Influence of Graph Construction on Semi-supervised Learning

Celso André R. de Sousa, Solange O. Rezende, and Gustavo E.A.P.A. Batista

Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Campus de São Carlos, Brazil
{sousa,solange,gbatista}@icmc.usp.br

Abstract. A variety of graph-based semi-supervised learning (SSL) algorithms and graph construction methods have been proposed in the last few years. Despite their apparent empirical success, the field of SSL lacks a detailed study that empirically evaluates the influence of graph construction on SSL. In this paper we provide such an experimental study. We combine a variety of graph construction methods as well as a variety of graph-based SSL algorithms and empirically compare them on a number of benchmark data sets widely used in the SSL literature. The empirical evaluation proposed in this paper is subdivided into four parts: (1) best case analysis; (2) classifiers' stability evaluation; (3) influence of graph construction; and (4) influence of regularization parameters. The purpose of our experiments is to evaluate the trade-off between classification performance and stability of the SSL algorithms on a variety of graph construction methods and parameter values. The obtained results show that the mutual k-nearest neighbors (mutKNN) graph may be the best choice for adjacency graph construction while the RBF kernel may be the best choice for weighted matrix generation. In addition, mutKNN tends to generate smoother error surfaces than other adjacency graph construction methods. However, mutKNN is unstable for a relatively small value of k. Our results indicate that the classification performance of the graph-based SSL algorithms are heavily influenced by the parameters setting and we found no evident explorable pattern to relay to future practitioners. We discuss the consequences of such instability in research and practice.

Keywords: Semi-supervised learning, graph-based methods, experimental study, classification.

1 Introduction

Semi-supervised learning (SSL) has gained increased attention in the last few years [3,15]. Among all SSL algorithms, graph-based methods are widely used because the weighted graph may approximate the low dimensional manifold in which the data should lie. The research community has proposed a variety of graph-based SSL algorithms [1,8,14,16] as well as a variety of graph construction

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part III, LNAI 8190, pp. 160–175, 2013.

methods [5,8,13]. Despite its increasing popularity, the SSL literature lacks a comprehensive and unbiased empirical study that shows the influence that graph construction methods have in both classification performance and stability of the graph-based SSL algorithms.

1.1 Contributions

In this paper, we provide a detailed empirical comparison of the state-of-the-art, graph-based SSL algorithms combined with a variety of graph construction methods. The empirical analysis proposed in this paper is subdivided into four parts as follows:

Best case analysis. We evaluate the best error rates of each combination of SSL algorithm and graph construction method for a number of sparsification parameter values. Although this is the most common approach to evaluate SSL algorithms in the literature [3], this empirical setting alone may not provide all the necessary information to choose the best classifiers for real applications. For instance, stable classifiers may be preferable over classifiers which are able to provide excellent performance for a very narrow range of parameter values and mediocre performance for the remaining values;

Classifiers' stability evaluation. We evaluate the stability of the SSL algorithms combined with the graph construction methods as we vary the value of the sparsification parameter. As we mentioned before, this analysis is important because a classifier may achieve the best overall classification performance for a very narrow range of the parameter values. Then, this analysis is an invaluable tool to identify which classifiers provide a good trade-off between classification performance and stability;

Influence of graph construction. We also evaluate the graph construction methods combined with the SSL algorithms over a wide range of sparsification parameter values. We want to verify: (1) how the graph construction methods affect the classification performance of each SSL algorithm and (2) the stability of the graph construction methods as we vary the sparsification parameter values. For the classifiers that have at least one regularization parameter, we fixed the regularization parameter(s) with the value that achieved the best average error rate and then varied the sparsification parameter value;

Influence of regularization parameters. We evaluate the error surfaces generated by the SSL algorithms that have regularization parameters. We first chose the sparsification parameter that achieved the best average error rate and then we varied the regularization parameters of the SSL algorithms.

The obtained results show that the mutual k-nearest neighbors (mutKNN) graph may be the best choice for adjacency graph construction while the RBF kernel may be the best choice for weighted matrix generation. In addition, mutKNN tends to generate smoother error surfaces than other adjacency graph construction methods. However, mutKNN is unstable for a relatively small value of k.

Our results indicate that the classification performance of the graph-based SSL algorithms are heavily influenced by *internal* parameters (such as regularization parameters) and *external* parameters (such as the number of neighbors in a k-nearest neighbor graph). Such variability showed no evident explorable pattern to relay to future practitioners. In addition, the SSL assumption that only a very restricted set of labeled examples exists may make parameter estimation techniques commonly used in classification unfeasible.

We believe that our results have two major consequences:

For practitioners. Given a data set, it is difficult to recommend an SSL algorithm, a graph sparsification parameter value or a regularization parameter value that is expected to provide good classification performance. As the number of labeled examples is usually very restricted in SSL applications, the practitioner has no tools to make an informed choice of these parameter values. As we will show, an incorrect choice of the parameter values may seriously affect the classification results;

For researchers. Changes in the parameter values also cause changes in the relative ranking among the classifiers. It means that for a specific data set several methods may figure as the best classifier for a certain range of parameter values. This is a serious issue since the empirical evidence that one method outperforms the competitors might be confirmed only for a restricted set of the parameter values. In addition, this performance variability may hinder the reproduction of the experimental results for papers that do not clearly report every parameter value used in the empirical evaluation.

1.2 Outline

The remainder of this paper is organized as follows. Section 2 describes the notation used throughout the paper and revises the graph construction methods. Section 3 revises the state-of-the-art, graph-based SSL algorithms. Section 4 empirically evaluates the graph construction methods combined with the graph-based SSL algorithms. Finally, Section 5 concludes the paper and suggests directions for future research.

2 Graph Construction

In this section we revise widely used methods to generate sparse weighted graphs, which are frequently considered the heart of graph-based SSL [15]. Section 2.1 describes the notation used throughout the paper. Section 2.2 revises approaches used to generate a sparse undirected[1] graph (or adjacency matrix) from the training sample. Section 2.3 revises approaches used to generate a weighted matrix from the sparse graph.

[1] This paper focus on undirected graphs, which are commonly used in SSL [15].

2.1 Notation and Preliminaries

Consider a training sample $\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^{n} \subset \mathbb{R}^d$ in which the first l examples are labeled, i.e., $\boldsymbol{x}_i$ has label $y_i \in \mathbb{N}_c$ where $\mathbb{N}_p := \{i \in \mathbb{N}^* | 1 \leq i \leq p\}$ with $p \in \mathbb{N}^*$ and c being the number of classes. Let $u := n - l$ be the amount of unlabeled examples and $\mathbf{Y} \in \mathbb{B}^{n \times c}$ be a label matrix in which $\mathbf{Y}_{ij} = 1$ if and only if $\boldsymbol{x}_i$ has label $y_i = j$. Consider an undirected graph $\mathcal{G} := (\mathcal{X}, \mathcal{E})$ in which each $\boldsymbol{x}_i$ is a node of $\mathcal{G}$. Let $\mathcal{N}_i \subset \mathcal{X}$ be the set of neighbors of $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i_k}$ the k-th nearest neighbor of $\boldsymbol{x}_i$. In order to generate a sparse weighted matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ from $\mathcal{G}$ one uses a similarity function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ to compute the weights $\mathbf{W}_{ij}$.

