POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Enhanced Graph Reconstruction using Graph Neural Networks

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Francesco Puddu

Advisor: Prof. Giacomo Boracchi

Co-advisor: Prof. Mauro Sozio

Academic year: 2021-2022

1. Introduction

Graphs are data structures that model a set of items connected by relationships, represented by nodes and edges respectively. Each node or edge can be described by a set of properties, called features. The graph structure is typically used to model a large number of systems from areas such as natural sciences, social networks, web search, recommendation systems and others.

Due to the high expressive power of graphs and their widespread use, there is growing interest in analysing them with Machine Learning techniques. This research area has proposed various successful applications, but recently great attention is being paid to the introduction of the class of models known as Graph Neural Networks (GNNs). Thanks to their effective results, they have quickly established themselves as the benchmark for graph analysis.

Models such as GNNs generally assume that the feature set associated with the graph is fully observable. However, this is often not the case in real-world scenarios, where each feature is generally only available in a subset of the nodes. For this reason, effective missing data estimation (an operation known as "imputation") for graph-structured data is now a critical topic to enable the widespread adoption of Machine Learning algorithms in this domain. The goal of this thesis is to present a novel unsupervised and parametric approach to graph-structured data imputation. The proposed solution learns directly on the graph from the available part of information - both structural and semantic - to reconstruct the missing features.

2. Problem Formulation

In this thesis, we tackle the task of feature imputation for graph-structured data. Given as input:

- An attributed graph $\mathcal{G}$ composed of N nodes. The connections are encoded by the adjacency matrix $A \in \{0,1\}^{N \times N}$.
- A feature matrix $X \in R^{N \times F}$, where F is the number of features, composed of a description vector for each node. Missing values are filled with a placeholder μ.
- A binary mask $M \in \{0,1\}^{N \times F}$ that specifies whether feature values are available (1) or not (0).

The goal is to produce a new feature matrix X' in which the originally missing features are imputed by the proposed model. Critically, the setting is unsupervised, as we do not require fully observable data even in the training phase.

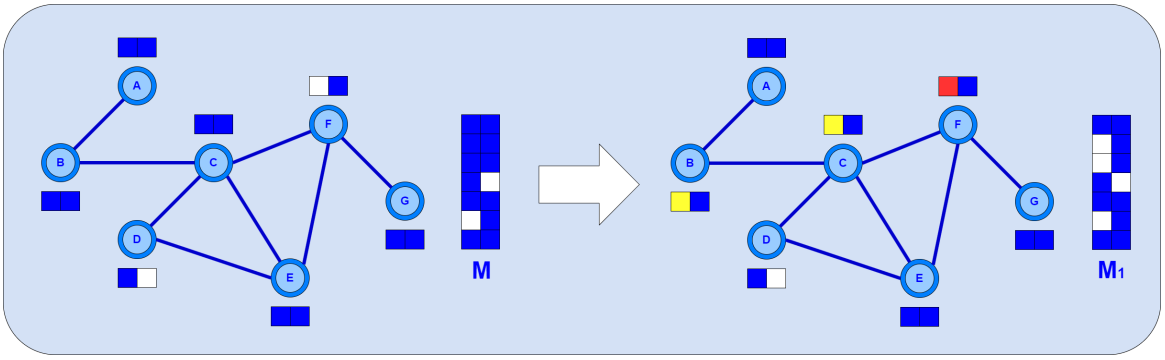

Figure 1: To optimize a reconstruction model for the first feature (on the left), a subset of the known (blue) values is hidden (yellow) to extract ground truth labels for training. At inference time, the model imputes actual unknown values (red).

Without loss of generality, we assume the missing mechanism of node features to be *missing at random* (MAR) and the missing pattern to be non-monotonic and multivariate. For our experiments, we create synthetic masks to derive X from the ground-truth feature matrix $\tilde{X}$:

$$x_{ij} = \begin{cases} \mu, & \text{if } m_{ij} = 0 \\ \tilde{x}_{ij} & \text{if } m_{ij} = 1 \end{cases} \quad (1)$$

The reference metrics for the experiments are:

- The reconstruction loss measured between real and imputed values.
- The performance loss in the downstream task. We evaluate this metric by measuring the difference in accuracy between inferences of the same model trained on $\tilde{X}$ and X'.

3. Related Work

Imputation solutions typically concentrate on the case where there are no defined relationships within the set, i.e. where the items are not graph-structured. For instance, probabilistic approaches based on explicitly learning the joint distribution of features [11]. Examples of local regression methods from the field of Machine Learning include K-Nearest Neighbors [1]. But more recently, Deep Learning-based solutions have established themselves. While approaches based on Generative Adversarial Networks [13] train the model to produce imputations that are undetectable by a discriminator from real data, autoencoder-based approaches aim to learn a functional data representation to impute missing data [5].

