Graph Construction Based on Labeled Instances for Semi-supervised Learning

Conference Paper · August 2014

DOI: 10.1109/ICPR.2014.428

2 authors:

L. Berton
Universidade Federal de São Paulo
54 PUBLICATIONS **308** CITATIONS
SEE PROFILE

Alneu de Andrade Lopes
University of São Paulo
98 PUBLICATIONS **1,120** CITATIONS
SEE PROFILE

2014 22nd International Conference on Pattern Recognition

Graph construction based on labeled instances for Semi-Supervised Learning

Lilian Berton, Alneu de Andrade Lopes
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - Campus de São Carlos
13560-970 São Carlos, SP - Brazil
Email: {lberton, alneu}@icmc.usp.br

***Abstract*—Semi-Supervised Learning (SSL) techniques have become very relevant since they require a small set of labeled data. In this context, graph-based algorithms have gained prominence in the area due to their capacity to exploiting, besides information about data points, the relationships among them. Moreover, data represented in graphs allow the use of collective inference (vertices can affect each other), propagation of labels (autocorrelation among neighbors) and use of neighborhood characteristics of a vertex. An important step in graph-based SSL methods is the conversion of tabular data into a weighted graph. The graph construction has a key role in the quality of the classification in graph-based methods. This paper explores a method for graph construction that uses available labeled data. We provide extensive experiments showing the proposed method has many advantages: good classification accuracy, quadratic time complexity, no sensitivity to the parameter $k > 10$, sparse graph formation with average degree around 2 and hub formation from the labeled points, which facilitates the propagation of labels.**

***Keywords*—graph construction; semi-supervised learning classification; complex network**

I. Introduction

Labeled data are difficult, expensive and time consuming to be prepared in contrast to unlabeled data. Because of this, Semi-Supervised Learning (SSL) algorithms have gained prominence, due to their ability to learn from limited amounts of labeled data combined with widely available unlabeled data. In particular, graph-based SSL algorithms have been successfully used in different tasks [19], [6]. Graph-based SSL methods require a data set whose instances are represented by the vertices of a graph. The labeled vertices are used to propagate information to the unlabeled ones. These methods generally use a transductive approach.

Most of the graph-based SSL algorithms concentrate on the label inference task, i.e. assigning labels to unlabeled nodes once the graph has already been constructed, with very little emphasis on the construction of the graph itself. Only recently, the issue of graph construction has received attention [11], [13], [16]. Zhu (2005) [19] argues that it is more important to construct a good graph than to choose among the inference methods.

Neighborhood graphs have been used in many areas of Machine Learning to model local relationships between data points. The most popular algorithm for generating graphs is k-nearest neighbor (kNN), in which each vertex considers its k nearest neighbors using a similarity function and instantiates k undirected edges between itself and these neighbors. There are also the mutual kNN in which there is a connection between two vertices only if the rule of nearest neighbor is reciprocal, i.e. each one belongs to the k-nearest neighbors of the other. Hence the mutual kNN are considered more restrictive and it is traditionally used in unsupervised learning [5], [12].

Most of the graph construction methods are unsupervised, i.e. they do not employ available label information during the graph construction process. Dhillon et al., (2010) [9] addressed this problem and explored labeled points to compute similarity between pair of instances. Rohban and Rabiee (2012) [14] proposed a supervised graph construction, showing that under the using of large enough manifold sampling rate, the optimal neighborhood graph is subgraph of a kNN graph.

Labeled data may be seen as a type of prior information which can be useful for improving graph construction for the current learning task. In earlier research we reported a method for graph construction that uses the available labeled data [2], denominated here by *Graph-based on informativeness of labeled instances* (GBILI). Inspired by the effectiveness of the approach, we extended it by improving the connectivity and sparsity of the graph, which is reflected in the label propagation process.

The approach was demonstrated providing extensive evidence in the following topics: i) The proposed technique (GBILI) leads to good classification accuracy achieving better results than the traditional method kNN. By Nemenyi statistics [8] GBILI is better ranked than kNN. The classification accuracy results are measured on various data sets commonly used in SSL [6]; ii) A detailed complexity analysis shows GBILI has a quadratic time complexity, the same as kNN graph; iii) A parameter sensitivity analysis varying k from 1 to 50 shows that for $k > 10$ GBILI presents a stability in classification accuracy. How parameter selection is a problem for many methods [14], GBILI has an advantage in this point; iv) Analysis about network density show that kNN graphs become very dense as k increases. In contrast, GBILI graph converges for a constant average degre (around 2) independent of the value k. It is sparse, meaning it reduces the processing cost and ensure the SSL algorithms remain efficient. Besides, by using the ϕ-edge ratio measure, when the parameter k becomes higher, the number of edges connecting vertices with different labels increases in kNN graphs, resulting in propagation of wrong label information. This situation does not happen in GBILI graphs; v) GBILI method leads the labeled points to become hubs. It is indicated by calculating centrality measures

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.428

of Complex Network [15], like *node degree*, *betweenness*, *eigenvector* and *pageRank*. These measures are related with diffusion processes in a network, like information or disease spreading. As the labeled points in GBILI graphs are hubs they facilitate the label propagation.

