Graph Based Semi-Supervised Learning with Sharper Edges

Hyunjung (Helen) Shin[1][2], N. Jeremy Hill[3], and Gunnar Rätsch[1]

[1] Friedrich Miescher Laboratory, Max Planck Society, Spemannstrasse. 37, 72076 Tübingen, Germany
[2] Dept. of Industrial & Information Systems Engineering, Ajou University, San 5, Wonchun-dong, Yeoungtong-gu, 443–749, Suwon, Korea
[3] Max Planck Institute for Biological Cybernetics, Spemannstrasse. 38, 72076 Tübingen, Germany
{shin, jez, raetsch}@tuebingen.mpg.de
http://www.kyb.tuebingen.mpg.de/∼shin

Abstract. In many graph-based semi-supervised learning algorithms, edge weights are assumed to be fixed and determined by the data points' (often symmetric) relationships in input space, without considering directionality. However, relationships may be more informative in one direction (e.g. from labelled to unlabelled) than in the reverse direction, and some relationships (e.g. strong weights between oppositely labelled points) are unhelpful in either direction. Undesirable edges may reduce the amount of influence an informative point can propagate to its neighbours – the point and its outgoing edges have been "blunted." We present an approach to "sharpening" in which weights are adjusted to meet an optimization criterion wherever they are directed towards labelled points. This principle can be applied to a wide variety of algorithms. In the current paper, we present one ad hoc solution satisfying the principle, in order to show that it can improve performance on a number of publicly available benchmark data sets.

1 Introduction

Given sets of labelled and unlabelled data points, the task of predicting the missing labels can under some circumstances be aided by the information from unlabelled data points, for example by using information about the *manifold structure* of the data in input space. Many state-of-the art methods implement a *semi-supervised learning* (SSL) approach in that they incorporate information from unlabelled data points into the learning paradigm—see [3,6,24,23,17,5]. Our focus will be on a graph-based SSL approach. Despite their many differences as regards both guiding philosophy and performance, one thing common to most algorithms is the use of a matrix of values representing the pairwise relationships between data points. In graph-based SSL, the matrix of edge weights often denoted as W reflects the points' *influence* on each other,[1] which is an

[1] Many such systems are equivalent to a form of *spreading activation network* in which information is propagated around the graph.

inherently directional concept. The graph may therefore in principle be asymmetric. It is typically a sparse matrix. By contrast, in kernel-based methods like the TSVM [21,13,10], the kernel K denotes the points' *similarity* to each other, an intrinsically symmetrical property.

When adopting the kernel approach we can utilize the recent approaches of *learning the kernel matrix* [9,15,8,20,18]. In particular, the methods of [22] and [1] are focused on the use of unlabelled as well as labelled data. Using a kernel method requires that the similarity matrix satisfy the conditions of positive definiteness and symmetry to be a valid kernel [16]. It will often be a dense matrix. Most kernel learning methods are computationally demanding because of the operations involved on dense matrices–simply computing the product of two dense matrices already takes $O(n^3)$. It is possible to fit graph-based representations of pairwise relationships into a kernel-learning framework. One can directly calculate K from a graph using the diffusion kernel method [14], but this generally requires fairly expensive computation. Alternatively one can simply define similarity from the outset in terms of the graph, taking a simple formula such as $K = W^\top W$—note that this already entails a decrease in sparseness.

One of the merits of graph-based SSL lies in its computational efficiency: learning can often be done by solving a linear system with a sparse matrix W, which is nearly linear in the number of non-zero elements in W [19]. To preserve this advantage, it will be desirable that learning or manipulating W be achieved directly, without going via the route of learning a graph-based kernel matrix. To the best of our knowledge there have been relatively few approaches to learning the weights W of a graph, *Zhu et al* [24]'s being a notable exception. They address the issue of manipulating the edge weights, by a computationally intensive procedure for learning the scaling parameters of the Gaussian function that best aligns W with the data. The width parameters reflect the importance of input features, which makes their approach useful as a *feature selection* mechanism.

