Journal of the Operations Research
Society of Japan
2007, Vol. 50, No. 3, 219-232

ELECTRIC NETWORK CLASSIFIERS FOR SEMI-SUPERVISED LEARNING ON GRAPHS

Hiroshi Hirai
Kyoto University

Kazuo Murota
University of Tokyo

Masaki Rikitoku
Justsystem

(Received September 2, 2005; Revised December 15, 2006)

Abstract We propose a new classifier, named *electric network classifiers*, for semi-supervised learning on graphs. Our classifier is based on nonlinear electric network theory and classifies data set with respect to the sign of electric potential. Close relationships to C-SVM and graph kernel methods are revealed. Unlike other graph kernel methods, our classifier does not require heavy kernel computations but obtains the potential directly using efficient network flow algorithms. Furthermore, with flexibility of its formulation, our classifier can incorporate various edge characteristics; influence of edge direction, unsymmetric dependence and so on. Therefore, our classifier has the potential to tackle large complex real world problems. Experimental results show that the performance is fairly good compared with the diffusion kernel and other standard methods.

Keywords: Network flow, semi-supervised learning, SVM (support vector machine), graph kernel

1. Introduction

We consider semi-supervised classification problems on graphs, in which some vertices of a graph are labeled as positive or negative, and others are unlabeled. The task is to classify the unlabeled data. Such problems arise in biological networks [19] and text classification [10]. One possible approach to this problem is the support vector machine (SVM) and other kernel-based methods [16]. The central issue in kernel-based methods is how to construct or learn a kernel from a given graph. The *diffusion kernel* [12] is such a graph kernel constructed from the graph Laplacian. On top of the diffusion kernel, several learning kernel algorithms have been proposed (see [10, 17, 18, 21]). However, to construct kernel matrix using graph Laplacian is a very heavy computational task; it requires a large amount of memory for the kernel matrix, diagonalizations, and optimizations on the matrix space.

Here we introduce a new binary classifier, named *electric network classifier*, for semi-supervised learning on graphs, based on nonlinear electric network theory. Our approach constructs a kernel only implicitly and classifies unlabeled data directly using electric potential. In so doing, we can avoid heavy kernel computations and obtain the potential using fast network flow algorithms. Furthermore, our classifier can incorporate the influence of edge direction (unilateral or unsymmetric dependence) and other edge characteristics in contrast to other graph kernel methods that construct a *symmetric* kernel matrix representing pairwise similarity on vertices of a graph.

Thus our classifier has the potential to tackle large complex real-world problems. Experimental results show that the performance is fairly good compared with diffusion and linear kernels. Our classifier can be understood as a kind of discrete version of *Coulomb classifiers* introduced by Hochreiter, Mozer, and Obermayer [8] that relies on an analogy

with electrostatics. It can also be regarded as a nonlinear extension of semi-supervised learning on graphs based on *Gaussian random field model* proposed by Zhu, Ghahramani, and Lafferty [21]. Therefore, our model provides a unified algorithmic framework to a larger class of the *graph regularization methods*, which has come to be a major current research topic in the semi-supervised learning literature after the Gaussian random field model; see the survey [20].

This paper is organized as follows. In Section 2, before introducing our classifier, we discuss a general framework for semi-supervised learning using *monotropic programming*. This framework is very flexible and clarifies the mathematical structure of our classifier. Then we introduce electric network classifiers as its special case. In Section 3, we show experimental results.

2. Electric Network Classifiers

In this section, we introduce electric network classifiers and investigate their mathematical properties, with emphasis on their connection to the standard C-SVM framework of [16].

Let V be an input data space, $U \subseteq V$ a training data set, and $y : U \to \{-1, 1\}$ its label. Our task is to classify the unlabeled data set $V \setminus U$. Our classifier is essentially based on the minimization of the sum of empirical risk functional $\Omega_{\rm emp}$ and regularizer $\Omega_{\rm reg}$ as

$$\min_{p:V\to\mathbf{R}} \Omega_{\rm emp}(\{p_j, y_j \mid j \in U\}) + \Omega_{\rm reg}(p); \tag{2.1}$$

see [16, Chapter 4]. Here, $\Omega_{\rm emp}$ is a loss function penalizing the discrepancy between p_j and y_j in sign, and $\Omega_{\rm reg}$ is a regularizer relating values p of labeled and unlabeled data smoothly with respect to the discrete structure of data space V. Let p^* be an optimal solution of (2.1). Then we classify the data set according to the sign of p^* as

$$\begin{cases} p^*_i \geq 0 & \Rightarrow \quad \text{the label of } i \text{ is } +1, \\ p^*_i < 0 & \Rightarrow \quad \text{the label of } i \text{ is } -1. \end{cases} \tag{2.2}$$

Quite recently, many methods using graph Laplacian regularizer

$$\Omega_{\rm reg}(p) = \sum_{i,j\in V} w_{(i,j)}(p_i - p_j)^2 \tag{2.3}$$

have been introduced and intensively studied; see the survey [20, Section 6.1]. This regularizer is based on the following heuristics: if a pair of vertices is highly interrelated in some sense, then the pair tends to have the same label.

Our electric network classifier also essentially follows this type of regularizations motivated by electric network models; the electric potential represents labels on vertices, and the electric resistance represents the interrelation of pairs of vertices and provides the electric energy as a regularizer.

Although many existing methods take special forms of $\Omega_{\rm emp}$ to reduce (2.1) to the problem of solving linear equations, our method allows $\Omega_{\rm emp}$ and $\Omega_{\rm reg}$ to be more general convex functions, and provides a unified learning algorithm based on efficient network flow algorithms. It is noted that fast computation of graph regularization methods is one of the current central issues; see the survey [20, Section 6.3].

To clarify the mathematical structure of our method, we first propose a general framework for semi-supervised learning using *monotropic programming* of Rockafellar [15] and discuss its relationship to kernel methods. Next, we introduce electric network classifiers as its special case.

2.1. Monotropic programming framework for semi-supervised learning

To design a classifier, we assume an auxiliary space E together with a linear map $A : \mathbf{R}^E \to \mathbf{R}^V$, or a matrix called *structure matrix*, which represents a discrete structure of V. In the canonical case of a directed graph, V is the vertex set, E is the edge set, and A is the incidence matrix.

