Semi-Supervised Learning on Graphs Based on Local Label Distributions

Evgeniy Faerman, Felix Borutta, Julian Busch, Matthias Schubert
Ludwig-Maximilians-Universität München
Munich, Germany
{faerman,borutta,busch,schubert}@dbs.ifi.lmu.de

ABSTRACT

Most approaches that tackle the problem of node classification consider nodes to be similar, if they have shared neighbors or are close to each other in the graph. Recent methods for attributed graphs additionally take attributes of neighboring nodes into account. We argue that the class labels of the neighbors bear important information and considering them helps to improve classification quality. Two nodes which are similar based on class labels in their neighborhood do not need to be close-by in the graph and may even belong to different connected components. In this work, we propose a novel approach for the semi-supervised node classification. Precisely, we propose a new node embedding which is based on the class labels in the local neighborhood of a node. We show that this is a different setting from attribute-based embeddings and thus, we propose a new method to learn label-based node embeddings which can mirror a variety of relations between the class labels of neighboring nodes. Our experimental evaluation demonstrates that our new methods can significantly improve the prediction quality on real world data sets.

KEYWORDS

Feature learning, Graph representations, Node embeddings, Node classification

ACM Reference Format:
Evgeniy Faerman, Felix Borutta, Julian Busch, Matthias Schubert. 2018. Semi-Supervised Learning on Graphs Based on Local Label Distributions. In *ACM Workshop@SIGKDD (MLG'18).* ACM, London, UK, 8 pages.

1 INTRODUCTION

Graphs are the most general way to represent structured data. In general, a set of entities with some given pairwise relationships between them can be modeled as a graph $G = (V, E)$ with a corresponding node set V and an edge set $E \subseteq V \times V$. Real-world examples of graph-structured data are abundant and include social networks, co-citation networks or biological networks.

In addition to the graph structure, further attribute information may be provided for the entities described by the graph nodes. In an attributed graph, each node $v_i \in V$ is associated with an attribute vector $f_i \in \mathbb{R}^d$. For instance, social network users might be enriched with personal information or documents in a co-citation network might be described by bag-of-words vectors. The increasing relevance of graph-structured data has been accompanied by an increased interest in learning algorithms which can leverage underlying graph structure to make accurate predictions for the modeled entities.

An important semi-supervised learning task on graphs is node classification, where each node $v_i \in V$ can be associated with a set of class labels (simply referred to as labels in the following) represented by a label vector $y_i \in \{0, 1\}^l$ where l is the number of possible labels. Given a set of already labeled nodes in a graph, the goal is to predict new likely labels for unlabeled nodes. The task is semi-supervised in the sense that connectivity information about the whole graph is available and at least some of the class labels are already known. In the case of attributed graphs, attributes of all nodes can additionally be used for prediction, including those of the unlabeled nodes in the graph. Important applications include recommendation in social networks, where the node labels represent user interests, or document classification in co-citation networks, where the node labels indicate associated fields of research.

Approaches for node classification on graphs may employ additional node attributes or operate on the graph structure alone. We will refer to these approaches as *attribute-based* and *connectivity-based* approaches, respectively. Among the most successful connectivity-based methods are node embedding techniques [7, 9, 13, 16, 17, 21, 32, 33, 37, 40]. An underlying assumption of these techniques is that nodes which are closely connected in the graph, should have similar labels, which is commonly referred to as homophily [25]. Our method does not rely on the homophily assumption, but is still able to relate close-by nodes. Furthermore, unlike most node embedding techniques our new approach can be used to classify nodes unseen during training. In the attribute-based setting the graph structure can be incorporated in different ways, for instance by using regularization [5, 41, 44, 45], combining attributes with node embeddings [42], or aggregating them over local neighborhoods [4, 8, 12, 20, 23, 26, 27, 39]. While regularization-based methods rely on the homophily assumption and most of them are not able to classify instances unseen during training, all other methods focus on node attributes. In addition to connectivity and node attributes, the labels available during training further provide valuable information that is in general complementary to connectivity and attribute features, and are useful to improve classification. In general learning tasks on independent and identically distributed (iid) data, labels indicate that an observation is sampled from a particular distribution. However, in a graph we have non-iid data and thus, the labels of connected objects allow for a novel use of label information which has not been exploited before for learning graph embeddings.

MLG'18, August 2018, London, UK

.

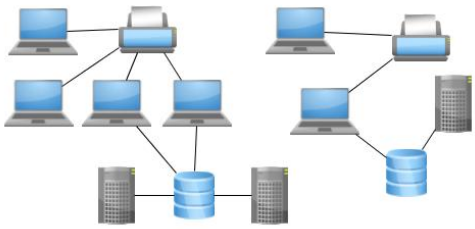


Figure 1: Consider a communication network with nodes labeled according to their device type (user, server, database, printer). Assume the labels for the database and printer nodes in the right connected component are unknown while the remaining labels are provided. Further node attributes are not given. We can observe that the roles of printers and databases are clearly defined by the labels of their neighboring nodes, e.g., printers are not connected to server nodes. Homophily-based methods would fail to classify these nodes correctly, since their labels differ from their neighbors. Further, connectivity alone does not explain the roles, since for instance the printer and database in the left part of the graph have the same degree and even their neighbors have the same degrees.

