# Joint Learning of the Graph and the Data Representation for Graph-Based Semi-Supervised Learning

Mariana Vargas-Vieyra[1], Aurélien Bellet[2], and Pascal Denis[3]

[1,2,3]Magnet Team, Inria Lille - Nord Europe
[1]Université de Lille, CNRS, UMR 9189 - CRIStAL
59650 Villeneuve d'Ascq, France
first-last@inria.fr

## Abstract

Graph-based semi-supervised learning is appealing when labels are scarce but large amounts of unlabeled data are available. These methods typically use a heuristic strategy to construct the graph based on some fixed data representation, independently of the available labels. In this paper, we propose to jointly learn a data representation and a graph from both labeled and unlabeled data such that (i) the learned representation indirectly encodes the label information injected into the graph, and (ii) the graph provides a smooth topology with respect to the transformed data. Plugging the resulting graph and representation into existing graph-based semi-supervised learning algorithms like label spreading and graph convolutional networks, we show that our approach outperforms standard graph construction methods on both synthetic data and real datasets.

## 1 Introduction

An important bottleneck for the development of accurate Natural Language Processing (NLP) tools for many applications and languages is the lack of annotated data. A natural remedy to this issue lies in semi-supervised learning (SSL) methods, which are able to leverage smaller labeled datasets in combination with large amounts of unannotated text data (which are often more easily available). In particular, graph-based SSL algorithms (Subramanya and Talukdar, 2014), among which variants of label propagation (Zhu and Ghahramani, 2002; Zhu et al., 2003; Zhou et al., 2004) and more recently graph convolutional networks (Kipf and Welling, 2017; Chen et al., 2018; Wu et al., 2019), have attracted a lot of attention due to their interesting theoretical properties and good empirical performance. They have successfully been applied to several NLP problems, such as sentiment analysis (Goldberg and Zhu, 2006), word sense disambiguation (Alexandrescu and Kirchhoff, 2007), text categorization (Subramanya and Bilmes, 2008), POS tagging (Subramanya et al., 2010), semantic parsing (Das and Smith, 2011), machine translation (Saluja et al., 2014), or lexicon induction (Faruqui et al., 2016). As their name suggests, graph-based SSL methods represent all data points (that is, labeled and unlabeled) as nodes in a graph with weighted edges encoding the similarity between pairs of points. This graph is then used as a propagation operator to transfer labels from labeled to unlabeled points. Despite differences in the way this propagation is achieved, graph-based SSL approaches all rely on two assumptions: (i) the graph representing the data provides a faithful approximation of the manifold on which the data actually live, and (ii) the underlying labels are smooth with respect to this manifold.

In many NLP problems, there is often no *a priori*-known graph, which raises the question of how to best construct this graph over the dataset given some data representation. Most existing work rely on classic graph construction heuristics such as $k$-nn graphs, $\epsilon$-graphs or radial kernel graphs (Subramanya and Talukdar, 2014), which may poorly adapt to the intrinsic structure of the data manifold and hence violate assumption (i). More recently, in the machine learning and signal processing communities, some algorithms were proposed to learn more flexible graphs by enforcing the topology to be smooth with respect to the input data (Daitch et al., 2009; Kalofolias, 2016; Dong et al., 2016). All these approaches





*Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 35–45
Barcelona, Spain (Online), December 13, 2020

heavily depend on the choice of data representation and disregard the label information, making them unable to adapt to the prediction task and therefore potentially violating assumption (ii). While supervised representation learning techniques such as metric learning (Bellet et al., 2015) could be used to adapt the representation to the task of interest, for instance by bringing closer points with the same label, the lack of labeled data in the semi-supervised learning scenario makes them prone to overfitting.

In this paper, we propose an original semi-supervised algorithm for graph construction that adapts to both the data and the predictive task. Specifically, our approach leverages the labeled and unlabeled data to jointly learn a graph and a data representation. On the one hand, the graph is learned to provide a smooth topology with respect to the learned representation. On the other hand, the representation should bring closer (labeled and unlabeled) points that are neighbors in the graph as well as similarly labeled points, while pulling away points of different labels. A key feature of our approach is that the learned representation indirectly encodes and injects label information into the graph beyond the labeled points alone. We formulate our problem as a joint optimization problem over the representation and the graph weights, with a hyperparameter to easily control the sparsity of the resulting graph and thereby obtain a good approximation of the underlying manifold. We discuss some appropriate parameterizations for learning the representation, which revolve around adapting pre-trained embeddings so as to avoid overfitting. We then propose to solve our joint problem by alternating optimization on the representation and the graph. We validate our approach through several graph-based SSL experiments using label spreading (Zhou et al., 2004) and graph convolutional networks (GCN) (Kipf and Welling, 2017), both on synthetic and real text classification datasets. Incidentally, note that our approach is generic and could in principle be used in combination with any existing graph-based SSL framework. The results show that our approach outperforms previous methods which rely on heuristic graphs, generally by a considerable margin. Interestingly, we also observe that our approach effectively bridges the accuracy gap between a simple method like label spreading and a richer neural-based approach like GCN.

