> 课程链接：CS224W: Machine Learning with Graphs
> 课程视频：【课程】斯坦福 CS224W: 图机器学习 (2019 秋 | 英字)

目录

1. SNAP简介

Stanford Network Analysis Platform (SNAP) 是一种用于大型网络分析和操作的通用、高性能系统。

官方文档：http://snap.stanford.edu/snappy/doc/reference/index-ref.html
Github项目地址：https://github.com/snap-stanford/snap-python

SNAP提供超过70个网络数据集，包括：

- **Social networks:** online social networks, edges represent interactions between people
- **Twitter and Memetracker :** Memetracker phrases, links and 467 million Tweets
- **Citation networks:** nodes represent papers, edges represent citations
- **Collaboration networks:** nodes represent scientists, edges represent collaborations (co-authoring a paper)
- **Amazon networks :** nodes represent products and edges link commonly co-purchased products

snap.py的安装：

```
# for Windows
pip install snap-stanford
```

其他系统的安装方式可以看官网的介绍。

2. snap中方法的一些命名惯例

Variable types/names:

- **...Int**: an integer operation, variable: **GetValInt()**
- **...Flt**: a floating point operation, variable; **GetValFlt()**
- **...Str**: a string operation, variable; **GetDateStr()**

Classes vs. Graph Objects:

- **T...:** a class type; **TUNGraph**
- **P...:** type of a graph object; **PUNGraph**

Data Structures:

- **...V:** a vector, variable **TIntV InNIdV**
- **...VV:** a vector of vectors (i.e., a matrix), variable **FltVV**
  TFltVV ... a matrix of floating point elements
- **...H**: a hash table, variable **NodeH**
  TIntStrH ... a hash table with **TInt** keys, **TStr** values
- **...HH**: a hash of hashes, variable **NodeHH**
  TIntIntHH ... a hash table with **TInt** key 1 and **TInt** key 2
- **...Pr:** a pair; type **TIntPr**

- **Get...:** an access method, **GetDeg()**
- **Set...:** a set method, **SetXYLabel()**
- **...I:** an iterator, **NodeI**
- **Id:** an identifier, **GetUId()**
- **NId:** a node identifier, **GetNId()**
- **EId:** an edge identifier, **GetEId()**
- **Nbr:** a neighbor, **GetNbrNId()**
- **Deg:** a node degree, **GetOutDeg()**
- **Src:** a source node, **GetSrcNId()**
- **Dst:** a destination node, **GetDstNId()**

3. 基本教程

教程官网
官方文档——官网的基本教程只是对一些基本操作进行了介绍，具体的一些函数和方法在官方文档中有比较详细的阐述。

3.1 基本类型

在snap中，有三个基本类型：

`TInt` ：Integer

`TFlt` ：Float

`TStr` ：String

```
i = snap.TInt(10)
print(i.Val)
'''
out:
10
'''
```

> Note: do not use an empty string “”in TStr parameters

3.2 向量类型（Vector type）

> Vector——**Sequences** of values of the **same type**
> 新的元素可以增加在队尾，同时支持可以访问和设置向量中元素的值。

在声明vecctor时，通常使用 `T<type_name>V` 。例如 `v = snap.TIntV()` 为声明一个数据类型为 `Int` 的向量。

```
# Create an empty vector创建一个空向量
v = snap.TIntV()
# Add elements 增加元素
v.Add(1)
v.Add(2)
v.Add(3)
v.Add(4)
v.Add(5)
# Print vector size 输出向量的维度
print(v.Len())
'''out:
5
'''
# Get and set element value 可以访问和设置向量中元素的值
print(v[3])
```

```
'''out:
4
'''
v[3] = 2 *v[2]
print(v[3])
'''out:
6
'''
# print vector elements 输出向量的元素
for item in v:
    print(item)
'''out:
1
2
3
6
5
'''
for i in range(0, v.Len()):
    print(i, v[i])
'''out:
0 1
1 2
2 3
3 6
4 5
'''
```

3.3 哈希表类型（Hash Table Type）

> 一系列的键值对 (key, value) pairs
> Keys必须为同一类型，values必须为同一类型，但是key和value可以为不同的数据类型。
> 支持增加新的(key, value）键值对
> 可以通过访问key来获取或者修改相应的value

