Transductive Inference for Text Classification using Support Vector Machines

Thorsten Joachims
Universität Dortmund, LS VIII
44221 Dortmund, Germany
joachims@ls8.cs.uni-dortmund.de

Abstract

This paper introduces Transductive Support Vector Machines (TSVMs) for text classification. While regular Support Vector Machines (SVMs) try to induce a general decision function for a learning task, Transductive Support Vector Machines take into account a particular test set and try to minimize misclassifications of just those particular examples. The paper presents an analysis of why TSVMs are well suited for text classification. These theoretical findings are supported by experiments on three test collections. The experiments show substantial improvements over inductive methods, especially for small training sets, cutting the number of labeled training examples down to a twentieth on some tasks. This work also proposes an algorithm for training TSVMs efficiently, handling 10,000 examples and more.

1 Introduction

Over the recent years, text classification has become one of the key techniques for organizing online information. It can be used to organize document databases, filter spam from people's email, or learn users' newsreading preferences. Since hand-coding text-classifiers is impractical — or at best costly — in many settings, it is preferable to learn classifiers from examples. It is crucial that the learner be able to generalize well using little training data. A news-filtering service, for example, requiring a hundred days' worth of training data is unlikely to please even the most patient users.

The work presented here tackles the problem of learning from small training samples by taking a *transductive* [Vapnik, 1998], instead of an inductive approach. In the inductive setting the learner tries to induce a decision function which has a low error rate on the whole distribution of examples for the particular learning task. Often, this setting is unnecessarily complex. In many situations we do not care about the particular decision function, but rather that we classify a *given set of examples (i.e. a test set) with as few errors as possible.* This is the goal of transductive inference.

Some examples of transductive text classification tasks are the following. All have in common that there is little training data, but a very large test set.

Relevance Feedback : This is a standard technique in free-text information retrieval. The user marks some documents returned by an initial query as relevant or irrelevant. These compose the training set of a text classification task, while the remaining document database is the test set. The user is interested in a good classification of the test set into those documents relevant or irrelevant to the query.

Netnews Filtering : Each day a large number of netnews articles is posted. Given the few training examples the user labeled on previous days, he or she wants today's most interesting articles.

Reorganizing a document collection : With the advance of paperless offices, companies start using document databases with classification schemes. When introducing new categories, they need text classifiers which, given some training examples, classify the rest of the database automatically.

This paper introduces *Transductive Support Vector Machines* (TSVMs) for text classification. They substantially improve the already excellent performance of SVMs for text classification [Joachims, 1998, Dumais et al., 1998]. Especially for very small training sets, TSVMs reduce the required amount of labeled training data down to a twentieth for some tasks. To facilitate the large-scale transductive learning needed for text classification, this paper also proposes a new algorithm for efficiently training TSVMs with 10,000 examples and more.

2 Text Classification

The goal of text classification is the automatic assignment of documents to a fixed number of semantic categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples which assign categories automatically. This is a supervised learning problem. To facilitate effective and efficient learning, each category is treated as a separate binary classification problem. Each such problem answers the question of whether or not a document should be assigned to a particular category.

Documents, which typically are strings of characters, have to be transformed into a representation suitable for the learning algorithm and the classification task. Information Retrieval research suggests that word stems work well as representation units and that for many tasks their ordering can be ignored without losing too much information. The word stem is derived from the occurrence form of a word by removing case and flection information [Porter, 1980]. For example "computes", "computing", and "computer" are all mapped to the same stem "comput". The terms "word" and "word stem" will be used synonymously in the following.

This leads to an attribute-value representation of text. Each distinct word w_i corresponds to a feature with $TF(w_i, x)$, the number of times word w_i occurs in the document x, as its value. Figure 1 shows an example feature vector for a particular document. Refining this basic representation, it has been shown that scaling the dimensions of the feature vector with their *inverse document frequency* $IDF(w_i)$ [Salton and Buckley, 1988] leads to an improved performance. $IDF(w_i)$ can be calculated from the document frequency $DF(w_i)$, which is the number of documents the word w_i occurs in.

