Transductive Learning via Spectral Graph Partitioning

Thorsten Joachims TJ@CS.CORNELL.EDU
Cornell University, Department of Computer Science, Upson Hall 4153, Ithaca, NY 14853 USA

Abstract

We present a new method for transductive learning, which can be seen as a transductive version of the k nearest-neighbor classifier. Unlike for many other transductive learning methods, the training problem has a meaningful relaxation that can be solved globally optimally using spectral methods. We propose an algorithm that robustly achieves good generalization performance and that can be trained efficiently. A key advantage of the algorithm is that it does not require additional heuristics to avoid unbalanced splits. Furthermore, we show a connection to transductive Support Vector Machines, and that an effective Co-Training algorithm arises as a special case.

1. Introduction

For some applications, the examples for which a prediction is needed are already known when training the classifier. This kind of prediction is called *Transductive Learning* (Vapnik, 1998). An example of such a task is relevance feedback in information retrieval. In relevance feedback, users can give positive and negative examples for the kinds of documents they are interested in. These documents are the training examples, while the rest of the collection is the test set. The goal is to generalize from the training examples and find all remaining documents in the collection that match the users information need.

Why is the transductive setting different from the regular inductive setting? In the transductive setting, the learner can observe the examples in the test set and potentially exploit structure in their distribution. Several methods have been designed with this goal in mind. Vapnik introduced transductive SVMs (Vapnik, 1998) which were later refined by (Bennett, 1999) and (Joachims, 1999). Other methods are based on s-t mincuts (Blum & Chawla, 2001) or on multi-way cuts (Kleinberg & Tardos, 1999). Related is also the idea of Co-Training (Blum & Mitchell, 1998), which exploits structure resulting from two redundant representations. We will study what these approaches have in common and where they have problems. In particular, we will focus on s-t Mincuts, Co-Training, and TSVMs and show that they have undesirable biases that require additional, difficult to control heuristics.

To overcome this problem, we first propose and motivate a set of design principles for transductive learners. Following these principles, we introduce a new transductive learning method that can be viewed as a transductive version of the k nearest-neighbor (kNN) rule. One key advantage is that it does not require greedy search, but leads to an optimization problem that can be solved efficiently and globally optimally via spectral methods. We evaluate the algorithm empirically on 6 benchmarks, showing improved and more robust performance than for transductive SVMs. Furthermore, we show that Co-Training emerges as a special case and that the new algorithm performs substantially better than the original Co-Training algorithm.

2. Transductive Learning Model

The setting of transductive inference was introduced by V. Vapnik (see (Vapnik, 1998)). The learning task is defined on a fixed array X of n points $(\vec{x}_1, \vec{x}_2, ..., \vec{x}_n)$. Each data point has a desired classification $Y = (y_1, y_2, ..., y_n)$. For simplicity, let's assume the labels y_i are binary, i.e. $y_i \in \{+1, -1\}$, and that the data points $\vec{x}$ are vectors in $\Re^N$. For training, the learner receives the labels for a random subset

$$Y_l \subset [1..n] \qquad |Y_l| = l \tag{1}$$

of $l < n$ (training) data points. The goal of the learner is to predict the labels of the remaining (test) points in X as accurately as possible.

Vapnik gives bounds on the deviation of error rates observed in the training sample and in the test sample (Vapnik, 1998). The bounds depend on the capacity $d_X^{\mathcal{L}}$ of the hypothesis space $H_X^{\mathcal{L}}$, which can be measured, for example, as the number of different labelings the learner $\mathcal{L}$ can potentially produce on X. The smaller $d_X^{\mathcal{L}}$, the smaller the bound. Following

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

the idea of structural risk minimization (SRM) (Vapnik, 1998), one can consider a sequence of learners $\mathcal{L}_1, \mathcal{L}_2, ...$ with nested hypothesis spaces so that their capacity increases $d_X^{\mathcal{L}_1} < d_X^{\mathcal{L}_2} < ... < 2^n$. If the structure is well aligned with the learning task, the bound limits the generalization error. What information can we exploit to built a good structure?

3. Principles for Designing a Transductive Learner

In contrast to inductive learning, the transductive learner can analyze the location of all data points $\vec{x}_i \in X$, in particular those in the test set. Therefore, a transductive learner can structure its hypothesis space based on X. How can the location of the test points help design a good hypothesis space?

Imagine we knew the labels of all examples $\vec{x}_i \in X$, including both the training and the test examples. Let's call this the *perfect* classification. If we gave all these examples to an inductive learner $\mathcal{L}^{ind}$, it would be learning from a large training set and it might be reasonable to assume that $\mathcal{L}^{ind}$ learns an accurate classifier. For example, our experience might tell us that for text classification, SVMs typically achieve low prediction error if we have at least 10,000 training examples. Therefore, if X contains 10,000 data points, we would expect that an SVM given all labels Y achieves low leave-one-out cross-validation error $Err_{loo}^{\mathcal{L}^{SVM}}(X,Y)$ on (X,Y), since $Err_{loo}^{\mathcal{L}}(X,Y)$ is an (almost) unbiased estimate of the prediction error that typically has low variability.