The graph Laplacians are important tools for machine learning. The *combinatorial* Laplacian is defined by $\boldsymbol{\Delta} := \mathbf{D} - \mathbf{W}$ where $\mathbf{D} := \operatorname{diag}(\mathbf{W}\mathbf{1}_n)$ such that $\mathbf{1}_n$ is an n-dimensional 1-entry vector. The *normalized* Laplacian is defined by $\mathbf{L} := \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ where $\mathbf{I}_n$ is the n-by-n identity matrix.

All matrices can be subdivided into labeled and unlabeled submatrices. Let $\mathbf{F} \in \mathbb{R}^{n \times c}$ be the output of a given graph-based SSL algorithm. The $\mathbf{F}$ and $\mathbf{Y}$ matrices are subdivided into two submatrices while all others are subdivided into four submatrices. For instance:

$$\mathbf{W} := \begin{bmatrix} \mathbf{W}_{\mathcal{L}\mathcal{L}} & \mathbf{W}_{\mathcal{L}\mathcal{U}} \\ \mathbf{W}_{\mathcal{U}\mathcal{L}} & \mathbf{W}_{\mathcal{U}\mathcal{U}} \end{bmatrix} \qquad \mathbf{Y} := \begin{bmatrix} \mathbf{Y}_{\mathcal{L}} \\ \mathbf{Y}_{\mathcal{U}} \end{bmatrix}$$

where $\mathbf{W}_{\mathcal{L}\mathcal{L}} \in \mathbb{R}^{l \times l}$ and $\mathbf{Y}_{\mathcal{L}} \in \mathbb{B}^{l \times c}$ are the submatrices of $\mathbf{W}$ and $\mathbf{Y}$, respectively, on labeled examples, and so on. By definition, $\mathbf{Y}_{\mathcal{U}}$ is an $u \times c$ null matrix. This paper focus on the multi-class problem; hence, $\mathbf{Y}_{\mathcal{L}}\mathbf{1}_c = \mathbf{1}_l$.

2.2 Adjacency Graph Construction

The adjacency graph construction process generates a graph $\mathcal{G}$ (or adjacency matrix $\mathbf{A}$) from $\mathcal{X}$ using a distance function $\Psi : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. Let $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$ be a distance matrix in which $\boldsymbol{\Psi}_{ij} := \Psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\mathbf{A} \in \mathbb{B}^{n \times n}$ be an adjacency matrix[2] in which $\mathbf{A}_{ij} = 1$ if and only if $\boldsymbol{x}_j \in \mathcal{N}_i$. We now describe the two most used adjacency graph construction methods for graph-based learning.

ϵ-neighborhood (ϵN). There exists an undirected edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in an ϵN graph if and only if $\Psi(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \epsilon$ where $\epsilon \in \mathbb{R}_+^*$ is a free parameter. In general, ϵN graphs are not widely used in practical situations because they can generate graphs with many disconnected components for an improper value of ϵ. Due to this fact, we did not use the ϵN graph in our experiments.

k-nearest neighbors (kNN). There exists an edge from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ if and only if $\boldsymbol{x}_j$ is one of the k closest examples of $\boldsymbol{x}_i$. Because the adjacency matrix of a kNN graph may not be symmetric, three strategies are commonly used to symmetrize it: *mutual* kNN (mutKNN), which generates $\widehat{\mathbf{A}} = \min\left(\mathbf{A}, \mathbf{A}^\top\right)$; *symmetric* kNN (symKNN), which generates $\widehat{\mathbf{A}} = \max\left(\mathbf{A}, \mathbf{A}^\top\right)$; and *symmetry-favored* kNN (symFKNN) [8], which generates $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{A}^\top$ (a non-binary adjacency matrix).

[2] Non-binary adjacency matrices may also be applied.

2.3 Weighted Matrix Generation

Given an adjacency matrix $\mathbf{A}$, we generate a sparse weighted matrix $\mathbf{W}$ using a similarity function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. We describe three widely used approaches to generate $\mathbf{W}$. Two of them, RBF kernel and similarity function of Hein & Maier [5], define the $\mathbf{W}$ matrix using the relation $\mathbf{W}_{ij} = \mathbf{A}_{ij}\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The third approach, based on local reconstruction minimization [13], generates a sparse weighted matrix $\mathbf{W}$, not necessarily symmetric, without an explicit $\mathcal{K}$.

RBF kernel. The RBF (or Gaussian) kernel computes the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp\left(-\Psi^2(\boldsymbol{x}_i, \boldsymbol{x}_j)/\left(2\sigma^2\right)\right)$ in which $\sigma \in \mathbb{R}_+^*$ is the kernel bandwidth parameter.

Similarity function of Hein & Maier [5] (HM). Given a function $\psi(\cdot, \cdot)$ in which $\psi(\boldsymbol{x}_i, k) := \Psi(\boldsymbol{x}_i, \boldsymbol{x}_{i_k})$ with $k \in \mathbb{N}^*$, the HM similarity function is defined by $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp\left(-\Psi^2(\boldsymbol{x}_i, \boldsymbol{x}_j)/\left(\max\left\{\psi(\boldsymbol{x}_i, k), \psi(\boldsymbol{x}_j, k)\right\}\right)^2\right)$. This is an RBF kernel with an adaptive kernel size.

Local Linear Embedding (LLE). The LLE approach [13] generates the $\mathbf{W}$ matrix by solving the following optimization problem:

$$\min_{\mathbf{W} \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \mathbf{W}_{ij} \boldsymbol{x}_j \right\|_2^2 \quad \text{s.t.} \quad \mathbf{W}\mathbf{1}_n = \mathbf{1}_n, \;\; \mathbf{W} \geq 0 \qquad (1)$$

The symbol $\|\cdot\|_2$ represents the l_2-norm.

3 Label Diffusion

Given a weighted matrix $\mathbf{W}$, a graph-based SSL algorithm uses $\mathbf{W}$ and the label matrix $\mathbf{Y}$ to generate the output matrix $\mathbf{F}$ by label diffusion in the weighted graph. We now revise the state-of-the-art graph-based SSL algorithms used in our empirical comparison. We should note that these algorithms have an intrinsic condition to classify all unlabeled examples in $\mathcal{X}$, which frequently is not explicit in the literature. Assumption 1 describes this condition.

