Semi-Supervised Classification by Low Density Separation

Olivier Chapelle, Alexander Zien
Max Planck Institute for Biological Cybernetics
72076 Tübingen, Germany

Abstract

We believe that the cluster assumption is key to successful semi-supervised learning. Based on this, we propose three semi-supervised algorithms: 1. deriving graph-based distances that emphazise low density regions between clusters, followed by training a standard SVM; 2. optimizing the Transductive SVM objective function, which places the decision boundary in low density regions, by gradient descent; 3. combining the first two to make maximum use of the cluster assumption. We compare with state of the art algorithms and demonstrate superior accuracy for the latter two methods.

1 INTRODUCTION

The goal of semi-supervised classification is to use unlabeled data to improve the generalization. The *cluster assumption* states that the decision boundary should not cross high density regions, but instead lie in low density regions. We believe that virtually all successful semi-supervised algorithms utilize the cluster assumption, though most of the time indirectly.

For instance, manifold learning algorithms (e.g., [1]) construct decision functions that vary little along the manifolds occupied by the data. Often, different classes form separate manifolds. Then, manifold learning indirectly implements the cluster assumption by not cutting the manifolds.

The Transductive SVM [20] implements the cluster assumption more directly by trying to find a hyperplane which is far away from the unlabeled points. In our opinion, the rationale for maximizing the margin is very different for the labeled and unlabeled points:

- For the labeled points, it implements regularization [20]. Intuitively, the large margin property makes the classification robust with respect to perturbations of the data points [6].
- For the unlabeled points, the margin maximization implements the cluster assumption. It is not directly related to regularization (in this respect, we have a different view than Vapnik [20]). Consider for instance an example where the cluster assumption does not hold: a uniform distribution of input points. Then the unlabeled points convey almost no information, and maximizing the margin on those points is useless (and can even be harmful).

TSVM might seem to be the perfect semi-supervised algorithm, since it combines the powerful regularization of SVMs with a direct implementation of the cluster assumption. However, its main drawback is that the objective function is non-convex and thus difficult to minimize. Consequently, optimization heuristics like `SVMlight` [12] sometimes give bad results and are often criticized. The main points of this paper are:

- The objective function of TSVM is appropriate, but different ways of optimizing it can lead to very different results. Thus, it is more accurate to criticize a given implementation of the TSVM rather than the objective function itself.
- The search for a low density decision boundary is difficult. The task of the TSVM algorithm can be eased by changing the data representation.

To substantiate our claims, we develop and assess corresponding algorithms. Firstly, we propose a graph-based semi-supervised learning method exploiting the cluster assumption. Secondly, it is shown that a gradient descent on the primal formulation of the TSVM objective function performs significantly better then the optimization strategy pursued in `SVMlight` [12]. Finally, by combining these two ideas in one algorithm, we are able to achieve clearly superior generalization accuracy.

2 ALGORITHMS

Let the given data consist of n labeled data points $\mathbf{x}_i, 1 \le i \le n$, and m unlabeled data points $\mathbf{x}_i, n+1 \le i \le n+m$. For simplicity, we assume that the labels $y_i, 1 \le i \le n$, are binary, i.e. $y_i = \pm 1$; for multi-class problems, we use the one-against-rest scheme that is common for SVMs (e.g., [17]).

In the following sections, we describe two different ways to enforce the cluster assumption in SVM classification and how they can be implemented.

2.1 GRAPH-BASED SIMILARITIES

Let the graph $\mathsf{G} = (V, E)$ be derived from the data such that the nodes are the data points, $V = \{\mathbf{x}_i\}$. If sparsity is desired, edges are placed between nodes that are nearest neighbors (NN), either thresholding the degree (k-NN)[1] or the distance (ϵ-NN). Many semi-supervised learning methods operate on nearest neighbor graphs, see e.g. [1, 14, 18, 23, 22]. Usually they do not require the data points themselves, but only their pairwise distances along the edges. In the following we assume that the edges $(i, j) \in E$ are weighted by Euclidean distances $d(i, j) := ||\mathbf{x}_i - \mathbf{x}_j||_2$ (missing edges correspond to $d(i, j) = \infty$), although other distances are possible as well.