How can this help a transductive learner that has access to only a small subset Y_l of the labels? Assuming that an inductive learner $\mathcal{L}^{ind}$ achieves low prediction error when trained on the full X implies that the perfect classification of the test set is highly self-consistent in terms of leave-one-out error. *If a transductive learner $\mathcal{L}^{trans}$ uses only those labelings of the test set for which the corresponding inductive learner $\mathcal{L}^{trans}$ has low leave-one-out error, the transductive learner will not exclude the perfect classification, while potentially excluding bad labelings.* This suggests the following SRM structure for a transductive learner

$$H_X^{\mathcal{L}_k^{trans}} = \left\{ (y_1, ..., y_n) | Err_{loo}^{\mathcal{L}^{ind}}(X,Y) \leq \frac{k}{n} \right\} \tag{2}$$

leading to a general principle for defining a transductive learner from an inductive learner. A transductive learner should label the test set so that

Postulate 1: it achieves low training error, and

Postulate 2: the corresponding inductive learner is highly self-consistent (e.g. low leave-one-out).

While initially not phrased in these terms, a transductive SVM (Vapnik, 1998)(Joachims, 1999) follows these two postulates (Joachims, 2002). A transductive SVM labels the test examples so that the margin is maximized. A large-margin SVM can be shown to have low leave-one-out error (Vapnik, 1998). Other transductive learning algorithms like transductive ridge-regression (Chapelle et al., 1999) and mincuts (Blum & Chawla, 2001) minimize leave-one-out error as well. However, leave-one-out is not the only measure of self-consistency. The co-training algorithm (Blum & Mitchell, 1998) maximizes consistency between two classifiers. It will be discussed in more detail in Section 5.2.

However, the following shows that Postulates 1 and 2 are not enough to define an effective transductive learner. Consider the case of k-Nearest Neighbor (kNN) (k odd). The kNN-rule makes a leave-one-out error on example $(\vec{x}_i, y_i)$, if the majority of the nearest neighbors are not from the same class. For a similarity-weighted kNN, we can define a margin-like quantity

$$\delta_i = y_i \frac{\sum_{j \in kNN(\vec{x}_i)} y_j w_{ij}}{\sum_{m \in kNN(\vec{x}_i)} w_{im}} \tag{3}$$

where w_{ij} reflects the similarity between x_i and x_j. The similarity weighted kNN-rule makes a leave-one-out error whenever $\delta_i \leq 0$. Therefore, an upper bound on the leave-one-out error is

$$Err_{loo}^{\mathcal{L}^{kNN}}(X,Y) \quad \leq \quad \sum_{i=1}^{n} (1 - \delta_i). \tag{4}$$

While it is computationally difficult to find a labeling of the test examples that minimizes the leave-one-out error of kNN while having a low training error, there are efficient algorithms for minimizing the upper bound (4). We can write this as the following optimization problem:

$$max_{\vec{y}} \quad \sum_{i=1}^{n} \sum_{j \in kNN(\vec{x}_i)} y_i y_j \frac{w_{ij}}{\sum_{m \in kNN(\vec{x}_i)} w_{im}} \tag{5}$$

$$s.t. \quad y_i = 1, \text{ if } i \in Y_l \text{ and positive} \tag{6}$$

$$y_i = -1, \text{ if } i \in Y_l \text{ and negative} \tag{7}$$

$$\forall j : y_j \in \{+1, -1\} \tag{8}$$

In matrix notation, the objective can be written equivalently as $\vec{y}^T A \vec{y}$ where $A_{ij} = \frac{w_{ij}}{\sum_{k \in kNN(\vec{x}_i)} w_{ik}}$ if $\vec{x}_j$ is among the k nearest neighbors of $\vec{x}_i$ and zero otherwise. While the problem is typically not convex, there are efficient methods for its solution. In particular, A can be thought of as the adjacency matrix of a graph, so that vertices represent examples and edges represent similarities (see Figure 1). At

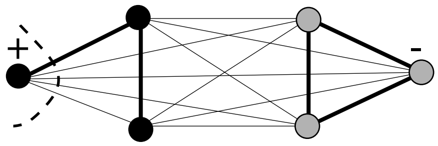

Figure 1. Mincut example.

the solution $\vec{y}^*$, denote with G^+ the set of examples (i.e. vertices) with $y_i = 1$, and with G^- those with $y_i = -1$. G^+ and G^- define a cut (bi-partitioning) of the graph. For an undirected graph, the cut-value $cut(G^+, G^-) = \sum_{i \in G^+} \sum_{i \in G^-} A_{ij}$ is the sum of the edge-weights across the cut. Since the maximum of (5) is determined by the matrix entries A_{ij} with $y_i y_j = -1$, minimizing $\sum_{y_i y_j = -1} A_{ij} = 2 \cdot cut(G^+, G^-)$ maximizes (5). Therefore, for undirected graphs, maximizing (5) subject to (6)-(8) is equivalent to finding the s-t mincut where the positive examples form the source, and the negative examples form the sink. The s-t mincut is a cut that separates positive and negative examples while minimizing the cut value. This connects to the transduction method of Blum and Chawla (Blum & Chawla, 2001). They use mincut/maxflow algorithms, starting from the intuition that the separation of positive and negative examples should put strongly connected examples into the same class. Blum and Chawla discuss the connection to leave-one-out error in the case of 1-NN.

