Graph similarity

Laura Zager and George Verghese
EECS, MIT

March 2005
Words you won’t hear today

- impedance matching
- thyristor
- oxide layer
- VARs
Some quick definitions

\[ G(V, E) \]  \hspace{1cm} \text{a graph } G

- \( V \) \hspace{1cm} \text{the set of vertices or nodes}
- \( E \subset V \times V \) \hspace{1cm} \text{the set of edges – can be directed or undirected.}

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

a directed graph and its node-node adjacency matrix
Graph theory: some perspective

The Königsberg bridge problem
(18th c.)

The Four Color Theorem
(1976)
Graph theory: some perspective

The Königsberg bridge problem
(18th c.)

The Four Color Theorem
(1976)

Erdös and Rényi random graph models
(1959)
The Königsberg bridge problem
(18\textsuperscript{th} c.)

The Four Color Theorem
(1976)

Erdős and Rényi random graph models
(1959)

present and future:
graphs that arise in the natural world
Applications

- Comparing biological networks
  - Deriving phylogenetic trees from metabolic pathway data [Heymans, Singh, 2003].

- Social network mapping
  - Small world phenomena [Milgram, 1967; Watts, 1999].

- Web searching
  - Improving searching results using WWW structure [Kleinberg, 1999].

- Chemical structure matching
  - Finding similar structures in a chemical database [Hattori et al., 2003].
Applications

- Comparing biological networks
  - Deriving phylogenetic trees from metabolic pathway data [Heymans, Singh, 2003].

- Social network mapping
  - Small world phenomena [Milgram, 1967; Watts, 1999].

- Web searching
  - Improving searching results using WWW structure [Kleinberg, 1999].

- Chemical structure matching
  - Finding similar structures in a chemical database [Hattori et al., 2003].

---

one common thread: similarity
Notions of similarity

- Isomorphism – identifying a *bijection* between the nodes of two graphs which preserves (directed) adjacency.

Notions of similarity

- Isomorphism – identifying a *bijection* between the nodes of two graphs which preserves (directed) adjacency.

Notions of similarity

- Edit distance – given a cost function on *edit operations* (e.g. addition/deletion of nodes and edges), determine the *minimum cost transformation* from one graph to another.

Notions of similarity

- **edit distance**
- **isomorphism**
  - Maximum common subgraph – identifying the ‘largest’ isomorphic subgraphs of two graphs.
  - Minimum common supergraph – identifying the ‘smallest’ graph that contains both graphs.

Notions of similarity

- Statistical methods – assessing aggregate measures of graph structure (e.g. degree distribution, diameter, betweenness measures).

- Albert, Barabasi, Reviews of Modern Physics, 2002
Notions of similarity

- Iterative methods:
  Two graph elements (e.g., edges or nodes) are similar if their neighborhoods are similar.

- Jeh & Widom, 8th *Intl. Conf. on Knowledge Discovery and Data Mining*, 2002.
- Melnik, Garcia-Molina, 18th *Intl. Conf. on Data Engineering*, 2002.
Motivated by demands of web searching

Step 1: Use text-based search methods to identify a candidate graph containing relevant websites and their neighbors.
Kleinberg, 1999

- Relevant search results might be:
  - Hubs – pages which *point to* many good authorities
  - Authorities – pages which *are pointed to* by many good hubs

- Step 2: Compute hub and authority scores for every node in the candidate graph.
Kleinberg, 1999

- Denote:
  - $x_{1p}(k) = \text{hub score of node } p \text{ at iteration } k$
  - $x_{2p}(k) = \text{authority score of node } p \text{ at iteration } k$

- Update rule:
  
  $$x_{2p}(k + 1) = \sum_{q:(q,p) \in E} x_{1q}(k)$$
  
  i.e. the sum of hub scores of nodes that point to node $p$

  $$x_{1p}(k + 1) = \sum_{q:(p,q) \in E} x_{2q}(k)$$
  
  i.e. the sum of authority scores of nodes that are pointed to by node $p$

- Normalize the scores so that $\sum_p x_{ip} = 1$ and repeat.
Kleinberg, 1999

- Denote:
  - $x_{1p}(k)$ = hub score of node $p$ at iteration $k$
  - $x_{2p}(k)$ = authority score of node $p$ at iteration $k$

- Update rule:

  $$x_{2p}(k + 1) = \sum_{q:(q,p) \in E} x_{1q}(k)$$
  i.e. the sum of hub scores of nodes that point to node $p$

  $$x_{1p}(k + 1) = \sum_{q:(p,q) \in E} x_{2q}(k)$$
  i.e. the sum of authority scores of nodes that are pointed to by node $p$

- Normalize the scores so that $\sum_p x_{ip} = 1$ and repeat.
Kleinberg, 1999

- Denote:
  - $x_{1p}(k)$ = hub score of node $p$ at iteration $k$
  - $x_{2p}(k)$ = authority score of node $p$ at iteration $k$