The rest of this paper is organized as follows. We introduce some notations and discuss the related work in Section 2. We then describe our approach and algorithm in Section 3. Our experimental results are presented in Section 4, and we conclude with future work directions in Section 5.

## 2 Notations and Related Work

Our work lies in the intersection of two topics: graph-based semi-supervised learning and graph learning. In this section, we start by introducing some useful notations. We then briefly review some related work in both areas.

### 2.1 Notations

We consider a dataset consisting of $l$ labeled points $L = \{(x_i, y_i)\}_{i=1}^{l}$ and $u$ unlabeled points $U = \{x_i\}_{i=l+1}^{l+u}$, where data points $x_i$ lie in some space $\mathcal{X}$ (typically $\mathcal{X} \subset \mathbb{R}^d$) and labels $y_i \in \{1, \dots, C\}$ are discrete. We denote the combined data by $X \in \mathbb{R}^{n \times d}$, where $n = l+u$. We place ourselves in the typical semi-supervised scenario where $l \ll u$, and the goal is to predict $y_{l+1}, \dots, y_{l+u}$. Let $G = (V, W)$ be a graph composed of a set of nodes $V = \{v_1, \dots, v_n\}$ and a symmetric nonnegative weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$. We denote the set $\mathcal{W}$ of admissible weighted adjacency matrices by

$$\mathcal{W} = \{W : W \geq 0, \mathrm{diag}(W) = 0, W^\top = W\}. \tag{1}$$

Every observation $x_i$ (labeled or unlabeled) can be seen as a signal occurring at node $v_i$, and $W$ assigns a weight to each pair of nodes. We say that two nodes $v_i$ and $v_j$ are connected when $W_{ij} > 0$.

### 2.2 Graph-based SSL

Graph-based SSL takes as input a graph $G = (V, W)$ over the labeled and unlabeled data, the known labels and optionally some feature representation $X \in \mathbb{R}^{n \times d}$ of the data, and aims to predict the labels of the unlabeled points.

Many approaches exist for graph-based SSL, see (Subramanya and Talukdar, 2014) for a review of standard approaches. Classic algorithms take as input only the graph and use it to propagate the known

labels to the unlabeled points (Zhu and Ghahramani, 2002; Zhou et al., 2004; Bengio et al., 2006). A popular approach is label spreading (Zhou et al., 2004), which can be formulated as a convex optimization problem implementing a trade-off between two terms. The first term is a graph Laplacian regularization term which encourages similar predictions for strongly connected points in the graph, while the second one tries to keep the predictions accurate for points with known labels. There also exist algorithms formalized as graph partitioning problems inspired from spectral clustering (Joachims, 2003).

Graph-based SSL methods can also leverage a feature representation of the data in addition to the graph. A generic strategy is to add the graph Laplacian regularization term to existing supervised learning algorithms such as SVM or neural networks (Belkin et al., 2006; Weston et al., 2012). Following a different direction, (Yang et al., 2016) builds upon graph embeddings approaches to propose a method to predict labels based on the input representation as well as some embeddings learned from the input graph and known labels. Finally, there has been some recent interest in graph convolutional networks (GCN) (Kipf and Welling, 2017; Chen et al., 2018; Wu et al., 2019). Much like CNNs for images, they rely on an (approximation of) a notion of graph convolution, allowing them to learn a nonlinear transformation of the input representation while encoding the graph structure when propagating inputs from a layer to the next.

In all these approaches, the graph is fixed and given as input to the algorithm (for several methods, this is also true for the data representation). Their performance is thus very sensitive to the relevance of the graph for the task at hand: in particular, the underlying labeling should be smooth with respect to the graph. Our method is a principled approach to learn such a task-specific graph, and also infers a relevant data representation.

### 2.3 Graph Construction and Learning

As pointed out by (Subramanya and Talukdar, 2014), graph-based SSL methods typically rely on graphs constructed from the input data representation with a simple heuristic strategy. Popular choices include $k$-nearest neighbor graphs (connecting pairs of points that are among the $k$-closest to each other), $\epsilon$-graph (connecting points that are within distance $\epsilon$), and radial kernel graphs (building a fully connected graph with exponentially decreasing weights $W_{ij} = e^{-\gamma\|x_i - x_j\|^2}$). Recently, more sophisticated methods that learn the graph weights as the solution of an optimization problem have been introduced (Daitch et al., 2009; Kalofolias, 2016; Dong et al., 2016). Essentially, the weights are learned to be smooth over the data representation (i.e., assigning large weights to nearby points) with some regularization to enforce or control some properties such as connectedness and sparsity. In any case, the graphs obtained with the above approaches are task-independent in the sense that they ignore the labels.

