Revisiting Semi-Supervised Learning with Graph Embeddings

Zhilin Yang ZHILINY@CS.CMU.EDU
William W. Cohen WCOHEN@CS.CMU.EDU
Ruslan Salakhutdinov RSALAKHU@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University

Abstract

We present a semi-supervised learning framework based on graph embeddings. Given a graph between instances, we train an embedding for each instance to jointly predict the class label and the neighborhood context in the graph. We develop both transductive and inductive variants of our method. In the transductive variant of our method, the class labels are determined by both the learned embeddings and input feature vectors, while in the inductive variant, the embeddings are defined as a parametric function of the feature vectors, so predictions can be made on instances not seen during training. On a large and diverse set of benchmark tasks, including text classification, distantly supervised entity extraction, and entity classification, we show improved performance over many of the existing models.

1. Introduction

Semi-supervised learning aims to leverage unlabeled data to improve performance. A large number of semi-supervised learning algorithms jointly optimize two training objective functions: the supervised loss over labeled data and the unsupervised loss over both labeled and unlabeled data. *Graph-based semi-supervised learning* defines the loss function as a weighted sum of the supervised loss over labeled instances and a graph Laplacian regularization term (Zhu et al., 2003; Zhou et al., 2004; Belkin et al., 2006; Weston et al., 2012). The graph Laplacian regularization is based on the assumption that nearby nodes in a graph are likely to have the same labels. Graph Laplacian regularization is effective because it constrains the labels to be consistent with the graph structure.

Recently developed unsupervised representation learning methods learn embeddings that predict a distributional *context*, e.g. a word embedding might predict nearby *context words* (Mikolov et al., 2013; Pennington et al., 2014), or a node embedding might predict nearby nodes in a graph (Perozzi et al., 2014; Tang et al., 2015). Embeddings trained with distributional context can be used to boost the performance of related tasks. For example, word embeddings trained from a language model can be applied to part-of-speech tagging, chunking and named entity recognition (Collobert et al., 2011; Yang et al., 2016).

In this paper we consider not word embeddings but graph embeddings. Existing results show that graph embeddings are effective at classifying the nodes in a graph, such as user behavior prediction in a social network (Perozzi et al., 2014; Tang et al., 2015). However, the graph embeddings are usually learned separately from the supervised task, and hence do not leverage the label information in a specific task. Hence graph embeddings are in some sense complementary to graph Laplacian regularization that does not produce useful features itself and might not be able to fully leverage the distributional information encoded in the graph structure.

The main highlight of our work is to incorporate embedding techniques into the graph-based semi-supervised learning setting. We propose a novel graph-based semi-supervised learning framework, *Planetoid* (Predicting Labels And Neighbors with Embeddings Transductively Or Inductively from Data). The embedding of an instance is jointly trained to predict the class label of the instance and the context in the graph. We then concatenate the embeddings and the hidden layers of the original classifier and feed them to a softmax layer when making the prediction.

Since the embeddings are learned based on the graph structure, the above method is *transductive*, which means we can only predict instances that are already observed in the graph at training time. In many cases, however, it may be desirable to have an *inductive* approach, where predictions can be made on instances unobserved in the graph seen at training time. To address this issue, we further develop an inductive variant of our framework, where we define the

Proceedings of the 33^{rd} *International Conference on Machine Learning*, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

embeddings as a parameterized function of input feature vectors; i.e., the embeddings can be viewed as hidden layers of a neural network.

To demonstrate the effectiveness of our proposed approach, we conducted experiments on five datasets for three tasks, including text classification, distantly supervised entity extraction, and entity classification. Our inductive method outperforms the second best inductive method by up to 18.7%[1] points and on average 7.8% points in terms of accuracy. The best of our inductive and transductive methods outperforms the best of all the other compared methods by up to 8.5% and on average 4.1%.

2. Related Work

2.1. Semi-Supervised Learning

Let L and U be the number of labeled and unlabeled instances. Let $\mathbf{x}_{1:L}$ and $\mathbf{x}_{L+1:L+U}$ denote the feature vectors of labeled and unlabeled instances respectively. The labels $y_{1:L}$ are also given. Based on both labeled and unlabeled instances, the problem of semi-supervised learning is defined as learning a classifier $f : \mathbf{x} \to y$. There are two learning paradigms, transductive learning and inductive learning. Transductive learning (Zhu et al., 2003; Zhou et al., 2004) only aims to apply the classifier f on the unlabeled instances observed at training time, and the classifier does not generalize to unobserved instances. For instance, transductive support vector machine (TSVM) (Joachims, 1999) maximizes the "unlabeled data margin" based on the low-density separation assumption that a good decision hyperplane lies on a sparse area of the feature space. Inductive learning (Belkin et al., 2006; Weston et al., 2012), on the other hand, aims to learn a parameterized classifier f that is generalizable to unobserved instances.