- Update rule:
  - Stack the scores $x_{1p}(k)$ into a vector $[x_1]_k$, then stack $[x_1]_k$ and $[x_2]_k$.
  - Let $B$ be the node-node adjacency matrix of the candidate graph. Then:

$$
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}_{k+1} =
\begin{bmatrix}
  0 & B \\
  B' & 0
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}_k
$$
Kleinberg, 1999

Blondel, Van Dooren, et al., 2004*  

- Views Kleinberg’s iteration as a comparison between the web graph and a *hub-authority graph*:

```
1
hub node
```

```
2
authority node
```

\[
A = \begin{bmatrix}
0 & 1 \\
0 & 0 \\
\end{bmatrix}
\]

- Observe that the matrix form of Kleinberg’s update can be written as follows:

\[
\begin{bmatrix}
x_1 \\
x_2 \\
\end{bmatrix}_{k+1} = \begin{bmatrix}
0 & B \\
B' & 0 \\
\end{bmatrix}\begin{bmatrix}
x_1 \\
x_2 \\
\end{bmatrix}_k = \left( A \otimes B + A' \otimes B' \right)\begin{bmatrix}
x_1 \\
x_2 \\
\end{bmatrix}_k
\]

- Is this generalizable to any two graphs \( G_A \) and \( G_B \)?

A first step toward generalizing Kleinberg’s approach: consider comparing the graph $G_B$ to the following graph using a similar update:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

1. hub node
2. central node
3. authority node

$$x_{1p}(k + 1) = \sum_{q : (p, q) \in E} x_{2q}(k)$$

$$x_{2p}(k + 1) = \sum_{q : (q, p) \in E} x_{1q}(k) + \sum_{q : (p, q) \in E} x_{3q}(k)$$

$$x_{3p}(k + 1) = \sum_{q : (q, p) \in E} x_{2q}(k)$$
A first step toward generalizing Kleinberg’s approach: consider comparing the graph $G_B$ to the following graph using a similar update:

$$A = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}$$

1. **hub node**
2. **central node**
3. **authority node**

\[
x_{1p}(k+1) = \sum_{q:(p,q)\in E} x_{2q}(k)
\]
\[
x_{2p}(k+1) = \sum_{q:(q,p)\in E} x_{1q}(k) + \sum_{q:(p,q)\in E} x_{3q}(k)
\]
\[
x_{3p}(k+1) = \sum_{q:(q,p)\in E} x_{2q}(k)
\]
A first step toward generalizing Kleinberg’s approach: consider comparing the graph $G_B$ to the following graph using a similar update:

$$\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}$$

$$x_{1p}(k + 1) = \sum_{q:(p,q)\in E} x_{2q}(k)$$

$$x_{2p}(k + 1) = \sum_{q:(q,p)\in E} x_{1q}(k) + \sum_{q:(p,q)\in E} x_{3q}(k)$$

$$x_{3p}(k + 1) = \sum_{q:(q,p)\in E} x_{2q}(k)$$
A first step toward generalizing Kleinberg’s approach: consider comparing the graph $G_B$ to the following graph using a similar update:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

1. **Hub node**
2. **Central node**
3. **Authority node**

$$x_{1p}(k+1) = \sum_{q:(p,q) \in E} x_{2q}(k)$$

$$x_{2p}(k+1) = \sum_{q:(q,p) \in E} x_{1q}(k) + \sum_{q:(p,q) \in E} x_{3q}(k)$$

$$x_{3p}(k+1) = \sum_{q:(q,p) \in E} x_{2q}(k)$$
A first step toward generalizing Kleinberg’s approach: consider comparing the graph $G_B$ to the following graph using a similar update:

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}_{k+1} = \begin{bmatrix}
0 & B & 0 \\
B' & 0 & B \\
0 & B' & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}_k = (A \otimes B + A' \otimes B') \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}_k
\]

(use this construction for automatic synonym extraction)
In general, the nodes of two graphs $G_A$ and $G_B$ can be compared via the following update:

$$\bar{x}_{k+1} = (A \otimes B + A' \otimes B')\bar{x}_k$$

**Ex.**

<table>
<thead>
<tr>
<th>nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.443</td>
<td>0.104</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.280</td>
<td>0.396</td>
<td>0.086</td>
</tr>
<tr>
<td>3</td>
<td>0.086</td>
<td>0.396</td>
<td>0.280</td>
</tr>
<tr>
<td>4</td>
<td>0.222</td>
<td>0.049</td>
<td>0.222</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.104</td>
<td>0.443</td>
</tr>
</tbody>
</table>
Coupled edge and node scoring

- **Idea:** use this iterative approach to assign *edge similarity scores* as well as *node similarity scores*.