To the best of our knowledge, there have been very few attempts to learn task-specific graphs for SSL. (Alexandrescu and Kirchhoff, 2007) propose to train a supervised classifier on labeled points and using the soft label predictions as the representation to build the graph. While this gives a way to incorporate label information, the supervised predictions are very dependent on the initial representation and the classifier itself can heavily overfit due to scarce labels.

## 3 Proposed Model

Our approach learns a graph and a data representation for use in downstream graph-based SSL algorithms. In this section, we start by introducing our formulation as a joint optimization problem over the representation and the graph. We then discuss some relevant choices for the parameterization of the learned representation, and finally present our alternating optimization scheme.

### 3.1 Problem Formulation

For the sake of generality, in this section we formulate our problem with respect to a generic representation function $\phi_\Theta : \mathcal{X} \to \mathbb{R}^k$, parameterized by $\Theta$, which represents any data point $x \in \mathcal{X}$ as a $k$-dimensional vector $\phi_\Theta(x) \in \mathbb{R}^k$. We discuss some relevant choices of representation functions in Section 3.2.

We propose to learn a weighted adjacency matrix $W^*$ and a representation function $\phi_{\Theta^*}$ by minimizing a joint objective function $f$ that involves both the labeled and unlabeled data points:

$$W^*, \Theta^* = \underset{W \in \mathcal{W}, \Theta}{\arg\min}\, f(W, \Theta).$$

Once the above optimization problem has been solved, the learned graph $W^*$ (which is based on the learned representation function $\phi_{\Theta^*}$) and possibly the representation $\phi_{\Theta^*}$ can then be given as input to any graph-based SSL algorithm to obtain predictions for the unlabeled data.

Our objective function $f(W, \Theta)$ decomposes into three terms:

$$f(W, \Theta) = f_1(\Theta) + \alpha[f_2(W) + f_3(W, \Theta)] \tag{2}$$

where $f_1(\Theta)$ and $f_2(W)$ are respectively the representation and graph specific terms, while $f_3(W, \Theta)$ is the joint term. Hyperparameter $\alpha \geq 0$ controls the trade-off between the (supervised) representation learning term $f_1$ and the unsupervised part ($f_2$ and $f_3$).

We now define these three terms. For notational convenience, let us denote by $Z \in \mathbb{R}^{n \times n}$ the matrix whose entries are the normalized squared Euclidean distances between data points in the transformed space, i.e. $(Z_\Theta)_{ij} = \frac{||\phi_\Theta(x_i) - \phi_\Theta(x_j)||^2}{\sum_{i<j} ||\phi_\Theta(x_i) - \phi_\Theta(x_j)||^2}$. The normalization conveniently removes the dependency on the scale of the data and $\Theta$. The representation term $f_1(\Theta)$ is defined on the labeled data points only and takes the following form:

$$f_1(\Theta) = \sum_{\substack{x_i, x_j, x_k \in L \\ y_i = y_j, y_i \neq y_k}} \big[(Z_\Theta)_{ij} - (Z_\Theta)_{ik} + 1\big]_+, \tag{3}$$

where $[\cdot]_+ = \max(0, \cdot)$. This is a large-margin triplet loss similar to those used in metric learning (Bellet et al., 2015): it attempts to learn a representation function $\phi_\Theta$ that brings each point $x_i$ closer to points $x_j$ with the same label than to differently labeled points $x_k$, with a safety margin of 1. In practice, we can subsample instead of summing over all possible triplets.

The graph term $f_2(W)$ is inspired from the (unsupervised) graph learning approach proposed by (Kalofolias, 2016):

$$f_2(W) = \beta ||W||_F^2 - \mathbf{1}^\top \log(\mathbf{1}^\top W), \tag{4}$$

The log-barrier term on the degrees prevents any node from being isolated in the graph, while the Frobenius norm is a shrinkage term over the graph weights. Combined with our joint term (6) defined below, hyperparameter $\beta \geq 0$ directly controls the sparsity of the learned graph: the smaller $\beta$, the more concentrated the weights of each point on its nearest neighbors in the learned representation (hence the sparser the graph). On the other hand, as $\beta \to +\infty$, the graph becomes complete with uniform weights. Sparsity allows to enforce the locality property (only close points are connected in the graph) which is necessary to obtain a good approximation of the data manifold. It also reduces the computational cost in downstream graph-based SSL algorithms, whose complexity typically depends on the number of edges in the graph.

Other options are possible for $f_2(W)$ depending on the prior we want to have on the structure of the graph. For instance, one may use

$$f_2(W) = (1/\gamma) \textstyle\sum_{i,j} W_{ij}[\log(W_{ij}) - 1], \tag{5}$$

where $\gamma > 0$ is a hyperparameter. This will force the graph to be fully connected.

