Machine Learning with Graphs
숙제 1 : 그래프 기초, 페이지랭크

1 개요

캠퍼 여러분, 안녕하세요? 'Machine Learning with Graphs' 과목에 오신 것을 환영합니다! 앞서 강의에서 그래프에 대한 다양한 지식을 배우셨을텐데요, 이 과제에서는 1) 그래프를 컴퓨터 언어로 표현하는 방법과, 2) 대표적인 검색 알고리즘인 페이지랭크(PageRank) 알고리즘을 구현해 보겠습니다.

2 그래프 기초(HW 1-1)

그래프(graph)는 정점(node)과 간선(edge)으로 이루어진 수학적인 구조로, 다양한 복잡계(complex systems)를 표현하고 분석할 수 있는 언어입니다. 일반적으로 그래프는 G=(V, E) 로 표현되며 V와 E는 각각 정점 집합과 간선 집합을 의미합니다. 여기서 간선 $e = (v1, v2) \in E$ 는 두 개의 정점 $v1, v2 \in V$ 을 잇는 선으로, 방향성(direction)과 가중치(weight)를 가질 수 있습니다. 본 실습에서는 방향성이 있고 가중치가 없는 (directed-unweighted graph) 그래프를 다룹니다.

그래프는 인접 리스트 방식과 인접 행렬 방식으로 표현할 수 있습니다. 인접 리스트 방식이란 그래프의 한 정점에서 나가는 이웃들과 들어오는 이웃들을 저장하는 방식으로 메모리 사용량을 줄일 수 있습니다. 하지만, 간선을 탐색할 때 모든 이웃들을 검색해야하는 단점이 있습니다. 반대로 인접 행렬 방식은 그래프를 (정점 수) × (정점 수) 크기의 행렬로 나타내며 행렬 i행 j열의 원소에 대해서 정점 i와 j 사이에 간선이 있으면 1, 없으면 0 로 표현합니다. 이 방식은 간선을 탐색할 때 행렬 indexing으로 바로 접근할 수 있지만, (정점 수) × (정점 수)의 메모리를 사용하는 단점이 있습니다. 과제 1-1에서는 위에서 설명한 두 가지의 그래프 표현 방식을 구현합니다. 여러분이 본 실습에서 구현할 함수들은 아래와 같습니다.

1. edges()
   - 그래프의 간선 집합$[e_1, e_2, ..., e_n]$을 반환합니다.
2. add_vertex(v)
   - 그래프에 정점(v)를 추가합니다.

3. remove_vertex(v)
   - 그래프에서 정점(v)과 그 정점이 속한 간선들을 모두 제거합니다. 만약 그래프에 정점이 존재하지 않는 경우, 함수를 종료합니다.

4. out_neighbors(v)
   - 그래프의 정점(v)의 나가는 이웃($out-neighbors$)을 반환합니다.

5. out_degree(v)
   - 그래프의 정점(v)에서 나가는 간선의 수를 반환합니다.

6. has_edge(v_1, v_2)
   - 그래프에 간선(v_1, v_2)이 존재하는 지 검사합니다. 만약 간선이 존재하면 True를 반환하고, 간선이 존재하지 않으면 False를 반환합니다.

2.1 구현

본 실습에선 그래프를 표현하기 위한 다양한 함수들을 구현합니다. 과제1-1에는 추상클래스 DiGraph를 상속받은 Graph_dict, Graph_numpy_array 클래스가 있으며, 각각의 클래스는 딕셔너리(dictionary)와 넘파이 행렬(numpy array)를 이용해서 간선을 저장하는 그래프입니다. 여러분은 각 클래스의 함수 edges(), add_vertex(), remove_vertex(), out_neighbors(), out_degree(), has_edge() 를 구현해야 합니다. 여러분의 학습을 돕기 위해 일부 함수들을 사전에 구현해놓았으며, 이를 참고해서 다른 함수들을 구현하시길 바랍니다.

본 실습에서는 두 개의 정점에 대해서 하나의 간선만 존재할 수 있다고 가정합니다(if $e_1 = (v_1, v_2) \in E$ and $e_2 = (v_1, v_2) \in E$, then $e_1 = e_2$). Graph_dict의 subgraph 함수는 과제 1-2에서 사용할 예정으로 과제 1-1에서는 구현하지 않습니다.

2.2 실행 및 평가

코드 실행과 평가를 위해 test script를 제공합니다. 스크립트에 있는 모든 assertion을 통과하면 성공입니다!