$$IDF(w_i) = log\left(\frac{n}{DF(w_i)}\right) \qquad (1)$$

Here, n is the total number of documents. Intuitively,

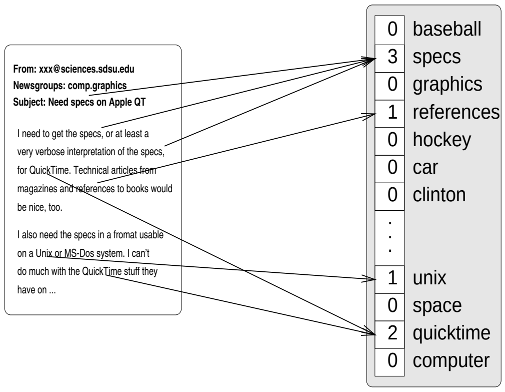

Figure 1: Representing text as a feature vector.

the inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. To abstract from different document lengths, each document feature vector $\vec{x}_i$ is normalized to unit length.

3 Transductive Support Vector Machines

The setting of transductive inference was introduced by Vapnik (see for example [Vapnik, 1998]). For a learning task $P(\vec{x}, y) = P(y|\vec{x})P(\vec{x})$ the learner L is given a hypothesis space H of functions $h : X \longrightarrow \{-1, 1\}$ and an i.i.d. sample S_{train} of n training examples

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n) \qquad (2)$$

Each training example consists of a document vector $\vec{x} \in X$ and a binary label $y \in \{-1, +1\}$. In contrast to the inductive setting, the learner is also given an i.i.d. sample S_{test} of k test examples

$$\vec{x}_1^*, \vec{x}_2^*, ..., \vec{x}_k^* \qquad (3)$$

from the same distribution. The transductive learner L aims to selects a function $h_L = L(S_{train}, S_{test})$ from H using S_{train} and S_{test} so that the expected number of erroneous predictions

$$R(L) = \int \frac{1}{k} \sum_{i=1}^{k} \Theta(h_L(\vec{x}_i^*), y_i^*) dP(\vec{x}_1, y_1) \cdots dP(\vec{x}_k^*, y_k^*)$$

on the test examples is minimized. $\Theta(a, b)$ is zero if $a = b$, otherwise it is one. Vapnik [Vapnik, 1998] gives bounds on the relative uniform deviation of training error

$$R_{train}(h) = \frac{1}{n}\sum_{i=1}^{n}\Theta(h(\vec{x}_i), y_i) \quad (4)$$

and test error

$$R_{test}(h) = \frac{1}{k}\sum_{i=1}^{k}\Theta(h(\vec{x}_i^*), y_i^{true}) \quad (5)$$

With probability $1-\eta$

$$R_{test}(h) \leq R_{train}(h) + \Omega(n, k, d, \eta) \quad (6)$$

where the confidence interval $\Omega(n, k, d, \eta)$ depends on the number of training examples n, the number of test examples k, and the VC-Dimension d of H (see [Vapnik, 1998] for details).

This problem of transductive inference may not seem profoundly different from the usual inductive setting studied in machine learning. One could learn a decision rule based on the training data and then apply it to the test data afterwards. Nevertheless, to solve the problem of estimating k binary values $y_1^*, ..., y_k^*$ we need to solve the more complex problem of estimating a function over a possibly continuous space. This may not be the best solution when the size n of the training sample (2) is small.

What information do we get from studying the test sample (3) and how can we use it? The training and the test sample split the hypothesis space H into a finite number of equivalence classes H'. Two functions from H belong to the same equivalence class if they both classify the training and the test sample in the same way. This reduces the learning problem from finding a function in the possibly infinite set H to finding one of finitely many equivalence classes H'. Most importantly, we can use these equivalence classes to build a structure of increasing VC-Dimension for *structural risk minimization* [Vapnik, 1998].

$$H'_1 \subset H'_2 \subset \cdots \subset H' \quad (7)$$

Unlike in the inductive setting, we can study the location of the test examples when defining the structure. Using prior knowledge about the nature of $P(\vec{x}, y)$ we can build a more appropriate structure and learn more quickly. What this means for text classification is analyzed in section 4. In particular, we can build the structure based on the margin of separating hyperplanes on both the training and the test data. Vapnik shows that with the size of the margin we can control the maximum number of equivalence classes (i. e. the VC-Dimension).