Finally, we introduce the joint term bringing together the graph and the representation:

$$f_3(W, \Theta) = \mathrm{tr}(W Z_\Theta) = \textstyle\sum_{i,j} W_{ij} (Z_\Theta)_{ij}. \tag{6}$$

This can be seen as a weighted $L_1$ norm term on $W$ (which is why it induces sparsity), and equivalently written as a quadratic form of the Laplacian matrix of the graph encoded by the symmetric matrix $W$.

It is also used in approaches based on graph Laplacian regularization (see Section 2), but in our case both the graph and the representation are learned in joint manner. This term makes the graph and the representation as smooth as possible with respect to each other on *both labeled and unlabeled points*.

Overall, our joint objective function (2) is designed to produce a sparse topology that tends to be smooth with respect to the data manifold and the underlying labeling function through an appropriate representation. We now discuss the choice of representation function $\phi_\Theta$.

### 3.2 Choices of Representation Functions

Many options are possible for the representation function $\phi_\Theta$ depending on the nature of the data and task at hand. However, it is important to keep in mind that the amount of labeled information is scarce, hence learning complex text representations from scratch is likely to lead to severe overfitting. We argue that it is preferable to adapt pre-trained representations, which generally requires to optimize much fewer parameters. We give some concrete examples below.

**Linear transformation.** Pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014) are commonly used to represent texts in a vectorial space, e.g. by averaging the embeddings of the words occurring in a document. In order to adapt the representation to the task, we can learn a simple linear mapping $\phi_\Theta(x) = \Theta x$ which transforms the initial $d$-dimensional representation into a $k$-dimensional one, with $\Theta \in \mathbb{R}^{k \times d}$ and $k \leq d$. Such a strategy has been previously explored in the supervised setting to "re-embed" words in a task-specific manner (Denis and Ralaivola, 2017). This is the representation function that we use in our experiments (see Section 4).

**Weighted combination.** Recent work in learning deep contextualized word representations such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) allows to learn a task-specific combination of the token representations obtained at the $K$ layers of the model, which typically capture different aspects of tokens (from syntax to semantics). In this case, we have $K$ initial $d$-dimensional representations $x \in \mathbb{R}^{K \times d}$ for each text $x$ and we learn a weighted combination $\phi_\Theta(x) = \Theta x \in \mathbb{R}^d$ where $\Theta \in \mathbb{R}^K$ is simply a $K$-dimensional parameter vector.

### 3.3 Optimization

We propose to optimize the cost function $f(W, \Theta)$ by alternating minimization over $W$ and $\Theta$, which is guaranteed to converge to a local optimum. This is a natural approach: one step learns a smooth graph given the current representation $\Theta$, while the other learns a smooth representation with respect to the current graph (this can be seen as a regularizer for $\Theta$ based on unlabeled data) and also tries to keep labeled points of the same class closer than points of different class.

As the joint problem is nonconvex, initialization plays an important role. We propose to initialize the graph weights to zero and to start by optimizing $\Theta$ so that the initial representation focuses only on the (scarce) labeled data. The graph learned on this representation will thus strongly connect together the labeled points as well as unlabeled points that are very close to the labeled points and are thus likely to share the same label. At the next iteration, these unlabeled points will then contribute in learning a better representation and in turn a graph which strongly connects new unlabeled points. This process can be seen as a principled version of self-training heuristics popular in traditional (non-graph-based) semi-supervised learning (Triguero et al., 2015).

The subproblem of optimizing $W$ given $\Theta$ is convex regardless of whether we define $f_2(W)$ as (4) or (5). Using (5) is computationally convenient as the subproblem has a closed-form solution: the weights are exponentially decreasing with the distance in the current representation $\phi_\Theta$, as given by the radial kernel $W_{ij} = \exp(-\gamma(Z_\Theta)_{ij})$ (Kalofolias, 2016). Note that unlike the classic radial kernel baseline construction method mentioned in Section 2.3, our graph is computed based on the learned representation $\phi_\Theta$ by minimizing the joint objective function with respect to $W$. One drawback of using (5) is that the resulting graphs are always fully connected. Using (4) instead, we can obtain sparse graphs but the solution must be computed with an iterative algorithm. We found that the primal-dual algorithm introduced by (Kalofolias, 2016) converges slowly in practice — we instead optimize $W$ by simple

gradient descent over the "effective" $n(n-1)/2$ weights, adding a small positive constant inside the log term in (4) to make the objective function smooth.

As $\phi_\Theta$ is typically differentiable in $\Theta$ (as in the examples outlined in Section 3.2), we also solve the subproblem in $\Theta$ by (stochastic) gradient descent. Note that this subproblem is generally nonconvex due to the distance difference in $f_1(\Theta)$.