In this paper, we propose a *label-based* approach to learn a node embedding which allows for more accurate node classifications. The main idea of our approach is that there often exists a correlation between the labels of a node and the distribution of labels in its local neighborhood. Thus, considering the local label distribution when computing a node embedding can exploit this correlation to improve the descriptiveness of the learned embedding. In Figure 1, we illustrate this for a typical case for which the label of a node is determined by the labels of neighboring nodes and not by node attributes or connectivity. As an additional example, the function of a protein can be expected to correlate strongly with the functions of interacting proteins. As mentioned above, we assume that the labels of at least some of the neighboring nodes are known for each new node with unknown labels. In the majority of applications, this is realistic because new nodes usually connect to already known parts of the network. For instance, new papers usually cite established articles and new members of a social network will usually already know multiple friends in the network to connect to.

Though labels can be considered as another type of node attributes, there exists an important difference between labels and attributes which prevents attribute-based embeddings to generalize well on label information. Though the attribute values of the predicted node are allowed to be used for learning the embedding, using the node labels even in an transitive way leads to overfitting and a bad generalization performance of the learned embedding. We will discuss these issues in more detail in Section 3 and introduce a simple baseline method. In our new method, we aggregate labels from relevant nodes directly and thus, we can completely exclude any influence of the nodes' own labels. In a first step, we determine the relevant neighbors of a given node based on *Approximate Personalized PageRank (APPR)*. Since this might be an expensive task for large graphs, we use an adaption of the highly efficient algorithm from [36]. After determining the neighborhood, we compute the label distribution within the neighborhood and classify the node based on this novel representation. We compare our new representation to state-of-the-art graph embeddings based on several benchmark datasets for node classification.

The remainder of the paper is structured as follows: After providing a formal problem definition for our approach in Section 2, we introduce our new method in Section 3, starting with a discussion on the possibility of incorporating label-based features into existing models in Section 3.1. After a discussion of related work in Section 4, the performance of our model is evaluated experimentally and compared to state-of-the-art methods in Section 5. Finally, Section 6 concludes the paper and proposes directions for future work.

2 PROBLEM SETTING

We consider (possibly directed) graphs $G = (V, E)$, with node set $V = \{v_1, \ldots, v_n\}$ and edge set $E \subseteq V \times V$. A graph can be represented by an $n \times n$ adjacency matrix $\mathbf{A} = (a_{ij})_{v_i, v_j \in V}$, where $a_{ij} \in \mathbb{R}$ denotes the weight of the edge (v_i, v_j). In case of an unweighted graph, $a_{i,j} = 1$ indicates the existence and $a_{i,j} = 0$ the absence of an edge between v_i and v_j. Furthermore, we do not allow self-links, i.e., $a_{i,i} = 0$ for all nodes $v_i \in V$. In an attributed graph, additional node attributes are provided in the form of an attribute vector $f_i \in \mathbb{R}^d$ for each node v_i. The attribute information for the whole graph can be represented by an $n \times d$ attribute matrix $\mathbf{F}$, where the ith row of $\mathbf{F}$ corresponds to v_i's attribute vector f_i. Let us note that an important difference between attributes and labels is that attributes are usually known for all nodes, in particular those nodes without known labels.

Our problem setting is *semi-supervised node classification*, where the node set V is partitioned into a set of labeled nodes L and unlabeled nodes U, such that $V = L \cup U$ and $L \cap U = \emptyset$. Thereby, each node $v_i \in V$ is associated with a label vector $y_i \in \{0, 1\}^l$, where l is the number of possible labels and an entry one indicates the presence of the corresponding label for a certain node. The labels available for training can be represented by an $n \times l$ label matrix $\mathbf{Y}_{train}$, where the i'ths row of $\mathbf{Y}_{train}$ corresponds to the label vector y_i of v_i if $v_i \in L$. For unlabeled nodes, we assign constant zero vectors. The task is now to train a classifier using $\mathbf{A}$, $\mathbf{Y}_{train}$ and possibly $\mathbf{F}$ which accurately predicts y_i for each $v_i \in U$. In *multi-class* classification, each node is assigned to exactly one class, such that $y_i = e_j$ is the j's unit vector, if v_i is assigned to class j. *Multi-label* classification denotes the general case, in which each node may be assigned to one or more classes and the goal is to predict all labels assigned to a particular node.

3 SEMI-SUPERVISED LEARNING ON GRAPHS BASED ON LOCAL LABEL DISTRIBUTION

3.1 Labels as Attributes

The main idea of our approach is to learn a more descriptive node representation by incorporating the known labels in the neighborhood of a node. In the following, we will show why existing methods are not suitable to consider this information. Methods relying on neighborhood similarity [7, 9, 13, 16, 32, 37, 40] learn representations in an unsupervised manner and thus, only rely on the topology of the graph and not on attributes or labels. The *Planetoid-T* model [42] considers labels by partly enforcing the similarity between members of the same class and therefore, nodes are related to each other based only on their own labels.

