Learning from Multiple Graphs using a Sigmoid Kernel

Thomas Ricatte, Gemma C. Garriga, Rémi Gilleron, Marc Tommasi

Learning from Multiple Graphs using a Sigmoid Kernel

Thomas Ricatte
SAP Research

Gemma Garriga
INRIA Lille

Rémi Gilleron
Lille 3 University and INRIA Lille

Marc Tommasi
Lille 3 University and INRIA Lille

***Abstract*—This paper studies the problem of learning from a set of input graphs, each of them representing a different relation over the same set of nodes. Our goal is to merge those input graphs by embedding them into an Euclidean space related to the commute time distance in the original graphs. This is done with the help of a small number of labeled nodes. Our algorithm output a combined kernel that can be used for different graph learning tasks. We consider two combination methods: the (classical) linear combination and the sigmoid combination. We compare the combination methods on node classification tasks using different semi-supervised graph learning algorithms. We note that the sigmoid combination method exhibits very positive results.**

I. Introduction

In many real-life problems, data can be represented as a graph. Graphs can be natural networks, such as the Web graph or social networks, but also any set of data instances can be easily transformed into a graph, where each node represents a data point and edges correspond to relationships of similarity among the points.

One difficulty arises when data exhibits heterogeneity of the features or of the relations. For example, this is the case of datasets with both categorical and numerical attributes describing each data instance; while it can be hard to design a similarity measure that takes into account all attributes, it becomes far easier to define specialized similarity measures for small subsets of attributes. For example, we can build similarity measures for categorical attributes based on the number of shared features (see [19]). Many similarity measures can be derived as well for numerical attributes (euclidean distances, ...) This idea leads naturally to complex graphs embedded with multiple layers of information. Likewise, natural networks exhibit as well heterogeneity in the form of different type of links between the entities; for example, a social network might have links of "friendship" or "family" or "work" between the nodes. Again, this heterogeneity leads to graphs that can be expressed in multiple topologies or layers: nodes are common across topologies, however edges might carry different information at each layer.

The problem addressed in this paper can be broadly stated as follows: Given two (or more) input undirected graphs $G_1 = (N, E_1)$ and $G_2 = (N, E_2)$, representing each a topology or type of relation (E_1 or E_2) for the same shared set of nodes (N), combine them in order to perform a learning task, such as spectral clustering [11] or semi-supervised classification [3], [16], [22]. To allow for such combination, we only suppose given a partial supervision via a given function $y : S \subset N \to \{+1, -1\}$ assigning labels (classes or groups) to a subset of nodes S in the input graphs G_1 and G_2. This partial supervision indicates that when $y(i) = y(j)$ and $y(i) \neq y(k)$ for nodes $i, j, k \in N$, then i and j should be topologically "closer" in the final graph than i and k. This is a challenging merging task since one graph could be topologically better than the other, globally or only for some specific subset of nodes. In this case, we would like to combine the two graphs such that learning with the combination yields the same results than learning on the best graph. It could also be the case that graphs are complementary. Then, we would like that learning with the combination outperforms learning results on the two graphs.

Our proposal is to exploit the topology of a set of input graphs via graph Laplacian operators [3], [4] on graphs, which are closely related to the notion of commute time distance (specific geodesic distance on the graph). The strategy we follow is to embed every graph into an Euclidean latent space in which the Euclidean distance between nodes is directly related to the commute time distance. We consider then the product space of these original latent space and look for the best Euclidean structure (scalar product). We use two different objective functions to search for the best combination: the first one is an energy-based function that leverage the results of [2] and the second is based on the idea that the resulting structure is good if it allows the small subset S to be efficiently labeled in the product space. In both cases, the objective function only depends on the small amount of labeled points.

The optimal Euclidean space will finally allow us to define a new Laplacian-like operator. In the general case, we are not be able to interpret this operator as the Laplacian of a graph. However, we consider some specific constraints in our optimization process to preserve some important regularization properties. We consider two combination methods:

- The *linear* method where the final scalar product is a linear convex combination of the original scalar products
- The *sigmoid* combination where the final scalar product is based on a sigmoid function applied on top of the linear combination

We compare the combination methods on both similarity graphs constructed from vectorial data and network data. For

This work was partially supported by ANR Project Lampada (ANR-09-EMER-007).

the comparison, we use the classification accuracy on the combination result of two different semi-supervised graph learning algorithms ([22] and [18]) and also we compute the smoothness of the combination. Experimental results show very encouraging results for the sigmoid combination. Indeed, the sigmoid combination allows in some cases to outperform the best of the two graphs.

