Bike-sharing Systems Network Analysis

M.Sc. Siyu Chen

***Abstract*—In bike-sharing systems (BSS) tasks, how to determine the weight threshold matrix and develop a predictive model for link prediction are two main challenges. In task 1, the minimum absolute spectral similarity (MASS) technique for establishing the weight threshold matrix is used to limit the number of connections and generate the resulting network while keeping the network features. In task 2, we use Neural Network (NN) based classifier to develop a model to predict the directed binary BSS network. The weighted sampler is applied to improve pre-processed dataset to fix the class imbalance between edge and non-edge classes. Finally, the optimal weighted edge to non-edge ratio is determined by analyzing the models' performance.**

I. Introduction

Over the past decade, bike-sharing systems (BSS) have been growing in number and popularity in cities across the world. However, it is impossible to analyze too dense graphs based on network science. Furthermore, we need to infer the new interactions among its members in the near future.

Our tasks are based on the Divvy bike system in Chicago. In Task 1, the main goal is to transform the weighted network to a resulting binary network with an appropriate weight threshold matrix. Task 2 aims to develop a predictive model for the resulting binary network. This problem is a fundamental task in graph data analytic, termed as *link prediction* [1]. Link prediction is a common machine learning task applied to graphs: training a model to learn between pairs of nodes in a graph, where relationships exist.

In task 1 of this BSS problem, we first establish the original trip network based on the trip raw data in July 2016. The technique, termed *minimum absolute spectral similarity (MASS)*, is introduced to determine the weight threshold matrix by estimating the variation of graph properties when edges are removed. The original weighted network can be transformed to the resulting binary network compared with the threshold matrix. Finally, we use *Gephi* to visualize and analyze the network. For link prediction in task 2, we design a link prediction framework based on Neural Network (NN) classifier. The weighted sampler for NN based classifier is designed to address the imbalance data for edge and non-edge classes. The optimal weights can be determined by comparing the performance of two classes with different weights in the experiments.

II. Methods

A. Task 1

The main problem in task 1 is to generate a weight threshold matrix to reduce the number of links while maintaining the organization of the network. Inspired by [2], we use a minimum absolute spectral similarity (MASS) for finding a threshold matrix on complex networks.

1) Preliminaries: Consider a weighted and directed graph $G = (V, E)$ composed of N vertices and M edges. Edges have positive weights, and the topology is described by the symmetric weight matrix W, where the generic element $w_{uv} = w_{vu} > 0$ if there is a weighted edge between nodes u and v, while $w_{uv} = w_{vu} = 0$ otherwise. The resulting graph $\tilde{G}$ has a threshold weight matrix X, whose generic element $x_{uv} = x_{vu} = w_{uv}$ if $w_{uv} \geqslant \theta$, and $x_{uv} = x_{vu} = 0$ otherwise. The graph $\tilde{G}$ is therefore a subgraph of G with the same number of nodes. The Laplacian of G is defined as $L = D - W$, where D is the diagonal matrix of the weighted degrees, with entries $d_u = \sum_v w_{uv}$. The Laplacian of $\tilde{G}$ is defined as $\tilde{L} = \tilde{D} - \tilde{W}$.

2) Minimum absolute spectral similarity (MASS): The ratio $\sigma(x)$ between the value of the measure on the resulting graph and the corresponding value on the original graph is used to calculate global efficiency. The term "absolute spectral similarity" refers to the degree of spectral similarity between two graphs, which is defined as

$$\sigma(x) = 1 - \frac{x^T \Delta L x}{x^T \lambda_N x} \tag{1}$$

where λ_N is the largest eigenvalue of the original graph Laplacian, and $\Delta L = \Delta D - \Delta W$ is the graph Laplacian of the difference graph ΔG, while the edges are the ones removed by the chosen sparsification procedure. From the requirement of BSS tasks, we can obtain $|x| = 1$.

The worst-case scenario is taken into account, and the minimum absolute spectral similarity (MASS) technique is developed:

$$\sigma_{\min} = \min_{|x|=1} \left(1 - \frac{x^T \Delta L x}{\lambda_N}\right) = \frac{\lambda_N - \lambda_N^{\Delta}}{\lambda_N} \tag{2}$$

where λ_N^{Δ} is the largest eigenvalue of the difference Laplacian ΔL.