We consider a monotropic programming problem [15], which consists of the following dual pair of convex optimization problems:

$$\text{[P]} \quad \begin{cases} \min\limits_{\xi \in \mathbf{R}^E, u \in \mathbf{R}^U} & \sum\limits_{e \in E} f_e(\xi_e) + \sum\limits_{j \in U} g_j(u_j) \\ \text{s.t.} & (A\xi)_i = \begin{cases} 0 & \text{if } i \in V \setminus U, \\ u_i & \text{if } i \in U, \end{cases} \end{cases} \tag{2.4}$$

$$\text{[D]} \quad \begin{cases} \min\limits_{\eta \in \mathbf{R}^E, p \in \mathbf{R}^V} & \sum\limits_{e \in E} f_e^*(\eta_e) + \sum\limits_{j \in U} g_j^*(-p_j) \\ \text{s.t.} & \eta = A^\top p, \end{cases} \tag{2.5}$$

where $f_e, g_j : \mathbf{R} \to \mathbf{R} \cup \{+\infty\}$ are convex functions and $f_e^*, g_j^* : \mathbf{R} \to \mathbf{R} \cup \{+\infty\}$ are the *Legendre transforms* of f_e and g_j defined, respectively, as

$$f_e^*(\eta_e) = \sup_{\xi_e \in \mathbf{R}} \{\eta_e \xi_e - f_e(\xi_e)\}, \quad g_j^*(q_j) = \sup_{u_j \in \mathbf{R}} \{q_j u_j - g_j(u_j)\}. \tag{2.6}$$

The problem [D] will play the role of (2.1). Indeed, substituting $\eta = A^\top p$ in f_e^*, the problem [D] can be regarded as the minimization of the sum of an empirical risk functional $\Omega_{\text{emp}} = \sum_{j \in U} g_j^*(-p_j)$ and a regularizer $\Omega_{\text{reg}} = \sum_{e \in E} f_e^*((A^\top p)_e)$. Therefore, the convex function g_j^* represents a loss function. So we should choose g_j^* to be a function having the property

$$g_j^*(-p_j) \quad \begin{cases} = 0 & \text{if } 1 - y_j p_j \le 0, \\ > 0 & \text{otherwise,} \end{cases} \tag{2.7}$$

and g_j is determined as the Legendre transform of g_j^*. One of the choices of $g_j^*(-p_j)$ is $C \max(0, 1 - y_j p_j)$ for some $C > 0$. This corresponds to the C-SVM formulation; see Corollary 2.2. On the other hand, the convex functions f_e and f_e^* represent regularization. The canonical choice is the following pair of squared-norm type functions:

$$f_e(\xi_e) = r_e \xi_e^2/2, \quad f_e^*(\eta_e) = \eta_e^2/2r_e \quad (e \in E), \tag{2.8}$$

where r_e is a positive parameter.

On the basis of an optimal solution (η^*, p^*) to [D], we classify the data set V according to the sign of p^* as (2.2). We call this p^* an *optimal discriminant potential*. Although one might feel that the variables ξ, u, and η are redundant, it is convenient to retain those variables as they admit natural physical and algorithmic interpretations in the case that A is the adjacency matrix of a graph, which we will see in Section 2.2.

The relationship between our approach and kernel methods is revealed in the special case of f_e given by (2.8). Let $A^+ : \mathbf{R}^V \to \mathbf{R}^E$ be a *reflexive minimum-norm generalized inverse* of A with respect to the squared norm $\sum_{e \in E} r_e \xi_e^2/2$, i.e.,

$$AA^+A = A, \quad A^+AA^+ = A^+$$

and for any $y \in \operatorname{Im} A$, A^+y is the minimum norm point of $\{x \in \mathbf{R}^E \mid Ax = y\}$; see [13] for generalized inverses. From A^+, we define a positive semidefinite kernel $K : V \times V \to \mathbf{R}$ as

$$K(i,j) = ((A^+)^\top R A^+)_{ij} \quad (i,j \in V), \tag{2.9}$$

where R is the diagonal matrix whose diagonal entries are $\{r_e\}_{e\in E}$. Then, we have the following (see Section 2.4 for the proof).

Theorem 2.1. *Let D be a matrix satisfying* $\operatorname{Im} A = \operatorname{Ker} D$. *Then the problem* [P] *with f_e of* (2.8) *is equivalent to*

$$[\mathrm{P}'] \qquad \min_{u\in\mathbf{R}^U} \quad \frac{1}{2}\sum_{i,j\in U} K(i,j)u_iu_j + \sum_{j\in U} g_j(u_j) \tag{2.10}$$

$$\text{s.t.} \quad \sum_{j\in U} D_{kj}u_j = 0 \ (\forall k : \textit{row index of } D). \tag{2.11}$$

Let u^* be an optimal solution to [P′] *and μ an optimal Lagrange multiplier of the equality constraints* (2.11). *Then an optimal discriminant potential p^* is given as*

$$p^*_i = \sum_{j\in U} K(i,j)u^*_j + (D^\top\mu)_i \quad (i \in V). \tag{2.12}$$

Recall the C-SVM classifier [16], which is obtained by solving the following optimization problem

$$\text{[C-SVM]: } \min_{\alpha\in\mathbf{R}^U} \quad \frac{1}{2}\sum_{i,j\in U}\alpha_i\alpha_j y_i y_j K(i,j) - \sum_{i\in U}\alpha_i$$

$$\text{s.t.} \quad \sum_{i\in U} y_i\alpha_i = 0,\ 0 \le \alpha_i \le C \quad (i \in U),$$

where C is a penalty parameter that is a positive real number or $+\infty$. Let α^* be an optimal solution of [C-SVM] and b^* an optimal Lagrange multiplier of the equality constraint. Then SVM decision function $\mu : V \to \mathbf{R}$ is given as

$$\mu(i) = \sum_{j\in U} y_j\alpha^*_j K(j,i) + b^* \quad (i \in V). \tag{2.13}$$

The relationship to the C-SVM framework is summarized as follows, where we denote by $\mathbf{1} = (1,1,\ldots,1,1)^\top$ the vector of all ones.