**Remark.** *Updating $W$ requires to optimize over $O(n^2)$ variables, which was manageable for the datasets used in our experiments. To scale to larger datasets, one can restrict the optimization to the weights corresponding to pairs of points that are close enough in the learned representation space*[1] *(other weights are kept to 0). This has a negligible impact on the solution in sparse regimes (small $\beta$).*

## 4 Experiments

In this section, we study the practical behavior of our method by comparing the accuracy of downstream graph-based SSL algorithms when the graph (along with the underlying representation) is learned with our approach ( **ours**) rather than constructed with the following baseline strategies:

- **radial**: Complete graph with weights $W_{ij} = \exp(-\gamma\|x_i - x_j\|^2)$.
- **knn**: $W_{ij} = 1$ for $x_i$ in the $k$-neighborhood of $x_j$ (or vice versa), and $W_{ij} = 0$ otherwise.
- **kalo**: Unsupervised graph learning with the method of (Kalofolias, 2016). This corresponds to our approach when using the graph term (4) and keeping the original representation fixed.

In all cases the graph is constructed over the union of labeled (train set) and unlabeled data (validation and test sets). For experiments with our method, the learned representation is a linear transformation of the initial features as explained in Section 3.2.

We perform experiments with two graph-based SSL approaches: Label Spreading (LS) (Zhou et al., 2004) and the Graph Convolutional Network (GCN) method of (Kipf and Welling, 2017). We used the scikit-learn (Pedregosa et al., 2011) implementation of LS. For GCN, we used the TensorFlow implementation provided by the authors[2] and follow the recommended architecture:

$$\hat{Y} = \mathrm{softmax}(\tilde{L}\max(0, \tilde{L}XH_0)H_1), \tag{7}$$

where $\tilde{L} = \tilde{D}^{1/2}\tilde{W}\tilde{D}^{1/2}$ is the normalized Laplacian corresponding to the graph $\tilde{W} = W + \lambda I$ (the input graph augmented with self-loops), $\tilde{D}$ is the diagonal degree matrix ($\tilde{D}_{ii} = \sum_j \tilde{D}_{ij}$), and $H_0 \in \mathbb{R}^{d\times h}$, $H_1 \in \mathbb{R}^{h\times C}$ are the parameters to be learned. We set the number of hidden units $h$ to 16 and $\lambda$ to 1 as done in (Kipf and Welling, 2017).

To illustrate the behavior of our approach, we first present some experiments on synthetic data. We then show some results on real text classification datasets.

### 4.1 Synthetic Data

We generated a 3-dimensional dataset consisting of 100 points evenly distributed in two classes (Figure 1). We have two clusters per class placed far from each other while keeping clusters from different classes closer. We randomly picked $60\%$ of the points and removed their labels.

We compare the classification error of GCN and Label Spreading when the input graph is given by our approach instead of using baseline graph construction methods. For GCN, we also give as input the representation learned with our approach. For our approach, we use the graph term (4) and for each labeled point $x_i$, we random sample 2 points $x_j$ of the same class and 3 points $x_k$ of different class and construct all combinations $(x_i, x_j, x_k)$, leading to 6 triplets for each $x_i$ in the triplet loss (3). The results given in Table 1 show that our approach clearly and consistently outperforms all methods in both GCN and Label Spreading.[3] The improvements are especially large for Label Spreading, as LS makes

[1] These can be identified in near-linear time using approximate nearest-neighbor techniques (Muja and Lowe, 2014).

[2] `https://github.com/tkipf/gcn`

[3] For this illustrating experiment, we picked the values of hyperparameters giving the best results for each method.

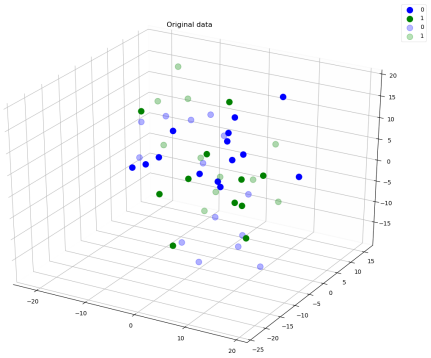



Figure 1: Original 3-dimensional synthetic data. Semi-transparent points are unlabeled.

| | Label Spreading | GCN |
|---|---|---|
| radial | 50.7 | 93.3 |
| knn | 81.3 | 93.3 |
| kalo | 77.3 | 88.0 |
| ours | **96.0** | **96.0** |

Table 1: Classification accuracy on the synthetic dataset.

predictions based on the graph only. In contrast, GCN learns its own (nonlinear) transformation of the representation given as input in an end-to-end manner. Still, our method is able to provide some gains for GCN as well, by providing it with a better graph. Note for instance the significant improvement compared to **kalo**, which learns the graph on the original representation.