Important previous literature has studied multiple-view learning on graphs [2], [17], [8] or proposed solutions for combining graphs [15], [20], [21] for specific learning tasks. So far, several approaches have been proposed for learning from multiple graphs. The work in [15] proposes a co-training algorithm to learn from a set of input graphs. Other works like [20], [21] employ a Markov random walk method for the tasks of ranking or document recommendation. Closer to our approach is the work in [2]. They combine graph Laplacians under an objective function based on semi-supervised classification. Their approach is part of the multiple kernel learning approach, see e.g. [7]. Learning from multiple views in graphs is also related to our contributions. The work in [17] studies different objective functions where the essential ingredient is to form a mixture of Markov chains defined on the different views for the specific learning task. More recently, the contribution in [8] studies a solution based on graphs for problems with multiple heterogeneity (features and task heterogeneity); the method is based on the construction of a bipartite graph within each view and task and using an iterative algorithm for optimizing a final classification task.

Compared to these approaches our contributions are:
(i) We propose to combine multiple graphs independently of the learning task at hand using only a small set of labeled nodes.
(ii) The result of our combination method allows to define a smoothness measure which can be used by many graph learning algorithms.
(iii) Our experiments with the sigmoid kernel-based combination show very positive results: significant improvement wrt the original input graphs in semi-supervised learning and smoothness of the combination.

II. Undirected Graphs and Laplacian

A simple undirected weighted graph $\mathcal{G}$ is a pair (N, E) where $N = (u_1, \ldots, u_n)$ is a set of nodes and $E \subset N \times N$ is a set of edges. Edges are weighted by the symmetric positive function $w : E \to \mathbb{R}^+$. Simple graphs have no self loop, that is $\forall u, (u, u) \notin E$. We extend w to $N \times N$ by setting the weight value to 0 when no edge is present. The adjacency matrix denoted by W is defined by $W_{i,j} = w(u_i, u_j)$. The matrix W is symmetric and all coefficients are non negative.

The spectral framework introduces a regularization model to estimate the smoothness of a real valued node function $f \in \mathbb{R}^n$. First, given a real valued node function f, let us consider the cost function on the edges called *unnormalized gradient* defined by

$$grad(f)(u, v) = \sqrt{w(u, v)}(f(v) - f(u))$$

The unnormalized gradient satisfies the *constant gauge* property: any constant node function has an unnormalized gradient equal to the zero edge function. Let us note that the unnormalized gradient is defined up to an arbitrary orientation of the edges. It is a linear application that can be represented by a $m \times n$ real valued matrix G where m is the number of edges in the graph.

The unnormalized gradient allows to define the global smoothness $\Omega(f)$ (regularization) of a node real valued function f by $\Omega(f) = \|Gf\|^2 = (Gf)^T(Gf) = f^T G^T G f$. The quantity $\Delta = G^T G$ is called *unnormalized graph Laplacian* of the graph. The term $f^T \Delta f$ is widely used as a regularization term to enforce the smoothness of a real valued node function f (for instance a node labeling function). We recall that the unnormalized Laplacian Δ of a graph has been shown to be equivalently defined by $\Delta = D - W$ where W is the adjacency matrix of the graph and D is the diagonal degree matrix: $D_{u,u} = \sum_v w(u, v)$. The matrix Δ does not depend on the orientation of the edges and has two essential properties:

$(\mathcal{P}_1)$ Δ is symmetric positive semi-definite
$(\mathcal{P}_2)$ $\Delta \mathbf{1} = 0$ where $\mathbf{1} = (11 \ldots 1)^T$ (constant gauge).

If a matrix Δ (not necessarily a graph Laplacian) satisfies the two properties $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ then the function $f \to f^T \Delta f$ is a pseudo-norm operator measuring the smoothness of real valued node functions. Such a matrix Δ will be called a *smoothness operator*. We can use a smoothness operator to define the smoothness $\mathcal{Q}(\Delta, f)$ of a node function f by

$$\mathcal{Q}(\Delta, f) = \frac{f^T \Delta f}{2 \sum_i \Delta_{i,i}}$$

The smoothness belongs to $[0, 1]$. When Δ is the unnormalized Laplacian of a graph $\mathcal{G}$, the smoothness $\mathcal{Q}(\Delta, f)$ measures the compatibility of a node function f w.r.t. the graph $\mathcal{G}$.

An important property of the unnormalized Laplacian is its relation with random walks in graphs. The *hitting time* $m(i|j)$ is the expected time taken by a random walker to travel from node i to j. Notice that $m(\cdot|\cdot)$ is not symmetric. Therefore, the commute time distance c_{ij} between $i, j \in N$ is defined by $c_{ij} = m(i|j) + m(j|i)$. The commute time distance captures the global topology of the graph. A result from spectral graph analysis establishes a link between the commute time distance and the unnormalized Laplacian.

