Multilabel Classification on Heterogeneous Graphs with Gaussian Embeddings

Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari

Sorbonne Universities, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France
{ludovic.dossantos,benjamin.piwowarski,patrick.gallinari}@lip6.fr

Abstract. We consider the problem of node classification in heterogeneous graphs, where both nodes and relations may be of different types, and different sets of categories are associated to each node type. While graph node classification has mainly been tackled for homogeneous graphs, heterogeneous classification is a recent problem which has been motivated by applications in fields such as social networks, where graphs are intrinsically heterogeneous. We propose a transductive approach to this problem based on learning graph embeddings and model the uncertainty associated to the node representations using Gaussian embeddings. A comparison with representative baselines is provided on three heterogeneous datasets.

Keywords: node graph classification, representation learning, gaussian embeddings

1 Introduction

Classification of nodes in graphs is a relational classification problem where the labels of each node depend on its neighbors. Many problems in domains like image, biology, text or social data labeling can be formulated as graph node classification and this problem has been tackled with different approaches like collective classification [21], random walks [1], and transductive regularized models [10]. Most approaches consider homogeneous graphs, where all the nodes share the same set of labels and operate an iterative label propagation from seed nodes to their neighbors. Many problems in domains like biology or social data analysis involve heterogeneous networks where the nodes and the relations between nodes are of different types, each node type being associated to a specific set of labels. For example, the LastFM social network, one of the datasets used in our experiments, links users, tracks, artists and albums via seven different types of relations such as *friendship*, *most listened tracks*, and *authorship*. In such a network, nodes of different types influence each other and their labels are interdependent. The dependency is however more complex than with homogeneous networks and depends both on the nodes type and on their specific relation. Classical methods for homogeneous graphs based for example on label propagation, usually relies on a simple relational hypothesis like homophily in social networks. They cannot be easily extended to heterogeneous networks, and new methods have to be developed for dealing with this relational classification problem.

In this paper, we consider the problem of node classification in heterogeneous graphs. We propose a transductive approach based on graph embeddings where the node embeddings are learned so as to reflect both the classification objective for the different types of nodes and the relational structure of the graph. When most embedding techniques consider deterministic embeddings where each node is represented as a point in a representation space, we focus here on density based embeddings which allow us to capture some form of uncertainty about the learned representations. Uncertainty can have various causes related to the lack of information (isolated nodes in the graph) or because of the contradiction between neighboring nodes (different labels). Our hypothesis is that, because of these different factors, training will result in learned representations with different confidence, and that this uncertainty is important for this classification problem. For that, we will use Gaussian embeddings which have been recently proposed for learning word [23] and knowledge graph [7] embeddings in an unsupervised setting. More precisely, we make each graph node representation correspond to a Gaussian distribution where the mean and the variance are learned. The variance term is a measure of uncertainty associated to the node representation. The objective function is composed of two terms, one reflecting the classification task and the other one reflecting the relations between the nodes. Both mono and multi-label classification can be handled by the model. For the experiments, we focus on classification in social network data. This type of data offers a variety of situations which allows us to illustrate the behavior and the performance of the model for different types of heterogeneous classification problems.

To summarize, our contributions are as follows: (i) We propose a new method for learning to classify nodes adapted to heterogeneous graph data; (ii) We model the uncertainty associated to the nodes representation; (iii) We provide a comparison with state of the art baselines on a series of social data classification problems representative of different situations.

2 Related Work

2.1 Graph Node Classification

Several different models have been proposed to solve the graph node classification task. We discuss below three main families [4] (i) Random Walk type methods, (ii) collective classification, and (iii) semi-supervised/transductive graph regularized model.

Random Walk Type Methods This family gathers methods where labels are iteratively propagated from seed nodes to all the other nodes in a network. Propagation follows a random walk or a similar iterative mechanism. [8, 28] are among the early Machine Learning (ML) models using random walks for classification in homogeneous graphs. [27] propose an extension of these models for heterogeneous graphs. It relies on hand defined projections of the graph onto homogeneous graphs, the approach being difficult to adapt automatically to new datasets. The Graffiti random surfer model [1] is a state of the art random walk classifier for heterogeneous graphs. It is based on two intertwined random walks. Both are between nodes of the same type, but allowing either one hop (standard) or two hops (extended) steps in the graph. It models up to a certain extent the influence among nodes of different types. In our preliminary tests on different datasets, this model was among the best ones.

Collective Classification Collective classification algorithms are extensions of classical inductive classification to relational data. They take as input a fixed size vector composed of node features and of statistics on the node neighbors current labels. Sen et al. [21] provide an introduction and a comparison of some of these models. They distinguish between two families: local and global models. The former make use of local classifiers. In [15, 16] for example, naive Bayes classifiers are used iteratively, dynamically updating the attributes of nodes as inferences are made about their neighbors. Along these lines, [19] recently introduced an iterative model for sparsely labeled network which forces the label predictions to map the distribution of the observed data with a maximum entropy constraint. Global classifiers optimize a global loss function using graphical models, like e.g. Markov Random Fields. Iterative methods suppose features associated to nodes to learn the classifier, which is not the case in our work.