In this paper, we present a method which is immediately applicable to the weight matrix W. The proposed method is based on the following intuition. In an undirected graph, all connections are reciprocated and so the matrix of edge weights W is symmetric. However, when W describes relationships between labelled and unlabelled points, it is not necessarily desirable to regard all such relationships as symmetric. Some edges may convey more useful information in one direction (e.g. from labelled to unlabelled) than in the reverse direction. Propagating activity in the reverse direction, from unlabelled to labelled, may be harmful since it allows points about which information is uncertain to corrupt the very source of information in the system. Since we are already using the language of "points" and "edges", we will say that this causes the point and its outgoing edges to be "blunted", reducing their effectiveness. There are many problem settings (for example protein function prediction and other applications in the field of bio-informatics) in which (a) there is a high degree of certainty about the input-space representation of each labelled point and its label, and (b) the number of labelled points is very low. In such a situation, it seems intuitively desirable to avoid blunting, to preserve the effectiveness of the precious sources of information. Propagation of information *between* unlabelled points is a different issue—while some edges of the graph may be more helpful than others in solving the overall problem, a priori we do not know which these might be. Allowing the unlabelled points to harmonize themselves with their neighbours (implementing the assumption of *smoothness* common to most such learning approaches) is a desirable process.

To confirm this intuition, we begin with the well-known graph-based SSL formulation of [2] using Tikhonov regularization. First, we re-formulate the objective function in terms of W. Blockwise consideration of the weight matrix will allow us to state a condition which solutions W must satisfy if the objective function is to be optimized—there are many such solutions, some of which will be trivial and not lead to learning. Exploring the class of solutions, and developing a basis for comparison of their potential generalization ability, is beyond the scope of this paper and is left as an open problem. However, we propose one very simple specific solution, concordant with the logic already stated. Blockwise analysis of the inverse matrix used to make predictions will show the implications of this solution for the unlabelled points. This in turn makes clear the link between the Tikhonov regularization formulation we started with and the harmonic function solution to the Gaussian random field formulation as presented by [24].

The paper is organized as follows. In section 2, we briefly introduce the graph-based SSL algorithm under consideration. In section 3, we present the proposed idea in detail, and provide an ad hoc solution as a preliminary work, showing the connection to an earlier work based on harmonic function. In section 4, we show experimental results: illustrating the effects before and after the removal of the undesired weights. We summarize and conclude in section 5.

2 Graph-Based Semi-supervised Learning

A data point $\boldsymbol{x}_i \ \ (i = 1, \ldots, n)$ is represented as a node i in a graph, and the relationship between data points is represented by an edge where the connection strength from each node j to each other node i is encoded in element w_{ij} of a weight matrix W. Often, a Gaussian function of Euclidean distance between points, with length scale σ, is used to specify connection strength:

$$w_{ij} = \begin{cases} \exp\left(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)}{\sigma^2} \right) & \text{if} \quad i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

The $i \sim j$ stands for node i and j has an edge between them which can be established either by k nearest neighbors or by Euclidean distance within a certain radius r, $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < r$. [2] The labelled nodes have labels $\boldsymbol{y}_l \in \{-1, 1\}$,

[2] We represents scalars as lower case, vectors as boldface lower case, and martrices are uppercase. $\mathbf{0}$ (or $\mathbf{1}$) are a vector or matrix of variable-dependent size containing of all zeros (or ones).

while the unlabeled nodes have zeros $\boldsymbol{y}_u = \mathbf{0}$. Our algorithm will output an n-dimensional real-valued vector $\boldsymbol{f} = [\boldsymbol{f}_l^\top \ \boldsymbol{f}_u^\top]^\top = (f_1, \cdots, f_l, f_{l+1}, \cdots, f_{n=l+u})^\top$. which can be thresholded to make label predictions on $f_{l+1}, \ldots, f_n$ after learning. It is assumed that (a) f_i should be close to the given label y_i in labelled nodes, and (b) overall, f_i should not be too different from f_j of adjacent nodes ($i \sim j$). One can obtain $\boldsymbol{f}$ by minimizing the following quadratic functional [2]:

$$\sum_{i=1}^{l}(f_i - y_i)^2 + \mu \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 + \mu_u \sum_{i=l+1}^{n} f_i^2. \tag{1}$$