Assumption 1. *Each unlabeled example is on a connected subgraph in which there exists at least one labeled example.*

Gaussian Random Fields (GRF). The GRF algorithm [16] solves the optimization problem $\mathbf{F} = \arg\min_{\mathbf{F} \in \mathbb{R}^{n \times c}} \operatorname{tr}\left(\mathbf{F}^\top \boldsymbol{\Delta} \mathbf{F}\right)$ s.t. $\mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L}$, which gives the closed-form solution $\mathbf{F}_\mathcal{U} = -\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1} \boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}} \mathbf{Y}_\mathcal{L}$.

Local and Global Consistency (LGC). The LGC algorithm [14] solves the optimization problem $\mathbf{F} = \arg\min_{\mathbf{F} \in \mathbb{R}^{n \times c}} \operatorname{tr}\left(\mathbf{F}^\top \mathbf{L} \mathbf{F} + \mu (\mathbf{F} - \mathbf{Y})^\top (\mathbf{F} - \mathbf{Y})\right)$, which gives the closed-form solution $\mathbf{F} = \left(\mathbf{I}_n + \mathbf{L}/\mu\right)^{-1} \mathbf{Y}$.

Laplacian Regularized Least Squares (LapRLS). The LapRLS algorithm [1] minimizes the following regularization framework:

$$\min_{f \in \mathcal{H}_\mathcal{K}} \frac{1}{l} \sum_{i=1}^{l} \mathcal{V}(\boldsymbol{x}_i, y_i, f) + \gamma_A \|f\|_{\mathcal{H}_\mathcal{K}} + \gamma_I \mathbf{f}^\top \boldsymbol{\Delta} \mathbf{f} \qquad (2)$$

where $\mathcal{V}(\boldsymbol{x}_i, y_i, f) = (y_i - f(\boldsymbol{x}_i))^2$, $\mathcal{H}_\mathcal{K}$ is the *Reproducing Kernel Hilbert Space* (*RKHS*) for the kernel $\mathcal{K}$, $\mathbf{f} := [f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)]^\top \in \mathbb{R}^n$, $\|\cdot\|_{\mathcal{H}_\mathcal{K}}$ is the norm in $\mathcal{H}_\mathcal{K}$, and γ_A and γ_I are the regularization parameters. Let $\boldsymbol{y} := [y_1, \cdots, y_l, 0, \cdots, 0] \in \mathbb{R}^n$ be the label vector in which $y_i \in \{-1, +1\}$ and $\mathbf{K} \in \mathbb{R}^{n \times n}$ a gram matrix such that $\mathbf{K}_{ij} := \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Due to the *Representer Theorem* in [1], the solution of (2) can be written as an expansion of kernel functions over both labeled and unlabeled examples, i.e., $f(\boldsymbol{x}) = \sum_{i=1}^n \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i)\boldsymbol{\alpha}_i$ with $\boldsymbol{\alpha} \in \mathbb{R}^n$. Solving (2) using this expansion, we get $\boldsymbol{\alpha} = (\mathbf{JK} + \gamma_A l \mathbf{I}_n + \gamma_I l \boldsymbol{\Delta}\mathbf{K})^{-1} \boldsymbol{y}$ where $\mathbf{J} := \operatorname{diag}\left([1, \cdots, 1, 0, \cdots, 0]^\top\right)$ whose first l diagonal entries are 1 and the rest 0.

Laplacian Support Vector Machine (LapSVM). The LapSVM algorithm [1] minimizes the problem in (2) with $\mathcal{V}(\boldsymbol{x}_i, y_i, f) = \max(0, 1 - y_i f(\boldsymbol{x}_i))$. Solving (2) using the expansion $f(\boldsymbol{x}) = \sum_{i=1}^n \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i)\boldsymbol{\alpha}_i$, we get the solution $\boldsymbol{\alpha} = \frac{1}{2}(\gamma_A \mathbf{I}_n + \gamma_I \boldsymbol{\Delta}\mathbf{K})^{-1} \overline{\mathbf{J}}^\top \overline{\mathbf{Y}} \boldsymbol{\beta}^*$ where $\overline{\mathbf{J}} := [\mathbf{I}_l \ \mathbf{O}_{l \times u}]$ such that $\mathbf{O}_{l \times u}$ is an $l \times u$ null matrix, $\overline{\mathbf{Y}} := \operatorname{diag}\left([y_1, \cdots, y_l]^\top\right)$, and $\boldsymbol{\beta}^* \in \mathbb{R}^l$ is given by

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^l}{\arg\min} \quad \mathbf{1}_l^\top \boldsymbol{\beta} - \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{Q} \boldsymbol{\beta} \quad \text{s.t.} \quad \boldsymbol{y}^\top \boldsymbol{\beta} = 0, \ \ 0 \le \boldsymbol{\beta} \le \frac{1}{l}$$

such that $\mathbf{Q} = \frac{1}{2}\overline{\mathbf{Y}}\,\overline{\mathbf{J}}\mathbf{K}\,(\gamma_A \mathbf{I}_n + \gamma_I \boldsymbol{\Delta}\mathbf{K})^{-1}\,\overline{\mathbf{J}}^\top \overline{\mathbf{Y}}$.

Robust Multi-class Graph Transduction (RMGT). The RMGT algorithm [8] solves the convex optimization problem $\mathbf{F} = \arg\min_{\mathbf{F} \in \mathbb{R}^{n \times c}} \operatorname{tr}\left(\mathbf{F}^\top \boldsymbol{\Delta} \mathbf{F}\right)$ s.t. $\mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L}$, $\mathbf{F}\mathbf{1}_c = \mathbf{1}_n$, $\mathbf{F}^\top \mathbf{1}_n = n\boldsymbol{\omega}$ where $\boldsymbol{\omega} \in \mathbb{R}^c$ is the class prior probabilities. The solution of this optimization problem is given by:

$$\mathbf{F}_\mathcal{U} = -\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1} \boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}} \mathbf{Y}_\mathcal{L} + \frac{\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1} \mathbf{1}_u}{\mathbf{1}_u^\top \boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1} \mathbf{1}_u} \left(n\boldsymbol{\omega}^\top - \mathbf{1}_l^\top \mathbf{Y}_\mathcal{L} + \mathbf{1}_u^\top \boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1} \boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}} \mathbf{Y}_\mathcal{L}\right)$$

4 Experimental Evaluation

In this section we provide a detailed empirical comparison of the graph-based SSL algorithms described in Section 3 combined with the graph construction methods described in Section 2 on a number of benchmark data sets. The objective of these experiments is to evaluate the influence that graph construction methods have in the classifiers' performance. We performed experiments in a transductive setting using different sets of labeled and unlabeled examples in each execution.

For a fair comparison and ease of reproducibility, we used the source code of the authors of the algorithms when possible. As some authors implemented their methods in Matlab, we used the matlabcontrol[3] library to link the Matlab code and Java. Due to reasons concerning reproducibility, all source codes and data sets used in our experiments are freely available[4].

