Learning Social Graph Topologies using Generative Adversarial Neural Networks

Sahar Tavakoli[1], Alireza Hajibagheri[1], and Gita Sukthankar[1]

[1]University of Central Florida, Orlando, Florida
sahar@knights.ucf.edu,alireza@eecs.ucf.edu,gitars@eecs.ucf.edu

Abstract. Although sources of social media data abound, companies are often reluctant to share data, even anonymized or aggregated, for fear of violating user privacy. This paper introduces an approach for learning the probability of link formation from data using generative adversarial neural networks. In our generative adversarial network (GAN) paradigm, one neural network is trained to generate the graph topology, and a second network attempts to discriminate between the synthesized graph and the original data. After the generative network is fully trained, the learned weights can be disseminated and used to "clone" the hidden dataset with minimal risk of privacy breaches. We believe that the learned neural network also has the potential to serve as a more general model of social network evolution.

Keywords: synthetic network generator; link formation; generative adversarial networks

1 Introduction

A key issue in agent-based modeling is how to create a synthetic population with realistic behaviors from a limited amount of survey and census data. To achieve maximum verisimilitude with minimal data collection effort, the modeler must focus on the subset of agent attributes most likely to affect the outcome of the simulation scenario. However, since humans are by nature social animals, accurately modeling social connections has cross-cutting impact for any type of simulation in which the humans are affected by their peers due to social influence, norms, collective behavior, and group dynamics. For instance, in a city evacuation scenario, Parikh et al. [1] show significant changes in the outcome of the simulation based on the addition of family regrouping behaviors. Hence for many scientific questions, accurately reconstructing the topology of the peer network can dramatically affect the simulation fidelity.

Synthetic network generators can be used to create network topologies for agent-based social simulations in cases where limited network data is available. Most generators assume that the social network fits one of the standard mathematical models of network formation and rewiring, When data is available, fitting techniques can be used to select the model parameters that best match the data. However, human networks often emerge from the confluence of many

social processes and are not well modeled by a single model, limiting the applicability of parametric models of network generation. Moreover, these techniques are difficult to extend to networks with node features or multiple layers.

Rather than assuming a specific parametric model, we introduce a semi-supervised model of network generation in which the network topology is learned by generative adversarial neural networks. Generative adversarial networks (GANs) [2] can approximate hidden probability distributions using a competitive learning process in which one neural network generates a sample and a second neural network attempts to discriminate between synthetic samples and examples drawn from the real distribution. In the domains of computer generated artwork and computer vision, GANs have a proven track record of being able to synthetically generate images capable of fooling human observers. We train our GANs by treating the network adjacency matrix as an image; large datasets can be created by sampling the permutation distribution of a single example network. This paper describes our training procedure and compares the GAN output to a synthetic network generator that attempts to directly match network attributes using stochastic optimization (Attribute Synthetic Generator [3]). We demonstrate that our GAN can successfully learn the topology of networks with very different degree distributions, making it more versatile than standard mathematical models.

2 Related Work

Rather than learning the network topology from data, most synthetic network generators start with a detailed model of social interactions. In an agent-centric approach, the social interactions are integrated into the agent's activity and mobility model [4]; this method is well suited for cases where there is ample information about the subjects' preferences and constraints. In contrast, network-centric approaches are better equipped to examine a single facet of human behavior (e.g. social media usage) across a large population. Random graphs are powerful at capturing cross-domain commonalities in network topologies; models such as Erdös-Renyi [5], small world [6], and preferential attachment [7] have been successfully used to generate networks resulting from both computational and biological systems. Statistical tools for analyzing networks often implicitly assume that the network structure is represented as an exponential random graph and estimate the parameters with Monte Carlo sampling [8]. Although ERGMs can be used to generate new networks, they may experience problems converging and are better suited to validating experimental hypotheses.