Although these solutions are directly applicable to the feature set of graph-structured data, they completely ignore the available structure of the graph, which intuitively constitute a valuable source of information for the imputation. Techniques that aim to use structure to improve the quality of reconstruction (such as the one presented in this work) include mainly interpolation algorithms derived from signal theory [7], and various methods based on the smooth propagation of known information on the graph structure. One of them, Feature Propagation [8], presents an interpolation process using messages exchanged iteratively across the graph edges.

Moreover, a family of methods addresses the problem of missing data by adapting the learning process rather than aiming at imputing the missing entries. The "sparsity normalisation" technique revisits zero imputation, the most straightforward and natural approach in this field of study [12]. Among the methods adapting the GNNs to the issue at hand, GCNmf [10] computes the expected activation of input layer neurons using a Gaussian Mixture Model to represent the missing data.

4. Background

The primary tasks in the field of Graph Machine Learning (GML) can be classified as supervised or unsupervised. The tasks - primarily classification or property regression - for the supervised methods can be at the level of a single node, an edge, or the entire graph. The emphasis of the work presented in this thesis is on supervised node-level tasks.

Graph Neural Networks, which serve as the main technology for the proposed solution, essentially consist of an iterative process which propagates the node representations on the graph structure until equilibrium; followed by a neural network, which produces an output for each node based on its final representation. A single propagation step updates the i-th node's latent vector h_i:

$$h_i^{(l+1)} = \sigma\left(\sum_j \frac{1}{c_{ij}} h_j^{(l)} W^{(l)}\right) \tag{2}$$

In order be consistent with the nomenclature used in the literature, we use the abbreviations x_i for the original node features and h_i for the subsequent latent representations. The combination of the vectors of each neighbour j, normalised by the factor c, is passed through a single layer neural network, where W are the layer weights and σ the non-linear activation.

A key distinction between Graph Neural Networks and previous solutions in the field of graph analysis is precisely how they incorporate the structural information of the graph into the learning process.

5. Proposed Solution

We present a novel approach to the feature reconstruction problem for graph-structured data. This will be articulated mainly in two aspects:

- A new formulation of the problem as an unsupervised node-level task.
- A GNN architecture to impute node features, called Graph Reconstruction Network (GRN) that employs custom convolutional layers aware of the partial lack of node features.

The learning procedure for the feature imputation task entirely relies on the input graph with partially observable features, as was mentioned in Section 2. To target the reconstruction of a (partially missing) feature f on $\mathcal{G}$, a subset t_f of its known instances is used to obtain ground truth labels for training purposes. The elements of t_f are hidden from the training data, thus defining a new mask matrix M_f. Note that the set complementary to t_f remains available to the model as a source of information. Figure 1 represents an example of the setting just described.

The rationale behind selecting GNNs for the imputation of node features is to exploit the dual nature of the information present in the input data:

- **Semantic**: encoded in the descriptive features of each node.
- **Structural**: encoded in the connections between nodes.

As discussed in Section 4, this family of models can indeed learn complex feature patterns on the graph for prediction.

Similar to CNNs in computer vision, this search is intrinsic to the iterated operation of updating the latent representations of nodes. Critically, the general form of graph convolution shown in (2) is not directly applicable to the case where we have missing features in the graph. This calls for the need to use a mask-aware variant in the input layer that takes the incomplete feature matrix X as input. We can define a masked convolution derived from the one proposed in [4]:

$$h_i^{(l+1)} = \sigma\left(\frac{\sum_j \left(m_j \odot h_j^{(l)}\right)}{\sum_j m_j} W^{(l)}\right) \tag{3}$$

By allowing the known information on the graph to propagate and subsequently update the vector representations of the nodes, this variant aims to eliminate the contributions of non-observable channels. This is achieved through the hadamard product between each feature vector h_j and the corresponding mask m_j at the numerator, which excludes the contribution of the missing entries from the process. Note that this is fundamentally distinct from, for instance, what we could accomplish by imputing missing values with zero: since neighbourhood aggregation typically involves averaging values, considering null entries would introduce noise into the process. In practice, this layer has been implemented using the matrix form of the previous equation:

$$H' \leftarrow \sigma[(A(M \odot H) \oslash AM)W] \tag{4}$$

Where $\odot$ and $\oslash$ denote element-wise multiplication and division respectively. Since both M and A are sparse matrices, the overall operation is computationally efficient.