[3] `https://code.google.com/p/matlabcontrol/downloads/list`

[4] `http://www.icmc.usp.br/~gbatista/ECML2013`

4.1 Data Sets

We used in our experiments the USPS, COIL_2, DIGIT-1, G-241C, G-241N, and TEXT data sets. These data sets are freely available[5] and very popular in the SSL literature [3]. USPS and DIGIT-1 are data sets for digit recognition, TEXT is a data set for text classification, G-241N and G-241C are data sets for classification of Gaussian distributions, and COIL_2 is a data set for image classification. We used the data splits of 10 labeled examples suggested in [3].

We run *principal component analysis* (PCA) to reduce the dimensionality of the data sets. In high-dimensional data, the distance to the nearest neighbor approaches the distance of the farthest neighbor [2]. It degenerates the quality of the graph and possibly decreases the classification performance of the SSL algorithms. After some preliminary experimental evaluation, we decided to reduce the dimensionality of the data to 50 features using the *Matlab Toolbox for Dimensionality Reduction*[6] library. We did not run PCA only on the TEXT data set to maintain the sparseness property of these data.

4.2 Empirical Setup

In this section, we describe the experimental design decisions that we have taken in our experiments in order to facilitate the reproduction of our results.

Distance functions. Due to its high popularity in the text classification literature, we used the cosine distance in the experiments using the TEXT data set. The cosine distance is defined as $\Psi(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_d / (\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2)$ where $\langle \cdot, \cdot \rangle_d$ is the inner product of vectors in $\mathbb{R}^d$. For all other data sets we used the l_2 norm as a distance function.

Graph Laplacians. Since the normalized Laplacian $\mathbf{L}$ may lead to better empirical results in comparison with the combinatorial Laplacian $\boldsymbol{\Delta}$ [7], we used $\mathbf{L}$ instead of $\boldsymbol{\Delta}$ in the formulation of the graph-based SSL algorithms. We obtained poor results using $\mathbf{L}$ in the RMGT algorithm during preliminary experiments; therefore, we report the results of RMGT using $\boldsymbol{\Delta}$. In preliminary experiments, we observed some errors using RMGT in the COIL_2 data set. These errors occurred because at least one of the eigenvalues of the graph Laplacian was equal to (or approximately) zero. In an attempt to avoid numerical instabilities while solving linear systems using the graph Laplacians, we generated the combinatorial Laplacian as $\boldsymbol{\Delta} = \gamma \mathbf{D} - \mathbf{W}$ and the normalized Laplacian as $\mathbf{L} = \gamma \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}$ where a small $\gamma > 1$ is used to increase the eigenvalues of the graph Laplacians. In our experiments, we set $\gamma = 1.01$.

Mutual kNN. The procedure $\widehat{\mathbf{A}} = \min\left(\mathbf{A}, \mathbf{A}^\top\right)$ may generate a graph with isolated vertices. It may degenerate the output of the SSL algorithms because the label diffusion process could not be effective. In an attempt to avoid this

[5] `http://olivier.chapelle.cc/ssl-book/benchmarks.html`.

[6] `http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`.

problem, we created an undirected edge between each isolated vertex and its nearest neighbor. Other strategies may also be applied as well [12].

LLE. We used the *Local Anchor Embedding* (LAE) method [9][7] to solve the optimization problem in (1). LLE is an example of LAE if we generate a bipartite graph whose "anchor" points are exactly the training examples. Since LLE may not generate a symmetric weighted matrix, we symmetrize the output matrix of LLE, $\mathbf{W}_{LLE}$, as $\mathbf{W} = \frac{1}{2}\left(\mathbf{W}_{LLE} + \mathbf{W}_{LLE}^{\top}\right)$.

SymFKNN + LLE. Because the adjacency matrix of the symFKNN graph is non-binary, we compute $\widehat{\mathbf{W}} = \mathbf{W}_{LLE} \odot \mathbf{A}$ where $\odot$ is the Hadamard product. Then, we generate $\mathbf{W} = \frac{1}{2}\left(\widehat{\mathbf{W}} + \widehat{\mathbf{W}}^{\top}\right)$.

LapSVM. We run LapSVM using the source code in [11][8]. We trained LapSVM using Newton's method, which gave better results than the preconditioned conjugate gradient method during preliminary experiments.

LapRLS. We used the multi-class version of LapRLS; hence, we compute $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha} = \left(\mathbf{JK} + \gamma_A l \mathbf{I}_n + \gamma_I l \boldsymbol{\Delta} \mathbf{K}\right)^{-1} \mathbf{Y}$ and get the output matrix $\mathbf{F} = \mathbf{K}\boldsymbol{\alpha}$.

Classification. In order to classify the unlabeled examples, we used the *class mass normalization* (CMN) procedure [16]. This is an useful procedure when we are dealing with data sets with imbalanced labels. We obtained poor results using CMN in RMGT; therefore, we report the results for RMGT using the *argmax* operator. We report the results for GRF, LGC, and LapRLS using CMN while the results for LapSVM are reported using the *sign* function. For GRF, we computed CMN using $\mathbf{F}_{\mathcal{U}}$ instead of $\mathbf{F}$, as suggested in [16].

4.3 Parameter Setting

We now describe the parameter setting used in our experimental evaluation.

SymKNN, mutKNN, and symFKNN. The sparsification parameter k was chosen at the range $\{4, 6, 8, \cdots, 40\}$.

RBF kernel. Because it is not straightforward to find an adequate value for the kernel bandwidth σ when labeled examples are scarce, we estimate its value by $\sigma = \sum_{i=1}^{n} \Psi(\boldsymbol{x}_i, \boldsymbol{x}_{i_k})/(3n)$, as suggested in [6].

Gram matrix. We generated the gram matrix $\mathbf{K}$ using the RBF kernel. We used the same distance function $\Psi(\cdot, \cdot)$, the sparsification parameter k, and the kernel bandwidth σ used during graph construction to compute $\mathbf{K}$.

LGC. The regularization parameter μ in the LGC framework was chosen at range $\{0.01, 0.05, 0.1, 0.5, 1, 2, 5, 10, 50, 100\}$.

LapRLS and LapSVM. The regularization parameters γ_A and γ_I were chosen at range $\left\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10, 100\right\}$, as suggested in [11]. All other parameters were set to their default values.

RMGT. For the RMGT algorithm, we assumed a uniform class distribution, i.e., we set $\boldsymbol{\omega} = \mathbf{1}_c/c$ instead of using the class prior probabilities, as suggested in [8]. We achieved better results in preliminary experiments using

[7] `http://www.ee.columbia.edu/ln/dvmm/downloads/WeiGraphConstructCode/dlform.htm`.

[8] `http://www.dii.unisi.it/~melacci/lapsvmp/index.html`.

the uniform class distribution in most data sets; therefore, we report the results for RMGT using this setting for all data sets, excluding USPS. For the USPS data set, we used the class prior probabilities, which achieved the best results.