The remaining of the paper is organized as follows: Section II introduces definitions, notations and basic quantities used to describe the topology of a network; Section III presents the proposed graph construction method; Section IV demonstrates the complexity analysis; section V reports experimental results on various data sets commonly used in SSL, besides provides a parameter sensitivity analysis, a network density analysis and a network characterization by centrality measures from Complex Network. Finally, Section VI presents concluding remarks.

II. Definitions

Given a set of l labeled instances $L = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ and a set of u unlabeled instances $U = \{x_{l+1}, \ldots, x_{l+u}\}$, the goal of SSL is to infer labels for the set of unlabeled instances U. In most cases, the data instances are assumed to be independent and identically distributed (i.i.d.). At this point, for applying any label propagation technique the common practice is first to create a graph from the data instances, and then to apply one of the graph-based SSL methods on the constructed graph.

A graph [1] $G = (V, E)$ consists of two sets V and E. The elements of $V = \{v_1, v_2, \ldots, v_N\}$ are the nodes or vertices of the graph G where each vertex v_i is associated with the instance x_i from the input data X and the cardinality of $|V|$ is N. The elements of $E = \{e_1, e_2, \ldots, e_M\}$ are links or edges between nodes and the cardinality of $|E|$ is M. An edge connecting the vertices v_i and v_j is denoted by e_{ij}. A graph can be weighted, in this case it is represented by a set of values (weights) $W = \{w_1, w_2, \ldots, w_M\}$ that are real numbers assigned to the links.

For a graph G of size N, the number of edges M is at least 0 and at most $N(N-1)/2$ (when all the nodes are pairwise adjacent). G is said to be *sparse* if $M \ll N^2$ and *dense* if $M = O(N^2)$. A graph is said to be connected if, for every pair of distinct nodes v_i and v_j, there is a path from v_i to v_j, otherwise it is said disconnected. A *component* of the graph is a maximally connected induced subgraph.

It is usually considered a matricial representation of a graph. A graph $G = (V, E)$ can be described by the adjacency matrix P, a $N \times N$ square matrix whose entry p_{ij} $(i, j = 1, \ldots, N)$ is equal to 1 when the link p_{ij} exists, and 0 otherwise. The diagonal of the adjacency matrix contains zeros. This is, therefore, a symmetric matrix for undirected graphs.

The degree g_i of a node i is the number of edges incident with the node, and is defined in terms of the adjacency matrix P as $g_i = \sum_{j \in N} p_{ij}$. It is is related with the number of edges and vertices as $\langle g \rangle = \frac{2|E|}{|V|}$. If a node has a degree much bigger than the others nodes, it is called *hub*. The average degree for a network is defined as $\langle g \rangle = \frac{1}{N} \sum_{n \in N} g_n$.

[1] We make no distinction between graph and network.

III. Graph construction based on informativeness of labeled instances - GBILI

Figure 1(A) illustrates a toy example with two groups of points forming a square and another one forming a line of points. If we take into account the visual patterns in the example, a human clearly would observe the three groups. However, learning methods based on distance generally have difficulty in making this pattern recognition. Based on this fact, the proposed method for graph construction attempts to recognize certain patterns, especially those related to variations in data density. For that it makes use of the mutual kNN, since this method prevents connection between subgraphs of different groups. Figure 1(A) shows the result by employing the proposed technique and it identifies three groups of points. Figure 1(B and C) shows the result by using kNN which do not identify the groups correctly.

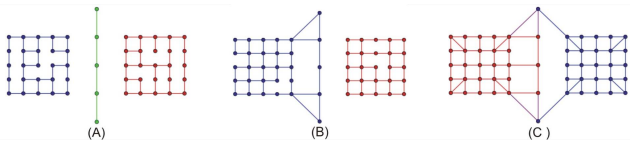


Fig. 1. Classification results using the proposed method (A) and kNN (B, C) for graph construction. (A) k = 3, (B) k = 2, (C) k = 3.

The technique also seeks to exploit a prior information available (in this case the labels of labeled vertices) for the network construction, prioritizing connections between vertices that are closer to a labeled point. Such strategy turns labeled points into hubs, especially when the values of k increase (more details in Section V).

Since this technique for graph construction considers information conveyed by labeled instances and their neighborhood to make the connections, it is referred as *Graph-Based on Informativeness of Labeled Instances* (GBILI). The method, for each vertex, chooses the most "informative" node to establish a connection. Such informativeness is obtained optimizing the equation: $min \sum_i \sum_j (D_{ij} + \sum_l D_{jl})$, s.t. $D_{ij} \geq 0,\ i \in \{1 \ldots n\},\ j \in$ Mutual kNN of $i, l \in L$, where Mutual kNN is the set of mutual neighbors of a vertex, L is the set of labeled vertices and D is the distance matrix.

