Técnicas de amostragem

Prof: Douglas O. Cardoso

Alunos:

- Karina Brandão
- Michel Ferreira
- Rodrigo Hamacher

1. Ponto de partida

Antes de aplicarmos técnicas para a amostragem de redes complexas, primeiramente deve-se ter em mente os objetivos da amostragem, e como mensurar se de fato aquela amostragem representa minimamente as características originais da rede em questão. No artigo de Leskovec e Faloutsos[2], algumas questões sobre a amostragem de grafos são levantadas, tais como:

- O que é um bom método de amostragem?
- Como a amostragem deve ser realizada? Amostrando nós, arestas, ou outra estratégias?
- Qual tamanho é uma boa amostra (nós/arestas)?
- Como devem ser avaliados os parâmetros obtidos da amostragem?
- Quais características/features de redes complexas devem ser avaliadas para se considerar que uma amostragem satisfatória foi realizada?

De forma resumida, diversos algoritmos em datasets reais foram avaliados e medidas de redes propostas em [2]. Das conclusões obtidas pelos autores, métodos baseados em caminhos aleatórios mostraram resultados melhores que outras estratégias no caso de redes "estáticas". Os estudos foram realizados considerando grafos direcionados. Para a demonstração das técnicas de amostragem, será utilizado o pacote *little ball of fur* [1].

2. Artigo Selecionado

Para a aplicação das técnicas e reprodução de experimentos, foi escolhido o artigo de Rezvanian e Meybodi denominado "Sampling social networks using shortest paths" [3]. Neste artigo, um método de amostragem baseado no conceito de caminho mínimo é utilizado para a amostragem de redes complexas, com foco em redes sociais.

No artigo selecionado, as seguintes técnicas são exploradas e comparadas com a nova técnica:

- Random Edge Sampling (RES)
- Random Node Sampling (RNS)
- Random Walking Sampling (RWS)
- Metropolis-Hasting Random Walk Sampling (MHRW)
- Distributed Learning Automata Based Sampling (DLAS)

3. Descrição do Algoritmo SSP

Como descrito no artigoo algoritmo proposto SSP utiliza o conceito de caminhos mínimos para a amostragem da rede. Considerando um grafo $G(V, E)$ composto por um conjunto $V = \{v_1, v_2, \ldots, v_n\}$ de nós e $E = \{e_1, e_2, \ldots, e_m\}$ de arestas, o algoritmo SSP calcula o caminho mínimo π entre l nós não adjacentes. Em uma iteração i, dois nós não adjacentes v_s (nó de origem) e v_d (nó de destino) são escolhidos aleatoriamente, e o menor caminho π_i é calculado para tal par de nós. Após as l rodadas, é realizado um ranqueamento dos nós mais acessadas durante o procedimento, e a amostragem é baseada utilizando uma porcentagem dos nós mais visitadas durante o procedimento de busca pelos caminhos mais curtos.

O pseudocódigo, como descrito no artigo [3], é mostrado abaixo. A implementação dessa técnica de amostragem é encontrada em [1]. A quantidade de nós a se considerar na amostragem é passada como argumento para a instanciação da classe do Sampler no pacote Little Ball of Fur [4].

Experimentos de outros pesquisadores mostraram a importância dos caminhos mais curtos em redes de colaboração científica, onde o caminho mais curto entre dois cientistas possui em média 64% dos casos um colaborador/cientista considerado como *top-ranked*. Outras situações mostradas no artigo evidenciam que mensagens entre redes sociais se propagam mais rapidamente e verifica-se o conceito de caminho mais curto.

De forma geral, o algoritmo SSP é de simples entendimento, pois se baseia em conceitos básicos da teoria de grafos, como a busca pelo caminho mais curto. Porém, algumas perguntas e hipóteses são levantadas e investigadas no artigo, como por exemplo:

- Qual a porcentagem de caminhos mínimos necessária a ser explorada para que seja atendida uma taxa de amostragem específica?
- Qual o melhor mecanismo para a escolha dos nós iniciais e finais do algoritmo? Devemos escolher os nós baseado em alguma característica específica do mesmo (centralidade, grau, etc)?
- Deve existir um tamanho de caminho mínimo entre os nós para que tal caminho seja considerado pelo processo de amostragem?
- Caso um valor mínimo para a distância entre os nós de início e fim seja estabelecido, qual o impacto desse valor na capacidade de amostragem da rede?