Many graph-based semi-supervised algorithms work by enforcing smoothness of the solution with respect to the graph, i.e. that the output function varies little between connected nodes. Here we use the graph to derive pairwise similarities between points, thereby "squeezing" the distances in high density regions while leaving them in low density regions. This idea has been proposed before, e.g. in [5, 21] and [4, section 3]. It has been implemented and used in `Isomap` [19], cluster kernels [7], and connectivity clustering [10].

2.1.1 Motivation

According to the cluster assumption, the decision boundary should preferably not cut clusters. A way to enforce this for similarity-based classifiers is to assign low similarities to pairs of points that lie in different clusters. To do so, we construct a Parzen window density estimate with a Gaussian kernel of width $\frac{1}{\sqrt{2}}\sigma$,

$$\hat{p}(\mathbf{x}') \quad = \quad \frac{1}{\sqrt{\pi}\sigma} \sum_{i=1}^{n+m} \exp\left(-\frac{||\mathbf{x}' - \mathbf{x}_i||^2}{\sigma^2}\right).$$

If two points are in the same cluster, it means that there exists a continuous connecting curve that only goes through regions of high density; if two points are in different clusters, every such curve has to traverse a density valley. We can thus define the similarity of two points by maximizing over all continuous connecting curves the minimum density along the connection, but this is hard to compute.

Two observations, illustrated in Figure 1, allow to approximate the above similarity with paths on a graph: (a) An optimal connecting curve can be well approximated by conjoining short line segments that directly connect points. (b) The minimum density is assumed at the middle of a line segment, and dominated by the closest points.

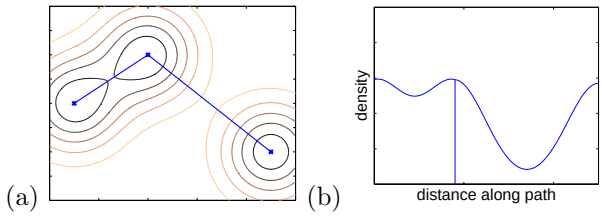


Figure 1: Optimal connecting curves are well approximated by paths of short distance edges on a graph.

2.1.2 A density-sensitive distance measure

Formally, we define $\mathsf{p} \in V^l$ to be a path of length $l =: |\mathsf{p}|$ on a graph $\mathsf{G} = (V, E)$, if $(p_k, p_{k+1}) \in E$ for $1 \le k < |\mathsf{p}|$. A path p is said to connect the nodes p_1 and $\mathsf{p}_{|\mathsf{p}|}$; let $\mathsf{P}_{i,j}$ denote the set of all paths connecting $\mathbf{x}_i$ and $\mathbf{x}_j$. We obtain

$$\begin{aligned} &\max_{\mathsf{p} \in \mathsf{P}_{i,j}} \min_{k<|\mathsf{p}|} \hat{p}\left(\frac{1}{2}(x_{\mathsf{p}_k} + x_{\mathsf{p}_{k+1}})\right) \\ &\quad \approx \quad c \cdot \exp\left[-\frac{1}{2\sigma^2}\left(\min_{\mathsf{p} \in \mathsf{P}_{i,j}} \max_{k<|\mathsf{p}|} d(\mathsf{p}_k, \mathsf{p}_{k+1})\right)^2\right] \quad (1) \\ &\quad \equiv \quad k(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

This k, called "connectivity kernel", is positive definite and was suggested for clustering previously [10].

The kernel values do not depend on the length of the paths, which may lead to the connection of otherwise separated clusters by single outliers ("bridge" points). To avoid this problem, we "soften" the max in Equation (1) by replacing it with

$$\mathrm{smax}^\rho(\mathsf{p}) := \frac{1}{\rho} \ln\left(1 + \sum_{k=1}^{|\mathsf{p}|-1}\left(e^{\rho d(\mathsf{p}_k, \mathsf{p}_{k+1})} - 1\right)\right). \quad (2)$$

Equation (1) is recovered by taking $\rho \to \infty$. If $\rho \to 0$, $\mathrm{smax}^\rho(\mathsf{p})$ becomes simply the sum of original distances along the path $\mathsf{p} \in \mathsf{P}_{i,j}$. Due to the triangular inequality, this is never less than $d(i, j)$, so that in a full graph with Euclidean distances the minimum path distance

[1] made symmetric by including (j, i) in E if $(i, j) \in E$

becomes $||\mathbf{x}_i - \mathbf{x}_j||_2$. Thus, the standard Gaussian RBF kernel is recovered, and no use is made of the unlabeled data. However, for a sparse graph computing the minimum path distance when $\rho \to 0$ is equivalent to `Isomap` [19].