Corollary 2.2. *If* $\operatorname{Im} A = \operatorname{Ker}\mathbf{1}$ (*i.e., $D = \mathbf{1}$ in* [P′]) *and*

$$g_j(u_j) \;=\; \begin{cases} -y_ju_j & \text{if } 0 \le y_ju_j \le C, \\ +\infty & \text{otherwise,} \end{cases} \tag{2.14}$$

$$g^*_j(q_j) \;=\; C\max(0, 1 + q_j/y_j) \tag{2.15}$$

for $j \in U$ with some positive parameter C, then the problem [P′] *coincides with the C-SVM and the optimal discriminant potential p^* defined as* (2.12) *coincides with the SVM decision function* (2.13).

Proof of Corollary 2.2. Note that $y_i = 1/y_i$ by $y_i \in \{1,-1\}$. By the transformation $\alpha_i = y_iu_i$ for $i \in U$, we obtain the standard C-SVM formulation. □

Remark 2.3. If data set V and structure matrix A are stored in computer memory, the computation of kernel matrix K is not necessary since we can obtain potential p^* by solving the dual problem [D] directly. This p^* generally does not coincide with the one given by (2.12) if the optimal potential is not unique.

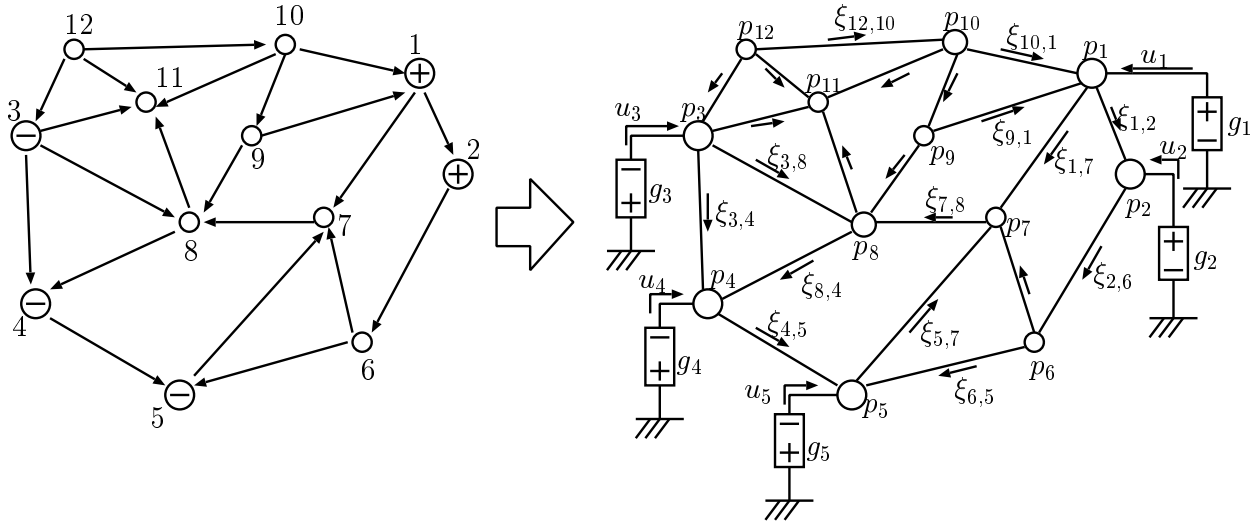

Figure 1: Physical interpretation of electric network classifiers

2.2. Electric network classifiers

We introduce *electric network classifiers* on graphs as a special case of the monotropic programming framework. Let $G = (V, E)$ be a directed graph, $U \subseteq V$ a training data, and $y : U \to \{-1, 1\}$ its label. Although there are real world data having such a natural graph structures justifying this setting, it is often necessary to construct such a graph structure from the given labeled and unlabeled data. In our argument, we assume that such a graph G connecting labeled and unlabeled data is given.

We treat the vertex set V as the data space, the edge set E as the auxiliary space, and the incidence matrix

$$A(v,e) = \begin{cases} 1 & \text{if } e \text{ leaves } v, \\ -1 & \text{if } e \text{ enters } v, \\ 0 & \text{otherwise,} \end{cases} \qquad (v \in V,\ e \in E) \tag{2.16}$$

as the structure matrix. We assume that G is connected. In this setting, the optimization problems [P] and [D] with convex functions $\{f_e\}_{e\in E}$ and $\{g_j\}_{j\in U}$ reduce respectively to

$$\min_{\xi\in\mathbf{R}^E} \sum_{e\in E} f_e(\xi_e) + \sum_{j\in U} g_j(u_j) \quad \text{s.t.} \sum_{j\in V:(j,i)\in E} \xi_{(j,i)} - \sum_{j\in V:(i,j)\in E} \xi_{(i,j)} = \begin{cases} 0 & \text{if } i \notin U \\ 1 & \text{if } i \in U \end{cases} \qquad (i \in V) \tag{2.17}$$

and

$$\min_{p\in\mathbf{R}^V} \sum_{(i,j)\in E} f^*_{(i,j)}(p_i - p_j) + \sum_{j\in U} g^*(-p_j). \tag{2.18}$$

This problem is exactly the same as the *nonlinear network flow problem* [9, 15]. Then the following physical interpretation is valid; see also Figure 1.

$\xi \in \mathbf{R}^E$	: currents on edges
$u \in \mathbf{R}^U$	: currents flowing into labeled vertices from the earth
f_e, g_j	: current energy on edges
$\eta \in \mathbf{R}^E$	: potential differences on edges
$p \in \mathbf{R}^V$	: potential on vertices
f^*_e, g^*_j	: potential energy on edges

Each potential on vertices is normalized so that the earth has zero potential. We call this classifier an *electric network classifier*. With general convex functions on the edges, the electric network classifier can incorporate various types of edge characteristics; influence of edge direction, unsymmetric dependence, and so on.