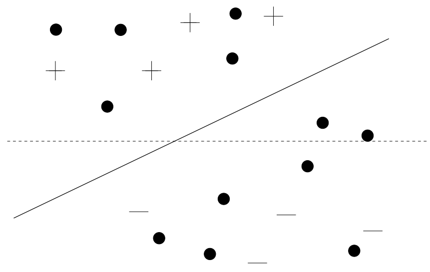

Figure 2: The maximum margin hyperplanes. Positive/negative examples are marked as $+/-$, test examples as dots. The dashed line is the solution of the inductive SVM. The solid line shows the transductive classification.

Theorem 1 ([Vapnik, 1998])
Consider hyperplanes $h(\vec{x}) = sign\{\vec{x}\cdot\vec{w}+b\}$ as hypothesis space H. If the attribute vectors of a training sample (2) and a test sample (3) are contained in a ball of diameter D, then there are at most

$$N_r < exp\Bigg(d\left(\frac{n+k}{d}+1\right)\Bigg), d = min\Bigg(a, \left[\frac{D^2}{\rho^2}\right]+1\Bigg)$$

equivalence classes which contain a separating hyperplane with

$$\forall_{i=1}^{n}\left|\frac{\vec{w}}{||\vec{w}||}\cdot\vec{x}_i+b\right| \geq \rho \qquad \forall_{j=1}^{k}\left|\frac{\vec{w}}{||\vec{w}||}\cdot\vec{x}_j^*+b\right| \geq \rho$$

(i.e. margin larger or equal to ρ). a is the dimensionality of the space, and $[b]$ is the integer part of b.

Note that the VC-Dimension does not necessarily depend on the number of features, but can be much lower than the dimensionality of the space. Let's use this structure based on the margin of separating hyperplanes. Structural risk minimization tells us that we get the smallest bound on the test error if we select the equivalence class from the structure element H'_i which minimizes (6). For linearly separable problems this leads to the following optimization problem [Vapnik, 1998].

OP 1 (Transductive SVM (lin. sep. case))
Minimize over $(y_1^*, ..., y_n^*, \vec{w}, b)$:

$$\frac{1}{2}||\vec{w}||^2$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i+b] \geq 1$$
$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j^*+b] \geq 1$$

Solving this problem means finding a labelling $y_1^*, ..., y_k^*$ of the test data and a hyperplane $< \vec{w}, b >$, so that this hyperplane separates both training and test data with maximum margin. Figure 2 illustrates this. To be able to handle non-separable data, we can introduce slack variables ξ_i similarly to the way we do with inductive SVMs.

OP 2 (Transductive SVM (non-sep. case))
Minimize over $(y_1^*, ..., y_n^*, \vec{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*)$:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^*$$

subject to:

$$\begin{aligned} &\forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \\ &\forall_{j=1}^{k} : y_j^*[\vec{w} \cdot \vec{x}_j^* + b] \geq 1 - \xi_j^* \\ &\forall_{i=1}^{n} : \xi_i > 0 \\ &\forall_{j=1}^{k} : \xi_j^* > 0 \end{aligned}$$

C and C^* are parameters set by the user. They allow trading off margin size against misclassifying training examples or excluding test examples. How this optimization problem can be solved efficiently is the subject of section 4.1.

4 What Makes TSVMs Especially well Suited for Text Classification?

The text classification task is characterized by a special set of properties. They are independent of whether text classification is used for information filtering, relevance feedback, or for assigning semantic categories to news articles.

High dimensional input space:
When learning text classifiers one has to deal with very many (more than 10,000) features, since each (stemmed) word is a feature.

Document vectors are sparse:
For each document, the corresponding document vector $\vec{x}_i$ contains few entries that are not zero.

Few irrelevant features:
Experiments in [Joachims, 1998] suggest that most words are relevant. So aggressive feature selection has to be handled with care, since it can easily lead to a loss of important information[1].

[1] This does not mean that aggressive feature selection cannot be beneficial for certain learning algorithms or certain tasks (see [Yang and Pedersen, 1997][Mladenič, 1998]).

	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 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.

Arguments from [Joachims, 1998] show that SVMs are especially well-suited for this setting, outperforming conventional methods substantially while also being more robust. Dumais et al. [Dumais et al., 1998] come to similar conclusions. TSVMs inherit most properties of SVMs so that the same arguments apply to TSVMs as well.