Semi-Supervised Transductive Learning The third family has been developed for exploiting the manifold assumption in semi-supervised learning. The loss function is composed of two main terms, one is for classification on the labeled nodes, the other one is a propagation equation which encourages neighbor nodes to share similar labels. Seminal works in this direction include [26, 2, 24, 20]. All these models have been developed for homogeneous graphs and perform some form of label propagation as random walk formulations. The difference with the latter is that the problem is formulated as a loss minimization one, which is more general than simply formulating a propagation rule. Relations between random walks and loss based models is discussed more at length in [29, 4]. Extensions have been proposed over the years to handle more general situations. Multi-relational graphs where nodes are all of the same type, but can be linked by different relations are considered in [9, 13]. This also allows them to extend the transductive models to inductive formulations. Some authors have attempted to extend homogeneous formulations to the heterogeneous setting. All follow more or less the idea of projecting the heterogeneous graph onto a series of homogeneous ones, thus creating a series of homogeneous classification problems. Work in this direction includes [11] which is a direct extension of the homogeneous formulation in [25]. Graph projections have to be defined for each new problem and none of these models is able to directly exploit the correlation between nodes of different types. The work closer to ours is [10] who was among the first to propose an embedding model for transductive heterogeneous graph classification. This has been the starting point of our work, but they only consider deterministic representations where we use a more general transductive formulation with probabilistic embeddings.

To summarize among the approaches that consider heterogeneous graph classification, very few allow modeling the influences between nodes of different types. In the experimental section, we will compare our model to [1] and [10] which have been designed specifically for heterogeneous classification, as well as an unsupervised graph embedding model [22] and an homogeneous graph model [28].

2.2 Learning Representations for Graphs and Relational Data

In the last years, there has been a growing interest in learning latent representations. This has led to breakthroughs in domains like image recognition, speech or natural language processing [3, 14]. Graph embeddings have been proposed for unsupervised learning where the goal is to learn a representation that preserves the graph structure and that can be exploited latter for different purposes like visualization, clustering or classification. For graphs, [18] learns graph representations by performing truncated walks on the graph – and supposing that nodes along the path should be close together in the representation space. [22] propose an algorithm designed for very large graphs, that can be used for different types of graphs (undirected, directed, weighted or not) – we use their method as our unsupervised baseline that embeds all data points and then train a classifier on labeled ones. Somewhat related to this topic is the learning of triplets in knowledge graphs where both relations and nodes representations are learned for ranking positive triplets over negative ones [5–7]. The setting is however quite different from the one considered here. Finally, modeling uncertainty via Gaussian embeddings has been proposed recently for unsupervised learning in [23, 7]. Based on sentences in the former and knowledge graph in the latter, they propose energy-based models to learn Gaussian embeddings. In this paper, we borrow their formalization and graph regularization cost in a transductive setting.

3 Model

In this section we present our model, namely *Heterogeneous Classification with Gaussian Embeddings (HCGE).*

3.1 Notations

We first introduce the notations used throughout this paper. An heterogeneous network is modeled as a directed weighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{W})$ where $\mathcal{N}$ is the set of nodes, $\mathcal{E}$ the set of edges and $\mathcal{W}$ the weights associated to the edges. Each node $x_i \in \mathcal{N}$ of the graph has a type $t_i \in \mathcal{T}$, where $\mathcal{T} = 1, 2, \dots, T$. We denote N_i the neighbors of x_i.

Regarding the classification task, let $\mathcal{Y}^t$ denotes the set of categories associated to nodes of type t, and $\#\mathcal{Y}^t$ the cardinality of $\mathcal{Y}^t$. $\mathcal{L} \subset \mathcal{N}$ is the set of indices of labeled nodes. For $i \in \mathcal{L}$, y_i is the class vector associated to x_i: node x_i belongs to category c if $y_i^c = 1$ and does not belong if $y_i^c = -1$.

In our model, each node x_i is mapped onto a representation which is a Gaussian distribution over the space $z_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ in $\mathbb{R}^Z$. The latent space is common to all nodes. In this paper, we compare two different parameterizations of Σ. We experimented with a spherical ($\Sigma_i = \sigma_i Id$) and a diagonal ($\Sigma_i = diag\,(\sigma_i^p)_p$) covariance matrix. We use a weight w_r for each type of relation and to simplify we will denote w_{ij} the weight for edge (i, j) linking node i to node j with a given relation.

3.2 Learning Gaussian Embeddings

Loss Function We learn the representations of nodes and classifiers parameters by minimizing an objective loss function. It takes the general form of transductive regularized loss [12, 25], with a classification (Δ_C) and a regularization term (Δ_G), both being detailed later:

$$L(z, \theta) = \sum_{i \in \mathcal{L}} \Delta_C(f_{\theta^{t_i}}(z_i), y_i) + \lambda \sum_{i \in \mathcal{N}} \sum_{j \in N_i} w_{ij} \Delta_G(z_i, z_j) \tag{1}$$

As for classical transductive graph losses, the minimization in (1) aims at finding a trade-off between the difference between observed and predicted labels in $\mathcal{Y}^t$, and the amount of information shared between two connected nodes. There are however major differences, since here z is not a label as in classical formulations, but a node embedding. Finally, the function $f_{\theta^t}(.)$ is a parametric classifier for a node of type t – there is one such classifier for each node type. Since we are using Gaussian embeddings, the zs are random variables and the regularization term is a dissimilarity measure between distributions.