4.4 Analysis of the Results

In this section we analyze the obtained results. Our empirical analysis is subdivided into four parts: (1) best cases analysis; (2) graph-based SSL algorithm comparison; (3) influence of graph construction on SSL; and (4) influence of regularization parameters on the classifiers' performance.

Best case analysis. Table 1 shows the obtained results for the best case analysis. Each numerical result in this table is the lowest average error rate obtained by a combination of an SSL algorithm, a graph construction method and a data set for all parameter values (sparsification and regularization, if applicable), as described in Section 4.3. The four worst results obtained by an SSL algorithm in each data set have a grey background while the best one is in bold. The best overall result for each data set is boxed.

We can see in Table 1 that the symKNN-LLE and symFKNN-LLE graphs may not be adequate for GRF, LGC, and LapRLS because they achieved unsatisfactory results in all data sets. We also see that mutKNN outperformed the symKNN and symFKNN graphs in most situations, independent of the weighted matrix generation method or the SSL algorithm used. Therefore, for the data sets considered in this study, mutKNN presented the best performance among all adjacency graph construction methods.

We ran the Friedman's test[9] with Nemenyi's post test using a confidence level of 0.05 to statistically compare the performance of the graph construction methods. Table 2 shows the average rankings. The best rankings are marked in bold face and the results that were outperformed by the best ranked method are marked with grey background. We can see that symFKNN-RBF and mutKNN-RBF obtained the best rankings for most SSL algorithms. However, the statistical test found significant differences for only 7 cases.

After analyzing the classifiers, we see that RMGT achieved the best overall classification performance in 4 out of 6 data sets. Although RMGT achieved satisfactory results on most data sets, it did not perform well on the USPS data set.

Classifiers' stability evaluation. As we mentioned earlier, the best case analysis does not allow us to investigate the stability of the classifiers. In this analysis, we investigate the stability of the SSL algorithms as we vary the graph sparsification parameter value. Due to space restrictions and because the mutKNN-RBF graph achieved the best overall classification performance in the best case analysis, we show here only the results obtained with the mutKNN-RBF graph. The interested reader will find the results for other graph construction methods on the paper's website.

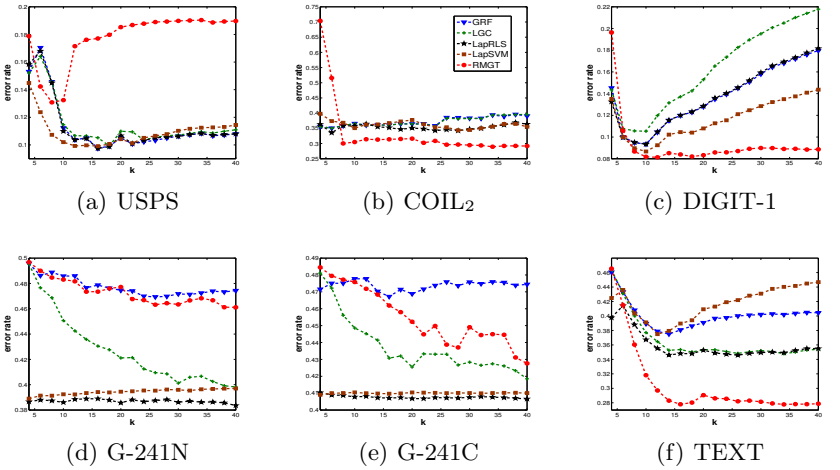
[9] See [4] and references therein for a review on statistical tests for machine learning.

Table 1. Average error rates and standard deviations of the SSL algorithms for each graph construction method and data set

