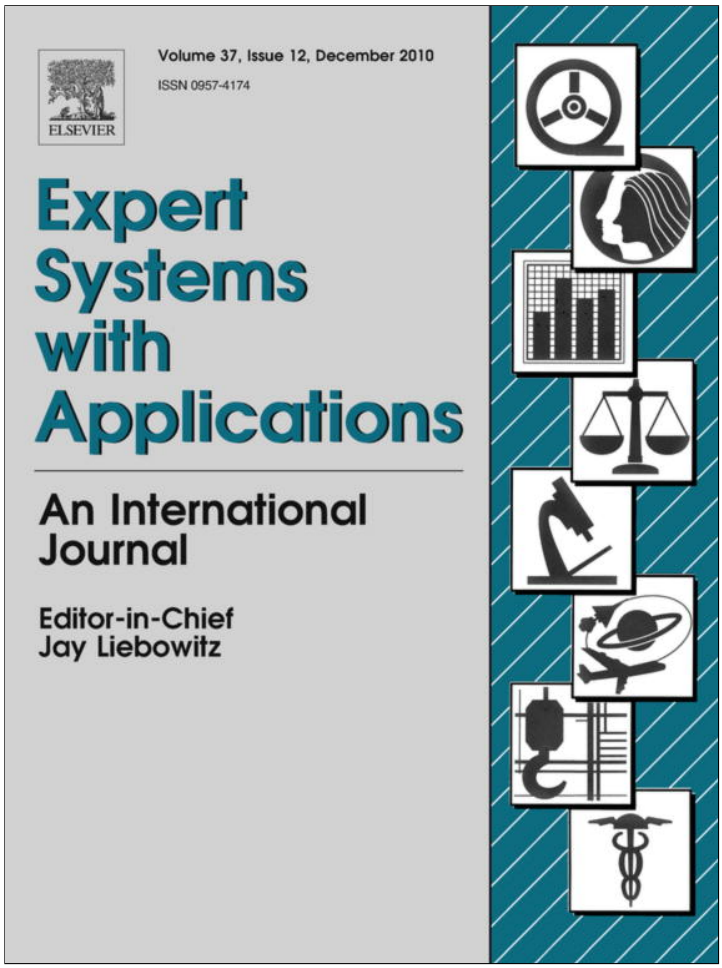

Expert Systems with Applications 37 (2010) 7870–7879

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Graph sharpening

Hyunjung Shin [a,*], N. Jeremy Hill [b], Andreas Martin Lisewski [c], Joon-Sang Park [d]

[a] Department of Industrial & Information Systems Engineering, Ajou University, San 5, Wonchun-dong, Yeoungtong-gu, 443–749 Suwon, Republic of Korea
[b] Max Planck Institute for Biological Cybernetics, Spemannstrasse 38, 72076 Tübingen, Germany
[c] Department of Molecular & Human Genetics, Baylor College of Medicine, One Baylor Plaza, Houston, TX 77030, USA
[d] Department of Computer Engineering, Hongik University, 72-1 Sangsoo-dong, Mapo-gu, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Machine learning
Semi-supervised learning

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 this paper, we present one solution satisfying the principle, in order to show that it can improve performance on a number of publicly available bench-mark data sets. When tested on a real-world problem, protein function classification with four vastly different molecular similarity graphs, sharpening improved ROC scores by 16% on average, at negligible computational cost.

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 (Belkin & Niyogi, 2003; Chapelle, Schölkopf, & Zien, 2006; Chapelle, Weston, & Schölkopf, 2003; Shin, Tsuda, & Schölkopf, 2009; Sindhwani, Niyogi, & Belkin, 2005; Soon-Ong, Smola, & Williamson, 2005; Tsuda, Shin, & Schölkopf, 2005; Zhu, Ghahramani, & Lafferty, 2003; Zhou, Bousquet, Weston, & Schölkopf, 2004). 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 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 (De Bie & Cristianini, 2004; Joachims, 1999; Vapnik, 1998), 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* (Crammer, Keshet, & Singer, 2003; Cristianini, Shawe-Taylor, & Kandola, 2002; Lanckriet, Cristianini, Ghaoui, Bartlett, & Jordan, 2002; Sonnenburg, Rätsch, Schäfer, & Schölkopf, 2006; Tsuda, Rätsch, & Warmuth, 2005). In particular, the methods of Zhang, Yeung, and Kwok (2004) and Bach and Jordan (2004) 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 (Schölkopf & Smola, 2002). 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 (Kondor & Lafferty, 2002), 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

* Corresponding author. Tel.: +82 (0)31 219 2417; fax: +82 (0)31 219 1610.
E-mail addresses: shin@ajou.ac.kr (H. Shin), jez@tuebingen.mpg.de (N.J. Hill), lisewski@bcm.edu (A.M. Lisewski), jsp@hongik.ac.kr (J.-S. Park).

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

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.04.050

non-zero elements in W (Spielman & Teng, 2004). 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 learn the weights W of a graph, Zhu et al.'s (2003) 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, the importance of information flow can be implicitly different: Some edges may convey more useful information in one direction than in the reverse direction. Some edges may propagate unhelpful information in either direction. And for some edges, it is not yet available to draw a distinction on usefulness between one direction and the reverse direction, so those should be left undirected. To confirm this idea, we begin with the well-known graph-based SSL formulation of Belkin, Matveeva, and Niyogi (2004) 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 Zhu et al. (2003). The optimal solution suggested by our criterion reconstructs the known labels without loss, as in Zhu et al. (2003), while still incorporating a variable hyperparameter for smoothness like that of Belkin et al. (2004).

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 starting from intuition to mathematical foundation, and then provide one particular solution, showing the connection to an earlier work based on harmonic function. In Section 4, we first show experimental results on bench-mark datasets illustrating the effects before and after the removal of the undesirable weights. We further extend our experiments to a real-world problem of protein function prediction based on graphs, a central challenge in recent bio-informatics research. We show that the proposed method consistently raises overall performance on protein function annotation. Finally, we conclude with some remarks on future work.[2]

2. Graph-based semi-supervised learning

In the graph-based semi-supervised learning algorithm (Zhou et al., 2004), 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 (see Fig. 1). A weight w_{ij} can take a binary value (*0* or *1*) in the simplest case. Often, a Gaussian function of Euclidean distance between points, with length scale σ, is used to specify connection strength:

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

The $i \sim j$ stands for node i and j having an edge between them which can be established either by connecting each point to its k nearest neighbors for some k, or by connecting each node i to all other nodes j for which $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| < r$, i.e. connecting all points within a certain radial Euclidean distance r of each other.[3] The labelled nodes have labels $\boldsymbol{y}_l \in \{-1, 1\}$, while the unlabelled nodes have zeros $\boldsymbol{y}_u = \boldsymbol{0}$. A graph-based semi-supervised learning algorithm will output an n-dimensional real-valued vector $\boldsymbol{f} = [\boldsymbol{f}_l^\top \ \boldsymbol{f}_u^\top]^\top = (f_1,\ldots,f_l,f_{l+1},\ldots,f_{n=l+u})^\top$ 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 the f_j of adjacent nodes ($i \sim j$). One can obtain $\boldsymbol{f}$ by minimizing the following quadratic functional (Belkin et al., 2004; Chapelle et al., 2003; Zhou et al., 2004):

$$\min_{\boldsymbol{f}} (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \mu \boldsymbol{f}^T L \boldsymbol{f}, \tag{1}$$

where $\boldsymbol{y} = (y_1,\ldots,y_l,0,\ldots,0)^\top$, and the matrix L, called the *graph Laplacian matrix* (Chung, 1997), is defined as $L = D - W$ where $D = \mathrm{diag}(d_i)$, $d_i = \sum_j w_{ij}$. 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 trades off loss versus smoothness. The solution of this problem is obtained as

$$\boldsymbol{f} = (I + \mu L)^{-1}\boldsymbol{y}, \tag{2}$$

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 non-zero entries in the coefficient matrix (Spielman & Teng, 2004). Therefore, computation gets faster as the Laplacian matrix gets sparser.

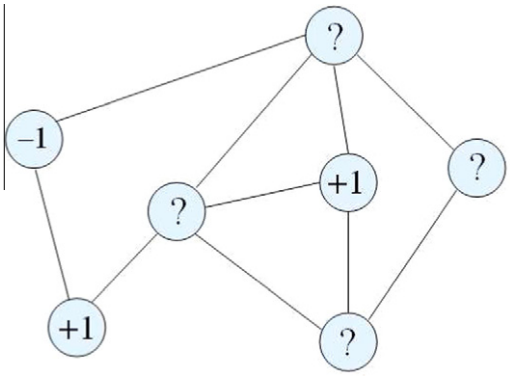

Fig. 1. Graph-based semi-supervised learning: a node is labelled either by ' +1' or ' −1', indicating the class to which it belongs. The relationship between nodes i and j is represented as a weight value on the connecting edge, w_{ij} (for nodes depicted as unconnected, the corresponding w_{ij} is zero. Graph-based semi-supervised learning seeks to classify the unlabelled nodes marked as '? ' through label propagation via edges.

[2] The current paper is an extended version of the authors' conference paper (Shin, Hill, & Raetsch, 2006).

[3] We represent scalars as lower case, vectors as boldface lower case, and matrix as uppercase. $\boldsymbol{0}$ (or $\boldsymbol{1}$) is a vector or matrix of variable-dependent size containing of all zeros (or all ones).

3. Graph sharpening

3.1. Intuition on sharpening the edges

Before unfolding the details of `graph sharpening`, let us first give an intuition about the proposed method. In an undirected graph as in Fig. 1, all connections are reciprocated and so the matrix of edge weights W is symmetric as in Fig. 2(a). However, when W describes relationships between labelled and unlabelled points, it is not necessarily desirable to regard all such relationships as symmetric. That is, we may differentiate the importance of information flow so far equally weighing all edges. *First*, some edges may convey more useful information in one direction (e.g. from labelled to unlabelled) than in the reverse direction. Propagating information in the reverse direction, from unlabelled to labelled, may be undesirable since it allows points about which information is uncertain to corrupt the 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. This is not necessarily a bad thing. If we have a large enough number of labelled data points in hand, we might wish to tolerate this phenomenon since

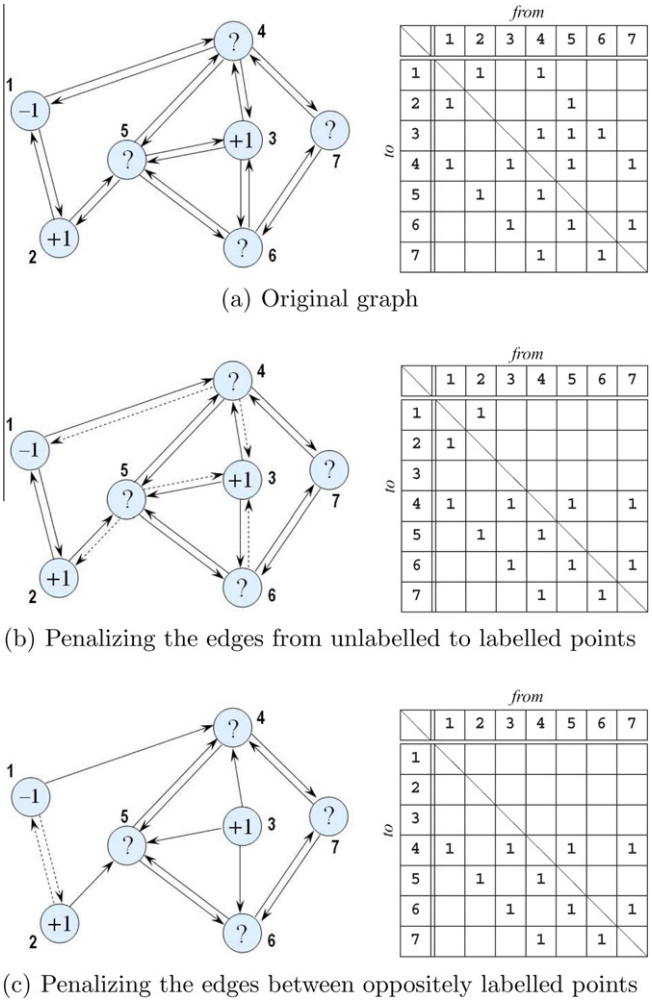

Fig. 2. Graph sharpening: (a) an original undirected (bi-directed) graph and its symmetric weight matrix W. Note that in the interpretation of W, the row index denotes the destination and the column index the source—so for example w_{ij} should be read as "the weight of the edge *from* j *to* i ($j \to i$). (b) Edges from unlabelled to labelled points (denoted as dotted arrows) are disconnected. (c) Edges between oppositely labelled points are further removed. `Sharpening` leaves the edges between unlabelled points intact.