The first term corresponds to the *loss function* in terms of condition (a), and the second term represents the *smoothness* of the predicted outputs in terms of condition (b). The parameter μ (and μ_u) trades off loss versus smoothness. The last term is a regularization term to keep the scores of unlabelled nodes in a reasonable range. Alternative choices of smoothness and loss functions can be found in [6]. Hereafter, we focus on the special case of $\mu_u = 1$ [23] so that it is incorporated into the loss function. Then, the three terms degenerate to the following two:

$$\min_{\boldsymbol{f}} \ (\boldsymbol{f} - \boldsymbol{y})^\top(\boldsymbol{f} - \boldsymbol{y}) + \mu \boldsymbol{f}^T L \boldsymbol{f}, \tag{2}$$

where $\boldsymbol{y} = (y_1, \ldots, y_l, 0, \ldots, 0)^\top$, and the matrix L, called the *graph Laplacian matrix* [7], is defined as $L = D - W$ where $D = \operatorname{diag}(d_i)$, $d_i = \sum_j w_{ij}$. Instead of L, the *normalized Laplacian*, $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ can be used to get a similar result [7]. The solution of this problem is obtained as

$$\boldsymbol{f} = (I + \mu L)^{-1}\boldsymbol{y} \tag{3}$$

where I is the identity matrix.

The values of $\boldsymbol{f}$ are obtained by solving *a large sparse linear system* $\boldsymbol{y} = (I + \mu L)\boldsymbol{f}$. This numerical problem has been intensively studied, and there exist efficient algorithms, of which computational time is nearly linear in the number of nonzero entries in the coefficient matrix [19]. Therefore, the computation gets faster as the Laplacian matrix gets sparser.

3 Sharpening the Edges

3.1 Optimal Weight Matrix

Equation (3) gives us a closed-form solution that minimizes the objective function with respect to $\boldsymbol{f}$ for a given μ and fixed W. We now pose the question: what if W is not considered fixed? Is it possible to change some or all of the w_{ij} such that our algorithm performs better? We begin by re-formulating our objective function in terms of W. The smoothness term of (2) can also be expressed as

$$\mu \boldsymbol{f}^\top L \boldsymbol{f} = \boldsymbol{f}^\top \boldsymbol{y} - \boldsymbol{f}^\top \boldsymbol{f}, \tag{4}$$

by using $\boldsymbol{f}^\top(I+\mu L)\boldsymbol{f} = \boldsymbol{f}^\top\boldsymbol{y}$ which follows from $(I+\mu L)\boldsymbol{f} = \boldsymbol{y}$ from (3). Plugging (4) into (2) we have

$$\begin{aligned}\min_W \quad & (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \mu\boldsymbol{f}^\top L\boldsymbol{f} \\ &= (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \boldsymbol{f}^\top\boldsymbol{y} - \boldsymbol{f}^\top\boldsymbol{f} \\ &= \boldsymbol{y}^\top\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{f} \\ &= \boldsymbol{y}^\top\boldsymbol{y} - \boldsymbol{y}^\top(I+\mu L)^{-1}\boldsymbol{y}. \end{aligned} \tag{5}$$

The constant term $\boldsymbol{y}^\top\boldsymbol{y}$ does not affect our optimization. Eliminating this constant term and negating, (5) becomes

$$\begin{aligned}\max_W \quad & d(W) = \boldsymbol{y}^\top(I+\mu L)^{-1}\boldsymbol{y}, \\ \text{s.t.} \quad & W \geq 0,\end{aligned} \tag{6}$$

where the non-negativeness constraint of W is introduced from the natural assumption of semi-supervised learning. Given an undirected graph, (6) is a convex problem since W and hence $(I+\mu L)^{-1}$ are positive symmetric—a function $z(A) = A^p$ of a positive symmetric matrix A is convex for $-1 \leq p \leq 0$ or $1 \leq p \leq 2$ [4]. Since we wish to consider asymmetric W, we cannot guarantee convexity. We could optimize W by a gradient descent method, the derivative of (6) with respect to w_{ij} being equal to by $\mu g_i(f_i - f_j)$, where $\boldsymbol{g} = (I+\mu L^\top)^{-1}\boldsymbol{y}$ and $\boldsymbol{f}$ is given as usual by (3). However, without imposing some additional constraint, one can see that the problem has trivial solutions since any diagonal W gives an optimal value by leading $(I+\mu L)^{-1}$ to the identity matrix. Removal of all the weights clearly does not fit the goals of learning since no generalization will be possible if no information is propagated between nodes.

