Enhancing Label Inference Algorithms Considering Vertex Importance in Graph-Based Semi-Supervised Learning

Byonghwa Oh
Hyundai Card Corporation
3 Uisadang–daero, Yeongdeungpo–gu, Seoul 07237, Korea
Email: byonghwa.oh@hyundaicard.com

Jihoon Yang
Department of Computer Science and Engineering
Sogang University
35 Baekbeom–ro, Mapo–gu, Seoul 04107, Korea
Email: yangjh@sogang.ac.kr

***Abstract*—Graph-based semi-supervised learning has recently come into focus for to its two defining phases: graph construction, which converts the data into a graph, and label inference, which predicts the appropriate labels for unlabeled data using the constructed graph. And the label inference is based on the smoothness assumption of semi-supervised learning. In this study, we propose an enhanced label inference approach which incorporates the importance of each vertex into the existing inference algorithms to improve the prediction capabilities of the algorithms. We also present extensions of three algorithms which are capable of taking the vertex importance variable to apply in learning. Experiments show that our algorithms perform better than the base algorithms on a variety of datasets, especially when the data is less smooth over the graphs.**

I. Introduction

While there may be large number of databases available, only a small portion of the data is labelled. Moreover, labels are relatively hard to obtain because the act of labelling the data requires human labor or additional resources. Semi-supervised learning (SSL) has emerged as a solution to this problem and has been applied to many machine learning use cases. SSL employs both labeled and unlabeled data in order to train a model and has the potential to improve performance over models which are trained via supervised learning tasks. There has been much research dedicated to SSL (See [1], [2]), and many studies have shown that of the SSL approaches, graph-based SSL has many advantages: efficacy in practice (outperforms other SSL approaches), convexity (majority of graph-based SSL algorithms optimize a convex objective), ease of scalability (many graph-based SSL algorithms can be easily parallelized contrary to non-graph-based techniques) [3].

A SSL algorithm learns a function f:$X \rightarrow Y$ which associate labels Y with data X. However there is no information about this mapping in unlabeled data, so some assumptions are required to infer useful information related to the mapping. The most common assumptions are the smoothness assumption and the manifold assumption. The smoothness assmption implies that 'if two points x_1 and x_2 in a high density region are close then their corresponding labels y_1 and y_2 are also close'. Thus many algorithms minimize the total inconsistency of labels in neighborhood vertices. The manifold assumption states that 'high dimensional data generally lies within a low dimensional manifold'. The graph is regarded as a proxy for the manifold in graph-based SSL. By combining these two assumptions, graph-based SSL aims to learn a label function from X to Y using a graph constructed from the data.

Generally, we can expect a sufficiently good prediction result if the training data satisfies the smoothness assumption on the graph. But not every graph is smooth enough to use to infer labels. Such graphs would lead to poorer prediction results than those produced via supervised learning. Therefore we sought out measures to improve the performance of graph-based SSL so as to improve the application of the smoothness assumption to existing algorithms, by introducing the concept of vertex importance (or weight) and propose a method which lowers the importance of vertices in order to improve the application of the smoothness assumption in the label inference.

Each graph-based SSL algorithm is unique in its structure and learning approach. Therefore, we must design techniques which enable us to apply the concept of vertex importance dynamically for a variety of algorithms. Therefore we present an extension of one existing algorithm, Probabilistic Graph-based Pointwise Smoothness (PGP) [4], as well as extensions of two other widely applied algorithms, Gaussian Random Fields (GRF) [5] and Label Spreading (LS) [6]. We further propose proofs of convergence of the algorithms. These results are able to improve the performance of existing qualified algorithms and thus are expected to contribute to several studies and application areas which use graph-based SSL.

II. Related Work: Graph-based SSL

In the case of SSL, the training data $\{D_l, D_u\}$ consists of l labeled instances D_l=$\{(\mathrm{x}_i,y_i)\}_{i=1}^{l}$ and u unlabeled instances D_u=$\{\mathrm{x}_i\}_{i=1}^{u}$ ($n = l + u$, generally $u \gg l$). Graph construction converts the data D_l, D_u into a graph G=(V,W), which will be used in the next phase. Here V is the set of vertices satisfying $|V|$=n and $W \in \mathbb{R}^{n \times n}$ is the edge weights. Typically an undirected graph, i.e. W_{ij}=W_{ji}, is used, and when W_{ij}=0, it implies the absence of an edge between vertices i and j.