While the s-t mincut algorithm is intuitively appealing, it easily leads to degenerate cuts. Consider the graph in Figure 1, where line thickness indicates edge weight A_{ij}. The graph contains two training examples which are labeled as indicated. All other nodes represent test examples. While we would like to split the graph according to the two dominant clusters, the s-t mincut leads to a degenerate solution that just cuts off the positive example. The behavior is due to the fact that s-t mincut minimizes the sum of the weight, and that balanced cuts have more potential edges to cut. While using a sparser graph would help in the example, this degenerate behavior also occurs for kNN graphs whenever the correct partitioning has more "wrong" neighbors than edges connecting the training examples to the unlabeled examples. This is practically always the case for sufficiently large numbers of unlabeled examples. Consider an (undirected) 100-NN graph, where each example has 99 of its neighbors in the correct class, and 1 in the incorrect class. If there is one positive training example, then this example will on average have 200 in/out edges. So, if there are more than 200 unlabeled examples, s-t mincut will return a degenerate cut even for such a strongly clustered graph. Since these degenerate cuts fulfill Postulate 1 (i.e. zero training error) and Postulate 2 (high self consistency in terms of leave-one-out), Postulates 1 and 2 do not yet specify a transductive learner sufficiently. One other reasonable constraint to put on the transductive solution is the following postulate.

Postulate 3: Averages over examples (e.g. average margin, pos/neg ratio) should have the same expected value in the training and in the test set.

Again, this postulate can be motivated using the perfect classification. For example, the average margin of kNN should fulfill

$$\sum_{Y_l} \left[\frac{1}{l} \sum_{i \in Y_l} \delta_i \right] P(Y_l) = \sum_{Y_l} \left[\frac{1}{n-l} \sum_{i \notin Y_l} \delta_i \right] P(Y_l) \quad (9)$$

for the perfect classification, since the distribution $P(Y_l)$ of drawing a training set is uniform over all subsets.

The s-t mincut violates Postulate 3 both for the pos/neg ratio, as well as for the average margin. In particular, training examples have negative margins, while test examples have large margin. Other functions are conceivable as well (Joachims, 2002). Blum and Chawla experiment with different heuristics for pre-processing the graph to avoid degenerate cuts. However, none appears to work well across all problems they study. Other transductive learning algorithms have similar degeneracies. For example, in transductive SVMs (Joachims, 1999) and in Co-Training (Blum & Mitchell, 1998) the fraction of positive examples in the test set has to be fixed a priori. Such constraints are problematic, since for small training sets an estimated pos/neg-ratio can be unreliable. How can the problem of degenerate cuts be avoided in a more principled way?

4. Normalized Graph Cuts with Constraints

The problem of s-t mincut can be traced to its objective function (5), which aims to minimize the sum of the edge weights cut through. The number of elements in the sum depends directly on the size of the two cut sets. In particular, the number of edges a cut with $|G^+|$ vertices on one side and $|G^-|$ vertices on the other side can potentially cut through is $|G^+||G^-|$. The s-t mincut objective is inappropriate, since it does not account for the dependency on the cut size. A natural way to normalize for cut size is by dividing the objective with $|G^+||G^-|$. Instead of minimizing the sum of the weights, the following optimization problem minimizes the average weight of the cut.

$$\max_{\vec{y}} \quad \frac{cut(G^+, G^-)}{|\{i : y_i = 1\}||\{i : y_i = -1\}|} \tag{10}$$
$$s.t. \quad y_i = 1, \text{ if } i \in Y_l \text{ and positive} \tag{11}$$
$$y_i = -1, \text{ if } i \in Y_l \text{ and negative} \tag{12}$$
$$\vec{y} \in \{+1, -1\}^n \tag{13}$$

This problem is related to the ratiocut (Hagen & Kahng, 1992). However, the traditional ratiocut problem is unsupervised, i.e. there are no constraints (11) and (12). Solving the unconstrained ratiocut is known to be NP hard (Shi & Malik, 2000). However, efficient methods based on the spectrum of the graph exist that give good approximations to the solution (Hagen & Kahng, 1992). The following will generalize these methods to the case of constrained ratiocuts for transduction.

