6 Transductive Support Vector Machines

Thorsten Joachims TJ@CS.CORNELL.EDU

In contrast to learning a general prediction rule, V. Vapnik proposed the transductive learning setting where predictions are made only at a fixed number of known test points. This allows the learning algorithm to exploit the location of the test points, making it a particular type of semi-supervised learning problem. Transductive support vector machines (TSVMs) implement the idea of transductive learning by including test points in the computation of the margin. This chapter will provide some examples for why the margin on the test examples can provide useful prior information for learning, in particular for the problem of text classification. The resulting optimization problems, however, are difficult to solve. The chapter reviews exact and approximate optimization methods and discusses their properties. Finally, the chapter discusses connections to other related semi-supervised learning approaches like co-training and methods based on graph cuts, which can be seen as solving variants of the TSVM optimization problem.

6.1 Introduction

The setting of transductive inference was introduced by Vapnik (e.g. (Vapnik, 1998)). As an example of a transductive learning task, consider the problem of learning from relevance feedback in information retrieval (see (Baeza-Yates and Ribeiro-Neto, 1999)). The user marks some documents returned by a search engine in response to an initial query as relevant or irrelevant. These documents then serve as a training set for a binary text classification problem. The goal is to learn a rule that accurately classifies all remaining documents in the database according to their relevance. Clearly, this problem can be thought of as a supervised learning problem. But it is different from many other (inductive) learning problems in at least two respects.

First, the learning algorithm does not necessarily have to learn a general rule, but it only needs to predict accurately for a finite number of test examples (i.e.,

the documents in the database). Second, the test examples are known a priori and can be observed by the learning algorithm during training. This allows the learning algorithm to exploit any information that might be contained in the location of the test examples. Transductive learning is therefore a particular case of semi-supervised learning, since it allows the learning algorithm to exploit the unlabeled examples in the test set. The following focuses on this second point, while chapter 24 elaborates on the first point.

transductive learning setting

More formally, the transductive learning setting can be formalized as follows.[1] Given is a set

$$S = \{1, 2, ..., n\} \tag{6.1}$$

that enumerates all n possible examples. In our relevance feedback example from above, there would be one index i for each document in the collection. We assume that each example i is represented by a feature vector $\mathbf{x}_i \in \mathbb{R}^d$. For text documents, this could be a TFIDF vector representation (see e.g. (Joachims, 2002)), where each document is represented by a scaled and normalized histogram of the words it contains. The collection of feature vectors for all examples in S is denoted as

$$X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n). \tag{6.2}$$

For the examples in S, labels

$$Y = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n) \tag{6.3}$$

are generated independently according to a distribution $P(\mathbf{y}_1, ..., \mathbf{y}_n) = \prod_{i=1}^{n} P(\mathbf{y}_i)$. For simplicity, we assume binary labels $\mathbf{y}_i \in \{-1, +1\}$.

As the training set, the learning algorithm can observe the labels of l randomly selected examples $S_{train} \subset S$. The remaining $u = n - l$ examples form the test set $S_{test} = S \setminus S_{train}$.

$$S_{train} = \{l_1, ..., l_l\} \qquad S_{test} = \{u_1, ..., u_u\} \tag{6.4}$$

When training a transductive learning algorithm $\mathcal{L}$, it not only has access to the training vectors X_{train} and the training labels Y_{train},

$$X_{train} = (\mathbf{x}_{l_1}, \mathbf{x}_{l_2}, ..., \mathbf{x}_{l_l}) \qquad Y_{train} = (\mathbf{y}_{l_1}, \mathbf{y}_{l_2}, ..., \mathbf{y}_{l_l}), \tag{6.5}$$

but also to the unlabeled test vectors

$$X_{test} = (\mathbf{x}_{u_1}, \mathbf{x}_{u_2}, ..., \mathbf{x}_{u_l}). \tag{6.6}$$

The transductive learner uses X_{train}, Y_{train}, and X_{test} (but not the labels Y_{test} of

1. While several other, more general, definitions of transductive learning exist (Vapnik, 1998; Joachims, 2002; Derbeko et al., 2003), this one was chosen for the sake of simplicity.

