Kompleksne mreže – primjeri zadataka za ZI

iz 2. ciklusa

Napomene:

Završni ispit nosi 80 bodova. Ovaj ogledni primjer nosi više bodova, jer je dano više zadataka za vježbu nego što se očekuje u završnom ispitu.

Studente se savjetuje da samostalno pokušaju riješiti sve praktične zadatke.

1. Simulirajte zajednice pomoću networkX na proizvoljnom grafu. Detektirajte zajednice algoritmom. Objasnite.
2. Iterativnom metodom izračunajte modularity score za n zajednica od 2 do 6 na karate klub

grafu. Objasnite kako odabrati koji je broj zajednica? Zašto?

1. Simulirajte Beyesian learning na grafu koristeći jednostavni Bala Goyal 98 pristup s dva A ili B stanja. Koristite barbell graf koji će biti povezan putem centralnog čvora. Detektirajte kojem stanju će konvergirati mreža. Objasnite koncept.
2. Simulirajte DeGroot širenje mišljena na random grafu od 10 ljudi. Izračunajte kada će sustav konvergirati. Objasnite koncept DeGroot učenja.
3. Učitajte karate klub graf. Primjenite k-clique community algoritam s proizvoljno odabranim parametrima. Konceptualno objasnite dobivene rezultate s obzirom na odabrane parametre.

[File podsjetnik\_plus\_auditorne.pdf]

# 1. Simulirajte zajednice pomoću networkX na proizvoljnom grafu. Detektirajte zajednice algoritmom. Objasnite.

import networkx as nx

import matplotlib.pyplot as plt

# Učitavanje grafa članova karate kluba

G\_karate = nx.karate\_club\_graph()

# Detekcija zajednica pomoću Girvan-Newman algoritma

communities = list(nx.community.girvan\_newman(G\_karate))

# Ispis dobivenih zajednica

print(communities)

# Kreiranje proizvoljnog grafa

G = nx.karate\_club\_graph()

# Detekcija zajednica algoritmom

communities = list(nx.algorithms.community.greedy\_modularity\_communities(G))

# 2. Iterativnom metodom izračunajte modularity score za n zajednica od 2 do 6 na karate klub grafu.

# Objasnite kako odabrati koji je broj zajednica? Zašto?

from networkx.algorithms.community.quality import modularity

# Iterativno računanje modularity score za n zajednica od 2 do 6

for n in range(2, 7):

# Detekcija zajednica

communities = list(nx.community.greedy\_modularity\_communities(G\_karate, weight=None))

# Računanje modularity score

mod\_score = modularity(G\_karate, communities)

print(f"Modularity score for {n} communities: {mod\_score}")

Odabir broja zajednica trebao bi biti rezultat balansa između maksimiziranja modularity scorea i izbjegavanja prekomjerne podjele grafa.

Sources: p.20, p.52, p.40, p.16

# 3. Simulirajte Beyesian learning na grafu koristeći jednostavni Bala Goyal 98 pristup s dva A ili B stanja.

# Koristite barbell graf koji će biti povezan putem centralnog čvora. Detektirajte kojem stanju će konvergirati mreža.

# Objasnite koncept.

import networkx as nx

import random

import numpy as np

# Create a Barbell Graph

G = nx.barbell\_graph(5, 0)

# Add random state "A" or "B" to each node

random.seed(123) # Set random seed for reproducibility

for node in G.nodes():

if random.random() < 0.5:

G.nodes[node]["state"] = "A"

G.nodes[node]["value"] = 1

else:

G.nodes[node]["state"] = "B"

G.nodes[node]["value"] = 2 if random.random() < 0.55 else 0

# Simulate Bala Goyal 98 approach

num\_iterations = 20

p\_val = 0.55

for \_ in range(num\_iterations):

new\_states = {}

new\_values = {}

for node in G.nodes():

A\_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if G.nodes[neighbor]["state"] == "A"]

B\_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if G.nodes[neighbor]["state"] == "B"]

A\_average = sum(A\_neighbors) / len(A\_neighbors) if A\_neighbors else 0

B\_average = sum(B\_neighbors) / len(B\_neighbors) if B\_neighbors else 0

if A\_average > B\_average:

new\_states[node] = "A"

new\_values[node] = 1

elif B\_average > A\_average:

new\_states[node] = "B"

new\_values[node] = 2 if random.random() < p\_val else 0

for node in new\_states:

G.nodes[node]["state"] = new\_states[node]

G.nodes[node]["value"] = new\_values[node]

# Check the final state and value of each node

for node in G.nodes():

print(f'{node}: {G.nodes[node]["state"]}, {G.nodes[node]["value"]}')

This code creates a Barbell Graph, adds random states "A" or "B" to each node, simulates the Bala Goyal 98 approach for a specified number of iterations, and finally prints the final state and value of each node. The concept here is to observe the convergence of the network to a stable state (A or B) based on the interaction of nodes and their neighbors