The proposed method can be summarized as follows:

1. Build nearest neighbor graph G from all (labeled and unlabeled) data.
2. Compute the $n \times (n+m)$ distance matrix D^ρ of minimal ρ-path distances according to
$$D^\rho_{i,j} = \frac{1}{\rho^2} \ln \left(1 + \min_{\mathsf{p} \in \mathsf{P}_{i,j}} \sum_{k=1}^{|\mathsf{p}|-1} \left(e^{\rho d(\mathsf{p}_k, \mathsf{p}_{k+1})} - 1 \right) \right)^2$$
from all labeled points to all points.
3. Perform a non-linear transformation on D^ρ to get kernel K,
$$K_{i,j} = \exp\left(-\frac{D^\rho_{i,j}}{2\sigma^2} \right)$$
The linear case corresponds to $\sigma = \infty$ and $K = -\frac{1}{2} H^n D^\rho H^{n+m}$, with H^p being the $p \times p$ centering matrix (as in Multidimensional Scaling [8]): $H^p_{ij} = 1_{i=j} - 1_{i \le n}/n$.
4. Train an SVM with K and predict.

2.1.3 Comments

A few comments can be made on these steps.

1- The use of a sparse graph G is merely a way to save computation time. This is in contrast to some other graph-based methods, that require sparseness for detecting the manifold structure (e.g. `Isomap`). In our method, the sparse graph is always seen as an approximation to the full graph. However, the accuracy of this approximation depends on the value of the softening parameter ρ: for $\rho \to 0$, the direct connection is always shortest, so that every deletion of an edge can cause the corresponding distance to increase. For $\rho \to \infty$, shortest paths almost never contain any long edge, so that long edges can safely be deleted.

2- For large values of ρ, the distances between points in the same cluster are decreased. In contrast, the distances between points from different clusters are still dominated by the gaps between the clusters and, as a result, those gaps become more pronounced.

Instead of Equation (2), it is possible to use other interpolations between the max and the mean such as the Minkowski metric, $\left(\sum_{k=1}^{|\mathsf{p}|-1} d(\mathsf{p}_k, \mathsf{p}_{k+1})^{\rho+1} \right)^{1/(\rho+1)}$.

3- K is in general not positive definite (p.d.), except for $\rho = 0$ (standard RBF) and $\rho = \infty$ (then D^ρ is an ultrametric and thus negative definite [10], yielding a p.d. kernel [17]). In practice, negative eigenvalues can be observed, but they are few and small in absolute value, as documented in Table 1. In our experiments, the SVM training still converges quickly. Moreover, recent papers have argued in favor of the use of non-positive definite kernels for learning [11, 16].

ρ	0	0.5	1	2	4	8	∞
ν	0	0.19	2.96	4.66	1.89	0.02	0

Table 1: Empirically found weight on the `Coil20` dataset of the negative eigenvalues as percentage of the weight of all eigenvalues, $\nu := 100 \sum_i \max(0, -\lambda_i) / \sum_i |\lambda_i|$.

2.2 MARGIN MAXIMIZATION

The Transductive Support Vector Machine (TSVM), first introduced in [20] and implemented by [3, 12], aims at minimizing the following functional,

$$\min \frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} \xi_i + C^* \sum_{i=n+1}^{n+m} \xi_i,$$

under the constraints:

$$y_i \quad \begin{array}{ll} (\mathbf{w} \cdot \mathbf{x}_i + b) \ge 1 - \xi_i & 1 \le i \le n \\ |\mathbf{w} \cdot \mathbf{x}_i + b| \ge 1 - \xi_i & n+1 \le i \le n+m \end{array} .$$

This can be rewritten without constraint as the minimization of

$$\frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} L(y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + C^* \sum_{i=n+1}^{n+m} L(|\mathbf{w} \cdot \mathbf{x}_i + b|), \tag{3}$$

with $L(t) = \max(0, 1-t)$.