3 페이지랭크(PageRank) (HW 1-2)

웹사이트에는 수많은 페이지가 있고, 사용자는 많은 정보 중에서 소수의 유용한 정보를 찾고 싶어 합니다. 검색엔진은 사용자가 검색한 키워드에 대해서 관련성 있고 신뢰할 수 있는 페이지를 사용자에게 추천합니다. 본 과제에서는 다양한 검색 알고리즘 중 페이지랭크 알고리즘을 구현합니다.

페이지랭크 알고리즘은 그래프의 각 정점의 중요도 값을 찾는 알고리즘입니다. 페이지랭크 값은 아래와 같이 이웃 정점들에게 받은 가중치 합으로 정의합니다.

$$r_i = \sum_{i \in N_{in}(j)} \frac{r_i}{d_{out}(i)}$$

여기서 $N_{in}(j)$는 정점 j의 이웃을, $d_{out}(i)$는 정점 i의 나가는 연결성(out-degree)를 의미합니다. 우리는 각 정점의 페이지랭크 값이 수렴할 때 까지 페이지랭크 알고리즘을 반복합니다.

기본적인 페이지랭크 알고리즘엔 막다른 정점(dead end)과 스파이더 트랩(spider trap)이라는 두 가지 문제점이 있습니다. 수업에서는 (1) 각 막다른 정점에서 모든 정점들로 간선들을 추가한 뒤, (2) 순간이동(teleport)을 도입하는 방법으로 두 문제를 해결하였습니다. 순간이동이란 페이지랭크 점수를 이웃에게 부여할 때 0과 1사이의 감폭 비율(*damping factor*) β라는 변수를 두어서, $1-\beta$의 값을 모든 정점들에게 공평하게 나누어 주는 방법입니다. 순간이동을 추가한 페이지랭크 값은 아래와 같습니다.

$$r_i = \sum_{i \in N_{in}(j)} \left(\beta \frac{r_i}{d_{out}(i)} \right) + (1-\beta)\frac{1}{|V|}$$

하지만, 각 막다른 정점에서 모든 정점들로 간선들을 추가할 경우, 수 많은 간선이 추가되어, 수행 속도가 느려지거나 메모리가 부족해지는 등의 문제가 발생할 수 있습니다. 본 과제에서는 추가 간선 없이 막다른 정점 문제를 해결합니다. 구체적으로, 막다른 정점에서 사라진 페이지 랭크 점수들의 합을 모든 정점들에게 $\frac{1}{|V|}$ 씩 나누어 주는 방법을 통해 막다른 정점 문제를 해결합니다.

Algorithm 1의 의사 코드(pseudocode)는 막다른 정점과 스파이더 트랩 문제를 모두 해결한 페이지랭크 알고리즘입니다. 본 과제는 아래의 의사 코드(pseudocode)를 구현하는 것을 목표로 합니다.

Algorithm 1 PageRank

procedure PageRank(그래프 $G = (V, E)$, 감폭비율 β, 최대 반복 수 T, 임계값 ϵ)
  for each j in V **do**
    $r_j^{old} \leftarrow \frac{1}{|V|}$
  for $n \leftarrow 1, \cdots, T$ **do**
    $S \leftarrow 0$
    for each j in V **do**
      $r_j^{new} \leftarrow 0$
      for each i in $N_{in}(j)$ **do**
        $r_j^{new} \leftarrow r_j^{new} + \beta \frac{r_i^{old}}{d_{out}(i)}$
      $S \leftarrow S + r_j^{new}$
    for each j in V **do**
      $r_j^{new} \leftarrow r_j^{new} + \frac{(1-S)}{|V|}$ ▷ re-insert the leaked PageRank
    if $||r^{new} - r^{old}||_1 < \epsilon$ **then**
      break
    $r^{old} \leftarrow r^{new}$
  return r^{new}

4 구현

본 실습에서는 과제 1-1의 Graph_dict에서 구현하지 않았던 subgraph 함수와 PageRank 알고리즘을 구현합니다. 먼저, subgraph 함수는 부분 정점 집합을 인자로 받아서, 그 정점들과 정점들을 연결하는 간선으로만 이루어진 부분 그래프를 반환하는 함수입니다. 그리고, PageRank 함수는 방향성이 있는 그래프, 감폭 비율(β), 최대 반복 수(maxiter), 임계값 (ϵ)을 입력으로 받고, 각 정점 별로 페이지랭크 값이 저장된 딕셔너리를 반환합니다. 본 과제에서는 파이썬 내장 라이브러리만 사용할 수 있으며 numpy, networkx, snap 등의 **외부 라이브러리 사용을 금지**합니다.

4.1 실행 및 평가

과제 1-1과 마찬가지로 코드 실행과 평가를 위해 test script를 제공합니다. 스크립트에 있는 모든 assertion을 통과하면 성공입니다!