2.2. Graph-Based Semi-Supervised Learning

In addition to labeled and unlabeled instances, a graph, denoted as a $(L+U) \times (L+U)$ matrix A, is also given to graph-based semi-supervised learning methods. Each entry a_{ij} indicates the similarity between instance i and j, which can be either labeled or unlabeled. The graph A can either be derived from distances between instances (Zhu et al., 2003), or be explicitly derived from external data, such as a knowledge graph (Wijaya et al., 2013) or a citation network between documents (Ji et al., 2010). In this paper, we mainly focus on the setting that a graph is explicitly given and represents additional information not present in the feature vectors (e.g., the graph edges correspond to hyperlinks between documents, rather than distances between the bag-of-words representation of a document).

[1]% refers to absolute percentage points thoughout the paper.

Graph-based semi-supervised learning is based on the assumption that nearby nodes tend to have the same labels. Generally, the loss function of graph-based semi-supervised learning in the binary case can be written as

$$\begin{aligned} & \sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \sum_{i,j} a_{ij} \|f(x_i) - f(x_j)\|^2 \\ = & \sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \mathbf{f}^T \Delta \mathbf{f} \end{aligned} \tag{1}$$

In Eq. (1), the first term is the standard supervised loss function, where $l(\cdot,\cdot)$ can be log loss, squared loss or hinge loss. The second term is the graph Laplacian regularization, which incurs a large penalty when similar nodes with a large w_{ij} are predicted to have different labels $f(x_i) \neq f(x_j)$. The graph Laplacian matrix Δ is defined as $\Delta = A - D$, where D is a diagonal matrix with each entry defined as $d_{ii} = \sum_j a_{ij}$. λ is a constant weighting factor. (Note that we omit the parameter regularization terms for simplicity.) Various graph-based semi-supervised learning algorithms define the loss functions as variants of Eq. (1). Label propagation (Zhu et al., 2003) forces f to agree with labeled instances $y_{1:L}$; f is a label lookup table for unlabeled instances in the graph, and can be obtained with a closed-form solution. Learning with local and global consistency (Zhou et al., 2004) defines l as squared loss and f as a label lookup table; it does not force f to agree with labeled instances. Modified Adsorption (MAD) (Talukdar & Crammer, 2009) is a variant of label propagation that allows prediction on labeled instances to vary and incorporates node uncertainty. Manifold regularization (Belkin et al., 2006) parameterizes f in the Reproducing Kernel Hilbert Space (RKHS) with l being squared loss or hinge loss. Since f is a parameterized classifier, manifold regularization is inductive and can naturally handle unobserved instances.

Semi-supervised embedding (Weston et al., 2012) extends the regularization term in Eq. (1) to be $\sum_{i,j} a_{ij} \|\mathbf{g}(x_i) - \mathbf{g}(x_j)\|^2$, where $\mathbf{g}$ represents embeddings of instances, which can be the output labels, hidden layers or auxiliary embeddings in a neural network. By extending the regularization from f to $\mathbf{g}$, this method imposes stronger constraints on a neural network. Iterative classification algorithm (ICA) (Sen et al., 2008) uses a local classifier that takes the labels of neighbor nodes as input, and employs an iterative process between estimating the local classifier and assigning new labels.

2.3. Learning Embeddings

Extensive research was done on learning graph embeddings. A probabilistic generative model was proposed to learn node embeddings that generate the edges in a graph

Table 1. Comparison of various semi-supervised learning algorithms and graph embedding algorithms. √means using the given formulation or information; ×means not available or not using the information. In the column *graph*, *regularization* means imposing regularization with the graph structure; *features* means using graph structure as features; *context* means predicting the graph context.

Method	Features	Labels	Paradigm	Embeddings	Graph
TSVM (Joachims, 1999)	√	√	Transductive	×	×
Label propagation (Zhu et al., 2003)	×	√	Transductive	×	Regularization
Manifold Reg (Belkin et al., 2006)	√	√	Inductive	×	Regularization
ICA (Sen et al., 2008)	×	√	Transductive	×	Features
MAD (Talukdar & Crammer, 2009)	×	√	Transductive	×	Regularization
Semi Emb (Weston et al., 2012)	√	√	Inductive	√	Regularization
Graph Emb (Perozzi et al., 2014)	×	×	Transductive	√	Context
Planetoid (this paper)	√	√	Both	√	Context

(Snijders & Nowicki, 1997). A clustering method (Handcock et al., 2007) was proposed to learn latent social states in a social network to predict social ties.

More recently, a number of embedding learning methods are based on the Skipgram model, which is a variant of the softmax model. Given an instance and its context, the objective of Skipgram is usually formulated as minimizing the log loss of predicting the context using the embedding of an instance as input features. Formally, let $\{(i, c)\}$ be a set of pairs of instance i and context c, the loss function can be written as

$$-\sum_{(i,c)} \log p(c|i) = -\sum_{(i,c)} \left(\mathbf{w}_c^T \mathbf{e}_i - \log \sum_{c' \in \mathcal{C}} \exp(\mathbf{w}_{c'}^T \mathbf{e}_i) \right) \tag{2}$$

where $\mathcal{C}$ is the set of all possible context, $\mathbf{w}$'s are parameters of the Skipgram model, and $\mathbf{e}_i$ is the embedding of instance i. Skipgram was first introduced to learn representations of words, known as word2vec (Mikolov et al., 2013). In word2vec, for each training pair (i, c), the instance i is the current word whose embedding is under estimation; the context c is each of the surrounding words of i within a fixed window size in a sentence; the context space $\mathcal{C}$ is the vocabulary of the corpus. Skipgram was later extended to learn graph embeddings. Deepwalk (Perozzi et al., 2014) uses the embedding of a node to predict the context in the graph, where the context is generated by random walk. More specifically, for each training pair (i, c), the instance i is the current node whose embedding is under estimation; the context c is each of the neighbor nodes within a fixed window size in a generated random walk sequence; the context space $\mathcal{C}$ is all the nodes in the graph. LINE (Tang et al., 2015) extends the model to have multiple context spaces $\mathcal{C}$ for modeling both first and second order proximity.