To visualize this difference, Figure 2a shows the graph learned by **kalo**. Although the graph is learned to minimize the smoothness criterion with respect to the data, it fails to accurately capture the label distribution due to the limitations of the initial representation. Our alternating optimization approach overcomes this issue by learning a task-specific graph through an appropriate representation. In Figure 2b-2c-2d, we can see how label information is gradually injected at each step: after the first iteration, the graph is already significantly more smooth with respect to the underlying labeling and the graph is also sparser, but some edges between differently labeled points as well as an overly connected point remain. The following iterations further improve the graph quality. This explains the better performance obtained in downstream semi-supervised algorithms.

### 4.2 Real Data

We now evaluate our method on three text classification tasks derived from the 20NewsGroups dataset,[4] a collection of documents categorized into 20 topics, each one of which is partitioned into sub-topics. We chose the topics of *computers* with classes IBM and Mac ($n = 1945$ documents), *religion* with classes atheism and Christian ($n = 1796$), and *sports* with classes baseball and hockey ($n = 1993$).

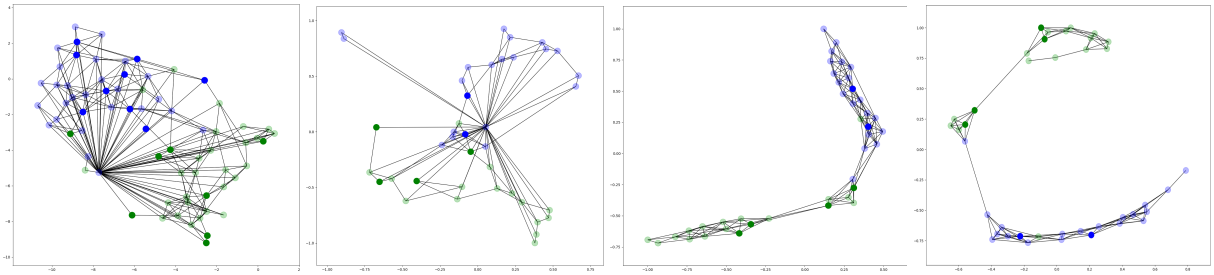
For all datasets, we represent data points using the average token embedding based on word2vec (Mikolov et al., 2013). These embeddings are of dimension $d = 300$ and were trained on a 100B word corpus of Google news data (vocabulary size is 3M).[5]

We experiment with different proportions of unlabeled points in the training set ($90\%$, $75\%$, $60\%$ and $40\%$), while the rest of the data is evenly split into a validation and a test set. As commonly done in semi-supervised learning, we train on the union of the (labeled) training set and the (unlabeled) validation and test sets, select the values of hyperparameters based on the accuracy on the validation set, and report the corresponding accuracy on the test set.

To evaluate our approach we optimize the objective (2) as described in Section 3.3 with the graph term defined as in (4). To compute the representation term of our objective defined in (3), we construct triplets

[4] `http://qwone.com/~jason/20Newsgroups/`

[5] `https://code.google.com/archive/p/word2vec/`



(a) **kalo** on original data. (b) **ours** after 1st iteration. (c) **ours** after 2nd iteration. (d) **ours** at last iteration.

Figure 2: Force-directed drawing (spring layout) of graphs learned with **kalo**, and with our method at several iterations of our alternating optimization algorithm. Semi-transparent points are unlabeled.

| dataset | % | radial | knn | kalo | ours |
|---|---|---|---|---|---|
| comp | 90 | 62.76 | 61.89 | 63.63 | **67.10**$^\dagger$ |
| comp | 75 | 67.35 | 67.35 | 69.87 | **74.67** |
| comp | 60 | 70.92 | 67.76 | 72.84 | **75.58** |
| comp | 40 | 76.58 | 70.99 | 75.86 | **77.48** |
| rel | 90 | 80.47 | 79.53 | 80.59 | **83.88**$^\dagger$ |
| rel | 75 | 84.74 | 85.05 | 84.66 | **85.18** |
| rel | 60 | 85.74 | 83.36 | 87.22 | **88.26** |
| rel | 40 | 84.60 | 85.77 | **86.74** | 85.96 |
| sports | 90 | 84.11 | 81.78 | 86.55 | **95.66**$^\dagger$ |
| sports | 75 | 89.81 | 90.87 | 89.93 | **96.02**$^\dagger$ |
| sports | 60 | 92.77 | 91.43 | 92.64 | **97.19** |
| sports | 40 | 95.43 | 93.32 | 95.08 | **97.36** |