在声明哈希表类型时，通常会使用 `T<key_type><value_type>H` ，例如 `TIntStrH` 就声明了一个哈希表，key的数据类型为 `Int` ， value的数据类型为 `Str` 。

```
###### Hash Table Type
# Create an empty tatble
h = snap.TIntStrH()
# Add elements
h[5] = "apple"
h[3] = "tomato"
h[9] = "orange"
h[6] = "banana"
h[1] = "apricot"
# print table size
print(h.Len())
'''out:
5
'''
# Get element value
print("h[3]=", h[3])
'''out:
h[3]= tomato
'''
# Set element value
h[3] = "peach"
print("h[3]=", h[3])
'''out:
h[3]= peach
'''
# print table elements
for key in h:
    print(key, h[key])
'''out:
5 apple
3 peach
9 orange
6 banana
1 apricot
'''
```

需要注意的是，在哈希表中，每个Key都有一个KeyId，Key是由用户创建的，KeyId是由程序自己分配的。例如在上面的表中，Key 3的KeyId是1，因为它是第二个键值。

```
# print KeyId
print(h.GetKeyId(3))
'''out:
1
'''
```

3.4 Pair Type

> (value1, value2)
> value1的数据类型可以和value2的不一致。
> 可以访问两个value的值

在声明Pair类型时，通常会使用 `T<type1><type2>Pr` ，例如 `TIntStrPr` 就声明了一个Pair，第一个value的数据类型为 `Int` ，第二个value的数据类型为 `Str` 。

```
###### Pair Type
# create a Pair
p = snap.TIntStrPr(1, "one")
# print pair values
print(p.GetVal1())
'''out:
1
'''
print(p.GetVal2())
'''out:
one
'''
```

Pair类型可以和Hash Table类型以及Vector类型进行组合：

- **TIntStrPrV**: a vector of (integer, string) pairs
- **TIntPrV**: a vector of (integer, integer) pairs
- **TIntPrFltH**: a hash table with (integer, integer) pair keys and float values

3.5 图和网络类（Basic Graph and Network Classes）

`TUNGraph`： 无向图

`TNGraph`： 有向图

TNEANet：在节点和边上具有属性的多重图

Graph classes in SNAP:

- TUNGraph: undirected graphs (single edge between an unordered pair of nodes)
- TNGraph: directed graphs (single directed edge between an ordered pair of nodes)

Network classes in SNAP:

- TNEANet: directed multigraphs (multiple directed edges between an ordered pair of nodes) with attributes for nodes and edges

在snap.py中，如果要使用图或者网络的具体实例，则要相对应地使用 PUNGraph 、PNGraph 和 PNEAnet 。

> Snap.py does not directly use instances of the graph and network classes, but utilizes smart pointers to those instances instead. The actual instances in the Python program are of type PUNGraph, PNGraph, or PNEAnet and correspond to TUNGraph, TNGraph, and TNEAnet, respectively.
> In general, if you need to call a class method, use T and if you need to specify an instance type, use P.

例如：

```
G1 = snap.TNGraph.New()
G2 = snap.GenRndGnm(snap.PNGraph, 100, 1000)
```

图的建立

```
###### Graph Creation
# create directed graph
G1 = snap.TNGraph.New()
# Add nodes before adding edges
G1.AddNode(1)
G1.AddNode(5)
G1.AddNode(12)

G1.AddEdge(1,5)
G1.AddEdge(5,1)
G1.AddEdge(5,12)

# Create undirected graph, directed network
G2 = snap.TUNGraph.New()
N1 = snap.TNEANet.New()
```

遍历图中的节点和边：

```
###### Graph Traversal
# Traverse nodes
for NI in G1.Nodes():
    print("node id %d, out-degree %d, in-degree %d" % (NI.GetId(), NI.G
'''out:
node id 1, out-degree 1, in-degree 1
node id 5, out-degree 2, in-degree 1
node id 12, out-degree 0, in-degree 1
'''
# Traverse edges
for EI in G1.Edges():
    print("(%d, %d)" % (EI.GetSrcNId(), EI.GetDstNId()))
'''out:
(1, 5)
(5, 1)
(5, 12)
'''
# Traverse edges by nodes
for NI in G1.Nodes():
    for DstNId in NI.GetOutEdges():
        print("edge (%d %d)" % (NI.GetId(), DstNId))
'''out:
edge (1 5)
edge (5 1)
edge (5 12)
'''
```

图的存储和加载

```
###### Graph save and loading
# Save text
snap.SaveEdgeList(G1, "test.txt", "List of edges")
# Load text
G2 = snap.LoadEdgeList(snap.PNGraph,"test.txt",0,1)
# Save binary
FOut = snap.TFOut("test.graph")
G2.Save(FOut)
FOut.Flush()
# Load binary
FIn = snap.TFIn("test.graph")
G4 = snap.TNGraph.Load(FIn)
```

可以看到存储的text文件如下：

```
# Directed graph: test.txt
# List of edges
# Nodes: 3 Edges: 3
# FromNodeId	ToNodeId
1	5
5	1
5	12
```