To avoid overfitting, following [23], we regularize the mean and the covariance matrix associated to each node. We add two constraints to prevent means and covariances to be too large and to keep the covariance matrices positive definite (this also prevents degenerate solutions):

$$||\mu_i|| \leq C \text{ and } \forall p,\, m \leq \sigma_i^p \leq M \tag{2}$$

where the different parameters C, m and M have been set manually after some trials on a subset of the DBLP training set to respectively 10, 0.01 and 10 (and not changed after that), but any other reasonable value will do.

The two following paragraphs refer to the respective parts of (1).

Classifier The mapping onto the latent space is learned so that the labels of each type of node can be predicted from the (Gaussian) embedding. For that, we use a parametric classification function f_{θ^t} depending on the type t of the node. This function takes as input a node representation and outputs a vector of scores for each label corresponding to the node type. The parameters θ^t of the classifier are learned by minimizing the following loss on labeled data:

$$L_{Classification} = \sum_{i \in \mathcal{L}} \Delta_C(f_{\theta^{t_i}}(z_i), y_i) \tag{3}$$

where $\Delta_C(f_{\theta^{t_i}}(z_i), y_i)$ is the loss associated to predicting labels $f_{\theta^{t_i}}(z_i)$ given the observed labels y_i. We recall that in this equation $f_{\theta^{t_i}}(z_i)$ and y_i have values in $\mathbb{R}^{\#\mathcal{Y}^t}$.

In our experiments, we used different losses for Δ_C. We first considered the case where a class decision is simply the expectation of the classifier score together with a hinge loss, adapting the loss proposed in [10]. For a given node x of type t with an embedding z, this gives:

$$\Delta_C(f_{\theta^t}(z), y) = \Delta_{EV}(f_{\theta^t}(z), y) \stackrel{\text{def}}{=} \sum_{k=1}^{\#\mathcal{Y}^t} \max\left(0; 1 - y^k \mathbb{E}_z[f_{\theta^t}^k(z)]\right) \tag{4}$$

where y^k is 1 if x belongs to category k and -1 otherwise, and $f_{\theta^t}^k(z)$ is a random variable for category k.

Alternatively, the density based formulation allows us to consider uncertainty through a probabilistic criterion. We used here for Δ_C the log-probability that $y^k f_{\theta^t}(z)$ will take a positive value. In this case, the variance will be influenced by the two loss terms: if the two terms act in opposite directions, one solution will be to increase variance. As we will see this is confirmed by the experiments.

$$\Delta_C(f_{\theta^t}(z), y) = \Delta_{Pr}(f_{\theta^t}(z), y) \stackrel{\text{def}}{=} -\sum_{k=1}^{\#\mathcal{Y}^t} \log \mathbb{P}\left(y^k f_{\theta^t}^k(z) > 0\right) \tag{5}$$

In our experiments and for both costs, we used a linear classifier for $f_{\theta^t}^k$, which allows to easily compute the different costs and gradients, since the random variable $f_{\theta^t}^k(z)$, being a linear combination of Gaussian variables, is Gaussian too. A basic derivation shows that:

$$\mathbb{P}\left(y^k f_{\theta^t}^k(z) > 0\right) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\mu \cdot \theta^t}{\sqrt{2\sum_p (\theta_p^t \sigma^p)^2}}\right)\right) \tag{6}$$

where erf is the Gauss error function.

There are some notable differences between the two classification losses during learning. In the case of a linear classifier f_{θ^t}, $\mathbb{E}_z[f_{\theta^t}^k(z)] = \mu \cdot \theta_k^t$. Thus, minimizing Δ_{EV} only updates the mean of the gaussian embedding: the covariance matrix of the embedding does not interfere with the classification term, and is only present in the second term of (1).

For the Δ_{Pr} loss, the probability is proportional to erf $\left(\frac{\mu \cdot \theta^t}{\sqrt{2 \sum_p (\theta^t_p \sigma^p)^2}}\right)$ where the variance is present. When the graph regularization and classification cost pull the representation mean in opposite directions (opposite gradients), the model will respond by increasing the variance for the spherical variance model[1]: this behavior is interesting since it transforms an opposition between regularization and classification costs into increased uncertainty.