Graph Neural Networks [24, 34] or Graph Convolution Networks (GCN) [4, 8, 12, 20, 23, 26, 27, 39] are special cases of a Message Passing Neural Network (MPNN) [14] which is a framework describing a family of neural network based models for attributed graphs. All MPNN methods have in common that they use some differentiable function to iteratively compute messages for each node which are passed to all its neighbors. These messages build an input to a differentiable update function which computes new node representations h:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t),$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}).$$

Here t denotes the current iteration, h_v^t is the representation of node v in iteration t and vector h_v^0 corresponds to the input features of node v. $N(v)$ denotes the set of direct neighbors of node v, M_t is the message and U_t the set of update functions. The obvious way to integrate the neighborhood label information into an MPNN-based prediction model is to include the label information into the messages directed to the neighbors in the first iteration. However, even after removing self-links each node would receive information about its own labels already in the second iteration during training. Thus, models learned on such representations overfit on the nodes' own labels and do not generalize well in the inference step where the node labels are unknown. The same applies to directed graphs with cycles. Therefore, applying MPNN models to communicate neighboring labels is restricted to one iteration only. We use a corresponding model as a baseline for our experiments.

Note that this problem does not apply to label diffusion algorithms [15, 19, 44, 45]. However, these methods infer node labels based on majority vote in local neighborhoods and do not make use of recurrent patterns in graph.

3.2 General Approach

To present our method for semi-supervised learning on graphs using local label distributions we first outline an efficient algorithm for computing node neighborhoods based on *Approximated Personalized PageRank* (APPR). Afterwards, we describe how to create node representations based on the label distribution in the local neighborhood based on APPR. Finally, the node representations can be used as feature descriptors in arbitrary classification models.

The *Personalized PageRank* (PPR) corresponds to the *PageRank* algorithm [31], where the probabilities in the starting vector s are biased towards some set of nodes. The result is the "importance" of all nodes in the graph from the viewpoint of the nodes in s.

The *push* algorithm described in [18] and [6] is an efficient way to compute an approximation of the *Personalized PageRank* (APPR) vector if the start distribution vector s is sparse. The idea behind the *push* algorithm is only to consider a node in the local neighborhood if the probability to visit the node is significantly larger than the probability to visit any other node from the rest of the graph. This leads to a sparse solution meaning that only relatively few nodes of the underlying graph are contained in the resulting APPR vector.

Algorithm 1 describes the computation of APPR using a variant of the *push* operation on lazy random walk transition matrices of undirected unweighted graphs. This algorithm was proposed in [3], where APPR is used to partition graphs. We describe an adapted version from [36] which converges faster. The algorithm maintains two vectors: the solution vector p and a residual vector r. The vector p is the current approximation of the PPR vector and vector r contains the approximation error or the not yet distributed probability mass. $p(u)$ and $r(u)$ are the entries in vectors p and r corresponding to node u, $d(u)$ is the degree of node u. In each iteration the algorithm selects a node with sufficient probability mass in vector r. This probability mass is spread between the node entry in p and the entries of its direct neighbors in r. In each step, the exact PPR is the linear combination of the current solution vector p and the PPR solution for r, i.e., $pr(s) = p + pr(r)$. The algorithm can also be trivially adapted to directed graphs and graphs with weighted edges.

Algorithm 1 ApproximatePPR

Input: Starting vector s, Teleportation probability α, Approximation threshold ϵ
Output: APPR vector p

```
p = 0⃗, r = s
while r(u) ≥ ε d(u) for some vertex u do
    pick any u where r(u) ≥ ε d(u)
    push(u)
end while
return p
```

Algorithm 2 push

Input: Vertex u

```
p(u) = p(u) + (2α/(1 + α))r(u)
for v with (u, v) ∈ E do
    r(v) = r(v) + ((1 − α)/(1 + α))r(u)/d(u)
end for
r(u) = 0
```

3.2.1 Local Label Distribution. In our approach we first compute the APPR vector for each node. Before APPR is computed for node v, the corresponding entry $s(v)$ in starting vector s is set to one and all other entries to zero. Therefore, the APPR vector of v describes the importance of local neighbors only from its point of view.

In the APPR result matrix $\mathbf{APPR}$, each row corresponds to the APPR vector of the corresponding node. The local label distribution representation $\mathbf{X} \in \mathbb{R}^{n \times l}$ is computed by manipulating $\mathbf{APPR}$ such that the diagonal is set to zero to exlude information about the own labels and then multiplying the resulting matrix $\widehat{\mathbf{APPR}}$ with the label matrix $\mathbf{Y}_{train}$. The entry $\mathbf{X}_{v\, y_j}$ can be interpreted as the probability that a random walk starting from node v stops at a neighbor with label y_j.

The local label distribution can be used as a node embedding vector which can be passed into an arbitrary classification algorithm. In our experiments, we employ a multi-layer perceptron with three layers, i.e., an input layer taking the local label distribution matrix $\mathbf{X}$ as input, a dense hidden layer with 16 units and an ReLU activation, and finally a dense layer retrieving the output. Formally, the hidden layer H can be described as

$$H = ReLU(\widehat{\mathbf{APPR}} \cdot \mathbf{Y}_{train} \cdot \mathbf{W}_1)$$

with $\mathbf{W}_1$ denoting the weight matrix. Note that the bias is omitted for the sake of better readability.

4 RELATED WORK

Numerous approaches for semi-supervised learning on graphs have been proposed recently. These can be categorized into unsupervised node embedding techniques and semi-supervised techniques.