The multi-layer architecture proposed for the model follows the typical structures of the Convolutional Neural Networks. The first phase is aimed at feature extraction and consists of a stack of convolutional layers that progressively update the latent representation of each node in the graph. In the second phase, a neural network (shared between the nodes) computes the actual prediction for each target node, taking the representation as input. The two phases are trained end-to-end.

As anticipated, only the first convolutional layer has to process representations from partially missing data and thus requires the mask-aware operator (3). A standard graph convolution (2) is implemented in subsequent layers.

The number of stacked convolutional layers, each with a dedicated set of weights W, is a hyper-parameter that is directly related to the concept of the receptive field. As mentioned in Section 4, each layer represents the aggregation of representations at a distance of 1-hop for each node. It makes intuitive sense that the stacking of l layers provides latent representations that are influenced by nodes located l hops away.

Although a dedicated predictive model must be trained for each feature to reconstruct, our experiments suggest that using lightweight architectures is sufficient in practice to ensure a competitive trade-off between training time, inference time, and prediction quality. In order to further lighten the training process, in this work we present an exploratory analysis of the efficacy of *subgraph sampling* techniques. Our first results suggest that the predictive model can generalize after being trained on a relatively small subset of nodes and the edges connecting them. We argue that this is due to the recurrence and locality of the most significant patterns in the neighbourhoods of the nodes.

6. Experiments

6.1. Datasets

We apply artificial masks to real-world datasets (graphs) in order to test the reconstruction performance. The domain of the chosen graphs is social networks, one of the areas of greatest interest for the task at hand.

We use publicly available data from the MUSAE project [9] database:

- **GitHub**: nodes (37k) stand for developers, and the edges (289k) represent the connections between them. The professional profile of the developer is encoded in the node features. Labels show whether a node represents a specialist in the field of machine learning based on the job title.
- **Facebook**: official Facebook pages are represented by nodes (22k), and site-to-site mutual likes are represented by edges (171k). The site descriptions that the page owners provided to summarize the purpose of the site are the source for node features. Labels such as "company" and "government organization" identify the category to which the page belongs the social platform.
- **Twitch**: nodes (7k) represent users and edges (35k) are mutual follower relationships between them. Vertex features are extracted based on the activity of the specific user on the platform. Labels indicate whether the user has been flagged for the use of explicit language.

As anticipated in Section 2, we apply synthetic masks on these datasets, for which we have the complete feature matrix. The use of artificial masks allows us to measure the quality of the reconstruction against ground truth values. The procedure we used to produce masks that are consistent with the data assumptions and that represent various missingness scenarios is essentially based on two probabilistic distributions: a skew-normal ϕ_1 and a Bernoullian ϕ_2. For each feature, a first sample from ϕ_1 specifies the missingness rate, which serves as parameter for ϕ_2. Finally, we sample from the Bernoullian whether each value is available (1) or not (0). As required by the MAR hypothesis, the result is independent of the value of the single features.

We are able to flexibly investigate various missingness scenarios thanks to the shape parameter of ϕ_1. In this thesis, we thoroughly discuss this aspect to assess the robustness of GRN in various scenarios. For the sake of brevity, in this summary we only present the results obtained in the intermediate scenario in which the parameter is set to 0 and ϕ_1 is thus equivalent to a normal distribution.

	Reconstruction Loss			Downstream Loss		
	GitHub	**Facebook**	**Twitch**	**Github**	**Facebook**	**Twitch**
GM	0.488	0.375	0.530	0.354	0.283	0.412
NM	0.429	0.318	0.422	0.292	0.258	0.331
VAE	0.313	0.239	0.350	0.225	0.202	0.267
GAN	0.307	0.240	0.341	0.242	0.210	0.253
FP	0.226	0.201	0.258	0.196	0.185	0.224
KNN	0.231	**0.195**	0.243	0.201	**0.184**	0.236
GRN	**0.172**	0.211	**0.208**	**0.165**	0.186	**0.191**

Table 1: Comparison of feature reconstruction models

	Downstream Loss		
	GitHub	**Facebook**	**Twitch**
GCNmf	0.168	**0.169**	0.178
GRN & GCN	0.165	0.186	0.191
GRN & Spline	**0.119**	0.179	**0.160**

Table 2: Comparison of complete node classification pipelines

6.2. Network Architecture

For each dataset, we carry out a tuning process of hyper-parameters such as the number of graph convolutions, latent dimensions, weight decay, and the dropout rate. Adam serves as our SGD optimizer. Although it makes intuitive sense that the complexity of the architecture tends to rise as the complexity of the reconstruction task rises, it is interesting to note that in the proposed experimental setting, the competitiveness of lightweight models emerges empirically. In particular the optimal number of model parameters does not exceed 400k for any of the three benchmarks under consideration.

