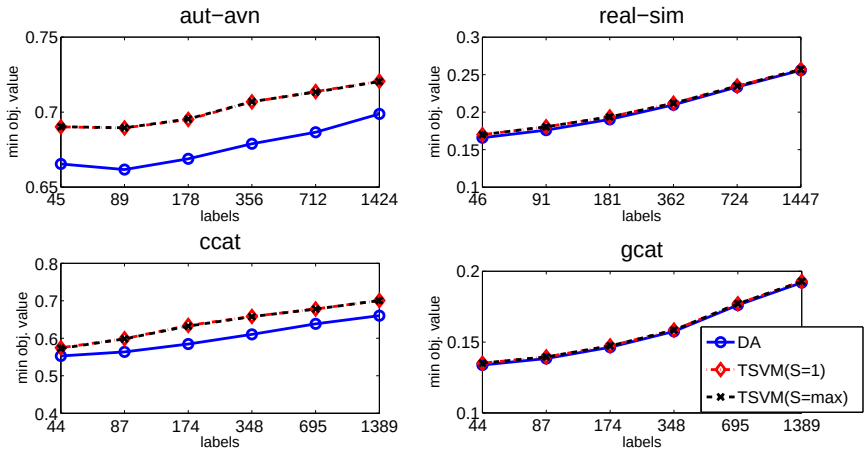
On full-ccat and full-gcat , we train SVM, DA and TSVM with $l = 100, 1000$ labels; for kdd99 we experiment with $l = 1000$ labels.

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 values of $\lambda = 0.001$, and $\lambda' = 1$ for all datasets except[8] for aut-avn and ccat where $\lambda' = 10$.

**Minimization of Objective Function**

We first examine the effectiveness of DA, TSVM with single switching (S=1) and TSVM with maximum switching (S=u/2) in optimizing the objective function. These three procedures are labeled DA,TSVM(S=1) and TSVM(S=max) in Figure 1.3, where we report the minimum value of the objective function with respect to varying number of labels on aut-avn, real-sim, ccat and gcat.

**Figure 1.3**: DA versus TSVM(S=1) versus TSVM(S=max): Minimum value of objective function achieved.




The following observations can be made.

1. Strikingly, *multiple switching leads to no loss of optimization as compared to single switching*. Indeed, the minimum objective value plots attained by single and multiple switching are virtually indistinguishable in Figure 1.3.

8. This produced better results for both TSVM and DA.

**Table 1.2**: Comparison of minimum value of objective functions attained by TSVM(S=max) and DA on full-ccat and full-gcat.

| | full-ccat | | full-gcat | |
|---|---|---|---|---|
| $l, u$ | TSVM | DA | TSVM | DA |
| 100, 402107 | 0.1947 | 0.1940 | 0.1491 | 0.1491 |
| 100, 804314 | 0.1945 | 0.1940 | 0.1500 | 0.1499 |
| 1000, 402107 | 0.2590 | 0.2588 | 0.1902 | 0.1901 |
| 1000, 804314 | 0.2588 | 0.2586 | 0.1907 | 0.1906 |

**Table 1.3**: Comparison of minimum value of objective functions attained by TSVM(S=max) and DA on kdd99

| $l, u$ | TSVM | DA |
|---|---|---|
| 1000, 4897431 | 0.0066 | 0.0063 |

2. As compared to TSVM(S=1 or S=max), DA performs significantly better optimization on aut-avn and ccat; and slightly, but consistently better optimization on real-sim and gcat. These observations continue to hold for full-ccat and full-gcat as reported in Table 1.2 where we only performed experiments with TSVM(S=max). Table 1.3 reports that DA gives a better minimum on the kdd99 dataset too.

**Generalization Performance**

We now examine the generalization performance of DA, TSVM with single and maximum switching. In Figure 1.4 we plot the mean error rate on the (unseen) test set with respect to varying number of labels on aut-avn, real-sim, ccat and gcat. In Figure 1.5, we superimpose these curves over the performance curves of a standard SVM which ignores unlabeled data. Tables 1.4, 1.5 report the corresponding results for full-ccat, full-gcat and kdd99. The following observations can be made.

1. Comparing the performance of SVM against the semi-supervised algorithms in Figure 1.5, the benefit of unlabeled data for boosting 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. On full-ccat and full-gcat too one can see significant gains with unlabeled data. On kdd99, SVM performance with 1000 labels is already very good.