4.1 Unsupervised Node Embedding

Lots of recent developments related to learning from structural relationships have focussed on learning *node embeddings*, where a latent vector representation is learned for each node, reflecting its connectivity in the underlying graph. The learned node embeddings can be used as an input to a subsequent down-stream task, such as node classification. Random walk based methods [16, 32] sample a number of random walks from the graph and nodes are related if they have common neighbors. *LINE* [37] is another variant, which considers direct first- and second-order proximities instead of random walks. *Graph2Gauss* [7] learns similarity to hop neighborhoods and embeds each node as a Gaussian distribution to allow for uncertainty in the representation. *GECS* [2] uses connections subgraphs to determine appropriate node neighborhood. More closely related to our approach, *LASAGNE* [13] relies on APPR to determine relevant context nodes. Other works perform matrix factorization. For instance, *GraRep* [9] factorizes a sequence of k-step log-probability matrices with SVD and concatenates the resulting low-dimensional node representations to form the final representations. Abu-El-Haija et al. propose matrix factorization of random-walk occurrence matrix with different approaches to determine context window size distribution [1]. *SDNE* [40] uses a multi-layer auto-encoder model to capture non-linear structures based on direct first- and second-order proximities. Authors of [10] propose embeddings in hyperbolic space. *HARP* [11] addresses the local minima problem and introduce an iterative scheme for learning of node representations which can be used with different embedding learning methods. An input graph is coarsened on different levels and node representations are learned starting with the coarsest graph and learned embeddings are provided as initializations for the embeddings of subsequent finer graphs. While the above methods rely on the homophily assumption, *struc2vec* [33] aims at learning representations which relate structurally similar nodes instead of nodes which are close in the graph. It does so by using degree sequences in neighborhoods of different sizes. All of the above approaches are *transductive* in the sense that labels can only be predicted for unlabeled nodes observed already at training time. The *GraphSAGE* [17] framework introduces *inductive* node embeddings. The basic idea is to learn an embedding function by sampling and aggregating node attributes in local neighborhoods. The embedding function can further be learned with a supervised loss function. Inductive models are also obtained by considering node attributes. *Variational Graph Auto-Encoders* [21] learn node representations using a variational auto-encoder, where the encoder is a two-layer GCN. The model can be applied to attributed and non-attributed graphs.

4.2 Semi-Supervised Learning on Graphs

Compared to separately optimizing steps in a semi-supervised learning pipeline, as is the case for semi-supervised learning with pretrained node embeddings, end-to-end training usually leads to better performance on the supervised learning objective.

One direction is *Laplacian Regularization*, where the prediction loss is augmented with an unsupervised loss function based on the graph's Laplacian matrix, encoding the homophily assumption that close-by nodes should have the same label. Related approaches include *Manifold Regularization* [5], a kernel-based method, and *Deep Semi-Supervised Embedding* [41] which incorporates node embeddings by augmenting neural network models with an embedding layer. Both of these methods generalize to attributed graphs. The *ICA* algorithm [30] starts with the observed labels and iteratively classifies unlabeled nodes based on aggregated node attributes in local neighborhoods. At the end of each phase, the nodes classified with highest certainty are added to the ground truth for the next phase. *Label Diffusion* methods [15, 19, 44, 45] are more closely related to our work. Similarly to our method they create embeddings based on labels in local neighborhoods. The basic idea is based on mincuts [15] and labels are inferred based on majority vote. Therefore, Label Diffusion approaches do not exploit the effect of similar label distributions in a graph. More recent methods, as proposed in [29] and [43], also classify nodes based on labels in local neighborhoods. They learn a model which predicts node labels from a feature vector describing the local k-neighborhood. Both methods assume unattributed graphs.

Instead of imposing regularization, *Planetoid* [42] combines the prediction loss with node embeddings by training a joint model which predicts class labels as well as graph context for a given node. The graph context sampled from random walks as well as the set of nodes with shared labels. This allows Planetoid to relate nodes with similar labels even if they are not close in the graph. Thus, Planetoid does not rely on a strong homophily assumption. In addition to a connectivity-based variant, *Planetoid-G*, the authors propose two further architectures, which incorporate node attributes. The transductive variant *Planetoid-T* starts with pre-trained embeddings and alternately optimizes the prediction and embedding loss functions. The inductive variant *Planetoid-I* on the other hand predicts the graph context from the node features instead.

Another important direction which has recently gained increasing attention is concerned with generalizing deep neural network architectures to graph-structured domains. As the general approach consists of incorporating graph structure into supervised learning, these models assume an attributed graph. However, they can naturally be applied to non-attributed graphs by using the identity matrix as the attribute matrix. The vast majority of neural network based models for semi-supervised learning on graphs can be described within a message-passing framework. In a *Message Passing Neural Network (MPNN)* [14], each node has a hidden state which is updated iteratively during training. The initial hidden state of a node corresponds to its attribute vector. In a first step, messages from v_i's neighborhood are received and aggregated, where a message from neighbor v_j depends on v_i's and v_j's hidden states. In a second step, v_i's state is updated by combining it with the aggregated messages. An important special case are *Graph Convolution Networks* [8, 12, 20, 23, 26, 27, 39] which aggregate node attributes over local neighborhoods with spatially localized filters, similar to classical convolutional networks on images [22]. The *ChebNet* [12] aggregates messages from neighbors analogously to the eigenvectors of the graph's Laplacian matrix. The update function ignores the previous state and applies a non-linear activation. The resulting filters are k-localized. The *GCN* [20] is a simplification of the ChebNet, which only considers one-hop neighbors. Messages are aggregated according to a normalized adjacency matrix. In the update phase, the aggregated messages are multiplied with a learned

filter matrix with a ReLU activation. For graph convolution networks, the number of message passing iterations corresponds to the number of layers.