# 4. Simulirajte DeGroot širenje mišljenja na random grafu od 10 ljudi. Izračunajte kada će sustav konvergirati.

# Objasnite koncept DeGroot učenja.

import networkx as nx

import numpy as np

# Create a random graph with 10 nodes

G = nx.erdos\_renyi\_graph(10, 0.3, directed=True)

# Initialize opinions randomly between 0 and 1 for each node

opinions = {node: np.random.rand() for node in G.nodes}

# Define the number of iterations for opinion convergence

max\_iterations = 100

# DeGroot opinion dynamics simulation

for \_ in range(max\_iterations):

new\_opinions = {}

for node in G.nodes:

neighbors = list(G.predecessors(node))

neighbor\_opinions = [opinions[neighbor] for neighbor in neighbors]

new\_opinions[node] = np.mean(neighbor\_opinions)

opinions = new\_opinions

# Check for convergence

if all(abs(opinions[node] - np.mean(list(opinions.values()))) < 0.01 for node in G.nodes):

print("Opinions have converged after", \_+1, "iterations.")

break

This code uses the NetworkX library to create a random directed graph with 10 nodes and then simulates the DeGroot opinion dynamics. It initializes random opinions for each node and iteratively updates the opinions based on the average opinions of their neighbors. The code also checks for convergence by comparing the nodes' opinions to the average opinion of all nodes.

The concept of DeGroot learning is a model for opinion dynamics where individuals update their opinions based on the opinions of their neighbors in a network. Over time, the opinions of individuals in the network converge to a consensus opinion. The DeGroot model is used to study how opinions, beliefs, and information spread through social networks.

# 5. Učitajte karate klub graf. Primjenite k-clique community algoritam s proizvoljno odabranim parametrima.

# Konceptualno objasnite dobivene rezultate s obzirom na odabrane parametre.

import networkx as nx

# Load the karate club graph

G\_karate = nx.karate\_club\_graph()

# Apply the k-clique community algorithm

communities = list(nx.algorithms.community.kclique.k\_clique\_communities(G\_karate, k=3, cliques=None))

# Explain the obtained results with respect to the chosen parameters

In this code, the k parameter in the k\_clique\_communities function represents the size of the cliques to be used to define the communities. You can adjust the value of k to observe how it affects the resulting communities. Additionally, the cliques parameter can be used to provide a list of cliques if already computed, but it's set to None in the example.

After obtaining communities, you can analyze the results based on the chosen k value and the structure of the graph to understand how the k-clique community algorithm partitioned the karate club graph into communities.

Sources: p.20, p.68, p.16, p.41

[File sve\_preze.pdf]

Teoretska pitanja:

1. Objasnite u par rečenica DeGroot model učenja. ✅
2. Objasnite što je to mudrost masa. ✅
3. Definirajte pojam ekvilibrija i pojasnite. ✅
4. Objasnite razliku između strateških komplemenata i strateških supstituta. ✅
5. Definirajte što je društvena mreža. ✅
6. Definirajte strukturu korisničkog profila. Od kojih se elemenata sastoji? ✅

1. DeGroot model učenja je model koji opisuje kako se uvjerenja ljudi mijenjaju kroz komunikaciju sa svojim susjedima u mreži. Ovaj model pretpostavlja da ljudi usvajaju uvjerenja drugih ljudi u svojoj okolini i prilagođavaju svoja uvjerenja kroz komunikaciju.

2. Mudrost masa je koncept koji se odnosi na ideju da grupa ljudi može donijeti bolje odluke nego pojedinac, posebice u situacijama gdje se kombiniraju različite perspektive i znanja.

3. Ekvilibrij je stanje ravnoteže ili stabilnosti u sustavu. U kontekstu kompleksnih mreža, to može značiti da se sustav nalazi u stabilnom stanju gdje nema većih promjena ili da su sile unutar sustava u ravnoteži.

4. Strateški komplemenati su proizvodi ili usluge koje se međusobno nadopunjuju i čije se potražnje povećavaju kada se koristi jedan proizvod ili usluga. Strateški supstituti su proizvodi ili usluge koje se mogu zamijeniti jedan za drugi, tj. kada se koristi jedan proizvod ili usluga, potražnja za drugim opada.

5. Skup entiteta međusobno povezanih odnosima entiteti: ljudi, grupe ljude, organizacije, brandovi

6. Struktura korisničkog profila sastoji se od različitih elemenata kao što su osnovni osobni podaci, poput imena, adrese i kontaktnih informacija, povijest aktivnosti, interesi, preferencije, i bilo koja dodatna informacija koja opisuje korisnika i njegove navike.

Sources: p.221, p.4, p.269, p.240