But how can TSVMs be any better? In the field of information retrieval it is well known that words in natural language occur in strong co-occurrence patterns (see [van Rijsbergen, 1977]). Some words are likely to occur together in one document, others are not. For examples, when asking the search engine Altavista about all documents containing the words `pepper` and `salt`, it returns 327,180 web pages. When asking for the documents with the words `pepper` and `physics`, we get only 4,220 hits, although `physics` is a more popular word on the web than `salt`. Many approaches in information retrieval try to exploit this cluster structure of text (see [van Rijsbergen, 1977]). And it is this co-occurrence information that TSVMs exploit as prior knowledge about the learning task.

Let's look at the example in figure 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 $D4$ (the test set)? Even if we did not understand the meaning of the words, we would classify $D2$ and $D3$ into class A, and $D3$ and $D4$ 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 topic, source, or style. For these type of classification tasks we find stronger co-occurrence patterns within categories than between

Algorithm TSVM:

Input:	– training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
	– test examples $\vec{x}^*_1, ..., \vec{x}^*_k$
Parameters:	– C,C^*: parameters from OP(2)
	– num_+: number of test examples to be assigned to class +
Output:	– predicted labels of the test examples $y^*_1, ..., y^*_k$

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$

`Classify the test examples using` $< \vec{w}, b >$. `The` num_+ `test examples with the highest value of` $\vec{w} * \vec{x}^*_j + b$ `are assigned to the class +` $(y^*_j := 1)$; `the remaining test examples are assigned to class` $-$ $(y^*_j := -1)$.

$C^*_- := 10^{-5};$ // some small number
$C^*_+ := 10^{-5} * \frac{num_+}{k - num_+};$
`while(`$(C^*_- < C^*) \parallel (C^*_+ < C^*)$`){` // Loop 1
  $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}^*_1, y^*_1)...(\vec{x}^*_k, y^*_k)], C, C^*_-, C^*_+);$
  `while(`$\exists m, l : (y^*_m * y^*_l < 0) \& (\xi^*_m > 0) \& (\xi^*_l > 0) \& (\xi^*_m + \xi^*_l > 2)$`) {` // Loop 2
    $y^*_m := -y^*_m;$ // take a positive and a negative test
    $y^*_l := -y^*_l;$ // example, switch their labels, and retrain
    $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}^*_1, y^*_1)...(\vec{x}^*_k, y^*_k)], C, C^*_-, C^*_+);$
  }
  $C^*_- := min(C^*_- * 2, C^*);$
  $C^*_+ := min(C^*_+ * 2, C^*);$
}
`return(`$y^*_1, ..., y^*_k$`);`

Figure 4: Algorithm for training Transductive Support Vector Machines.

different categories. In our example we analyzed the co-occurrence information in the test data and found two clusters. These clusters indicate different topics of $\{D1, D2, D3\}$ vs. $\{D4, D5, D6\}$, 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 dichotomies of $D2$ to $D5$ can be achieved with linear separators. Assigning $D2$ and $D3$ to class A and $D3$ and $D4$ to class B is the maximum margin solution (i.e. the solution of optimization problem OP1). We see that the maximum margin bias reflects our prior knowledge about text classification well. By analyzing the test set, we can exploit this prior knowledge for learning.

4.1 Solving the Optimization Problem

Training a transductive SVM means solving the (partly) combinatorial optimization problem OP2. For a small number of test examples, this problem can be solved optimally simply by trying all possible assignments of $y^*_1, ..., y^*_k$ to the two classes. However, this approach become intractable for test sets with more than 10 examples. Previous approaches using branch-and-bound search [Wapnik and Tscherwonenkis, 1979] push the limit to some extent, but still lag behind the needs of the text classification problem. The algorithm proposed next is designed to handle the large test sets common in text classification with 10,000 test examples and more. It finds an approximate solution to optimization problem OP2 using a form of local search.

The key idea of the algorithm is that it begins with a labeling of the test data based on the classification of an inductive SVM. Then it improves the solution by switching the labels of test examples so that the objective function decreases. The algorithm takes the training data and the test examples as input and outputs the predicted classification of the test examples. Besides the two parameters C and C^*, the user can specify the number of test examples to be assigned to class +. This allows trading-off recall vs. precision (see section 5.2). The following description of the algorithm covers only the linear case. A generalization to non-linear hypothesis spaces using kernels is straightforward.