Unfortunately, the last term makes this problem non-convex and difficult to solve [3, 12]. The implementation of TSVM that we propose in this paper is to perform a standard gradient descent on (3). However, since this latter is not differentiable, we replace it by

$$\frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} L^2(y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + C^* \sum_{i=n+1}^{n+m} L^*(\mathbf{w} \cdot \mathbf{x}_i + b), \tag{4}$$

with $L^*(t) = \exp(-3t^2)$ (c.f. Figure 2).

To enforce that all unlabeled data are not put in the same class, we add the additional constraint,

$$\frac{1}{m} \sum_{i=n+1}^{n+m} \mathbf{w} \cdot \mathbf{x}_i + b = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{5}$$

This is in analogy to the treatment of the min-cut problem in spectral clustering, which is usually replaced by the normalized cut to enforce balanced solutions [13].

Finally, note that unlike traditional SVM learning algorithms, which solve the problem in the dual, we directly solve the problem in the primal. If we want to use a non-linear kernel, it is possible to compute the coordinates of each point in the kernel PCA basis [17]. More directly, one can compute the Cholesky decomposition of the Gram matrix, $K = \tilde{X}\tilde{X}^\top$ and minimize (4) with $\mathbf{x}_i \equiv (\tilde{X}_{i,1} \dots \tilde{X}_{i,n+m})$.

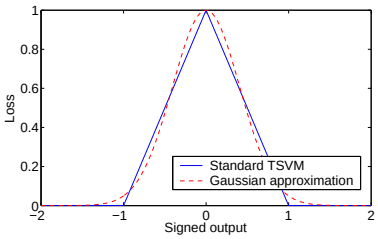


Figure 2: TSVM cost functions for unlabeled data.

We decided to initially set C^* to a small value and increase it exponentionally to C; thereby following `SVMlight`. Note that the choice of setting the final value of C^* to C is somewhat arbitrary. Ideally, it would be preferable to consider this value as a free parameter of the algorithm.

2.3 IMPLEMENTATION

From the methods discussed above, we derive three algorithms:

1. `graph`, training an SVM on a graph-distance derived kernel;
2. `∇TSVM`, training a TSVM by gradient descent;
3. `LDS` (Low Density Separation), combining both of the previous algorithms.

For `SVM`, we use the `Spider`[2] machine learning package for `matlab`. For `∇TSVM`, a conjugate gradient descent method was used.[3]

The distance computation for `graph` can be carried out using the shortest path algorithm by Dijkstra [9]. For `LDS`, the full $(n+m) \times (n+m)$ matrix D^ρ of pairwise distances has to be computed.

[2]available at `http://www.kyb.tuebingen.mpg.de/bs/people/spider`

[3]available at `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize`

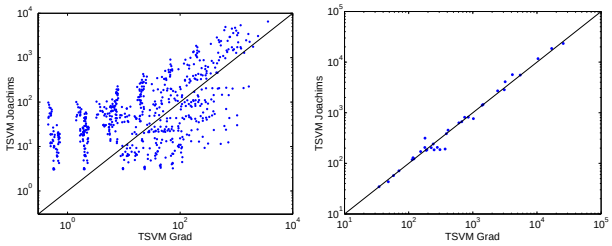


Figure 3: Each point represents the values of the objective function reached by the `TSVM` and `∇TSVM` for some value of C, σ. Points above the diagonal mean that `∇TSVM` found a better local minimum. Left: `Coil20` dataset, right: `g10n` (both described below).

Since the derived kernel is (in general) not positive definite, we can apply Multidimensional Scaling (MDS) [8] to find a Euclidean embedding of D^ρ before applying `∇TSVM`. The embedding found by the classical MDS are the eigenvectors corresponding to the positive eigenvalues of $-HD^\rho H$, where $H_{ij} = \delta_{ij} - 1/(n+m)$. For computational reasons, we decided to take only the first p eigenvectors such that

$$\sum_{i=1}^{p} \lambda_i \geq (1-\delta) \sum \max(0, \lambda_i) \quad \text{and} \quad \lambda_p \leq \delta\lambda_1, \quad (6)$$

with decreasing eigenvalues $\lambda_1 \geq \dots \geq \lambda_{n+m}$.