Let's denote with $L = B - A$ the Laplacian of the graph with adjacency matrix A and diagonal degree matrix B, $B_{ii} = \sum_j A_{ij}$. We require that the graph is undirected, so that L is symmetric positive semi-definite. Following (Dhillon, 2001) and ignoring the constraints, the unsupervised ratiocut optimization problem can equivalently be written as

$$\min_{\vec{z}} \frac{\vec{z}^T L \vec{z}}{\vec{z}^T \vec{z}} \qquad \text{with } z_i \in \{\gamma_+, \gamma_-\} \tag{14}$$

where $\gamma_+ = \sqrt{\frac{|\{i:z_i<0\}|}{|\{i:z_i>0\}|}}$ and $\gamma_- = -\sqrt{\frac{|\{i:z_i>0\}|}{|\{i:z_i<0\}|}}$. It is straightforward to verify that $\vec{z}^T \vec{z} = n$ and $\vec{z}^T 1 = 0$ for every feasible point. While this problem is still NP hard, the minimum of its real relaxation

$$\min_{\vec{z}} \quad \vec{z}^T L \vec{z} \tag{15}$$
$$\text{s.t.} \quad \vec{z}^T 1 = 0 \text{ and } \vec{z}^T \vec{z} = n \tag{16}$$

is equal to the second eigenvalue of L and the corresponding eigenvector is the solution. Using this solution of the relaxed problem as an approximation to the solution of (14) is known to be effective in practice.

Moving to the supervised ratiocut problem, we propose to include constraints (11) and (12) by adding a quadratic penalty to the objective function.

$$\min_{\vec{z}} \quad \vec{z}^T L \vec{z} + c(\vec{z} - \vec{\gamma})^T C (\vec{z} - \vec{\gamma}) \tag{17}$$
$$s.t. \quad \vec{z}^T 1 = 0 \text{ and } \vec{z}^T \vec{z} = n \tag{18}$$

For each labeled example, the corresponding element of $\vec{\gamma}$ is equal to $\hat{\gamma}_+$ ($\hat{\gamma}_-$) for positive (negative) examples, and it is zero for test examples. $\hat{\gamma}_+$ and $\hat{\gamma}_-$ are estimates of γ_+ and γ_- (e.g. based on the number of observed positive and negative examples in the training data). We will see later that these estimates do not need to be very precise. c is a parameter that trades off training error versus cut-value, and C is a diagonal cost matrix that allows different misclassification costs for each example. Taking the eigendecomposition $L = U \Sigma U^T$ of the Laplacian, one can introduce a new parameter vector $\vec{w}$ and substitute $\vec{z} = U\vec{w}$. Since the eigenvector of the smallest eigenvalue of a Laplacian is always $\vec{1}$, the constraint (16) becomes equivalent to setting $w_1 = 0$. Let V (D) be the matrix with all eigenvectors U (eigenvalues Σ) except the smallest one, then we get the following equivalent optimization problem.

$$\min_{\vec{w}} \quad \vec{w}^T D \vec{w} + c(V\vec{w} - \vec{\gamma})^T C (V\vec{w} - \vec{\gamma}) \tag{19}$$
$$s.t. \quad \vec{w}^T \vec{w} = n \tag{20}$$

Defining $G = (D + cV^T C V)$ and $\vec{b} = cV^T C \vec{\gamma}$, the objective function can also be written as $\vec{w}^T G \vec{w} - 2\vec{b}^T \vec{w} + c\vec{\gamma}^T C \vec{\gamma}$, where the last term can be dropped since it is constant. Following the argument in (Gander et al., 1989), Problem (19)-(20) is minimized for $\vec{w}^* = (G - \lambda^* I)^{-1} \vec{b}$, where λ^* is the smallest eigenvalue of

$$\begin{bmatrix} G & -I \\ -\frac{1}{n}\vec{b}\vec{b}^T & G \end{bmatrix}. \tag{21}$$

I is the identity matrix. From this we can compute the optimal value of (17) and (18) as $\vec{z}^* = V\vec{w}^*$, producing a predicted value for each example. We can use this value to rank the test examples, or use a threshold to make hard class assignment. An obvious choice for the threshold is the midpoint $\Theta = \frac{1}{2}(\hat{\gamma}_+ + \hat{\gamma}_-)$ which we will use in the following, but more refined methods are probably more appropriate.

5. Spectral Graph Transducer

The basic method for computing supervised ratiocuts suggests the following algorithm for transductive learning, which we call a Spectral Graph Transducer (SGT). An implementation is available at `http://sgt.joachims.org`. Input to the algorithm are the training labels Y_l, and a weighted undirected graph on X with adjacency matrix A. In the following, we will use the similarity-weighted k nearest-neighbor graph

$$A'_{ij} = \begin{cases} \frac{sim(\vec{x}_i, \vec{x}_j)}{\sum_{\vec{x}_k \in knn(\vec{x}_i)} sim(\vec{x}_i, \vec{x}_k)} & \text{if } \vec{x}_j \in knn(\vec{x}_i) \\ 0 & \text{else} \end{cases} \tag{22}$$

over X symmetricized by $A = A' + A'^T$. The first step preprocesses the graph, which has to be done only once:

- Compute diagonal degree matrix B, $B_{ii} = \sum_j A_{ij}$
- Compute Laplacian $L = B - A$, or compute normalized Laplacian $L = B^{-1}(B - A)$ corresponding to the normalized-cut criterion of (Shi & Malik, 2000).