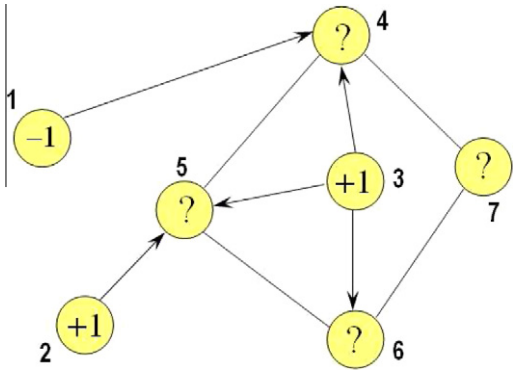

Fig. 3. The sharpened graph: In contrast to the original in Fig. 1, the sharpened graph is no longer fully reciprocated and so the matrix of edge weights W becomes asymmetric.

it may accurately reflect the data manifold structure, and can have a regularizing effect by reducing the damaging influence of labelled points that are quite simply in the wrong place (for example, through measurement error) or have the wrong label (due to label noise). However, there are many problem settings (for example, protein function prediction and other applications in the field of bioinformatics) 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 low. In this situation, it is indicated to avoid blunting, thus to preserve the effectiveness of information sources. *Second*, edges directly connecting oppositely labelled points may propagate unhelpful information in either direction. The smoothness condition in (1) plays the role of forcing both predicted scores to be similar to each other, and again, both important points, the very sources of information, are blunted. Further, those edges unnecessarily incur a conflict of the opposite flows in the area of a high certainty in the system. *Third*, propagation of information between unlabelled points is different—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 with their neighbours (thus implementing smoothness condition, common to most such learning approaches) is a desirable process.

Fig. 2 illustrates how this intuition is realized by the proposed method. Fig. 2(a) shows an original graph of seven nodes (points) and its (so far) symmetric weight matrix W. For simplicity, we set the value of '1' as a weight on every edge. Remember that, in the interpretation of W, the row index denotes the destination and the column index the source, and w_{ij} reads "the weight of the edge *from* j *to* i ($j \to i$)." In Fig. 2(b), the edges from unlabelled to labelled points w_{ij} (denoted as dotted arrows) are disconnected where i belongs to the set of labelled points $l := \{1,2,3\}$ and j to the set of unlabelled points $u := \{4,5,6,7\}$. Fig. 2(c) presents the case of connection between oppositely labelled points, w_{12} and w_{21}. Therefore, both edges are further removed from Fig. 2(b). Finally, we obtain a *sharpened* graph in Fig. 3. Note that the weight matrix becomes asymmetric, and the edges between unlabelled points remain intact.[4] We will re-visit this intuition with a particular solution followed a detailed mathematical foundation in the next section.

3.2. Optimal weight matrix

We now pose the general question with respect to the objective function (1): What if W is not considered fixed? Is it possible to

[4] Instead of disconnecting or removing an edge from W, we can penalize an undesirable edge weights by introducing a penalty term, $w_{ij}^{\text{NEW}} = w_{ij} - \delta_{ij}$ where $0 \leqslant \delta_{ij} \leqslant w_{ij}$.

change some or all of the w_{ij} such that our algorithm performs better? To implement these ideas, we begin by re-formulating our objective function in terms of W. The smoothness term of (1) can be expressed as

$$\mu \boldsymbol{f}^{\top} L \boldsymbol{f} = \boldsymbol{f}^{\top} \boldsymbol{y} - \boldsymbol{f}^{\top} \boldsymbol{f}, \tag{3}$$

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 (2). Plugging (3) into (1) we have

$$\begin{aligned} \min_{W} \quad (\boldsymbol{f}-\boldsymbol{y})^{\top}(\boldsymbol{f}-\boldsymbol{y}) + \mu \boldsymbol{f}^{\top} L \boldsymbol{f} &= (\boldsymbol{f}-\boldsymbol{y})^{\top}(\boldsymbol{f}-\boldsymbol{y}) + \boldsymbol{f}^{\top}\boldsymbol{y} - \boldsymbol{f}^{\top}\boldsymbol{f} \\ &= \boldsymbol{y}^{\top}\boldsymbol{y} - \boldsymbol{y}^{\top}\boldsymbol{f} \\ &= \boldsymbol{y}^{\top}\boldsymbol{y} - \boldsymbol{y}^{\top}(I+\mu L)^{-1}\boldsymbol{y}. \end{aligned} \tag{4}$$

The constant term $\boldsymbol{y}^{\top}\boldsymbol{y}$ does not affect our optimization. Eliminating this constant term and negating, (4) becomes

$$\begin{aligned} \max_{W} \quad & d(W) = \boldsymbol{y}^{\top}(I+\mu L)^{-1}\boldsymbol{y} \\ \text{s.t.} \quad & W \geqslant 0, \end{aligned} \tag{5}$$

where the non-negativeness constraint of W is introduced from the natural assumption of semi-supervised learning. Given an undirected graph, (5) 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 $-1 \leqslant p \leqslant 0$ or $1 \leqslant p \leqslant 2$ (Boyd & Vandenberghe, 2004). Since we wish to consider asymmetric W, we cannot guarantee convexity. We could optimize W by a gradient descent method, the derivative of (5) 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 (2). 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 3.1 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 (5) 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, (5) is simplified by (2) and becomes

$$\begin{aligned} \max_{W} \quad & d(W) = \boldsymbol{y}_l^{\top}\boldsymbol{f}_l \\ \text{s.t.} \quad & W \geqslant 0, \end{aligned} \tag{6}$$

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 $-1 \leqslant f_i \leqslant 1$ (Doyle & Snell, 1984; Zhu et al., 2003), 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,

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

Reading W_{lu} as *from* unlabelled *to* labelled ($u \to l$), is still valid in blockwise representation of W. For notational simplicity, let us also define $M = (I+\mu L)$, which has similar blockwise structure:

$$M = \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} = \begin{bmatrix} I+\mu(D_{ll}-W_{ll}) & -\mu W_{lu} \\ -\mu W_{ul} & I+\mu(D_{uu}-W_{uu}) \end{bmatrix}. \tag{7}$$

Rearranging (2) in terms of $\boldsymbol{y}$ and writing it in a similar blockwise fashion, we obtain:

$$\begin{bmatrix} \boldsymbol{y}_l \\ \boldsymbol{y}_u \end{bmatrix} = \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_l \\ \boldsymbol{f}_u \end{bmatrix}. \tag{8}$$

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

$$\boldsymbol{y}_l = [I+\mu(D_{ll}-W_{ll})]\boldsymbol{f}_l - \mu W_{lu}\boldsymbol{f}_u, \tag{9}$$

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

$$\boxed{(D_{ll}-W_{ll})\boldsymbol{y}_l = W_{lu}\boldsymbol{f}_u,} \tag{10}$$

must hold. Equally, any solution that satisfies (10) is also optimal. This begins to show the role of the individual blocks of W in finding a solution. To express (10) solely in terms of block matrices, we use the block inverse of M,

$$M^{-1} = \begin{bmatrix} M_{ll}^{-1} + M_{ll}^{-1}M_{lu}S^{-1}M_{ul}M_{ll}^{-1} & -M_{ll}^{-1}M_{lu}S^{-1} \\ -S^{-1}M_{ul}M_{ll}^{-1} & S^{-1} \end{bmatrix}, \tag{11}$$

where S is the *Schur complement* (see Boyd & Vandenberghe, 2004) of M_{ll} in M,

$$S = M_{uu} - M_{ul}M_{ll}^{-1}M_{lu}. \tag{12}$$

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

$$\underbrace{\left[M_{ll}^{-1} + M_{ll}^{-1}M_{lu}S^{-1}M_{ul}M_{ll}^{-1} - I\right]}_{(a)}\boldsymbol{y}_l = 0, \tag{13}$$

from which we see that there exist a potentially large class of solutions that satisfying this condition—the right-hand side of (10) may be matched to the left through manipulation of any of the four blocks of W, to affect $\boldsymbol{f}_u$.

