Graph-based Learning

Larry Holder
School of Electrical Engineering and Computer Science
Washington State University
Graph-based Learning

- Multi-relational data mining and learning
- SUBDUE graph-based relational learner
  - Discovery
  - Clustering
  - Graph grammar learning
  - Supervised learning
Multi-Relational Data Mining

Looking for patterns involving multiple tables (relations) in a relational database

<table>
<thead>
<tr>
<th>ID</th>
<th>Last</th>
<th>First</th>
<th>Age</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Doe</td>
<td>John</td>
<td>30</td>
<td>80000</td>
</tr>
<tr>
<td>P2</td>
<td>Doe</td>
<td>Sally</td>
<td>29</td>
<td>90000</td>
</tr>
<tr>
<td>P3</td>
<td>Smith</td>
<td>Robert</td>
<td>35</td>
<td>100000</td>
</tr>
</tbody>
</table>

RichCouple(X,Y) \(\rightarrow\) Person(X,LastX,FirstX,AgeX,IncX) & Person(Y,LastY,FirstY,AgeY,IncY) & Married(X,Y) & (IncX + IncY) > 150000.
Multi-Relational Data Mining

Approaches
- Transform to non-relational problem
- First-order logic based
  - Inductive Logic Programming (ILP)
- Graph based
Graph-based Data Mining

Finding all subgraphs $g$ within a set of graph transactions $G$ such that

$$\frac{freq(g)}{|G|} > t$$

- where $t$ is the minimum support
Graph-based Data Mining

**Systems**

- Apriori-based Graph Mining (AGM)
  - Inokuchi, Washio and Motoda, 2003
- Frequent Sub-Graph discovery (FSG)
  - Kuramochi and Karypis, 2001
- Graph-based Substructure pattern mining (gSpan)
  - Yan and Han, 2002

**Focus on pruning and fast, code-based graph matching**
Graph-based Relational Learning

Finding patterns in graph(s)

- Discovery
- Clustering
- Supervised learning
Graph-based Relational Learning

- **Graph-Based Induction (GBI)**
  - Yoshida, Motoda and Indurkhya, 1994

- **SUBstructure Discovery Using Examples (SUBDUE)**
  - Cook and Holder, 1994

- Focus on efficient subgraph generation and compression-based heuristic search
SUBDUE Graph-based Discovery

- Graph representation
- Graph compression and MDL
- Discovery algorithm
- Inexact graph match
- Background knowledge
- Parallel/distributed discovery
Graph Representation

- Input is a labeled (vertices and edges) directed graph
- A substructure is a connected subgraph
- An instance of a substructure is an isomorphic subgraph of the input graph
- Input graph compressed by replacing instances with vertex representing substructure
Graph Representation
Graph Compression and MDL

Minimum Description Length (MDL) principle

- Best theory minimizes description length of theory and the data given theory

Best substructure $S$ minimizes description length of substructure definition $DL(S)$ and compressed graph $DL(G|S)$

$$\min_S (DL(S) + DL(G|S))$$
Discovery Algorithm

1. Create substructure for each unique vertex label

Substructures:
triangle (4), square (4), circle (1), rectangle (1)
Discovery Algorithm

2. Expand best substructures by an edge or edge+neighboring vertex

Substructures:
- triangle on square
- square on rectangle
- triangle on circle
- square on rectangle
- triangle on square
- square on rectangle
- triangle on circle
- square on rectangle
- triangle on triangle
- square on square
- circle on rectangle
- rectangle on triangle
Discovery Algorithm