Optimization must proceed under some constraints which reflect our prior assumptions about the problem. Consideration of the block structure of the problem will allow us to implement the intuition expressed in section 1 and indicate parts of the weight matrix W that can be optimized, without running foul of the "no free lunch" limitation. First, note that the most part of (6) that involve $\boldsymbol{y}_u$ simply vanish since $\boldsymbol{y} = \begin{bmatrix}\boldsymbol{y}_l \\ \boldsymbol{y}_u\end{bmatrix} = \begin{bmatrix}\boldsymbol{y}_l \\ \boldsymbol{0}\end{bmatrix}$. Accordingly, (6) is simplified by (3) and becomes

$$\begin{aligned}\max_W \quad & d(W) = \boldsymbol{y}_l^\top \boldsymbol{f}_l, \\ \text{s.t.} \quad & W \geq 0,\end{aligned} \tag{7}$$

which implies that the objective is simply to maximize the dot product of $\boldsymbol{y}_l$ and $\boldsymbol{f}_l$ with respect to weight matrix W. Given that all f_i must satisfy $-1 \leq f_i \leq 1$ [12,24], the solution that maximizes $d(W)$ must clearly satisfy $\boldsymbol{y}_l = \boldsymbol{f}_l$. Next, let us represent the weight matrix as a block matrix,

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}.$$

Remember that, in the interpretation of W as a matrix of edge weights in a directed graph, the row index denotes the destination and the column index the source—so for example W_{lu} should be read as "the weights of the edges *from* unlabelled *to* labelled points, $u \to l$." For notational simplicity, let us also define $M = (I + \mu L)$, which has similar blockwise structure:

$$\begin{aligned} M &= \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} && (8) \\ &= \begin{bmatrix} I + \mu(D_{ll} - W_{ll}) & -\mu W_{lu} \\ -\mu W_{ul} & I + \mu(D_{uu} - W_{uu}) \end{bmatrix} . \end{aligned}$$

Rearranging (3) in terms of $\boldsymbol{y}$ and writing it in a similar blockwise fashion, we obtain :

$$\begin{bmatrix} \boldsymbol{y}_l \\ \boldsymbol{y}_u \end{bmatrix} = \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_l \\ \boldsymbol{f}_u \end{bmatrix} \tag{9}$$

Considering only the top row, we obtain the following relationship between $\boldsymbol{y}_l$ and $\boldsymbol{f}_l$:

$$\boldsymbol{y}_l = [I + \mu(D_{ll} - W_{ll})] \boldsymbol{f}_l - \mu W_{lu} \boldsymbol{f}_u . \tag{10}$$

from which we see by substituting the optimal solution $\boldsymbol{f}_l = \boldsymbol{y}_l$, that the condition

$$\boxed{(D_{ll} - W_{ll})\boldsymbol{y}_l = W_{lu} \boldsymbol{f}_u , \qquad (11)}$$

must hold. Equally, any solution that satisfies (11) is also optimal. This begins to show the role of the individual blocks of W in finding a solution. To express (11) solely in terms of block matrices, we use the block inverse of M,

$$M^{-1} = \tag{12}$$
$$\begin{bmatrix} M_{ll}^{-1} + M_{ll}^{-1} M_{lu} S^{-1} M_{ul} M_{ll}^{-1} & -M_{ll}^{-1} M_{lu} S^{-1} \\ -S^{-1} M_{ul} M_{ll}^{-1} & S^{-1} \end{bmatrix}$$

where S is the *Schur complement* (see [4]) of M_{ll} in M,

$$S = M_{uu} - M_{ul} M_{ll}^{-1} M_{lu} . \tag{13}$$

With $\boldsymbol{f}_l = \boldsymbol{y}_l$, this yields

$$\underbrace{\left[M_{ll}^{-1} + M_{ll}^{-1} M_{lu} S^{-1} M_{ul} M_{ll}^{-1} - I\right]}_{(a)} \boldsymbol{y}_l = 0 \tag{14}$$

from which we see that there exist a potentially large class of solutions that satisfying this condition—the right hand side of (11) may be matched to the left through manipulation of any of the four blocks of W, to affect $\boldsymbol{f}_u$. So far, however, it seems intractable to calculate a general solution class for this complex system.