The key of MASS allowing to achieve the threshold matrix comes from the intuition that disconnecting a few peripheral vertices, while a bigger change than removing redundant edges, should have a small impact on the organization of the system, which can be present by the value σ. The pseudocode for MASS technique is shown in Algorithm 1.

B. Task 2

Establishing a predictive model for the binary network is the main challenge in Task 2, which can be regarded as link prediction. In this case, the link prediction is converted to a classification problem in machine learning. Specifically, we adopt Neural Networks (NN) for classification to fit a model.

Technical University of Munich, Department of Electrical and Computer Engineering, Arcisstr. 21, 80333 Munich, Germany.

Algorithm 1: pseudocode for MASS

```
Generate original weighted matrix W from data file;
Initialization for threshold matrix X = W and sample period p;
t = 0 : p : 1
for i = 1 to 1/p do
    Calculate the value of weight x in t(i) * 100-th percentile;
    X(W < x) = 0;
    Calculate degree matrices D_0 and D of W and X;
    L = D_0 − W and L̃ = D − X;
    ΔλN = L − L̃;
    Calculate minimum eigenvalues λ_N^Δ and λ_N of ΔL and L;
    Calculate σ(i) according to Eq. (2);
end for
```

1) Dataset: We consider a resulting binary graph $\tilde{G} = (\tilde{V}, \tilde{E})$ as a dataset, in with data point is directed relationship between each pair of vertices. Given two vertices $\tilde{v}, \tilde{u} \in \tilde{V}$ representing a pair of vertices from graph $\tilde{G}$ with node attributes $a \in \mathbb{R}^d$, their directed relationship can be taken as a data point. At this data point, we take the node attributes of $\tilde{v}$ and $\tilde{u}$ as the feature vector x_i of this data point and take the existence of an edge as the label y_i of this data point, where $x_i \in \mathbb{R}^{2d}$ and $y_i \in \{-1, 1\}$. We describe the dataset as $\mathcal{D} = \{\mathrm{X}, \mathrm{Y}\}$, where $\mathrm{X} = \{x_1, ..., x_n\} \in \mathbb{R}^{2d \times n}$, $\mathrm{Y} = \{y_1, ..., y_n\} \in \mathbb{R}^n$, $i = 1, ..., n$ and n is the number of data points.

2) Sampler: A sampler is a container that sample data points from the dataset to batches. A routine sampler samples data points in a standard manner. However, if several targets are sparse in the dataset, a routine sampler will not be able to assist the algorithm in training a model that performs well on all targets. This problem is termed class imbalance in classification datasets [3]. In this task, we design a weighted sampler to sample elements from the dataset with given ratio. We use a weighted sampler to generate batches, which contains more balanced training data for all classes, to mitigate this problem.

3) Neural Network based classifier: In this task, we apply classification approach in machine learning, neural network (NN), where fully-connected layers and a loss function are designed in this framework to obtain a model for processing the data points. Fully-connected layers are a set of dependent nonlinear functions, which transform the input vector the input vector through weight matrix

$$\vec{y}_{li} = f(\mathrm{W} \cdot \vec{x}_{li} + \vec{b}), \tag{3}$$

where $\vec{x}_{li} \in \mathbb{R}^n$ is the input of the l-th layer, $\vec{y}_{li} \in \mathbb{R}^m$ is the output of this layer, $\vec{b}$ is a bias term, $\mathrm{W} \in \mathbb{R}^{m \times n}$ and $f(\cdot)$ denotes a nonlinear function. By concatenating layers, the final output of a data point x_i is $\hat{y}_i \in [0, 1]$, which could be seen as two classes at two ends. The structure of the NN-based classifier is shown as in Fig. 1.

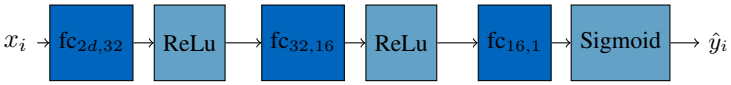

Fig. 1: Structure of the $l = 3$ fully-connected layers. $x_i \in \mathbb{R}^{2d}$ and $\hat{y}_i \in \mathbb{R}^1$.