the test examples) to produce predictions,

$$Y^*_{test} = (\mathbf{y}^*_{u_1}, \mathbf{y}^*_{u_2}, ..., \mathbf{y}^*_{u_u}), \tag{6.7}$$

for the labels of the test examples. The learner's goal is to minimize the fraction of erroneous predictions,

$$Err_{test}(Y^*_{test}) = \frac{1}{u} \sum_{i \in S_{test}} \delta_{0/1}(\mathbf{y}^*_i, \mathbf{y}_i), \tag{6.8}$$

on the test set. $\delta_{0/1}(a, b)$ is zero if $a = b$, otherwise it is one.

At first glance, the problem of transductive learning may not seem profoundly different from the usual inductive setting. One could learn a classification rule based on the training data and then apply it to the test data afterward. However, a crucial difference is that the inductive strategy would ignore any information potentially conveyed in X_{test}.

What information do we get from studying the test sample X_{test} and how could we use it? The fact that we deal with only a finite set of points means that the hypothesis space $\mathcal{H}$ of a transductive learner is necessarily finite — namely, all vectors $\{-1, +1\}^n$. Following the principle of structural risk minimization (Vapnik, 1998), we can structure $\mathcal{H}$ into a nested structure

structural risk minimization

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H} = \{-1, +1\}^n. \tag{6.9}$$

The structure should reflect prior knowledge about the learning task. In particular, the structure should be constructed so that, with high probability, the correct labeling of S (or labelings that make few errors) is contained in an element $\mathcal{H}_i$ of small cardinality. This structuring of the hypothesis space $\mathcal{H}$ can be motivated using generalization error bounds from statistical learning theory. In particular, for a learner $\mathcal{L}$ that searches for a hypothesis $(Y^*_{train}, Y^*_{test}) \in \mathcal{H}_i$ with small training error,

$$Err_{test}(Y^*_{train}) = \frac{1}{l} \sum_{i \in S_{train}} \delta_{0/1}(\mathbf{y}^*_i, \mathbf{y}_i), \tag{6.10}$$

it is possible to upper-bound the fraction of test errors $Err_{test}(Y^*_{test})$ (Vapnik, 1998; Derbeko et al., 2003). With probability $1 - \eta$

transductive generalization error bound

$$Err_{test}(Y^*_{test}) \leq Err_{train}(Y^*_{train}) + \Omega(l, u, |\mathcal{H}_i|, \eta) \tag{6.11}$$

where the confidence interval $\Omega(l, u, |\mathcal{H}_i|, \eta)$ depends on the number of training examples l, the number of test examples u, and the cardinality $|\mathcal{H}_i|$ of $\mathcal{H}_i$ (see (Vapnik, 1998) for details). The smaller the cardinality $|\mathcal{H}_i|$, the smaller is the confidence interval $\Omega(l, u, |\mathcal{H}_i|, \eta)$ on the deviation between training and test error.

The bound indicates that a good structure ensures accurate prediction of the test labels. And here lies a crucial difference between transductive and inductive learners. Unlike in the inductive setting, we can study the location X_{test} of the test