Graph Embedding We make the hypothesis that two nodes connected in the graph should have similar representations, whatever their type is. Intuitively, this will force nodes of the same type which are close in the graph to be close in the representation space. The strength of this attraction between nodes of the same class will be proportional to their closeness in the graph and to the weight of the path(s) linking them. We use the asymmetric loss proposed in [23, 7]:

$$L_{Graph} = \sum_i \sum_{j \in N_i} w_{ij} D_{KL}(z_j || z_i) \tag{7}$$

where $\Delta_G(z_i, z_j) = D_{KL}(z_j || z_i)$ is the Kullback-Leibler divergence between the distributions of z_i from z_j:

$$\begin{aligned} D_{KL}(z_j || z_i) &= \int_{x \in \mathbb{R}} \mathcal{N}(x; \mu_j, \Sigma_j) \log \frac{\mathcal{N}(x; \mu_j, \Sigma_j)}{\mathcal{N}(x; \mu_i, \Sigma_i)} dx \\ &= \frac{1}{2} \left(\mathrm{tr}(\Sigma_i^{-1} \Sigma_j) + (\mu_i - \mu_j)^T \Sigma_i^{-1} (\mu_i - \mu_j) - d - \log \frac{\det(\Sigma_j)}{\det(\Sigma_i)} \right) \end{aligned} \tag{8}$$

The loss L_{Graph} is a sum over the neighbors N_i of i, where w_{ij} is the weight of the edge between x_i and x_j. Other similarity measures between distributions could be used as well, the Kullback-Leibler divergence having the advantage of being asymmetric, which fits well the social network datasets used in the experiments.

Algorithm Learning the gaussian embeddings $z \sim \mathcal{N}(\mu, \Sigma)$ and the classifiers parameters θ consists in minimizing loss function in (1). We used here a Stochastic Gradient Descent Method to learn the latent representations, i.e. the μ_i, Σ_i as well as the parameters θ of the classifiers.

Our algorithm samples a pair of connected nodes and then makes a gradient update of the nodes parameters. For each sampled node z_i that is part of the labeled training set $\mathcal{L}$, the algorithm performs an update according to the first term of (3). This update consists in successively modifying the parameters of the classification functions θ^{t_i} and of the latent representations μ_i and Σ_i so as to minimize the classification loss term. Then, the model updates its parameters

[1] the increase will be in the direction of the normal to the classifier hyperplane for the diagonal variance model

with respect to the smoothness term of (7). Note that, while we use a stochastic gradient descent, other methods like mini-batch gradients or batch algorithms could be used as well.

4 Experiments

4.1 Datasets

		Type	**Nb. Nodes**	**Nb. Labeled Nodes**	**Nb. Labels**
DBLP	Nodes	Paper	14,376	14,376	20
		Author	14,475	4,057	4
	Edges	**Type**	**Nb. Edges**		
		Author↔Paper	41,794		
Flickr	Nodes	Photos	46,926	8,766	21
		User	4,760	3,476	42
	Edges	User↔User	175,779		
		User↔Photo	46,926		
LastFM	Nodes	Users	1,013	321	59
		Tracks	35,181	24,562	28
		Albums	32,118	15,966	47
		Artists	17,138	11,564	47
	Edges	User↔User	1,109		
		User↔Album	47,541		
		User↔Artist	47,812		
		User↔Track	47,807		
		Track↔Album	29,647		
		Track↔Artist	35,181		
		Album↔Artist	32,118		

Table 1: Datasets

Experiments have been performed on three datasets respectively extracted from DBLP, Flickr and LastFM. For all but the first dataset (DBLP), each node can have multiple labels. The three datasets are described below and summarized in Table 1.

The **DBLP** dataset is a bibliographic network composed of authors and papers. Authors are labeled with their research domain (4 different domains) while papers are labeled with the conference name they were published in (20 labels). Authors and papers are connected through an *authorship* relation. The graph is thus composed of two types of nodes and is bipartite with only one relation type. Classification is monolabel on papers and authors.

The **Flickr** corpus is a dataset composed of photos and users. The photo labels correspond to different possible tags while the user labels correspond to their subscribed groups. The classification problem is multi-label: images and users may belong to more than one category. Photos are related to users through an *authorship* relation, while users are related to others through a *following* relation. We have kept the image tags that appear in at least 500 images, and user categories that appear also at least 500 times in the dataset resulting in 21 possible labels for photos and 42 for authors.

The **LastFM** dataset is a social network composed of users, tracks, albums and artists. This dataset was extracted using the LastFM API[2]. The task is multi-label and all node types have their specific set of labels. Users are labeled with the type of music they like (59 labels), tracks with the kind of music they belong to (28 labels), album with their type (47 labels) and artists with the kind of music they play the most (47 labels). Users are related to users (*friendship*), tracks (*favorite tracks*), albums (*favorite albums*) and artists (*favorite artists*). Tracks are related to albums (*belong to*) and artists (*singer*). Finally, albums are related to artists (*sing in*). Note that one track can be related to several artists, and an album can be related to several artists. This dataset contains tracks labeled by their genre (*rock*, *indie*, ...), users by the type of music they like (*female vocalists*, *ambient*, ...), albums by their type (*various artists*, *live*, ...) and artists by the kind of music they make (*folk*, *singer songwriter*, ...). Some labels may be the same string-wise for different types of nodes, but we consider that labels of different type of nodes are distinct, e.g. *pop* is not the same for an artist or a track.