- Compute the smallest 2 to $d+1$ eigenvalues and eigenvectors of L and store them in D and V.
- To normalize the spectrum of the graph, replace the eigenvalues in D with some monotonically increasing function. We use $D_{ii} = i^2$ (see also (Chapelle et al., 2002)). Fixing the spectrum of the graph in this way abstracts, for example, from different magnitudes of edge weights, and focuses on the ranking among the smallest cuts.

The following steps have to be done for each new training set:

- Estimate $\hat{\gamma}_+ = \sqrt{\frac{l_-}{l_+}}$ and $\hat{\gamma}_- = -\sqrt{\frac{l_+}{l_-}}$, where l_+ (l_-) is the number of positive (negative) training examples. Set $\gamma_i = \hat{\gamma}_+$ ($\gamma_i = \hat{\gamma}_-$) for positive (negative) examples.
- To give equal weight to positive and negative examples, set cost of positive (negative) training examples to $C_{ii} = \frac{l}{2l_+}$ ($C_{ii} = \frac{l}{2l_-}$). $C_{ii} = 0$ for all test examples.
- Compute $G = (D + cV^TCV)$ and $\vec{b} = cV^TC\vec{\gamma}$ and find smallest eigenvalue λ^* of Equation (21).
- Compute predictions as $\vec{z}^* = V(G - \lambda^* I)^{-1}\vec{b}$, which can be used for ranking.
- Threshold $\vec{z}^*$ wrt. $\Theta = \frac{1}{2}(\hat{\gamma}_+ + \hat{\gamma}_-)$ to get hard class assignments $y_i = sign(z_i - \Theta)$.

5.1. Connection to Transductive SVMs

The following argument shows the relationship of the SGT to a TSVM. We consider the TSVM as described in (Joachims, 1999). For hyperplanes passing through the origin, the TSVM optimizes

$$\begin{aligned} min_{\vec{y}}\ max_{\vec{\alpha}\geq 0}\ & 1^T\alpha - \frac{1}{2}\vec{\alpha}^T diag(\vec{y})\ A\ diag(\vec{y})\vec{\alpha} && (23)\\ s.t.\ & y_i = 1,\ \text{if } i \in Y_l \text{ and positive} && (24)\\ & y_i = -1,\ \text{if } i \in Y_l \text{ and negative} && (25)\\ & \vec{y} \in \{+1,-1\}^n && (26)\\ & |\{i : y_i = 1\}| = p. && (27) \end{aligned}$$

For our analysis, we simplify this problem by adding the constraint $\alpha_1 = \alpha_2 = ... = \alpha_n$. Since the objective (23) can now be written as $n\alpha - \frac{1}{2}\alpha^2\vec{y}^T A\vec{y}$ where α is a scalar, the maximum is achieved for $\alpha^* = \frac{n}{\vec{y}^T A\vec{y}}$. Substituting the solution into the objective shows that the value of the maximum is $\frac{1}{2}\frac{n^2}{\vec{y}A\vec{y}}$. This shows that the simplified TSVM problem is equivalent to an s-t mincut on graph A, where the balance of the cut is fixed by (27). The SGT removes the need for fixing the exact cut size a priori.

5.2. Connection to Co-Training

Co-training can be applied, if there are two redundant representations A and B of all training examples (Blum & Mitchell, 1998). The goal is to train two classifiers h^A and h^B, one for each representation, so that their predictions are maximally consistent, i.e. $h^A(\vec{x}) = h^B(\vec{x})$ for most examples $\vec{x}$. With this goal, Blum and Mitchell propose a greedy algorithm that iteratively labels examples for which one of the current classifiers is most confident. However, also for this algorithm the ratio of predicted positive and negative examples in the test set must be fixed a priori to avoid degenerate solutions.

Co-training emerges as a special case of the SGT. Consider a kNN classifiers for each of the two representations and note that $\sum_{i=1}^n (2 - \delta_i^A - \delta_i^B)$ (see Eq. (3)) is an upper bound on the number of inconsistent predictions. Therefore, to maximize consistency, we can apply the SGT to the graph that contains k links for the kNN from representation A, as well as another k links per example for the kNN from representation B.

5.3. Connections to other Work

Several other approaches to using unlabeled data for supervised learning exist. Most related is the approach to image segmentation described in (Yu et al., 2002). They aim to segment images under higher level constraints. One difference is that they arrive at constrained cut problems where all the constraints are homogeneous, leading to a different technique for their solution. The spectrum of the Laplacian is also considered in the recent work in (Chapelle et al., 2002) and (Belkin & Niyogi, 2002). They use the leading eigenvectors for feature extraction and the design of kernels. In addition, Chapelle et al. use the same normalization of the spectrum. Szummer and Jaakkola apply short random walks on the kNN graph for labeling test examples, exploiting that a random walk will less likely cross cluster boundaries, but stay within clusters (Szummer & Jaakkola, 2001). There might be an interesting connection to the SGT, since the normalized cut minimizes the transition probability of a random walk over the cut (Meila & Shi, 2001). This might also lead to a connection to the generative modeling approach of Nigam et al., where the label of each test example is a latent variable (Nigam et al., 2000).