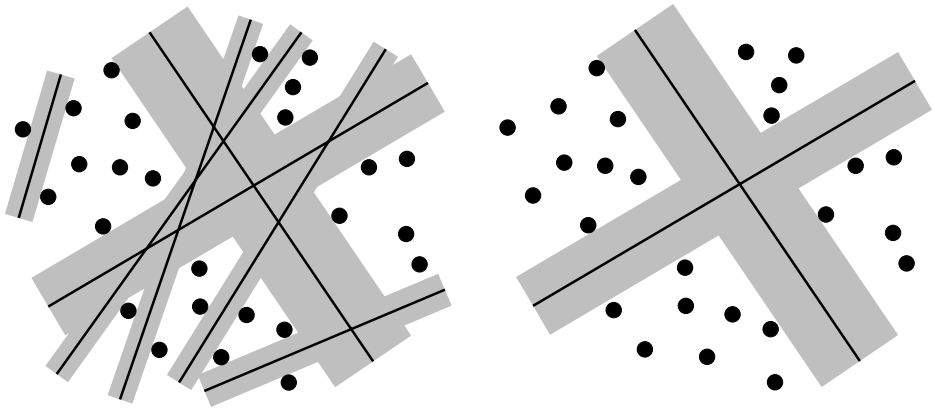

Figure 6.1 The two graphs illustrate the labelings that margin hyperplanes can realize dependent on the margin size. Example points are indicated as dots: the margin of each hyperplane is illustrated by the gray area. The left graph shows the separators $\mathcal{H}_\rho$ for a small margin threshold ρ. The number of possible labelings N_ρ decreases as the margin threshold is increased, as in the graph on the right.

examples when defining the structure. *In particular, in the transductive setting it is possible to encode prior knowledge we might have about the relationship between the geometry of* $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$ *and* $P(\mathbf{y}_1, ..., \mathbf{y}_n)$. If such a relationship exists, we can build a more appropriate structure and reduce the number of training examples necessary for achieving a desired level of prediction accuracy. This line of reasoning is detailed in chapter 24.

6.2 Transductive Support Vector Machines

train and test set margin

Transductive support vector machines (TSVMs) assume a particular geometric relationship between $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$ and $P(\mathbf{y}_1, ..., \mathbf{y}_n)$. They build a structure on $\mathcal{H}$ based on the margin of hyperplanes $\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}$ on the complete sample $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, including both the training and the test vectors. The margin of a hyperplane on X is the minimum distance to the closest example vectors in X.

$$\min_{i \in [1..n]} \left[\frac{\mathbf{y}_i}{\|\mathbf{w}\|} (\mathbf{w} \cdot \mathbf{x}_i + b) \right] \tag{6.12}$$

The structure element $\mathcal{H}_\rho$ contains all labelings of X which can be achieved with hyperplane classifiers $h(\mathbf{x}) = sign\{\mathbf{x} \cdot \mathbf{w} + b\}$ that have a margin of at least ρ on X. The dependence of $\mathcal{H}_\rho$ on ρ is illustrated in figure 6.1. Intuitively, building the structure based on the margin gives preference to labelings that follow cluster boundaries over labelings that cut through clusters. Vapnik shows that the size of the margin ρ can be used to control the cardinality of the corresponding set of

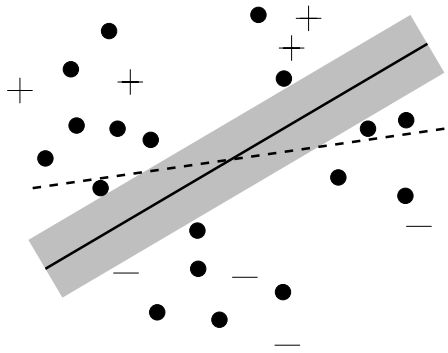

Figure 6.2 For the same data as in figure 6.1, some examples are now labeled. Positive/negative examples are marked as $+/-$. The dashed line is the solution of an inductive SVM, which finds the hyperplane that separates the training data with largest margin, but ignores the test vectors. The solid line shows the hard-margin transductive classification, which is the labeling that has zero training error and the largest margin with respect to both the training and the test vectors. The TSVM solution aligns the labeling with the cluster structure in the training and test vectors.

labelings $\mathcal{H}_\rho$. More formally, the following theorem provides an upper bound on the number of labelings $|\mathcal{H}_\rho|$ that can be achieved with hyperplanes that have a margin of at least ρ.

Theorem 6.1 ((Vapnik, 1998))
For any n vectors $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ that are contained in a ball of diameter R, the number $|\mathcal{H}_\rho|$ of possible binary labelings $\mathbf{y}_1, ..., \mathbf{y}_n \in \{-1, +1\}$ that can be realized with hyperplane classifiers $h(\mathbf{x}) = sign\{\mathbf{x} \cdot \mathbf{w} + b\}$ of margin at least ρ,

$$\forall_{i=1}^{n} : \frac{\mathbf{y}_i}{\|\mathbf{w}\|} [\mathbf{w} \cdot \mathbf{x}_i + b] \geq \rho \tag{6.13}$$

is bounded by

$$|\mathcal{H}_\rho| \leq e^{d\left(\ln \frac{n+k}{d}+1\right)}, \qquad d = \frac{R^2}{\rho^2} + 1. \tag{6.14}$$