3.3. An ad hoc solution

Exploring every class of solutions for the complex system (13) is currently regarded as intractable and the full exploitation is left as an open problem. However, we can specify one simple solution as a preliminary work, concordant with the intuition previously stated. So to speak, we may focus on a subset of solutions by confining (a) in (13) to be a zero matrix $\boldsymbol{0}$ although we know that many other solutions can be obtained from non-zero matrices.

As mentioned earlier, any W as a form of diagonal matrix produces optimal value of (5) or (6). 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 (10). Thus, let us selectively decide which block matrix can be null matrix or diagonal, by examining

$$\boldsymbol{f}_u = -S^{-1}M_{ul}M_{ll}^{-1}\boldsymbol{y}_l. \tag{14}$$

First, note that M_{ul} should not be a null matrix thus $W_{ul} \neq \boldsymbol{0}$ from (7), $\boldsymbol{f}_u$ will be $\boldsymbol{0}$ 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. Remember the relationship in (7), the diagonal matrix D_{ll} is established from the row sum of W_{ll} and W_{lu}. 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 (12) 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

$$\boldsymbol{f}_u = \mu W_{ul}\boldsymbol{y}_l. \tag{15}$$

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 la-

belled to unlabelled points. Therefore, we should not allow W_{uu} to be a diagonal matrix. Note that if the row sum of W_{ul} is a full vector, (15) stands for output prediction by a similar approach to k-nearest neighbour method. To summarize, by setting W_{ll} to a non-negative diagonal matrix (including null matrix) and W_{lu} to $\mathbf{0}$,

$$W_s = \begin{bmatrix} \text{diagonal matrix} & \mathbf{0} \\ W_{ul} & W_{uu} \end{bmatrix}, \tag{16}$$

we can satisfy the condition (10) but still expect to obtain meaningful output prediction for unlabelled data points

$$\boldsymbol{f}_u = \mu(I + \mu(D_{uu} - W_{uu}))^{-1} W_{ul} \boldsymbol{y}_l. \tag{17}$$

In spreading activation network terms, (17) 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 (2) 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.

3.4. Harmonic functions revisited

The condition (10) provides a link to the formulation of Zhu et al. (2003), which characterized semi-supervised learning in terms of a *harmonic function* solution to an energy minimization problem. Particularly, the solution (17) is very similar to their solution

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

to which our (17) converges as μ becomes arbitrarily large—infinite smoothness. 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 (1) of semi-supervised learning, it is not natural to hold this assumption due to the smoothness term. In the light of that, (10) plays a role of bridge between two methods (Belkin et al., 2004 & Zhu et al., 2003): we begin with the formulation of Belkin et al. (2004) and reach at the minimum-energy solution of Zhu et al. (2003) without the necessity of assuming $\boldsymbol{f}_l = \boldsymbol{y}_l$ *a priori*. Note that in our formulation hyperparameter μ naturally remains from (2) through to (17), which can be regarded as an advantage over a *harmonic function* solution since μ can be tuned to the needs of each particular learning problem, i.e., setting different values of μ depending on degree of noise in a weight matrix.

4. Experiments

We compared the performance of the sharpened solution (17) from W_s and the original (2) from W, using five bench-marking data sets and a biological data set. Hereafter, the `original` will be used as an abbreviation for the *original* graph or the performance of the *original* solution or the *original* method depending on the context, and the `sharpened` likewise.

4.1. Bench-marking problems

4.1.1. Data

Five data sets were taken in our experiment from `http://www.kyb.tuebingen.mpg.de/ssl-book/` that has been used as bench-marking site for the purposes of testing various semi-supervised learning algorithms (Chapelle et al., 2006). 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 the web-site. 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 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 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 (2) and (17) 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.

Table 1
Summary of the five bench-mark data sets used.

Name	Points	Dims	Comment
`Digit1`	1500	241	Artificial images
`USPS`	1500	241	2s and 5s vs. rest
`COIL`$_2$	1500	241	Images
`BCI`	400	117	Small, noisy
`g241c`	1500	241	Artificial

4.1.2. Results

The results are summarized in Fig. 4. 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 $\sigma \times 4$ settings of $\mu \times 12$ random splits). In addition, red crosses show the best setting for each method—performance of the `sharpened` on the 12 splits under the $\{k, \sigma, \mu\}$ setting for which method yielded the best mean ROC score across splits, against performance of the `original` on the 12 splits under *its* best setting. We can see from Fig. 4(a) that `sharpening` leads to performance that is equal to or better than the `original`. In some cases the improvement is small in magnitude, but it is consistent in sign. For data sets `USPS`, `COIL`$_2$ and `BCI`, 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` is better than the best performance obtained by the `original`. We illustrate the algorithms' hyperparameter dependence in Fig. 4(b). From this representation, we see that the performance of the `sharpened` is generally equal to or better than the `original`. We also see that, for data sets `USPS`, `COIL`$_2$ and `BCI`, one of the `sharpening`'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 Demšar, 2006). Considering the best averaged ROCs, the highest scores (the numbers