When the electric network consists exclusively of Ohmic resistors, we have

$$f_e(\xi_e) = r_e\xi_e^2/2 \quad (e \in E), \tag{2.19}$$

where r_e denotes the resistances. With the choice of g_j given in (2.14) our electric network classifier coincides with C-SVM using kernel (2.9), where the graph (V, E) is assumed to be connected. Furthermore, this kernel admits an intuitive interpretation, as follows.

Theorem 2.4 ([7]). *For f_e in* (2.19)*, the kernel K in* (2.9) *can be taken as*

$$K(i,j) = \{d(i,i_0) + d(j,i_0) - d(i,j)\}/2 \quad (i,j \in V), \tag{2.20}$$

where $d : V \times V \to \mathbf{R}$ is the electric distance defined as

$$d(i,j) = \textit{the electric resistance between } i \textit{ and } j \quad (i,j \in V) \tag{2.21}$$

and $i_0 \in V$ is an arbitrarily fixed root vertex.

This kernel K is called the *electric network kernel* and its explicit formulas for some classes of graphs are known [7]. Furthermore, if we take g_j and g_j^* as

$$g_j(u_j) = -y_j u_j, \quad g_j^*(q_j) = \begin{cases} 0 & \text{if } q_j = y_j \\ +\infty & \text{otherwise} \end{cases} \tag{2.22}$$

then this model is equivalent to the Gaussian random field model by [21] (for a certain edge weight) and the optimal potential p^* has the meaning that $(p_i^* + 1)/2$ is equal to the probability of a random walk, starting at i, that reaches $+1$ before reaching -1; see [5] for the relationship between electric networks and random walks.

Unsymmetric influence or dependence represented by edge directions can also be incorporated in our electric network classifier, while Ohmic resistors, having a symmetric characteristic $f_e(\xi_e) = f_e(-\xi_e)$, can only represent symmetric dependence. Unsymmetric dependence appears, for example, in expressing the link structure of the Web or the citation graph of papers. To represent unsymmetric dependence for an edge $e \in E$, we introduce an unsymmetric electric resistor that has the current energy f_e given as

$$f_e(\xi_e) = \begin{cases} r_e^+\xi_e^2/2 & \text{if } \xi_e \geq 0 \\ r_e^-\xi_e^2/2 & \text{if } \xi_e < 0 \end{cases} \tag{2.23}$$

where r_e^+ and r_e^- are electric resistances (> 0) in positive and negative directions, respectively. In particular, taking sufficiently large r_e^-, we can represent a series connection of a diode and a resistor as Figure 2. Since a diode from i to j imposes strong penalty on the case $p_j > p_i$, this construction serves as a model for the situation that if p_j is positive, then p_i tends to be positive. Such technique will be effective, for instance, for the citation graph of research papers if the following is true: if a paper A refers to a paper B having a positive label, then paper A tends to have a positive label, and the converse does not necessarily hold.

Furthermore, C-SVM interpretation is also possible (see Section 2.4 for the proof).

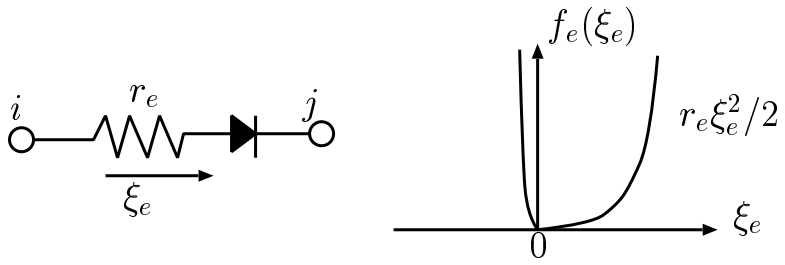

Figure 2: A series connection of a diode and a resistor

Theorem 2.5. *Consider the problems* [P] *and* [D] *with* f_e *of* (2.23) *for each edge* $e \in E$ *and some convex functions* $\{g_j\}_{j\in U}$. *Let* (ξ^*, u^*) *and* (η^*, p^*) *be optimal solutions to* [P] *and* [D], *respectively. Consider the modified problems* [P*] *and* [D*] *with* $\tilde{f}_e$ *defined as*

$$\tilde{f}_e(\xi_e) = \hat{r}_e\xi_e^2/2 \quad \textit{with} \quad \hat{r}_e = \begin{cases} r_e^+ & \textit{if } \xi_e^* \geq 0 \\ r_e^- & \textit{if } \xi_e^* < 0 \end{cases} \tag{2.24}$$

for each edge $e \in E$ *and the same* $\{g_j\}_{j\in U}$. *Then* (ξ^*, u^*) *and* (η^*, p^*) *are also optimal to* [P*] *and* [D*]. *In particular, if we choose* g_j *as* (2.14), u^* *is an optimal solution to the C-SVM problem with the electric network kernel with respect to the resistance* (2.24).

Summarizing our discussion, we give a learning algorithm using the electric network classifier in a generic form as follows.

Electric network classifier

Input: Data set V and training set $U \subseteq V$ with its label $y : U \to \{-1, +1\}$.

Output: Optimal discriminant potential $p^* : V \to \mathbf{R}$.

S1: Construct a graph that represents the discrete structure of V.

S2: Set cost function f_e on each edge $e \in E$.

S3: For each $i \in U$, connect the loss function g_i as a voltage source between training data i and the artificial earth.

S4: Obtain optimal potential p^* by solving the corresponding convex cost network flow problem; see the next subsection.

S5: Normalize the optimal potential so that the earth has zero potential, i.e., $p^* \leftarrow p^* - p_0^*\mathbf{1}$ with the earth potential p_0^*.

S6: Classify the data set by (2.2) according to the optimal potential p^*.

2.3. ASSP algorithm for convex cost network flow problems

As was seen in the previous subsection, the learning by the electric network classifier is a convex cost network flow problem. Therefore we can solve it by convex cost network flow algorithms. Several algorithms for this type of problems have been proposed in the network flow theory literature; see [3, Chapter 14],[1, 2, 4, 11]. Here we employ the *auction/sequential shortest path* (ASSP) method by Bertsekas, Polymenakos, and Tseng [4]. This method is implemented to the code `asspg` by Guerriero and Tseng [6] (in a more general form), available at http://www.math.washington.edu/~tseng/. Since this program uses a primal-dual type algorithm, we can obtain an optimal potential from this program. Recall that what we need is the potential, and not the flow.