Proposition 1: [5], [9] Let $\mathcal{G}$ be a positively weighted connected graph and let $\Delta^\dagger$ be the Moore-Penrose inverse of the unnormalized Laplacian of $\mathcal{G}$. Then

$$c_{i,j} = vol(\mathcal{G}) \left(\Delta^\dagger_{i,i} + \Delta^\dagger_{j,j} - 2\Delta^\dagger_{i,j} \right),$$

where $vol(\mathcal{G}) = \sum_{i \in N} D_{i,i}$ is the volume of the graph.

Let us note that the pseudo inverse of a positive semi definite matrix is positive semi definite. Thus the pseudo inverse of a graph Laplacian is positive semi definite and it is a kernel function. We call *graph kernel* the pseudo inverse of a graph Laplacian.

III. Combining Graphs through Euclidean Embedded Spaces

We consider now the problem of combining different graphs on the same set of nodes. We assume that the input graphs are connected, which allow us to make a link between commute-time distance and Laplacian matrices (see Proposition 1). We split the combination process in two parts: first, we use the Laplacian matrix to associate graphs with what we call *embedded Euclidean spaces*; second, we merge these spaces into a new Euclidean space associated to a new kernel (Gram matrix) which we ensure to be a smoothness operator.

A. Euclidean Embedded Spaces

Given a graph $\mathcal{G} = (N, E)$, an *embedded Euclidean space* is a tuple $(\mathcal{E}, K(\cdot,\cdot), \phi)$ where $(\mathcal{E}, K(\cdot,\cdot))$ is an Euclidean space and ϕ is an injective function, called *node map*, from N to $\mathcal{E}$. The Gram matrix $K = (K(\phi(i), \phi(j)))_{i,j\in N}$ is the *kernel* of the space. An embedded Euclidean space satisfies *the commute-time property* for $\mathcal{G}$ if, for every pair of nodes $i, j \in N$, we have $K(\phi(i), \phi(j)) = \Delta^{\dagger}_{i,j}$. In this case, we have $\|\phi(i)-\phi(j)\|^2 = c_{i,j}/vol(\mathcal{G})$, where $\|.\|$ is the Euclidean norm associated with $K(\cdot,\cdot)$, and $c_{i,j}$ is the commute-time distance between nodes i and j.

We can use the singular value decomposition of the unnormalized Laplacian to compute an embedded Euclidean space that satisfies this property. Let us write $\Delta = V\Lambda V^T$, where Λ is the diagonal matrix of the eigenvalues $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_N$ associated with the eigenvectors (columns of V). We can thus consider the space

$$V(\mathcal{G}) = \left(\mathbb{R}^{|N|},\ K(x, y) = x^T \Lambda^{\dagger} y,\ \phi : i \to e_i^T V\right),$$

then $K(\phi(i), \phi(j)) = e_i^T V \Lambda^{\dagger} V^T e_j = e_i^T \Delta^{\dagger} e_j = \Delta^{\dagger}_{i,j}$.

While we do not consider it in the paper, a dimensionality reduction of the embedded spaces can be done while preserving *approximately* the commute-time property. Using the SVD embedded space, one can use the so-called PCA approach as proposed in [6].

Let us now consider the k input graphs $\mathcal{G}_1, \ldots, \mathcal{G}_k$ on the same node set N. For each graph $\mathcal{G}_i$, let $V_i = (\mathcal{E}_i, K(\cdot,\cdot)_i, \phi_i)$ be an associated embedded space. Let K be a symmetric bilinear form on $\mathcal{E} = \mathcal{E}_1 \times \cdots \times \mathcal{E}_k$. We define the *$K$-merged space* by $(\mathcal{E}, K, \phi : i \to (\phi_1(i), \ldots, \phi_k(i)))$. K allows us to combine the different embedded spaces and it can be viewed as a combination of the original the inner products (kernels) of the original embedded spaces and is denoted by $K = F(K_1, \ldots, K_k)$. It should be noted that, when the original spaces satisfy the commute-time property (no dimension reduction), the kernels K_i are equals to the original graph kernels $\Delta_i^{\dagger}$ and we do not need to compute the full embedding.

B. Convex Linear Combination

A simple choice for the combination function is to build a linear convex combination of the input kernels:

$$K_{\mathrm{LIN}} = F_{\mathrm{LIN}}(K_1, \ldots, K_k) = \sum_{\ell=1}^{k} w_\ell K_\ell, \tag{1}$$

with $\forall 1 \leq \ell \leq k, w_\ell \geq 0$, and $\sum_{\ell=1}^{k} w_\ell = 1$. In this case, the Euclidean distance in the merged space defined by K_{LIN} is the weighted sum of the commute-time distances in the original graphs (recall that the original embedded spaces satisfy the commute-time property).

The Moore-Penrose pseudo inversion preserves the properties $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ (see [14]) of the smoothness operators. It is also the case for the convex linear combination. Thus we can consider the matrix $\Delta_{\mathrm{LIN}} = K_{\mathrm{LIN}}^{\dagger}$ as the new smoothness operator.