3.2 An Ad-Hoc Solution

In this paper, we present an ad hoc solution for the optimal condition (11) as a preliminary work. This simplest and blockwise form of solution will be used in our experiment to exemplify the effect of the condition. Many solutions can be obtained from non-zero matrix of (a) in (14), however, we focus on a subset of solutions by confining (a) to be $\mathbf{0}$.

As we mentioned earlier, any W as a form of diagonal matrix produces optimal value of (6) or (7). More concisely speaking, both W_{ll} and W_{uu} can be any diagonal matrices, while W_{lu} and W_{ul} should be null matrices. However, no one wants to compensate a null vector of $\boldsymbol{f}_u$ as a return for holding the condition (11). Thus, let us selectively decide which block matrix can be null matrix or diagonal, by examining

$$\boldsymbol{f}_u = -S^{-1} M_{ul} M_{ll}^{-1} \boldsymbol{y}_l. \tag{15}$$

First, note that M_{ul} should not be a null matrix thus $W_{ul} \neq \mathbf{0}$ from (8), $\boldsymbol{f}_u$ will be $\mathbf{0}$ otherwise. Second, M_{ll}^{-1} will not matter unless M_{ll} is singular, which implies we can regard W_{ll} as a diagonal matrix and W_{lu} as a null matrix. Then M_{ll} becomes an identity matrix. Next, let us take S^{-1} into consideration. With W_{ll} as a diagonal matrix, W_{lu} as a null matrix, and W_{ul} as a non-zero matrix, S defined in (13) will not be singular:

$$S = I + \mu(D_{uu} - W_{uu}).$$

This allows W_{uu} to be a diagonal matrix. However, one should be careful of setting W_{uu} be a diagonal matrix which will lead to

$$\boldsymbol{f}_u = \mu W_{ul} \boldsymbol{y}_l. \tag{16}$$

This means we cannot obtain the output prediction for the unlabelled data points unless they are *directly* connected to labelled points. Remembering that W is a sparse matrix in graph-based semi-supervised learning, we hardly expect full connection from labelled to unlabelled points. Therefore, we should not allow W_{uu} to be a diagonal matrix. Note that if W_{ul} is a full matrix, (16) stands for output prediction by *k-nearest neighbor* method. To summarize, by setting W_{ll} to a non-negative diagonal matrix (including null matrix) and W_{lu} to $\mathbf{0}$,

$$\boxed{W_s = \begin{bmatrix} \text{diagonal matrix} & \mathbf{0} \\ W_{ul} & W_{uu} \end{bmatrix}, \tag{17}}$$

we can satisfy the condition (11) but still expect to obtain meaningful output prediction for unlabelled data points

$$\boldsymbol{f}_u = \mu(I + \mu(D_{uu} - W_{uu}))^{-1} W_{ul} \boldsymbol{y}_l. \tag{18}$$

In spreading activation network terms, is equivalent to activity being propagated from labelled to unlabelled data *once* ($W_{ul}\boldsymbol{y}_l$) to set the initial condition for subsequent spreading activation among $u \leftrightarrow u$, analogous to (3) but now *excluding* the labelled points. This also has intuitive appeal. First, for labelled points, it assures $\boldsymbol{f}_l = \boldsymbol{y}_l$— *there is no loss of information on labelled data points*. By disconnecting unnecessary and unhelpful edges, we allow the labelled points and their outgoing edges to stay "sharp" in their influence on the rest of the network. Second, for unlabelled points, it preserves an important principle of SSL, namely *exploitation of the manifold structure inferred from unlabelled data points*, by keeping the edges, $u \leftrightarrow u$ and $l \rightarrow u$, of W.

3.3 Harmonic Functions Revisited

The condition (11) provides a link to the formulation of [24], which characterized semi-supervised learning in terms of a *harmonic function* solution to an energy minimization problem. Particularly, the solution (18) is very similar to their solution

$$\boldsymbol{f}_u = (D_{uu} - W_{uu})^{-1} W_{ul} \boldsymbol{y}_l,$$

to which our (18) converges as μ becomes arbitrarily large. But note that their optimization proceeds from the *a priori assumption* that labels should be reconstructed without loss, $\boldsymbol{f}_l = \boldsymbol{y}_l$. Unfortunately, in the general formulation (2) of semi-supervised learning, it is not natural to hold this assumption due to the smoothness term. In the light of that, (11) plays a role of bridge between two methods, [2] and [24]: we begin with the formulation of [2] and reach at the minimum-energy solution of [24] without the necessity of assuming $\boldsymbol{f}_l = \boldsymbol{y}_l$ a priori. Note that in our formulation hyperparameter μ naturally remains from (3) through to (18) and can be tuned to the needs of each particular learning problem.