Data sets	USPS	$COIL_2$	DIGIT-1	G-241N	G-241C	TEXT
GRF-symKNN-RBF	11.07 (3.33)	35.13 (6.92)	10.19 (4.27)	**46.12 (7.61)**	46.28 (6.98)	39.15 (5.69)
GRF-mutKNN-RBF	**9.75 (4.50)**	**35.07 (3.82)**	**9.35 (4.51)**	46.94 (4.81)	46.72 (4.85)	**37.51 (6.85)**
GRF-symFKNN-RBF	10.75 (3.77)	35.22 (6.92)	10.01 (3.93)	**46.12 (7.65)**	46.34 (6.83)	38.50 (5.87)
GRF-symKNN-HM	15.53 (2.76)	38.55 (6.06)	10.73 (4.27)	46.86 (5.28)	**46.19 (7.25)**	42.32 (8.54)
GRF-mutKNN-HM	11.01 (3.59)	35.30 (3.87)	10.02 (6.36)	46.66 (6.18)	46.58 (5.06)	41.18 (9.87)
GRF-symFKNN-HM	15.17 (3.09)	37.77 (6.25)	10.24 (4.36)	46.77 (5.38)	46.27 (7.05)	42.20 (8.62)
GRF-symKNN-LLE	16.03 (2.47)	36.04 (5.60)	10.94 (4.69)	47.54 (3.77)	47.33 (4.77)	43.56 (6.96)
GRF-mutKNN-LLE	11.64 (3.39)	35.20 (3.79)	10.30 (5.89)	47.14 (2.87)	46.98 (3.64)	42.34 (6.48)
GRF-symFKNN-LLE	15.55 (2.74)	36.10 (5.88)	10.31 (4.68)	47.25 (4.14)	47.46 (4.36)	43.54 (6.95)
LGC-symKNN-RBF	11.22 (3.07)	34.96 (6.69)	10.68 (4.91)	38.06 (6.91)	40.24 (5.13)	35.42 (5.58)
LGC-mutKNN-RBF	**9.93 (4.34)**	35.07 (3.82)	10.54 (5.21)	39.82 (5.36)	41.85 (4.32)	**34.78 (6.55)**
LGC-symFKNN-RBF	10.97 (3.00)	**34.81 (6.22)**	**10.47 (4.66)**	**37.95 (6.66)**	**40.10 (5.46)**	35.51 (5.64)
LGC-symKNN-HM	14.49 (5.25)	37.20 (7.32)	11.53 (5.00)	38.36 (6.83)	40.27 (4.48)	37.51 (4.48)
LGC-mutKNN-HM	10.79 (3.75)	35.19 (4.90)	10.96 (5.34)	39.51 (5.80)	41.94 (4.20)	36.01 (5.63)
LGC-symFKNN-HM	14.63 (3.33)	36.36 (8.25)	11.07 (4.76)	38.13 (7.11)	40.17 (4.75)	37.49 (4.35)
LGC-symKNN-LLE	15.05 (4.33)	35.95 (6.09)	11.49 (5.41)	41.22 (4.12)	43.33 (3.03)	39.18 (4.02)
LGC-mutKNN-LLE	11.04 (3.82)	35.18 (3.77)	10.96 (6.34)	42.12 (3.90)	42.94 (3.09)	35.89 (9.20)
LGC-symFKNN-LLE	14.51 (2.81)	35.98 (6.07)	10.97 (5.01)	41.24 (4.37)	43.06 (3.34)	39.03 (3.82)
LapRLS-symKNN-RBF	10.99 (3.05)	34.92 (5.98)	10.22 (4.25)	38.09 (6.76)	40.35 (6.23)	35.12 (5.68)
LapRLS-mutKNN-RBF	**9.75 (4.53)**	33.56 (7.32)	**9.33 (4.48)**	38.36 (5.96)	40.66 (5.45)	**34.58 (6.14)**
LapRLS-symFKNN-RBF	10.57 (2.90)	35.50 (5.84)	10.02 (3.92)	38.08 (6.64)	40.36 (6.02)	35.34 (5.73)
LapRLS-symKNN-HM	14.56 (3.89)	37.58 (5.91)	10.76 (4.24)	38.18 (6.70)	40.24 (6.07)	37.12 (4.52)
LapRLS-mutKNN-HM	10.57 (4.66)	32.80 (7.67)	9.92 (5.50)	38.29 (6.00)	40.67 (5.50)	35.90 (5.61)
LapRLS-symFKNN-HM	14.38 (4.14)	36.93 (4.95)	10.28 (4.32)	**38.06 (6.52)**	**40.11 (6.06)**	37.32 (4.38)
LapRLS-symKNN-LLE	14.73 (3.24)	36.85 (5.25)	10.93 (4.66)	38.68 (5.60)	40.61 (5.51)	38.49 (4.00)
LapRLS-mutKNN-LLE	11.28 (4.09)	**31.78 (7.81)**	10.19 (5.92)	38.66 (5.72)	40.59 (5.61)	37.28 (5.26)
LapRLS-symFKNN-LLE	14.55 (3.37)	36.17 (4.81)	10.31 (4.63)	38.69 (5.56)	40.61 (5.52)	38.62 (4.13)
LapSVM-symKNN-RBF	11.42 (4.03)	34.96 (6.81)	9.42 (3.97)	39.16 (6.07)	40.91 (6.08)	39.88 (6.02)
LapSVM-mutKNN-RBF	**9.91 (2.51)**	34.37 (6.47)	**8.67 (3.89)**	**38.90 (6.50)**	40.90 (6.08)	**37.49 (7.07)**
LapSVM-symFKNN-RBF	11.04 (3.43)	34.04 (6.92)	9.47 (4.19)	39.16 (6.07)	40.91 (6.08)	39.45 (6.30)
LapSVM-symKNN-HM	14.63 (5.47)	36.40 (4.07)	10.13 (3.65)	39.15 (6.04)	40.91 (6.08)	43.06 (4.99)
LapSVM-mutKNN-HM	10.04 (2.83)	33.08 (6.35)	9.58 (4.73)	39.00 (6.43)	40.90 (6.08)	42.10 (6.03)
LapSVM-symFKNN-HM	14.35 (4.29)	36.57 (3.57)	9.93 (3.95)	39.14 (6.05)	40.91 (6.08)	42.71 (5.15)
LapSVM-symKNN-LLE	14.82 (3.38)	35.39 (4.80)	10.31 (4.11)	39.12 (6.43)	40.90 (6.07)	42.77 (6.20)
LapSVM-mutKNN-LLE	10.61 (2.49)	**31.54 (6.24)**	10.22 (5.52)	38.95 (6.46)	**40.82 (6.66)**	41.80 (7.65)
LapSVM-symFKNN-LLE	14.41 (3.23)	35.21 (4.58)	9.83 (3.99)	39.00 (6.40)	40.84 (6.40)	42.62 (4.92)
RMGT-symKNN-RBF	16.62 (2.90)	31.05 (4.81)	8.63 (3.35)	44.99 (6.97)	38.44 (6.22)	30.43 (6.26)
RMGT-mutKNN-RBF	**13.08 (3.41)**	28.95 (3.88)	8.13 (3.14)	46.11 (4.50)	42.76 (6.11)	**27.77 (5.95)**
RMGT-symFKNN-RBF	16.02 (2.85)	32.94 (4.20)	8.55 (3.36)	45.25 (6.07)	**38.31 (6.02)**	29.65 (6.46)
RMGT-symKNN-HM	19.08 (1.22)	31.20 (6.14)	8.07 (2.69)	44.31 (9.03)	38.48 (6.91)	34.86 (6.04)
RMGT-mutKNN-HM	16.99 (2.45)	**28.00 (4.67)**	**7.50 (2.43)**	44.73 (5.48)	40.53 (4.37)	31.12 (6.35)
RMGT-symFKNN-HM	18.88 (2.26)	30.56 (5.52)	7.92 (2.58)	44.68 (7.89)	38.48 (6.67)	34.61 (6.25)
RMGT-symKNN-LLE	19.04 (1.19)	30.63 (3.94)	7.91 (2.49)	42.83 (6.00)	42.25 (3.32)	36.61 (4.79)
RMGT-mutKNN-LLE	17.85 (1.95)	29.49 (4.16)	7.53 (2.11)	43.75 (6.40)	42.12 (4.08)	33.89 (5.32)
RMGT-symFKNN-LLE	18.97 (1.18)	30.41 (3.71)	7.73 (2.43)	**42.75 (7.33)**	41.77 (3.33)	36.25 (4.78)

Table 2. Average rankings of the graph construction methods for each SSL algorithm

	GRF	LGC	LapRLS	LapSVM	RMGT	mean
symKNN-RBF	2.9167	2.8333	3.5	5.1667	5.1667	3.9167
mutKNN-RBF	2.6667	3	**3.1667**	**2**	4.8333	3.1333
symFKNN-RBF	**2.5833**	**1.6667**	3.25	4.6667	5	3.4333
symKNN-HM	6	6.5	6	7.75	6.25	6.5
mutKNN-HM	3.8333	4.5833	4.25	3.4167	**3.5**	3.9167
symFKNN-HM	5	5.6667	4.8333	6.9167	5.0833	5.5
symKNN-LLE	8.3333	8	7.9167	7	6.1667	7.4833
mutKNN-LLE	5.8333	5.4167	4.6667	3.1667	4.1667	4.65
symFKNN-LLE	7.8333	7.3333	7.4167	4.9167	4.8333	6.4667



(a) USPS (b) $COIL_2$ (c) DIGIT-1

(d) G-241N (e) G-241C (f) TEXT

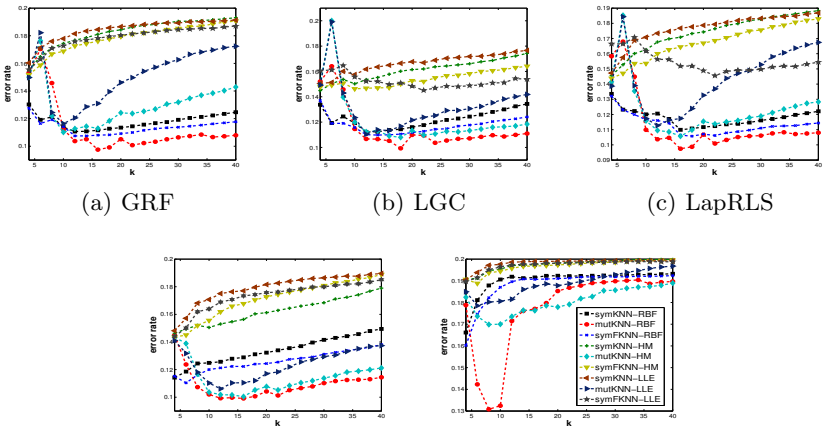
Fig. 1. Average error rates of the SSL algorithms using the mutKNN-RBF graph