As mentioned above, we consider a partial supervision to drive our combination process and find the best set of values for $w_1, \ldots, w_k$. We assume without loss of generality that the first n_L nodes of N are labeled. The associated label vector is denoted by y_L. Argyriou *et al* describe in [2] a framework for the convex linear combination based on the minimization of the following objective function:

$$E_\gamma(\Delta_{\mathrm{LIN}}) = \min_f \left\{ f^T \Delta_{\mathrm{LIN}} f + \gamma \mathcal{L}(f, y_L) \right\} \tag{2}$$

where $\mathcal{L}(f, y_L)$ is a loss term. Thus, the minimization problem is a semi-supervised problem defined as a trade-off between a loss term $\mathcal{L}(f, y_L)$ and a smoothness regularization term $f^T \Delta_{\mathrm{LIN}} f$. $E_\gamma(\Delta_{\mathrm{LIN}})$ is the minimal energy that can be obtained with the smoothness operator Δ_{LIN}. Argyriou *et al.* leverage the RKHS framework to simplify the computation of E_γ: with reasonable assumptions on the loss function $\mathcal{L}$, we can apply the *representer theorem* to state that the optimal point f of (2) can be expressed as:

$$f = \sum_{i=1}^{n_L} c_i (\Delta^{\dagger}_{\mathrm{LIN}})_i$$

where $(\Delta^{\dagger}_{\mathrm{LIN}})_i$ is the i-th column of $\Delta^{\dagger}_{\mathrm{LIN}}$. Thus, the energy can be computed using the simpler dual formulation:

$$E_\gamma = -\min_c \left\{ \frac{1}{4\gamma} c^T (\Delta^{\dagger}_{\mathrm{LIN}})_L c + V^*(c, y_L) \right\} \tag{3}$$

where $\mathcal{L}^* = \sup_{\lambda\in\mathbb{R}}(\lambda c - \mathcal{L}(y, \lambda))$ and $(\Delta^{\dagger}_{\mathrm{LIN}})_L = (\Delta^{\dagger}_{\mathrm{LIN}})_{1\ldots n_L, 1\ldots n_L}$ is the associated kernel. This dual optimization problem only involves a $n_L \times n_L$ matrix with $n_L \ll n$.

Leveraging former results from [1], they finally propose an algorithm to reduce the search of the minimum of $E_\gamma(\Delta_{\mathrm{LIN}})$ to a sequence of line-search steps. A major drawback of this method is that we have to choose carefully the parameter γ (trade-off between regularization and loss) in order to get a meaningful energy E_γ.

As an alternative of this method, we consider a non-parametrical objective function $\mathrm{cv\_svm}(K_L, y_L)$ defined as the cross-validation error on the set of labeled example of an SVM classifier trained with kernel K_L. It should be noted that the framework of [2] that allow us to replace the exhaustive search by a sequence of line search do not apply to our objective function. Thus we will consider an exhaustive search of the parameters of the linear combination.

C. Sigmoid Combination

We also consider the sigmoid combination of kernels defined by:

$$K_{\mathrm{SIG}}(i,j) = 1/(1+\exp(-(K_{\mathrm{LIN}}(i,j))/\sigma)) \qquad (4)$$

Luxburg *et al* states in the part 4 of [12] that the sigmoid can be used to improve the quality of the commute time distance (as a topological indicator) for highly connected graphs because of its "normalization" effect. Indeed, the main idea is to normalize kernel values in $[0,1]$. This idea was leveraged in [10] in order to design an efficient kernel clustering algorithm on a single graph.

Let us suppose that the w_l are fixed and let us denote by K_{SIG} the Gram matrix of the combined kernel. The situation is more intricate because the matrix $K_{\mathrm{SIG}}^{\dagger}$ may not be positive semi-definite and may not satisfy $K_{\mathrm{SIG}}^{\dagger}\mathbf{1}=0$. Thus it is not in general a smoothness operator. Thus, the solution is to define a "proxy" matrix in the space of smoothness operators. It must be close to $K_{\mathrm{SIG}}^{\dagger}$, positive semi-definite and it must satisfy the constant gauge property. To do this given as input the matrix $K_{\mathrm{SIG}}^{\dagger}$, we propose the *flip method* defined by:

1) Modify the diagonal terms of $K_{\mathrm{SIG}}^{\dagger}$ in order to have null sum for the rows and columns (property $(\mathcal{P}_2)$)
2) Compute the eigenvalues of the resulting matrix, and flip the negative eigenvalues (replace λ by $-\lambda$) to get a positive semi-definite matrix $K_{\mathrm{SIG}}^{\dagger}$.

And we consider the result of the flip method, denoted by Δ_{SIG}, to be the smoothness operator output for the sigmoid combination.