Algorithm 1 shows the steps to construct the graph G. Initially, it is necessary to generate a distance matrix D, Euclidean distance can be employed to generate it. In this matrix we can find the k nearest neighbors of the elements. It is necessary to set the parameter K with a natural value and generate a list of the labeled points L.

In the algorithm, steps 5 to 8 find the k nearest neighbors for a vertex v_i, then, steps 9 to 12 search for the mutual kNN for v_i and store them into a list. Steps 13 to 16 calculate the sum of the distances from v_i to each element of the list of mutual kNN and from these elements to a labeled point. It creates a connection between v_i and v_j that minimizes this sum. Steps 17 to 22 post-process the graph connecting isolated components. It is important because mutual kNN graph, particularly for small values of k, often contain many disconnected components. These steps perform a Breadth-First Search (BFS) looking for components in the network. Components with no labeled point are connected with a neighboring

component with labeled point. We limit these new connections by a reduced number of links in order to avoid the network become too much dense.

Algorithm 1 *GBILI algorithm*

```
generate a distance matrix D
generate a list of labeled points L
set the parameter K
For i = 1; i < |V|; i++
    For k = 1; k < K; k++
        For j = 1; j < |V|; j++
            if D(v_i, v_j) is the k-nearest neighbor
                Store v_j in the kNN-List(v_i)
    For j = 1; j < kNN-List(v_i); j++
        For k = 1; k < K; k++
            if D(v_j, v_i) is the k-nearest neighbor
                Store v_j in the M-kNN(v_i)
    For j = 1; j < M-kNN(v_i); j++
        For l = 1; l < |L|; l++
            if D(v_i, v_j) + D(v_j, v_l) is min
                Store e_ij in G
Do BFS and return Component(G)
  For i = 1; i < |V|; i++
    if Component(v_i) ∉ L
        For k = 1; k < kNN-List(v_i); k++
            if Component(v_k) ∈ L
                Store e_ik in G
return G
```

IV. Complexity analysis

Initially, it is necessary to compute a similarity among all pairs of nodes using a similarity function. This function generates a full adjacency matrix $D \in \Re^{n \times n}$, where $D_{ij} = d(v_i, v_j)$ is computed using Euclidean distance. To calculate the Euclidean distance it takes $\frac{n(n-1)}{2}$ steps, where n is the number of elements. Its complexity is $O(n^2)$.

Subsequently, we construct the graph. In this step the matrix D is sparsified and reweighted to produce the final matrix W. The sparsification is important because it improves the efficiency in the label inference stage. It generates a binary matrix $\widehat{P} \in B^{n \times n}$, where $\widehat{P}_{ij} = 1$ indicates that there is an edge between v_i and v_j, and $\widehat{P}_{ij} = 0$ indicates the edge is absent (assume $\widehat{P}_{ii} = 0$).

Based on the similarity matrix it is necessary to find the k nearest neighbors of each element. For finding the kNN, the complexity is $O(kn^2)$, where n is the number of elements and k is the number of neighbors considered. But we need to calculate the mutual kNN, in this case, for each jth neighbor of an element v_i we need to check if $v_i \in k_{neighbors}(v_j)$. For this reason, it takes k^2n more steps. The final complexity to find the k mutual neighbors is $O(kn^2 + k^2n)$. Ozaki et al. (2011) [16] use a Fibonacci heap-based implementation and construct the mutual kNN graph in $O(n^2 + kn\ log\ n)$ time.

The proposed method creates a connection between v_i and v_j that minimizes the sum of the distances from v_i to its mutual kNN and from this mutual kNN to a labeled point v_l. For each element, we need to access the similarity matrix kl times, where k is the mutual neighbors of v_i and l is the number of labeled points. This cost is added with the cost of finding the kNN mutual neighbor.

In the post-processing step it is necessary to find all components of the network, this can be done by Breadth-First Search (BFS). This algorithm begins at a root node and inspects all the neighboring nodes. For each node it inspects their neighbor nodes which were unvisited, and so on. Its time complexity is $O(|V| + |E|)$ [7], where $|V|$ is the number of vertices, and $|E|$ is the number of edges. Since the constructed graph has average degree 2 (shown in Section V), by the equation $\langle k \rangle = \frac{2E}{V}$, the number of edges is equal the number of vertices. When a component is identified, it is marked if it has or not a labeled point. If a vertex v_i belongs to a component that have no labeled points, it will connect to one k nearest neighbors that belongs to a component with labeled point. In the worst case we need to look for k neighbors for n points. Its complexity is $O(kn)$.

Finally, the complexity of the proposed algorithm results in $O(kn^2 + k^2n + kln)$ + $O(2n + kn)$, close to kNN method.