4 Experiment

We compare the performance of the original solutions (3) with W and sharpened (18) with W_s on 5 real and artificial data sets that have been used as benchmarks for comparing the performance of semi-supervised learning algorithms by [5]. We used five of the nine data sets made available by the authors of [5] for the purposes of testing semi-supervised learning algorithms. The data sets encompass both artificial and real data in a number of different settings, and are summarized in table 1. More details, and the data sets themselves, are available at: `http://www.kyb.tuebingen.mpg.de/ssl-book/`. Each data set has binary labels, and comes with 24 pre-determined splits, i.e. sets of indices dictating which data points are to be labelled. Of these, 12 splits each contain 10 randomly chosen labelled points (at least one in each class), and the other 12 splits each contain 100 randomly chosen labelled points. For each data set, an initial undirected edge graph W was constructed by making a symmetrical connection between each point and its k nearest neighbours as measured by Euclidean separation in the input space, with k set either to 10 or to 100. Weights were then set

Table 1. Summary of the five benchmark data sets used

index	name	points	dims	comment
1	`Digit1`	1500	241	artificial images
2	`USPS`	1500	241	2s and 5s vs rest
3	`COIL`$_2$	1500	241	images
4	`BCI`	400	117	small, noisy
5	`g241c`	1500	241	artificial

for each edge according to the function $w_{ij} = \exp(-s_{ij}^2/\sigma^2)$ of edge length s_{ij}, with σ set either to 10 times or to 1 times the median length of the connected edges (the former setting ensured in practice that all connected weights were roughly equal, and close to 1, and the latter setting ensured some variation in the weights). For each of the available splits, we obtain solutions (3) and (18) for four different smoothing parameter values $\mu \in \{0.1, 1, 10, 100\}$, and record the ROC scores of the unlabelled outputs $\boldsymbol{f}_u$ with respect to the true labels.

The results are summarized in Fig.1. Each pair of subplots corresponds to one of the five data sets, with results from the splits with 10 labelled points on the left and from the splits with 100 labelled points on the right. The grey points show the comparison between the two methods for each of 192 runs (2 settings of k times 2 settings of σ times 4 settings of μ times 12 random splits). In addition, red crosses show the best setting for each method—performance of the sharpened method on the 12 splits under the $\{k, \sigma, \mu\}$ setting for which that method yielded the best mean ROC score across splits, against performance of the original method on the 12 splits under *its* best setting. We can see from Fig.1(a) that the sharpening modification leads to performance that is equal to or better than the original algorithm. In some cases the improvement is small in magnitude, but it is consistent in sign. For data set 2, 3 and 4, in particular, we clearly see that the majority of grey points lie above the diagonal, indicating that, for a randomly chosen hyperparameter setting among those explored, sharpening is very likely to result in easier model selection and improvements in performance. The sharpening modification tends to gain more improvement when more labelled points are given. In the subplots of the right column (of 100 labelled points), consistently across the random splits, the best performance obtained by the sharpened method is better than the best performance obtained by the original method. We illustrate the algorithms' hyperparameter dependence in Fig.1(b). From this representation we see that the sharpened method's performance is generally equal to or better than the original. We also see that, for data sets 2, 3 and 4, one of the sharpened method's advantages lies in its relative insensitivity to the values of smoothness-loss tradeoff parameter μ. This relative insensitivity is a desirable property in situations where correct hyperparameter selection is a hit-and-miss affair. Table 2 shows the best ROC scores and the results of the Wilcoxon signed-ranks test (see [11]). Considering the best averaged ROCs across the splits, the highest scores (the numbers in boldface in the second column) are obtained by the sharpened method in 9 out of the 10