The label inference phase allows for the provision of initial labels on graph vertices and for a method to infer labels on

unlabeled vertices. The most typical algorithm for doing this is GRF, which minimizes the quadratic energy function on real-valued label functions of vertices $F{:}V{\rightarrow}\mathbb{R}^{n\times c}$ (where F_i is a label function of the vertex i consisting of c real values and c is the number of classes). In this way, we are able to adjust the labels for unlabeled vertices that are nearby in the graph to have similar labels by the following objective:

$$\underset{F\in\mathbb{R}^{n\times c}, F_L=Y_L}{\operatorname{argmin}} \sum_{i,j\in V}^{n} W_{ij}\|F_i - F_j\|^2 \tag{1}$$

where $Y_L{\in}\mathbb{R}^{l\times c}$ is the seed labels of D_l, and $F_L{=}Y_L$ is a constraint that preserves the labels of D_l during learning and also after learning. For F_L and Y_L, only a single element of each row is 1 (the class representative) and others are zero. $F_U{\in}\mathbb{R}^{u\times c}$ is the labels of D_u, and initialized to zero (Note that $F{=}\{F_L, F_U\}$). Meanwhile, the graph Laplacian is given by $L{=}D{-}W$, where D is a diagonal degree matrix with $D_{ii}{=}\sum_j W_{ij}$ and $D_{ij}{=}0$ for every $i \neq j$. Thus $\sum W_{ij}(F_i{-}F_j)^2$ in (1) is simply expressed as a form of multiplications, $F^T LF$.

The potential issue of using GRF with k-NN graphs is the potential resulting degenerate solutions in which most of the unlabeled vertices are classified as a single class, caused by some vertices having a relatively large number of degrees. To solve this, [6] proposed an LS objective by modifying (1) as:

$$\frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}\left\|\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right\|^2 + \mu\sum_{i=1}^{n}\|F_i - Y_i\|^2\right) \tag{2}$$

where Y_i is a zero vector for $i{\in}V_u$ (unlabeled data) and for $i{\in}V_l$ (labeled data), Y_i consists of all zeros except for $Y_{ik}{=}1$ if i has the k-th label.

The first term of (2) has essentially the same notion of non-smoothness in GRF, but there is a major difference in that it reduces the impact of the label functions by the degree of each vertex for regularizing the influence of the vertices with high degrees. The second term means that the labels inferred are not to be exactly same as the seed labels. Parameter μ balances the influence of the terms. The condition $Y_i{=}0$ for $i{\in}V_u$ also seeks the sparse representation of F.

GRF and LS have simple structures and are easy to implement. Therefore they have been applied in various languages and are being used as the default methods of several semi-supervised learning toolkits such as Junto, scikit-learn, and so on. Moreover they are often used as the main graph inference algorithms in many research studies in graph construction.

PGP, which operates on a new form of smoothness called pointwise smoothness, shows robust and accurate classification results compared to existing cutting-edge algorithms. Therefore we also propose an extension method to PGP, in addition to those presented for GRF and LS.

III. Label Inference Algorithms Considering Vertex Importance

A. Necessity of Considering Vertex Importance

Each label inference algorithm has a common mathematical term which reflects the smoothness assumption. That is, they expect a positive effect on accurate labeling by supposing the smoothness assumption works well on graphs, definitely. However there may be some graphs that do not meet such supposition, and the application of existing algorithms on such non-smooth graphs can lead to poor inference results.

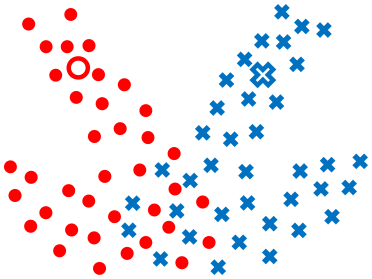

Fig. 1. Example of how a bad graph can lead to poor performance using existing label inference algorithms

Figure 1 illustrates an example of this phenomenon. There are two large symbols O (positive), X (negative) which lay in the upper part of the figure. The large symbols represent labeled instances. The other small symbols are unlabeled instances. Some positive and negative samples are mixed at the bottom and middle of the figure, and they have similar distances to the positive labeled sample and to the negative labeled sample. Therefore they are not likely to be classified properly. The problem here then, is that the samples in this area can propagate improper information about their nearest neighbors, resulting in inaccurate inferences for the instances which lie at both of the bottom-left area and the bottom-right area. We may get better results by eliminating the samples in the center area or by restricting their influence during the processes of inference. In other words, we can acquire better inference results by designing a method that identify aberrant data points and then reduce the impact of such samples.

Therefore it is essential for the algorithm to have an iterative solution for the changes in vertex importance at each iteration to take effect in the inference, and the convergence of it must be guaranteed even if weights of vertices change.