The algorithm is summarized in figure 4. It starts with training an inductive SVM on the training data and classifying the test data accordingly. Then it uniformly increases the influence of the test examples by incrementing the cost-factors C^*_- and C^*_+ up to the user defined value of C^* (loop 1). The algorithm uses unbalanced costs C^*_- and C^*_+ to better accomodate the user defined ratio num_+. While the criterion in the condition of loop 2 identifies two examples for which changing the class labels leads to a decrease in the current objective function, these examples are switched.

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))
Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C^*_-\sum_{j:y^*_j=-1}\xi^*_j + C^*_+\sum_{j:y^*_j=1}\xi^*_j$$

subject to: $\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$
$\forall_{j=1}^{k} : y^*_j[\vec{w}\cdot\vec{x}^*_j + b] \geq 1 - \xi^*_j$

This optimization problem can be solved in its dual formulation using SVM^{light} [Joachims, 1999][2]. Especially designed for text classification, SVM^{light} can efficiently handle problems with many thousand support vectors, converges fast, and has minimal memory requirements. Let's finally look at an algorithmic property of the algorithm before evaluating its performance empirically in section 5.

Theorem 2 *Algorithm 1 converges in a finite number of steps.*

Proof: To prove this, it is necessary to show that loop 2 is exited after a finite number of iterations. This holds since the objective function of optimization problem OP2 decreases with every iteration of loop 2 as the following argument shows. The condition $y^*_m y^*_l < 0$ in loop 2 requires that the examples to be switched have different class labels. Let $y^*_m = 1$ so that we can write

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*_-\sum_{j:y^*_j=-1}\xi^*_i + C^*_+\sum_{j:y^*_j=1}\xi^*_i$$

$$= \frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + ... + C^*_+\xi^*_m + ... + C^*_-\xi^*_l + ...$$

$$> \frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + ... + C^*_-(2-\xi^*_m) + ... + C^*_+(2-\xi^*_l) + ...$$

$$= \frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + ... + C^*_-\xi^{*'}_m + ... + C^*_+\xi^{*'}_l + ...$$

It is easy to verify that the constraints of OP2 are fulfilled for the new values of y^*_m, y^*_l, $\xi^{*'}_m$, and $\xi^{*'}_l$ (potentially, after setting negative $\xi^{*'}_m$ or $\xi^{*'}_m$ to zero). The inequality holds due to the selection criterion in loop 2, since $\xi^{*'}_m = max(2 - \xi^*_m, 0) < \xi^*_l$ and $\xi^{*'}_l = max(2 - \xi^*_l, 0) < \xi^*_m$. This means that loop 2 is exited after a finite number of iterations, since there is only a finite number of permutations of the test examples. Loop 1 also terminates after a finite number of iterations, since C^*_- is bounded by C^*. □

5 Experiments

5.1 Test Collections

The empirical evaluation is done on three test collection. The first one is the Reuters-21578 dataset[3] collected from the Reuters newswire in 1987. The "ModApte" split is used, leading to a corpus of 9,603 training documents and 3,299 test documents. Of the 135 potential topic categories only the most frequent 10 are used, while keeping all documents. Both stemming and stop-word removal are used.

The second dataset is the WebKB collection[4] of WWW pages made available by the CMU text-learning group. Following the setup in [Nigam et al., 1998], only the classes `course`, `faculty`, `project`, and `student` are used. Documents not in one of these classes are deleted. After removing documents which just contain the relocation command for the browser, this leaves 4,183 examples. The pages from Cornell University are used for training, while all other pages are used for testing. Like in [Nigam et al., 1998], stemming and stop-word removal are not used.

The third test collection is taken from the Ohsumed corpus[5] compiled by William Hersh. From the 50,216 documents in 1991 which have abstracts, the first 10,000 are used for training and the second 10,000 are

[2] Available at http://www-ai.cs.uni-dortmund.de/svm_light

[3] Available at http://www.research.att.com/~lewis/reuters21578.html

[4] Available at http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data

[5] Available at ftp://medir.ohsu.edu/pub/ohsumed

	Bayes	SVM	TSVM
earn	78.8	91.3	95.4
acq	57.4	67.8	76.6
money-fx	43.9	41.3	60.0
grain	40.1	56.2	68.5
crude	24.8	40.9	83.6
trade	22.1	29.5	34.0
interest	24.5	35.6	50.8
ship	33.2	32.5	46.3
wheat	19.5	47.9	54.4
corn	14.5	41.3	43.7
average	35.9	48.4	60.8

Figure 5: P/R-breakeven point for the ten most frequent Reuters categories using 17 training and 3,299 test examples. Naive Bayes uses feature selection by empirical mutual information with local dictionaries of size 1,000. No feature selection was done for SVM and TSVM.