Unfortunately random graph models often fail to accurately recreate key patterns in human social networks including power law degree distributions, community structure, and self-similarity, leading to the introduction of matrix models such as R-Mat [9] and Kronecker graphs [10]. Recursive matrix multiplications naturally create similarity across multiple scales, and fitting procedures can be used to select the best parameter values for copying the degree distribution of specific datasets.

Note that mimicking graph topology is only one aspect of cloning real datasets, which often contain node features as well. When node feature data is drawn from user datasets, privacy preservation becomes an issue. Nettleton [11] specifically addresses the anonymization problem by including a variable level of dispersion in the cloning process. In this paper, we benchmark our method against the Attribute Synthetic Generator [3] which simultaneously clones the topology and node features using a combination of random graph generation to form connections and stochastic optimization to match feature distributions.

3 Method

In contrast to prior work, our approach makes no assumptions about the social network structure. Instead, we learn the link probability distributions in a semi-supervised way using a pair of generative adversarial neural networks. This section provides an introduction to generative adversarial networks (GANs), describes our neural network architecture, data preparation and training procedures.

3.1 Generative Adversarial Networks

After observing examples from the true distribution, GANs can synthesize new data without an explicit model. We follow the original training paradigm introduced by Goodfellow et al. [2] in which two neural networks, a generator and discriminator, are simultaneously trained. The generator tries to synthesize data capable of "fooling" the discriminator which endeavors to separate the generator output from the true samples. Feedback from the discriminator about the convincingness of the synthetic samples is then used by the generator to modify its network weights (Figure 1).

During the training process, the generator (G) and the discriminator (D) seek to respectively minimize and maximize the value function of $V(D, G)$ in a min-max game:

$$\min_G \max_D V(D,G) = E_{x \sim P_{data}(x)}[log(D(x))] + E_{z \sim P_z(z)}[log(1 - D(G(z)))]$$

G receives noise samples from an arbitrary $P_z(z)$ noise distribution and creates new samples. D is a classifier network that outputs the probability of each sample belonging to the true distribution. The goal of D is to output a label of zero when it receives a sample generated by G as input and to output a label of one when it observes x drawn from the true distribution $P_{data}(x)$. In their seminal work, Goodfellow et al. proved that if G and D have enough capacity, the value function will attain the optimum, and it is reachable when the generated samples match the true distribution [2].

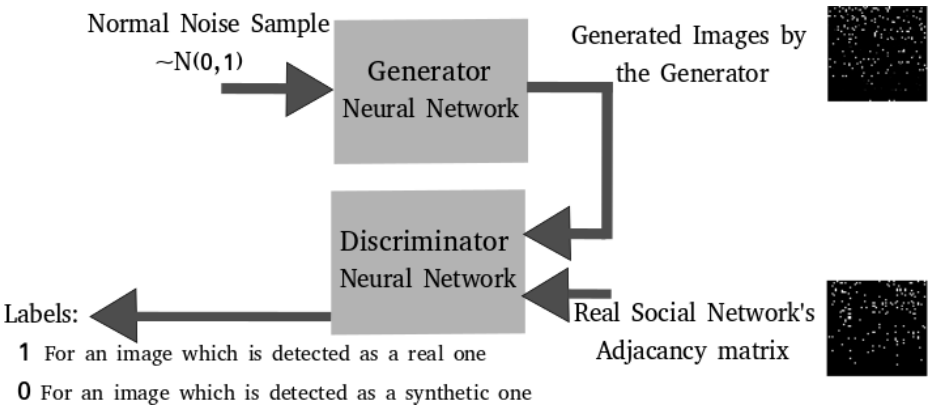

Fig. 1: Generative adversarial neural networks. GANs employs a competitive learning procedure that can be treated as a min-max game between the networks. The generator (G) takes an input of normal noise samples and generates an output adjacency matrix, visualized as a greyscale image. The aim of the discriminator (D) is to correctly label the graph topologies synthesized by the generator (G) with zeros and the real network data with ones.