Although Skipgram-like models for graphs have received much recent attention, many other models exist. TransE (Bordes et al., 2013) learns the embeddings of entities in a knowledge graph jointly with their relations. Autoencoders were used to learn graph embeddings for clustering on graphs (Tian et al., 2014).

2.4. Comparison

We compare our approach in this paper with other methods in semi-supervised learning and embedding learning in Table 1. Unlike our approach, conventional graph Laplacian based methods (Zhu et al., 2003; Belkin et al., 2006; Talukdar & Crammer, 2009) impose regularization on the labels but do not learn embeddings. Semi-supervised embedding method (Weston et al., 2012) learns embeddings in a neural network, but our approach is different from this method in that instead of imposing regularization, we use the embeddings to predict the context in the graph. Graph embedding methods (Perozzi et al., 2014; Tian et al., 2014) encode the graph structure into embeddings; however, different from our approach, these methods are purely unsupervised and do not leverage label information for a specific task. Moreover, these methods are transductive and cannot be directly generalized to instances unseen at training time.

3. Semi-Supervised Learning with Graph Embeddings

Following the notations in the previous section, the input to our method includes labeled instances $\mathbf{x}_{1:L}$, $y_{1:L}$, unlabeled instances $\mathbf{x}_{L+1:L+U}$ and a graph denoted as a matrix A. Each instance i has an embedding denoted as $\mathbf{e}_i$.

We formulate our framework based on feed-forward neural networks. Given the input feature vector $\mathbf{x}$, the k-th hidden layer of the network is denoted as $\mathbf{h}^k$, which is a nonlinear function of the previous hidden layer $\mathbf{h}^{k-1}$ defined as: $\mathbf{h}^k(\mathbf{x}) = \text{ReLU}(\mathbf{W}^k \mathbf{h}^{k-1}(\mathbf{x}) + b^k)$, where $\mathbf{W}^k$ and b^k are parameters of the k-th layer, and $\mathbf{h}^0(\mathbf{x}) = \mathbf{x}$. We adopt rectified linear unit $\text{ReLU}(x) = \max(0, x)$ as the nonlinear function in this work.

The loss function of our framework can be expressed as

$$\mathcal{L}_s + \lambda \mathcal{L}_u,$$

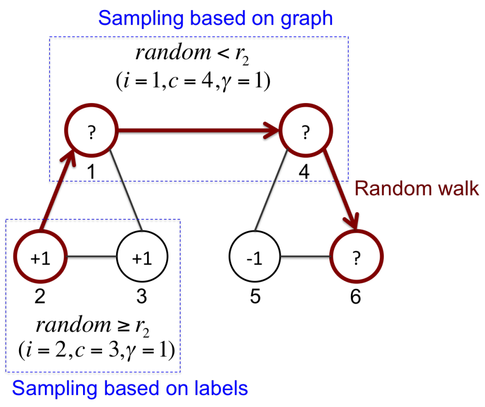


Figure 1. An example of sampling from context distribution $p(i, c, \gamma)$ when $\gamma = 1$ and $d = 2$. In circles, $+1$ denotes positive instances, -1 denotes negative instances, and ? denotes unlabeled instances. If $random < r_2$, we first sample a random walk $2 \rightarrow 1 \rightarrow 4 \rightarrow 6$, and then sample two nodes in the random walk within distance d. If $random \geq r_2$, we sample two instances with the same labels.

where $\mathcal{L}_s$ is a supervised loss of predicting the labels, and $\mathcal{L}_u$ is an unsupervised loss of predicting the graph context. In the following sections, we first formulate $\mathcal{L}_u$ by introducing how to sample context from the graph, and then formulate $\mathcal{L}_s$ to form our semi-supervised learning framework.

3.1. Sampling Context

We formulate the unsupervised loss $\mathcal{L}_u$ as a variant of Eq. (2). Given a graph A, the basic idea of our approach is to sample pairs of instance i and context c, and then formulate the loss $\mathcal{L}_u$ using the log loss $-\log p(c|i)$ as in Eq. (2). We first present the formulation of $\mathcal{L}_u$ by introducing negative sampling, and then discuss how to sample pairs of instance and context.