- **Couple the definitions in the following manner:**
  \[
  x_{ij} = \text{similarity between node } i \text{ in } G_B \text{ and node } j \text{ in } G_A \\
  = \text{sum of pairwise similarities between adjacent edges}
  \]

  \[
  y_{ij} = \text{similarity between edge } i \text{ in } G_B \text{ and edge } j \text{ in } G_A \\
  = \text{sum of similarities of source and terminal nodes}
  \]
Coupled edge and node scoring

- Idea: use this iterative approach to assign edge similarity scores as well as node similarity scores.
- Couple the definitions in the following manner:

\[ x_{ij} = \text{similarity between node } i \text{ in } G_B \text{ and node } j \text{ in } G_A \]
\[ = \text{sum of pairwise similarities between adjacent edges} \]
\[ y_{ij} = \text{similarity between edge } i \text{ in } G_B \text{ and edge } j \text{ in } G_A \]
\[ = \text{sum of similarities of source and terminal nodes} \]
Coupled edge and node scoring

- Idea: use this iterative approach to assign edge similarity scores as well as node similarity scores.
- Couple the definitions in the following manner:
  \[ x_{ij} = \text{similarity between node } i \text{ in } G_B \text{ and node } j \text{ in } G_A \]
  \[ = \text{sum of pairwise similarities between adjacent edges} \]
  \[ y_{ij} = \text{similarity between edge } i \text{ in } G_B \text{ and edge } j \text{ in } G_A \]
  \[ = \text{sum of similarities of source and terminal nodes} \]
Coupled edge and node scoring

- Idea: use this iterative approach to assign *edge similarity scores* as well as *node similarity scores*.
- Couple the definitions in the following manner:

  \[ x_{ij} = \text{similarity between node } i \text{ in } G_B \text{ and node } j \text{ in } G_A \]
  \[ y_{ij} = \text{sum of pairwise similarities between adjacent edges} \]

  \[ y_{ij} = \text{similarity between edge } i \text{ in } G_B \text{ and edge } j \text{ in } G_A. \]
  \[ = \text{sum of similarities of source and terminal nodes} \]

\[
\bar{x}_{k+1} = \left[ A_S \otimes B_S + A_T \otimes B_T \right] \bar{y}_k
\]

\[
\bar{y}_{k+1} = \left[ A_S' \otimes B_S' + A_T' \otimes B_T' \right] \bar{x}_k
\]

\[
\begin{align*}
[A_s]_{ij} &= \begin{cases} 
1 & s(j) = i \\
0 & \text{else}
\end{cases} \\
[A_t]_{ij} &= \begin{cases} 
1 & t(j) = i \\
0 & \text{else}
\end{cases}
\end{align*}
\]
Example

**Blondel, Van Dooren, et al. similarity scores**

<table>
<thead>
<tr>
<th>nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.443</td>
<td>0.104</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.280</td>
<td>0.396</td>
<td>0.086</td>
</tr>
<tr>
<td>3</td>
<td>0.086</td>
<td>0.396</td>
<td>0.280</td>
</tr>
<tr>
<td>4</td>
<td>0.222</td>
<td>0.049</td>
<td>0.222</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.104</td>
<td>0.443</td>
</tr>
</tbody>
</table>

**Coupled model similarity scores**

<table>
<thead>
<tr>
<th>nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.324</td>
<td>0.054</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.177</td>
<td>0.587</td>
<td>0.018</td>
</tr>
<tr>
<td>3</td>
<td>0.018</td>
<td>0.587</td>
<td>0.177</td>
</tr>
<tr>
<td>4</td>
<td>0.127</td>
<td>0.010</td>
<td>0.127</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.054</td>
<td>0.324</td>
</tr>
</tbody>
</table>
Application: Graph Matching

- Assign a correspondence between nodes and/or edges of each graph to maximize some performance criteria.
  - The Approach: apply Hungarian algorithm to node similarity matrix to maximize the sum of matched scores.
Application: Graph Matching

- Task: subgraph matching
  - Generate a random graph, $G$.
  - Select a subgraph, $S$.
  - Compute the node similarity matrices between $G$ and $S$.
  - Apply the Hungarian algorithm to `best' match the nodes of $S$ to those in $G$ by finding a matching that maximizes the sum of matched scores.
  - Record successes for nodes that are matched with their original identifier.
Application: Graph Matching

- Task: subgraph matching
  - Generate a random graph, $G$
  - Select a subgraph, $S$
  - Compute the node similarity matrices between $G$ and $S$
  - Apply the Hungarian algorithm to `best' match the nodes of $S$ to those in $G$ by finding a matching the maximizes the sum of matched weights.
  - Record successes for nodes that are matched with their original identifier

Yields a lower bound on the success of the matching process
Application: Graph Matching

- Using local edge similarity to improve scores:

\[ x_{aa'} \]

\[ x_{aa'}^* = x_{aa'} + m_{aa'} \]
Application: Graph Matching

Average Proportion of Correctly Matched Nodes
(Graph Size = 15 Nodes, Connectivity = 0.5)
Application: Graph Matching

- Exploring the impact of node labeling:

![Graph Matching Diagram]
Application: Graph Matching

Average Proportion of Correctly Matched Nodes
(Graph Size = 15 Nodes, Connectivity = 0.5)
Current/future work

- How does graph structure (e.g., cycles, paths, completeness) impact similarity scores?
- What can be inferred about a pair of graphs from a similarity measurement?
- What kinds of tasks is this measure appropriate for?
Acknowledgments

- George Verghese, MIT
- Sandip Roy, WSU
- Paul Van Dooren, Université catholique de Louvain

Work supported by a NSF Graduate Research Fellowship.