2. In Figure 1.4, we see that on aut-avn, DA outperforms TSVM signifi-

**Figure 1.4**: Error rates on Test set: DA versus TSVM(S=1) versus TSVM(S=max)

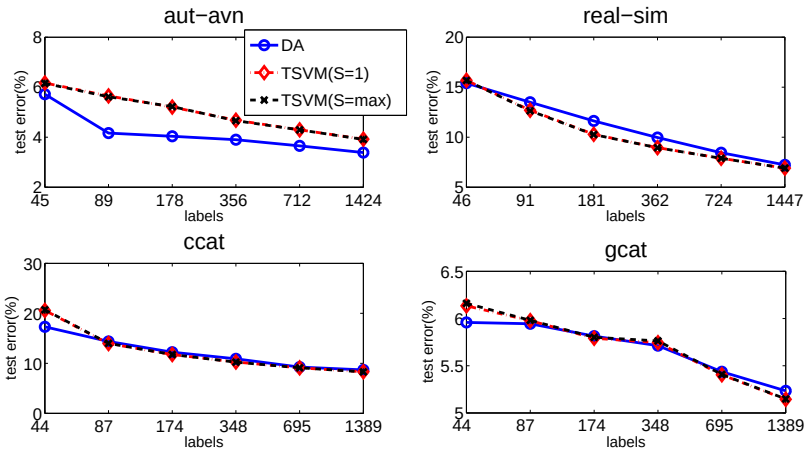



**Table 1.4**: TSVM(S=max) versus DA versus SVM: Error rates over unlabeled examples in full-ccat and full-gcat.

| | full-ccat | | | full-gcat | | |
|---|---|---|---|---|---|---|
| $l, u$ | TSVM | DA | SVM | TSVM | DA | SVM |
| 100, 402107 | 14.81 | 14.88 | 25.60 | 6.02 | 6.11 | 11.16 |
| 100, 804314 | 15.11 | 13.55 | 25.60 | 5.75 | 5.91 | 11.16 |
| 1000, 402107 | 11.45 | 11.52 | 12.31 | 5.67 | 5.74 | 7.18 |
| 1000, 804314 | 11.30 | 11.36 | 12.31 | 5.52 | 5.58 | 7.18 |

cantly. On real-sim, TSVM and DA perform very similar optimization of the transduction objective function (Figure 1.3), but appear to return very different solutions. The TSVM solution returns lower error rates as compared to DA on this dataset. On ccat, DA performed a much better optimization (Figure 1.3) but this does not translate into major error rate improvements. DA and TSVM are very closely matched on gcat. From Table 1.4 we see that TSVM and DA are competitive. On kdd99 (Table 1.5), DA gives the best results.

A lower objective value may not correlate with generalization performance when the cluster assumption is invalid or when it holds to a lesser degree. Also note that the prior knowledge of class balance, which we assume is exactly known in these experiments, is utilized differently by the algorithms.

3. On all datasets we found that *maximum switching returned nearly iden-*

**Figure 1.5**: Benefit of Unlabeled data

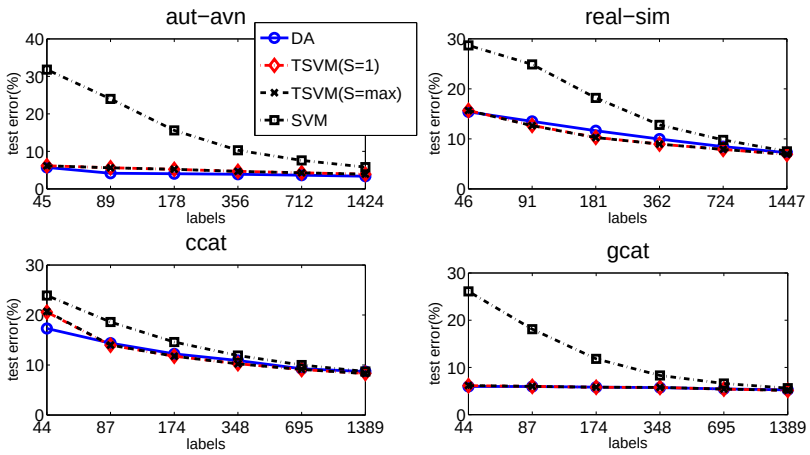



**Table 1.5**: DA versus TSVM($S$ = max) versus SVM: Error rates over unlabeled examples in kdd99.

| $l$,$u$ | TSVM | DA | SVM |
|---|---|---|---|
| 1000, 4897431 | 0.48 | 0.22 | 0.29 |

*tical performance as single switching*. Since it saves significant computation time, as we report in the following section, our study establishes multiple switching (in particular, maximum switching) as a valuable heuristic for training TSVMs.

