Policy Learning for Task-driven Discovery of Incomplete Networks

Peter Morales
MIT Lincoln Laboratory
peter.morales@ll.mit.edu

Rajmonda Sulo Caceres
MIT Lincoln Laboratory
rajmonda.caceres@ll.mit.edu

Tina Eliassi-Rad
Northeastern University
t.eliassirad@northeastern.edu

Abstract

Complex networks are often either too large for full exploration, partially accessible or partially observed. Downstream learning tasks on these incomplete networks can produce low quality results. In addition, reducing the incompleteness of the network can be costly and nontrivial. As a result, network discovery algorithms optimized for specific downstream learning tasks given resource collection constraints are of great interest. In this paper we formulate the task-specific network discovery problem in an incomplete network setting as a sequential decision making problem. Our downstream task is selective harvesting, the optimal collection of vertices with a particular attribute. We propose a framework, called Network Actor Critic (NAC), which learns a notion of future reward and policy in an offline setting via a deep reinforcement learning algorithm. A quantitative study is presented on several synthetic and real benchmarks. We show that offline models of reward and network discovery policies lead to significantly improved performance when compared to competitive online discovery algorithms.

1 Introduction

Complex networks are critical to many applications such as those in the social, cyber, and bio domains. We commonly have access to partially observed data. The challenge is to discover enough of the complex network so that we can perform a learning task well. This presents an optimization problem: how should we grow the incomplete network to achieve a learning objective on the network, while at the same time minimizing the cost of observing new data? In this work we view the network discovery problem from a decision theoretic lens, where notions of utility and resource cost are naturally defined and jointly used in a sequential, closed-loop manner. In particular, we leverage Reinforcement Learning (RL) and its mathematical formalism, Markov Decision Processes (MDP). RL approaches have been successfully used in many other application settings [1, 2, 6, 7, 8, 9]. However, the use of RL techniques in the context of task-driven discovery of large complex networks is a relatively new area of research [5]. We consider the discovery of an incomplete network in support of the selective harvesting objective [4], where the goal is to maximize the collection of nodes of a particular type, under budget constraints. We make the following contributions: **1)** We introduce an efficient deep RL framework for task-driven, incomplete network discovery. This formulation allows us to learn offline-trained models of environment dynamics and reward. **2)** We show that for a variety of complex learning scenarios, the added feature of learning from closely related scenarios leads to substantial performance improvements relative to existing online discovery methods. **3)** We

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

present an efficient way of organizing the state of possible discovered networks based on personalized Pagerank. Our approach achieves substantial reductions in training and convergence time.

2 Network Actor Critic (NAC) Framework

We start with the assumption that a network contains a target subnetwork representing a set of relevant vertices. The decision making agent is initially given partial information about the network $G_0 = (N_0, E_0)$. A subset of those vertices have their relevance status C_0 revealed as well, with 0 representing non-target vertices and 1 representing target vertices. We assume our exploration starts from a seed vertex belonging to the partial target subgraph. At each step, the agent can choose from a set of vertices that are observed, but whose label is unknown. We refer to this set of vertices as the boundary set $\mathcal{B}$. After selecting a vertex, the agent can gain knowledge of the vertex label, as well as the identity of all its neighbors. An immediate reward is given if the selected vertex belongs to the target subnetwork.

This problem can be stated as a Markov Decision Process (MDP). An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$: The **state space** $\mathcal{S} = \{s_t\}$ is the set of intermediate discovered networks. We consider a transformation of the network adjacency matrix given by personalized Pagerank (PPR) [10]. We use the PPR ranking to reorder the rows of the observed adjacency matrix and ultimately achieve a more efficient representation of the state space. We further truncate this adjacency matrix for additional efficiency gains and only retain the adjacency matrix defined by the top k vertices. k is a parameter we select and it defines the supporting network for computing potential discovery trajectories and long-term reward. The **action space** $\mathcal{A} = \{A_t\}$, where $A_t = \{a\}$ is the set of boundary vertices at step t. The **transition model**, $T(s, a, s') = P(s'|s, a)$ encodes how the network state changes by specifying the probability of state s transitioning to s' given action a. We do not model this transition function explicitly and take the model-free approach, where we iteratively define and approximate reward without having to directly specify the network state transition probabilities. The local **reward function**, $R(s_t, a_t)$ returns the reward gained by executing action a in state s and is defined as: $R(s_t, a_t) = 1$ if $C(a_t) = 1$. Total cumulative, action-specific reward is defined as the Q function: $Q(s, a) = [\sum_{t=0}^{h} \gamma^t R_{t+1} | s, a]$ with γ representing a discount factor that captures the utility of exploring future network states.