It should be noted that when the values $K_{\mathrm{LIN}}(i,j))/\sigma$ are small, then matrix $K_{\mathrm{SIG}}^{\dagger}\mathbf{1}$ is close to 0 (sum of columns/rows). Indeed, the power series expansion of the sigmoid function, that is, $\frac{1}{1+\exp(-x)} \sim 1/2 + x/4$, allows us to show that the sums of the rows of K_{SIG} should be close to $N/2$; therefore, from [14], the sums of the rows of $K_{\mathrm{SIG}}^{\dagger}$ should be close to $2/N$. Also, it should be noted that we do not use the usual shift trick which shifts the spectrum by minimal eigenvalue $\lambda_{\min}$: $\Delta \leftarrow \Delta + \lambda_{\min} I$ because the flip preserves the property $(\mathcal{P}_2)$.

We tune the parameters $\sigma, w_1, \ldots, w_k$ similarly to the case of the convex linear combination. We consider the objective functions $E_\gamma(\Delta_{\mathrm{SIG}})$ and $\mathrm{cv\_svm}$ and search for the best parameter values.

D. Combination Algorithm

We summarize our combination method in Algorithm 1. It should be noted that in the experiments we do not consider the dimensionality reduction mentioned in line 3. The parameter tuning methods of line 6 have been described above. It should be noted that in line 7, for the two types of combination, the output of our algorithm is a smoothness operator Δ_{LIN} or Δ_{SIG} allowing to compute the smoothness of any real valued node function f with the regularization term $f^T\Delta_{\mathrm{LIN}}f$ or $f^T\Delta_{\mathrm{SIG}}f$.

Algorithm 1 Combining embedded spaces for graphs.

```
Input: graphs $\mathcal{G}_1,\ldots,\mathcal{G}_k$ on a node set $N$; labeled sample $S \subset N$;
1: for each graph $\mathcal{G}_i$ do
2:     Compute an embedded space $V_i = (\mathcal{E}_i, K_i, \phi_i)$
3:     [Opt.] reduce the dimensionality: $K_i \leftarrow K'_i$
4: end for
5: Merge space $V = (\mathbb{R}^{|S|}, K, \phi)$ where $K = F(K_1,\ldots,K_n)$ ($F = F_{\mathrm{LIN}}$ or $F_{\mathrm{SIG}}$)
6: Tune the parameters of the combination method using one of the objective functions $\{E_\gamma, \mathrm{cv\_svm}\}$
7: return A smoothness measure $\Delta$ computed from $K$ (pseudo-inversion + additional flip trick for the sigmoid)
```

IV. Experiments

We now present some experimental results based on these two methods. We first describe the datasets and the graph construction process we consider in the experiments. Then, we detail the implementation of Algorithm 1 and the experimental setting. Last, we apply graph-based semi-supervised learning algorithms and compare results obtained using the Laplacian of every graph, using the smoothness operators Δ_{LIN} and Δ_{SIG}.

A. Datasets

We consider here similarity graphs defined from vectorial data or categorical data and graphs given by network data. For every dataset, we identify or build two graphs G_1 and G_2 that are given as input of Algorithm 1.

The datasets used are summarized in Table I. For the similarity graphs, we choose UCI datasets with both numerical and categorical attributes. For every UCI dataset, we build the graph G_1 as the *k-nearest neighbor graph* (k-nn) based on the Euclidean distance on normalized numerical attributes: every node n in the graph represents a data point; two nodes i and j are connected if j is among the k-nearest neighbors of i. As the k-nn relation might not be symmetric, a common choice to make the final graph undirected is by ignoring directions (if $\hat{W}$ is the non symmetric adjacency matrix built from the k-nn algorithm, we consider the symmetric adjacency matrix $W = (\max(\hat{W}_{i,j}, \hat{W^T}_{i,j}))_{i,j}$). We also build the graph G_2 using the categorical attributes. For every node i and j, we let $W_{ij} = \sum_{a\in\mathcal{A}} \frac{\delta_{a(i),a(j)}}{\mathcal{N}_a(a(i))}$, where $\mathcal{A}$ is the set of the categorical attributes; $a(i)$ is the value of the categorical attribute $a \in \mathcal{A}$ for the node i; $\delta_{x,y}$ is the Kroenecker delta; $\mathcal{N}_a(x)$ is the number of times attribute a has value x in the whole set (common values bring less information). This measure based on the work of [19] defines the weight adjacency matrix of the graph G_2.

We also consider the network datasets WebKB and IMDB (prodco). The IMDB dataset contains two graphs G_1 and

G_2 over a set of movies. For the WebKB dataset, the two graphs G_1 and G_2 are defined over a set of Web pages using respectively the hyperlinks and the co-citation links. See http://netkit-srl.sourceforge.net/data.html for more details.