It is usually intractable to directly optimize Eq. (2) due to normalization over the whole context space $\mathcal{C}$. Negative sampling was introduced to address this issue (Mikolov et al., 2013), which samples negative examples to approximate the normalization term. In our case, we are sampling (i, c, γ) from a distribution, where i and c denote instance and context respectively, $\gamma = +1$ means (i, c) is a positive pair and $\gamma = -1$ means negative. Given (i, c, γ), we minimize the cross entropy loss of classifying the pair (i, c) to a binary label γ:

$$-\mathbb{I}(\gamma = 1) \log \sigma(\mathbf{w}_c^T \mathbf{e}_i) - \mathbb{I}(\gamma = -1) \log \sigma(-\mathbf{w}_c^T \mathbf{e}_i),$$

where σ is the sigmoid function defined as $\sigma(x) = 1/(1 + e^{-x})$, and $\mathbb{I}(\cdot)$ is an indicator function that outputs 1 when the argument is true, otherwise 0. Therefore, the unsupervised loss with negative sampling can be written as

$$\mathcal{L}_u = -\mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i) \qquad (3)$$

The distribution $p(i, c, \gamma)$ is conditioned on labels $y_{1:L}$ and the graph A. However, since they are the input to our algorithm and kept fixed, we drop the conditioning in our notation.

We now define the distribution $p(i, c, \gamma)$ directly using a sampling process, which is illustrated in Algorithm 1. There are two types of context that are sampled in this algorithm. The first type of context is based on the graph A, which encodes the structure (distributional) information, and the second type of context is based on the labels, which we use to inject label information into the embeddings. We use a parameter $r_1 \in (0, 1)$ to control the ratio of positive and negative samples, and use $r_2 \in (0, 1)$ to control the ratio of two types of context.

With probability r_2, we sample the context based on the graph A. We first uniformly sample a random walk sequence S. More specifically, we uniformly sample the first instance S_1 from the set $1 : L + U$. Given the previous instance $S_{k-1} = i$, the next instance $S_k = j$ is sampled with probability $a_{ij} / \sum_{j'=1}^{L+U} a_{ij'}$. With probability r_1, we sample a positive pair (i, c) from the set $\{(S_j, S_k) : |j - k| < d\}$, where d is another parameter determining the window size. With probability $(1 - r_1)$, we uniformly corrupt the context c to sample a negative pair.

With probability $(1 - r_2)$, we sample the context based on the class labels. Positive pairs have the same labels and negative pairs have different labels. Only labeled instances $1 : L$ are sampled.

Our random walk based sampling method is built upon Deepwalk (Perozzi et al., 2014). In contrast to their method, our method handles real-valued A, incorporates negative sampling, and explicitly samples from labels with probability $(1 - r_2)$ to inject supervised information.

An example of sampling when $\gamma = 1$ is shown in Figure 1.

3.2. Transductive Formulation

In this section, we present a method that infers the labels of unlabeled instances $y_{L+1:L+U}$ without generalizing to unobserved instances. Transductive learning usually performs better than inductive learning because transductive learning can leverage the unlabeled test data when training the model (Joachims, 1999).

We apply k layers on the input feature vector $\mathbf{x}$ to obtain $\mathbf{h}^k(\mathbf{x})$, and l layers on the embedding $\mathbf{e}$ to obtain $\mathbf{h}^l(\mathbf{e})$, as illustrated in Figure 2(a). The two hidden layers are concatenated, and fed to a softmax layer to predict the class

Algorithm 1 Sampling Context Distribution $p(i, c, \gamma)$

```
**Input:** graph $A$, labels $y_{1:L}$, parameters $r_1, r_2, q, d$
Initialize triplet $(i, c, \gamma)$
**if** $random < r_1$ **then** $\gamma \leftarrow +1$ **else** $\gamma \leftarrow -1$
**if** $random < r_2$ **then**
    Uniformly sample a random walk $S$ of length $q$
    Uniformly sample $(S_j, S_k)$ with $|j - k| < d$
    $i \leftarrow S_j, c \leftarrow S_k$
    **if** $\gamma = -1$ **then** uniformly sample $c$ from $1 : L + U$
**else**
    **if** $\gamma = +1$ **then**
        Uniformly sample $(i, c)$ with $y_i = y_c$
    **else**
        Uniformly sample $(i, c)$ with $y_i \neq y_c$
    **end if**
**end if**
**return** $(i, c, \gamma)$
```

label of the instance. More specifically, the probability of predicting the label y is written as:

$$p(y|\mathbf{x}, \mathbf{e}) = \frac{\exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_{y'}}, \quad (4)$$

where $[\cdot, \cdot]$ denotes concatenation of two row vectors, the super script $\mathbf{h}^T$ denotes the transpose of vector $\mathbf{h}$, and $\mathbf{w}$ represents the model parameter.

Combined with Eq. (3), the loss function of transductive learning is defined as:

$$-\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|\mathbf{x}_i, \mathbf{e}_i) - \lambda \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i),$$

where the first term is defined by Eq. (4), and λ is a constant weighting factor. The first term is the loss function of class label prediction and the second term is the loss function of context prediction. This formulation is transductive because the prediction of label y depends on the embedding $\mathbf{e}$, which can only be learned for instances observed in the graph A during training time.