In the following, we briefly sketch the ASSP algorithm and discuss its complexity; see [4] for details. Let $G = (V, E)$ be a directed graph equipped with convex cost functions

$\{f_e : \mathbf{R} \to \mathbf{R} \cup \{+\infty\} \mid e \in E\}$ on edges. The convex cost network flow problem is formulated as follows.

$$\min_{\xi \in \mathbf{R}^E} \sum_{e \in E} f_e(\xi_e) \quad \text{s.t.} \quad \sum_{j \in V:(j,i) \in E} \xi_{(j,i)} - \sum_{j \in V:(i,j) \in E} \xi_{(i,j)} = 0 \quad (i \in V). \tag{2.25}$$

The dual problem is

$$\min_{p \in \mathbf{R}^V} \sum_{(i,j) \in E} f^*_{(i,j)}(p_i - p_j), \tag{2.26}$$

where f^*_e is the Legendre transform of f_e.

Our problems (2.17) and (2.18) of the electric network classifier on the graph (V, E) with a test data $U \subseteq V$ and convex functions $\{f_e\}_{e \in E}$ and $\{g_j\}_{j \in U}$ can be reduced to the problems (2.25) and (2.26) by introducing a new artificial vertex "earth" numbered 0, and setting $V \leftarrow V \cup \{0\}$ and $E \leftarrow E \cup \{(j, 0) \mid j \in U)\}$ with convex functions $\{f_e\}$ on the original edges and $\{g_j\}$ on the edges between U and the earth; see Figure 1. If we fix $p_0 = 0$, our ENC problems (2.17) and (2.18) are equivalent to (2.25) and (2.26). Conversely, since the objective function of (2.26) is invariant under the transformation $p \leftarrow p + \alpha \mathbf{1}$ with $\alpha \in \mathbf{R}$, an optimal solution p^* of (2.26) yields an optimal discriminant potential $p^* \leftarrow p^* - p^*_0 \mathbf{1}$ that is normalized to $p^*_0 = 0$.

To describe the ASSP algorithm, we need some notation. For a flow $\xi \in \mathbf{R}^E$, the *surplus* s_i at vertex $i \in V$ is defined as

$$s_i = \sum_{j \in V:(j,i) \in E} \xi_{(j,i)} - \sum_{j \in V:(i,j) \in E} \xi_{(i,j)}.$$

Let $f^+_e(\xi_e)$ and $f^-_e(\xi_e)$ be the right derivative and the left derivative of f_e at $\xi_e \in \mathbf{R}$, respectively. We say that a flow-potential pair $(\xi, p) \in \mathbf{R}^E \times \mathbf{R}^V$ satisfies ϵ-complementary slackness condition (ϵ-CS condition) if it satisfies

$$f_{(i,j)}(\xi_{(i,j)}) < +\infty, \quad \text{and} \quad f^-_{(i,j)}(\xi_{(i,j)}) - \epsilon \le p_i - p_j \le f^+_{(i,j)}(\xi_{(i,j)}) + \epsilon \quad ((i,j) \in E),$$

where ϵ is a positive parameter representing the precision of a solution. Indeed, it is shown [4] that if a feasible flow-potential pair (ξ, p) satisfies ϵ-CS condition, then ξ and p are primal and dual optimal, respectively within a factor proportional to ϵ.

For a pair $(\xi, p) \in \mathbf{R}^E \times \mathbf{R}^V$ satisfying ϵ-CS, we define the *admissible graph* $G^* = (V, E^*)$ as

$$\begin{aligned} E^* \;=\; & \{(i,j) \mid (i,j) \in E,\ \epsilon/2 < p_i - p_j - f^+_{(i,j)}(\xi_{(i,j)}) \le \epsilon\} \\ & \cup \{(j,i) \mid (i,j) \in E,\ -\epsilon \le p_i - p_j - f^-_{(i,j)}(\xi_{(i,j)}) < -\epsilon/2\}. \end{aligned}$$

For an edge $(i,j) \in E^*$, the *flow margin* $\delta_{(i,j)}$ is defined as

$$\delta_{(i,j)} = \begin{cases} \sup\{\delta \ge 0 \mid p_i - p_j \ge f^+_{(i,j)}(\xi_{(i,j)} + \delta)\} & \text{if } (i,j) \in E, \\ \sup\{\delta \ge 0 \mid p_j - p_i \le f^-_{(j,i)}(\xi_{(j,i)} - \delta)\} & \text{if } (j,i) \in E. \end{cases} \tag{2.27}$$

The ASSP algorithm tries to make ϵ-CS pair (ξ, p) feasible by pushing a flow from a vertex having positive surplus to a vertex having negative surplus while keeping ϵ-CS

condition. For sufficiently small $\epsilon > 0$, a feasible ϵ-CS pair (ξ, p) is almost optimal.

Algorithm ASSP

Input: Initial precision $\epsilon^0 > 0$ and target precision $\bar{\epsilon} > 0$.

Output: A feasible flow-potential pair (ξ, p) satisfying $\bar{\epsilon}$-CS.

Initialization: Set $\epsilon \leftarrow \epsilon^0$ and set $(\xi, p) \in \mathbf{R}^E \times \mathbf{R}^V$ so that it satisfies ϵ-CS and G^* is acyclic; one possibility is to select arbitrary potential p and to set flow ξ as

$$\xi_{(i,j)} = \sup\{\xi \mid f^+_{(i,j)}(\xi) \leq p_i - p_j - \epsilon/2\}, \tag{2.28}$$

which implies $E^* = \emptyset$.

s1: Select a vertex $i \in V$ with positive surplus; if no such vertex exists, go to **ϵ-scaling**.

s2: Construct a directed path P on G^* from i to a vertex having negative surplus by the following depth first search:

s2-1 If the terminal vertex j of (current) P has negative surplus, then go to **s3**.

s2-2 If there exists an edge $e \in E^*$ going out of j, then extend P by tracing e and go to **s2-1**.

s2-3 Increase p_j as long as (ξ, p) satisfies ϵ-CS. If $j \neq i$, delete from P the terminal vertex j as well as the edge of P ending at j.

s2-4 Go to **s2-1**.