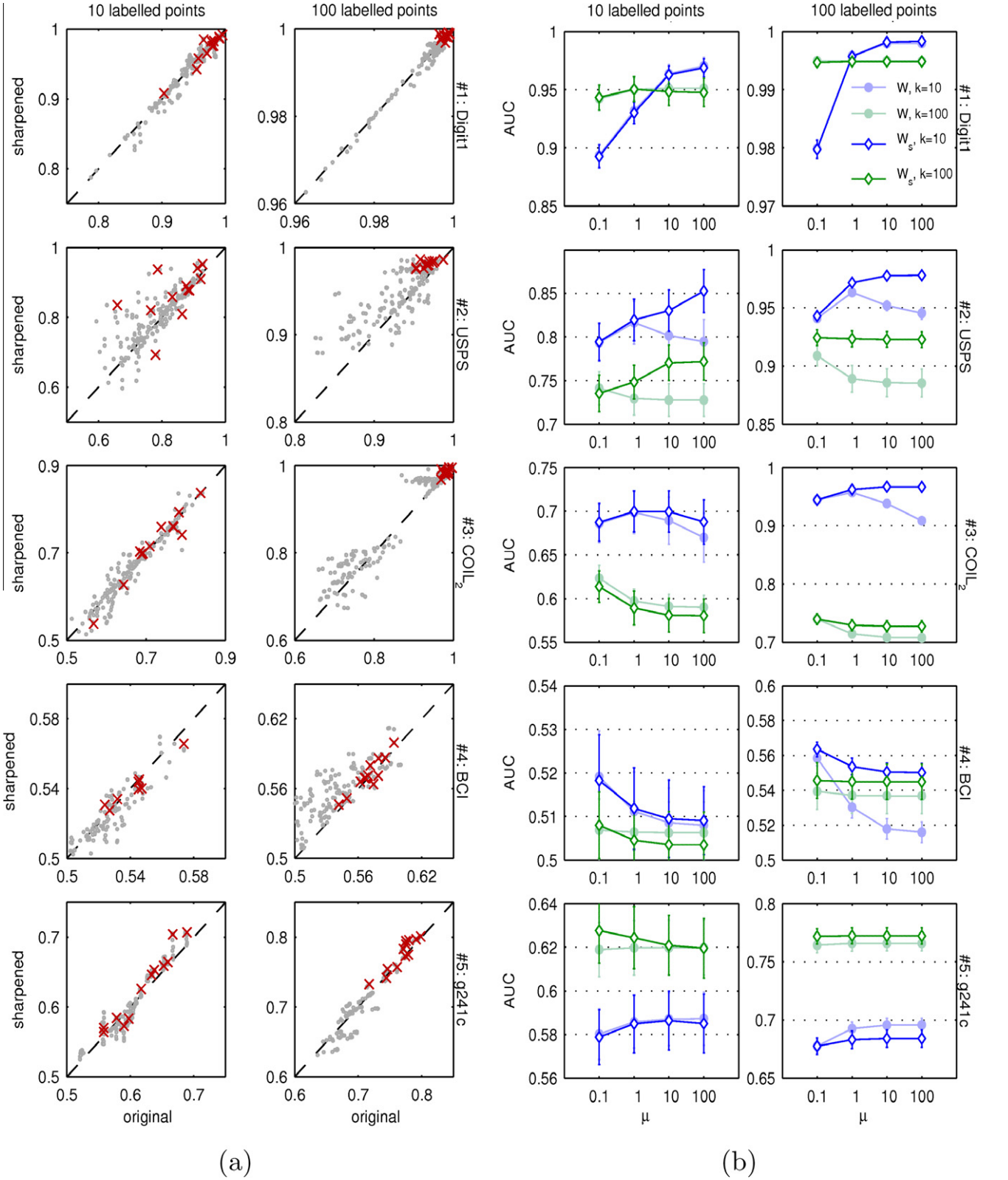

Fig. 4. Results: (a) ROC scores for the `sharpened` against those for the `original`, across 12 random splits of the data in each panel. Results are shown for all hyperparameter settings (grey points) and for each method's best hyperparameter setting (red crosses). (b) Hyperparameter dependence of the `sharpened` using modified weight matrix W_s (open diamonds) and for the `original` 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). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in boldface in the second column) are obtained by the `sharpened` in 9 out of the 10 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

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

Datasets		Best ROC score		Frequency of outperformance (#)		p-Value
		original	sharpened	original	sharpened	
Digit1	10 labelled	**0.9713**	0.9710	**200**	184	0.6280
	100 labelled	0.9981	**0.9984**			
USPS	10 labelled	0.8544	**0.8760**	107	**277**	0.0000
	100 labelled	0.9695	**0.9883**			
COIL$_2$	10 labelled	0.7324	**0.7449**	172	**212**	0.0006
	100 labelled	0.9839	**0.9850**			
BCI	10 labelled	0.5325	**0.5339**	136	**248**	0.0000
	100 labelled	0.5751	**0.5829**			
g241c	10 labelled	0.6205	**0.6309**	173	**211**	0.0564
	100 labelled	0.7577	**0.7751**			
Total				788	**1132**	0.0000

the 384 (=2 × 192) paired ROC comparisons per dataset. In 4 out of the 5 datasets, the sharpened outperformed the original. The p-values in the last column statistically present the significance of outperformance of the sharpened.

4.2. A real-world problem: protein function prediction

The prediction of biological functions of proteins is a central challenge in modern biology but few genomic or proteomic data have detailed annotations due to the demanding cost and time required to obtain this information (Friedberg, 2006). To aid the function annotation of proteins via machine learning, a natural approach is to take protein networks (graphs) as input – the most common representational form of the relationship between proteins, of which nodes depict proteins and edges represent similarities between protein pairs. Similarity values on edges may come in many forms, ranging from genomic and molecular to cellular and tissue contexts based on sequence or structure comparisons (Adai, Date, Wieland, & Marcotte, 2004; Friedberg & Godzik, 2005; Hou, Jun, Zhang, & Kim, 2005; Yona, Linial, & Linial, 1999), and there may exist noise because of inherent errors during wet-lab experiments or measurements. The need of this circumstance is perfectly matches to the goal of sharpening – accuracy improvement by removing unhelpful edges from graphs.

4.2.1. Data

Protein chains were obtained from the PDBselect25 list (version October 2004), where any pair shares no more than 25% sequence identity. Functional information was assigned in terms of the Gene Ontology (GO) in the category 'Molecular function' and mapped through the gene ontology annotation database (GOA PDB 24.0). Of all PDBselect25 chains, 599 proteins shared at least one of the 20 highly populated GO terms as in Hou et al. (2005), however, five GO terms having only a few labelled proteins (fewer than 10) were excluded from our experiment. Therefore, our experiment became 15 binary-class classification problems, determining membership or non-membership of unannotated (unlabelled) proteins for the respective GO terms. The 15 GO terms are presented in Table 3.

To generate weighted edges between nodes (or proteins), we used four different computational measures of molecular similarity. Each measure assigned to every protein pair (i,j) a positive weight $(0 \leqslant w_{ij} \leqslant 1)$ meaning the degree of molecular similarity between chain i and j. The four different similarity measures are Basic Local Alignment and Search Tool (BLAST, Altschul, Gish, Miller, Myers, & Lipman, 1990), the standard approach to alignment of primary sequence which resulted in a degree of sequence identity; length-corrected Contact Metric (CM, Lisewski & Lichtarge, 2006), a metric-based similarity score by considering vector representations of contacts from the entire tertiary structure; Fast Alignment and Search Tool (FAST, Zhu & Weng, 2005), an accurate and computationally efficient 3D geometrical alignment algorithm calculating positive similarity scores; Evolutionary Trace Annotation (ETA, Kristensen et al., 2006), a method that measures similarity based on local similarity of protein substructure, specifically 3D-templates that are small structural motifs of evolutionarily important residues identified by the evolutionary trace method (Lichtarge, Bourne, & Cohen, 1995). Our choice represented all currently and commonly used approaches to protein similarity measurement (Watson, Laskowski, & Thornton, 2005): Sequence alignment (BLAST), global 3D alignment (FAST), alignment-independent vector based approaches (CM), and local 3D alignment of small motifs (ETA). An edge weight from BLAST/CM/FAST is a continuous value while that of ETA is a binary value. A detailed description can be found in Shin, Lisewski, and Lichtarge (2007).