Note that the number of labelings $|\mathcal{H}_\rho|$ does not necessarily depend on the number of features d. As suggested by the theorem, TSVMs sort all labelings by their margin ρ on X to build the structure on $\mathcal{H}$. Structural risk minimization argues that a learning algorithm should select the labeling $Y^* \in \mathcal{H}_\rho$ for which training error $Err_{train}(Y^*_{train})$ and cardinality of $\mathcal{H}_\rho$ minimize the generalization error bound (6.11). For the special case of requiring zero training error (i. e. $Err_{train}(Y^*_{train}) = 0$), optimizing the bound means finding the labeling with the largest margin on the complete set of vectors. This leads to the following optimization problem (OP) (Vapnik, 1998).

hard-margin TSVM

OP1 (TRANSDUCTIVE SVM (HARD-MARGIN))

$$\text{minimize:} \quad V(\mathbf{y}^*_{u_1}, ..., \mathbf{y}^*_{u_u}, \mathbf{w}, b) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} \tag{6.15}$$

$$\text{subject to:} \quad \forall^l_{i=1} : \mathbf{y}_{l_i}[\vec{w} \cdot \mathbf{x}_{l_i} + b] \geq 1 \tag{6.16}$$

$$\forall^u_{j=1} : \mathbf{y}^*_{u_j}[\vec{w} \cdot \mathbf{x}^*_{u_j} + b] \geq 1 \tag{6.17}$$

$$\forall^u_{j=1} : \mathbf{y}^*_{u_j} \in \{-1, +1\} \tag{6.18}$$

Solving this problem means finding the labeling $\mathbf{y}^*_{u_1}, ..., \mathbf{y}^*_{u_k}$ of the test data for which the hyperplane that separates both training and test data has maximum margin. Figure 6.2 illustrates this. The figure also shows the solution that an inductive SVM (Cortes and Vapnik, 1995; Vapnik, 1998) computes. An inductive SVM also finds a large-margin hyperplane, but it considers only the training vectors while ignoring all test vectors. In particular, a hard-margin inductive SVM computes the separating hyperplane that has zero training error and the largest margin with respect to the training examples.

inductive SVM

To be able to handle nonseparable data, one can introduce slack variables ξ_i (Joachims, 1999) similar to inductive SVMs (Cortes and Vapnik, 1995).

soft-margin TSVM

OP2 (TRANSDUCTIVE SVM (SOFT-MARGIN))

$$\text{min: } W(\mathbf{y}^*_{u_1}, ..., \mathbf{y}^*_{u_u}, \mathbf{w}, b, \xi_1, ..., \xi_l, \xi^*_1, ..., \xi^*_u) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{l}\xi_i + C^*\sum_{j=1}^{u}\xi^*_j \tag{6.19}$$

$$\text{s.t.: } \forall^l_{i=1} : \mathbf{y}_{l_i}[\mathbf{w} \cdot \mathbf{x}_{l_i} + b] \geq 1 - \xi_i \tag{6.20}$$

$$\forall^u_{j=1} : \mathbf{y}^*_{u_j}[\mathbf{w} \cdot \mathbf{x}^*_{u_j} + b] \geq 1 - \xi^*_j \tag{6.21}$$

$$\forall^u_{j=1} : \mathbf{y}^*_{u_j} \in \{-1, +1\} \tag{6.22}$$

$$\forall^l_{i=1} : \xi_i \geq 0 \tag{6.23}$$

$$\forall^u_{j=1} : \xi^*_j \geq 0 \tag{6.24}$$

C and C^* are parameters set by the user. They allow trading off margin size against misclassifying training examples or excluding test examples. C^* can be used reduce sensitivity toward outliers (i.e., single examples falsely reducing the margin on the test data).

kernels

Both inductive and transductive SVMs can be extended to include kernels (Boser et al., 1992; Vapnik, 1998). Making use of duality techniques from optimization theory, kernels allow learning nonlinear rules as well as classification rules over nonvectorial data (see e.g. (Schölkopf and Smola, 2002)) without substantially changing the optimization problems.