(In fact, G^* remains acyclic in this process, and P is always a simple path.)

s3: Push a flow from i to j through P with amount $\min\{s_i, -s_j, \min_{e \in P} \delta_e\}$ and go to **s1**.

ϵ-scaling: If $\epsilon < \bar{\epsilon}$, then stop; the current (ξ, p) is feasible and satisfies $\bar{\epsilon}$-CS. Otherwise set $\epsilon \leftarrow \epsilon/2$ and, for each edge (i, j) violating ϵ-CS, set $\xi_{(i,j)} \leftarrow \sup\{\xi \mid f^+_{(i,j)}(\xi) \leq p_i - p_j - \epsilon/2\}$.

See [4] for the validity of ASSP algorithm. For implementation, we need calculations of left and right derivatives, flow margin, and (2.28). If cost function f_e is linear, quadratic, or piecewise linear, then left and right derivatives, flow margin, and (2.28) can be easily calculated. Of course, unsymmetric resistances (2.23) are also straightforward to implement.

As for the complexity of this algorithm, the running time is bounded by $O(|V|^3 \ln(\epsilon^0/\bar{\epsilon}))$, where we assume that (2.28) and (2.27) can be calculated in constant time. The memory requirement is clearly $O(|V|+|E|)$. Therefore, there is no memory requirement in addition to that for keeping the graph structure. Although this upper bound of running time appears to be costly, each iteration requires no matrix computations but only cheaper graph operations are needed. Furthermore, for the learning, it is not necessary to select the precision $\bar{\epsilon}$ so severely. Therefore, like SMO (sequential minimal optimization) algorithm for C-SVM, this algorithm is expected to achieve faster learning and have more scalability for larger problems, compared with the SVM with diffusion kernels; the computation of the diffusion kernel

$$K(i, j) := (\exp(-\beta L))(i, j) = \sum_{k=0}^{\infty} \frac{(-\beta)^k}{k!} L^k(i, j) \tag{2.29}$$

requires $O(|V|^3)$ operations for matrix diagonalizations and $O(|V|^2)$ memory requirement, where L is the Laplacian matrix of the graph and β is the diffusion parameter.

Remark 2.6. The sparsity of a graph does not affect the upper bound $O(|V|^3 \ln(\epsilon^0/\bar{\epsilon}))$ of ASSP algorithm. If the graph is sparse, the *cancel-and-tighten algorithm* by Karzanov

and McCormik [11] may be more efficient since its running time is bounded by $O((|E|^2 + |E||V|\ln|V|)\ln(\epsilon^0/\bar{\epsilon}))$.

2.4. Proofs

We use some basic notation and properties from convex analysis [14]. First, we note that (ξ^*, u^*) and (η^*, p^*) are optimal to the monotropic programming problems [P] and [D] if and only if they are feasible and satisfy

$$f_e(\xi_e^*) + f_e^*(\eta_e^*) = \xi_e^*\eta_e^*, \quad (e \in E), \tag{2.30}$$

$$g_j(u_j^*) + g_j^*(-p_j^*) = -u_j^*p_j^* \quad (j \in U) \tag{2.31}$$

(see [15, Chapter 8] [9, Chapter IV] for optimality conditions for nonlinear network flow problems and also see [15, Chapter 11] for general monotropic programming). Second, for a convex function $f : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$, $x^* \in \mathbf{R}^n$ is a minimizer of f if and only if it satisfies $0 \in \partial f(x^*)$, where $\partial f(x^*) := \{p \in \mathbf{R}^n \mid f(x) - f(x^*) \geq p^\top(x - x^*)\}$ is the *subdifferential* of f at x^* (see [14, Section 72] for optimality conditions of convex functions using subdifferential).

Proof of Theorem 2.1. The problem [P] with f_e defined as (2.8) is rewritten as

$$\min_{u \in \mathbf{R}^U} \quad \min_{\xi} \left\{ \xi^\top R\xi/2 \;\middle|\; A\xi = \begin{pmatrix} 0 \\ u \end{pmatrix} \right\} + \sum_{j \in U} g_j(u_j) \quad \text{s.t.} \quad D \begin{pmatrix} 0 \\ u \end{pmatrix} = 0.$$

Hence, using a reflexive minimum-norm generalized inverse A^+, the inner optimizer ξ^* is given as

$$\xi^* = A^+ \begin{pmatrix} 0 \\ u \end{pmatrix}.$$

Then, the inner optimal value is given by $(1/2)\sum_{i,j \in U}((A^+)^\top RA^+)_{ij}u_iu_j$. Thus, we obtain the first statement of Theorem 2.1. Next, we show that p^* defined as (2.12) and $\eta^* := A^\top p^*$ are optimal to [D]. Let u^* be an optimal solution of [P′] and μ an optimal Lagrange multiplier of Lagrange function of [P′]

$$\frac{1}{2}\sum_{i,j \in U} K(i,j)u_iu_j + \sum_{j \in U} g_j(u_j) + \mu^\top D \begin{pmatrix} 0 \\ u \end{pmatrix}. \tag{2.32}$$

Then, the subdifferential of the Lagrange function (2.32) at (u^*, μ) contains zero (see [14, Theorem 28.3]). From this, we have

$$\partial g_i(u_i^*) \ni -\sum_{j \in U} K(j,i)u_j^* - (D^\top \mu)_i = -p_i^* \quad (i \in U).$$

Hence, $-p_i^* \in \partial g_i(u_i^*)$ implies (2.31) (see [14, Theorem 23.5]). On the other hand, we have

$$\begin{aligned} \eta^* &= A^\top p^* = A^\top \left\{ (A^+)^\top RA^+ \begin{pmatrix} 0 \\ u^* \end{pmatrix} + D^\top \mu \right\} \\ &= RA^+AA^+ \begin{pmatrix} 0 \\ u^* \end{pmatrix} = RA^+ \begin{pmatrix} 0 \\ u^* \end{pmatrix} = R\xi^*, \end{aligned}$$

where the third equality follows from $RA^+A = (A^+A)^\top R$ (a consequence of minimum-normness of A^+ with respect to R) and $\mathrm{Im}\, A = \mathrm{Ker}\, D$, and the fourth follows from reflexivity of A^+. From $\eta_e^* = r_e\xi_e^*$, we obtain (2.30). □

Proof of Theorem 2.5. It is easy to check that in the modified problem, (ξ^*, u^*) and (η^*, p^*) satisfy the optimality conditions (2.30) and (2.31). □

3. Experimental Results

In this subsection, we describe experimental results. To apply our method, it is often necessary to construct a graph structure from the given data set; of course, there are some data having natural graph structures like DNA biological networks.

