Handling Noisy Constraints in Semi-supervised Overlapping Community Finding

Elham Alghamdi, Ellen Rushe, Mehran H.Z. Bazargani,
Brian Mac Namee, and Derek Greene

School of Computer Science, University College Dublin, Ireland

1 Introduction

Community structure is an essential property that helps us to understand the nature of complex networks. Since algorithms for detecting communities are unsupervised in nature, they can fail to uncover useful groupings, particularly when the underlying communities in a network are highly overlapping [1]. Recent work has sought to address this via semi-supervised learning [2], using a human annotator or "oracle" to provide limited supervision. This knowledge is typically encoded in the form of must-link and cannot-link constraints, which indicate that a pair of nodes should always be or should never be assigned to the same community. In this way, we can uncover communities which are otherwise difficult to identify via unsupervised techniques.

However, in real semi-supervised learning applications, human supervision may be unreliable or "noisy", relying on subjective decision making [3]. Annotators can disagree with one another, they might only have limited knowledge of a domain, or they might simply complete a labeling task incorrectly due to the burden of annotation. Thus, we might reasonably expect that the pairwise constraints used in a real semi-supervised community detection task could be imperfect or conflicting. The aim of this study is to explore the effect of noisy, incorrectly-labeled constraints on the performance of semi-supervised community finding algorithms for overlapping networks. Furthermore, we propose an approach to mitigate such cases in real-world network analysis tasks. We treat noisy pairwise constraints as anomalies, and use an autoencoder, a commonly-used method in the domain of anomaly detection, to identify such constraints. Initial experiments on synthetic network demonstrate the usefulness of this approach.

2 Methods and Experimental Design

The key aspect of our work is an iterative approach using an autoencoder to remove noisy pairwise constraints selected by the AC-SLPA algorithm [2]. An *autoencoder* (AE) refers to a neural network architecture that attempts to reconstruct a given input in an effort to learn an informative latent feature representation. Formally, for an input vector x, we attempt to map x to a reconstruction of itself x'. By doing this, a latent representation of the data is created in the hidden layer(s) of the network [4]. These networks can utilize a "bottleneck" configuration where the hidden layer(s) of the network compress the data [4]. The network is trained by minimizing the mean squared error (MSE) between the reconstruction and input. Additionally, autoencoders can be constrained to enforce sparsity in the network and therefore no longer require a compressed network capacity. One type of constrained autoencoder adds a sparsity penalty to hidden representations by constraining their absolute value. This penalty term is weighted and added to the cost function. In our work we employ the above neural network architecture to identify potentially noisy pairwise constraints selected by AC-SLPA before applying the community detection process.

Firstly, feature vectors are constructed as inputs to the autoencoder, one vector per input constraint pair. Along with the constraint type, the other features include standard measures based directly on the network topology: whether the pair of nodes shares an edge, their number of common neighbors, shortest path length, and cosine similarity. We also include more complex features: their SimRank similarity [6] and their similarity as computed on a *node2vec* embedding generated on the network [5]. From this data, the model then learns to reconstruct the original constraints from the latent representation. The reconstruction error is then given by the difference between the original constraints and the reconstruction. A large error is indicative of an anomaly (i.e. a noisy constraint), while a low error indicates a "normal" example (i.e. a correctly-labelled constraint). The expectation is that, as the vast majority of pairwise constraints are non-noisy, the autoencoder's latent representation will be biased towards these examples. This makes the model somewhat robust to outliers. Based on this property, it is then assumed that examples which are noisy will have a high reconstruction error.

As our initial evaluation, we assess the capability of autoencoders to detect noisy constraints. Once the set of constraints is selected by AC-SLPA and labeled by the oracle, the autoencoder is trained on this set. These are then passed through the autoencoder once again to obtain a reconstruction error for each constraint. The AUC over this error is calculated, which provides an estimate of the number of constraints that were successfully detected in the absence of a definitive threshold. The number of layers in each autoencoder is varied to examine whether this task benefits from a deeper model. We consider both compression-based autoencoders and sparse autoencoders.

Evaluations are performed on 64 LFR benchmark networks containing either small or large communities, for a variety of parameters $\{N, Om, On, \mu\}$ (see Table 1). The depth of the autoencoder is varied to assess its effect on performance. In the case of the compression autoencoders, the nodes are gradually decreased in the encoder and increased in the decoder, while this compression is not necessary for the L1 constrained models [4]. In the case of the constrained autoencoders, the sparsity weight is kept at 10^{-5}. All models were trained with a learning rate of 10^{-3} for a maximum of 100 epochs and a batch size of 256.