Offline Learning: In our setting, learning happens offline over a training set of possible discovery paths. We use simulated instances of both background networks and target subnetworks to generate paths or trajectories τ_h over the network state space. Each path τ_h represents an alternating sequence of discovered network, action $\langle s_0, a_0, s_1, a_1, \ldots, a_h, s_h \rangle$, taken over h steps. Since in this setting we have access to the ground truth vertex labels, we can map each discovery path to the corresponding cumulative reward value Q. Given tuples $\langle x_i = (s_i, a_j), y_i = Q_i \rangle$, one of NAC's learning objectives is to approximate Q by minimizing the loss function $L(\phi)$, $L^{VF}(\phi) = ||y_i - Q_\phi(x_i)||_2^2$. The approximated Q_ϕ function can then be utilized to estimate the policy function π_θ, which defines the action probability distribution at each state. We utilize a proximal policy optimization (PPO) method [12] in order to compute this function.

The NAC algorithm is updated differently during offline training versus online evaluation. During offline training, the ADAM optimizer [14] is used to compute learning rates for parameters ϕ and θ. For online updates, we use a fixed learning rate and a time horizon of $h = 1$.

3 Experiments

We evaluate our algorithm against several learning scenarios for both synthetic and realistic datasets. Next we describe our datasets and baselines used for comparison.

Synthetic Datasets: We approach the synthetic graph generation by individually modeling a background network (i.e., the network that does not contain any of the target nodes), and the foreground network (i.e., the network that only contains the target nodes and the interactions among them). We use two models to generate samples of background networks. *Stochastic Block model* (SBM) [15] is a common generative graph model that allows us to model community structure as dense subgraphs sparsely connected with the rest of the network. *Lancichinetti–Fortunato–Radicchi* (LFR) model [11] is another frequently used generative model that in contrast to SBM, allows us to simulate network samples with skewed degree distributions and skewed community sizes, therefore able to capture more

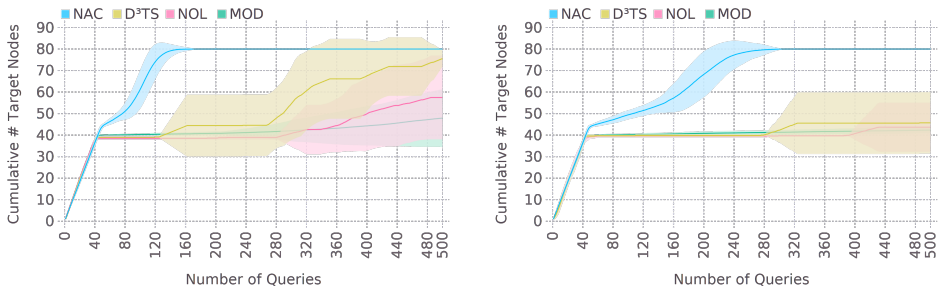
realistic and complex properties of real networks. Finally, we use the *Erdős-Renyi* (ER) model [16] to simulate the foreground network. ER is a simple generative model where vertices are connected with equal probability p_f controlling the density of the foreground network. To create a background plus foreground network sample, we select a subset of the nodes from the background network that will represent the identity of the target nodes. We then simulate an ER subnetwork on these nodes and replace their background induced subnetwork with the ER subnetwork. We reference this process in the rest of the paper as *embedding* the foreground subnetwork.

Real Datasets: We analyzed two Facebook datasets [17] representing pages of different categories as nodes and mutual likes as edges. For both cases, we study the discovery of a target set of vertices, where we control how we generate and embed them in the background network. In particular, we embed a synthetic foreground subnetwork consisting of a denser (anomalous) ER graph with size $n_f = 80$ and density $p_f = 0.003$ and background with a $p_f = 0.002$. We also consider the Livejournal dataset [4]. This dataset represents an online social network with users representing the nodes, and their self-declared friendships the edges. For each user, there is also information on the groups they have joined. Similarly to [4], we use one of the listed groups as the target class. The Livejournal dataset represents a departure from the two Facebook datasets, both in terms of its much larger size, but also because the target class does not represent an anomaly.