For the similarity graphs with numerical attributes to be connected, we choose k such that the graph has only one connected component. For the similarity graphs with categorical attributes and for the networks to be connected, we adopt the teleporting random walk approach of [13]: we consider if needed a small jumping probability (0.05) and adapt the adjacency matrices accordingly.

For every graph, we compute the graph Laplacian. Graph kernels are normalized by their Frobenius norm as proposed in [2].

Dataset	**Class attribute**	**Size**
Statlog heart	Presence of disease	270
Credit approval	Approval	690
Horse colic	Surgical lesion	368
Flags	Religion Catholic, Oth. Christians vs others	194
Adult (excerpt)	Income $> 50K$	1400
WebKB	Page type: Student vs others	1477
IMDB	Blockbuster	1441

TABLE I
UCI AND NETWORK DATASETS, SIZE AND CLASS LABEL.

B. *Experimental setting*

Because we want to show that our combination method is independent of the learning task, in order to evaluate our combination method, we consider different performance indicators:

- The efficiency of two popular semi supervised learning (SSL) algorithms described respectively in [22] (*ZGL*) and [18] (*ZHS*) for the semi-supervised problem described in Eq. (2)
- The compatibility measure between Δ and the real label function y, $\mathcal{Q}(\Delta, y)$.

In each experiment and for each pair of graphs (G_1, G_2) we repeat 12 times the following protocol:

(i) Randomly sample unlabeled data preserving the class proportions

(ii) For each objective function $\mathrm{cv\_svm}$ and E_γ apply Algorithm 1 and output two smoothness operators Δ_{LIN} and Δ_{SIG} (linear and sigmoid combination). We denote by $\Delta_{\mathrm{LIN(E_\gamma)}}$, $\Delta_{\mathrm{SIG(E_\gamma)}}$, $\Delta_{\mathrm{LIN(cv\_svm)}}$, $\Delta_{\mathrm{SIG(cv\_svm)}}$ the outcomes of Algorithm 1 depending on the combination operator and on the objective function. We use a basic implementation of the algorithm that performs an exhaustive search over the parameters. At the same time and for each performance indicator, we compute the best possible linear and sigmoid combinations evaluated with fully labeled graphs. This step allows us to add some optimal baselines $\Delta_{\mathrm{LIN(opt)}}$, $\Delta_{\mathrm{SIG(opt)}}$.

(iii) Run the different performance indicators *ZGL*, *ZHS* and quality on each input graph G_1 and G_2 and on the combined objects.

The value γ of E_γ has been manually tuned to 10. $\Delta_{\mathrm{LIN(E_\gamma)}}$ is equivalent to the algorithm proposed in [2] (for two input graphs, it reduces to a classic line search over the values of E_γ).

C. *Experimental results*

We first consider a fixed ratio of labeled examples chosen to be of 30%, thus 70% of nodes are unlabeled. Experimental results are shown in Tables II and III for *ZGL* and quality. Results for *ZHS* are very similar to the results for *ZGL* and therefore we do not report them. We will discuss and compare a combination method (sigmoid or linear) embedded with an objective function (E_γ or $\mathrm{cv\_svm}$) for a specific task (*ZGL*, *ZHS* or quality) with the two original graphs. We will also discuss whether we are close to the optimal combination.

We observe that the performance of Δ_{LIN} for *ZGL* is usually equivalent to the best of the two input graphs. This confirms the results obtained in previous works on linear combination of kernels (see for instance [2]). We should note that the objective function $\mathrm{cv\_svm}$ is slightly better than the energy-based function E_γ. We also observe that, in both cases, our combination method does not allow to obtain the accuracy of the optimal linear combination.

Let us now consider the sigmoid combination. It is worth-noting that the accuracy of the SSL algorithm using the sigmoid smoothness operator is better than the accuracy of the SSL algorithms on the two input graphs. To the best of our knowledge, it is the first time that a combination method allows to outperform the best of the input graphs.

We obtain similar observations for the quality measure and the *ZHS* algorithm. It should be noted that *ZHS* also depends on a learning parameter that is usually hard to tune. In all experiments, the sigmoid combination showed improved sturdiness with respect to the choice of this parameter (compared to the original graphs G_1, G_2 and to the linear combination).

Now, let us vary the proportion of labeled examples. Figure 1 shows the evolution of the accuracy of the *ZGL* algorithm when the unlabeled proportion varies between 30% and 95% on the Credit data set. It can be noted than the accuracy using the objective functions E_γ or $\mathrm{cv\_svm}$ and the sigmoid combination are closed to the accuracy of the optimal combination. We can also note that are very good up to 90% of unlabeled nodes and still good for 95% of unlabeled nodes.