B. Proposed Algorithms

1) GRF with Vertex Weighting: In (1), W_{ij} determines application of the differences between the label functions of vertices i and j. Thus in order to alter the importance of vertex j, it is necessary to evaluate the importance ω_j first and stick $\omega_j W_{ij}$ into W_{ij}. The difference caused by F_j in the calculation of non-smoothness changes with respect to ω_j.

As such, to make possible changing the values of W while in the midst of learning, we must solve the objective (1) by applying an iterative algorithm. Two methods are generally known [7] but both are not guaranteed to converge in the case of changing values of W. Therefore we present a novel iterative method for GRF. First, the closed form solution of GRF is represented as follows:

$$F_U = (I - P_{UU})^{-1} P_{UL} F_L \tag{3}$$

where the labeled data takes the first l-th instances and the unlabeled data takes the l+1-th to l+u-th instances. So, P_{UU} occupies the bottom-left quadrant of P. P=$D^{-1}W$ is a row-normalized transition matrix constructed from the training data and it satisfies $\|P\|$=1 and $\|P_{UU}\|\leq$1. If $\|P_{UU}\|<1$ is always true, we can replace $(I\text{-}P_{UU})^{-1}$ by a series and then can derive an update equation used for iteration by applying the definition of the Neumann series. The Neumann series $(I\text{-}T)^{-1}$=$\sum_{k=0}^{\infty} T^k$ converges if T is a bounded linear opeartor (i.e., square matrix) and its norm is smaller than 1 ($\|T\|<1$). Unfortunately the norm of P_{UU} is not always smaller than 1.

So we must ensure $\|P_{UU}\|<1$ for any graph. The way is simple: forcing every sum of rows to be smaller than 1. Specifically, we can use D_{ii}=$\sum_j W_{ij}$+ϵ in place of D_{ii}=$\sum_j W_{ij}$ when evaluating P=$D^{-1}W$ where ϵ is a very small positive real number (ϵ=2.2204$\times 10^{-16}$ in Matlab). This method yields the same solution compared to the closed-form solution, and has been found to always converges experimentally. Thus we use this to derive the update equation (5) from (3). (3) is represented as follows if the Neumann series is applied:

$$F_U = \left(\sum_{i=1}^{\infty} P_{UU}^i\right) P_{UL} F_L \tag{4}$$

And the iteration equation is derived as follows:

$$\begin{aligned} F_U(t) &= \left(\sum_{i=1}^{t} P_{UU}^i\right) P_{UL} F_L \\ F_U(t-1) &= \left(\sum_{i=1}^{t-1} P_{UU}^i\right) P_{UL} F_L \\ \therefore F_U(t) &= P_{UU} F_U(t-1) + P_{UL} F_L \end{aligned} \tag{5}$$

Therefore the iterative algorithm of GRF with vertex weighting is performed as follows: First, at the current iteration, compute ω_i (the values of ω should be in [0, 1]) for every vertex i∈V, and then substitute the values of W with $\omega_j W_{ij}$ for every i and j. Next, evaluate P_{UU} by adopting the latter strategy of constraining $\|P_{UU}\|<1$. Finally, calculate $F_U(t)$ using the changed P_{UU}. We can obtain a unique solution by repeating the above procedures iteratively. In Section 4, we describe how the importance of vertices are calculated and the convergence proofs of the algorithm.

2) LS with Vertex Weighting: Unlike GRF, LS is extended by changing the objective which should be minimized. Analogous to the strategy of regularizing vertices with high degrees (Section 2), we can change the rate of influence of the vertex i by multiplying the importance ω_{ii} by F_i. ω_{ii} is a diagonal element of ω, similar to the relation between D_{ii} and D.

$$\frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij} \left\| \frac{\sqrt{\omega_{ii}}}{\sqrt{D_{ii}}} F_i - \frac{\sqrt{\omega_{jj}}}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - Y_i\|^2 \right) \tag{6}$$

To minimize (6), we first rewrite it as follows:

$$Q(F) = \frac{1}{2}\left\{\mathrm{tr}(\mathcal{F}^T L \mathcal{F}) + \mu \|F - Y\|_F^2\right\} \tag{7}$$

where $\mathcal{F} = \omega^{1/2} D^{-1/2} F$ and $L = D - W$.