3.2 Network Architecture

We experimented with multiple neural network architectures before adopting the following procedure. To synthesize the graph topology for the four social networks, the discriminator was provided with flattened adjacency matrices from the original networks, along with 10,000 permutations of the adjacency matrix with randomly swapped node indexes. These matrices can be treated and visualized as greyscale images. Increasing the diversity of input samples makes it more difficult for the GAN to memorize the training set and has been shown to improve the quality of the synthesized data [12].

Our discriminator network includes one linear hidden layer, followed by a rectified linear unit (ReLU) activation layer, and an output layer with a sigmoid activation function (Figure 2). The generator input consists of 100 random samples drawn from a normal distribution ($N(0, 1)$). Our generator network consists of two linear hidden layers each with 1000 neurons, batch normalization and ReLU activation functions, and an output layer with n^2 sigmoid units where n is the number of nodes in the original network (Figure 3).

Batch normalization helps deeper neural networks learn faster and achieve better accuracy by reducing the internal covariate shift [13]. Internal covariate shift is the deviation from zero mean and unit variance which is produced by the output of hidden layers during training. If the original dataset was a sparse network, only one hidden layer with 500 nodes was used during training. In our experiments, the sparsity of the network was determined by dividing the number

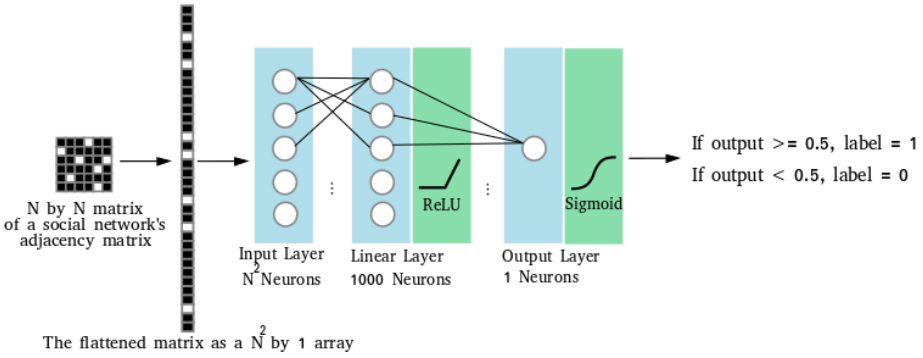

Fig. 2: Architecture of the discriminator network (D). The input is a flattened adjacency matrix, followed by a linear layer with 1000 neurons and a ReLU activation function. The discriminator seeks to produce a label of 1 when it is presented with from data from the real network or its permutations and a label of 0 for a sample synthesized from the generator.

of edges by the number of nodes and treating it as sparse if the ratio is less than or equal to ten.

3.3 Training Procedure

Despite the simplicity of the GAN model, the training procedure can be tricky because the discriminator and generator must improve in a harmonious fashion such that the discriminator challenges the generator without decisively beating it at the min-max game. The training process starts by training the discriminator, as the generator needs the error output from the trained discriminator to progress to the next level. The key point is that D should not be trained completely, which leads to overfitting, and does not improve G in the next level.

Each network is trained using stochastic gradient descent (SGD), and this alternating training process was repeated 1000 times. 10,000 data samples (the original adjacency matrix plus permutations) were presented to the discriminator along with the same number of synthetic samples from the generator in batches of 100. Learning rates for the discriminator and the generator were set to 0.0001 and 0.01 respectively.

The final output of the generator is a matrix with values ranging from zero to one, visualized as a greyscale image. To use this output as a graph adjacency matrix, it needs to be thresholded using a fitting procedure to clone the original dataset. For each dataset, we selected the best of 25 threshold levels using two features to determine the best candidate: 1) edges and 2) degree assortativity.

Our implementation uses the Torch scientific framework [14], and the experiments were performed on a PC with an Intel Xeon(R) CPU W3565 @ 3.20GHz 4 and 23.5 GB memory.