Note that in both the hard-margin formulation (OP1) and the soft-margin formulation (OP2) of the TSVM, the labels of the test examples enter as integer variables. Due to the constraints in Eqs. 6.18 and 6.22 respectively, both OP1 and OP2 are no longer convex quadratic programs like the analogous optimization problems for inductive SVMs. Before discussing methods for (approximately) solving the TSVM

	nuclear	physics	atom	parsley	basil	salt	and
D1	**1**						**1**
D2	1	1	1				1
D3			1				1
D4				1	1		1
D5				1		1	1
D6					**1**	**1**	**1**

Figure 6.3 Example of a text-classification problem with co-occurrence pattern. Rows correspond to documents, columns to words. A table entry of 1 denotes the occurrence of a word in a document.

optimization problems, let's first discuss some intuition about why structuring the hypothesis space based on the margin on the test examples might be reasonable.

6.3 Why Use Margin on the Test Set?

Why should it be reasonable to prefer a labeling with a large margin over a labeling with a smaller margin, even if both have the same training error? Clearly, this question can only be addressed in the context of a particular learning problem. In the following, we will consider text classification as an example. In particular, for topic-based text classification it is known that good classification rules typically have a large margin (Joachims, 2002). The following example gives some intuition for why this is the case.

In the field of information retrieval it is well known that words in natural language occur in co-occurrence patterns (see e.g. (van Rijsbergen, 1977)). Some words are likely to occur together in one document; others are not. For examples, when asking Google about all documents containing the words `pepper` and `salt`, it returns 3,500,000 webpages. When asking for the documents with the words `pepper` and `physics`, we get only 248,000 hits, although `physics` (162,000,000 hits) is a more popular word on the web than `salt` (63,200,000 hits). Many approaches in information retrieval try to exploit this cluster structure of text (see e.g. (Baeza-Yates and Ribeiro-Neto, 1999, chapter 5)). It is this co-occurrence information that TSVMs exploit as prior knowledge about the learning task.

Consider the example in figure 6.3. Imagine document $D1$ was given as a training example for class A and document $D6$ was given as a training example for class B. How should we classify documents $D2$ to $D5$ (the test set)? Even if we did not understand the meaning of the words, we would classify $D2$ and $D3$ into class A, and $D4$ and $D5$ into class B. We would do so even though $D1$ and $D3$ do not share any informative words. The reason we choose this classification of the test data over the others stems from our prior knowledge about the properties of text and common text-classification tasks. Often we want to classify documents by

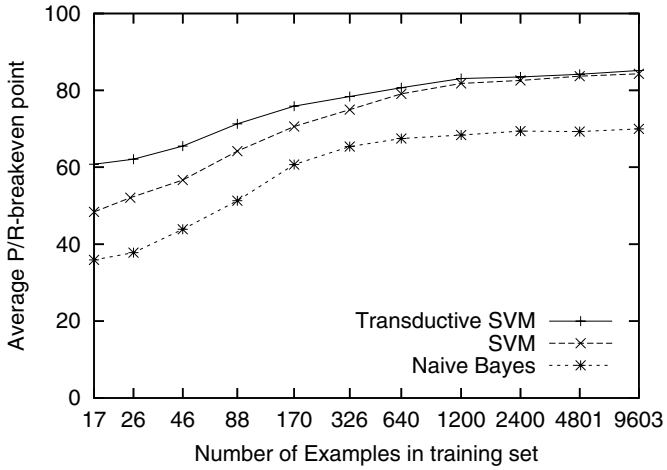

Figure 6.4 Macro-averaged PRBEP on the Reuters data set for different training set sizes and a test set size of 3299.

topic, source, or style. For these types of classification tasks we find stronger co-occurrence patterns within classes than between different classes. In our example we analyzed the co-occurrence information in the test data and found two clusters. These clusters indicate different topics of $\{D1, D2, D3\}$ versus $\{D4, D5, D6\}$, and we choose the cluster separator as our classification. Note again that we got to this classification by studying the location of the test examples, which is not possible for an inductive learner.