4. Reprodução dos experimentos

No artigo, cinco experimentos foram conduzidos para avaliar a performance do algoritmos proposto, utilizando tanto conjuntos de dados reais quanto redes sintéticas. O algoritmo SSP foi comparado com os algoritmos descritos anteriormente. De forma sucinta, os experimentos são descritos abaixo

- Experimento 1: O primeiro experimento consiste em verificar qual a porcentagem dos caminhos mínimos deviam ser percorridos para que a quantidade mínima de nós

definida pela taxa de amostragem fosse atingida. Como esse experimento demanda investigar e utilizar o código do algoritmo de certa forma, não foi explorado. </br>

- Experimento 2: O segundo experimento verifica como a distorção entre as distribuições de graus e caminhos mínimos são reproduzidas a partir das redes amostradas. Somente a distribuição dos nós foi executada nesse notebook.</br>

- Experimento 3: O experimento realiza a verificação da escolha dos nós iniciais para a verificação do caminho mínimo do algoritmo do SSP. Questões como a escolha aleatória ou direcionada dos nós, tamanho mínimo de caminho para aceitação e outras medidas de redes para auxiliar na escolha dos nós é discutida e analisada.

- Experimento 4: Qual o impacto na distribuição das características do grafo quando definimos um tamanho específico de

- Experimento 5: Qual o impacto da taxa de amostragem nos parâmetros das redes amostradas?

Os seguintes conjuntos de dados foram utilizados para a comparação entre os diversos métodos de amostragem:

Network	Nós	Arestas
Karate	34	78
Football	115	613
Email	1133	5451
Jazz	1198	2742
Cit-HepTh	27770	352807
Ego-Facebook	4039	88234
Wiki-Vote	7115	103689
ER-10000	10000	19990294
WS-10000	10000	123942
BA-10000	10000	99945

Para as redes simuladas citadas anteriomente, os seguintes parâmetros foram utilizados:

- ER-10000 (Erdos-Rényi) - N = 10.000 e p = 0.2
- WS-10000 (Watts-Strogatz) - N = 10.000, k = 4 e p = 0.2
- BA-10000 (Barabási-Albert) - N = 10.000, $m_0 = m = 5$

Infelizmente, nem todas as redes conseguimos executar por motivos de leitura dos arquivos e/outros que não solucionamos.

5 Simulações dos experimentos

In [ ]:

```
# Pacotes utilizados
import numpy as np
import pandas as pd
import networkx as nx
import scipy.stats as stats
```

```
import plotly.express as px
import littleballoffur as lbf
```

In [ ]:

```
# Networks
karate     = nx.karate_club_graph() # Karate Club dataset
jazz       = nx.read_edgelist("../datasets/out.arenas-jazz", create_using = n
wiki_vote  = nx.read_adjlist('../datasets/wiki-vote.txt')
email      = nx.read_edgelist('../datasets/email-univ.edges')
facebook   = nx.read_adjlist('../datasets/face.edges')
cit_hepth  = nx.read_adjlist('../datasets/cit-hepth.txt')

# Leitura do dataset football
football_file = open('../datasets/football.gml', 'r').read()
football  = nx.parse_gml(football_file.split('\n'))
```

In [ ]:

```
# Lista de redes
dataset_list  = [karate, jazz,
                 #wiki_vote,
                 email,
                 facebook,
                 #cit_hepth,
                 football]

dataset_names = ['karate', 'jazz',
                 #'wiki',
                 'email',
                 'facebook',
                 #'cit_hepth',
                 'football']
```

Experimento I

O primeiro experimento aqui reproduzido corresponde ao experimento II do artigo, onde o objetivo é verificar a performance do SSP em relação aos outros métodos de amostragem descritos anteriormente. Para duas taxas de amostragem fixadas, 10% e 20%, é realizado um teste KS para a verificação da discrepância entre a distribuição cumulativa do grau dos nós para a rede original e a rede amostrada. Para distribuições discrepantes ou não similares, temos um valor de KS próximo de 1, e próximo a 0 quanto mais similares forem.

In [ ]:

```
def run_ks_degree_distribution_experiment(sampling_rate: float,
                                          sampling_technique,
                                          trials: int = 50) -> dict:

    results_sampling = dict()

    # Loop para a realização dos cálculos de divergência KS
    for dataset, network in zip(dataset_names, dataset_list):

        # Lista com resultados das divergências KS
        ks_results = list()

        # Distribuição da probabilidade dos graus graus de uma determinada re
        degree_distribution_graph = np.cumsum(nx.degree_histogram(network)/np

        # Como descrito no artigo, os valores são resultados dos experimentos
        # com 50 execuções
        for trial in range(trials):
```

```
            # Amostragem das redes utilizando SSP
            sampler = sampling_technique(int(sampling_rate * len(network.node
            network = nx.convert_node_labels_to_integers(network) # Para func
            sampled_network = sampler.sample(network)

            # Distribuição dos graus para a rede amostrada
            degree_distribution_sampled_graph = np.cumsum(nx.degree_histogram
                                                np.sum(nx.degree_histogram(sa

            # Comparação entre as duas distribuições
            ks_value = stats.kstest(degree_distribution_graph,
                                    degree_distribution_sampled_graph)

            ks_results.append(ks_value.statistic)


        # Armazenamento do resultado para as redes
        results_sampling[dataset] = np.mean(ks_results)

    return results_sampling
```