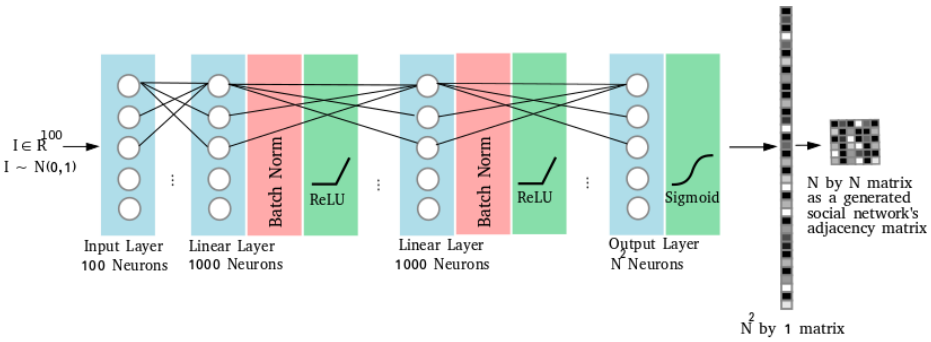

Fig. 3: Architecture of the generative network (G) used for dense social network data. The input is 100 random samples drawn from a normal distribution and the network consists of two linear hidden layers (1000 neurons), with batch normalization and ReLU activation functions. The output is an array ranging from 0 to 1 which can be binarized to an adjacency matrix after a thresholding process. For the sparse social network, we use a single hidden layer with 500 neurons.

4 Results

To evaluate the performance of our approach, we learned the graph topology for four real-world networks. These networks were selected from different domains and are considerably disparate in structure:

- **Enron:** The Enron e-mail dataset [15] is a repository of e-mails exchanged between the executives of the former Enron Corporation. Vertices represent email addresses, and each e-mail sent to or by an executive can be considered as a directed and timestamped edge in a dynamic social network. This was the only network that qualified as dense according to our criteria.
- **Karate Club:** The Zachary karate club network [16] depicts the social relationships between karate club members. It is divided into two roughly equal-sized groups due to a disagreement between club administrator and the principal karate teacher.
- **American Football:** The American college football network represents the schedule of games during the 2000 season. This network was previously used by Girvan and Newman [17], and its community structure is known. Each node represents a football team, and each edge marks a game between two teams. Teams are divided into 12 conferences, and we can treat each conference as a community since games are held more frequently between teams of the same conference.

Table 1: Statistics of real networks vs. generated networks

Feature	Method	Enron	Karate	Football	Dolphins
No. of Nodes	Real	154	34	115	62
	GAN	154	34	115	62
	ASG	154	34	115	62
No. of Edges	Real	1917	78	613	159
	GAN	2288	77	472	153
	ASG	497	103	360	194
Average degree	Real	24	4	10	5
	GAN	29	4	8	5
	ASG	6	6	6	6
Max degree	Real	138	19	21	19
	GAN	68	17	19	15
	ASG	50	18	42	28
Min degree	Real	5	1	1	1
	GAN	8	1	1	1
	ASG	1	1	1	2
Assortativity	Real	-0.068	-0.547	-0.144	-0.174
	GAN	-0.093	-0.250	-0.146	-0.127
	ASG	-0.140	-0.035	-0.080	-0.127
Network Diameter	Real	3	4	5	5
	GAN	3	5	5	5
	ASG	8	5	8	6
Avg. Path Length	Real	1.923	2.157	2.335	2.620
	GAN	1.837	2.456	2.460	2.558
	ASG	3.586	2.770	3.619	3.190
Runtime (hrs)	Real	-	-	-	-
	GAN	62.26	2.92	33.47	9.27
	ASG	< 1	< 1	< 1	< 1

– **Dolphin:** The dolphin network [18] represents the relationships of 62 bottlenose dolphins; based on the biologists' observations, there are two groups in the network.