We compare our algorithms to one state of the art supervised method, `SVM`, and to two state of the art semi-supervised methods, the `TSVM` optimization scheme as implemented in `SVMlight` [12] and a graph-based `manifold` learning, which is closely related to those in [1, 22, 23]. More precisely, we estimate the labels of the unlabeled points by minimizing the functional

$$\sum_{i=1}^{n}(f_i - y_i)^2 + \frac{\lambda}{\sum_{i,j} w_{ij}} \sum_{i,j=1}^{n+m} (f_i - f_j)^2 w_{ij}, \quad (7)$$

where $w_{ij} = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$ if $\mathbf{x}_i$ is among the k nearest neighbors of $\mathbf{x}_j$ (or vice-versa), and 0 otherwise. This methods depends on the sparsity of the graph.

Figure 3 compares how both implementations of TSVM are able to minimize the cost function (3). Note that our proposed implementation does not minimize (3), but the differentiable approximation (4) and for this reason it has a disadvantage in the comparison shown in Figure 3. Nevertheless, on average it produces better values of the objective function, which translate, as we will see later, into better test errors.

2.3.1 Computational Complexity

We implement the search for the next-closest unexplored node in Dijkstra's algorithm with a prior-

ity queue based on a binary heap. This results in $\mathcal{O}(|E|\log(n+m))$ run time for computing the path distances of one labeled point to all other points. Thus, the entire matrix D^ρ costs $\mathcal{O}(nk(n+m)\log(n+m))$ on a k-NN graph.

The time complexity of a gradient descent algorithm is approximately equal to that of evaluating the cost function multiplied by the square of the number of variables. For ∇TSVM , this amounts to $\mathcal{O}\left((n+m)^3\right)$. The MDS is of the same time complexity, since it computes the eigendecomposition of an $(n+m)\times(n+m)$ matrix. For both algorithms, the complexity can be reduced if one considers only the first p eigenvectors.

While ∇TSVM needs to store the entire kernel matrix (on both labeled and unlabeled points), for graph an $n \times (n+m)$ part is sufficient. Memory can be reduced to the $n \times n$ part required for SVM training, but the (worst case) time required to compute individual shortest paths is as much as is required for computing all paths from a single source to all targets. For both SVM and TSVM, in practice only parts of the kernel matrices have to be (computed and) stored, because of the sparsity of the solution.

For training the manifold algorithm as given in Eq. 7, a sparse $(n+m)\times(n+m)$ matrix needs to be stored and inverted. Due to the use of a k-NN graph, the matrix has about $k(n+m)$ entries (at most $2k(n+m)$).

2.3.2 Parameters

For each algorithm, the values for a number of parameters have to be fixed. In practical applications, this is usually done by cross-validation (CV). While this is no major problem for two parameters (like the SVMs have), it is impractical for the five parameters of the graph algorithm. To reduce this number, we fix three of them in advance, as shown in the table:

algorithm	free parameters; [fixed parameters]
SVM	σ, C
TSVM	σ, C
manifold	σ, k, λ
∇TSVM	σ, C
graph	C, ρ; [$\sigma=\infty$, $k=n+m$, $\delta=0.1$]
LDS	C, ρ; [$\sigma=\infty$, $k=n+m$, $\delta=0.1$]

Figure 4 demonstrates that for LDS the parameter fixing proposed above leads only to a minor loss in accuracy. As shown in (a), a fully connected graph is good (for the optimum value of ρ). As shown in (b), $\sigma=\infty$ (i.e. no further non-linear transformation) is good (again, for the optimum value of ρ). In general the resulting kernel will not be positive definite (except for $\rho=0$ and $\rho=\infty$, see also Table 1). As shown in (c), the SVM seems to handle negative eigenvalues reasonably well. This can be seen on the right of (c): almost the same results were obtained with and without MDS. It seems safe to discard the eigenvectors corresponding to small (positive) eigenvalues (c.f. left side of the plot (c)). In the rest of the experiments, we set $\delta=0.1$.

