NetworkX tutorial

EECS 4414 Information Networks

Tilemachos Pechlivanoglou

Basic Example

```
>>> import networkx as nx
>>> G = nx.Graph()
>>> G.add_node("spam")
>>> G.add_edge(1,2)
>>> print(G.nodes())
[1, 2, 'spam']
>>> print(G.edges())
[(1, 2)]
```

Installation

install manually from

`http://pypi.python.org/pypi/networkx`

or use built-in python package manager, easy install

```
$ easy_install networkx
```

or use macports

```
$ sudo port install py27-networkx
```

use pip (replacement for easy_install)

```
$ sudo pip install networkx
```

or use debian package manager

```
$ sudo apt-get install python-networkx
```

Graph types

Graph : Undirected simple (allows self loops)

DiGraph : Directed simple (allows self loops)

MultiGraph : Undirected with parallel edges

MultiDiGraph : Directed with parallel edges

can convert to undirected: `g.to_undirected()`

can convert to directed: `g.to_directed()`

Graph types

Graph : Undirected simple (allows self loops)

DiGraph : Directed simple (allows self loops)

MultiGraph : Undirected with parallel edges

MultiDiGraph : Directed with parallel edges

```
>>> g = nx.Graph()
>>> d = nx.DiGraph()
>>> m = nx.MultiGraph()
>>> h = nx.MultiDiGraph()
```

Adding nodes

`add_nodes_from()` takes any iterable collection and any object

```
>>> g = nx.Graph()
>>> g.add_node('a')
>>> g.add_nodes_from( ['b','c','d'])
>>> g.add_nodes_from('xyz')
>>> h = nx.path_graph(5)
>>> g.add_nodes_from(h)
>>> g.nodes()
[0,1,'c','b',4,'d',2,3,5,'x','y','z']
```

Adding edges

Adding an edge between nodes that don't exist will automatically add those nodes

`add_nodes_from()` takes any iterable collection and any type (anything that has a `__iter__()` method)

```
>>> g = nx.Graph( [('a','b'),('b','c'),('c'
    ,'a')] )
>>> g.add_edge('a', 'd')
>>> g.add_edges_from([('d', 'c'), ('d', 'b')
    ])
```

Node attributes

Can add node attributes as optional arguments along with most add methods

```
>>> g = nx.Graph()
>>> g.add_node(1,name='Obrian')
>>> g.add_nodes_from([2],name='Quintana'])
>>> g[1]['name']
'Obrian'
```

Edge attributes

Can add edge attributes as optional arguments along with most add methods

```
>>> g.add_edge(1, 2, w=4.7 )
>>> g.add_edges_from([(3,4),(4,5)], w =3.0)
>>> g.add_edges_from([(1,2,{'val':2.0})])
# adds third value in tuple as 'weight' attr
>>> g.add_weighted_edges_from([(6,7,3.0)])
>>> g.get_edge_data(3,4)
{'w' : 3.0}
>>> g.add_edge(5,6)
>>> g[5][6]
{}
```

Simple properties

Number of nodes :

```
>>> len(g)
>>> g.number_of_nodes()
>>> g.order()
```

Number of Edges

```
>>> g.number_of_edges()
```

Check node membership

```
>>> g.has_node(1)
```

Check edge presence

```
>>> g.has_edge(1)
```

Neighbors

```
>>> G = nx.Graph()
>>> G.add_path([0,1,2,3])
>>> [e for e in G.edges_iter()]
[(0, 1), (1, 2), (2, 3)]
>>> [(n,nbrs) for n,nbrs in G.adjacency_iter
    ()]
[(0, {1: {}}), (1, {0: {}, 2: {}}), (2, {1:
    {}, 3: {}}), (3, {2: {}})]
>>> G[1][2]['new_attr'] = 5
>>> G[1][2]['new_attr']
5
```

Degree

```
 >>> G.degree(0)
1
>>> G.degree([0,1])
{0: 1, 1: 2}
>>> G.degree()
{1: 1, 2: 2, 3: 2, 4: 1}
>>> G.degree().values() # useful for degree
    dist
[1, 2, 2, 1]
```

Simple graph generators

located in `networkx.generators.classic` module

Complete Graph

```
nx.complete_graph(5)
```

Chain

```
nx.path_graph(5)
```

Bipartite

```
nx.complete_bipartite_graph(n1, n2)
```

Random graph generators

located in module `networkx.generators.random_graphs`

Preferential Attachment

```
nx.barabasi_albert_graph(n, m)
```

$G_{n,p}$

```
nx.gnp_random_graph(n,p)
```

```
nx.gnm_random_graph(n, m)
```

```
nx.watts_strogatz_graph(n, k, p}
```

Useful functions

shortest path

```
nx.shortest_path(G,s,t)
nx.betweenness_centrality(G)
```

clustering

```
nx.average_clustering(G)
```

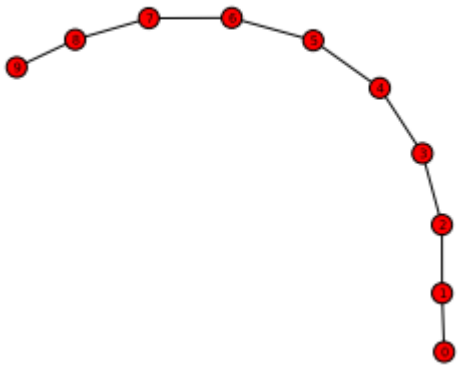
```
>>> G=nx.complete_graph(5)
>>> nx.clustering(G)
{0: 1.0, 1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0}
```

diameter

```
nx.diameter(G)
```

Plotting with matplotlib

```
import networkx as nx
import matplotlib.pyplot as plt
>>> G = nx.path_graph(10)
>>> nx.draw(G)
>>> plt.savefig("path_graph.pdf")
```



Showing plots

Instead of saving the figure, simply show it with:

```
plt.show()
```

Won't work in Linux Subsystem for Windows!

Instead, copy your files to a Windows folder (/mnt/c/Users/…) and run it with python from inside Windows (cmd/PowerShell)

Resources

NetworkX Docs

`http://networkx.lanl.gov/tutorial/index.html`

NetworkX Tutorial

`http://networkx.lanl.gov/contents.html`

Matplotlib Docs

`http://matplotlib.sourceforge.net/contents.html`

Matplotlib Tutorial

`http://matplotlib.sourceforge.net/users/pyplot_tutorial.html`

Numpy Docs

`http://numpy.scipy.org/`

MacPorts

`http://macports.org`

Thanks

Slides based on NetworkX Tutorial by Evan Rosen

YORK
UNIVERSITÉ
UNIVERSITY