Table 1 provides the network statistics for each of the real datasets, along with the version of each network cloned using the GAN. As a benchmark, we also provide network cloning results from the Attribute Synthetic Generator [3] which uses stochastic optimization. Although it is difficult to evaluate the generality of the learned link formation model, we can examine the performance of its fitting procedure for cloning networks. Compared to ASG, the GAN performs slightly better at recreating many aspects of the network topology across the four datasets. Note that the time required to generate a network using a GAN is significantly higher than ASG, particularly for the dense Enron network.

Figure 4 shows degree distributions of the real networks (left) as well as the synthetic version generated by the GAN (right). Although many networks possess power law degree distributions, this is not always true for networks generated from different social processes; our four datasets display considerable diversity in the degree distributions. The best model for each network (synthetic and real) was selected based on the value of R^2 which indicates the goodness of a fit (see figure inset for model and R^2 value). As shown in the figures, the GAN successfully recreates the degree distributions of the original networks; however unlike most synthetic network generators, the GANs do not assume that the degree distribution will follow a specific form.

5 Conclusion and Future Work

This paper introduces a novel semi-supervised approach for learning graph topologies of social networks using generative adversarial neural networks. We illustrate that it is possible to train the GANs using a single network snapshot by augmenting the dataset provided to the discriminator with permutation samples. After the generative network is fully trained, the learned weights can be disseminated and used to clone the hidden dataset with minimal risk of privacy breaches. It is also possible to increase the level of anonymization by adding additional noise to the samples provided to the discriminator.

There are several promising directions for future work. For instance, alternative network representations such as stochastic block models would be useful for scaling the learning to larger networks. Introducing convolutional layers into the network might facilitate the learning of local structures such as communities. Recurrent neural networks have been used to generate many types of sequential data and could be extended to the problem of learning network dynamics. In the future, we plan to create new architectures for synthetically generating more complex graph topologies including dynamic and multilayer networks.