And we take the derivative of $Q(F)$ with respect to F:

$$\begin{aligned} &\frac{\partial}{\partial F}\left\{\frac{1}{2}\Big(\mathrm{tr}(F^T \omega^{1/2} D^{-1/2}(D-W) D^{-1/2} \omega^{1/2} F) \right. \\ &\left. + \mu \|F - Y\|_F^2\Big)\right\} \\ &= (\omega - \omega^{1/2} S \omega^{1/2}) F + \mu(F - Y) \\ &= \omega F - S' F + \mu(F - Y) \end{aligned} \tag{8}$$

where S=$D^{-1/2}WD^{-1/2}$ and S'=$\omega^{1/2}S\omega^{1/2}$. We can get a closed-form solution which minimizes (7), by replacing F by F* and set this derivative equal to zero.

$$\begin{aligned} &\frac{\partial Q(F)}{\partial F}(F = F^*) = \omega F^* - S' F^* + \mu(F^* - Y) = 0 \\ &IF^* - IF^* - \frac{(S' - \omega)}{1+\mu} F^* + \frac{\mu}{1+\mu} F^* = \frac{\mu}{1+\mu} Y \\ &\therefore F^* = \frac{\mu}{1+\mu}\left\{I - \frac{1}{1+\mu}(S' - \omega + I)\right\}^{-1} Y \end{aligned} \tag{9}$$

(9) must be solved iteratively just as was in the case of GRF. Because $\|S' - \omega + I\|$=1, the matrix inversion can be replaced with a series given that μ has a value larger than 0. We also derive an update equation similarly. In (10), $F(0)$=Y.

$$\begin{aligned} F(t) &= \left[\sum_{k=0}^{t}\left\{I - \frac{1}{1+\mu}(S' - \omega + I)\right\}^k\right] \frac{F(0)}{1+\mu} \\ F(t-1) &= \left[\sum_{k=0}^{t-1}\left\{I - \frac{1}{1+\mu}(S' - \omega + I)\right\}^k\right] \frac{F(0)}{1+\mu} \\ \therefore F(t) &= \left\{I - \frac{1}{1+\mu}(S' - \omega + I)\right\} F(t-1) + \frac{F(0)}{1+\mu} \end{aligned} \tag{10}$$

As in the closed-form solution of LS, Y_i(=$F_i(0)$) is initialized to a zero vector for i∈V_u. For i∈V_l, Y_i is initialized to all zeros except for Y_{ik}=1 if i has the k-th label. We can obtain a unique solution by updating $F(t)$ according to (10). The convergence proofs is also described in Section 4.

3) PGP with Vertex Weighting: PGP defines the transition between the states (vertices) by establishing the transition matrix P (Equation (14) in [4]) of a Markov chain. It also provides the label prediction procedure (Equation (15) in [4]) which requires the stationary distribution of the Markov chain. Since P utilizes the smoothness assumption of SSL, we are able to change the values of P such that the important vertices are more likely to propagate their label information, in order to apply the concept of vertex importance.

However, we cannot conduct both the calculation of stationary distribution and the measuring importance of vertices simultaneously. Unfortunately, there is no stable method for estimating the stationary distribution of a time-inhomogeneous Markov chain (Markov chain with varying P over time). Thus we present a 2-pass algorithm which consists of a step for evaluating the importance of vertices and changing P and a step for label inference with modified P.

The algorithm is applied as follows: For pass 1, the original PGP algorithm is run and the importance of vertices is calculated using the label distribution. For pass 2, using transition matrix P from pass 1, the new transition matrix is computed using $P_{ij} \leftarrow \omega_j P_{ij}$. The modified P should be normalized to satisfy $\sum_j P_{ij}$=1. After that we can infer a new label distribution. It has been proven that the Markov chain converges to a unique solution with a static transition matrix P. Therefore this extension of PGP also converges.

C. Determination of Weights of Vertices

We present four criterions for determining weights of vertices. The initial values of ω are initialized to 1. ω varies over time. The reasoning for these criterions is that the weights must be small if the label distribution F_i (normalized to sum to 1) is close to uniform, and vice versa. All criterions below produce values within [0, 1]. $0I$ represents a zero vector.

- MinMax Criterion (MM):

$$\omega_i = \begin{cases} 1 - \frac{\min_k(F_{ik})}{\max_k(F_{ik})} & \text{if } \max_k(F_{ik}) \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (11)$$

- Ambiguity Ratio (AR): Represents the ambiguity of choosing the label of vertex i

$$\omega_i = \begin{cases} 1 - \frac{\text{2nd}\max_k(F_{ik})}{\max_k(F_{ik})} & \text{if } \max_k(F_{ik}) \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (12)$$

- KLD Criterion: Exploits Kullback-Leibler Divergence

$$\omega_i = \begin{cases} 1 - \frac{\text{KLD}(F_i \| u)}{\forall v \in V \max_v(\text{KLD}(F_v \| u))} & \text{if } F_i \neq 0I \\ 1 & \text{otherwise} \end{cases} \quad (13)$$