Fig. 1 shows the results for this empirical analysis using the mutKNN-RBF graph as we vary the sparsification parameter value. Notice that the legend for all graphics in Fig. 1 can be found in Fig. 1(b). The RMGT algorithm achieved good classification performance and stability on the $COIL_2$, TEXT, and DIGIT-1 data sets when $k \geq 14$. However, RMGT was generally the worst classifier for the USPS data set and the second worst for the G-241N and G-241C data sets. Moreover, RMGT appears to be unstable for relatively small values of k. For instance, the instability of RMGT is evidenced in the $COIL_2$ data set for $k \leq 6$ while all other classifiers achieved satisfactory results with this setting.

LapRLS and LapSVM achieved exceptional stability on the G-241C and G-241N data sets. Due to this high stability, we suppose that LapRLS and LapSVM may be the best SSL algorithms for classification of Gaussian distributions. We also note in Fig. 1 that the assumption that sparse graphs give better results than dense graphs may not necessarily be true. For instance, the results for the GRF, LGC, and RMGT algorithms on the G-241C and G-241N data sets using dense graphs are better than those for sparse graphs. In addition, the results for all SSL algorithms on the TEXT data set for relatively small values of k are not satisfactory while the results for the LGC, LapRLS, and RMGT algorithms with dense graphs are.

Influence of graph construction. We now evaluate how different graphs can influence the classification performance of the SSL algorithms. Once again, we perform this analysis as we vary the sparsification parameter value in order to analyze the stability of the graph construction methods combined with the SSL algorithms. Due to lack of space, we only present the plots for the USPS data set in Fig. 2. Once again, we invite the interested reader to check the paper's website. It is clear from Fig. 2 that the results show

a lot of variability. For a given classifier, we can observe that several graph construction methods figure among the best and the worst method as we vary the value of k. The variability problem is more intense for small values of k, specially $k \leq 14$. This seems to be a permissive problem since small values of k performed better for this specific data set, but too small values might greatly degrade the classifiers' performance.



(a) GRF (b) LGC (c) LapRLS

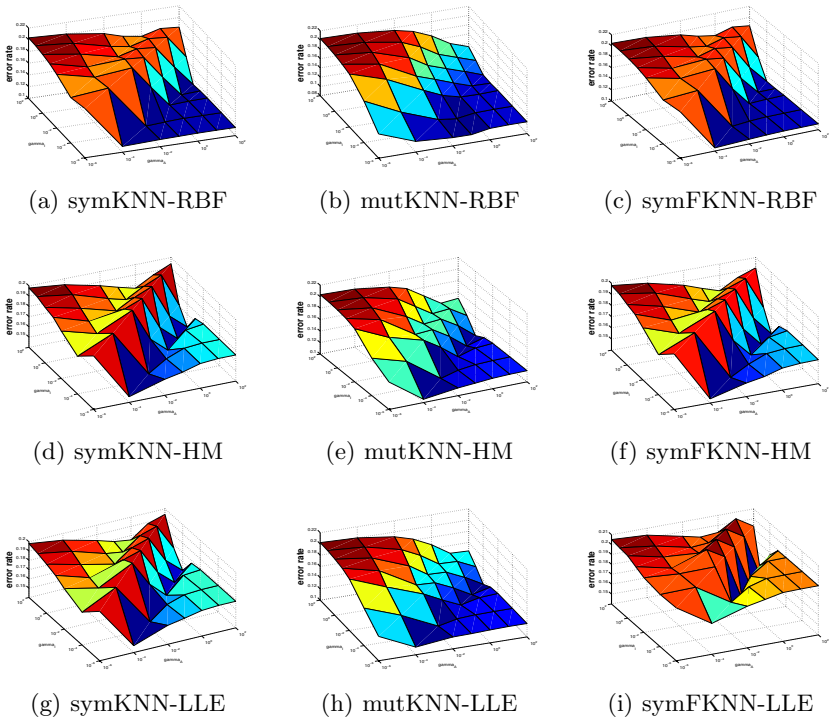
(d) LapSVM (e) RMGT

Fig. 2. Average error rates of the graph construction methods on the USPS data set

The USPS data set is an excellent example of the high influence of the graph construction methods and the sparsification parameter values over the SSL algorithms. As we vary the k parameter, the classifiers' performance vary significantly; some of them in a range of almost 10%. Such performance variation is certainly a concern for the practitioner, who would have difficulties in finding a parameter setting that guarantees a good classification performance. Moreover, such high variability causes several changes in the relative rankings of the classifiers. In some cases, the same classifier might figure among the best and the worst methods as we vary the k parameter in the narrow range of $[4, 14]$. These changes of relative order may cause some serious concerns for the research community. Without an extensive analysis of the influence of parameter values, some studies may experimentally show that a proposed algorithm outperforms the state-of-the-art algorithms, being that this conclusion only holds for certain parameter values. We are not claiming here that such an incident has ever happened, and we have not observed any such evidence, however; it is certainly undesirable for the research community to be affected of such a situation.

We suggest that every research paper that proposes a new SSL algorithm or graph construction method to fully analyze the influence of its parameters. The experimental setup used in this paper is a proposal of how newly

proposed methods should be evaluated. It is important to evaluate the algorithms' performance for a wide range of *external* parameters, such as k, and graph construction methods. Some algorithms also have *internal* parameters, such as regularization parameters, that also need to be evaluated, as we show next.



(a) symKNN-RBF (b) mutKNN-RBF (c) symFKNN-RBF

(d) symKNN-HM (e) mutKNN-HM (f) symFKNN-HM

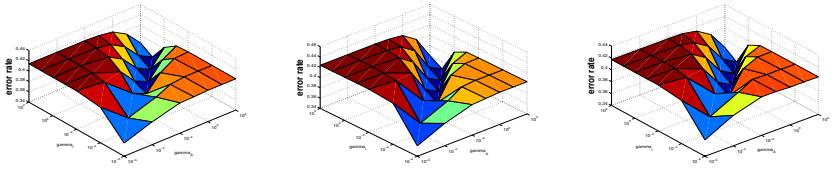
(g) symKNN-LLE (h) mutKNN-LLE (i) symFKNN-LLE

Fig. 3. Error surfaces for the LapRLS algorithm on the USPS data set

Influence of regularization parameters. We evaluate the influence of regularization parameters on the classification performance of the graph-based SSL algorithms. We evaluate the error surfaces generated by the SSL algorithms for each graph construction method and data set. Due to lack of space, we only show the most relevant results for LapRLS and LapSVM. We fixed the value of k that achieved the best error rate for each combination of SSL algorithm and graph construction method. In the sequence, we varied the values of γ_A and γ_I, as described in Section 4.3.

