monte carlo simulation

August 25, 2020

1 All the dependencies

[1]:
```
import time
import numpy as np
import matplotlib.pyplot as plt
from igraph import Graph
from tqdm import tqdm
from itertools import product
import itertools
```

2 Creating Graph

[2]:
```
source = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5]
target = [2, 3, 4, 5, 6, 7, 8, 9, 2, 3, 4, 5, 6, 7, 8, 9, 6, 7, 8, 9]

# create a directed graph
graph = Graph(directed=True)

# add the nodes/vertices (the two are used interchangeably) and edges
# 1. the .add_vertices method adds the number of vertices
# to the graph and igraph uses integer vertex id starting from zero
# 2. to add edges, we call the .add_edges method, where edges
# are specified by a tuple of integers.
graph.add_vertices(10)
graph.add_edges(zip(source,  target))
print('vertices:',  graph.vcount())
print('edges:',  graph.ecount())
```

```
vertices: 10
edges: 20
```

3 Simulation of Independent cascade model using monte carlo simulation

[3]:
```
def independent_cascade_model(graph, seed_nodes, prob, n_iters):
    total_spread = 0

    # simulate the spread process over multiple runs
    for i in range(n_iters):
        np.random.seed(i)
        active = seed_nodes[:]
        new_active = seed_nodes[:]

        # for each newly activated nodes, find its neighbors that becomes␣
↪activated
        while new_active:
            activated_nodes = []
            for node in new_active:
                neighbors = graph.neighbors(node, mode='out')
                success = np.random.uniform(0, 1, len(neighbors)) < prob
                activated_nodes += list(np.extract(success, neighbors))

            # ensure the newly activated nodes doesn't already exist
            # in the final list of activated nodes before adding them
            # to the final list
            new_active = list(set(activated_nodes) - set(active))
            #print(new_active)
            active += new_active

        total_spread += len(active)

    return total_spread / n_iters
```

In the simulation of independent cascade model using monte carlo simulation we have four parameters first one is graph, second one is seed notes, third one is probaility(probability of each node to become active when influenced by its neighbor), and fourth one is number of iteration(how many time we have to simulate this independent cascade to get expected spread). This simulation returns the expected number of nodes those became active during this simulation process. later we can calculate the percentage of nodes who became active.

NOTE: In the independent cascade model initially all the nodes will be inactive except seed nodes and these seed nodes will be treated as newly activated nodes in the first iteration,and these newly activated nodes will influence its neighbor to become active but only few of them become active on the basis of probaility parameter, all the nodes who became active in the current iteration will be newly activated nodes for the next iteration, once a node influnces its neighbor then it can not influence its neighbor again in the next iteration i,e. only newly activated node will particepate in the spreading process in the next iteration. This spreading process will halt when there is no newly activated nodes present in the graph.

4 Experiment to see how probaility of being influenced by its neighbor play role in the process of sprading awareness in the graph

[4]:
```
# assuming we start with [0 1 ] as seed nodes
seed_nodes = [0,1]

x=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
for i in x:
    spread=independent_cascade_model(graph, seed_nodes, prob=i,n_iters=1000)
    print('when probability='+
          str(i)+',spread='+str(spread*100/graph.vcount()))
```

```
when probability=0.1,spread=35.78
when probability=0.2,spread=50.980000000000004
when probability=0.3,spread=64.66999999999999
when probability=0.4,spread=75.13
when probability=0.5,spread=84.16999999999999
when probability=0.6,spread=90.57000000000001
when probability=0.7,spread=95.06
when probability=0.8,spread=98.03
when probability=0.9,spread=99.67999999999999
when probability=1.0,spread=100.0
```

5 Conclusion

We can see the result of experiment which tells that if the probability of being influenced by its neighbor is 1 then all the nodes in the graph will be influenced by the independent cascade model provided there is no disconnected component in the graph. However this doesn't happen in real social network so we can use probability perameter by analyzing the fact how often the neighbors become active when influenced by its neighbor. So we can tune this probability parameter as per our need.