V. Experiments

In this section we present extensive empirical results exploring the GBILI method. Subsection V-A presents the description of the data sets used and the setup of the experiments; subsection V-B shows the classification results; subsection V-C shows a parameter sensitivity analysis, a density analyzes of the graphs and the ϕ-Edge Ratio comparison between kNN and GBILI graphs. It is also provided the network visualization, which helps to understand the network topology; subsection V-D presents some centrality measures from Complex Networks used to characterize GBILI network.

A. Data sets and experimental setup

The experiments were carried out on six data sets described in Table I. These data sets are frequently used in SSL literature and they were proposed by Chapelle et al. (2006) [6].

TABLE I. Data sets descriptions.

Data set	# Instances	# Atrributes	# Classes	Comment
Digit_1	1500	241	2	artificial and balanced
g241c	1500	241	2	artificial and balanced
g241n	1500	241	2	artificial and balanced
COIL_2	1500	241	2	real and balanced
COIL_6	1500	241	6	real and balanced
USPS	1500	241	2	real and imbalanced

First, we apply *Principal Component Analysis* (PCA) to all data sets reducing the dimensions to 50, because in high-dimensional data, the distance to the nearest neighbor approaches the distance of the farthest neighbor, degenerating the quality of the graph. Then, we run experiments using 10 and 100 labeled vertices randomly selected from all the points.

For the graph construction we apply kNN and GBILI methods. All such methods operate on the distance matrix $D \in \Re^{n \times n}$, obtained from the Euclidean distance among the points. This process will result in a matrix $\widehat{P}$ where $\widehat{P}_{ij} = 1$ if a point j have a connection to i and $\widehat{P}_{ij} = 0$ otherwise. Finally, this matrix is symmetrized as follows $P_{ij} = max(\widehat{P}_{ij}, \widehat{P}_{ji})$. To generate the weighted graph W we use the binary weighting approach ($W = P$). Some authors use the *Gaussian kernel* for weigh the graph. However, they do not have an agreement about which value apply in the kernel bandwidth parameter σ.

The algorithm used for the label inference task is Local and Global Consistency (LGC) [17], which is frequently used in the literature [11], [16]. LGC solves the optimization problem $F = argmin_{F \in \Re^{n \times c}} tr(F^T L F + \mu (F-Y)^T (F-Y))$, which gives the closed-form solution $F = (I + L/\mu)^{-1} Y$. I is the identity matrix, Y the set of known labels and L is the normalized Laplacian defined by $L = \mathbf{D}^{1/2} L \mathbf{D}^{1/2} = I - \mathbf{D}^{1/2} W \mathbf{D}^{1/2}$, where $\mathbf{D}$ is the diagonal matrix with elements $\mathbf{D}_{ii} = \sum_j W_{ij}$. Average classification accuracy of 30 runs is used as the evaluation measure.

Any other method based on graph can be applied for the label propagation task, for example, Gaussian Random Fields (GRS) [18], Laplacian Regularized Least Squares (Laplacian RLS) [1], etc. We compare the classification accuracy to the results presented by Chapelle et al. (2006) [6], for the following algorithms: 1-NN, Discrete Reg., Transductive SVM, Cluster Kernel, Low-density separation (LDS), Laplacian RLS. It is worth pointing out that we use 10 and 100 labeled data randomly selected from all the points, while Chapelle et al. (2006) split the data into 12 partitions and select 10 and 100 labeled points from these partitions.

B. Classification results

In Tables II and III are the accuracy of the different methods for 10 and 100 labeled points for g241c, g241, digit_1, COIL_6 and USPS data sets.

TABLE II. AVERAGE ACCURACY WITH 10 LABELED POINTS.

	g241c	**g241n**	**digit$_1$**	**COIL$_6$**	**USPS**
1-NN	55.95	56.78	76.53	34.09	80.18
Discrete Reg.	50.41	50.95	87.36	36.62	83.93
TSVM	75.29	49.92	82.23	32.5	74.8
Cluster-Kernel	51.72	57.95	81.27	32.68	80.59
LDS	71.15	49.37	84.37	38.1	82.43
Laplacian RLS	56.05	54.32	94.56	45.46	81.01
***k*NN+LGC**	55.16	51.69	89.61	41.59	83.54
GBILI+LGC	56.96	56.81	84.37	39.95	85.07

TABLE III. AVERAGE ACCURACY WITH 100 LABELED POINTS.

	g241c	**g241n**	**digit$_1$**	**COIL$_6$**	**USPS**
1-NN	59.72	62.51	93.88	76.73	92.36
Discrete Reg.	56.35	58.35	97.23	90.39	95.32
TSVM	81.54	77.58	93.85	74.2	90.23
Cluster-Kernel	86.51	95.05	96.21	78.01	90.32
LDS	81.96	76.26	96.54	86.28	95.04
Laplacian RLS	75.64	73.54	97.08	88.08	95.32
***k*NN+LGC**	59.29	57.81	95.6	82.92	85.07
GBILI+LGC	61.77	66.95	95.6	82.92	94.1

We run the Nemenyi post-hoc test [8] to verify if it is possible to detect significant differences among algorithms from the results of the Tables II and III. According to the Nemenyi statistics, the critical value for comparing the average-ranking of two different algorithms at 95 percentile is 3.50. The analysis is shown in Figure 2. The critical difference (CD) is on the top and the average ranks of measures are in the axis of the diagram. The lowest (best) ranks are in the left side, where we note that GBILI is better ranked than kNN. The methods analyzed have no significant difference, therefore they are connected by a black line in the diagram.