To determine good values of the remaining free parameters (eg, by CV), it is important to search on the right scale. We therefore fix default values for C and σ that have the right order of magnitude. In a c-class problem, we use the $1/c$ quantile of the pairwise distances $D^\rho_{i,j}$ of all data points as the default for σ. The default for C is the inverse of the empirical variance s^2 of the data in feature space, which can be calculated by $s^2=\frac{1}{n}\sum_i K_{ii}-\frac{1}{n^2}\sum_{ij}K_{ij}$ from a $n\times n$ kernel matrix K. Below, all values for these parameters will be given relative to the respective default values, making them comparable for different data sets.

2.3.3 LDS algorithm

The final LDS algorithm is summarized in Figure 1. Note that slight changes are required for the extreme settings of ρ: for $\rho=0$, steps 1 to 3 have to be replaced by simply running the shortest path algorithm on $d(i,j)$ to compute $d_{i,j}$; for $\rho=\infty$, a modified version of Dijkstra that keeps track of maximum distances instead of sums along paths must be used. A matlab implementation of LDS can be obtained at http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/.

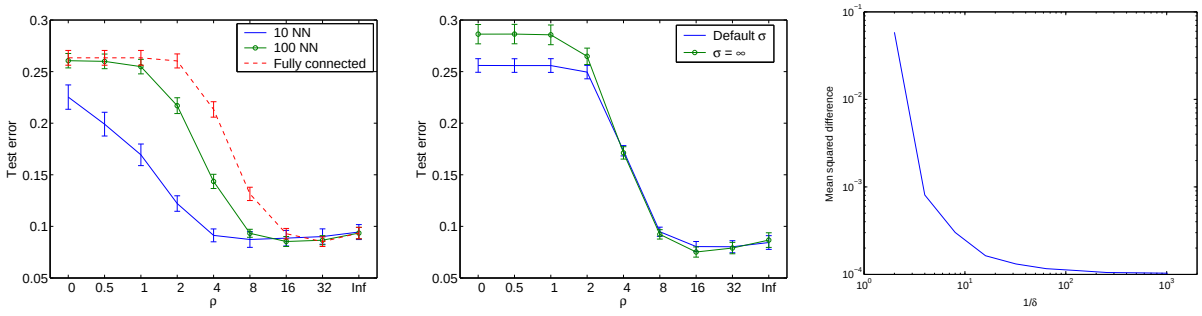
3 EXPERIMENTAL RESULTS

3.1 DATA SETS

In order to get a good picture of the effectiveness of the algorithms, we compare their generalization performance on two artificial and three real world data sets with different properties.

data set	classes	dims	points	labeled
g50c	2	50	550	50
g10n	2	10	550	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50

The artificial data sets are inspired by [2]: the data are generated from two standard normal multi-variate Gaussians. In g50c, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes error is 5%. In contrast, g10n is a deterministic problem in 10 dimensions, where the decision function traverses the centers of the Gaussians (thus violating the cluster assumption), and depends on only two of the input dimensions.



(a) (b) (c)

Figure 4: Influence of parameter choice on the test error of LDS on the `Coil20` data: (a) the graph structure; (b) σ; and (c) the approximation accuracy of the MDS. Plot (c) shows the square difference between the test error achieved with and without MDS, averaged over different values of C and ρ.

Algorithm 1 `LDS` algorithm

Require: ρ, C

 # Compute ρ-distances:

Build a fully connected graph with edge lengths $w_{ij} = \exp(\rho d(i,j)) - 1$.

Use Dijkstra's algorithm [9] to compute the shortest path lengths $d_{SP}(i,j)$ for all pairs of points.

Form the matrix D of squared ρ-path distances by $D_{ij} = \left(\frac{1}{\rho}\log(1 + d_{SP}(i,j))\right)^2$.

 # Perform multidimensional scaling:

$U\Lambda U^\top = -HDH$, where $H_{ij} = \delta_{ij} - 1/(n+m)$.

Find the threshold p such that (6) holds.

The new representation of $\mathbf{x}_i$ is $\tilde{x}_{ik} = U_{ik}\sqrt{\lambda_k}, 1 \le k \le p$.