We compare our approach with four state-of-the-art models (see Sect. 2):

- **LINE** [22], which is representative of unsupervised learning of graph embeddings suitable for various tasks such as classification. We performed a logistic regression with the learned representations as inputs.
- **HLP** [28], which is representative of transductive graph algorithms developed for semi-supervised learning. As HLP is designed for homogeneous graphs, we perform as many random walks as the number of node types, considering each time that all the nodes are of a same given type.
- **Graffiti** [1], which is a state of the art model for the task of classification with random walk in heterogeneous graph.
- **LSHM** [10], which is another state of the art model for the task of classification with deterministic vector representations in heterogeneous graph.

Evaluation Measures and Protocol For the evaluation, we have considered two different evaluation measures. The **Precision at 1 (P@1)** measures the percentage of nodes for which the category with the highest score is among the observed labels. The **Precision at k (P@k)** is the proportion of correct labels in the set of k labels with the highest predicted scores. Here micro P@k is an average on all the node types, with k set to the number of relevant categories. This is a measure of the capacity of a model to correctly pick the k relevant categories of any node. In the case of DBLP (mono-label dataset), we consider

[2] To access the API go to http://www.lastfm.fr/api

that the predicted category is the category with the highest score. We make use of the **Precision at 1 (P@1)** measure as there is at most one label per node. We optimize and compare the different models with regard to micro-average, we also report macro-average.

Regarding the experimental protocol, we partition a dataset into two different subsets, namely a training set and a testing set. As all the models have hyperparameters, one subset of the training set is used as a validation set to optimize by grid search the hyperparameters. The optimization is done with respect to the **Micro P@k** measure, which corresponds to the mean of P@k over all nodes. The other part of the training set is used to learn the parameters of the different models. We then compare the different models based on the results on the testing set, by using the model for which the performance over the validation set was the best.

Experiments are performed with different training set sizes: 10%, 30%, 50%. Within our transductive setting, the training set size refers to **the proportion of labeled nodes** used in the training set[3]. The training nodes are selected at random. The proportion of nodes used during the parameters training phase and used for the hyperparameters selection depends of the size of the training set. We use 50-50 for a training set size of 10% and 80-20 (train/validation) for the others. Experiments are performed with 5 random splits. The hyper-parameters are selected for each split using the validation set. We then average 5 runs over each split.

4.2 Results

In this section we present the results of four variants of our gaussian embedding model, and compare to LINE [22], Graffiti [1], HLP [28] and LSHM [10]. The experiments are performed on the three datasets described in Table 1 and the results are described in Tables 2 (DBLP), 3 (FlickR) and 4 (LastFM). The best performing classifier (on the test set) is presented in bold.

Concerning the four variants of our model, HCGE($\varDelta_\bullet$,X) refers to the HCGE model with the classification loss $\varDelta_\bullet$ ($\varDelta_{EV}$ or $\varDelta_{Pr}$) and a spherical (X=S) or diagonal (X=D) covariance matrix.

For micro P@k, our model generally outperforms the others on all the datasets. Supervised models (HLP, Graffiti, LSHM and HCGE) using the class information outperform unsupervised representation learning, which matches the results reported in [10]. On all datasets, the performances of HLP are below the performances of Graffiti, LSHM and HCGE. This clearly shows that modeling the heterogeneity of the graph brings noteworthy improvements. Comparing the heterogeneous models, both LSHM and HCGE outperform Graffiti on all datasets. On average, comparing to Graffiti, LSHM is 2.4 better on DBLP, 2.1 better on FlickR and 2.5 better on LastFM. We observed the same behavior for HCGE, with +2.8 on DBLP, +4.4 on FlickR and +6.0 on LastFM. We can note that the

[3] We did not prune the graph

Train size	Model	Train	**Val**	Test			
		Micro		**Micro**	Macro	Author	Paper
10%	LINE	25.1	18.9	19.5	23.0	29.1	16.8
	HLP	100	24.7	24.1	27.2	32.6	21.8
	Graffiti	100	32.4	30.9	38.1	50.8	25.3
	LSHM	99.8	33.8	**32.1**	**40.0**	**53.9**	**26.0**
	HCGE($\varDelta_{EV}$,S)	99.7	33.1	30.9	38.5	52.1	24.9
	HCGE($\varDelta_{EV}$,D)	95.6	31.4	30.4	37.4	49.9	24.9
	HCGE($\varDelta_{Pr}$,S)	83.8	29.0	27.9	34.3	45.6	22.9
	HCGE($\varDelta_{Pr}$,D)	92.9	29.0	28.3	34.3	45.1	23.6
30%	LINE	24.0	21.5	21.9	24.8	30.1	19.5
	HLP	100	35.8	36.0	41.9	52.4	31.4
	Graffiti	100	39.6	38.5	46.6	61.1	32.1
	LSHM	99.7	43.0	41.2	52.9	73.8	31.9
	HCGE($\varDelta_{EV}$,S)	98.5	44.4	**42.3**	52.6	71.0	**34.3**
	HCGE($\varDelta_{EV}$,D)	98.8	42.9	41.2	50.8	68.0	33.6
	HCGE($\varDelta_{Pr}$,S)	97.5	41.8	41.3	52.1	71.4	32.8
	HCGE($\varDelta_{Pr}$,D)	97.4	43.8	**42.3**	**54.1**	**75.0**	33.1
50%	LINE	24.2	21.1	22.3	25.0	29.8	20.2
	HLP	100	39.7	39.4	46.5	59.3	33.7
	Graffiti	100	41.5	41.2	49.4	64.1	34.8
	LSHM	99.9	45.5	44.4	56.8	**79.2**	34.5
	HCGE($\varDelta_{EV}$,S)	99.3	45.6	44.6	55.2	74.1	36.3
	HCGE($\varDelta_{EV}$,D)	98.1	44.7	43.9	53.7	71.0	36.3
	HCGE($\varDelta_{Pr}$,S)	99.4	45.8	45.5	57.1	77.8	**36.4**
	HCGE($\varDelta_{Pr}$,D)	97.6	45.9	**45.7**	**57.7**	**79.2**	36.2