C. Parameter sensitivity and network density analysis

Since we do not know a priori the optimal parameter k to construct the graph, we test values of k ranging from 1 to 50 to know how sensitive the graph generation methods are to this input parameter. Figure 3 shows the classification accuracy of kNN and GBILI methods for 100 labeled examples. From these results we observe that kNN method usually leads to good results for values of k smaller than 10, because higher values of k turn the graph dense. In contrast, GBILI leads to better results for values of k bigger than 10, because it uses mutual kNN that finds less neighbors than kNN method, so it is necessary a higher value of k to have more mutual neighbors. In GBILI graphs, we observe the classification accuracy stabilizes after $k > 10$, indicating we could use a fixed value of k bigger than 10.

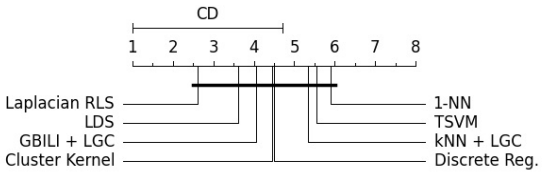


Fig. 2. Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different are connected.

Figure 4 shows the comparison of average degree among kNN, mutual kNN and GBILI. Just GBILI is not affected by k and achieves a constant number of edges, presenting an average degree around 2, independent of the k. Hence, the proposed method generates sparser graphs than kNN.

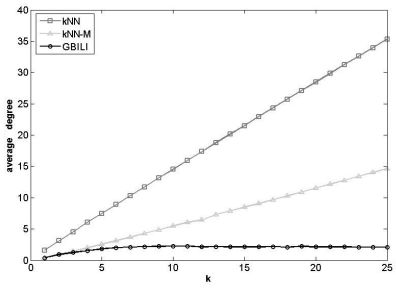
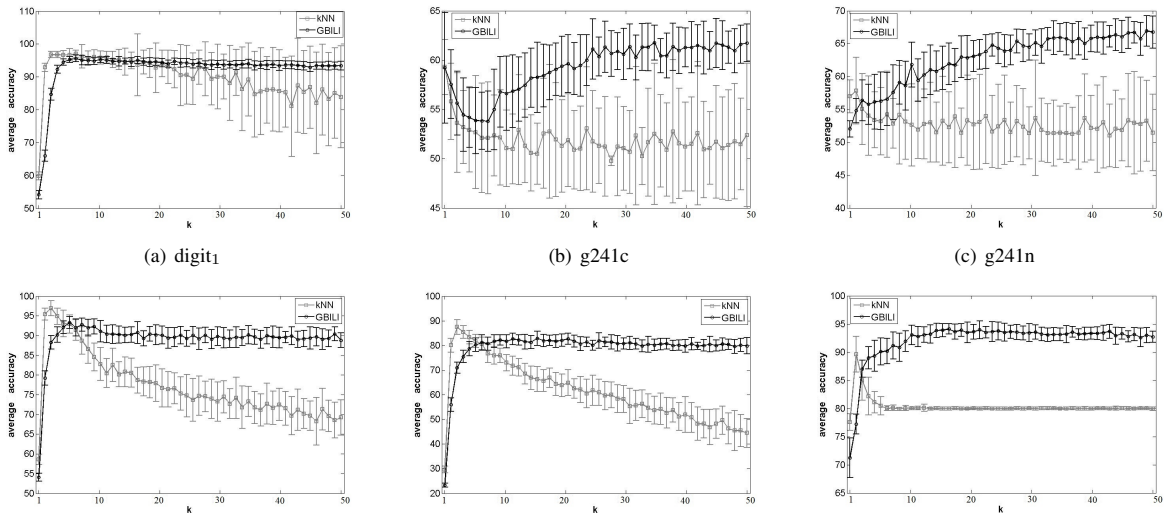


Fig. 4. Average degree in a kNN, mutual kNN and GBILI graphs built using the USPS data set.

Figure 5 shows the graphs generated by kNN and GBILI methods for USPS data set with 10 labeled points. We set the value of k according to the high accuracy achieved by the methods. Hence kNN uses k equal 3 and generates a network with average degree around 4.5 and GBILI uses a k equal 30 and generates a network with average degree around 2. The quantity of edges in a network is related with the processing cost and efficiency of the SSL algorithms, so it is interesting have sparser graphs.

