CSE 590
Data Science Fundamentals

Graph data Mining

Klaus Mueller

Computer Science Department
Stony Brook University and SUNY Korea
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro, schedule, and logistics</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Data Science components and tasks</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Data types</td>
<td>Project #1 out</td>
</tr>
<tr>
<td>4</td>
<td>Introduction to R, statistics foundations</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Introduction to D3, visual analytics</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Data preparation and reduction</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Data preparation and reduction</td>
<td>Project #1 due</td>
</tr>
<tr>
<td>8</td>
<td>Similarity and distances</td>
<td>Project #2 out</td>
</tr>
<tr>
<td>9</td>
<td>Similarity and distances</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Cluster analysis</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Cluster analysis</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Pattern miming</td>
<td>Project #2 due</td>
</tr>
<tr>
<td>13</td>
<td>Pattern mining</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Outlier analysis</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Outlier analysis</td>
<td>Final Project proposal due</td>
</tr>
<tr>
<td>16</td>
<td>Classifiers</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Midterm</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Classifiers</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Optimization and model fitting</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Optimization and model fitting</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Causal modeling</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Streaming data</td>
<td>Final Project preliminary report due</td>
</tr>
<tr>
<td>23</td>
<td>Text data</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Time series data</td>
<td></td>
</tr>
<tr>
<td><strong>25</strong></td>
<td><strong>Graph data</strong></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Scalability and data engineering</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Data journalism</td>
<td>Final Project slides and final report due</td>
</tr>
</tbody>
</table>
Graphs are everywhere
- chemical compounds (Cheminformatics)
- protein structures, biological pathways/networks (Bioinformactics)
- program control flow, traffic flow, and workflow analysis
- XML databases, Web, and social network analysis

Graph is a general model
- trees, lattices, sequences, and items are degenerated graphs

Diversity of graphs
- directed vs. undirected, labeled vs. unlabeled (edges & vertices), weighted, with angles & geometry (topological vs. 2-D/3-D)

Complexity of algorithms:
- many problems are of high complexity (NP complete)
Graphs are Everywhere

Aspirin

Yeast protein interaction network

Internet

Co-author network

Graphs are Everywhere

Social Network

Event Log Graph

Workflow

Mesh
Graphs are suitable for capturing arbitrary relations between the various elements.

Provide enormous flexibility for modeling the underlying data as they allow the modeler to decide on what the elements should be and the type of relations to be modeled.
Frequent subgraphs

- a (sub)graph is **frequent** if its support (occurrence frequency) in a given dataset is no less than a minimum support threshold
- support of a graph $g$ is defined as the percentage of graphs in $G$ which have $g$ as subgraph

Applications of graph pattern mining:

- mining biochemical structures
- program control flow analysis
- mining XML structures or Web communities
- building blocks for graph classification, clustering, compression, comparison, and correlation analysis
EXAMPLE – FREQUENT SUBGRAPHS

GRAPH DATASET

(T1)

(T2)

(T3)

FREQUENT PATTERNS
(MIN SUPPORT IS 2)

(1)

N
\equiv

(2)

N
\equiv

Example – Frequent Subgraphs

Graph Dataset

Frequent Patterns (Min Support Is 2)

1: makepat
2: esc
3: addstr
4: getccl
5: dodash
6: in_set_2
7: stclose
Graph Representations

- adjacency list

- adjacency matrix

- incidence matrix
Formalizes the notion of equal graphs

More formally an isomorphism of graphs \( G_1 \) and \( G_2 \) is a bijection \( f : V(G_1) \mapsto V(G_2) \) that preserves adjacency.

If \( G_1 = G_2 \) then the obtained mapping becomes an automorphism - a isomorphism from the graph to itself.

- if there is an automorphism of \( f \) of graph \( G \) such that the vertex \( v \) is mapped to vertex \( u \) then in a way the neighborhood of \( u \) and \( v \) "looks" the same.
Two subgraphs that are isomorphic

Maximum common subgraph (MCS)
- the largest possible subgraph that cannot be extended by an addition of a vertex
- finding it is an optimization problem that is known to be NP-hard
- there can be many MCS for a pair of graphs $G_1$ and $G_2$
When two graphs share a large subgraph in common, it is indicative of similarity

Un-normalized non-matching measure

\[ U(G_1, G_2) = |G_1| + |G_2| - 2 \cdot |MCS(G_1, G_2)| \]

- equal to the number of non-matching nodes between the two graphs because it subtracts out the number of matching nodes \(|MCS(G_1,G_2)|\) from each of \(|G_1|\) and \(|G_2|\) and then sums them
- unnormalized because the value of the distance depends on the raw size of the underlying graphs.
- not desirable because it is more difficult to compare distances between pairs of graphs of varying size
- more effective when the different graphs in the collection are of approximately similar size.
Union-normalized distance

- within [0.1]

$$UDist(G_1, G_2) = 1 - \frac{|MCS(G_1, G_2)|}{|G_1| + |G_2| - |MCS(G_1, G_2)|}$$