Table 3
Fifteen functional Gene Ontology terms from the category 'Molecular function'. Among 20 GO terms in Hou et al. (2005), five terms having fewer than ten annotated proteins were excluded from our experiment. The number in parentheses represents the number of labelled proteins.

GO term	Molecular function
0003677	DNA binding (137)
0005515	Protein binding (49)
0016491	Oxidoreductase activity (39)
0005524	ATP binding (95)
0003723	RNA binding (50)
0006118	Electron transport (87)
0003676	Nucleic acid binding (63)
0003824	Catalytic activity (57)
0005198	Structural molecule activity (22)
0005509	Calcium ion binding (32)
0008270	Zinc ion binding (56)
0005489	Electron transporter activity (37)
0004871	Signal transducer activity (18)
0004672	Protein kinase activity (29)
0004867	Serine-type endopeptidase inhibitor activity (15)

4.2.2. Results

Since a protein can belong to several GO categories, we posed a binary-class classification problem for each GO term in Table 3. For each GO term, we calculated the fivefold cross-validation (5CV) ROC score as a performance measurement. We examined the performance change in terms of original vs. sharpened for the four individual graphs obtained from BLAST, FAST, CM, and ETA, respectively. The value of the smoothing parameter was obtained by 5CV searching over $\mu \in \{0.1, 1, 5, 10, 50, 100\}$, and the results were compared at their best parameters.

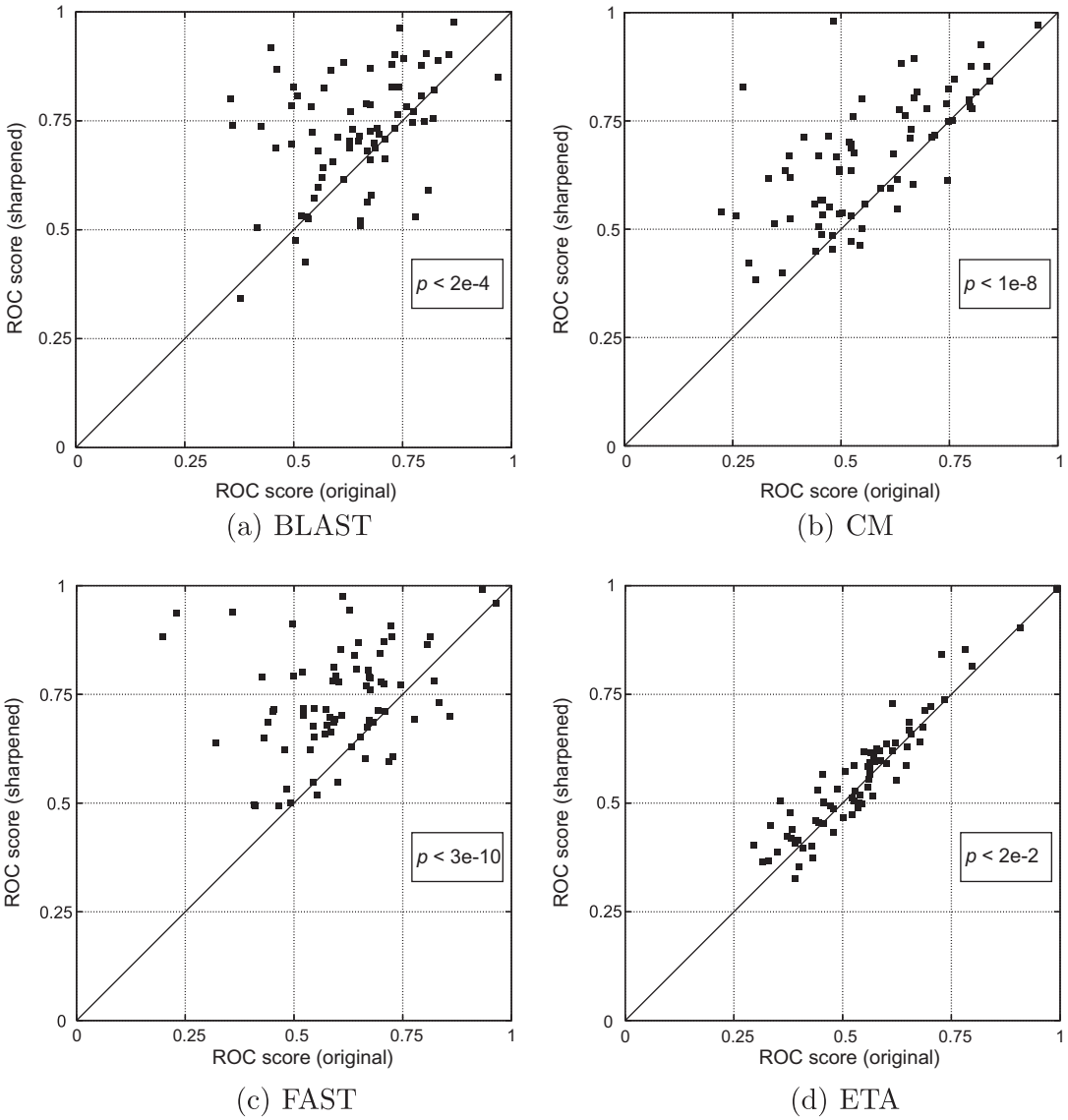

Fig. 5. Positive effect of `sharpening` in pairwise ROC score comparison of the `sharpened` vs. the `original`: The four graphs are from (a) BLAST, (b) CM, (c) FAST, and (d) ETA, respectively. More dots in upper diagonal half indicate that the method in vertical axis outperforms the other assigned in the horizontal axis; Most dots of 75 ROC pairs per graph are in the upper diagonal. The *p*-values (from sign tests) represent the statistical deviation from a random distribution of dots in the upper and lower half.

Table 4
ROC scores of individual graphs: Boldface stands for the higher score between the `original` and the `sharpened`. In most comparisons, the `sharpened` gave a higher ROC score.