6. Experiments

To evaluate the SGT, we performed experiments on six datasets and report results for all of them. The datasets are the ten most frequent categories from the Reuters-21578 text classification collection following the setup in (Joachims, 1999), the UCI Reposi-

Table 1. PRBEP (macro-) averages for five datasets and training sets of size l. n is the total number of examples, and N is the number of features.

Task	N	n	l	SGT	kNN	TSVM	SVM
Optdigits	64	1797	10	83.4	62.4	61.5	61.6
Reuters	11846	3299	17	57.0	46.7	51.5	49.1
Isolet	617	7797	20	55.4	47.4	-	46.3
Adult	123	32561	10	54.5	46.0	47.3*	46.2
Ionosphere	34	351	10	79.6	76.7	80.6	80.6

tory datasets OPTDIGITS (digit recognition), ISOLET (speech recognition), IONOSPHERE, as well as the ADULT data in a representation produced by John Platt. To evaluate the Co-Training connection, we use the WebKB data of Blum and Mitchell with TFIDF weighting.

The goal of the empirical evaluation is threefold. First, we will evaluate whether the SGT can make use of the transductive setting by comparing it against inductive learning methods, in particular kNN and a linear SVM. Second, we compare against existing transduction methods, in particular a TSVM. And third, we evaluate how robustly the SGT performs over different data sets and parameter settings.

For all learning tasks, both kNN and the SGT (with normalized Laplacian $L = B^{-1}(B - A)$) use the cosine as the similarity measure. While this is probably suboptimal for some tasks, the following results indicate that it is a reasonable choice. Furthermore, it equally affects both kNN and the SGT, so that relative comparisons between the two remain valid. If an example has zero cosine with all other examples, it is randomly connected to k nodes with uniform weight.

To make sure that performance differences against the other learning methods are not due to bad choices for their parameters, we give the conventional learning methods (i.e. kNN, SVM, TSVM) an unfair advantage. For these methods we report the results for the parameter setting with the best average performance *on the test set.* For the SGT, on the other hand, we chose $c = 3200$ and $d = 80$ constant over all datasets. The choice of k for building the k nearest-neighbor graph is discussed below.

All results reported in the following are averages over 100 stratified transductive samples[1]. So, all substantial differences are also significant with respect to this distribution. Samples are chosen so that they contain at least one positive example. While we report error rate where appropriate, we found it too unstable for a fair comparison with very unbalanced class ratios. We therefore use rank-based measures for most comparisons. The most popular such measure in information retrieval is the Precision/Recall curve, which we summarize by its Break-Even Point (PRBEP) (see e.g. (Joachims, 1999)). For tasks with multiple classes (i.e. Reuters, OPTDIGITS, and ISOLET), we summarize the performance by reporting an average over all classes (i.e. macro-averaging).

[1]An exception is the TSVM on Adult and Isolet, where 100 samples would have been prohibitively expensive.

Table 2. PRBEP / error rate on the WebKB "course" data averaged over training sets of size 12.

Features	SGT	kNN	TSVM	SVM	B&M
cotrain	93.4/3.3	-/-	-/-	-/-	-/5.0
page+link	87.1/5.9	79.7/10.1	92.0/4.3	82.3/20.3	-/-
page	86.2/6.2	73.2/13.3	91.4/4.6	75.4/21.6	-/12.9
link	75.9/22.1	71.9/13.1	79.9/8.9	75.0/18.5	-/12.4

Does the Unlabeled Data Help Improve Prediction Performance? The results are summarized in Table 1. On all tasks except Ionosphere, the SGT gives substantially improved prediction performance compared to the inductive methods. Also, the SGT performs better than kNN (as its inductive variant), on each individual binary task of Reuters and Optdigits. For Isolet, the SGT performs better than kNN on 23 of the 26 binary tasks.

The improvements of the TSVM are typically smaller. On Adult, the TSVM was too inefficient to be applied to the full dataset, so that we give the results for a subsample of size 2265. The TSVM failed to produce reasonable results for Isolet. While the TSVM does improve performance on Reuters, the improvement is less than reported in (Joachims, 1999). There, the assumption is made that the ratio of positive to negative examples in the test set is known accurately. However, this is typically not the case and we use an estimate based on the training set in this work. If the true fraction is used, the TSVM achieves a performance of 62.3. While (Joachims, 2002) proposes measures to detect when the wrong fraction was used, this can only be done after running the TSVM. Repeatedly trying different fractions is prohibitively expensive.