The density affects the methods classification accuracy as we can see by comparing the ϕ-edge ratios of kNN and GBILI methods. We utilize *ϕ-edge ratio* as Ozaki et al. (2011) [16] to measure the quality of a graph. These authors define *ϕ-edge* of a labeled graph $(G, \mathbf{y})$ as any edge (v_i, v_j) for which $y_i \neq y_j$, and *ϕ-edge ratio* of a graph as the number of ϕ-edge divided by the total number of edges in the graph.

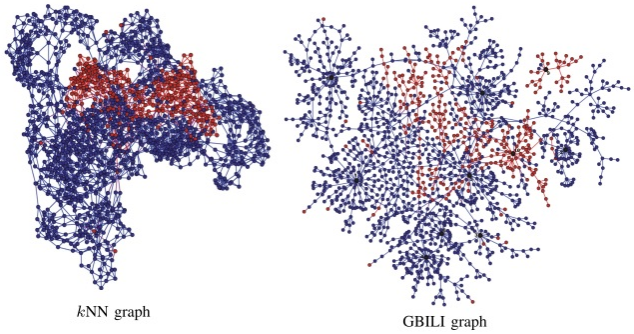
Since most graph-based SSL classification methods propa-



(a) digit$_1$ (b) g241c (c) g241n

(d) COIL$_2$ (e) COIL$_6$ (f) USPS

Fig. 3. Average accuracy rates and standart deviations comparisons for (a)digit$_1$, (b)g241c, (c)g241n, (d)COIL$_2$, (e)COIL$_6$ and (f)USPS data sets using 100 labeled examples.



kNN graph GBILI graph

Fig. 5. kNN and GBILI graphs for USPS data set using 10 labeled examples.

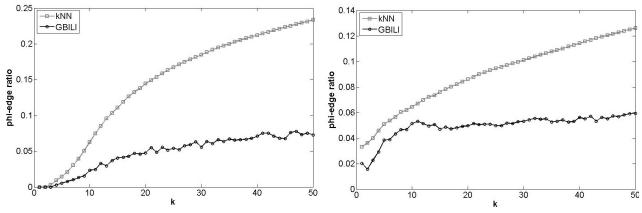


Fig. 6. Average ϕ-edge comparisons for COIL$_2$ (left) and USPS (right) data sets using 10 labeled examples with kNN and GBILI graph construction. The smaller the ϕ-edge ratio, the better the graph.

gate label information throw the network, edges connecting vertices with different labels may lead to misclassification. Hence, a graph with a smaller ϕ-edge ratio is better.

Figure 6 shows the plots of ϕ-edge ratios of GBILI and kNN methods for COIL$_2$ and USPS data sets. We notice that the same pattern occurs for all data sets. The y-axe denotes the ϕ-edge ratio of the constructed graphs and the x-axe denotes the parameter k used. But the number of edges is different in the resulting graphs for these methods: as the value of k increases, the number of edges in kNN increases too, but in GBILI the number of edges is always the same. This way, kNN method achieves higher ϕ-edge ratio than GBILI and a worst classification accuracy when the value of k becomes higher.

D. Complex networks measures

GBILI method generates hubs from the labeled points (black nodes in Figure 5). This network approximates to an "exponential network" since the probability of finding a node with connectivity (or degree) k different from the average connectivity decays exponentially fast for large k. We explore centrality measures from the GBILI network and we demonstrate that labeled vertices are important in the network topology. The measures explored are *node degree*, *betweenness* [10], *eigenvector centrality* [3] and *pageRank* [4].

These measures are presented in Tables: IV (node degree), V (betweenness), VI (eigenvetor) and VII (pageRank) for digit$_1$, g241c, g241n, COIL$_2$, COIL$_6$ and USPS data sets with 10 labeled points. We present the 10 highest values from the GBILI networks. The vertices numerate from 1 to 10 are labeled and their measures are shown in bold in the tables. The subscript is the number of the vertex.

In real networks the presence of nodes with a very large number of connections (hubs) facilitates spreading information or epidemics, especially if the hubs are infected. The GBILI algorithm generate hubs from the labeled points, as we can see in Table IV, where most of the labeled points have the biggest degree in the network. In Table V many labeled points have high betweenness, so they belong to many geodesic paths. It means these points influence many nodes in the network and are important to transmit information throw the network. Furthermore, a vertex is important also when it receives connections from important vertices. Such relevance is

indicated by Eingenvector centrality and PageRank, see Tables VI and VII, respectively.

The network topology is related to information propagation on Complex Networks. GBILI method construct networks exploring the labeled vertices and these points can become hubs. The centrality measures, calculated here, indicate these vertices are important in the network, in this way, GBILI topology can facilitate the label propagation task.