In [ ]:
```
# lista de técnicas utilizadas:
samplers = [lbf.RandomEdgeSampler,
            lbf.RandomNodeSampler,
            lbf.RandomWalkSampler,
            lbf.MetropolisHastingsRandomWalkSampler,
            lbf.ShortestPathSampler]

# Nomes das técnicas
samplers_name = ['RES', 'RNS', 'RWS', 'MHRW', 'SSP']

# Armazenamento dos resultados e laço
results_from_sampler_10 = dict()
results_from_sampler_20 = dict()

for sampler_name, sampler in zip(samplers_name, samplers):
    results_from_sampler_10[sampler_name] = run_ks_degree_distribution_experi
    results_from_sampler_20[sampler_name] = run_ks_degree_distribution_experi
```

In [ ]:
```
# Valores das médias para os datasets disponíveis e porcentagem de amostragem
pd.DataFrame.from_dict(results_from_sampler_10)
```

Out[ ]:

	RES	RNS	RWS	MHRW	SSP
karate	0.421111	0.734444	0.487778	0.497778	0.444444
jazz	0.306089	0.342098	0.411382	0.411377	0.370353
email	0.238500	0.299960	0.384778	0.378147	0.379717
facebook	0.924780	0.974748	0.987970	0.987729	0.982041
football	0.561795	0.820000	0.556093	0.560712	0.616385

In [ ]:
```
# Valores das médias para os datasets disponíveis e porcentagem de amostragem
pd.DataFrame.from_dict(results_from_sampler_20)
```

Out[ ]:

	RES	RNS	RWS	MHRW	SSP
karate	0.321556	0.539000	0.533222	0.547667	0.494778

	RES	RNS	RWS	MHRW	SSP
jazz	0.232046	0.347206	0.383334	0.378322	0.369638
email	0.192889	0.274086	0.320416	0.331379	0.328767
facebook	0.818346	0.811190	0.800949	0.823913	0.882744
football	0.571513	0.736604	0.550969	0.558085	0.589632

O esperado, de acordo com os dados do artigo, eram valores de KS em SSP menores que os das outras redes. O que foi visto experimentalmente é que não há melhora da representação da distribuição de graus das redex complexas, pois a distorção obtidas por outras redes é menor em todos os casos, para as duas taxas de amostragens testadas.

Experimento II

O Experimento II (no artigo, experimento V) se dedica a entender o impacto da taxa de amostragem da rede na métrica *Normalized Root Mean Square Error*. A métrica NMSE é definida como $NMSE = \frac{\sqrt{(E|\theta - \theta_s|)^2}}{\theta}$, onde θ e θ_s são os valores dos parâmetros originais da rede, e das redes amostradas. No caso, a comparação é realizada utilizando o coeficiente de clusterização de cada rede. Espera-se que, com o aumento da taxa de amostragem, tenhamos redes mais fiéis às originais, e tal métrica diminua com o aumento da amostragem.

In [ ]:
```
def run_nmse_experiment(network,
                        sampling_rate: float,
                        trials: int = 50) -> list:

    # Coeficiente de clusterização
    clustering_average = nx.average_clustering(network)
    clustering_values = list()

    for trial in range(trials):
        sampler = lbf.ShortestPathSampler(number_of_nodes=int(len(network.nod
        network = nx.convert_node_labels_to_integers(network)
        sampled_graph = sampler.sample(network)
        clustering_values.append(nx.average_clustering(sampled_graph))

    # Cálculo nmse
    nmse = np.sqrt(np.linalg.norm(clustering_average - np.array(clustering_va

    return nmse
```

In [ ]:
```
# Execução para os datasets
nmse_from_datasets = dict()

for network, network_name in zip(dataset_list, dataset_names):
    nmse_list = list()
    for sampling_rate in range(5, 100, 5):
         nmse_list.append(run_nmse_experiment(network, sampling_rate/100))

    nmse_from_datasets[network_name] = nmse_list
```

In [ ]:
```
# Gráfico
nmse_df = pd.DataFrame.from_dict(nmse_from_datasets)
```

```
nmse_df.index = list(range(5, 100, 5))

px.line(nmse_df,
        title='NMSE x Sampling Rate para diferentes redes',
        labels={'value' : 'NMSE', 'index' : 'Sampling Rate', 'variable' : 'ne
```

Como esperado, com o aumento da taxa de amostragem da rede (ou seja, da quantidade de nós considerados no SSP), mais semelhante à rede original a rede amostrada será, diminuindo assim a distorção entre os parâmetros calculados nas mesmas (no caso acima, o índice médio de clusterização do grafo). Um fato que é destacado no artigo e visto também na prática é o NMSE possuindo um altíssimo valor para redes grandes, mostrando a dificuldade da reprodutibilidade dos parâmetros para uma taxa de amostragem baixa.

Referências

[1] Little Ball of Fur: A Python Library for Graph Sampling - Link </br> [2] Sampling from Large Graphs - Link</br> [3] Sampling social networks using shortest paths - Link</br> [4] Implementação do Algoritmo SSP - Link