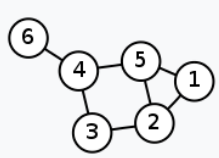
Graph_Theory_Overview

June 19, 2020

1 Graph Theory and Linear Alegbra: The Fun That Never Ends

1.1 The Creation and Application of Laplaican Matricies

The examples seen in this section are based on the Laplacian matrix page found on Wikipedia. To start, note the graph itself, its degree matrix, and its adjacency matrix in the figure below.

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

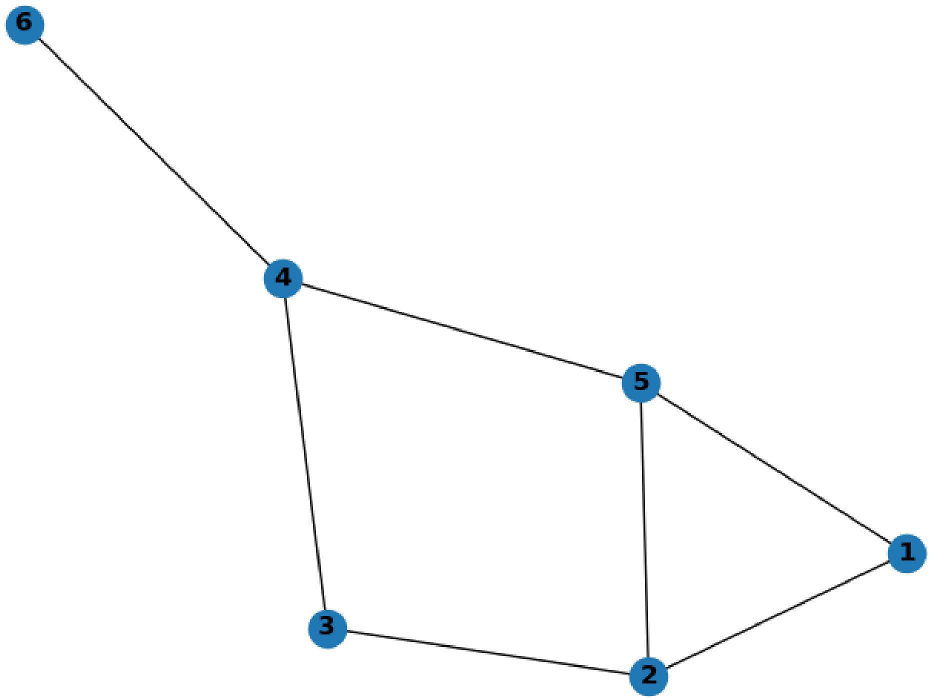
[2]:
```
import networkx            as nx  # Graph creation and analysis.
import numpy               as np  # All things linear algebra.
import matplotlib.pyplot as plt # Drawing graphs.
```

[3]:
```
G = nx.Graph() # Create empty graph

# Adding the six verticies of the Wiki example graph
V = [1, 2, 3, 4, 5, 6]
G.add_nodes_from(V)

# Adding the seven edges from the Wiki example graph
E = [(1,2), (1,5), (2,3), (2,5), (3,4), (4,5), (6,4)]
G.add_edges_from(E)
```

This is the output from the graph that was just created using NetworkX. Note that it contains the same set of vertices and edges that the Wiki examples does.



[4]:
```
# Print current graph info
numOfVerticies = G.number_of_nodes()
numOfEdges = G.number_of_edges()

print("Number of verticies: {verts}".format(verts=numOfVerticies))
print("Number of edges: {edges}\n".format(edges=numOfEdges))
```

```
Number of verticies: 6
Number of edges: 7
```

Consider the following equation:

$$D_{i,j} := \begin{cases} deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

[5]:
```
D = []
for i in G:
    for j in G:
        if(i == j):
            D.append(G.degree(i))
        else:
            D.append(0)

D = np.array(D)
D = D.reshape(numOfVerticies, numOfVerticies)
```

```python
print("Degree Matrix:\n{DMatrix}".format(DMatrix=D))
```

```
Degree Matrix:
[[2 0 0 0 0 0]
 [0 3 0 0 0 0]
 [0 0 2 0 0 0]
 [0 0 0 3 0 0]
 [0 0 0 0 3 0]
 [0 0 0 0 0 1]]
```

$$A_{i,j} := \begin{cases} deg(v_i) & \text{if } (v_i, v_j) \epsilon E \\ 0 & \text{otherwise} \end{cases}$$

[6]:
```python
A = []
for i in G:
    for j in G:
        if(G.has_edge(i,j)):
            A.append(1)
        else:
            A.append(0)

A = np.array(A)
A = A.reshape(numOfVerticies, numOfVerticies)

print("Adjacency Matrix:\n{AMatrix}".format(AMatrix=A))
```

```
Adjacency Matrix:
[[0 1 0 0 1 0]
 [1 0 1 0 1 0]
 [0 1 0 1 0 0]
 [0 0 1 0 1 1]
 [1 1 0 1 0 0]
 [0 0 0 1 0 0]]
```

[7]:
```python
L = D - A

print("Laplacian Matrix:\n{LMatrix}".format(LMatrix=L))
```

```
Laplacian Matrix:
[[ 2 -1  0  0 -1  0]
 [-1  3 -1  0 -1  0]
 [ 0 -1  2 -1  0  0]
 [ 0  0 -1  3 -1 -1]
 [-1 -1  0 -1  3  0]
 [ 0  0  0 -1  0  1]]
```

[8]:
```
eigVals = np.linalg.eigvals(L)
eigVals = set(eigVals)

fiedlerValue = sorted(eigVals)[1] # index 1, which is where the second-smallest␣
↪eigenvalue is located in the set

print("The Fiedler value comes to: {:.3f}".format(fiedlerValue))
```

```
The Fiedler value comes to: 0.722
```