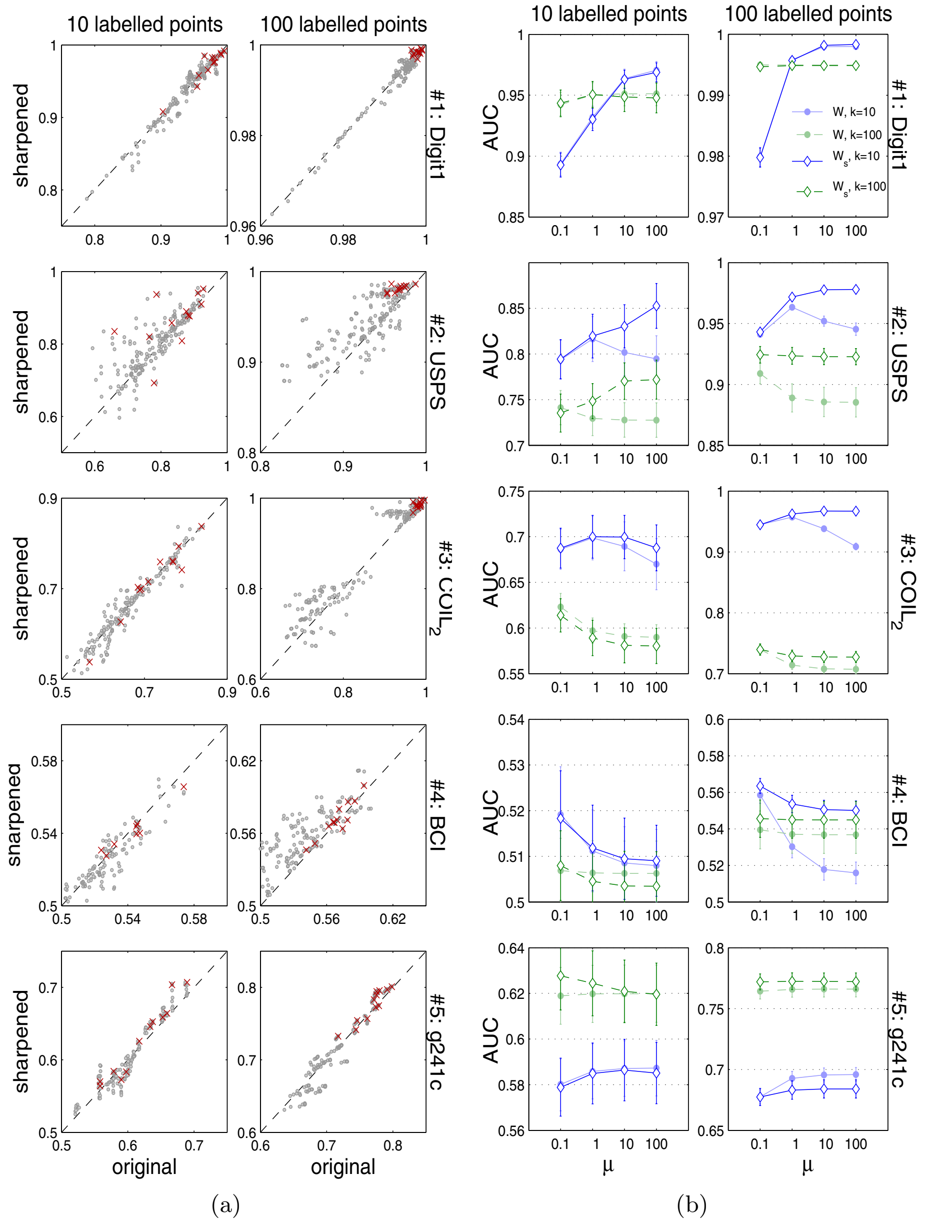

Fig. 1. Results: (a) ROC scores for the sharpened method against those for the original method, across 12 random splits of the data in each panel. Results are shown for all hyperparameter settings (grey points) and for each method's best hyperparameter setting (red crosses). (b) Hyperparameter dependence of the sharpened method using modified weight matrix W_s (open diamonds) and for the original method using W (filled circles). Mean ROC scores across the 12 splits are shown as a function of μ (on the abscissa) and k (blue–solid for $k = 10$, green–dashed for $k = 100$). Results are only shown for $\sigma = 10$ (results for $\sigma = 1$ follow a roughly similar pattern).