5 EVALUATION

We evaluate our approach by performing node-label prediction and compare the quality in terms of micro F_1 score for multiclass prediction tasks, respectively micro F_1 and macro F_1 scores for multilabel prediction tasks, against state-of-the-art methods.

For both tasks, we compare our model against the following approaches:

- *Adj*: a baseline approach which learns node embeddings only based on the information contained in the adjacency matrix
- GCN_1*_only_L*: a GCN which applies convolution on label matrix **Y**. We use one convolution layer on the adjacency matrix without self-links, followed by a dense output layer[1]
- *noFeat* GCN_2: the standard 2-layer GCN as published by Kipf et al. [20] without using the node attributes
- *DeepWalk*: the DeepWalk model as proposed in [32]
- *node2vec*: the node2vec model as proposed in [16]
- *Planetoid-G*: the Planetoid variant which does not use attribute information [42] [2]

Our model is denoted as *LD* (short for Label Distribution). For these experiments we train a simple feed-forward neural network which takes the label distribution based representations as input and retrieves class probabilities as output.

Note that we omit the comparison to label propagation [45] since Yang et al. already showed that the *Planetoid* model outperforms this approach [42].

5.1 Multiclass Prediciton

5.1.1 Experimental Setup. For the multiclass label prediciton task we use the following three text classification benchmark graph datasets [28, 35]:

- Cora. The Cora dataset contains 2'708 publications from seven categories in the area of ML. The citation graph consists of 2'708 nodes, 5'278 edges, 1'433 attributes and 7 classes.
- CiteSeer. The CiteSeer dataset contains 3'264 publications from six categories in the area of CS. The citation graph consists of 3'264 nodes, 4'536 edges, 3'703 attributes and 6 classes.
- Pubmed. The Pubmed dataset contains 19'717 publications which are related to diabetes and categorized into 3 classes. The citation graph consists of 19'717 nodes, 44'324 edges, 500 attributes and 3 classes.

For each graph, documents are denoted as nodes and undirected links between documents represent citation relationships. If node attributes are applied, bag-of-words representations are used as attribute vectors for each document.

We split the data as suggested in [42], i.e., for labeled data our training sets contain 20 randomly selected instances per class, the test sets consist of 1'000 instances, and the validation sets contain 500 instances for each method. The remaining instances are used as unlabeled data. For comparison we use the prediction micro F_1 scores which we collected over 10 different data splits.

Since the numbers of iterations for sampling the graph contexts and the label contexts for *Planetoid* are suggested only for the *CiteSeer* data set, we adapted these values relative to the number of nodes for each graph. For *node2vec*, we perform grid searches over the hyperparameters p and q with $p, q \in \{0.25, 0.5, 1.0, 2.0, 4.0\}$ and use window size 10 as proposed by the authors. For all models except *Planetoid* unless otherwise noted, we use one hidden layer with 16 neurons and regularization, learning rate and training procedure as in [20]. Considering our model, we use $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$ as values for the teleportation parameter and $\epsilon = 1e^{-5}$ as approximation threshold to compute the APPR vectors for each node.

We present results computed on the test sets for the best performing hyperparameters. The best performing hyperparameters for all models are determined by using the validation sets.

5.1.2 Results. Figure 2 shows boxplots depicting the micro F_1 scores we achieved for the multiclass prediction task for each considered model on the *Cora*, *CiteSeer* and *Pubmed* networks.

The baseline approach GCN_1*_only_L*, i.e., the one layer GCN model which only uses the label distributions of the neighboring nodes to predict a node's label, shows worst results among the considered models. However, these scores are still promising that the labels may improve the task of learning "good" representations. The baseline method which considers the corresponding rows of the adjacency matrix as node representations, i.e., *Adj*, achieves slightly better results for all three datasets. For the *GCN* and *Planetoid* models that do not make recourse to attribute information, i.e., *noFeat* GCN_2, resp. *Planetoid-G*, the retrieved micro F_1 values are slightly lower than the ones achieved by *DeepWalk* and *node2vec*. Our model improves the results produced by *node2vec*, which means that the label distributions are indeed a useful source of information, although the baseline GCN_1*_only_L* shows, especially for *Pubmed*, rather poor results. This may be reasoned by the fact that this model only considers the label distribution of a very local neighborhood (in fact one hop neighbors). However, collecting the label distribution from a more spacious neighborhood gives a significant boost in terms of prediction accuracy. Indeed the best results for the *LD* approach are reached for $\alpha = 0.1$, which corresponds to a rather spacious neighborhood exploration.