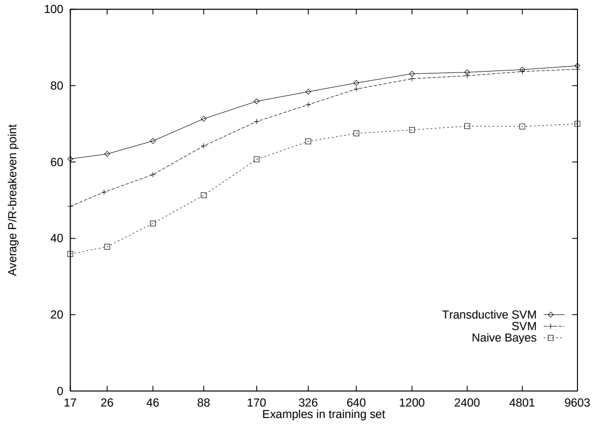

Figure 6: Average P/R-breakeven point on the Reuters dataset for different training set sizes and a test set size of 3,299.

used for testing. The task is to assign documents to one or multiple categories of the 5 most frequent MeSH "diseases" categories. A document belongs to a category if it is indexed with at least one indexing term from that category. Both stemming and stop-word removal are used.

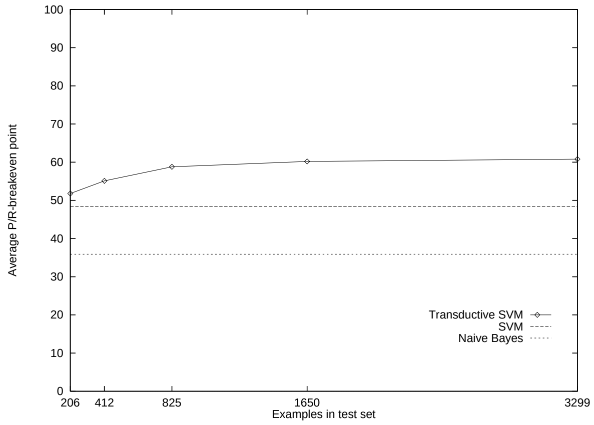

Figure 7: Average P/R-breakeven point on the Reuters dataset for 17 training documents and varying test set size for the TSVM.

5.2 Performance Measures

Since for both the Reuters dataset and the Ohsumed collection documents can be in multiple categories, the *Precision/Recall-Breakeven Point* is used as a measure of performance. The P/R-breakeven point is a common measure for evaluating text classifiers. It is based on the two well know statistics *recall* and *precision* widely used in information retrieval. Precision is the probability that a document predicted to be in class "+" truly belongs to this class. Recall is the probability that a document belonging to class "+" is classified into this class (see [Raghavan et al., 1989]). Both can be estimated from the contingency table.

Between high recall and high precision exists a tradeoff. The P/R-breakeven point is defined as that value for which precision and recall are equal. The transductive SVM uses the breakeven point for which the number of false positives equals the number of false negatives. For the inductive SVM and the Naive Bayes classifier the breakeven point is computed by varying the threshold on their "confidence value".

5.3 Results

The following experiments show the effect of using the transductive SVM instead of inductive methods. To provide a baseline for comparison, the results of the inductive SVM and a multinomial Naive Bayes classifier as described in [Joachims, 1997, McCallum and Nigam, 1998] are added. Where applicable, the results are averaged over a number of random training (test) samples.