(a) Label spreading

| dataset | % | radial | knn | kalo | ours |
|---|---|---|---|---|---|
| comp | 90 | 69.60 | 65.91 | **70.36**$^\dagger$ | 67.97 |
| comp | 75 | 74.55 | 67.95 | 73.71 | **75.15** |
| comp | 60 | **77.78** | 68.86 | 74.21 | 76.82 |
| comp | 40 | **81.08** | 67.21 | 80.72 | 76.76 |
| rel | 90 | 83.06 | 82.35 | 81.53 | **83.41** |
| rel | 75 | 83.49 | 83.62 | 83.88 | **85.57**$^\dagger$ |
| rel | 60 | 83.36 | 83.21 | **86.92** | 86.03 |
| rel | 40 | **88.30** | 82.65 | 87.33 | 86.16 |
| sports | 90 | 94.70 | 92.48 | 93.33 | **95.13**$^\dagger$ |
| sports | 75 | **96.84** | 94.85 | 95.78 | 96.25 |
| sports | 60 | **98.80**$^\dagger$ | 95.85 | 97.19 | 96.92 |
| sports | 40 | **98.77** | 97.01 | 97.72 | 97.89 |

(b) GCN

Table 2: Classification accuracies of Label Spreading and GCN for different graph construction methods and proportions of unlabeled data. McNemar test to compare **ours** vs. the best baseline is statistically significant for those results marked with a dagger symbol †.

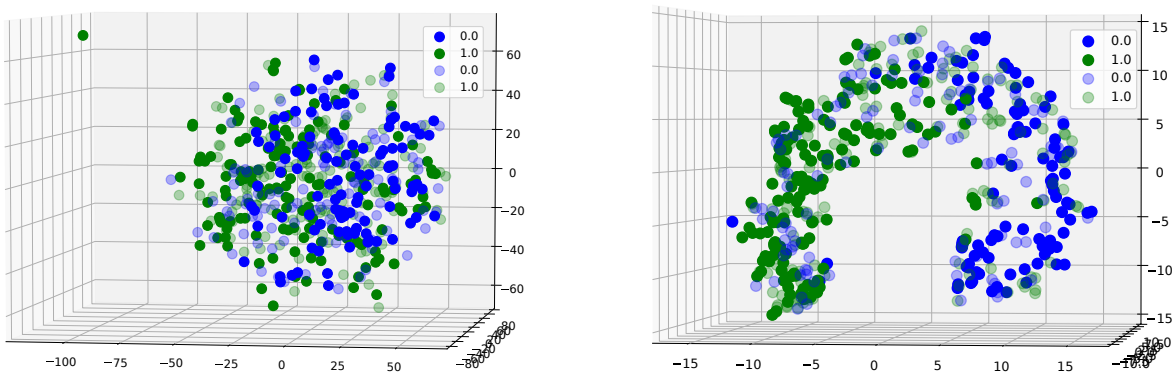
as follows: for each pair $(x_i, y_i)$ in the labeled set we obtain the closest points with labels other than $y_i$ ("imposters"), and the closest points with label $y_i$ ("targets"). We picked 8 imposters and 3 targets. We tune the hyperparameters $\alpha$ from $\{0.001, 0.01, 1\}$, $\beta$ from $\{0.00001, 0.001, 0.1, 1\}$, the dimension $k$ of the learned representation from $\{16, 32, 64\}$, and perform early stopping with respect to the number of alternating steps between learning the graph and learning the representation (up to 10 alternating steps). We also tuned the hyperparameters of each baseline method ($\gamma$ for **radial**, $k$ for **knn** and $\beta$ for **kalo**) and the trade-off hyperparameter of Label Spreading. Finally, we computed the McNemar test of significance (McNemar Quinn, 1947) to compare the performance of our method against the best baseline. Results marked with a dagger symbol † yield a statistically significant test for a significance level of 0.05.

**Label Spreading.** Table 2a reports test classification accuracies obtained on the test set for each configuration of dataset and proportion of unlabeled data. Our approach clearly outperforms all baselines, most of the time by a large margin. Also, McNemar test indicates that we tend to be significantly better than the best baseline in the more challenging settings where labeled data is the most scarce. The results also show that learning the representation along with the graph makes a clear difference compared to learning the graph only (as seen by the superior performance of **ours** over **kalo**).

As LS only uses the graph to make predictions, these results provide strong evidence of the superior quality of the graphs learned with our method.

**Graph Convolutional Networks.** We now turn to the more complex GCN prediction model. We reuse the same setup as for LS and feed GCN with both the learned representation and the learned graph.

Table 2b summarizes the results. The gains obtained with our approach are smaller than those ob-




(a) Original representation

(b) Learned representation

Figure 3: 3D PCA visualization of the original representation (left) and the representation learned with our approach (right) on the *rel* dataset ($\%75$ unlabeled). Transparent dots represent unlabeled documents.