- Variance Criterion (Var): Exploits the variance between F_i and uniform distribution

$$\omega_i = \begin{cases} 1 - \frac{\sum_k (F_{ik} - u_k)^2}{\forall v \in V \max_v(\sum_k (F_{vk} - u_k)^2)} & \text{if } F_i \neq 0I \\ 1 & \text{otherwise} \end{cases} \quad (14)$$

D. Non-smoothness Measure of Graph with Labeled Data

In Section 3.1, we examined how the vertex importance is used in label inference. However, the proposed approach may fail if the training data does not include samples which may harm the performance of the algorithms. Therefore a condition is required to establish the need for considering the weights of vertices. If possible, we can freely use the extended algorithms without concern for failure.

For this we define a threshold modifying the non-smoothness measure of GRF (i.e., $F^T L F$). Specifically, we calculate the non-smoothness of a graph which consists of labeled vertices and edges between a pair of labeled vertices only. Here we present a hypothesis for how this measure can be applied: If the non-smoothness of a labeled graph is considerably greater than some threshold value, then the graph constructed from the whole instances of training data is not likely to be sufficiently smooth to guarantee high quality predictions. So in this case we must consider the importance of vertices in order to achieve better results. A related issue is that of selecting the appropriate values for the threshold. In addition, it is necessary to scale this measure so that it may be applicable to a number of diverse graphs.

In conclusion, we evaluate the non-smoothness measure of a graph to determine whether we will consider the weights on vertices by applying the procedure below:

- Get W_l from W, such that W_l consists only of labeled vertices having edge(s).
- Evaluate the graph Laplacian L_l using W_l
- Let the non-smoothness measure ns_l = $tr(F_l^T L_l F_l)$ / (the number of edges in W_l $\times$ mean of W_l)

The threshold is empirically set to 0.15. Therefore, the extended algorithms are applied if ns_l>0.15. Otherwise, the base algorithms are instead applied.

IV. Convergence Proofs

We prove the convergence of the two algorithms (GRF, LS with vertex weighting) described in Section 3.2. For the PGP, we analyzed the convergences within the section.

A. GRF with Vertex Weighting

We must confirm $\|P_{UU}\|$<1, and show that the convergence also holds for varying T through iteration.

Theorem 1 $\|P_{UU}\|$<1.

Proof: P_{UU} is a square matrix consisting of positive real values such that the sum of each row is less than 1. Then the maximal absolute value (or the spectral radius) is less than 1 by the Perron-Frobenius theorem. Moreover, the spectral radius of P_{UU} is the same as its norm because P_{UU} is a normal (symmetric) matrix. Therefore $\|P_{UU}\|$<1. ■

Theorem 2 The algorithm converges if ω varies.

Proof: For V=P_{UU}, Y=$P_{UL}F_L$ and the V_t=V at iteration t, (5) is expanded in two terms as follows ($F(t)$=$F_U(t)$):

$$F(t) = V_1 \cdots V_{t-1} F(0) + Y \cdot \{I + V_t + V_{t-1}V_t + V_{t-2}V_{t-1}V_t + \cdots + (V_2 \cdots V_{t-1}V_t)\} \quad (15)$$

Because the norm of V is always less than 1, q=$\max\{\|V_1\|,\ldots,\|V_t\|\}$<1. For bounded linear operators A, B, and C=BA, $\|C\| \leq \|B\|\|A\|$ is satisfied. Then for the latter term of (15), $\|V_2 \cdots V_{t-1}V_t\| \leq \|V_t\|\|V_{t-1}\| \cdots \|V_2\| \leq q^{t-1}$. And $q^{t-1} \rightarrow 0$ for $t \rightarrow \infty$. Therefore $\|V_2 \cdots V_{t-1}V_t\| \rightarrow 0$ and every element of $V_2 \cdots V_{t-1}V_t$ converges to 0 for a considerably large value of t. Thus the coefficient of Y becomes a finite series and converges to some value. The coefficient of $F(0)$, $V_1(V_2 \cdots V_{t-1}V_t)$, becomes $V_1 \times 0$ and also converges to 0. Therefore (15) converges. ■

B. LS with Vertex Weighting

It is necessary to confirm the convergence of the Neumann series and to show the convergence of the update equation.

Theorem 3 $\|S^{'} - \omega + I\|$=1.