Here, we use 20 newsgroups corpus for the performance evaluations. These are available at http://www.cs.umass.edu/~mccallum/. Each document of the 20 newsgroups is processed into the bag of words representation by the Mallet tool kit. We select three binary problems,

(1) rec.auto vs. rec.motorcycles,
(2) soc.religion.christian vs. alt.atheism, and
(3) comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware.

Graph structures are constructed as follows. We connect each document to its five nearest neighbors, where the distance on documents is measured by the cosine similarity $1-(x,y)/\|x\|\|y\|$ for a vector representation of documents x,y and the ordinary inner product $(\cdot,\cdot)$. We use this distance as edge weight. For the electric network classifier, we connect positive and negative data to the earth, and take f_e of (2.19), r_e as edge weight, and g_j as (2.14); recall Figure 1. Then we obtain the whole graph structure and apply the ASSP algorithm by using `asspg` code. The initial precision ϵ^0 for ASSP algorithm is set to be the default value of the `asspg` code, which is one third of the maximum absolute value of the linear costs of the edges; $\epsilon^0 = 1/3$ in our case. The target precision $\bar{\epsilon}$ is set to be 0.0001. Note that the optimal discriminant potential is normalized so that the earth has zero potential. Resulting graph sizes are (1) 1995 vertices and 17963 edges, (2) 1996 vertices and 19960 edges and (3) 1993 vertices and 19930 edges.

For comparison, we use C-SVM with linear and diffusion kernels which are implemented to LIBSVM package available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. For the diffusion kernel, we use weighted Laplacian, i.e., $L(i,j) = -1/r_{(i,j)} (i \neq j)$ and $L(i,i) = \sum_j 1/r_{(i,j)}$, where r is the edge weight. Then the diffusion kernel is defined by $(\exp(-\beta L))(i, j)$ with the diffusion parameter $\beta > 0$. By preliminary experiments, the parameters for electric network classifier, C-SVM with linear kernel, and C-SVM with diffusion kernel are selected as follows: For electric network classifier, C of g_j (2.14) is selected as (1) $C = 10$, (2) $C = 20$ and (3) $C = 10$. For C-SVM with linear kernel, C-SVM parameter C is selected as (1) $C = 0.1$, (2) $C = 0.01$ and (3) $C = 0.01$. For C-SVM with diffusion kernel, the diffusion parameter β and C-SVM parameter C are selected as (1)$\beta = 0.2$, $C = 10$, (2) $\beta = 0.2$, $C = 20$ and (3) $\beta = 0.3$, $C = 10$.

A half of the whole documents are randomly selected as unlabeled data for the test. The training labeled data are selected in a specific ratio from the rest half. Therefore, the whole graph structure consists of the unlabeled data for the test, which are a half of the whole data, and training labeled and unlabeled data. Experiments are carried out, by varying the ratio of training labeled data. This procedure is repeated ten times. Average accuracy (the ratio of correct answers for unlabeled test data) are reported in Figure 3. We mention that our primary interest lies in the effectiveness of propagation of labels via unlabeled data, and the goal of the semi-supervised learning is to achieve high accuracy using a small portion of training labeled set.

The results show that the performance of our electric network classifier is fairly good, compared with C-SVM with linear and diffusion kernels, except that in the data set (3) linear SVM shows high accuracy in the region of high labeled data ratio. In particular, in the range of small ratio of labeled data, our classifier shows good performance. This implies