- normalizes the number of non-matching nodes $U(G_1,G_2)$ between the two graphs (unnormalized measure) with the number of nodes in the union of the two graphs

$$UDist(G_1, G_2) = \frac{\text{Non-matching nodes between } G_1 \text{ and } G_2}{\text{Union size of } G_1 \text{ and } G_2}$$

- intuitively easier to interpret– two perfectly matching graphs will have a distance of 0 from one another, and two perfectly nonmatching graphs will have a distance of 1
Max-normalized distance

- within [0.1]

\[ MDist(G_1, G_2) = 1 - \frac{|MCS(G_1, G_2)|}{\max\{|G_1|, |G_2|\}} \]

Any of these distance measures can be computed effectively only for small graphs

- for larger graphs, it becomes computationally too expensive to evaluate these measures because of the need to determine the maximum common subgraph between the two graphs
- use lexicon-base metrics for large graphs (see next)
Frequent Substructure-Based Distance Computation

Algorithm
- create a lexicon of frequent subgraph patterns by frequent subgraph mining
- reduce the overlap among the frequent subgraph patterns
- create a new feature $f_i$ for each frequent subgraph $S_i$ selected
- it will create a feature set of size $d$
- for each graph $G_i$, create a vector-space representation in terms of the features $f_1 \ldots f_d$
- each graph contains the features, corresponding to the subgraphs that it contains
- the frequency of each feature is the number of occurrences of the corresponding subgraph in the graph $G_i$
- optionally apply $tf-idf$ normalization
- use any similarity function to compute distances between graph objects
Example
A-Priori Based Graph Mining

Looks for frequent sub-structures (sub-graphs)

Recall a-priori pruning principle:

- if there is any item set which is infrequent, then its superset should not be generated/tested
- apply the same principle for sub-graphs
- size of a subgraph may refer to either its nodes or edges depending on the specific algorithm used
Algorithm \text{GraphApriori}(\text{Graph Database}: \mathcal{G}, \\
\text{Minimum Support}: \text{minsup});

begin
\mathcal{F}_1 = \{ \text{All Frequent singleton graphs} \};
\quad k = 1;
\quad \text{while } \mathcal{F}_k \text{ is not empty do begin}
\quad \quad \text{Generate } \mathcal{C}_{k+1} \text{ by joining pairs of graphs in } \mathcal{F}_k \text{ that share a subgraph of size } (k - 1) \text{ in common;}
\quad \quad \text{Prune subgraphs from } \mathcal{C}_{k+1} \text{ that violate downward closure;}
\quad \quad \text{Determine } \mathcal{F}_{k+1} \text{ by support counting on } (\mathcal{C}_{k+1}, \mathcal{G}) \text{ and retaining subgraphs from } \mathcal{C}_{k+1} \text{ with support at least } \text{minsup};
\quad \quad k = k + 1;
\quad \text{end};
\quad \text{return}(\bigcup_{i=1}^{k} \mathcal{F}_i);
\end{end}

\textbf{Downward closure}

- any subset of a frequent itemset must be frequent
**Example: Node-Based Join**

Number of edges is not constrained
- this can lead to ambiguities
Number of nodes is not constrained
- new level could have the same number than next level
- another form of ambiguity
Either distance-based or frequent substructure-based

- distance-based methods are more effective for smaller graphs, in which distances can be computed robustly and efficiently
- frequent substructure-based methods are appropriate for larger graphs where distance computations become qualitatively and computationally impractical

Distance-based clustering

- use methods like k-medoids or spectral clustering
- computationally expensive to compute distances between large graph objects
- effectiveness also suffers for large graphs because these graphs may be similar only in some portions that repeat frequently
  → the rare (and unique) portions of the graph may not factor in
- might use a substructure-based distance function instead
Graph Clustering

Frequent substructure-based methods

- extract frequent subgraphs from the data and use their membership in input graphs to determine clusters

Algorithm

- apply frequent subgraph mining methods discussed to discover frequent subgraph patterns in the underlying graphs
- select a subset of subgraphs to reduce overlap among the different subgraphs
- create a new feature $f_i$ for each frequent subgraph $S_i$ discovered
- gives rise to lexicon of $d$ features
- create a $d$-vector of features – one element per feature
- represent each graph in terms of the features and their frequency
- cluster as usual
A frequent substructure-based clustering method

Originally proposed for XML graphs
- a substructure can be viewed as a PROJection of the graph

Algorithm $XProj$ (Graph Database: $G$, Minimum Support: $\text{minsup}$
Structural Size: $l$, Number of Clusters: $k$ )

begin
    Initialize clusters $C_1 \ldots C_k$ randomly;
    Compute frequent substructure sets $F_1 \ldots F_k$ from $C_1 \ldots C_k$;
repeat
    Assign each graph $G_j \in G$ to the cluster $C_i$ for which the former’s similarity to $F_i$ is the largest $\forall i \in \{1 \ldots k\}$;
    Compute frequent substructure set $F_i$ from $C_i$ for each $i \in \{1 \ldots k\}$;
until convergence;
end
Use either distance or pattern/structure based methods discussed before to form the decision metric for any standard classifier

- kernel-based methods are also possible