TABLE IV. NODES DEGREE.

digit$_1$	**g241c**	**g241n**	**COIL$_2$**	**COIL$_6$**	**USPS**
30$_2$	**32**$_{10}$	**31**$_6$	94$_{17}$	**30**$_{10}$	**28**$_1$
29$_6$	**31**$_9$	**28**$_8$	**86**$_5$	**28**$_1$	**26**$_6$
27$_7$	25$_{77}$	24$_{1491}$	**56**$_9$	**25**$_5$	**25**$_4$
26$_3$	**25**$_3$	24$_{1348}$	**50**$_1$	**23**$_9$	**25**$_7$
23$_5$	**22**$_4$	**21**$_2$	**45**$_{10}$	**21**$_8$	**23**$_5$
22$_1$	19$_{1060}$	19$_{20}$	**43**$_4$	21$_{402}$	**21**$_2$
18$_4$	19$_{1205}$	**19**$_1$	40$_{50}$	**17**$_3$	**19**$_8$
17$_{10}$	18$_{468}$	17$_{1456}$	**28**$_{11}$	17$_{257}$	**19**$_{10}$
16$_9$	17$_{32}$	16$_{192}$	26$_{65}$	**16**$_4$	16$_{936}$
12$_{612}$	**16**$_8$	16$_{769}$	25$_{38}$	12$_{682}$	14$_{398}$

TABLE V. NODES BETWEENNESS.

digit$_1$	**g241c**	**g241n**	**COIL$_2$**	**COIL$_6$**	**USPS**
320972$_2$	**374096**$_9$	**318244**$_6$	**235375**$_5$	**129459**$_8$	**290077**$_1$
231152$_6$	**363743**$_{10}$	**264527**$_8$	156545$_{337}$	76138$_{130}$	**257168**$_5$
201053$_5$	**245290**$_3$	226145$_{20}$	151665$_{36}$	**71748**$_5$	186240$_{581}$
184264$_3$	**201235**$_4$	175395$_{1348}$	140075$_{17}$	71437$_{1142}$	**180672**$_7$
179110$_9$	**140562**$_5$	167572$_{63}$	**132353**$_1$	59952$_{286}$	**179414**$_4$
167644$_{133}$	135788$_{32}$	**166780**$_1$	128723$_{92}$	58196$_{379}$	**160360**$_6$
154881$_1$	127991$_{77}$	162047$_{631}$	**102044**$_8$	50977$_{658}$	**154759**$_{10}$
141761$_4$	97753$_{23}$	149359$_{33}$	65243$_{91}$	50804$_{1069}$	149967$_{398}$
137102$_{1040}$	94256$_{113}$	140287$_{1491}$	56913$_{144}$	49324$_{189}$	**149408**$_8$
127247$_{114}$	88480$_{1205}$	137298$_{410}$	56233$_{70}$	49314$_{24}$	138086$_{983}$

TABLE VI. EINGENVECTOR CENTRALITY.

digit$_1$	**g241c**	**g241n**	**COIL$_2$**	**COIL$_6$**	**USPS**
1.000$_2$	**1.000**$_9$	**1.000**$_6$	1.000$_{17}$	1.000$_{402}$	**1.000**$_1$
0.837$_6$	**0.862**$_{10}$	**0.870**$_8$	**0.863**$_5$	0.919$_{257}$	**0.699**$_6$
0.497$_3$	**0.584**$_3$	0.393$_{1348}$	**0.470**$_9$	0.512$_{682}$	**0.687**$_4$
0.494$_7$	0.452$_{77}$	0.350$_{234}$	0.412$_5$0	**0.487**$_9$	**0.667**$_5$
0.388$_5$	0.399$_{32}$	0.343$_{66}$	0.214$_3$8	**0.443**$_{10}$	**0.609**$_7$
0.346$_1$	**0.391**$_4$	0.328$_{14}$	0.187$_{158}$	**0.387**$_5$	**0.307**$_2$
0.246$_{1275}$	0.343$_{468}$	0.293$_{969}$	0.168$_{218}$	0.381$_{403}$	**0.301**$_{10}$
0.235$_{1219}$	0.305$_{461}$	0.291$_{46}$	0.167$_{15}$	0.381$_{495}$	**0.293**$_8$
0.233$_{34}$	**0.281**$_5$	0.267$_{232}$	0.167$_{929}$	0.372$_{219}$	0.282$_{398}$
0.222$_{1158}$	0.279$_{1063}$	0.266$_{1456}$	0.167$_{90}$	0.372$_{36}$	0.251$_{823}$

TABLE VII. PAGERANK.

digit$_1$	**g241c**	**g241n**	**COIL$_2$**	**COIL$_6$**	**USPS**
0.008$_2$	**0.008**$_{10}$	**0.007**$_6$	0.016$_{17}$	**0.008**$_{10}$	**0.007**$_1$
0.007$_6$	**0.007**$_9$	**0.007**$_8$	**0.015**$_5$	**0.008**$_1$	**0.007**$_6$
0.007$_7$	**0.006**$_3$	0.007$_{1491}$	**0.011**$_1$	**0.006**$_5$	**0.007**$_4$
0.007$_3$	0.006$_{77}$	**0.006**$_2$	**0.010**$_{10}$	**0.005**$_8$	**0.007**$_7$
0.006$_5$	0.006$_{1060}$	0.006$_{1348}$	**0.010**$_9$	**0.005**$_9$	**0.006**$_2$
0.006$_1$	**0.005**$_4$	**0.005**$_1$	**0.008**$_4$	**0.005**$_3$	**0.006**$_5$
0.005$_{10}$	0.005$_{1205}$	0.005$_{1456}$	0.007$_{50}$	0.004$_{402}$	**0.005**$_8$
0.005$_4$	0.005$_{1450}$	0.005$_{769}$	**0.006**$_8$	**0.004**$_4$	**0.005**$_{10}$
0.004$_9$	0.005$_{468}$	0.004$_{192}$	**0.005**$_2$	0.003$_{257}$	0.004$_{936}$
0.004$_{1002}$	**0.004**$_8$	0.004$_{20}$	**0.005**$_{11}$	0.002$_{809}$	0.004$_{398}$

