Active Learning on Graphs

Aneesh Shetty (170040022) [1] **Arnab Jana (170100082)** [1] **Riya Baviskar (170050011)** [1]

Abstract

In many settings, we might not have entire graph link structure available to us due to a variety of reasons, the primary issue being privacy. In such cases, it might be computationally expensive to retrieve such hidden edges. In this project, our task is to perform Link Prediction on graphs by querying particular links for their status, to retrieve the graph structure as quickly as possible, as an Active Learning problem on the Graph. Active learning aims to select the most informative unlabeled instances and request to an oracle for their labels to retrain a learning algorithm.

The code is available at https://github.com/Arnabjana1999/Active_Learning.git

The dataset can be downloaded from https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz

1. Problem Description

We are given a graph $G = (V, E)$. The $V \times V$ pairs set is partitioned into 4 sets, E_{public}, P_{pool}, and NE_{private} and P_{test}. Hidden from us is the status of each pair in P_{pool}, which could either be E_{private} or NE_{private} (this status is static). Thus, $E = E_{\text{public}} \cup E_{\text{private}} \cup E_{\text{test}}$ and $V \times V \backslash E = NE_{\text{public}} \cup NE_{\text{private}} \cup NE_{\text{test}}$.

Link prediction is an important task with a wide variety of applications, the most common being recommendation. In standard LP, the task is to propose to each node a ranked list of nodes that are currently non-neighbors, as the most likely candidates for future linkage. However, in case of graph with private node-pairs, very few edges are visible to provide any information.

Nevertheless, we can ask for status of some node-pair in P_{pool} but that comes with a cost. We have a fixed budget B and the cost of querying for the status of a node-pair is 1 unit. Our task is to use this budget effectively to query node-pairs, so as to learn the model with best predictive performance. We also want to see whether this extra information can improve predictive performance on the overall graph, thus helping us retrieve the graph structure.

[1]Department of CSE, Indian Institute of Technology, Bombay. Correspondence to: 170040022, 170100082, 170050011 <@iitb.ac.in>.

2. Previous Work

The pool-based approach is the most studied setting in active learning. The most important task of active learning methods is to perform good choices in the instance selection process, which in our case is node-pairs. A common approach is to use evaluation measures to estimate the example's utility and select the one with maximal utility values. Examples of some utility metrics are uncertainty sampling, query by committee and expected model change (Souza et al., 2017). However most of these methods work well in node classification setting.

In this work, we are interested in an active-learning method which works in ranking based Link Prediction setting. We present an instance selection mechanism which minimizes an expected ranking loss and hence is more aligned with our objective task.

3. Inference

Each node-pair in the training graph (with both training and private node-pairs) has an associated binary label. The label is 1 if there is an edge between the node-pair and 0 otherwise. Let

$$M^* = \left(\mu_1^*, \ldots, \mu_{|\mathcal{E}|}^*\right) \in \mathcal{M}$$

be the ground-truth labels of the node-pairs in the training graph. However, since all μ_i^* are not available due to some node-pairs being in the pool-set, there can be many possible label assignments for M^*, given the current data.

For simplicity, let us assume that initially all node-pairs belonging to the pool-set have an equal probability of being an edge or a non-edge. Let n be the number of node-pairs in the pool-set. Thus $P(M|D)$ is an uniform distribution over all 2^n possible label assignments. As more and more labels get discovered, some contradicting label assignments which had non-zero probability, now has probability 0. The

updated probability distribution, given feedback data, now again becomes an uniform distribution over its new support.

We want to measure the difference between label assignments using a loss function $\mathcal{L} : \mathcal{M} \times \mathcal{M} \to \Re$. To find good label estimates, we want to find an $M = (\mu_1, \ldots, \mu_{|\mathcal{E}|}) \in \mathcal{M}$ such that $\mathcal{L}(M, M^*)$ is small. However, noting that M^* is unknown, we want to find such a label assignment that minimizes the expected loss w.r.t M^* , namely $P(M|D)$:

$$\underset{M}{\operatorname{argmin}} E_{M^* \sim P(M|\mathcal{D})} [\mathcal{L}(M, M^*)]$$

4. Loss function

We have considered M to be the output of a Neural Network, which outputs a real relevance score for a node-pair. The above formulation then reduces to the following:-

$$\underset{\theta}{\operatorname{argmin}} [\mathcal{L}(NN_\theta(x_1, ..., x_\mathcal{E}), M^*)]$$

where $x_1, ..., x_\mathcal{E}$ are the features of the node-pairs in training set. Suppose the loss function $\mathcal{L}$ can be decomposed over node-pairs in $|E|$. Due to linearity of expectation, we can then decompose the loss into a form easier to work with:

$$\mathcal{L}(M, M^*) = \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \sum_{k=j+1}^{|\mathcal{V}|} E_{M^* \sim P(M|\mathcal{D})} \mathcal{L}^{\text{pair}}(M, M^*, E_{ij}, E_{ik})$$