Proof: A transition matrix is defined to be P=$D^{-1}W$=$D^{-1/2}SD^{1/2}$. We can define T similar to

S'-ω+I, as T=ω(P-I)+I=$D^{-1/2}(S'$-ω+$I)D^{1/2}$. If two matrices are similar, then their eigenvalues are equal. Meanwhile, for any row i of P, the sum of this row except the diagonal element p_{ii} is (1-p_{ii}). Therefore for T=ω(P-I)+I, its diagonal element in the i-th row is $\omega_{ii}(p_{ii}$-1)+1 and the sum of the i-th row not including the diagonal element is ω_{ii}(1-p_{ii}). The sum of all elements of the i-th row is thus $\omega_{ii}(p_{ii}$-1)+1+ω_{ii}(1-p_{ii})=1. Therefore by the Perron-Frobenius theorem, the spectral radius of T is 1 and the spectral radius of S'-ω+I is also 1 because they are similar. S'-ω+I is a normal matrix therefore $\|S' - \omega + I\|$=1. ■

Theorem 4 The algorithm converges if ω varies.

Proof: For V=$\frac{1}{1+\mu}(S' - \omega + I)$, Y=$\frac{1}{1+\mu}F(0)$ and the V_t=V at iteration t, (10) is expanded in two terms as in (15). Meanwhile, for a real value α and a matrix A, $\|\alpha A\|$=$|\alpha|\|A\|$ is satisfied. Then for 1/(1+μ) within (0,1), the norm of V is always less than 1 according to Theorem 3 and the norm of V_i is always equal to 1/(1+μ) (After this, $\|V_i\|$=1/(1+μ)=q).

Therefore for the latter term of (15), $\|V_2 \cdots V_{t-1}V_t\| \leq \|V_t\|\|V_{t-1}\| \cdots \|V_2\| = q^{t-1}$ and $q^{t-1} \to 0$ for $t \to \infty$. Thus $\|V_2 \cdots V_{t-1}V_t\| \to 0$, the algorithm converges as shown in the latter half of the proof of Theorem 2. ■

V. Experiments

A. Data and Settings

We apply the benchmark datasets provided from [1] for the experiments. The sources and statistics are described in detail, in Section 21 of [1]. Except BCI, each dataset consists of 12 subsets with 100 labeled and 1400 unlabeled instances (for BCI, 100 labeled and 300 unlabeled). For each dataset we evaluate the average classification error of the subsets.

In addition, we include one face, one voice and one object dataset, Extended Yale B, Isolet, COIL(COIL20, COIL100), in the experiments. Each dataset has its own source but we adopt the datasets created by [8]. The sources and statistics are also described in [8]. In order to use them in the experiments, we process these data as follows:

- Extended Yale B: We select a 32×32 pixel data file, and use first 10 subsets of 50 random splits. The number of classes is 38. There are 64 images per class. For experimentation, we take 10 or 30 images per class and regard them as labeled data and the rest as unlabeled.
- Isolet: First we take all five subsets, named Isolet1-5, and randomly select 10% or 50% of the samples in each class and treat them as labeled samples.
- COIL: For two two datasets, COIL20 and COIL100, we randomly select 10% or 50% of the images in each class and treat them as labeled images. On each percentage of labeled images, we produce 10 subsets.

The graphs for the benchmark datasets of [1] are constructed asymmetrically (add edges between vertices i and j where either i is a nearest neighbor of j) as 5/10/20-NNs (g241c, g241n: 20-NN, TEXT: 10-NN, and the rest: 5-NN) with a squared kernel width σ^2 which is equal to the average squared distance of the nearest neighbors. For the three Matlab datasets, first we normalize each sample so that it has a unit norm. Then we construct asymmetric 4/20-NNs (only 20-NN for Isolet) with kernel width σ equal to 1.

TABLE I
Experimental results of benchmark datasets (average classification errors)