Figure 5 gives the results for the Reuters dataset. For training sets of 17 documents and test sets of 3,299 documents, the transductive SVM leads to an improved performance on all categories, raising the av-

	Bayes	SVM	TSVM
course	57.2	68.7	93.8
faculty	42.4	52.5	53.7
project	21.4	37.5	18.4
student	63.5	70.0	83.8
average	46.1	57.2	62.4

Figure 8: Average P/R-breakeven points for the WebKB categories using 9 training and 3957 test examples. Naive Bayes uses a global dictionary with the 2,000 highest mutual information words. No feature selection was done for the SVM. Due to the large number of words, the TSVM used only those words which occur at least 5 times in the whole sample.

	Bayes	SVM	TSVM
pathology	39.6	41.8	43.4
Cardiovascular	49.0	58.0	69.1
Neoplasms	53.1	65.1	70.3
Nervous System	28.1	35.5	38.1
Immunologic	28.3	42.8	46.7
average	39.6	48.6	53.5

Figure 9: Average P/R-breakeven points for the Ohsumed categories using 120 training and 10,000 test examples. Here, Naive Bayes uses local dictionaries of 1,000 words selected by mutual information. No feature selection was done for the SVM. The TSVM again uses all words that occur at least 5 times in the whole sample.

erage of the P/R-breakeven points from 48.4 for the inductive SVM to 60.8. These averages correspond to the left-most points in figure 6. This graph 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. The influence of the test set size on the performance of the TSVM is displayed in figure 7. The bigger the test set, the larger the performance gap between SVM and TSVM. Adding more test examples beyond 3,299 is not likely to increase performance by much, since the graph is already very flat.

The results on the WebKB dataset are similar (figure 8). The average of the P/R-breakeven points increases from 57.2 to 62.4 by using the transductive approach. Nevertheless, for the category `project` the TSVM performs substantially worse, while the gain on the category `course` is large. Let's look at this in more detail. Figures 10 and 11 show how the per-

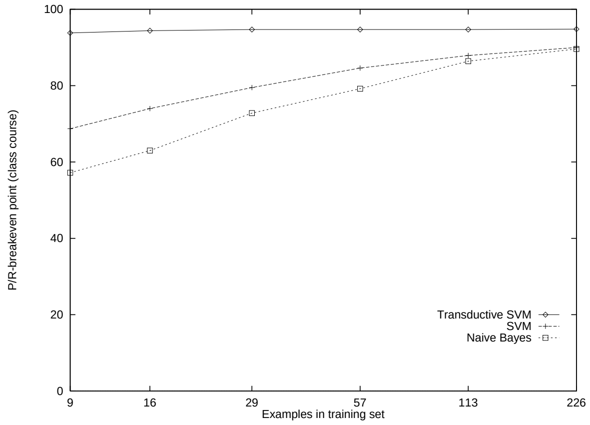

Figure 10: Average P/R-breakeven point on the WebKB category `course` for different training set sizes.

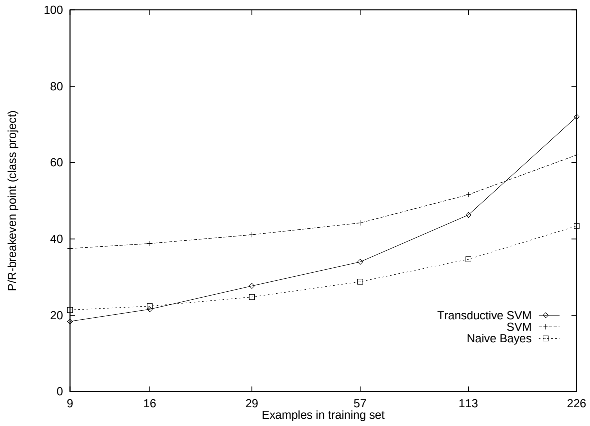

Figure 11: Average P/R-breakeven point on the WebKB category `project` for different training set sizes.

formance changes with increasing training set size for `course` and `project`. While for `course` the TSVM nearly reaches its peak performance immediately, it needs more training examples to surpass the inductive SVM for `project`. Why does this happen?

First, `project` is the least populous class. Among 9 training examples, there is only one from the `project` category. But more importantly, a look at the project pages reveals that many of them give a description of the project topic. My conjecture is that the margin along this "topic dimension" is large, and so the TSVM tries to separate the test data by topic. Only when there are enough project pages with different topics in the training set, the generalization along the project topic is ruled out. Most course pages at Cornell, on the other hand, do not give much topic information besides the title, but rather link to assignments, lecture notes etc. So the TSVM is not "distracted" by large margins along the topics.