Table 2. Summary of the results for the five data sets

Datasets		Best ROC score (%)		Frequency of outperformance (#)		p-value
		original	sharpened	original	sharpened	
`(1) Digit1`	10 labelled	**97.13**	97.10	**200**	184	0.6280
	100 labelled	99.81	**99.84**			
`(2) USPS`	10 labelled	85.44	**87.60**	107	**277**	0.0000
	100 labelled	96.95	**98.83**			
`(3)` $\texttt{COIL}_2$	10 labelled	73.24	**74.49**	172	**212**	0.0006
	100 labelled	98.39	**98.50**			
`(4) BCI`	10 labelled	53.25	**53.39**	136	**248**	0.0000
	100 labelled	57.51	**58.29**			
`(5) g241c`	10 labelled	62.05	**63.09**	173	**211**	0.0564
	100 labelled	75.77	**77.51**			
Total				788	**1132**	0.0000

cases, with almost similar performance being attained by both methods in the remaining one. The third column compares the two methods in frequency of outperformance for the 384 (=2 $\times$192) paired ROC comparisons per dataset. In 4 out of the 5 datasets, the sharpened method outperformed the original. The p-values in the last column statistically present the significance of outperformance of the sharpened method.

5 Conclusion

In this paper, we present a simple yet efficient method for manipulating weight matrix W based on graph-based semi-supervised learning. By analyzing the objective function in blockwise fashion according to the four combinations of labelled and unlabelled points, we show an optimal condition for W that tells us which block can be manipulated, and how they may be manipulated, in order to enhance the flow of activation. This approach provides two main advantages without high computational cost or resorting to heuristics. For labelled points, it ensures that the predicted output equals its given label: *there is no loss of information on labelled data points.* For unlabelled points, it preserves the principle of semi-supervised learning: *prediction with manifold structure for unlabelled data points.* This allows us to enjoy the best of both worlds: improved performance due to "sharpening" of previously "blunted" labelled points and edges (as is also the case for [24]) and the ability to explore different smoothing settings in search of the best generalization performance (as in [2]).

For the sake of analytical convenience, the current method takes a very simple, conventional semi-supervised framework as its basis. However, incorporated into more sophisticated state-of-the-art algorithms, it has the potential to improve considerably on their original performance.

References

1. F.R. Bach and M.I. Jordan. Learning spectral clustering. *NIPS 16*, 2004.
2. M. Belkin, I. Matveeva, and P. Niyogi. Regularization and regression on large graphs. *Lecture Notes in Computer Science (In COLT)*, pages 624–638, 2004.
3. M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. *NIPS 15*, 2003.
4. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
5. O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA (in press), 2006.
6. O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. *NIPS 15*, 2003.
7. F.R.K. Chung. *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.
8. K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. *NIPS 15*, 2003.
9. N. Cristianini, J. Shawe-Taylor, and J. Kandola. On kernel target alignment. *NIPS 14*, 2002.
10. T. De Bie and N. Cristianini. Convex method for transduction. *NIPS 16*, 2004.
11. J. Demsăr. Statistical comparisons of claissifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
12. P. Doyle and J. Snell. Random walks and electric networks. *Mathematical Association of America*, 1984.
13. T. Joachims. Transductive inference for text classification using support vector machines. *In Proc. ICML*, 1999.
14. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *In Proc. ICML*, 2002.
15. G.R.G. Lanckriet, N. Cristianini, L.E. Ghaoui, P. Bartlett, and M.I. Jordan. Learning the kernel matrix with semi-definite programming. *In Proc. ICML*, 2002.
16. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
17. V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. *In Proc. ICML*, 2005.
18. S. Sonnenburg, G. Rätsch, S. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 2006. accepted.
19. D.A. Spielman and S.H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. of the 26th annual ACM symposium on Theory of computing*, pages 81–90. ACM Press, 2004.
20. K. Tsuda, G. Rätsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.
21. V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
22. Z. Zhang, D.Y. Yeung, and J.T. Kwok. Bayesian inference for transductive learning of kernel matrix using the tanner-wong data augmentation algorithm. *In Proc. ICML*, 2004.
23. D. Zhou, O. Bousquet, J. Weston, and B. Schölkopf. Learning with local and global consistency. *NIPS 16*, 2004.
24. X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *In Proc. ICML*, 2003.