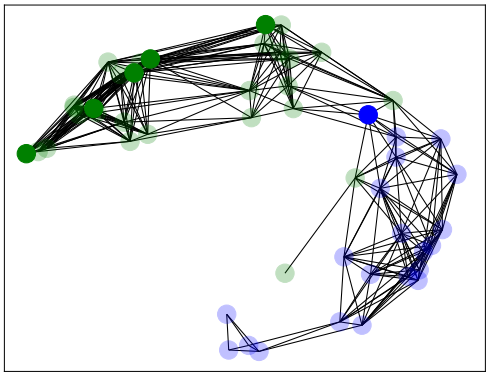


Figure 4: Force-directed drawing (spring layout) of a random $50$-node subgraph of the graph learned with our approach on the *rel* dataset ($\%75$ labeled).

tained in LS, which is to be expected since GCN has the ability to learn nonlinear transformations of the data. Nevertheless, we do observe some performance gains, as our approach generally improves upon or closely matches the performance of the best baseline. An interesting finding is that our method tends to close the gap of performance between LS and the richer neural-based GCN model. This suggests that simple propagation approaches may be sufficient in practice for many datasets, if provided with the right graph.

**Visualization.** We provide visualizations of the representation and the graph learned with our approach on the *rel* dataset. Figure 3 shows 3D PCA visualizations of the original representation and the representation learned with our approach. We see that the two classes are quite mixed up in the original representation while the learned representation is much smoother with respect to the underlying labeling (even in this crude low-dimensional summary). Figure 4 gives a snapshot of the graph learned with our approach by showing a subgraph of $50$ randomly sampled nodes (subsampling helps to avoid clutter). The graph is very smooth with respect to the underlying labeling, and suggests that the learned high-dimensional representation has a nice manifold structure, with some regions of higher densities.

## 5 Discussion and Future Work

We presented a novel method bringing together graph learning, representation learning and SSL by jointly inferring the graph and the representation from semi-supervised data. The output of our approach can then be plugged into any graph-based SSL algorithm in place of using common graph constructions. Our experimental results suggest that the gains are especially significant for graph-based SSL algorithms that are unable to adapt the data representation (like label spreading and its variants), although we observe

some gains also for GCN. To further improve the performance with richer models like GCNs, a promising direction is to extend our approach to learn the graph, the representation and the classifier in an end-to-end manner. We note that there has been very recent attempts in this general direction (Franceschi et al., 2019), though specific to GCN and with completely different modeling and assumptions.

The ideas underlying our approach could also be useful to tackle transfer learning settings and in particular domain adaptation (Ben-David et al., 2010). The latter can be seen as a SSL problem where the distribution of the target (unlabeled) data follows a different distribution that the source (training) data. Our objective function could be modified to encourage the learned representation and graph to serve as a "bridge" between the source and target distributions.

## Acknowledgements

The authors would like to thank the Agence Nationale de la Recherche for funding this project with the grant number ANR-16-CE33-0011, and the reviewers for their feedback and suggestions. We also thank Grid5000 team for providing the necessary resources and technical support for this project, and CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

## References

Andrei Alexandrescu and Katrin Kirchhoff. 2007. Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP. In *HLT-NAACL*.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2015. *Metric Learning*. Morgan & Claypool Publishers.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. Transductive Learning via Spectral Graph Partitioning. *Machine Learning*, 79(1–2):151–175.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label Propagation and Quadratic Criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press.

Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.

Samuel I. Daitch, Jonathan A. Kelner, and Daniel A. Spielman. 2009. Fitting a graph to vector data. In *ICML*.

Dipanjan Das and Noah A Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *ACL*.

Pascal Denis and Liva Ralaivola. 2017. Online Learning of Task-specific Word Representations with a JointBi-convex Passive-Aggressive Algorithm. In *EACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. 2016. Learning Laplacian Matrix in Smooth Graph Signal Representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning Discrete Structures for Graph Neural Networks. In *ICML*.

Andrew Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraph*.

Thorsten Joachims. 2003. Transductive Learning via Spectral Graph Partitioning. *Machine Learning*, 20(1):290–297.

Vassilis Kalofolias. 2016. How to learn a graph from smooth signals. In *AISTATS*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised Classification with Graph-convolutional Neural Networks. In *ICLR*.

McNemar Quinn. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*.

Marius Muja and David G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Avneesh Saluja, Hany Hassan, Kristina Toutanova, and Chris Quirk. 2014. Graph-based semi-supervised learning of translation models from monolingual data. In *ACL*.

Amarnag Subramanya and Jeff Bilmes. 2008. Soft-supervised Learning for Text Classification. In *EMNLP*.

Amarnag Subramanya and Partha Pratim Talukdar. 2014. *Graph-Based Semi-Supervised Learning*. Morgan & Claypool Publishers.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient Graph-based Semi-supervised Learning of Structured Tagging Models. In *EMNLP*.

Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep Learning via Semi-supervised Embedding. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 639–655.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. *arXiv:1902.07153*.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with Local and Global Consistency. In *NIPS*.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Xiaojin Zhu, Z Ghahramani, and John Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*.