Table 2: P@1 DBLP

more complex the dataset, the higher the gap compared to the baselines. This also shows that the use of representations can clearly improve the performances.

On each dataset, our model outperforms LSHM (and the other competitors) 8 times over 9, with on average +1.0 points for DBLP, +2.3 for FlickR, and +3.8 for LastFM over the second ranked model. According to the results, introducing uncertainty in representations clearly improves results when comparing to LSHM. Taking a closer look to the results, we can see that our models superiority increases with the graph complexity. Let us also point out that, according to our initial intuition, the effect of using uncertainty has more impact when the amount of training data is lower: the difference between LSHM and HCGE decreases in general when more training data is available (except for DBLP).

Let us compare the performance of the variants $\varDelta_{EV}$ and $\varDelta_{Pr}$. Globally, $\varDelta_{Pr}$ seems to be disadvantaged by a low number of training examples, when $\varDelta_{EV}$ seems to be more stable in comparison to other baselines. However, the more training data, the closer the $\varDelta_{Pr}$ variant is to $\varDelta_{EV}$. For example, on the DBLP dataset, moving from 10% to 30% improves on average $\varDelta_{Pr}$ results by

Train size	Model	Train	**Val**	Test			
		Micro		**Micro**	Macro	User	Photo
10%	LINE	24.4	19.4	20.7	23.2	29.1	17.3
	HLP	100	26.0	26.3	27.8	31.3	24.3
	Graffiti	100	24.3	24.5	27.0	32.7	21.2
	LSHM	99.3	29.6	29.3	29.1	28.6	29.5
	HCGE($\varDelta_{EV}$,S)	98.9	33.5	**32.7**	**32.6**	32.4	**32.8**
	HCGE($\varDelta_{EV}$,D)	99.1	33.4	32.6	**32.6**	32.7	32.5
	HCGE($\varDelta_{Pr}$,S)	96.0	30.4	29.7	29.2	28.1	30.3
	HCGE($\varDelta_{Pr}$,D)	98.7	31.7	31.9	32.2	**33.0**	31.5
30%	LINE	23.0	21.6	21.5	24.2	30.6	17.9
	HLP	100	47.6	47.7	43.7	34.5	53.0
	Graffiti	100	47.5	47.0	43.7	**36.1**	51.3
	LSHM	100	49.2	48.4	43.6	32.5	54.7
	HCGE($\varDelta_{EV}$,S)	99.1	51.5	50.0	45.6	35.4	55.8
	HCGE($\varDelta_{EV}$,D)	98.7	51.6	**50.1**	45.7	35.3	**56.0**
	HCGE($\varDelta_{Pr}$,S)	98.3	50.1	49.0	44.4	33.8	55.1
	HCGE($\varDelta_{Pr}$,D)	98.5	50.6	50.0	**45.8**	**36.1**	55.5
50%	LINE	23.2	21.8	21.8	24.6	31.0	18.2
	HLP	100	54.2	54.1	48.6	35.8	61.4
	Graffiti	100	54.4	54.0	48.8	36.9	60.8
	LSHM	99.9	55.1	54.0	47.9	33.7	62.0
	HCGE($\varDelta_{EV}$,S)	97.9	56.7	55.8	50.0	36.5	**63.4**
	HCGE($\varDelta_{EV}$,D)	97.3	56.6	55.8	50.0	36.5	**63.4**
	HCGE($\varDelta_{Pr}$,S)	98.8	55.7	54.8	49.0	35.5	62.5
	HCGE($\varDelta_{Pr}$,D)	98.4	56.4	**55.9**	**50.3**	**37.2**	63.3

Table 3: P@k FlickR

+13.7 but only by +11.1 for $\varDelta_{EV}$. For a training set size of 50%, the difference between $\varDelta_{Pr}$ and $\varDelta_{EV}$ is +1.1 on DBLP, and +0.1 on FlickR. For LastFM, the difference is resp. -14.6 for 10%, -6.5 for 30% and -1.5 for 50% of the dataset used for training. On the three datasets, the lower the training set size, the better $\varDelta_{EV}$ seems to be comparing to $\varDelta_{Pr}$. We could not explain this difference in the behavior between $\varDelta_{EV}$ and $\varDelta_{Pr}$, but believe that this is due to the fact that the covariance matrix is only optimized in the graph regularization term in the case of $\varDelta_{EV}$.