 # Train TSVM:

for i=0 to 10 **do**

  Set $C^* = 2^{i-10}C$

  Minimize by gradient descent (3) under constraint (5).

end for

The real world data sets consist of two-class and multi-class problems. In `Coil20`, the data are gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees [15]. The `Text` dataset are the classes `mac` and `mswindows` of the `Newsgroup20` dataset preprocessed as in [18]. Finally, our `Uspst` set contains the test data part of the well-known `USPS` data on handwritten digit recognition.

3.2 EXPERIMENTS

For each of the data sets, 10 different splits into labeled and unlabeled points were randomly generated. We took care to include at least one point of each class in the labeled set (two for `Coil20`).

We used a different model selection strategy for `LDS` than for the other algorithms. For `LDS`, we carry out 5-fold cross-validation (CV) on the training set for each split, thereby simulating the real world application scenario. Note that all data (training and test) can be (and is) used as unlabeled data. The reported test errors are obtained after training the selected model on the entire training set. For the other algorithms, we are interested in the best possible performance, and simply select the parameter values minimizing the test error. In both cases, we select combinations of values on a finite grid as follows:

parameter	values
width σ	$2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3$
exponent ρ	$0, 2^0, 2^1, 2^2, 2^3, 2^4, +\infty$
penalty C	$10^{-1}, 10^0, 10^1, 10^2$
degree k	$10, 100$, all
regulariz. λ	$4^{-2}, 4^{-1}, 4^0, 4^1, 4^2$

Although `LDS` and `graph` work with any kernel, we here fix the linear kernel ($\sigma = \infty$; c.f. section 2.3.2).

3.3 RESULTS

The results are presented in Table 2. Except for the data set `g10n`, `LDS` always achieves lower test errors with empirically found parameter settings than all the

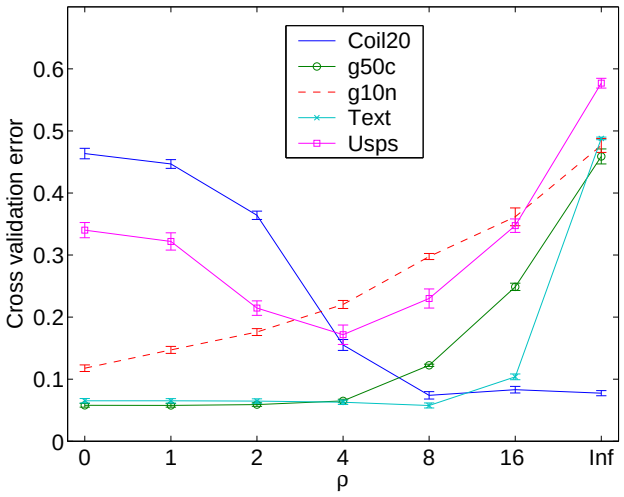


Figure 5: Cross-validation error (with standard deviation error bars) as a function of the parameter ρ.

other algorithms are capable of achieving, even when optimal parameter settings are known. This clearly demonstrates the superiority of `LDS`.

Although ∇`TSVM` always performs better (and usually, significantly better) than `TSVM`, it still fails to reach the level of `manifold` on the `Coil20` data set. But this shortcoming is eliminated by making use of the `graph` transform of the distances.

To better understand the role of the distance transform, we depict the 5-fold cross validation error for the best value of C, averaged over the 10 splits, as a function of ρ in Figure 5. We can distinguish three cases: the minimum is (i) at or close to 0; (ii) at or close to ∞; or (iii) somewhere in between.

(i) Linear classifiers are optimal by construction for the artificial data, and likely to be optimal for `Text` due to the high dimensionality. For `g50c` and `Text`, $\rho > 0$ does not substantially help ∇`TSVM`, but does not hurt either. Only for `g10n`, where the cluster assumption does not hold, increasing ρ immediatly increases the test error. **(ii)** In `Coil20`, the points of each class lie eqi-distantly on a ring. With $\rho = \infty$, all their pairwise distances are reduced to the distance of two neighboring points. Note that there is no noise which could cause unwanted bridging between two classes. **(iii)** Perhaps the most interesting case is `Uspst`, with an optimum of $\rho = 4$. While there definitely are clusters corresponding to the classes, there seem to exist outliers that would, for too large ρ, lead to erroneous merging of clusters.