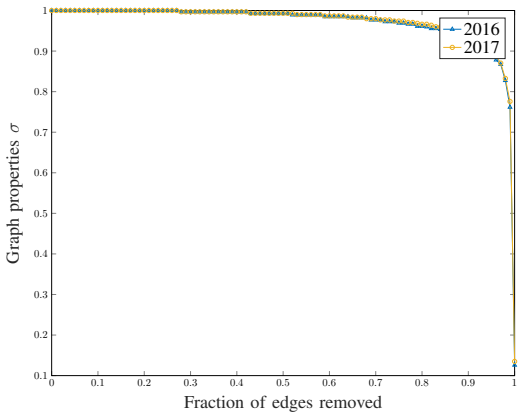

Fig. 2: Variation of MASS

A loss function is required in back-propagation of fully-connected layers in the training stage, when the matrix W needs updated by feedback from prediction deviation. We apply the cross entropy (CE) loss in the training algorithm, which is commonly used for binary classification [4].

III. Main Results

We provide our code to reproduce our experiments at Github: https://github.com/chensiyuagent/BSS.

A. Task 1

1) Establish original network and resulting network based on MASS: First, based on the data files *July trip raw data 2016.csv* and *July trip raw data 2017.csv*, we establish the original graph G_1 and G_2, respectively. In order to decide the threshold matrix X_1 and X_2, the MASS technique is applied to illustrate the change of the graph properties σ under the threshold matrix X. The decision for the weight threshold matrix is made by removing a fraction of selected edges. As is shown in Fig. 2, the networks G_1 and G_2 remain well connected (good graph property σ) even after we remove 30% of edges. Thus, the edges with weight in the 70-th percentile are selected and remain in the networks to decide the weight threshold matrix X_1 and X_2. Finally, by comparing G_1 and G_2 with weight threshold matrix X_1 and X_2 and transforming them to binary networks, the resulting networks $\tilde{G}_1$ and $\tilde{G}_2$ can be obtained.

2) Visualization and network statistics: Gephi is adopted to visualize and analyse the network statistics. In visualization for $\tilde{G}_1$ and $\tilde{G}_2$ as files 2016.*gephi* and 2017.*gephi* at Github, we set the nodes' variance of appearance according to the ranking of degree and Force Atlas as their layout. The network statistics for $\tilde{G}_1$ and $\tilde{G}_2$ are shown in Table I.

TABLE I: Resulting networks statistics

Network statistics	2016	2017
Average Degree	73.4	75.949
Network Diameter	10	9
Graph Density	0.135	0.139
Modularity	0.222	0.268
Average cluster coefficient	0.47	0.469

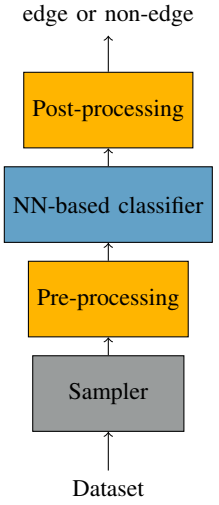

Fig. 3: The link prediction framework. The dataset $\mathcal{D} = \{\mathrm{X}, \mathrm{Y}\}$ is used as the framework's input. The balanced training dataset for edge and non-edge classes can be provided by sampler. In post-processing, the output of the NN based classifier is mapped into $0, 1$ by thresholding, and then a final link prediction is generated (edge or non-edge).

B. Task 2

1) Dataset: After removing redundant edges, the dataset can be obtained files in 2016 and 2017, containing $n = 338k$ training data points from 2016 dataset and $n = 342k$ validation data points from 2017 dataset, respectively. In *Divvy Station 2016* and *Divvy Station 2017*, the node attributes $a \in \mathbb{R}^3$ "latitude", "longitude" and "dpcapacity" from the each source node $\tilde{v}$ and the target node $\tilde{u}$ are chosen as a feature vector $x_i \in \mathbb{R}^6$. From the resulting graph $\tilde{G}_1$ and $\tilde{G}_2$, we define the connection between two nodes as the labels of data points with two classes: "edge" class $y_i = 1$ and "non-edge" class $y_i = 0$. In the training dataset, only 10 percent of samples belongs to the edge class, which brings a great challenge for the built classifier to accurately predict links in the near future. For this reason, accuracy, precision and recall for edge and non-edge predictions are also reported as metrics to indicate the performance of models properly [5].