Baselines: We evaluate the NAC algorithm by comparing performance with two top performing online network discovery approaches. The *Network Online Learning* (NOL) [3] algorithm learns an online regression function that maximizes discovery of previously unobserved nodes for a given number of queries. We modify the objective of NOL to match our problem setting by requiring the discovery of previously unobserved nodes of a particular type. A second baseline we consider is the *Directed Diversity Dynamic Thompson Sampling* (D^3TS) approach [4]. D^3TS is a stochastic multi-armed bandit approach that leverages different node classifiers and Thompson sampling to diversify the selection of a boundary node. Finally, we compare to a simple fixed node selection heuristic referenced in [4] called *Maximum Observed Degree* (MOD). At every decision step, MOD selects the node with the highest number of observed neighbors that have the desired label.

3.1 Learning Scenarios

In the first learning scenario, the goal is to detect a set of distributed anomalous vertices. They are represented by two cliques, each containing 40 vertices, that are embedded 2 to 3 hops away from each other. The training instances are networks generated by the SBM model, while the test cases are network instances generated by the LFR model. In this scenario, the discovery agent has to figure out 1) how to value longer exploration paths over the cost of including nodes not in target set, and 2) how to adjust to topological differences between training and testing instances.



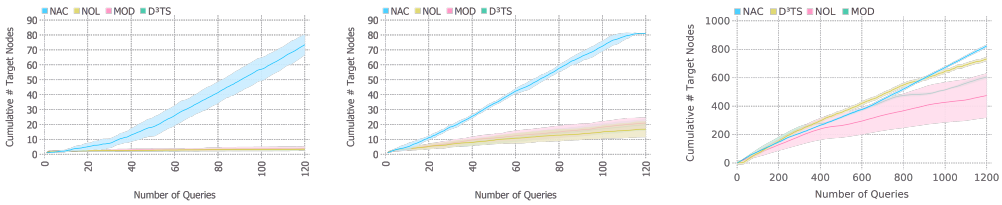
(a) Easier target detectability

(b) Harder target detectability

Figure 1: NAC discovers two anomalous cliques that are not adjacent.

In Figure 1(a), we consider a test case where detactability of the two cliques with complete network information is relatively easy (average background density where the cliques are embedded is comparatively low). We observe that all the methods are able to find the first clique, yet all the baselines struggle once they enter the region where no clique nodes are present. The baselines eventually find some clique nodes, but even then, they are unable to fully retrieve the second clique. NAC is able to leverage estimation of long-term reward and access to the offline policy to fully

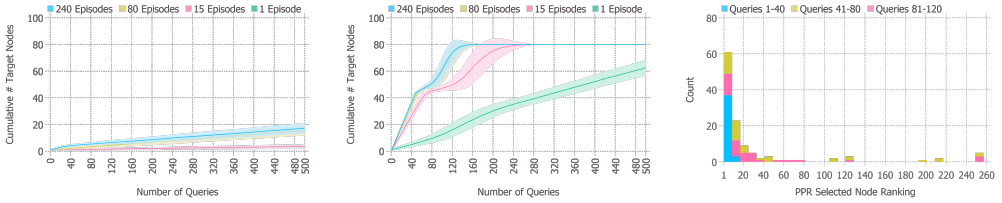
recover both cliques, and furthermore, is able to generalize to the more complex LFR topology. In Figure 1(b), we consider a much harder case: embedding two disjoint dense subgraphs, each with density 0.2 in a background of density 0.05. These parameters are close to the detactability bound [13] for the complete network case. In this case, neither of the baselines learns how to recover the second clique. NAC goes through a longer exploration phase, but eventually learns how to grow the network to identify the second clique. In Figure 2(a) and (b), we illustrate how our model trained on synthetic background networks generalizes to realistic background topologies. For this scenario, we trained with instances from both the LFR and SBM models. We observe that NAC generalizes very well to the Facebook network topologies and is able to fully discover the target nodes. In our