Fig. 3 shows the results for LapRLS on the USPS data set for different graph construction methods. We see that mutKNN generated smoother error surfaces than symKNN and symFKNN graphs, independent of the weighted matrix generation method used.

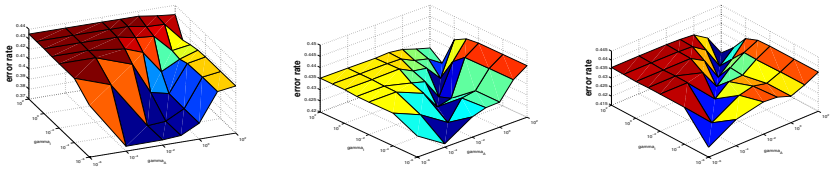
Many of the obtained results for this analysis are qualitatively equivalent fixing an SSL algorithm and a data set. However, we found some specific results that have an explicit pattern for parameters choice and others which may not have any evident pattern. We discuss them in the following.



(a) symKNN-RBF (b) mutKNN-RBF (c) symFKNN-RBF

Fig. 4. Error surfaces for LapRLS on the TEXT data set using the RBF kernel

Fig. 4 shows the obtained results for LapRLS on the TEXT data set using the RBF kernel combined with the adjacency graph construction methods. We see that the "optimal region" occurs only when $\gamma_A = \gamma_I$.



(a) mutKNN-RBF (b) mutKNN-HM (c) mutKNN-LLE

Fig. 5. Error surfaces for LapSVM on the TEXT data set using the mutKNN graph

Fig. 5 shows the obtained results for LapSVM on the TEXT data set using the mutKNN graph combined with the weighted matrix generation methods. We can not see any evident pattern that could help parameter choice. This may be an obstacle to apply LapSVM on real applications on text classification. For instance, the "optimal region" for the mutKNN-LLE graph occurs when $\gamma_A = \gamma_I$, which is not a good setting for the other graphs.

5 Conclusions and Further Research

In this paper, we provided a detailed empirical comparison of five state-of-the-art, graph-based SSL algorithms combined with three adjacency graph construction methods and three weighted matrix generation methods. Our experimental evaluation indicated that the SSL algorithms are strongly affected by the graph sparsification parameter value and the choice of the adjacency graph

construction and weighted matrix generation methods. The algorithms that have regularization parameters were also very dependent on a good setting of these parameters.

Consequently, we proposed an experimental setup that should be used in empirical comparisons in future work in SSL. Our idea is that a newly proposed algorithm should not be compared to other state-of-the-art algorithms using only the best case analysis. We believe that a detailed evaluation of all parameters is necessary. Due to the nature of SSL, in which there exists only a limited number of labeled examples, tuning all parameters might be unfeasible. Therefore, there is a need for algorithms that are slightly dependable on parameter tuning, i.e., that have a stable performance over the parameter space.

Our experimental results showed a superiority of mutKNN over the symKNN and symFKNN graphs. However, our results also showed that mutKNN is unstable for a relatively small value of k. In addition, we showed that mutKNN tends to generate smoother error surfaces than symKNN and symFKNN graphs. Our experiments also indicated a superiority of the RBF kernel in comparison to the HM and LLE methods.

As we analyzed our experimental results, we noticed other interesting patterns that we could not verify given the lack of experimental evidence. We propose an investigation concerning the validity of these observations as future research. Our empirical observations are as follows:

- Although RMGT achieved satisfactory results on most data sets, it did not perform well on the USPS data set. As USPS is an imbalanced dataset, a possible explanation is that RMGT is not effective on data sets with imbalanced labels;
- Maier et al. [10] have pointed out that the mutKNN graph should be chosen if one is only interested in identifying the "most significant" cluster. Based on this statement, we suppose that mutKNN is the best graph when we are dealing with data sets with imbalanced labels because it may identify the "most significant" class (the minority class in this case). This hypothesis is supported by the fact that, in Table 1, mutKNN achieved better classification performance than symKNN and symFKNN for all combinations of SSL algorithm and weighted matrix generation method on the USPS data set;
- Table 1 shows that RMGT achieved the best overall classification performance in 4 out of 6 data sets. This surprising classification performance may be due to the addition of the normalization constraints $\mathbf{F}\mathbf{1}_c = \mathbf{1}_n$ and $\mathbf{F}^\top \mathbf{1}_n = n\boldsymbol{\omega}$ in the optimization framework. It would be interesting to investigate if other SSL algorithms' classification performances could be improved if these constraints were included in their optimization framework;
- In Fig. 4, we observed that the "optimal region" occurs only when $\gamma_A = \gamma_I$. Since this behavior occurred for all graphs (the other results are not shown here due to lack of space), we ask if this setting should be chosen for text classification tasks when using LapRLS.

Acknowledgments. This research was supported by the Brazilian agencies CAPES and FAPESP. Thanks to Diego F. Silva, Vinícius M. A. de Souza, Rafael Giusti, Ricardo M. Marcacini, and Rafael G. Rossi for their help in the experiments.

References

1. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. JMLR 7, 2399–2434 (2006)
2. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
3. Chapelle, O., Schölkopf, B., Zien, A.: Semi-supervised learning. The MIT Press (2006)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR 7, 1–30 (2006)
5. Hein, M., Maier, M.: Manifold denoising. In: NIPS 19, pp. 561–568 (2007)
6. Jebara, T., Wang, J., Chang, S.F.: Graph construction and b-matching for semi-supervised learning. In: ICML, pp. 441–448 (2009)
7. Johnson, R., Zhang, T.: On the effectiveness of laplacian normalization for graph semi-supervised learning. JMLR 8, 1489–1517 (2007)
8. Liu, W., Chang, S.F.: Robust multi-class transductive learning with graphs. In: CVPR, pp. 381–388 (2009)
9. Liu, W., He, J., Chang, S.F.: Large graph construction for scalable semi-supervised learning. In: ICML, pp. 679–686 (2010)
10. Maier, M., Hein, M., von Luxburg, U.: Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. Theoretical Computer Science 410(19), 1749–1764 (2009)
11. Melacci, S., Belkin, M.: Laplacian support vector machines trained in the primal. JMLR 12, 1149–1184 (2011)
12. Ozaki, K., Shimbo, M., Komachi, M., Matsumoto, Y.: Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data. In: CoNLL, pp. 154–162 (2011)
13. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
14. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS 16, pp. 321–328 (2004)
15. Zhu, X.: Semi-supervised learning literature survey. Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison (2005)
16. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML, pp. 912–919 (2003)