2) Link prediction: The framework of link prediction is shown as Fig. 3. The NN-based classifier takes the CE loss as its loss function and training data from weighted sampler to mitigate the class imbalance in this dataset. Since the output of the classifier is $\hat{y}_i \in [0, 1]$, we adopt a constant threshold in a post-processing step to threshold the final prediction to $\{0, 1\}$ on the test dataset.

3) Evaluation: We evaluate the impact of two samplers in Table II. For a routine sampler to load the data, the algorithm trains a model with 0.89 accuracy for all predictions. However, scrutiny on edge precision and recall reveals that the model cannot distinguish the edges or non-edges and it makes all non-edge predictions since the class imbalance in the dataset causes this poor performance. With the help of the weighted sampler in training, the algorithm trained a better model in link prediction. The result indicates that edge prediction's precision and recall value increases significantly with a weighted sampler, while the accuracy decrease by only 10 per cent.

TABLE II: Evaluation based on two samplers

sampler	accuracy	e-precision	e-recall
routine sampler	0.89	0	0
weighted sampler	0.69	0.21	0.84

TABLE III: Evaluation based on different class weights from weighted sampler

edge: non-edge	accuracy	e-precision	e-recall	n-precision	n-recall
0.8: 0.2	0.80	0.50	0.05	0.90	0.99
0.9: 0.1	0.69	0.39	0.26	0.92	0.95
0.95: 0.05	0.56	0.28	0.65	0.95	0.79
0.99: 0.01	0.30	0.16	0.93	0.98	0.44

e-precision: precision for edge, e-recall: recall for edge, n-precision: precision for non-edge, and n-recall: precision for non-edge

Furthermore, we investigate which weight ratio of the "edge" class to the "non-edge" class can achieve good performance in this model. In Table III, models that are trained by four sets of edge-to-non-edge ratio demonstrate their prediction ability by the five metrics, where the edge-to-non-edge ratio as 0.9 to 0.1 has the most balanced five metrics, keeping all pair-wise relationships' accuracy and having good predictions for edges.

The precision p and recall r are computed over various thresholds, and the tuple (r, p) is treated as a point on a precision-recall curve. For both edge and non-edge predictions, the precision and recall are negative-correlated, which achieves numerical expectation from definition [6]. From Fig. 4, we choose 0.6 for thresholding to guarantee the balance between precision and recall in post-processing.

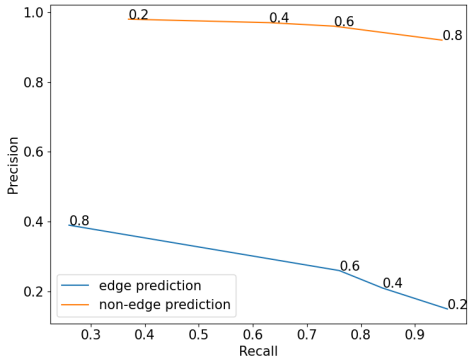

Fig. 4: The precision-recall curve of edge and non-edge classes with different thresholds.

IV. Conclusion

We determine the weight threshold matrix in this task by removing 30% of links using the MASS technique while maintaining good graph properties. In addition, a link prediction framework based on NN classifier is designed to fit a model to predict the network. A balanced training data set for edge and non-edge classes is obtained using a weighted sampler, which addresses the class imbalance and improves the model's performance. Finally, experiments provide an optimal weight ratio of edge to non-edge classes.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American society for information science and technology, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] X. Yan, L. Jeub, A. Flammini, F. Radicchi, and S. Fortunato, "Weight thresholding on complex networks," 06 2018.

[3] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intelligent data analysis, vol. 6, no. 5, pp. 429–449, 2002.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proceedings of the IEEE international conference on computer vision, pp. 2980–2988, 2017.

[5] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 2, pp. 539–550, 2008.

[6] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in Proceedings of the 23rd international conference on Machine learning, pp. 233–240, 2006.