For this problem, we propose a quadratic hinge-loss function. With uniform distribution of labels, the above summation can be broken down into the summation of 3 different kinds of set of node-pairs (E_{ij}, E_{jk})

If labels for both E_{ij} and E_{ik} are known,

$$E_{M^* \sim P(M|\mathcal{D})} \mathcal{L}^{\text{pair}}(M, M^*, E_{ij}, E_{ik}) = Hinge(1 + M(E_{ik}) - M(E_{ij})) * \mathbb{1}[M^*(E_{ij}) = 1 \wedge M^*(E_{ik}) = 0]$$

If labels for E_{ij} is unknown but E_{ik} is known,

$$E_{M^* \sim P(M|\mathcal{D})} \mathcal{L}^{\text{pair}}(M, M^*, E_{ij}, E_{ik}) = 0.5 * Hinge(1 + M(E_{ij}) - M(E_{ij})) * \mathbb{1}[M^*(E_{ik}) = 1] + 0.5 * Hinge(1 + M(E_{ik}) - M(E_{ij})) * \mathbb{1}[M^*(E_{ik}) = 0]$$

If labels for both E_{ij} and E_{ik} are unknown,

$$E_{M^* \sim P(M|\mathcal{D})} \mathcal{L}^{\text{pair}}(M, M^*, E_{ij}, E_{ik}) = Hinge(1 + M(E_{ik}) - M(E_{ij}))/4$$

In this project, we use the above loss function to train our model. Instead of explicitly computing the exact loss as mentioned above, which involves $O(|V|^3)$ computations, we have used sampling to make the computation tractable.

5. Oracle Query

5.1. Random Query

Select budget number of random node-pairs from the pool-set and query the oracle for their labels. This method, although naive, is expected to improve the model, due to potential feedback about labels of certain node-pairs.

5.2. Largest Expected Loss Pair

Select the pair of node-pair (i,j) which has the largest expected loss contribution. We present this node-pair (or the top few node-pairs with largest expected loss contribution) to the oracle to obtain its label.

$$E_{M^* \sim P(M^*|\mathcal{D})} \mathcal{L}^{\text{pair}}(M, M^*, E_{ij}) = \sum_{k=1, k \neq j}^{|\mathcal{V}|} E_{M^* \sim P(M^*|\mathcal{D})} (1 + M(E_{ik}) - M(E_{ij})) * \mathbb{1}_{misordered}$$

Hence our objective is to find:

$$\underset{i \in V, j \in V, i \neq j}{\operatorname{argmax}} E_{M^* \sim P(M^*|\mathcal{D})} \mathcal{L}^{\text{pair}}(M_{app}, M^*, E_{ij})$$

By obtaining the labels and adding them to training data, the feedback given will reduce the uncertainty in the relative ranking of the node-pairs. In the long run, this process should drive the expected loss contribution of the node-pairs down.

6. Sampling and Split

The graph is divided into train, test and private graphs. Private nodes are sampled from each cluster through random walk. Then for each node, the edges are randomly divided into train, test and private set depending on the split ratio.

Since we are using ranking loss, computing pairwise pairwise hinge-loss for all good/bad pairs is computationally expensive. Also, for most graphs, number of edges $<<$ number of non-edges. Hence we include only a fraction of all non-edges during training. This can be achieved by performing negative sampling of non-edges on a per-node basis from the training graph. This makes the gradient descent computationally tractable.

7. Neural Network Architecture

We have used GraphSAGE, which is a framework for inductive representation learning on large graphs. There is a simple 2 stage SAGEConv layer architecture (with a RELU map in between), which is used to create the node embeddings from the given raw features of the input Graph Nodes. The Node Embedding is created from the original graph using inductive Convolution Layers, and then a Dot Product

Layer is used to compute the Embedding for each edge in the Original Graph and the Negative Graph.

$$Embedding(u) = GraphSAGE(graph, u)$$
$$Embedding(v) = GraphSAGE(graph, v)$$
$$score(E_{uv}) = Embedding(u)^T.Embedding(v)$$

8. Evaluation

We have already discussed before about splitting the graph into train and test sets. We sample certain nodes with degree $>= 10$ before split. We report Mean Average Precision (MAP) across those selected nodes in the test-set and also report the MAP across the private-nodes. We have chosen higher degree nodes because, there is a larger number of hidden node-pairs adjacent to these nodes and these nodes have atleast 2 adjacent test-edges. Thus using this setup, we would be able to see significant change in the MAP with oracle queries. Both the MAP on private-nodes and overall MAP is expected to improve with the feedback obtained after successive queries to the oracle.

9. Dataset

We have used Cora dataset for this task. The dataset can be downloaded from the link given in the abstract. The Cora dataset consists of 2708 scientific publications, which act as nodes and the citation network consists of 5429 links. The total number of keywords for a node is 1433, which act as node features. The average degree of a node is 3.89.