VI. CONCLUSION

In this paper we explore how labeled instances, available in the SSL setting, can be used to construct a better graph for classification in SSL. We provide extensive empirical evidence that this technique called GBILI leads to good classification accuracy. Furthermore, we provide a detailed complexity analys in which we show GBILI has the same time complextiy of kNN graphs. We show that kNN generates more dense networks than GBILI and by using the ϕ-edge ratio measure we confirm that when the parameter k increases, the quantity of edges connecting vertices with different labels increases too, propagating wrong information. We also apply centrality measures from Complex Network to characterize GBILI networks. The exploited measures were node degree, betweenness, eigenvector and pageRank. These measures confirm the labeled points in GBILI graph are hubs and important in network topology. So, they can facilitate the label propagation process.

ACKNOWLEDGMENT

Grant 2011/21880-3 and 2011/22749-8 Sao Paulo Research Foundation (FAPESP).

REFERENCES

[1] Belkin, M.; Niyogi, P.; Sindhwani, V. (2006) *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*. Journal of Machine Learning Research, v. 7, p. 2399-2434.

[2] Berton, L.; Lopes, A. A. (2012) *Informativity-based graph: exploring mutual kNN and labeled vertices for semi-supervised learning*. In 4th International Conference on Computational Aspects of Social Networks, p. 14-19.

[3] Bonacich, P. (1972) *Factoring and weighting approaches to clique identification*. Journal of Mathematical Sociology, v. 2, p. 113-120.

[4] Brin, S.; Page, L. (1998) *The anatomy of a large-scale hypertextual web search engine*. Computer Networks and ISDN Systems, p. 107-117.

[5] Brito, M. R.; Chavez, E. L.; Quiroz, A. J.; Yukich, J. E. (1997) *Connectivity of the mutual k nearest neighbor graph in clustering and outlier detection*. Statistics and Probability Letters, v. 35, p. 33-42.

[6] Chapelle, O.; Schlkopf, B.; Zien, A. editors (2006) *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

[7] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2009) *Introduction to Algorithms*. MIT Press and McGraw-Hill, 3rd ed.

[8] Demsar, J. (2006) *Statistical Comparisons of Classifiers over Multiple Data Sets*. Journal of Machine Learning Research, v. 7, p. 1-30.

[9] Dhillon, P. S.; Talukdar, P. P.; Crammer, K. (2010) *Inference Driven Metric Learning (IDML) for Graph Construction*. Technical Reports (CIS), University of Pennsylvania.

[10] Freeman, L. C. (1977) *A set of measures of centrality based on betweenness*. Sociometry, v. 40, p. 35-41.

[11] Jebara, T.; Wang, J.; Chang, S.F. (2009) *Graph construction and b-matching for semi-supervised learning*. In Proceedings of the 26th Annual International Conference on Machine Learning, p. 441-448.

[12] Maier, M.; Hein, M.; Luxburg, U. (2007) *Cluster Identification in Nearest-Neighbor Graphs*. In Proceedings of the 18th international conference on Algorithmic Learning Theory, p. 196-210.

[13] Maier, M.; Luxburg, U. (2009) *Influence of graph construction on graph-based clustering measures*. The Neural Information Processing Systems, v. 22, p. 1025-1032.

[14] Rohban M. H.; Rabiee, H. R. (2012). *Supervised neighborhood graph construction for semi-supervised classification*. Pattern Recognition, v. 45, p. 1363-1372.

[15] Newman, M. (2003) *The structure and function of complex networks*. SIAM Review, v. 45, p. 167-256.

[16] Ozaki, K.; Shimbo, M.; Komachi, M.; Matsumoto, Y. (2011) *Using the Mutual k-Nearest Neighbor Graphs for Semi-supervised Classification of Natural Language Data*. In Proceedings of the 15th Conference on Computational Natural Language Learning, p. 154-162.

[17] Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; Schlkopf, B. (2004) *Learning with local and global consistency*. In Advances in Neural Information Processing Systems, v. 16, p. 321-328: MIT Press.

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

[19] Zhu, X. (2005) *Semi-supervised learning literature survey*. Technical report 1530 - Computer Sciences, University of Wisconsin-Madison.