Let us now compare the use of a spherical and a diagonal covariance matrix. For the $\varDelta_{EV}$ variant, it looks like moving from a spherical covariance matrix to a diagonal one brings no improvement. It even decreases the performance on DBLP. Concerning the $\varDelta_{Pr}$ variant, for which the covariance matrix plays a role in the classification cost, conclusions are reversed and using diagonal covariance matrices improve the results. On the FlickR dataset, the use of a diagonal variance improves the results by 1.4 on average. However, it looks like

Train size	Model	Train	**Val**	Test					
		Micro		**Micro**	Macro	User	Track	Album	Artist
10%	LINE	20.8	20.6	20.4	15.9	5.6	26.0	14.5	17.4
	HLP	98.7	38.1	38.4	30.0	9.9	47.8	27.2	35.1
	Graffiti	100	40.1	40.0	31.4	**10.6**	49.0	28.1	38.1
	LSHM	99.9	36.4	36.3	27.2	9.0	48.4	26.2	25.3
	HCGE(Δ_{EV},S)	99.8	44.4	**44.0**	**34.1**	9.6	**52.3**	**35.0**	**39.7**
	HCGE(Δ_{EV},D)	99.3	44.0	43.6	34.0	10.5	52.2	34.4	38.7
	HCGE(Δ_{Pr},S)	97.6	27.7	27.8	20.7	4.1	34.9	21.0	23.0
	HCGE(Δ_{Pr},D)	96.0	30.3	29.4	21.9	6.7	38.7	22.1	20.2
30%	LINE	20.5	20.9	20.5	17.0	10.1	25.9	14.4	17.5
	HLP	98.9	50.2	49.7	40.0	**17.2**	60.5	37.7	44.8
	Graffiti	100	50.8	50.3	40.4	**17.2**	61.7	36.2	46.5
	LSHM	99.8	54.2	53.3	40.3	9.7	65.8	42.7	42.9
	HCGE(Δ_{EV},S)	99.6	58.2	**57.3**	45.0	14.8	**68.2**	**45.9**	51.2
	HCGE(Δ_{EV},D)	99.5	57.9	57.0	**45.3**	16.8	67.5	45.7	**51.3**
	HCGE(Δ_{Pr},S)	97.5	50.5	50.4	37.7	9.9	66.4	32.6	42.0
	HCGE(Δ_{Pr},D)	96.9	51.5	50.8	38.5	13.2	65.0	41.4	34.4
50%	LINE	20.5	20.5	20.5	17.0	10.3	26.0	14.4	17.5
	HLP	98.8	51.9	52.1	42.3	19.4	63.1	40.2	46.4
	Graffiti	100	53.2	53.5	43.2	19.1	65.4	39.5	48.7
	LSHM	99.7	56.6	56.7	43.2	11.0	68.8	45.6	47.6
	HCGE(Δ_{EV},S)	99.4	60.3	**60.4**	**48.7**	**20.4**	**71.2**	**48.8**	**54.4**
	HCGE(Δ_{EV},D)	99.9	60.1	60.3	48.6	20.1	71.1	48.7	54.3
	HCGE(Δ_{Pr},S)	99.2	58.6	58.5	45.0	11.8	69.8	47.4	51.0
	HCGE(Δ_{Pr},D)	99.9	58.9	58.9	47.2	18.9	70.2	46.4	53.4

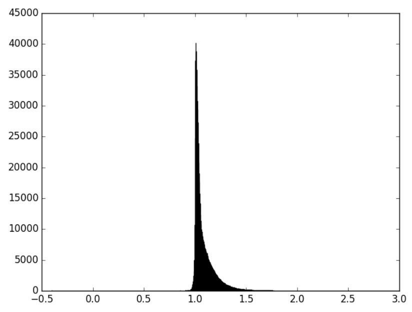
Table 4: P@k LastFM

the more training data, the less the improvement, with +2.2 improvement for a training set size of 10%, +1.0 for 30% and +1.1 for 50%.

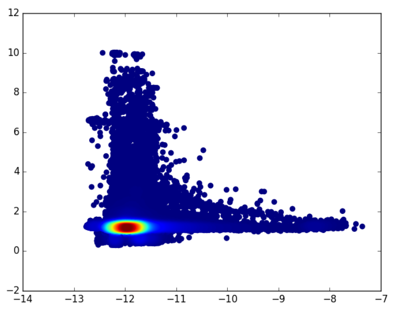
4.3 Qualitative Discussion

In this section, we focus on studying qualitatively the representations found by HCGE. We consider the most robust variant of our model (Δ_{EV},S), and the most challenging dataset, LastFM (similar observations were made on the other datasets). We will examine the respective role of regularization and classification costs on labeled training nodes, and the relationship between the learned variance of a node and the local node properties (like its number of neighbors).