10. Results

We have used an embedding dimension of 50 for node embeddings. Total no. of private node-pairs is 13866, which includes 10% of the edges in the original graph. For the experiments, we have used a budget of 200.

During the training process, we have repeatedly picked a batch of 40 node-pairs from the Pool set for querying the oracle and retrained our model based on the feedback. Table 1 compares the MAP values obtained using Random queries and the queries obtained by LEL Pair method.

Budget	Random query	LEL Pair
0	0.143	0.135
40	0.194	0.191
80	0.206	0.220
120	0.218	0.238
160	0.225	0.246
200	0.230	0.252

Table 1. MAP for Random query vs LEL Pair for CORA dataset.

Figure 1. Comparison of Random Query and LEL Pair

We can see that LEL Pair marginally outperforms Random query method for all Budget sizes. Table 2 shows the increase in MAP values across the selected nodes as well the private nodes with increase in the Budget size.

Budget	MAP on private nodes	MAP on selected nodes
0	0.101	0.135
40	0.104	0.191
80	0.122	0.220
120	0.140	0.238
160	0.132	0.246
200	0.145	0.252

Table 2. MAP for LEL Pair for CORA dataset.

Figure 3 shows the expected training loss vs Iteration number. Since, we are using sampling inside the loss function, to reduce the computational complexity, the loss function isn't strictly convex. Hence the loss is not monotonically decreasing. However in the long run, the expected loss decreases over iterations.

Also we can see flat regions at iteration numbers which are multiples of 10. These are the places where feedback is added to the training loss and our model gradually learns to adapts to this additional data.

11. Conclusion and Future Work

Thus, we analyse one method of application of Active Learning based on Largest Expected Loss Pair (which was initially applied in document ranking setting) on modelling of Graph Structure Retrieval for Hidden node pairs.
Further methods that can be used for this setting can range

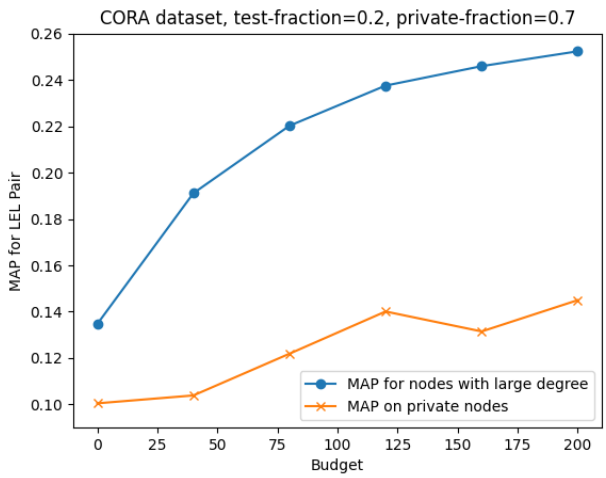

Figure 2. MAP on test set for LEL Pair

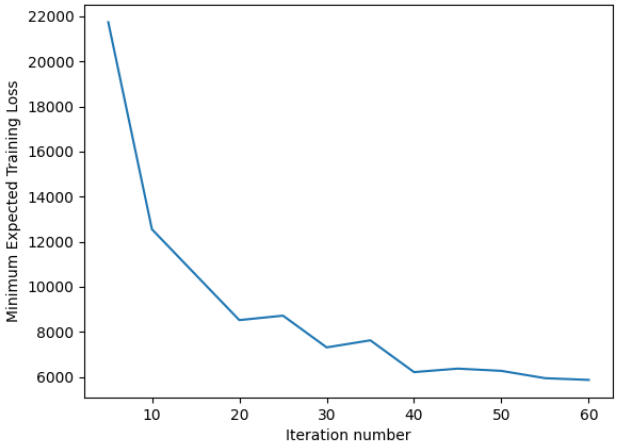

Figure 3. Expected Training Loss vs iteration number

from entropy minimization on the edge prediction probabilities on the pool set (In the sense that for most node pairs in the pool, we are highly confident about our predictions after we have taken the labels for a small subset of them). Other possibility is training ensemble of models for predicting private link in pool node pairs, and querying node pair with maximum disagreement in the predicted label. A different direction would be to formulate the problem as a variance minimization problem on the pool labelling probabilities.

References

De, A., Ganguly, N., and Chakrabarti, S. Discriminative link prediction using local links, node features and community structure. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, dec 2013. doi: 10.1109/icdm.2013.68. URL https://doi.org/10.1109%2Ficdm.2013.68.

Radlinski, F. and Joachims, T. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pp. 570–579, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936097. doi: 10.1145/1281192.1281254. URL https://doi.org/10.1145/1281192.1281254.

Souza, V. M., Rossi, R. G., Batista, G. E., and Rezende, S. O. Unsupervised active learning techniques for labeling training sets: An experimental evaluation on sequential data. *Intelligent Data Analysis*, 21(5):1061–1095, oct 2017. doi: 10.3233/ida-163075. URL https://doi.org/10.3233%2Fida-163075.