Table III reports the results where we can again observe that the sigmoid combination is better, even if we are far from the ideal value. The reason for this "gap" is that the task is harder in this case: the true minimum is computed using the complete knowledge of the labels.

V. Discussion and Conclusions

We have proposed a new method for combining several graphs that represent topologies of heterogeneous relations for the same set of nodes. Our combination is done through graph-specific latent spaces, where the distances between points are directly linked to the commute time distance in the graphs. Our merging algorithm focuses only on a small subset of labeled

Dataset	MR	Δ_1	Δ_2	$\Delta_{\mathrm{LIN(opt)}}$	$\Delta_{\mathrm{LIN(E_\gamma)}}$	$\Delta_{\mathrm{LIN(cv\_svm)}}$	$\Delta_{\mathrm{SIG(opt)}}$	$\Delta_{\mathrm{SIG(E_\gamma)}}$	$\Delta_{\mathrm{SIG(cv\_svm)}}$
Credit appr.	0.45	0.29 ± 0.02	0.42 ± 0.02	0.24 ± 0.02	0.41 ± 0.03	0.27 ± 0.02	0.14 ± 0.01	0.15 ± 0.02	0.15 ± 0.02
Flags	0.48	0.38 ± 0.04	0.20 ± 0.03	0.20 ± 0.02	0.20 ± 0.03	0.22 ± 0.04	0.14 ± 0.02	0.16 ± 0.03	0.17 ± 0.03
Stat. Heart	0.44	0.28 ± 0.02	0.29 ± 0.03	0.24 ± 0.02	0.29 ± 0.03	0.28 ± 0.02	0.16 ± 0.02	0.18 ± 0.02	0.20 ± 0.02
IMDB	0.43	0.27 ± 0.01	0.43 ± 0.0	0.27 ± 0.01	0.33 ± 0.07	0.27 ± 0.01	0.21 ± 0.01	0.22 ± 0.01	0.22 ± 0.01
WebKB	0.42	0.42 ± 0.0	0.15 ± 0.02	0.11 ± 0.01	0.42 ± 0.0	0.11 ± 0.01	0.08 ± 0.0	0.08 ± 0.0	0.08 ± 0.0
Adult	0.26	0.23 ± 0.01	0.26 ± 0.0	0.22 ± 0.01	0.24 ± 0.02	0.25 ± 0.01	0.20 ± 0.01	0.20 ± 0.01	0.22 ± 0.02
Horse	0.36	0.35 ± 0.02	0.36 ± 0.0	0.33 ± 0.02	0.36 ± 0.0	0.35 ± 0.02	0.16 ± 0.02	0.18 ± 0.03	0.16 ± 0.02

TABLE II
MEAN AND STANDARD DEVIATION OF ERROR RATES FOR THE *ZGL* ([22]) ALGORITHM FOR A PROPORTION OF 70% OF UNLABELED DATA. COLUMN MR CORRESPONDS TO THE MAJORITY VOTE RULE; Δ_i ARE RESULTS FOR *ZGL* ON GRAPH G_i WITHOUT COMBINATION; opt IS THE OPTIMAL VALUE FOR THE COMBINATION METHOD ; OTHER COMLUMS GIVE RESULTS FOR THE TWO COMBINATION METHODS WITH THE TWO OBJECTIVE FUNCTIONS.

Dataset	MR	Δ_1	Δ_2	$\Delta_{\mathrm{LIN(opt)}}$	$\Delta_{\mathrm{LIN(E_\gamma)}}$	$\Delta_{\mathrm{LIN(cv\_svm)}}$	$\Delta_{\mathrm{SIG(opt)}}$	$\Delta_{\mathrm{SIG(E_\gamma)}}$	$\Delta_{\mathrm{SIG(cv\_svm)}}$
Credit appr.	0.45	0.28	0.44	0.28	0.44 ± 0.0	0.35 ± 0.03	0.08	0.27 ± 0.0	0.27 ± 0.01
Flags	0.48	0.39	0.45	0.39	0.45 ± 0.0	0.44 ± 0.0	0.04	0.22 ± 0.0	0.26 ± 0.0
Stat. Heart	0.44	0.36	0.41	0.36	0.39 ± 0.02	0.36 ± 0.01	0.13	0.28 ± 0.02	0.26 ± 0.05
IMDB	0.43	0.43	0.48	0.43	0.45 ± 0.02	0.43 ± 0.0	0.31	0.32 ± 0.0	0.32 ± 0.0
WebKB	0.42	0.49	0.32	0.32	0.49 ± 0.0	0.38 ± 0.01	0.13	0.13 ± 0.0	0.14 ± 0.01
Adult	0.26	0.30	0.35	0.30	0.31 ± 0.01	0.33 ± 0.01	0.21	0.27 ± 0.0	0.31 ± 0.04
Horse	0.36	0.41	0.42	0.41	0.42 ± 0.0	0.42 ± 0.0	0.21	0.24 ± 0.04	0.21 ± 0.01

TABLE III
QUALITY MEASURE $\mathcal{Q}(\Delta)$ OF THE SMOOTHNESS OPERATORS FOR A PROPORTION 70% OF UNLABELED NODES.