We first examined the respective role of classification and regularization costs. In (1), the max-margin classification cost implies that the gradient of a node x is 0 if $y^k\mathbb{E}_z[f^k_{\theta^t}(z)]$ is above 1. In this case, the only constraints on the node are due to the graph regularization cost. We can see how many of the nodes are useful for classification by looking at the number of cases for which $y^k\mathbb{E}_z[f^k_{\theta^t}(z)]$ is below or equal to 1. In Figure (1a), we have shown an histogram of $y^k\mathbb{E}_z[f^k_{\theta^t}(z)]$ for labeled nodes in the training set (after convergence). For around one-third of the nodes, the value of the classifier is above 1.1 – they could be removed from the labeled set without harming the solution. This is clearly in agreement with the experiments where we have shown that representation-based models were performing better than the others, and suggests that it would be interesting to use these statistics to predict the performance of the model on held-out data.



(a) Histogram of $y^k \mathbb{E}_z \left[f^k_{\theta^t}(z) \right]$.



(b) Plot of σ against the log PageRank.

Fig. 1: Qualitative results for the model HCGE(Δ_{EV},S) on the LastFM dataset with 50% of the dataset used for train. In Fig. (1b), we computed gaussian kernel density to show high density regions in the plot.

Regarding the relationship between the learned variance and the local properties of each node, we looked at the relationship between the PageRank[4] [17] of a node and its variance. Figure (1b) shows that high PageRank implies a small variance. Which means that for central nodes, representations are less uncertain. However, the reverse implication is not true.

5 Conclusion

We have explored the use of uncertainty for learning to represent nodes in the challenging task of heterogeneous graph node classification. The proposed model, Heterogeneous Classification with Gaussian Embeddings (HCGE), learns for each node a Gaussian distribution over the representation space, parameterized by its mean and covariance matrix, by optimizing a loss function that includes a classification loss and graph regularization loss. We have examined four variants of this model, by using either spherical and diagonal covariance matrices, and by using two different loss functions for classification. Our model can easily be

[4] Using a standard damping factor of 0.15

extended to inductive learning by defining the gaussian representation z as a parameterized function of the input features.

Based on the experimental results obtained on datasets representative of different situations, our main findings are that (i) integrating uncertainty in representations improved classification (ii) according to our initial intuition, the effect of using uncertainty has generally more impact when the amount of training data is lower and (iii) according to our expectation, highly central nodes seem to have less variance associated to their representation.

Future work will address more in detail the relationship between the variance and node properties, as well as understanding the interplay between regularization and classification loss when both include the variance in their formulation.

6 Acknowledgement

This work has been partially supported by the following projects : Xu Guangqi 2016 Deep learning for Large Scale Dynamic and Spatio-Temporal Data ; REQUEST Investissement d'Avenir 2014 ; LOCUST ANR 2015.

References

1. Ralitsa Angelova, Gjergji Kasneci, and Gerhard Weikum. Graffiti: graph-based classification in heterogeneous networks. *World Wide Web*, 15(2):139–170, 2012.
2. Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, December 2006.
3. Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.
4. Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. *Semi-supervised learning*, 10, 2006.
5. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.
6. Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344, 2011.
7. Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM CIKM*, pages 623–632. ACM, 2015.
8. Martin Szummer Tommi Jaakkola and Martin Szummer. Partially labeled classification with markov random walks. *Advances in neural information processing systems (NIPS)*, 14:945–952, 2002.
9. Y Jacob, L Denoyer, and P Gallinari. Classification and annotation in social corpora using multiple relations. In *Proceedings of the 20th ACM international CIKM*, pages 1215–1220. ACM Press, 2011.

10. Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 373–382. ACM, 2014.
11. Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECML PKDD*, volume 0053, pages 570–586. Springer, 2010.
12. Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer, 2010.
13. T. Kato, H. Kashima, and M. Sugiyama. Integration of multiple networks for robust label propagation. In *SIAM Conf. on Data Mining*, pages 716–726. Citeseer, 2008.
14. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
15. Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.
16. Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 264–271. ACM, 2000.
17. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
18. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.
19. Joseph J Pfeiffer III, Jennifer Neville, and Paul N Bennett. Overcoming relational learning biases to accurately predict preferences in large scale networks. In *Proceedings of the 24th International Conference on World Wide Web*, pages 853–863. International World Wide Web Conferences Steering Committee, 2015.
20. R. Pimplikar, D. Garg, D. Bharani, and G. Parija. Learning to propagate rare labels. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 201–210. ACM, 2014.
21. Prithviraj Sen, Galileo Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
22. Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.
23. Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.
24. Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
25. Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

26. Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proc. of the 22nd intern. conf. on Machine learning*, ICML '05, pages 1036–1043, New York, NY, USA, 2005. ACM.
27. Yang Zhou and Ling Liu. Activity-edge centric multi-label classification for mining heterogeneous information networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1285. ACM, 2014.
28. Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002.
29. Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.