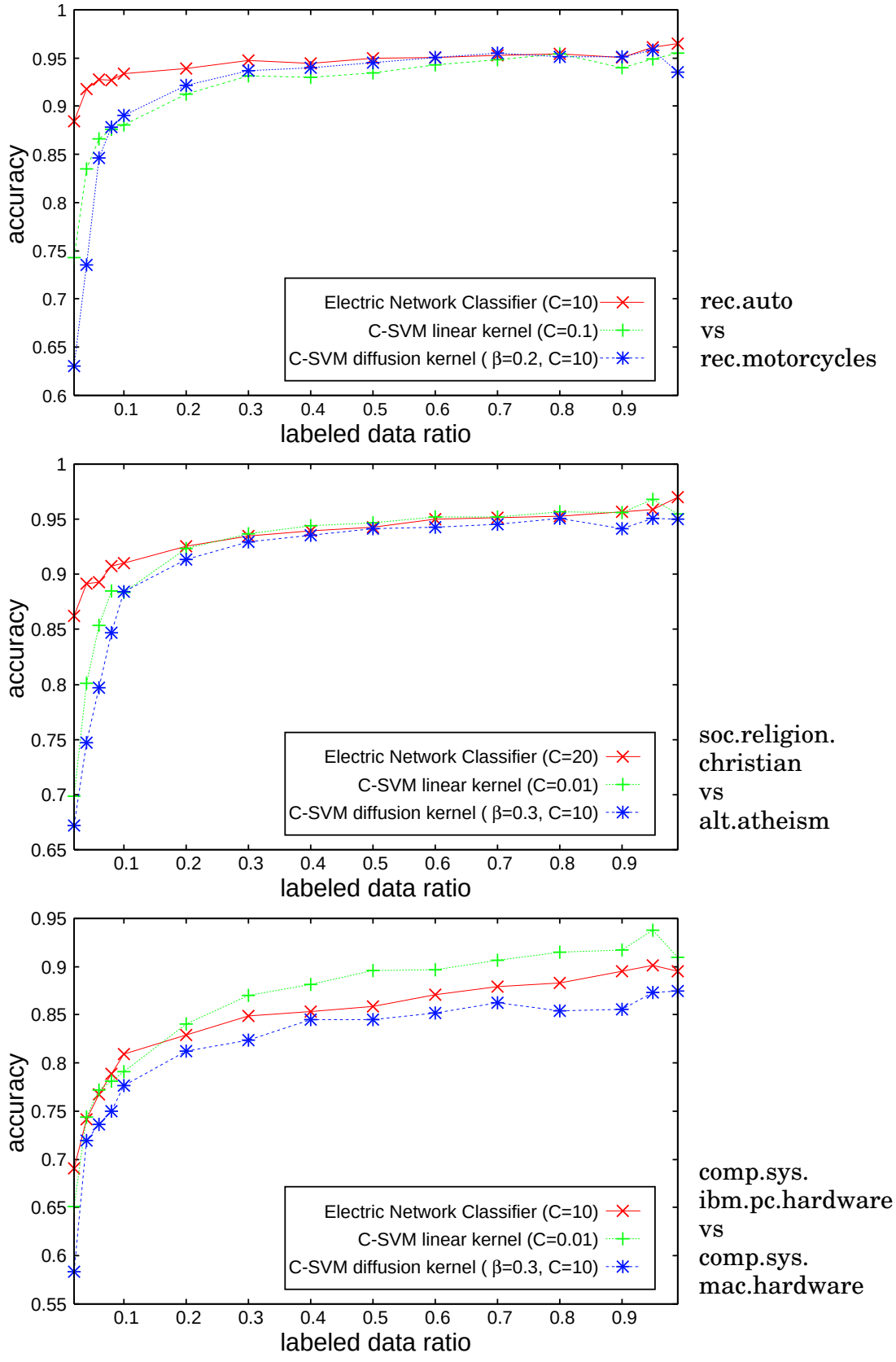

Figure 3: Accuracy of each classifier on three data sets

effectiveness of semi-supervised learning. Furthermore we emphasize that the learning time of our classifier is very short compared with the diffusion kernel, since diagonalization for computing diffusion kernel matrix is quite heavy. Indeed, average learning times of our classifier using `asspg` for data sets (1), (2), and (3) are 0.97 (s), 1.02 (s), and 1.27 (s), respectively. On the other hand, average computational times for the construction of diffusion kernel matrix $\exp(-\beta L)$ through diagonalizations for (1), (2), and (3) are 92.4 (s), 91.4 (s), and 92.5 (s), respectively. This experiment was done by Athron 64 2.2GHz CPU machine with 2GB memory, and matrix diagonalizations for the diffusion kernel were done by Matlab. This indicates that our classifier has the scalability for large problems.

Acknowledgements

The authors thank anonymous referees for helpful suggestions. This work is supported by the 21st Century COE Program on Information Science and Technology Strategic Core, and by a Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] R.K. Ahuja, D.S. Hochbaum, and J.B. Orlin: Solving the convex cost integer dual network flow problem. *Management Science*, **49** (2003), 950–964.

[2] R.K. Ahuja, D.S. Hochbaum, and J.B. Orlin: A cut-based algorithm for the nonlinear dual of the minimum cost network flow problem. *Algorithmica*, **39** (2004), 189–208.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin: *Network Flows—Theory, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, 1993).

[4] D.P. Bertsekas, L.C. Polymenakos, and P. Tseng: Epsilon-relaxation and auction methods for separable convex cost network flow problems. In P.M. Pardalos, D.W. Hearn, and W.W. Hager (eds.): *Network Optimization*, Lecture Notes in Economics and Mathematical Systems, **450** (Springer-Verlag, Berlin, 1997), 103–126.

[5] P.G. Doyle and J.L. Snell: *Random Walks and Electric Networks* (The Mathematical Association of America, 1984).

[6] F. Guerriero and P. Tseng: Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, **115** (2002), 113–144.

[7] H. Hirai, K. Murota, and M. Rikitoku: SVM kernel by electric network. *Pacific Journal of Optimization*, **1** (2005), 509–526.

[8] S. Hochreiter, M.C. Mozer, and K. Obermayer: Coulomb classifiers: generalizing support vector machines via an analogy to electrostatic systems. *Neural Information Processing Systems*, **15** (2003), 545–552.

[9] M. Iri: *Network Flow, Transportation and Scheduling—Theory and Algorithm* (Academic Press, New York, 1969).

[10] J. Kandola, J. Shawe-Taylor, and N. Cristianini: Learning semantic similarity. *Neural Information Processing Systems*, **15** (2003), 657–664.

[11] A.V. Karzanov and S.T. McCormick: Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal on Computing*, **26** (1997), 1245–1275.

[12] R.I. Kondor and J. Lafferty: Diffusion kernels on graphs and other discrete structures.

Proceedings of the 19th International Conference on Machine Learning (Morgan Kaufmann, 2002), 315–322.

[13] C.R. Rao and S.K. Mitra: *Generalized Inverse of Matrices and Its Applications* (Wiley, New York, 1971).

[14] R.T. Rockafellar: *Convex Analysis* (Princeton University Press, Princeton, 1970).

[15] R.T. Rockafellar: *Network Flows and Monotropic Optimization* (Wiley, New York, 1984).

[16] B. Schölkopf and A.J. Smola: *Learning with Kernels* (MIT Press, 2002).

[17] A.J. Smola and R. Kondor: Kernels and regularization on graphs. *Proceedings of the Annual Conference on Computational Learning Theory,* Lecture Notes in Computer Science, **2777** (Springer, 2003), 144–158.

[18] K. Tsuda and W.S. Noble: Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, **20** (2004), 326–333.

[19] J.-P. Vert and M. Kanehisa: Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. *Neural Information Processing Systems*, **15** (2003), 1425–1432.

[20] X. Zhu: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005. `http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`

[21] X. Zhu, Z. Ghahramani, and J. Lafferty: Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the Twentieth International Conference of Machine Learning* (2003), 912–919.

Hiroshi Hirai
Research Institute for Mathematical Science
Kyoto University
Kyoto 606-8502, Japan
E-mail: `hirai@kurims.kyoto-u.ac.jp`