The TSVM outputs the same classification as we suggested above, although all 16 labelings of $D2$ to $D5$ can be achieved with linear separators. Assigning $D2$ and $D3$ to class A and $D4$ and $D5$ to class B is the maximum-margin solution (i.e., the solution of OP1). The maximum-margin bias appears to reflect our prior knowledge about text classification well. By measuring margin on the test set, the TSVM exploits co-occurrence patterns that indicate boundaries between topics.

6.4 Experiments and Applications of TSVMs

Structuring the hypothesis space using margin was obviously beneficial in the toy example above. Experiments have confirmed that this also holds in practice.

TSVMs in text classification

Figures 6.4 and 6.5 (from Joachims (1999)) give empirical evidence that TSVMs improve prediction performance on real text-classification tasks, namely the Reuters-21578 text-classification benchmark. The standard "ModApte" training/test split is used, leading to a corpus of 9603 training documents and 3299 test documents. The results are averaged over the ten most frequent topics, while keeping all documents. Each topic leads to a binary classification problem, where documents about the topic are positive examples, and all other documents are neg-

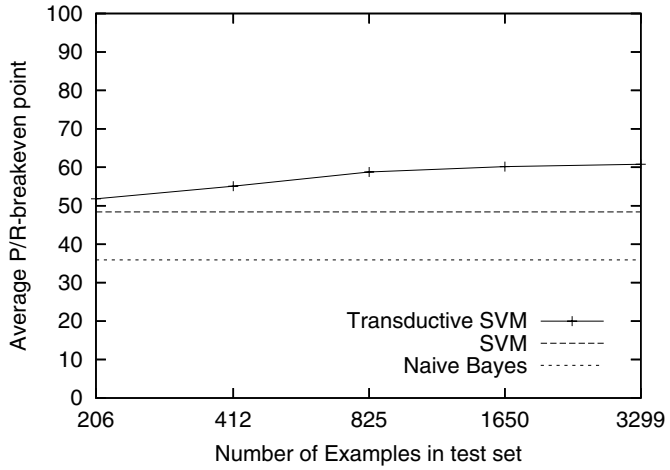

Figure 6.5 Macro-averaged PRBEP on the Reuters data set for 17 training documents and varying test set size for the TSVM.

ative examples. The performance of each binary classifier is measured in terms of the precision/recall breakeven point (PRBEP). The PRBEP is the percentage of positive test examples that are classified correctly, if the classifier is allowed to predict as many test examples as positive as there are true positives in the test set (see e.g. (Joachims, 2002)). The precise setup is described in (Joachims, 1999).

Figures 6.4 and 6.5 show the effect of using TSVM instead of inductive methods. To provide a baseline for comparison, the results of the inductive SVM and a multinomial naive Bayesnaive Bayes classifier are added. The SVM and the TSVM are trained using SVM^{light}, available at `svmlight.joachims.org`. Figure 6.4 shows the effect of varying the size of the training set. The advantage of using the transductive approach is largest for small training sets. For increasing training set size, the performance of the SVM approaches that of the TSVM. This is to be expected, since labeled examples eventually convey the same information about the distribution of the example vectors as the unlabeled data.

The influence of the test set size on the performance of the TSVM is displayed in figure 6.5. The bigger the test set, the larger the performance gap between SVM and TSVM. Adding more test examples beyond 3299 is not likely to increase performance by much, since the graph appears to flatten out. The curves are fairly typical and similar behavior was also observed on other problems. The results for other text classification data sets can be found in (Joachims, 2002).

Similar gains in performance of the TSVM over an inductive SVM were reported by Chapelle et al. (2003). For classifying net news articles they report that the TSVM almost halves the prediction error for small training sets of 16 examples. For an email classification problem, the results of Kockelkorn et al. (2003) also indicate that TSVMs substantially outperform inductive SVMs for small training sets. Small improvements on text classification problems are also reported by Tong and Koller (2001). However, they conclude that the effect of active learning,

where the algorithm can ask for the labels of particular examples, dominates the improvement seen from the TSVM. This is in contrast to the findings of Wang et al. (2003). They find that incorporating TSVMs into their active learning procedure for image retrieval based on relevance feedback substantially improves performance. For more text-classification experiments see chapter 3.