	Digit1	USPS	COIL	COIL2	BCI	TEXT	g241c	g241n
1-NN	0.0757	0.0787	0.2299	0.1250	0.4522	0.5028	0.3949	0.3728
MP	0.0214	0.1209	0.1555	0.0450	0.4828	0.3727	0.4803	0.4499
AGR	0.0467	0.1405	0.2199	0.0989	0.4739	0.3226	0.3510	0.3267
GRF	0.0207	0.0749	0.1527	0.0412	0.4728	0.3060	0.4352	0.4018
GRF (MM)	0.0207	0.0753	**0.1526**	**0.0408**	**0.4725**	**0.2968**	**0.4057**	**0.3691**
GRF (AR)	0.0207	0.0753	**0.1510**	**0.0408**	**0.4725**	**0.2968**	**0.4057**	**0.3691**
GRF (KLD)	0.0207	0.0753	**0.1515**	**0.0408**	**0.4683**	0.3319	**0.3985**	**0.3826**
GRF (Var)	0.0207	0.0753	**0.1511**	**0.0408**	**0.4692**	0.3265	**0.4000**	**0.3821**
LS	0.0279	0.0613	0.1593	0.0510	0.4806	0.2901	0.4188	0.3716
LS (MM)	0.0279	**0.0605**	0.1593	0.0510	**0.4769**	**0.2762**	**0.3711**	**0.3345**
LS (AR)	0.0279	**0.0605**	**0.1577**	0.0510	**0.4769**	**0.2762**	**0.3711**	**0.3345**
LS (KLD)	0.0279	**0.0607**	**0.1585**	0.0510	**0.4644**	**0.2839**	**0.3511**	**0.3152**
LS (Var)	0.0279	**0.0605**	**0.1576**	0.0510	**0.4669**	**0.2832**	**0.3510**	**0.3144**
PGP	0.0322	0.0602	0.1692	0.0548	0.4653	0.2818	0.3828	0.3465
PGP (MM)	0.0324	**0.0565**	**0.1686**	0.0548	**0.4633**	0.2839	**0.3615**	**0.3252**
PGP (AR)	0.0322	**0.0575**	0.1694	0.0549	**0.4636**	0.2835	**0.3629**	**0.3266**
PGP (KLD)	0.0323	**0.0565**	**0.1686**	0.0549	**0.4600**	0.2954	**0.3758**	**0.3408**
PGP (Var)	0.0326	**0.0570**	**0.1685**	0.0548	**0.4639**	0.2938	**0.3762**	**0.3382**

Algorithms and parameters for comparison are as follows:

- 1-NN: A 1-Nearest Neighbor classifier
- Measure Propagation (MP) [9]: A probabilistic graph-based SSL algorithm (μ=0.05. ν=0.01, α=1)
- AnchorGraph Regularization (AGR) [10]: A scalable and fast SSL algorithm (m=10% of n, s=3)
- GRF and its extensions: No parameter
- LS and its extensions: 1/(1+μ)= 0.9
- PGP and its extensions: α=0.2

The maximum number of iterations of the algorithms is set to 1000, but we further set an additional stopping criterion $\|\text{vec}(F(t)\text{-}F(t\text{-}1))\|/\|\text{vec}(F(t\text{-}1))\| < 10^{-6}$, where 'vec' denotes the vectorization operator. The left-hand side of the inequality is the relative residual of F. If this is met, the algorithms stop. In order to avoid reducing the performance of the algorithms, we only apply the extension methods which include the importance of vertices if ns_l>0.15 (Section 3.4).

B. Results and Analysis

In Table I and II, the bold names of the algorithms are the proposed extension methods, and the bold numbers represent that the proposed vertex weighting approaches which enhance classification performance of the base algorithms for GRF, LS, and PGP. The best results are underlined.

In Table I, the proposed methods perform better than the base algorithms when conducted on datasets such as BCI, TEXT, g241c and g241n. As explained in Section 3.1, the results indicate that the adjustment of the influences of vertices is justifies the application of the extension algorithms in label inference. For Digit1, USPS, COIL and COIL2 the performance of the proposed methods are relatively poor when ns_l=0, but such degradation in performance can be prevented by setting ns_l>0.15. There is some degradation in performance on TEXT (for GRF and PGP) but it is also avoidable by applying the MinMax or the Ambiguity Ratio criterion. The benchmark datasets have a small number of classes (6 for

TABLE II
EXPERIMENTAL RESULTS OF ADDITIONAL DATASETS (AVERAGE CLASSIFICATION ERRORS)