3.3. Inductive Formulation

While we consider transductive learning in the above formulation, in many cases, it is desirable to learn a classifier that can generalize to unobserved instances, especially for large-scale tasks. For example, machine reading systems (Carlson et al., 2010) very frequently encounter novel entities on the Web and it is not practical to train a semi-supervised learning system on the entire Web. However, since learning graph embeddings is transductive in nature, it is not straightforward to do it in an inductive setting. Perozzi et al. (2014) addressed this issue by retraining the embeddings incrementally, which is time consuming and does not scale (and not inductive essentially).

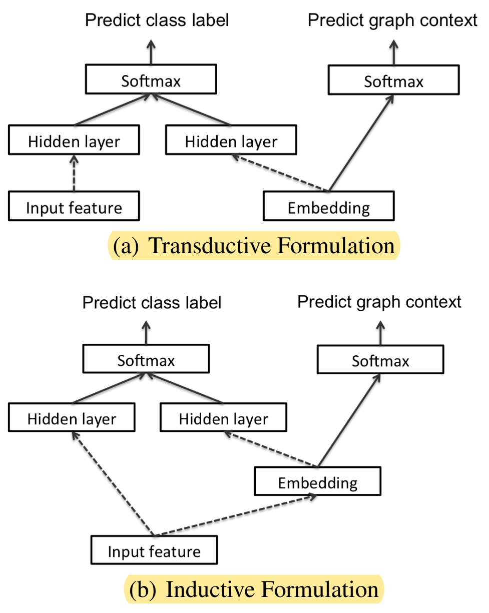


Figure 2. Network architecture: transductive v.s. inductive. Each dotted arrow represents a feed-forward network with an arbitrary number of layers (we use only one layer in our experiments). Solid arrows denote direct connections.

To make the method inductive, the prediction of label y should only depend on the input feature vector $\mathbf{x}$. Therefore, we define the embedding $\mathbf{e}$ as a parameterized function of feature $\mathbf{x}$, as shown in Figure 2(b). Similar to the transductive formulation, we apply k layers on the input feature vector $\mathbf{x}$ to obtain $\mathbf{h}^k(\mathbf{x})$. However, rather than using a "free" embedding, we apply l_1 layers on the input feature vector $\mathbf{x}$ and define it as the embedding $\mathbf{e} = \mathbf{h}^{l_1}(\mathbf{x})$. Then another l_2 layers are applied on the embedding $\mathbf{h}^{l_2}(\mathbf{e}) = \mathbf{h}^{l_2}(\mathbf{h}^{l_1}(\mathbf{x}))$, denoted as $\mathbf{h}^l(\mathbf{x})$ where $l = l_1 + l_2$. The embedding $\mathbf{e}$ in this formulation can be viewed as a hidden layer that is a parameterized function of the feature $\mathbf{x}$.

With the above formulation, the label y only depends on the feature $\mathbf{x}$. More specifically,

$$p(y|\mathbf{x}) = \frac{\exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{x})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{x})^T]\mathbf{w}_{y'}} \quad (5)$$

Replacing $\mathbf{e}_i$ in Eq. (3) with $\mathbf{h}^{l_1}(\mathbf{x}_i)$, the loss function of inductive learning is

$$-\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|\mathbf{x}_i) - \lambda \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{h}^{l_1}(\mathbf{x}_i))$$

where the first term is defined by Eq. (5).

Algorithm 2 Model Training (Transductive)

```
Input: $A$, $\mathbf{x}_{1:L+U}$, $y_{1:L}$, $\lambda$, batch iterations $T_1, T_2$ and sizes $N_1, N_2$
repeat
  for $t \leftarrow 1$ to $T_1$ do
    Sample a batch of labeled instances $i$ of size $N_1$
    $\mathcal{L}_s = -\frac{1}{N_1}\sum_i p(y_i|\mathbf{x}_i, \mathbf{e}_i)$
    Take a gradient step for $\mathcal{L}_s$
  end for
  for $t \leftarrow 1$ to $T_2$ do
    Sample a batch of context from $p(i, c, \gamma)$ of size $N_2$
    $\mathcal{L}_u = -\frac{1}{N_2}\sum_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i)$
    Take a gradient step for $\mathcal{L}_u$
  end for
until stopping
```

Table 2. Dataset statistics.

Dataset	#classes	#nodes	#edges
Citeseer	6	3,327	4,732
Cora	7	2,708	5,429
Pubmed	3	19,717	44,338
DIEL	4	4,373,008	4,464,261
NELL	210	65,755	266,144

3.4. Training

We adopt stochastic gradient descent (SGD) (Bottou, 2010) to train our model in the mini-batch mode. We first sample a batch of labeled instances and take a gradient step to optimize the loss function of class label prediction. We then sample a batch of context (i, c, γ) and take another gradient step to optimize the loss function of context prediction. We repeat the above procedures for T_1 and T_2 iterations respectively to approximate the weighting factor λ. Algorithm 2 illustrates the SGD-based training algorithm for the transductive formulation. Similarly, we can replace $p(y_i|\mathbf{x}_i, \mathbf{e}_i)$ with $p(y_i|\mathbf{x}_i)$ in $\mathcal{L}_s$ to obtain the training algorithm for the inductive formulation. Let θ denote all model parameters. We update both embeddings $\mathbf{e}$ and parameters θ in transductive learning, and update only parameters θ in inductive learning. Before the joint training procedure, we apply a number of training iterations that optimize the unsupervised loss $\mathcal{L}_u$ alone and use the learned embeddings $\mathbf{e}$ as initialization for joint training.