6.3. Metrics

As anticipated in Section 2, we assess various experiments in light of two relevant dimensions:

- **Reconstruction Loss**: similar to most studies, we report *RMSE*, with the error being $|\tilde{X} - X'|$. The measure is normalised with respect to the loss corresponding to zero imputation, for ease of interpretation.
- **Downstream Loss**: since this is a node classification task, we report the difference in terms of *average class accuracy* score.

6.4. Compared Methods

We compare the proposed Graph Reconstruction Network to various state-of-the-art feature imputation solutions applied on graphs. Moreover, as a baseline for each experiment, we report the results obtained with naive statistical methods like as global (GM) and neighbourhood-based (NM) mean imputation. The representative imputation approaches we select from Section 3 are Variational Autoencoder (VAE), Generative Adversarial Network (GAN), and Feature Propagation (FP).

We also compare against GCNmf, a method that extends GNN architectures to directly train on incomplete graphs for the downstream task (such as node classification) without an explicit intermediate feature imputation step. As it aims directly at the downstream task, GCNmf is not directly comparable with GRN. However, we can achieve a comparison by combining GRN with downstream classification models that complete the pipeline.

Finally, to further enrich the experimental setting, we present an approach that extends the KNN approach to the graph domain by defining a similarity function based on the graph structure. The rationale of this method is to exploit known node embedding techniques (in this work we resort to Node2Vec [3]) to project each node of the graph onto a continuous space in which distances encode graph proximity. This allows to exploit a KNN regression strategy based on Euclidean distances between nodes. To the best of our knowledge, this is the first such approach in the context of imputation on graphs.

6.5. Results

The first set of experiments compares imputation models. To assess the performance loss, we employ a standard GCN [6], which we train for the downstream classification task. The results, in Table 1, demonstrate a competitive performance on both metrics. In terms of comparison, the proposed solution slightly outperforms the other models on the GitHub and Twitch benchmarks. The KNN-based method performs best in both metrics on the Facebook dataset and demonstrates alignment with FP, with whom it shares the iterative local smoothing approach.

For the second set of experiments, aimed at comparing the proposed solution against GCNmf, we resort to two downstream classification models, again a standard GCN and a Spline [2] architecture, which employs a more sophisticated convolution strategy based on B-splines. The results, in Table 2, once more demonstrate a competitive reconstruction quality by the proposed solution on the benchmarks. The best results, in particular, are obtained against GCNmf on the GitHub and Twitch datasets. Empirically, the resulting performance appears to be strongly influenced by the selected downstream classification model: the pipeline using Spline consistently reports the best results.

7. Conclusions

In summary, we addressed the problem of imputing missing node features in a graph by framing it as a node-level Machine Learning task. The proposed solution consists of a novel GNN architecture in which convolutions exploit both the graph structure and available features to predict the missing ones. Such architecture is trainable independently of the availability of a complete graph. We tested the proposed solution on real-world benchmarks in various conditions showing a consistent competitive performance in terms of imputation quality against the actual state-of-the-art solutions. The results indicate that applying Graph Neural Networks to the problem of imputing missing node features on a graph is a promising research direction to enable higher quality estimations.

References

[1] Batista and Monard. A study of k-nearest neighbour as an imputation method. 2002.

[2] Fey, Lenssen, Weichert, and Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels, 2018.

[3] Grover and Leskovec. node2vec: Scalable feature learning for networks, 2016.

[4] Jiang and Zhang. Incomplete graph representation and learning via partial graph neural networks, 2021.

[5] Kingma and Welling. Auto-encoding variational bayes. 2013.

[6] Kipf and Welling. Semi-supervised classification with graph convolutional networks. 2016.

[7] Narang, Gadde, and Ortega. Signal processing techniques for interpolation in graph structured data. 2013.

[8] Rossi, Kenlaya, Gorinova, Chamberlain, Dong, and Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features, 2022.

[9] Rozemberczki, Allen, and Sarkar. Multi-scale attributed node embedding, 2021.

[10] Taguchi, Liu, and Murata. Graph convolutional networks for graphs containing missing features. 2020.

[11] van Buuren and Groothuis-Oudshoorn. Mice: Multivariate imputation by chained equations in r. 2011.

[12] Yi, Lee, Hwang, and Yang. Sparsity normalization: Stabilizing the expected outputs of deep networks. 2019.

[13] Yoona, Jordon, and van der Schaar. GAIN: missing data imputation using generative adversarial nets. 2018.