GO term	BLAST		CM		FAST		ETA	
	original	sharpened	original	sharpened	original	sharpened	original	sharpened
0003677	**0.7134**	0.7132	0.6432	**0.6445**	0.6507	**0.6547**	**0.5056**	0.5020
0005515	0.6952	**0.7640**	0.4608	**0.7313**	0.5257	**0.7832**	0.4297	**0.4853**
0016491	0.4130	**0.7176**	0.7480	**0.7645**	0.6393	**0.7263**	0.5337	**0.5523**
0005524	0.5586	**0.5619**	0.4861	**0.5167**	0.5486	**0.6317**	0.5736	**0.5849**
0003723	**0.5989**	0.5919	0.4094	**0.5620**	0.4283	**0.6185**	0.4558	**0.4578**
0006118	0.6814	**0.7368**	0.6101	**0.6279**	0.6725	**0.7152**	0.5700	**0.5969**
0003676	0.7012	**0.7206**	0.5593	**0.6827**	0.5512	**0.6910**	0.4494	**0.4776**
0003824	0.6008	**0.7985**	0.7561	**0.7524**	0.6965	**0.8104**	**0.4542**	0.4498
0005198	0.6461	**0.7126**	0.6674	**0.7435**	**0.6761**	0.6387	0.5291	**0.5735**
0005509	0.5932	**0.7478**	0.4994	**0.5575**	0.7133	**0.8204**	0.5614	**0.5829**
0008270	0.7402	**0.7588**	0.5894	**0.6549**	0.6015	**0.6734**	0.4998	**0.5200**
0005489	0.5923	**0.8395**	0.3633	**0.5883**	0.6179	**0.8207**	0.5865	**0.5888**
0004871	0.6787	**0.7391**	0.3948	**0.5598**	0.3701	**0.8351**	**0.7517**	0.7408
0004672	**0.6004**	0.5463	0.5137	**0.6561**	0.6147	**0.7284**	0.6421	**0.6695**
0004867	0.8276	**0.9178**	0.7529	**0.9293**	0.7563	**0.8828**	0.5293	**0.5563**
Avg.	0.6427	**0.7244**	0.5636	**0.6648**	0.6042	**0.7354**	0.5381	**0.5559**

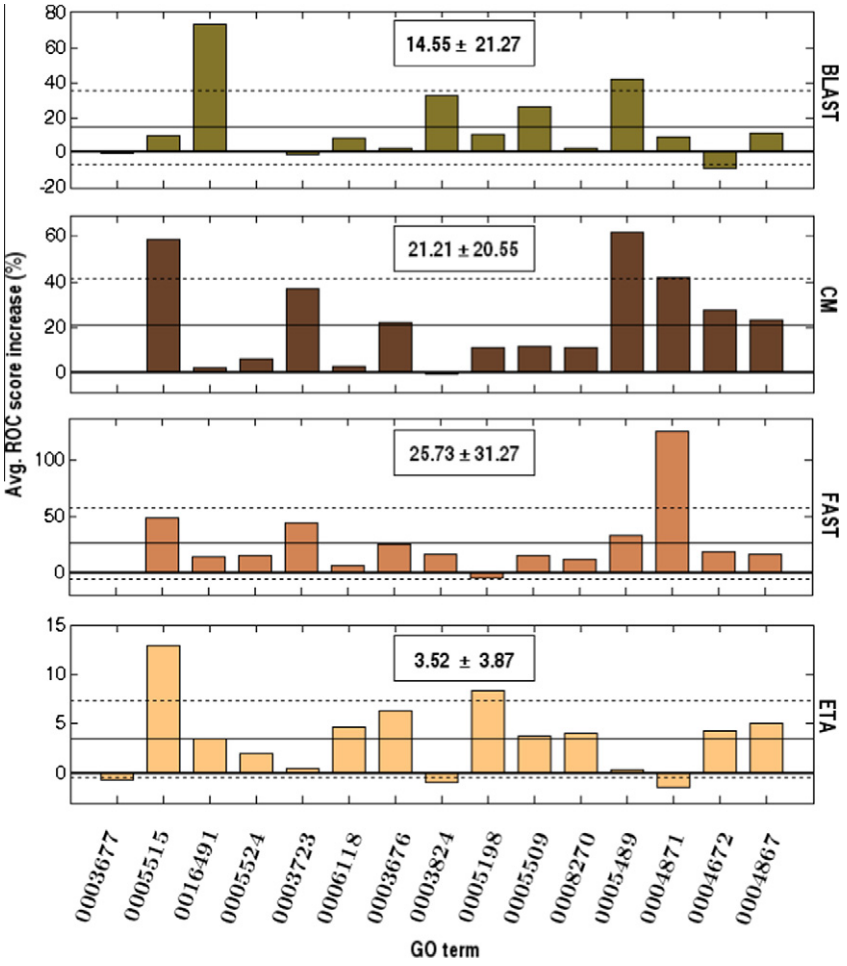

Fig. 6. ROC score increase by `sharpening`: an ROC score increase by `sharpening` was calculated with $(\texttt{sharpened} - \texttt{original}) \div \texttt{original} \times 100\%$. A bar in a panel represents the individual increase of ROC score per GO term (class). A solid line stands for the average ROC increase across the 15 GO terms, and a dotted line for the corresponding standard deviation. The results show that `sharpening` upgrades the ROC scores of the `original`s by 14.55% (±21.27), 21.21% (±20.55), 25.73% (±31.27), and 3.52% (±3.87) for BLAST, CM, FAST, and ETA, respectively.

Fig. 5 shows the distribution of 75 (=15 GO terms × 5 CVs) ROC score pairs of the `original` vs. the `sharpened` per graph. Most dots are scattered above the diagonal, thus indicating better performance of the `sharpened` against `original`. For all four graphs, a sign test rejected, at a high level of significance, the null hypothesis that the distribution would be evenly scattered above and below the diagonal (p-values are shown in the insert in each panel). Table 4 compares the ROC scores averaged over the five cross-validation folds. Fig. 6 verifies how `sharpening` upgrades the performance of the `original`. We calculated the ROC score increase with the formula, $(\texttt{sharpened} - \texttt{original}) \div \texttt{original} \times 100\%$. The results show that `sharpening` significantly contributes to ROC score; With `sharpening`, for instance, one can increase the original ROC score of the BLAST graph by an average of 14.55%, with a standard deviation of 21.27% across the 15 GO terms. Similarly, we see improvements of 21.21% (±20.55), 25.73% (±31.27), and 3.52% (±3.87) for CM, FAST, and ETA, respectively. Note that `sharpening` does not include any additional parameters, nor does it require computation. Rather, it makes the time for solving the linear system in (2) shorter since the Laplacian matrix is made sparser. The computation time for an original graph was nearly trivial (less than 0.001 cpu second with MATLAB in a standard 1500 MHz PC with 1 GByte of memory), and it became further faster with `sharpening` (less than 0.0003 cpu second).