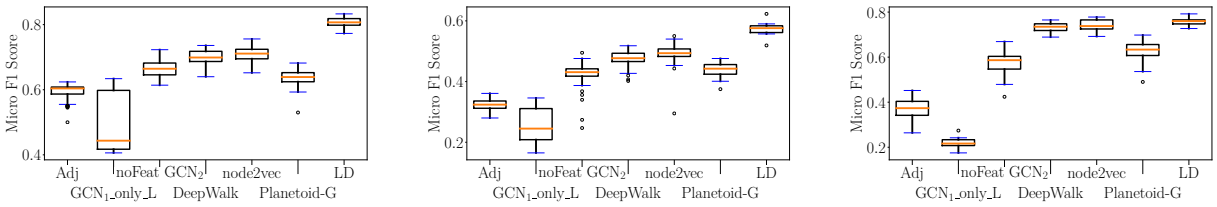
5.2 Multilabel Classification

5.2.1 Experimental Setup. We also perform multilabel node classifications on the following two multilabel networks:

- BlogCatalog [38]. This is a social network graph where each of the 10,312 nodes corresponds to a user and the 333,983 edges represent the friendship relationships between bloggers. 39 different interest groups provide the labels.
- IMDb Germany. This dataset is taken from [13]. It consists of 32,732 nodes, 1,175,364 edges and 27 labels. Each node represents an actor/actress who played in a German movie. Edges connect actors/actresses that were in a cast together and the node labels represent the genres that the corresponding actor/actress played.

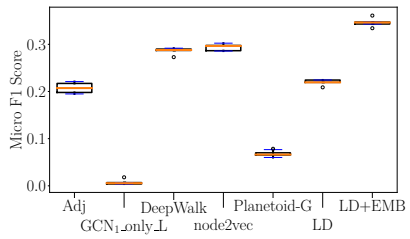
[1] See 3.1 for the explanation why only one convolution layer makes sense

[2] Unless stated differently we use for all competitors the parameter settings as suggested by the corresponding authors. Except for minor adaptations, e.g., to include label information in the one layer GCN models or to make the Planetoid models applicable for multilabel prediciton tasks, we use the original implementations as published by the correpsonding authors.

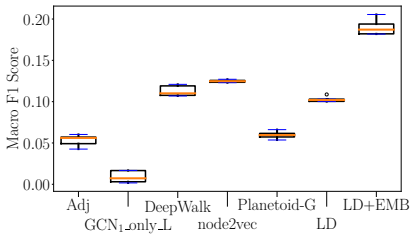


(a) Micro F_1 scores for *Cora*. (b) Micro F_1 scores for *CiteSeer*. (c) Micro F_1 scores for *Pubmed*.

Figure 2: Micro F_1 scores for the three benchmark data sets.

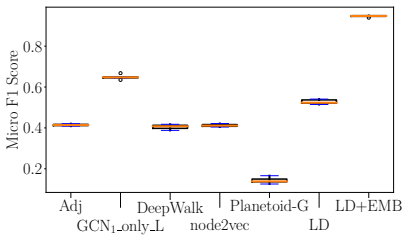


(a) Micro F_1 scores for *BlogCatalog*.

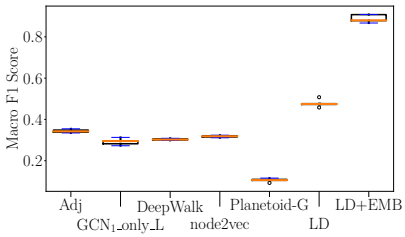


(b) Macro F_1 scores for *BlogCatalog*.

Figure 3: Micro F_1 and macro F_1 for *BlogCatalog*.



(a) Micro F_1 scores for *IMDb Germany*.



(b) Macro F_1 scores for *IMDb Germany*.

Figure 4: Micro F_1 and macro F_1 for *IMDb Germany*.

Since the fraction of positive instances is relatively small for most of the classes, we use weighted cross-entropy as loss function. Therefore, the loss caused by erroneously classified positive instances is weighted higher. We use weight 10 in all our experiments. For the same reason we report micro F_1 and macro F_1 score metrics to measure the quality of the considered methods. We compare our model to the featureless models that we already used for the multiclass experiments [3].

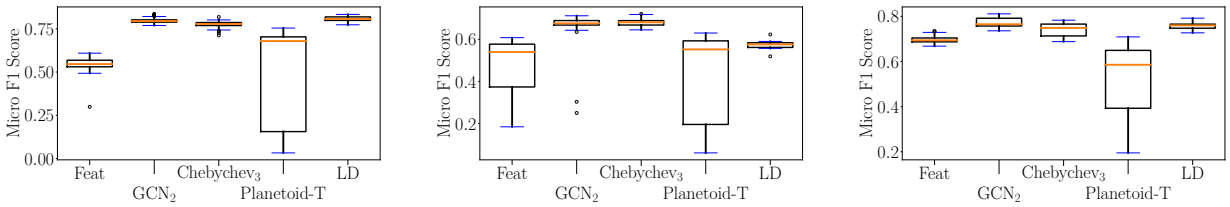
We split the data into training, validation and test set so that 70% of all nodes were used for training, 10% for validation and 20% of the data were used to test the model. Note that we could not use stratified sampling splits for these experiments since we optimize for all classes simultaneously instead of using one-vs-rest classifiers [4]. The hyperparameter setting is as described above. For this set of experiments we ran each model, except for *Planetoid-G*, 10 times on five different data splits. Due to the long runtime of *Planetoid-G* we trained this model only three times on two data splits.

[3]To adapt the *Planetoid-G* implementation for multilabel classification, we use a *sigmoid* activation function at the output layer and also slightly changed the embedding learning step. Entities that are used as context and have the same labels as the node itself are sampled from all classes to which the node belongs to.