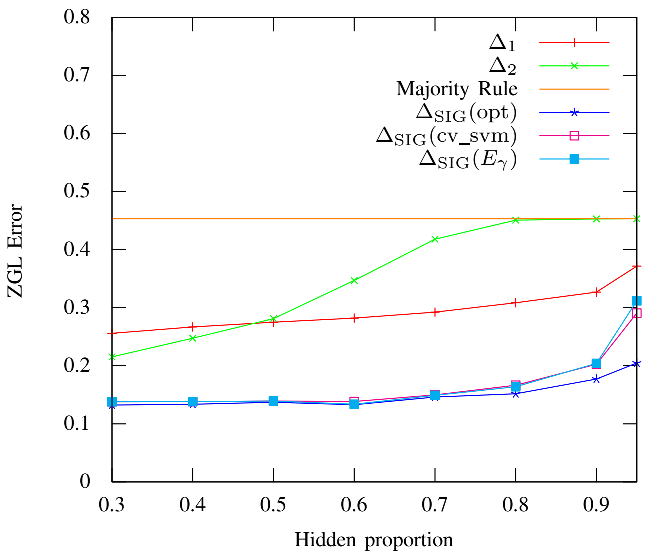

Fig. 1. Evolution of the error rate of the *ZGL* algorithm for the sigmoid combination on the Credit approval dataset depending of the ration of unlabeled examples.

points that supervise this combination through two different objective functions. The first one, based on the previous work of [2] is stable and efficient but require a preliminary step of parameter tuning. The second one based on SVM optimization is more volatile but has shown very positive results. Two kernel settings have been explored: linear and sigmoid. The linear kernel performs typically as good as the best of the input graphs while the sigmoid kernel outperforms significantly the two input graphs in semi-supervised learning and smoothness quality. Both settings lead to a final object that can be interpreted as a smoothness measure operator, what allow us to use it in many contexts (graph-like regularization term).

As future work, we plan to propose new strategies to tune the parameters of the sigmoid combination. We propose also to study more in detail the properties of other kernel combinations and experiment on other learning tasks (such as clustering, link prediction or ranking algorithms). Another problem to address is big graphs and networks whose set of relations is more diverse and complex.

REFERENCES

[1] A. Argyriou, , C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Learning Theory*, page 338–352. Springer, 2005.
[2] A. Argyriou, M. Herbster, and M. Pontil. Combining graph laplacians for semi-supervised learning. In *NIPS*, 2005.
[3] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*, chapter Discrete regularization, pages 221–232. MIT Press, 2006.
[4] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
[5] F. Fouss, A. Pirotte, J-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *TKDE*, 19(3):355–369, 2007.
[6] F. Fouss, A. Pirotte, and M. Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *Web Intelligence*, pages 550–556, 2005.
[7] M. Gönen and E. Alpaydin. Multiple kernel learning algorithms. *J. Mach. Learn. Res.*, 12:2211–2268, 2011.
[8] J. He and Lawrence R. A graph based framework for multi-task multi-view learning. In *ICML*, 2011.
[9] D. J. Klein and M. Randic. Resistance distance. *Mathematical Chemistry*, 12:81–95, 1993.
[10] Yen L., F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. Graph nodes clustering based on the commute-time kernel. In *PAKDD*, volume 4426, pages 1037–1045, 2007.
[11] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.
[12] U. Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the commute distance. In *NIPS*, 2010.
[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[14] K. Schmidt and G. Trenkler. The Moore-Penrose inverse of a semi-magic square is semi-magic. *International Journal of Mathematical Education in Science and Technology*, 32(4):624–629, 2001.

[15] W. Wang and Z. Zhou. A new analysis of co-training. In *ICML*, pages 1135–1142, 2010.

[16] D. Zhou, O. Bousquet, T. Navin Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2003.

[17] D. Zhou and J.C. Burges. Spectral clustering and transductive learning with multiple views. In *ICML*, pages 1159–1166, 2007.

[18] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, pages 1036–1043, 2005.

[19] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, pages 1601–1608, 2006.

[20] D. Zhou, S.A. Orshanskiy, H. Zha, and C.L. Giles. Co-ranking authors and documents in a heterogeneous network. In *ICDM*, pages 739–744, 2007.

[21] D. Zhou, S. Zhu, K. Yu, X. Song, B.L. Tseng, H. Zha, and C.L. Giles. Learning multiple graphs for document recommendations. In *WWW*, pages 141–150, 2008.

[22] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 20, pages 912–919, 2003.