4. Experiments

In our experiments, *Planetoid-T* and *Planetoid-I* denote the transductive and inductive formulation of our approach. We compare our approach with label propagation (LP) (Zhu et al., 2003), semi-supervised embedding (SemiEmb) (Weston et al., 2012), manifold regularization (ManiReg) (Belkin et al., 2006), TSVM (Joachims, 1999), and graph embeddings (GraphEmb) (Perozzi et al., 2014). Another baseline method, denoted as *Feat*, is a linear softmax model that takes only the feature vectors $\mathbf{x}$ as input. We also derive a variant *Planetoid-G* that learns embeddings to jointly predict class labels and graph context without use of feature vectors. The architecture of Planetoid-G is similar to Figure 2(a) except that the *input feature* and the corresponding hidden layers are removed. Among the above methods, LP, GraphEmb and Planetoid-G do not use the features $\mathbf{x}$, while TSVM and Feat do not use the graph A. We include these methods into our experimental settings to better evaluate our approach. Our preliminary experiments on the text classification datasets show that the performance of our model is not very sensitive to specific choices of the network architecture[2]. We adapt the implementation of GraphEmb[3] to our Skipgram implementation. We use the Junto library (Talukdar & Crammer, 2009) for label

Table 3. Accuracy on text classification. Upper rows are inductive methods and lower rows are transductive methods.

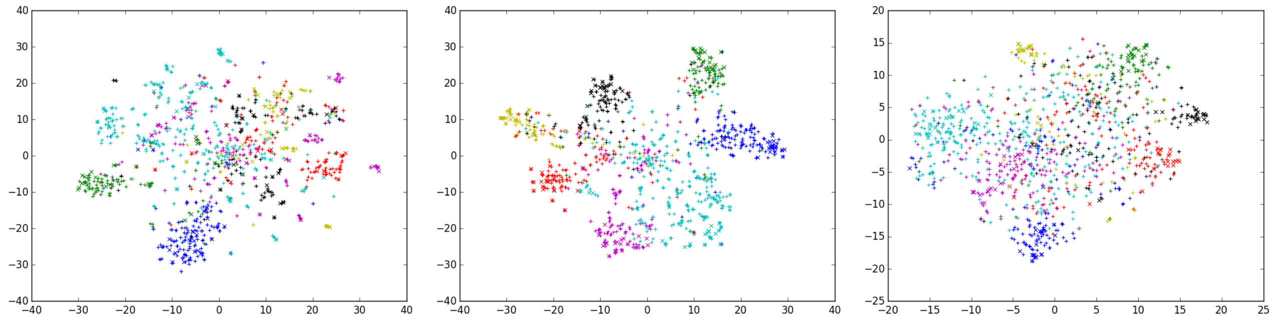
Method	Citeseer	Cora	Pubmed
Feat	0.572	0.574	0.698
ManiReg	0.601	0.595	0.707
SemiEmb	0.596	0.590	0.711
Planetoid-I	**0.647**	0.612	**0.772**
TSVM	0.640	0.575	0.622
LP	0.453	0.680	0.630
GraphEmb	0.432	0.672	0.653
Planetoid-G	0.493	0.691	0.664
Planetoid-T	0.629	**0.757**	0.757

Table 4. Recall@k on DIEL distantly-supervised entity extraction. Upper rows are inductive methods and lower rows are transductive methods. Results marked with $*$ are taken from the original DIEL paper (Bing et al., 2015) with the same data splits.

Method	Recall@k
*Feat	0.349
ManiReg	0.477
SemiEmb	0.486
Planetoid-I	**0.501**
*DIEL	0.405
*LP	0.162
GraphEmb	0.258
Planetoid-G	0.394
Planetoid-T	**0.500**
*Upper Bound	0.617

[2]We note that it is possible to develop other architectures for different applications, such as using a shared hidden layer for feature vectors and embeddings.

[3]https://github.com/phanein/deepwalk



(a) GraphEmb (b) Planetoid-T (c) SemiEmb

Figure 3. t-SNE Visualization of embedding spaces on the Cora dataset. Each color denotes a class.

Table 5. Accuracy on NELL entity classification with labeling rates of 0.1, 0.01, and 0.001. Upper rows are inductive methods and lower rows are transductive methods.

METHOD	0.1	0.01	0.001
FEAT	0.621	0.404	0.217
MANIREG	0.634	0.413	0.218
SEMIEMB	0.654	0.438	0.267
PLANETOID-I	0.702	0.598	0.454
LP	0.714	0.448	0.265
GRAPHEMB	0.795	0.725	0.581
PLANETOID-G/T	**0.845**	**0.757**	**0.619**

propagation, and SVMLight[4] for TSVM. We also use our own implementation of ManiReg and SemiEmb by modifying the symbolic objective function in Planetoid. In all of our experiments, we set the model hyper-parameters to $r_1 = 5/6$, $q = 10$, $d = 3$, $N_1 = 200$ and $N_2 = 200$ for Planetoid. We use the same r_1, q and d for GraphEmb, and the same N_1 and N_2 for ManiReg and SemiEmb. We tune r_2, T_1, T_2, the learning rate and hyper-parameters in other models based on an additional data split with a different random seed.