	YaleB (10)	YaleB (30)	Isolet (10%)	Isolet (50%)	COIL20 (10%)	COIL20 (50%)	COIL100 (10%)	COIL100 (50%)
1-NN	0.4460	0.2352	0.3363	0.2412	0.1290	0.0261	0.2335	0.0597
MP	0.2961	0.2035	0.2568	0.1704	0.0729	0.0385	0.1546	0.1023
AGR	0.7285	0.9279	0.3541	0.5218	0.1289	0.6067	0.2453	0.9580
GRF	0.2751	0.1759	0.2513	0.1514	0.0684	0.0238	0.1483	0.0807
GRF (MM)	**0.2750**	0.1760	0.2514	0.1514	0.0686	0.0238	0.1706	0.0808
GRF (AR)	**0.2733**	**0.1758**	0.2574	**0.1493**	0.0686	0.0238	0.1706	0.0808
GRF (KLD)	0.2752	0.1785	**0.2488**	**0.1488**	0.0686	0.0238	0.1704	0.0808
GRF (Var)	0.2772	0.1795	**0.2500**	**0.1496**	0.0686	0.0238	0.1703	0.0808
LS	0.2909	0.2014	0.2602	0.1922	0.0729	0.0385	0.1551	0.1023
LS (MM)	**0.2908**	0.2016	0.2602	0.1922	0.0729	0.0385	0.1551	0.1023
LS (AR)	**0.2897**	**0.1914**	**0.2590**	**0.1742**	0.0729	0.0385	**0.1542**	0.1023
LS (KLD)	0.2941	**0.1980**	**0.2585**	**0.1770**	0.0729	0.0385	**0.1550**	0.1023
LS (Var)	0.2956	**0.1933**	**0.2577**	**0.1652**	0.0729	0.0385	**0.1545**	0.1023
PGP	0.2949	0.1824	0.2498	0.1506	0.0754	0.0242	0.1588	0.0811
PGP (MM)	0.2956	0.1827	**0.2481**	**0.1496**	**0.0753**	**0.0240**	**0.1586**	**0.0809**
PGP (AR)	0.3022	0.1929	0.2533	**0.1473**	0.0754	0.0242	0.1588	**0.0809**
PGP (KLD)	0.3000	0.1887	0.2523	**0.1450**	0.0755	0.0242	0.1588	**0.0808**
PGP (Var)	0.3068	0.1976	0.2614	**0.1434**	0.0756	0.0242	0.1588	**0.0809**

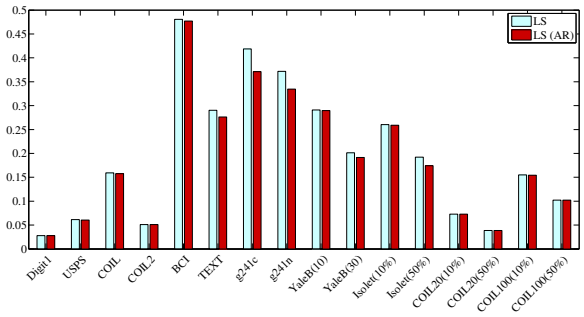

Fig. 2. Performance Comparison between LS and LS (AR) (Average Errors)

COIL and 2 for the rest), therefore we recommend using the MinMax or the AR criterion for stable results.

In Table II we can also see that the proposed approaches safely enhance performance, particularly for LS. The approaches are especially effective on the Isolet dataset. In addition, we can avoid the degradation in performance of GRF on COIL100 (10%) and PGP on YaleB (10%) by setting a larger threshold value of ns_l.

We recommend using the extended algorithm of LS with the MinMax or the AR criterion, as the more stable results are able to successfully avoid the decrease in performance. These criterion in conjunction with the extended LS algorithm are most effective on TEXT, g241c, g241n and COIL100. We can see this clearly in Figure 2, which is the visualized result of LS and LS (AR). We presume that the success of this algorithm comes from the changes in the values of the label functions, whereas the other algorithms change *W*.

VI. CONCLUSION

In this paper we propose a new approach to enhance existing label inference algorithms of graph-based SSL by incorporating the importance of vertices. We also propose three extensions of existing algorithms GRF, LS, and PGP. To ensure the quality of results before learning, we introduce a unique measure (ns_l) for determining whether to apply the proposed method on the data. The experimental results for 8 benchmark datasets and various additional datasets have demonstrated that our proposed approach enhances the base algorithms safely for an appropriate threshold value of the measure ns_l. There still remain several directions for future work: seeking other approaches of vertex weighting, conducting experiments on other datasets, and presenting additional extensions of other label inference algorithms.

ACKNOWLEDGMENT

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2012M3C4A7033348) and by the ICT R&D program of MSIP/IITP (R0126-16-1112, Development of Media Application Framework based on Multi-modality which enables Personal Media Reconstruction).

REFERENCES

[1] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 2006.

[2] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

[3] A. Subramanya and P. P. Talukdar, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, 2014.

[4] Y. Fang, K. C.-C. Chang, and H. W. Lauw, "Graph-based semi-supervised learning: Realizing pointwise smoothness probabilistically," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 406–414.

[5] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 912–919.

[6] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in Neural Information Processing Systems*, vol. 16, no. 16, pp. 321–328, 2004.

[7] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," *Semi-Supervised Learning*, vol. 10, 2006.

[8] D. Cai, "Matlab codes and datasets for feature learning," http://www.cad.zju.edu.cn/home/dengcai/Data/data.html, 2011.

[9] A. Subramanya and J. Bilmes, "Semi-supervised learning with measure propagation," *The Journal of Machine Learning Research*, vol. 12, pp. 3311–3370, 2011.

[10] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 663–670.