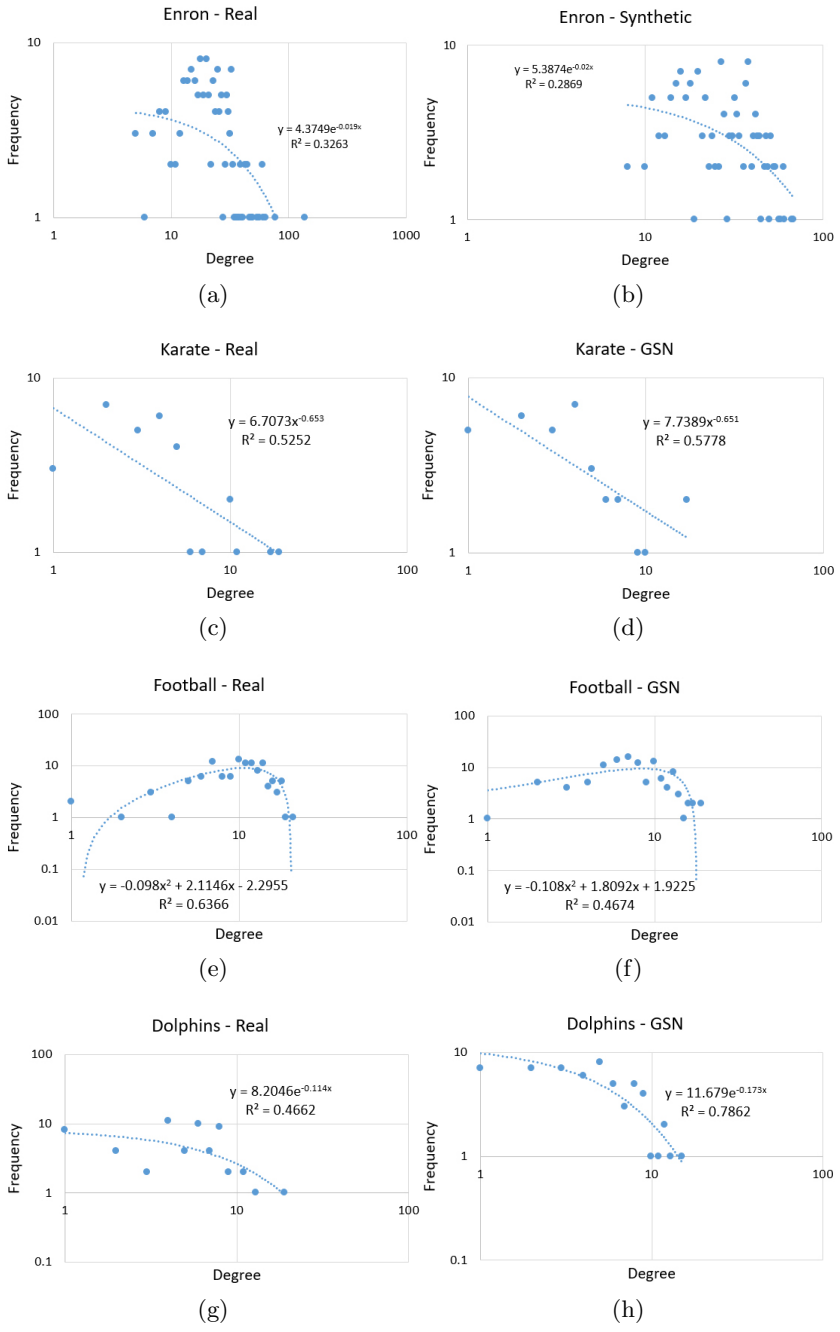

Fig. 4: Degree distributions of different real networks vs. networks generated by GAN. Note that the networks do not always follow a power law distribution. The best fitting distributions were selected using the R^2 fitness value which is displayed at the bottom of the figures, along with the model. The GAN does well at matching the distributions of the original networks without any *a priori* knowledge.

6 Acknowledgments

Research at University of Central Florida was supported with an internal Reach for the Stars award.

References

1. Parikh, N., Hayatnagarkar, H.G., Beckman, R.J., Marathe, M.V., Swarup, S.: A comparison of multiple behavior models in a simulation of the aftermath of an improvised nuclear detonation. Autonomous Agents and Multi-agent Systems (2016)
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. (2014) 2672–2680
3. Ali, A.M., Alvari, H., Hajibagheri, A., Lakkaraju, K., Sukthankar, G.: Synthetic generators for cloning social network data. In: Proceedings of the ASE International Conference on Social Informatics. (2014) P41:1–9
4. Bernstein, G., O'Brien, K.: Stochastic agent-based simulations of social networks. In: Proceedings of the Annual Simulation Symposium, Society for Computer Simulation International (2013)
5. Erdös, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci **5** (1960) 17–61
6. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393**(6684) (1998) 440–442
7. Barabási, A., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Vicsek, T.: Evolution of the social network of scientific collaborations. Physica A **311**(3–4) (2002) 590–614
8. Snijders, T.A.B.: The statistical evaluation of social network dynamics. Sociological Methodology **31** (2001) 361–395
9. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: A recursive model for graph mining. In: SIAM International Conference on Data Mining. (2004) 442–446
10. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C.: Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In: Knowledge Discovery in Databases: PKDD. Springer (2005) 133–145
11. Nettleton, D.F.: A synthetic data generator for online social network graphs. Social Network Analysis and Mining **6** (2016)
12. Im, D.J., Kim, C.D., Jiang, H., Memisevic, R.: Generating images with recurrent adversarial networks. arXiv preprint arXiv:1602.05110 (2016)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning. (2015)
14. Torch: `http://torch.ch/`.
15. Shetty, J., Adibi, J.: The Enron email dataset database schema and brief statistical report. Technical Report 4, Information Sciences Institute, University of Southern California (2004)
16. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical Review E **76**(3) (2007) 036106
17. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69**(2) (2004) 026113
18. Lusseau, D.: The emergent properties of a dolphin social network. Proceedings of the Royal Society of London B: Biological Sciences **270**(Suppl 2) (2003) S186–S188