The statistics for five of our benchmark datasets are reported in Table 2. For each dataset, we split all instances into three parts, labeled data, unlabeled data, and test data. Inductive methods are trained on the labeled and unlabeled data, and tested on the test data. Transductive methods, on the other hand, are trained on the labeled, unlabeled data, and test data without labels.

4.1. Text Classification

We first considered three text classification datasets[5], Citeseer, Cora and Pubmed (Sen et al., 2008). Each dataset contains bag-of-words representation of documents and citation links between the documents. We treat the bag-of-words as feature vectors $\mathbf{x}$. We construct the graph A based on the citation links; if document i cites j, then we set $a_{ij} = a_{ji} = 1$. The goal is to classify each document into one class. We randomly sample 20 instances for each class as labeled data, $1,000$ instances as test data, and the rest are used as unlabeled data. The same data splits are used for different methods, and we compute the average accuracy for comparison.

The experimental results are reported in Table 3. Among the inductive methods, Planetoid-I achieves the best performance on all the three datasets with the improvement of up to 6.1% on Pubmed, which indicates that our embedding techniques are more effective than graph Laplacian regularization. Among the transductive methods, Planetoid-T achieves the best performance on Cora and Pubmed, while TSVM performs the best on Citeseer. However, TSVM does not perform well on Cora and Pubmed. Planetoid-I slightly outperforms Planetoid-T on Citeseer and Pubmed, while Planetoid-T gets up to 14.5% improvement over Planetoid-I on Cora. We conjecture that in Planetoid-I, the feature vectors impose constraints on the learned embeddings, since they are represented by a parameterized function of the input feature vectors. If such constraints are appropriate, as is the case on Citeseer and Pubmed, it improves the non-convex optimization of embedding learning and leads to better performance. However, if such constraints rule out the optimal embeddings, the inductive model will suffer.

Planetoid-G consistently outperforms GraphEmb on all three datasets, which indicates that joint training with label information can improve the performance over training the supervised and unsupervised objectives separately. Figure 3 displays the 2-D embedding spaces on the Cora dataset using t-SNE (Van der Maaten & Hinton, 2008). Note that different classes are better separated in the embedding space of Planetoid-T than that of GraphEmb and

[4]http://svmlight.joachims.org/

[5]http://linqs.umiacs.umd.edu/projects//projects/lbc/

SemiEmb, which is consistent with our empirical findings. We also observe similar results for the other two datasets.

4.2. Distantly-Supervised Entity Extraction

We next considered the DIEL (Distant Information Extraction using coordinate-term Lists) dataset (Bing et al., 2015). The DIEL dataset contains pre-extracted features for each entity mention in text, and a graph that connects entity mentions to coordinate lists. The goal is to extract medical entities from text given feature vectors and the graph.

We follow the exact experimental setup as in the original DIEL paper (Bing et al., 2015), including data splits of different runs, preprocessing of entity mentions and coordinate lists, and evaluation. We treat the top-k entities given by a model as positive instances, and compute recall@k for evaluation (k is set to $240,000$ following the DIEL paper). We report the average result of 10 runs in Table 4, where *Feat* refers to a result obtained by SVM (referred to as DS-Baseline in the DIEL paper). The result of LP was also taken from (Bing et al., 2015). *DIEL* in Table 4 refers to the method proposed by the original paper, which is an improved version of label propagation that trains classifiers on feature vectors based on the output of label propagation. We did not include TSVM into the comparison since it does not scale. Since we use Freebase as ground truth and some entities are not present in text, the upper bound of recall as shown in Table 4 is 0.617.

Both Planetoid-I and Planetoid-T significantly outperform all other methods. Each of Planetoid-I and Planetoid-T achieves the best performance in 5 out of 10 runs, and they give a similar recall on average, which indicates that there is no significant difference between these two methods on this dataset. Planetoid-G clearly outperforms GraphEmb, which again shows the benefit of joint training.

4.3. Entity Classification

We sorted out an entity classification dataset from the knowledge base of Never Ending Language Learning (NELL) (Carlson et al., 2010) and a hierarchical entity classification dataset (Dalvi & Cohen, 2016) that links NELL entities to text in ClueWeb09. We extracted the entities and the relations between entities from the NELL knowledge base, and then obtained text description by linking the entities to ClueWeb09. We use text bag-of-words representation as feature vectors of the entities.

We next describe how to construct the graph based on the knowledge base. We first remove relations that are not populated in NELL, including "generalizations", "haswikipediaurl", and "atdate". In the knowledge base, each relation is denoted as a triplet (e_1, r, e_2), where e_1, r, e_2 denote head entity, relation, and tail entity respectively. We treat each entity e as a node in the graph, and each relation r is split as two nodes r_1 and r_2 in the graph. For each (e_1, r, e_2), we add two edges in the graph, (e_1, r_1) and (e_2, r_2).