4. These observations are also true for in-sample transductive performance for the medium scale datasets (detailed results not shown). Both TSVM and DA are found to provide high quality extension to unseen test data.

**Computational Timings**

In Figure 1.6 and Tables 1.6, 1.7 we report the computation time for our algorithms. The following observations can be made.

1. From Figure 1.6 we see that the single switch TSVM can be six to seven times slower than the maximum switching variant, particularly when labels are few. DA is significantly faster than single switch TSVM when labels are relatively few, but slower than TSVM with maximum switching.

**Figure 1.6**: Computation time with respect to number of labels for DA and Transductive $l_2$-SVM-MFN with single and multiple switches.

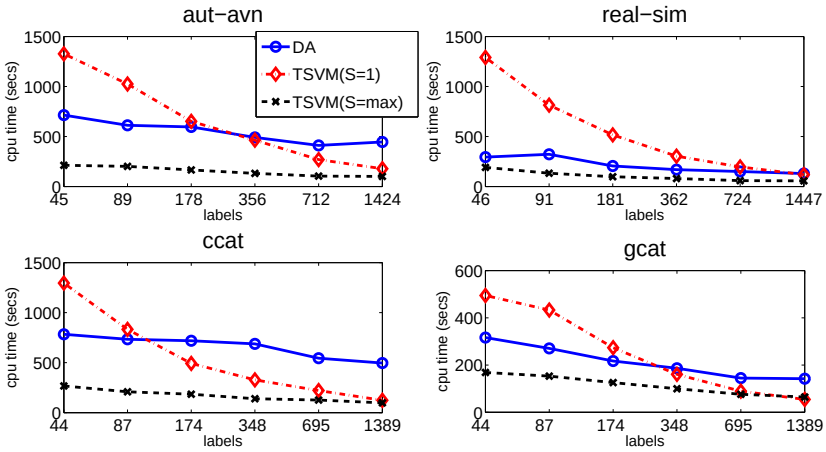



**Table 1.6**: Computation times (mins) for TSVM(S=max) and DA on full-ccat and full-gcat (804414 examples, 47236 features)

| | full-ccat | | full-gcat | |
|---|---|---|---|---|
| $l, u$ | TSVM | DA | TSVM | DA |
| 100, 402107 | 140 | 120 | 53 | 72 |
| 100, 804314 | 223 | 207 | 96 | 127 |
| 1000, 402107 | 32 | 57 | 20 | 42 |
| 1000, 804314 | 70 | 100 | 38 | 78 |

2. In Table 1.6, we see that doubling the amount of data roughly doubles the training time, empirically confirming the linear time complexity of our methods. The training time is also strongly dependent on the number of labels. On kdd99 (Table 1.7), the maximum-switch TSVM took around 15 minutes to process the 5 million examples whereas DA took 2 hours and 20 minutes.

3. On medium scale datasets, we also compared against SVM$^{\text{light}}$ which took on the order of several hours to days to train TSVM. We expect the multi-switch TSVM to also be highly effective when implemented in conjunction with the methods of (Joachims, 2006).

**Table 1.7**: Computation time (mins) for TSVM(S=max) and DA on kdd99 (4898431 examples, 127 features)

| $l, u$ | TSVM | DA |
|---|---|---|
| 1000, 4897431 | 15 | 143 |

**Importance of Annealing**

To confirm the necessity of an annealing component (tracking the minimizer while lowering $T$) in DA, we also compared it with an alternating $w$,$p$ optimization procedure where the temperature parameter is held fixed at $T = 0.1$ and $T = 0.001$. This study showed that annealing is important; it tends to provide higher quality solutions as compared to fixed temperature optimization. It is important to note that the gradual increase of $\lambda'$ to the user-set value in TSVM is also a mechanism to avoid local optima. The non-convex part of the TSVM objective function is gradually increased to a desired value. In this sense, $\lambda'$ 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 $\lambda'$ 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. On the other hand, on time-critical applications one may tradeoff quality of optimization against time by varying the annealing rate.

## 1.6 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 (Sindhwani et al., 2006), and may also be developed to handle clustering and one-class classification problems. These are subjects for future work.

## References

K. Bennett and A. Demirez. Semi-supervised support vector machines. In *Neural Information Processing Systems*, 1998.

G. Bilbro, R. Mann, T.K. Miller, W.E. Snyder, and D.E. Van den. Optimization by mean field annealing, 1989.

Ake Bjork. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

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

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

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

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

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1998.

T. Joachims. Training linear svms in linear time. In *KDD*, 2006.

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.

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.

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

C. Peterson & B. Soderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1:3–22, 1989.

V. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, 1998.