(a) Facebook Politician (b) Facebook TV Shows (c) Livejournal

Figure 2: NAC outperforms competitive online methods on real network topologies.

last learning scenario (Figure 2(c)), we illustrate how our model generalizes to a test case where both the background network and the target set from real world data. Our model has only seen target class examples represented by a dense ER model, yet is able to discover an online Livejournal group with 1400 users. We note the initial exploration cost, as NAC learns to adapt to the new target topology. Eventually by query 850, is able to more efficiently discover the group members and by query 1400 fully recovers the whole group. In Figure 3(a)(b), we demonstrate how re-ordering the adjacency matrix of the observed network by the PPR score supports a faster model convergence during training time. We illustrate by analyzing the convergence behavior on the test case described in Figure 1(a), but the behavior is consistent on all the test cases considered. Finally, in Figure 3(c), we illustrate, that NAC has learned strategies beyond picking a vertex with high PPR score. In particular, NAC has learned how to explore regions where delayed reward is critical (in this example, the region between the two disjoint cliques).



(a) Without PPR (b) With PPR (c) NAC vs PPR

Figure 3: NAC convergence on a test set, without and with PPR ranking (a,b). NAC queries do not always agree with highly ranked nodes (c).

4 Conclusions

We introduced NAC, a deep RL framework for task-driven discovery of incomplete networks. NAC learns offline models of reward and network discovery policies based on a synthetically generated training set. NAC is able to learn effective strategies for the task of selective harvesting, especially for learning scenarios where the target class is relatively small and difficult to discriminate. We show that NAC strategies transfer well to unseen and more complex network topologies including real networks.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis: Human-level control through deep reinforcement learning. Nature (2015).

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Ioannis, Aja Huang, Arthur Guez, Arthur, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, Demis Hassabis: Mastering the game of Go without human knowledge. Nature, (2017).

[3] Timothy LaRock, Timothy Sakharov, Saheli Bhadra, Tina Eliassi-Rad: Reducing network incompleteness through online learning: a feasibility study. The 14th International Workshop on Mining and Learning with Graphs (2018).

[4] Fabricio Murai, Diogo Rennó, Bruno Ribeiro, Gisele L. Pappa, Donald F. Towsley, Krista Gile: Selective harvesting over networks. Data Mining and Knowledge Discovery, Volume 32, Issue 1, pp 187–217 (2017).

[5] Harshavardhan Kamarthi, Priyesh Vijayan, Bryan Wilder, Balaraman Ravindran, Milind Tambe: Learning policies for Social network discovery with Reinforcement learning, CoRR, abs/1907.11625 (2019).

[6] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, Jianfeng Gao: M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search, NeurIPS (2018).

[7] Xiao Lin, Pero Subasic, Hongfeng Yin: Rel4KC: A Reinforcement Learning Agent for Knowledge Graph Completion and Validation. Workshop on Deep Reinforcement Learning for Knowledge Discovery (2019).

[8] Jiaxuan You, Bowen Liu, Rex Ying Vijay Pande, Jure Leskovec: Graph convolutional policy network for goal-directed molecular graph generation. Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp 6412–6422 (2018).

[9] Nicola De Cao, Thomas Kipf: MolGAN: An implicit generative model for small molecular graphs, CoRR, abs/1805.11973 (2018).

[10] Taher H. Haveliwala: Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. IEEE Transactions Knowledge Data Eng. 15(4), pp 784–796 (2003).

[11] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi: Benchmark graphs for testing community detection algorithms. Physical Review E, Volume 78 (2008).

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347 (2017).

[13] Raj Rao Nadakuditi, M. E. J. Newman: Graph spectra and the detectability of community structure in networks. CoRR, abs/1205.1813 (2012).

[14] Diederik P. Kingma, Jimmy Ba: Adam: a method for stochastic optimization. CoRR, abs/1412.6980 (2014).

[15] Paul W. Holland, Kathryn Blackmond Laskey, Samuel Leinhardt: Stochastic blockmodels: first steps. Social Networks 5 (2), 109–137 (1983).

[16] Paul Erdös, Alfréd Rényi: On random graphs. Publicationes Mathematicae, Volume 6, 290–297 (1959).

[17] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and, Charles A. Sutton: GEMSEC: Graph Embedding with Self Clustering.CoRR, abs/1802.03997 (2018).