3 Results

The results in Table 1 are divided into two parts, which represent the AUC scores of the autoencoder on networks with 10% and 50% overlapping nodes respectively, averaged across 10 runs. Each table entry shows the AUC value of an AE model (on the rows) for each network (on the columns). For each network, the AUC scores of AE models are ranked, and the best performance is highlighted in bold. The last column reports the average rank score of each model. As we can see, all AE models show high AUC scores,

Table 1: AUC scores on LFR networks with 10% of noise in pairwise constraints. AE* [layers dimension]: indicates the number of layers in compression autoencoders, and AE*_l1 [layers dimension]: indicates the number of layers in L1 constrained autoencoders: AE1: [7,3,7], AE1_L1: [7,7,7], AE2: [7,5,3,5,7], AE2_L1: [7,7,7,7,7], AE3: [7,6,5,3,5,6,7], AE3_L1: [7,7,7,7,7,7,7].

(a) AUC scores on networks with 10% overlapping nodes

Comm. size	Large Communities								Small Communities								Average Rank
μ	0.1				0.3				0.1				0.3				
Om	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	
AE1	0.752	0.736	0.777	0.736	0.829	0.770	0.744	0.751	0.775	0.757	0.756	0.773	0.795	0.733	0.773	0.739	4.4 (4)
AE1_L1	0.759	0.800	0.801	0.758	**0.837**	0.772	**0.829**	0.732	**0.832**	0.828	0.774	0.766	**0.810**	0.824	0.826	0.759	2.9 (2)
AE2	0.760	0.739	**0.803**	0.776	0.797	0.787	0.798	0.749	0.783	0.795	0.765	0.764	0.786	0.780	0.775	0.773	3.3 (3)
AE2_L1	**0.762**	0.706	0.791	0.760	0.792	**0.801**	0.789	0.784	0.775	0.795	0.798	0.769	0.770	0.834	**0.831**	0.813	2.9 (2)
AE3	0.754	**0.809**	0.771	**0.810**	0.794	0.797	0.796	**0.792**	0.817	**0.833**	**0.836**	**0.822**	0.769	**0.839**	0.827	**0.849**	**2.3 (1)**
AE3_L1	0.720	0.777	0.773	0.776	0.779	0.751	0.764	0.740	0.729	0.753	0.726	0.793	0.786	0.795	0.774	0.782	4.4 (4)

(b) AUC scores on networks with 50% overlapping nodes

Comm. size	Large Communities								Small Communities								Average Rank
μ	0.1				0.3				0.1				0.3				
Om	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	
AE1	0.744	0.797	0.823	0.793	**0.832**	0.815	0.804	**0.798**	0.779	**0.829**	0.828	0.778	0.813	0.831	0.849	0.806	3.3 (3)
AE1_L1	0.766	0.804	0.811	**0.834**	0.788	0.788	0.826	0.766	**0.847**	0.755	**0.874**	**0.818**	0.756	0.826	0.842	**0.836**	3.1 (2)
AE2	0.741	0.767	0.799	0.668	0.771	0.806	0.803	0.676	0.783	0.760	0.784	0.799	0.762	0.808	0.823	0.812	4.9 (5)
AE2_L1	**0.780**	0.798	0.791	0.833	0.794	**0.827**	0.818	0.783	0.801	0.776	0.867	0.790	0.792	**0.879**	0.848	0.819	**2.6 (1)**
AE3	0.696	0.720	0.706	0.757	0.782	0.745	0.823	0.779	0.822	0.770	0.858	0.669	**0.820**	0.837	0.808	0.803	4.4 (4)
AE3_L1	0.752	**0.824**	**0.835**	0.805	0.831	0.808	**0.837**	0.774	0.787	0.785	0.860	0.669	0.801	0.853	**0.852**	0.811	**2.6 (1)**

with the lowest scores around 70%. However, we see the AE3 models perform better on networks with $On = 10\%$, while AE1_L1 and AE2_L1 also perform well here. On the networks with $On = 50\%$, AE2_L1 and AE3_L1 are the top-ranked models. In general, these results suggest that deeper autoencoder models do not perform significantly better than simpler ones when detecting noisy constraints.

We have proposed a novel approach to handle noisy pairwise constraints in semi-supervised community finding for overlapping networks. This approach considers noisy onstraints as outliers and uses autoencoders to detect this noise. We employed this approach with the AC-SLPA algorithm [2], yielding promising results in detecting noisy constraints on benchmark networks. A second set of experiments is currently in progress, which directly evaluates the performance of AC-SLPA when incorporating reliable constraints as selected by the autoencoder model, on both synthetic and real-world networks.

Acknowledgements. This research was partly supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2.

References

1. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. Nature 466(7307), 761–764 (2010)
2. Alghamdi, E., Greene, D.: Active semi-supervised overlapping community finding with pairwise constraints. Applied Network Science 4 (2019)

3. Amini, M.R., Gallinari, P.: Semi-supervised learning with an imperfect supervisor. Knowledge and Information Systems 8(4), 385–413 (2005)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
5. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proc. SIGKDD'16. pp. 855–864. ACM (2016)
6. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: Proc. SIGKDD'02. pp. 538–543. ACM (2002)