How Effective is the SGT for Co-Training? Table 2 shows the results for the co-training on WebKB. We built the graph with 200NN from the page and 200NN from the links. The table compares the co-training setting with just using the page or the links, and a combined representation where both feature sets are concatenated. The SGT in the co-training setting achieves the highest performance. The TSVM also gives large improvements compared to the inductive methods, outperforming the SGT. However, the TSVM cannot take advantage of the co-training set-

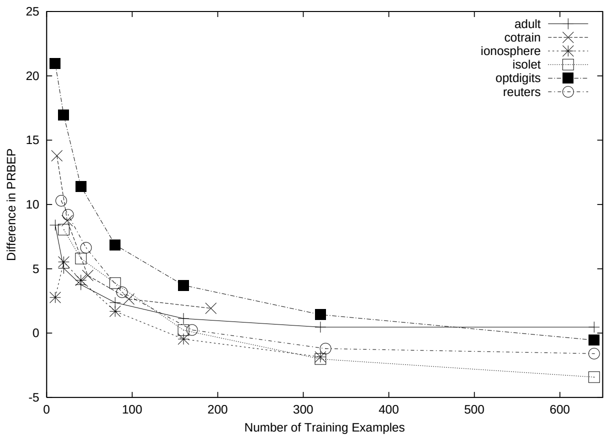

Figure 2. Amount by which the PRBEP of the SGT exceeds the PRBEP of the optimal kNN.

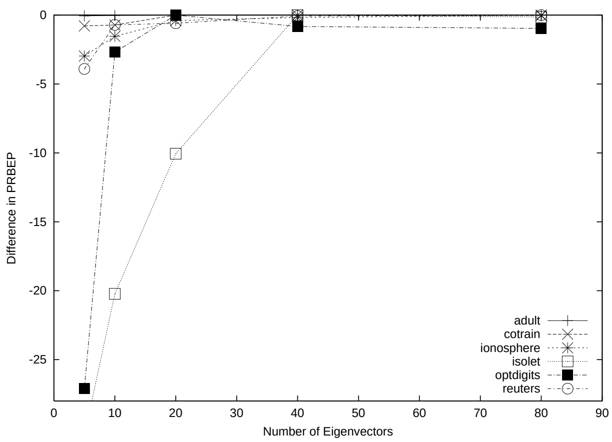

Figure 3. Amount by which the PRBEP of the SGT is lower for a particular number of eigenvectors d compared to the PRBEP of the best d (l as in Tables 1 and 2).

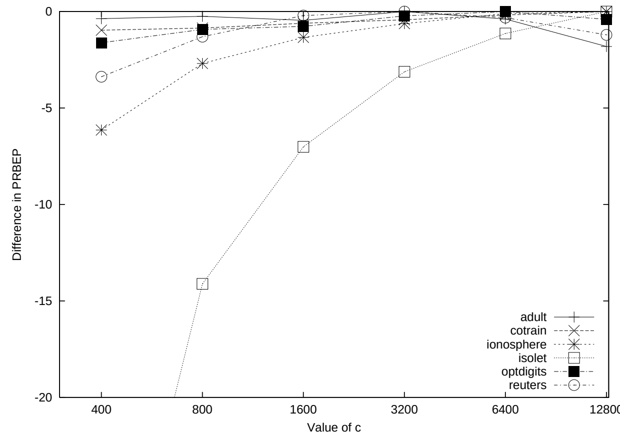

Figure 4. Amount by which the PRBEP of the SGT is lower for a particular value of c compared to the best c.

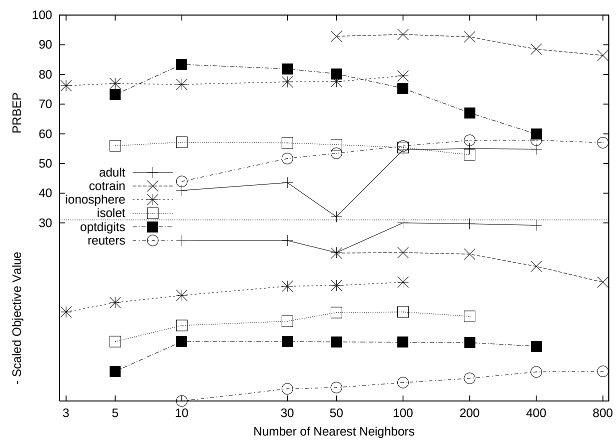

Figure 5. Average PRBEP and the average normalized value of the objective function (vertically flipped and positioned to fit the graph) for the SGT depending on the number of nearest neighbors (l as in Tables 1 and 2).

ting. The results from (Blum & Mitchell, 1998) are added in the last column.

For which Training Set Sizes is Transductive Learning most Effective? Figure 2 shows the difference in average PRBEP between the SGT and kNN for different training set sizes. For all learning tasks, the performance improvement is largest for small training sets. For larger sets, the performance of the SGT approaches that of kNN. The negative values are largely due to the bias from selecting the parameters of kNN based on the test set. If this is also allowed for the SGT, the differences vanish or become substantially smaller.