5. Conclusion and discussion

`Graph sharpening` for graph-based semi-supervised learning methods is a formal yet intuitive approach for disconnecting undesirable edges in graphs: without resorting to heuristics, `sharpening` yields a sparser graph that contains less noise and, in turn, reduces computational expense and improves prediction with no additional parameters. We analyzed an optimal condition for the weight matrix W in blockwise fashion, giving us an intuition of where and how to adjust graph weights in order to eliminate undesirable flow of information in graphs. For labelled points, `sharpening` 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 Zhu et al. (2003)) and the ability to explore different smoothing settings in search of the best generalization performance (as in Belkin et al., 2004). `Graph sharpening` particularly works more effectively when a graph is noisy, which can often be the case in many practical problems, for instance, biological networks of proteins as in our experiments.

This paper motivates possible future studies. For the sake of analytical convenience, the present version of `sharpening` takes

a very simple, conventional semi-supervised framework as its basis, and only takes one particular solution. However, incorporated into more sophisticated state-of-the-art algorithms, it has considerable potential to improve the original's base performance. And further, there exist solution classes where more various applications could be exploited. For example, the creation rather than elimination of graph edges could lead to discovery of novel relationships between nodes.

Acknowledgement

H.S. gratefully acknowledges the support from Post Brain Korea 21 and a research grant from National Research Foundation of the Korean Government. J.P. was supported in part by NAP of Korea Research Council of Fundamental Science & Technology.

References

Adai, A., Date, S., Wieland, S., & Marcotte, E. (2004). Connecting the protein structure universe by using sparse recurring fragments. *Journal of Molecular Biology, 340*(1), 179–190.

Altschul, S., Gish, W., Miller, W., Myers, E., & Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology, 215*, 403–410.

Bach, F., & Jordan, M. (2004). Learning spectral clustering. In *Advances in neural information processing systems (NIPS)*.

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and regression on large graphs. In *Lecture notes in computer science (COLT)* (pp. 624–638).

Belkin, M., & Niyogi, P. (2003). Using manifold structure for partially labelled classification. In S. Becker, S. Thrun, & K. Obermayer (Eds.). *Advances in neural information processing systems (NIPS)* (Vol. 15). MIT Press.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In S. Becker, S. Thrun, & K. Obermayer (Eds.). *Advances in neural information processing systems (NIPS)* (Vol. 15, pp. 585–592). MIT Press.

Chung, F. (1997). *Spectral graph theory*. Number 92 Regional Conference Series in Mathematics. Providence, RI: American Mathematical Society.

Crammer, K., Keshet, J., & Singer, Y. (2003). Kernel design using boosting. In *Advances in neural information processing systems*.

Cristianini, N., Shawe-Taylor, J., & Kandola, J. (2002). On kernel target alignment. In *Advances in neural information processing systems (NIPS)* (pp. 367–373).

De Bie, T., & Cristianini, N. (2004). Convex method for transduction. In *Advances in neural information processing systems (NIPS)*.

Demsăr, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Doyle, P., & Snell, J. (1984). *Random walks and electric networks*. Mathematical Association of America.

Friedberg, I. (2006). Automated protein function prediction – The genomic challenge. *Briefings in Bioinformatics, 7*(3), 225–242.

Friedberg, I., & Godzik, A. (2005). Connecting the protein structure universe by using sparse recurring fragments. *Structure, 13*(8), 1213–1224.

Hou, J. T., Jun, S. R., Zhang, C., & Kim, S. H. (2005). Global mapping of the protein structure space and application in structure-based inference of protein function. *Proceedings of the National Academy of Sciences of the United States of America, 102*(10), 3651–3656.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the international conference on machine learning (ICML)*.

Kondor, I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the international conference on machine learning (ICML)* (pp. 315–322).

Kristensen, D. M., Chen, B. Y., Fofanov, V. Y., Ward, R. M., Lisewski, A. M., Kimmel, M., et al. (2006). Recurrent use of evolutionary importance for functional annotation of proteins based on local structural similarity. *Protein Science, 15*(6), 1530–1536.

Lanckriet, G., Cristianini, N., Ghaoui, L., Bartlett, P., & Jordan, M. (2002). Learning the kernel matrix with semi-definite programming. In *Proceedings of the international conference on machine learning (ICML)* (pp. 323–330).

Lichtarge, O., Bourne, H., & Cohen, F. (1995). An evolutionary trace method defines binding surfaces common to protein families. *Journal of Molecular Biology, 257*(2), 342–358.

Lisewski, A. M., & Lichtarge, O. (2006). Rapid detection of similarity in protein structure and function through contact metric distances. *Nucleic Acids Research, 34*(22), e152.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Shin, H., Hill, N. J., & Raetsch, G. (2006). Graph-based semi-supervised learning with sharper edges. In *Proceedings of the 17th European conference on machine learning. LNAI* (Vol. 4212, pp. 402–413).

Shin, H., Lisewski, A. M., & Lichtarge, O. (2007). Graph sharpening plus graph integration: A synergy that improves protein functional classification. *Bioinformatics, 23*(23), 3217–3224.

Shin, H., Tsuda, K., & Schölkopf, B. (2009). Protein functional class prediction with a combined graph. *Expert Systems with Applications, 36*(2), 3284–3292.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: From transductive to semi-supervised learning. In *Proceedings of the international conference on machine learning (ICML)*.

Sonnenburg, S., Rätsch, G., Schäfer, S., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research, 7*, 1531–1565.

Soon-Ong, C., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research, 6*, 1043–1071.

Spielman, D., & Teng, S. (2004). Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 26th annual ACM symposium on theory of computing* (pp. 81–90). ACM Press.

Tsuda, K., Rätsch, G., & Warmuth, M. (2005). Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research, 6*, 995–1018.

Tsuda, K., Shin, H., & Schölkopf, B. (2005). Fast protein classification with multiple networks. *Bioinformatics*, 59–65.

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

Watson, J. D., Laskowski, R. A., & Thornton, J. M. (2005). Predicting protein function from sequence and structural data. *Current Opinion in Structural Biology, 15*(3), 275–284.

Yona, G., Linial, N., & Linial, M. (1999). Protomap: Automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins: Structure, Function, and Genetics, 37*, 360–678.

Zhang, Z., Yeung, D., & Kwok, J. (2004). Bayesian inference for transductive learning of kernel matrix using the Tanner–Wong data augmentation algorithm. In *Proceedings of the international conference on machine learning (ICML)*.

Zhou, D., Bousquet, O., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *Advances in neural information processing systems (NIPS)* (Vol. 16, pp. 321–328). MIT Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 912–919). AAAI Press.

Zhu, J., & Weng, Z. (2005). Fast: A novel protein structure alignment algorithm. *Proteins, 14*, 417–423.