We removed all classes with less than 10 entities. The goal is to classify the entities in the knowledge base into one of the 210 classes given the feature vectors and the graph. Let β be the labeling rate. We set β to 0.1, 0.01, and 0.001. $\max(\beta N, 1)$ instances are labeled for a class with N entities, so each class has at least one entity in the labeled data.

We report the results in Table 5. We did not include TSVM since it does not scale to such a large number of classes with the one-vs-rest scheme. Adding feature vectors does not improve the performance of Planetoid-T, so we set the feature vectors for Planetoid-T to be all empty, and therefore Planetoid-T is equivalent to Planetoid-G in this case.

Planetoid-I significantly outperforms the best of the other compared inductive methods—i.e., SemiEmb—by 4.8%, 16.0%, and 18.7% respectively with three labeling rates. As the labeling rate decreases, the improvement of Planetoid-I over SemiEmb becomes more significant.

Graph structure is more informative than features in this dataset, so inductive methods perform worse than transductive methods. Planetoid-G outperforms GraphEmb by 5.0%, 3.2% and 3.8%.

5. Conclusion

Our contribution is three-fold: a) incontrast to previous semi-supervised learning approaches that largely depend on graph Laplacian regularization, we propose a novel approach by joint training of classification and graph context prediction; b) since it is difficult to generalize graph embeddings to novel instances, we design a novel inductive approach that conditions embeddings on input features; c) we empirically show substantial improvement over existing methods (up to 8.5% and on average 4.1%), and even more significant improvement in the inductive setting (up to 18.7% and on average 7.8%).

Our experimental results on five benchmark datasets also show that a) joint training gives improvement over unsupervised learning; b) predicting graph context is more effective than graph Laplacian regularization; c) the performance of the inductive variant depends on the informativeness of feature vectors.

One direction of future work would be to apply our framework to more complex networks, including recurrent networks. It would also be interesting to experiment with datasets where a graph is computed based on distances between feature vectors.

Acknowledgements

This work was funded by the NSF under grants CCF-1414030 and IIS-1250956, and by Google.

References

Belkin, Mikhail, Niyogi, Partha, and Sindhwani, Vikas. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

Bing, Lidong, Chaudhari, Sneha, Wang, Richard C, and Cohen, William W. Improving distant supervision for information extraction using label propagation through lists. In *EMNLP*, 2015.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *NIPS*, pp. 2787–2795, 2013.

Bottou, Léon. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pp. 177–186. Springer, 2010.

Carlson, Andrew, Betteridge, Justin, Kisiel, Bryan, Settles, Burr, Hruschka Jr, Estevam R, and Mitchell, Tom M. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, pp. 3, 2010.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

Dalvi, Bhavana and Cohen, William W. Hierarchical semi-supervised classification with incomplete class hierarchies. In *WSDM*, 2016.

Handcock, Mark S, Raftery, Adrian E, and Tantrum, Jeremy M. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

Ji, Ming, Sun, Yizhou, Danilevsky, Marina, Han, Jiawei, and Gao, Jing. Graph regularized transductive classification on heterogeneous information networks. In *Machine Learning and Knowledge Discovery in Databases*, pp. 570–586. Springer, 2010.

Joachims, Thorsten. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pp. 200–209, 1999.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *NIPS*, pp. 3111–3119, 2013.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. *EMNLP*, 12:1532–1543, 2014.

Perozzi, Bryan, Al-Rfou, Rami, and Skiena, Steven. Deepwalk: Online learning of social representations. In *KDD*, pp. 701–710, 2014.

Sen, Prithviraj, Namata, Galileo, Bilgic, Mustafa, Getoor, Lise, Galligher, Brian, and Eliassi-Rad, Tina. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

Snijders, Tom AB and Nowicki, Krzysztof. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification*, 14(1):75–100, 1997.

Talukdar, Partha Pratim and Crammer, Koby. New regularized algorithms for transductive learning. In *Machine Learning and Knowledge Discovery in Databases*, pp. 442–457. Springer, 2009.

Tang, Jian, Qu, Meng, Wang, Mingzhe, Zhang, Ming, Yan, Jun, and Mei, Qiaozhu. Line: Large-scale information network embedding. In *WWW*, pp. 1067–1077, 2015.

Tian, Fei, Gao, Bin, Cui, Qing, Chen, Enhong, and Liu, Tie-Yan. Learning deep representations for graph clustering. In *AAAI*, pp. 1293–1299, 2014.

Van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

Weston, Jason, Ratle, Frédéric, Mobahi, Hossein, and Collobert, Ronan. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

Wijaya, Derry, Talukdar, Partha Pratim, and Mitchell, Tom. Pidgin: ontology alignment using web text as interlingua. In *CIKM*, pp. 589–598, 2013.

Yang, Zhilin, Salakhutdinov, Ruslan, and Cohen, William. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

Zhou, Dengyong, Bousquet, Olivier, Lal, Thomas Navin, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. *NIPS*, 16(16):321–328, 2004.

Zhu, Xiaojin, Ghahramani, Zoubin, Lafferty, John, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pp. 912–919, 2003.