TSVMs for image retrieval

Beyond text classification, Bennett and Demiriz (1999) have applied their L_1-norm variant of transductive SVMs to several UCI benchmark problems. They find small but fairly consistent improvements over these tasks. A key difference from most other experiments with transductive learning are the small test sets that were used. Due to efficiency limitations of the mixed-integer programming code they used for training, all test sets contained no more than 70 examples. Their evaluation of regular TSVMs on a subset of these UCI benchmarks shows mixed results (Demiriz and Bennett, 2000). Similar findings on UCI benchmarks are also reported by Joachims (2003), where the differences between inductive SVMs and TSVMs were found to be small.

TSVMs for UCI benchmarks

Several applications of TSVMs in bioinformatics have been explored. For example, they have been used to recognize promoter sequences in genes. Kasabov and Pang (2004) report that TSVMs substantially outperform inductive SVMs in their experiments. However, for the problem of predicting the functional properties of proteins, Krogel and Scheffer (2004) find that TSVMs significantly decrease performance compared to inductive SVMs.

TSVMs in bioinformatics

Goutte et al. (2002) apply TSVMs to a problem of recognizing entities (e.g., gene names, protein names) in medical text. They find that TSVMs substantially improve performance for medium-sized training sets, and perform at least comparably to an alternative transductive learning method based on Fisher kernels.

TSVMs for named entity recognition

Summarizing the results, it appears that TSVMs are particularly well suited for text classification and several other (typically high-dimensional) learning problems. However, on some problems the TSVM performs roughly equivalently to an inductive SVM, or sometimes even worse. This is to be expected, since it is likely that structuring the hypothesis space according to margin size is inappropriate for some applications. Furthermore, it is likely that the difficulty of finding the optimum of the TSVM optimization problem has led to suboptimal results in some cases. We discuss algorithms for solving the TSVM optimization problem next.

6.5 Solving the TSVM Optimization Problem

Both the hard soft-margin TSVM optimization problems can be written as mixed-integer problems with a quadratic objective and linear constraints. Unfortunately, currently no algorithm is known to efficiently find a globally optimal solution.

mixed-integer programming

Vapnik and colleagues (Vapnik and Sterin, 1977; Wapnik and Tscherwonenkis, 1979) proposed the use of branch-and-bound search to find the global optimium of the TSVM optimization problem. Similarly, Bennett and Demiriz (1999) consider standard mixed-integer programming software like CPLEX to solve a variant of

the TSVM optimization problem. To be able to use such software, they replace the term $\mathbf{w} \cdot \mathbf{w} = \|\mathbf{w}\|_2^2$ in the objective with $\|\mathbf{w}\|_1$ so that the objective becomes linear. However, while both approaches produce globally optimal solutions, they can solve only small problems with less than 100 test examples in reasonable time. Unfortunately, figure 6.5 suggests that the biggest benefits of transductive learning occur only for larger test sets.

SVM^{light}

The algorithm implemented in SVM^{light} does not necessarily produce a globally optimal solution, but can handle test sets with up to 100,000 examples in reasonable time (Joachims, 1999, 2002). Most of the empirical results in the previous section were produced using this algorithm. The algorithm performs a kind of coordinate-descent local search starting from an initial labeling of the test examples derived from an inductive SVM. The ratio of test examples that are classified as positive (by adjusting the hyperplane threshold b) in this initial labeling is specified by the user or estimated from the ratio of positive to negative examples in the training set. This ratio is maintained throughout the optimization process to avoid degenerate solutions that assign all test examples to the same class.[2] In every step of the local search, the algorithm selects two examples (one positive and one negative) and swaps their labels. The way the examples are selected guarantees a strict improvement of the objective function (i.e., the soft margin) in every such step. In addition, the algorithm starts with a small value of C^* and raises it throughout the optimization process. This means that most ξ^* are non-zero in the initial phase of the search, resulting in a smoother objective function. Toward the end of the search, incrementally increasing the value of C^* toward the desired target value makes the problem closer to the desired objective. A more detailed explanation of the algorithm is given in (Joachims, 2002).