How Sensitive is the SGT to the Choice of the Number of Eigenvectors? Figure 3 plots the loss of PRBEP compared to the PRBEP achieved on the test set for the optimal number of eigenvectors. On all tasks, the SGT achieves close to optimal performance, if more than 40 eigenvectors are included. We conclude that $d \geq 40$ should be sufficient for most tasks.

How Sensitive is the SGT to the Choice of the Error Parameter? Analogous to Figure 3, Figure 4 plots the loss of PRBEP compared to the best value of c. Due to the normalization of the spectrum of the Laplacian, the optimum values of c are comparable between datasets. For most tasks, the performance is less than two PRBEP points away from the optimum for any c between 1600 and 12800. An exception is Isolet, which requires larger values. We conclude that any c between 3200 and 12800 should give reasonable performance for most tasks.

How Sensitive is the SGT to the Choice of the Graph? Unlike c and d, the choice of k for building the k nearest-neighbor graph has a strong influence on the performance. The top part of Figure 5 shows average PRBEP depending on k. How should we select k? For small training set sizes (often only one positive example), cross-validation is not feasible. However, the value of the objective function can be interpreted as a measure of capacity and might be suitable for model selection. The bottom half of Figure 5 shows the average value of the objective function after normalization. In particular, the objective value o_{ik} for training set i and choice of k is normalized to $o_{ik}^{norm} = \frac{o_{ik}}{min_k o_{ik}}$. The average normalized objective tracks the performance curve very well, suggesting that there might be an interesting connection between this value and the capacity of the SGT. For all experiments reported in the previous section, we used the value of k that minimizes the average normalized objective. For Adult, this is $k = 100$, for Reuters $k = 800$, for Optdigits $k = 10$, for Isolet $k = 100$, for Ionosphere $k = 100$, and for Co-Training $k = 2 * 200$. Such a kind of model selection might be particularly useful for tasks like relevance feedback in information retrieval, where there are many learning tasks with few examples on the same collection of objects.

How Efficiently can the SGT be Trained? Due to our naive implementation, most of the time is spent on computing the k-NN graph. However, this can be sped up using appropriate data structures like inverted indices or KD-trees. Computing the 81 smallest eigenvalues takes approximately 1.5 minutes for a task with 10,000 examples and 100 neighbors on a 1.7GHz CPU using Matlab. However, these preprocessing steps have to be performed only once. Training on a particular training set and predicting 10,000 test examples takes less than one second.

7. Conclusions

We studied existing transductive learning methods and abstracted their principles and problems. Based on this, we introduced a new transductive learning method, which can be seen as the a transductive version of the kNN classifier. The new method can be trained efficiently using spectral methods. We evaluated the classifier on a variety of test problems showing substantial improvements over inductive methods for small training sets. Unlike most other algorithms that use unlabeled data, it does not need additional heuristics to avoid unbalanced splits. Furthermore, since it does not require greedy search, it is more robust than existing methods, outperforming the TSVM on most tasks. Modeling the learning problem as a graph offers a large degree of flexibility for encoding prior knowledge about the relationship between individual examples. In particular, we showed that Co-Training arises as a special case and that the new algorithm outperforms the original Co-Training algorithm. The algorithm opens interesting areas for research. In particular, is it possible to derive tight, sample dependent capacity bounds based on the cut value? Furthermore, it is interesting to consider other settings beyond co-training that can be modeled as a graph (e.g. temporal drifts in the distribution, co-training with more than two views, etc.).

This research was supported in part by the NSF projects IIS-0121175 and IIS-0084762 and by a gift from Google. Thanks to Lillian Lee, Filip Radlinski, Bo Pang, and Eric Breck for their insightful comments.

References

Belkin, M., & Niyogi, P. (2002). Semi-supervised learning on manifolds. *NIPS*.

Bennett, K. (1999). Combining support vector and mathematical programming methods for classification. In B. Schölkopf et al. (Eds.), *Advances in kernel methods - support vector learning*. MIT-Press.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincut. *ICML*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT*.

Chapelle, O., Vapnik, V., & Weston, J. (1999). Transductive inference for estimating values of functions. *NIPS*.

Chapelle, O., Weston, J., & Schoelkopf, B. (2002). Cluster kernels for semi-supervised learning. *NIPS*.

Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *KDD Conference*.

Gander, W., Golub, G., & von Matt, U. (1989). A constrained eigenvalue problem. *Linear Algebra and its Applications*, *114/115*, 815–839.

Hagen, L., & Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD*, *11*, 1074–1085.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML*.

Joachims, T. (2002). *Learning to classify text using support vector machines – methods, theory, and algorithms*. Kluwer.

Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *FOCS*.

Meila, M., & Shi, J. (2001). A random walks view of spectral segmentation. *AISTATS*.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, *39*, 103 – 134.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *PAMI*.

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with markov random walks. *NIPS*.

Vapnik, V. (1998). *Statistical learning theory*. Wiley.

Yu, S. X., Gross, R., & Shi, J. (2002). Concurrent object recognition and segmentation by graph partitioning. *NIPS*.