The results in figure 9 for the Ohsumed collection complete the empirical evidence given in this paper, also supporting its point.

6 Related Work

Previously, Nigam et al. [Nigam et al., 1998] proposed another approach to using unlabeled data for text classification. They use a multinomial Naive Bayes classifier and incorporate unlabeled data using the EM-algorithm. One problem with using Naive Bayes is that its independence assumption is clearly violated for text. Nevertheless, using EM showed substantial improvements over the performance of a regular Naive Bayes classifier.

Blum and Mitchell's work on co-training [Blum and Mitchell, 1998] uses unlabeled data in a particular setting. They exploit the fact that, for some problems, each example can be described by multiple representations. WWW-pages, for example, can be represented as the text on the page and/or the anchor texts on the hyperlinks pointing to this page. Blum and Mitchell develop a boosting scheme which exploits a conditional independence between these representations.

Early empirical results using transduction can be found in [Vapnik and Sterin, 1977]. More recently, Bennett [Bennett, 1999] showed small improvements for some of the standard UCI datasets. For ease of computation, she conducted the experiments only for a linear-programming approach which minimizes the L_1 norm instead of L_2 and prohibits the use of kernels. Connecting to concepts of algorithmic randomness, [Gammerman et al., 1998] presented an approach to estimating the confidence of a prediction based on a transductive setting.

7 Conclusions and Outlook

This paper has introduced Transductive Support Vector Machines for text classification. Exploiting the particular statistical properties of text, it has identified that the margin of separating hyperplanes is a natural way to encode prior knowledge for learning text classifiers. By taking a transductive instead of an inductive approach, the test set can be used as an additional source of information about margins.

Introducing a new algorithm for training TSVMs that can handle 10,000 examples and more, this work presented empirical results on three test collections. On all data sets the transductive approach showed improvements over the currently best performing method, most substantially for small training samples and large test sets.

There are still a lot of open questions regarding transductive inference and SVMs. Particularly interesting is a PAC-style model for transductive inference to identify which concept classes benefit from transductive learning. How does the sample complexity behave for both the training and the test set? What is the relationship between the concept and the instance distribution? Regarding text classification in particular, is there a better basic representation for text, aligning margin and learning bias even better? Besides questions from learning theory, more research in algorithms for training TSVMs is needed. How well does the algorithm presented here approximate the global solution? Will the results get even better, if we invest more time into search? Finally, the transductive classification implicitly defines a decision rule. Is it possible to use this decision rule in an inductive fashion and will it perform well also on new test examples?

8 Acknowledgements

Many thanks to Katharina Morik for comments on this paper and to Tom Mitchell for the discussion. Thanks also to Ken Lang for providing some of the code. This work was supported by the DFG Collaborative Research Center on Statistics "Complexity Reduction in Multivariate Data" (SFB475).

References

[Bennett, 1999] Bennett, K. (1999). Combining support vector and mathematical programming methods for classification. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Annual Conference on Computational Learning Theory (COLT-98)*.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*.

[Gammerman et al., 1998] Gammerman, A., Vapnik, V., and Vowk, V. (1998). Learning by transduction. In *Conference on Uncertainty in Artificial Intelligence*, pages 148–156.

[Joachims, 1997] Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of International Conference on Machine Learning (ICML).*

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML).*

[Joachims, 1999] Joachims, T. (1999). Making large-scale svm learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI/ICML Workshop on Learning for Text Classification*. AAAI Press.

[Mladeniċ, 1998] Mladeniċ, D. (1998). Feature subset selection in text learning. In *European Conference on Machine Learning (ECML)*, Springer LNAI.

[Nigam et al., 1998] Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of the AAAI-98*.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137.

[Raghavan et al., 1989] Raghavan, V., Bollmann, P., and Jung, G. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

[van Rijsbergen, 1977] van Rijsbergen, C. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Vapnik and Sterin, 1977] Vapnik, V. and Sterin, A. (1977). On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503.

[Wapnik and Tscherwonenkis, 1979] Wapnik, W. and Tscherwonenkis, A. (1979). *Theorie der Zeichenerkennung*. Akademie Verlag, Berlin.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. (1997). A comparative study on feature selection in text categorization. In *International Conference on Machine Learning (ICML).*