[4]That is why our results for *node2vec* and *DeepWalk* on the BlogCatalog network are slightly worse than reported in [16]

5.2.2 Results. The results for the *BlogCatalog* graph are shown in Figure 3. For this network, only using the label information from the direct neighborhood of a node is not useful to infer its labels, c.f., *GCN*$_1$*_only_L*. However, incorporating the label distribution of somewhat larger neighborhoods as for our model (again, we also use the APPR matrix calculated for small values of α to determine the label distribution in neighborhoods that span more than 1-hop neighbors) seems to improve the results for the prediciton task significantly. In fact, our model achieves similar, but slightly worse performance than *node2vec* and *DeepWalk*. Given these results, we also combined the node embeddings based on local label distributions with embeddings that capture structural properties. To capture the structural properties we select a very simple approach: we multiply an embedding matrix with the preprocessed adjacency matrix as in Kipf et al. [20]. The embedding matrix is randomly initialized. Note that the structural similarity is defined via direct neighbors. The resulting representation is concatenated with the hidden layer of the *LD* model and the rest of the *LD* model remains the same. The embedding weights are learned jointly with the rest of the model. The hidden layer H for the resulting model, denoted as *LD+EMB*, can be formalized as

$$H = ReLU\left(\left[\mathbf{E}\hat{\mathbf{A}}, \widehat{\mathbf{APPR}} \cdot \mathbf{Y}_{train} \cdot \mathbf{W}_1\right]\right),$$



(a) Micro F_1 scores for *Cora*. **(b) Micro F_1 scores for *CiteSeer*.** **(c) Micro F_1 scores for *Pubmed*.**

Figure 5: Comparison against attribute-based methods: micro F_1 scores for the three benchmark data sets.

with $[\cdot,\cdot]$ denoting the concatenation operation, $\mathbf{E}$ being the embedding matrix and $\hat{\mathbf{A}}$ being the preprocessed adjacency matrix as in [20]. Again, the bias is omitted for better readability. Having a look at the scores for the *LD+EMB* model, this combination further improves the outcome of the prediction.

For the *IMDb Germany* network, for which the results can be seen in Figure 4, the labels of even very local neighborhoods are already very expressive. Recalling how this network is constructed, we can expect the latter fact and also the superior performance of our model over the two random walk based methods. Particularly noteworthy for this network is the gain of accuracy that the combination of information from both sources, label distribution and structural properties, achieves.

5.3 Comparison to Attribute-Based Methods

To show the power of incorporating label information into the generation process for node embeddings, we also compare our model against the following state-of-the-art attribute-based methods:

- *Feat*: a baseline approach which predicts node labels only based on the node attributes without considering the underlying graph structure (borrowed from [42])
- *GCN*$_2$: the standard 2-layer GCN as published in [20]
- *Chebychev*$_3$: the spectral convolution method which uses chebychev filters as presented in [12]; as in [20] we also use 3rd order chebychev filters
- *Planetoid-T*: the semi-supervised Planetoid framework which uses attribute information as proposed in [42]

For this set of experiments, we again perform multiclass prediciton on the three benchmark text classification datasets and report the prediction accuracy in terms of micro F_1 scores to measure the quality of the retrieved node representations. Note that in contrast to the competitors, our model still does not make use of the node attribute information. The results are depicted in Figure 5 and clearly show that our model can definitely compete with the attribute-based methods and hence is a powerful alternative in cases when no node attributes are present.

5.4 Impact of the α Parameter

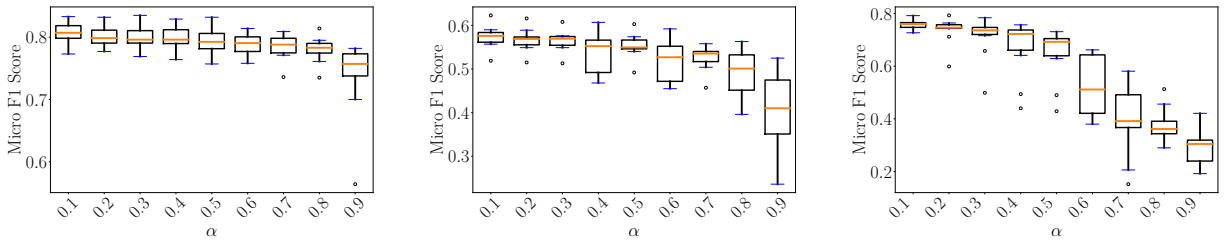
Figure 6 depicts the micro F_1 scores achieved for different values of the teleportation parameter α on the three benchmark datasets. As can be seen, particularly for the *Pubmed* network, the model is quite sensitive to the choice of this parameter. Recall that the teleportation parameter determines how far the neighborhood of each node shall be taken into consideration to get the label distributions for each node. Therefore it might make sense to set the α parameter to a small value so that more labels are collected which in turn leads to a more accurate estimation of the local label distribution. On the other hand, this may not hold in every scenario, for instance if the distribution of classes is heterogeneous, i.e., some classes may only appear in areas of the graph where classes are concentrated locally, while other classes may appear in areas where many classes are mixed even within local neighborhoods. An interesting direction for future work is therefore to optimize for some "good" α value in a data-driven manner. This may be done either by pre-defining a set of different values of α and approaching for the best of these, or by trying to optimize for some "good" α value during the learning procedure. Also, the underlying task, e.g., node classification, may benefit from finding "good" values of α for each node individually rather than relying on a global solution.