gradient descent

A related block coordinate descent method was proposed by Demiriz and Bennett (2000). The algorithm also alternates between changing the labels of the test examples and recomputing the margin. Differences compared to the SVM^{light}algorithm lie in the selection of the labels to change, the number of labels that are changed in each iteration, and in the heuristics that are aimed to avoid local optima. A similar algorithm for the L_1-norm variant of the TSVM is described by Fung and Mangasarian (2001).

semi-definite relaxation

De Bie and Cristianini (2004a) explore a convex approximation of the TSVM optimization problem (also see chapter 7). They present a relaxation that takes the form of a semi-definite program. While this program can be solved in polynomial time, it becomes too inefficient for test sets with more than 100 examples. However, assuming a low-rank structure of the test labels derived from a spectral decomposition technique, De Bie and Cristianini push the efficieny limit to several thousands of test examples.

2. In text classification, assigning all test examples to the same class typically gives larger margins than any other labeling. Clearly, this is an undesirable solution and indicates a problem with the TSVM approach. A method that does not exhibit this problem is presented in Joachims (2003).

6.6 Connection to Related Approaches

graph cuts

The difficulty in solving the TSVM optimization problem has led to much interest in other formulations of transductive learning algorithms. The goal is to exploit the same type of relationship between the geometry of the test examples — or unlabeled examples more generally — and their labels, but that have computationally more convenient properties. Graph partitioning approaches based on st-min-cuts (Blum and Chawla, 2001) and spectral graph partitioning explicitly or implicitly pursued this goal (Belkin and Niyogi, 2002; Chapelle et al., 2003; Joachims, 2003; Zhu et al., 2003b) (see also chapters 11, 12, 13, 14, and 15). For example, the method in (Joachims, 2003) is explicitly derived analogous to a TSVM as a transductive version of the k-nearest neighbor classifier.

ridge regression

Ridge regression is a method closely related to regression SVMs. Chapelle et al. (1999) derive a tranductive variant of ridge regression. Since the class labels do not need to be discrete for regression problems, they show that the solution of the associated optimization problem can be computed efficiently.

co-training

Co-training (Blum and Mitchell, 1998) exploits two redundant representations of a learning problem for semi-supervised learning. A connection to general transductive learning comes from the insight that co-training produces transductive learning problems that have large margin (Joachims, 2003, 2002). In fact, TSVMs and spectral partitioning methods appear to perform well on co-training problems (Joachims, 2003).

confidence estimation

Connecting to concepts of algorithmic randomness, Gammerman et al. (1998), Vovk et al. (1999), and Saunders et al. (1999) presented approaches to estimating the confidence of a prediction based on a transductive setting. A similar goal using a Bayesian approach is pursued by Graepel et al. (2000). Since their primary aim is not a reduced error rate in general, but a measure of confidence for a particular prediction, they consider only test sets with exactly one example.

6.7 Summary and Conclusions

Transductive support vector machines exploit the geometric (cluster) structure in the feature vectors of the test examples, which makes them a particular kind of semi-supervised learning method. In particular, TSVMs find the labeling of the test examples that maximizes margin jointly on the training and the test data. Intuitively, this produces labeling of the test examples so that class boundaries follow cluster boundaries. Empirical findings suggest that TSVMs are particularly well suited for text classification and several other (typically high-dimensional) learning problems, often showing large accuracy gains for small training sets and large test sets. However, on some problems the TSVM performs roughly equivalently to an inductive SVM, or sometimes even worse. Partially, failure on some tasks may be due to the difficulty of finding the optimum of the TSVM optimization problem.

Finding the globally optimal solution is intractable for interestingly sized test sets. Existing algorithms resort to local search or to relaxing the optimization problem. More work is needed on tractable formulations and algorithms for transductive learning, as well as a deeper theoretical and empirical understanding of its potential.