As the optimum value of ρ seems to correspond to features of the data set, prior knowledge on the data could possibly be used to narrow the range to be searched.

4 CONCLUSIONS

The TSVM objective function could, at a first sight, be interpreted as a straight-forward extension of the maximum margin principle of SVM to unlabeled data. We conjecture that it actually implements two different principles: the regularization by margin maximization on the labeled points, and the cluster assumption by margin maximization on the unlabeled points. The latter does not lead to smoother decision functions, but it enforces that the decision boundary lies in low density regions.

The strength of our gradient descent approach might be that it directly optimizes the objective according to the cluster assumption: to find a decision boundary that avoids high density regions. In contrast, `TSVM` (`SVMlight` implementation) might suffer from the combinatorial nature of its approach. By deciding, from the very first step, on the putative label of every point (even though with low confidence), it may lose important degrees of freedom at an early stage, and get trapped in a bad local minimum.

The pairwise distances computed by the `graph` algorithm attempt to reflect the cluster assumption: distances of points from the same cluster are shrunk, while for points in different clusters they are dominated by the inter-cluster distance. Used with an SVM, this clearly improves over standard (Euclidean) distances, but not over other semi-supervised methods.

The combination of the `graph` distance computation with the TSVM training yields a clearly superior semi-supervised algorithm. Apparently the preprocessed distances make it less likely for the TSVM to get stuck in very suboptimal local minima. Probably the preprocessing widens small density valleys so that they are more readily found by local searches.

Although manifold learning indirectly exploits the cluster assumption, as argued above, another feature may contribute to its successes. If the intrinsic dimensionality of the data manifolds is much smaller than that of the input space, restricting the learning process to the manifolds can alleviate the "curse of dimensionality". We plan to investigate how much performance can be gained in this manner.

Future work will be on a thorough comparison of discriminative semi-supervised learning methods. We observe that the time (and to some degree, also space) complexities of all methods investigated here prohibit the application to really large sets of unlabeled data, say, more than a few thousand. Thus, work should also be devoted to improvements of the computational efficiency of algorithms, ideally of `LDS`.

	methods from literature			proposed methods		
data set	SVM	manifold	TSVM	graph	∇TSVM	LDS
Coil20	24.64%	6.20%	26.26%	6.43%	17.56%	**4.86%**
g50c	8.32%	17.30%	6.87%	8.32%	5.80%	**5.62%**
g10n	9.36%	30.64%	14.36%	9.36%	9.82%	9.72%
Text	18.87%	11.71%	7.44%	10.48%	5.71%	**5.13%**
Uspst	23.18%	21.30%	26.46%	16.92%	17.61%	15.79%

Table 2: Mean test error rates. Note that model selection was done by cross-validation for LDS whereas by minimizing the test error for the other methods. Bold numbers are statistically significantly (95% confidence) better compared to all other methods.

Acknowledgements

We thank Bernhard Schölkopf and Matthias Hein for valuable comments.

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, 2004.

[2] Y. Bengio and Y. Grandvalet. Semi-supervised learning by entropy minimization. In *NIPS*, volume 17, 2004.

[3] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *NIPS*, volume 12, 1998.

[4] O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. In *NIPS*, 2004.

[5] O. Chapelle. *Support Vector Machines: Induction Principle, Adaptive Tuning and Prior Knwoledge.* PhD thesis, LIP 6, 2003.

[6] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *NIPS*, volume 13, 2000.

[7] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *NIPS*, volume 15, 2002.

[8] T. F. Cox and M. A. Cox. *Multidimensional Scaling.* Chapman & Hall, 1994.

[9] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271, 1959.

[10] B. Fischer, V. Roth, and J. M. Buhmann. Clustering with the connectivity kernel. In *NIPS*, volume 16, 2004.

[11] B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE TPAMI*, 2004. In press.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[13] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

[14] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *ICML*, 2002.

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, Columbia Univ., USA, February 1996.

[16] C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *ICML*, pages 639–646, 2004.

[17] B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[18] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 14, 2001.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[20] V. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[21] P. Vincent and Y. Bengio. Density-sensitive metrics and kernels. Presented at the Snowbird Learning Workshop, 2003.

[22] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, 2003.

[23] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.