6 CONCLUSION

In this paper, we have introduced a novel label-based approach for semi-supervised node classification. In particular, our method aggregates labels from local neighborhoods using APPR. Most existing approaches consider nodes to be similar, if they are closely related in the graph. Methods for attributed graphs additionally take attributes of the neighboring nodes into account. In contrast, our method can relate nodes even if they are not close-by in the graph and makes more effective use of the labels provided for training to improve the classification quality for graphs with and without node attributes. It is further applicable to nodes unseen during training. The results of our experiments on various real-work datasets demonstrate that local label distributions are able to significantly improve classification results in the multiclass and multilabel setting. Our model is even competitive with state-of-the-art models, which take node attributes into consideration. In a first experiment on multilabel datasets, we were already able to significantly boost the performance by using a simple combination of our model with node embeddings.

For future work, we plan to address the problem of selecting a suitable teleportation parameter α. The α parameter controls the extend of the considered local neighborhood and often has a significant impact on the prediction quality. Performing a grid search to determine a good parameter value is a time consuming task. Furthermore, for different classes varying teleportation parameters might yield the best results.

We also aim at further improving the prediction accuracy by further investigating how to effectively combine label-based features with different other kinds of features, such as node attributes, edge attributes or node embeddings in a semi-supervised model. Our



(a) Micro F_1 scores for *Cora*. (b) Micro F_1 scores for *CiteSeer*. (c) Micro F_1 scores for *Pubmed*.

Figure 6: Micro F_1 scores for the three benchmark data sets when considering different locality levels for node neighborhoods.

approach could also be extended to solve additional graph learning tasks, such as link prediction or identification of nodes with unexpected labels for detecting labeling errors or outlier nodes.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2017. Watch your step: Learning graph embeddings through attention. *arXiv preprint arXiv:1710.09599* (2017).

[2] Saba A Al-Sayouri, Pravallika Devineni, Sarah S Lam, Evangelos E Papalexakis, and Danai Koutra. 2016. GECS: Graph Embedding Using Connection Subgraphs. (2016).

[3] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *Proc. of IEEE FOCS*. IEEE, 475–486.

[4] James Atwood and Don Towsley. 2015. Search-Convolutional Neural Networks. *CoRR* abs/1511.02136 (2015). arXiv:1511.02136 http://arxiv.org/abs/1511.02136

[5] 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, Nov (2006), 2399–2434.

[6] Pavel Berkhin. 2006. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Mathematics* 3, 1 (2006), 41–62.

[7] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *CoRR* abs/1312.6203 (2013). http://arxiv.org/abs/1312.6203

[9] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proc. of CIKM*. ACM, 891–900.

[10] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. 2017. Neural Embeddings of Graphs in Hyperbolic Space. *arXiv preprint arXiv:1705.10359* (2017).

[11] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2017. HARP: Hierarchical Representation Learning for Networks. *arXiv preprint arXiv:1706.07845* (2017).

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). 3844–3852.

[13] Evgeniy Faerman, Felix Borutta, Kimon Fountoulakis, and Michael W Mahoney. 2017. LASAGNE: Locality And Structure Aware Graph Node Embedding. *arXiv preprint arXiv:1710.06520* (2017).

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212* (2017).

[15] David F Gleich and Michael W Mahoney. 2015. Using local spectral methods to robustify graph-based learning algorithms. In *Proc. of the ACM SIGKDD*. 359–368.

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proc. of ACM SIGKDD*. 855–864.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1025–1035.

[18] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proc. of the 12th WWW*. ACM, 271–279.

[19] Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proc. of ICML*. 290–297.

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[22] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.

[23] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. 2017. CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters. *CoRR* abs/1705.07664 (2017). arXiv:1705.07664

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.

[25] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[26] Yann LeCun Mikael Henaff, Joan Bruna. 2015. Deep Convolutional Networks on Graph-Structured Data. *arXiv preprint arXiv:1506.05163* (2015).

[27] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 2016. Geometric deep learning on graphs and manifolds using mixture model CNNs. *CoRR* abs/1611.08402 (2016). arXiv:1611.08402 http://arxiv.org/abs/1611.08402

[28] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*.

[29] Sharad Nandanwar and M Narasimha Murty. 2016. Structural neighborhood based classification of nodes in a network. In *Proc. of ACM SIGKDD*. 1085–1094.

[30] Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*. 13–20.

[31] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: bringing order to the web. (1999).

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proc. of ACM SIGKDD*. 701–710.

[33] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proc. of ACM SIGKDD*. ACM, 385–394.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

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

[36] Julian Shun, Farbod Roosta-Khorasani, Kimon Fountoulakis, and Michael W. Mahoney. 2016. Parallel Local Graph Clustering. *Proc. VLDB Endow.* 9, 12 (Aug. 2016), 1041–1052.

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

[38] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proc. of ACM SIGKDD*. ACM, 817–826.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *arXiv preprint arXiv:1710.10903* (2017).

[40] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proc. of ACM SIGKDD*. ACM, 1225–1234.

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

[42] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proc. of ICDM*. 40–48.

[43] Wei Ye, Linfei Zhou, Dominik Mautz, Claudia Plant, and Christian Böhm. 2017. Learning from Labeled and Unlabeled Vertices in Networks. In *Proc. of ACM SIGKDD*. ACM, 1265–1274.

[44] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.

[45] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML*. 912–919.