3. Keep only best $\text{beam-width}$ substructures on queue

4. Terminate when queue is empty or $\#\text{discovered substructures} > \text{limit}$

5. Compress graph and repeat to generate hierarchical description
DNA Example
Sample SUBDUE Input

sample.g:

v 1 object e 1 11 shape
v 2 object e 2 12 shape
v 3 object e 3 13 shape
v 4 object e 4 14 shape
v 5 object e 5 15 shape
v 6 object e 6 16 shape
v 7 object e 7 17 shape
v 8 object e 8 18 shape
v 9 object e 9 19 shape
v 10 object e 10 20 shape
v 11 triangle e 1 5 on
v 12 triangle e 2 6 on
v 13 triangle e 3 7 on
v 14 triangle e 4 8 on
v 15 square e 5 10 on
v 16 square e 9 10 on
v 17 square e 10 2 on
v 18 square e 10 3 on
v 19 circle e 10 4 on
v 20 rectangle
Inexact Graph Match

- Some variations may occur between instances
- Want to abstract over minor differences
- Difference = cost of transforming one graph to make it isomorphic to another
- Match if cost/size < threshold
Least-cost match is \{((1,4), (2,3))\}
Inexact Graph Match

- Vertices considered by degree
- Polynomials constrained
  - Greedy after $n^k$ partial mappings considered
  - Suboptimal mappings rare for $k > 2$
Background Knowledge

- User-defined substructures
- Two alternative uses
  - Prime search queue
  - Initial graph compression
- Variant of discovery algorithm used to generate instances
Parallel/Distributed Discovery

- Divide graph into $P$ partitions
- Distribute to $P$ processors
- Each processor performs serial discovery on local partition
- Broadcast best substructures, evaluate on other processors
- Master processor stores best global substructures
Graph-based Clustering

- Hierarchical, conceptual clustering
- Previous work defined classification trees
  - Inadequate in relational domains
- Better hierarchical description: classification lattice
  - A cluster can have more than one parent
  - A parent can be at any level (not only one level above)
- Use iterative graph-based discovery
Clustering: DNA
Clustering: DNA

Coverage
- 61%
- 68%
- 71%
Learning Graph Grammars

Graph grammar production: $S \rightarrow P$
- $S$ is a non-terminal
- $P$ is a graph containing terminals and/or non-terminals
- $S \rightarrow P_1 \mid P_2 \mid \ldots \mid P_n$

Recursive production: $S \rightarrow P \ S \mid P$
- $P$ linked to $S$ via a single edge
- Algorithm exponential in number of linking edges
Example Graph Grammar

\[ S_2 \rightarrow a \quad S_2 \]
\[ S_3 \rightarrow b \quad S_3 \]
\[ S_3 \rightarrow c \quad d \quad e \quad f \]
Graph Grammar Learning

SUBDUE Extensions (SubdueGL)

- Each iteration results in a graph grammar production substructure
- Production used to compress graph
  - Replace instances of right-hand side with new vertex labeled with non-terminal on left-hand side
- Iterations continue until entire graph compressed to single non-terminal
SubdueGL Example

- **Input graph**
  - **Edge labels:** ‘t’, ‘s’, ‘next’
SubdueGL Example

- First production rule

- Input graph parsed by first production
SubdueGL Example

Second and third production rules

Input graph parsed by productions
Graph-Based Supervised Learning

Input now a set of positive graphs and a set of negative graphs
Graph-Based Supervised Learning

Solution 1
- Find substructure compressing positive graphs, but not negative graphs
- Compress graphs and iterate until no further compression

Problem
- Compressing, instead of removing, partially-covered positive graphs leads to overly-specific hypotheses
Graph-Based Supervised Learning

Solution 2

- Find substructure covering (i.e., subgraph of) positive graphs, but not negative graphs
- Remove covered positive graphs and iterate until all covered

Substructure value = 1 - Error

\[
Error = \frac{\#PosEgsNotCovered + \#NegEgsCovered}{\#PosEgs + \#NegEgs}
\]
Supervised Learning: Cancer

- Chemical toxicity
- SUBDUE achieved 62% accuracy classifying carcinogenic vs. non-carcinogenic compounds
Application Domains

- **Biochemical domains**
  - Protein data
  - DNA data
  - Toxicology (cancer) data

- **Spatial-temporal domains**
  - Earthquake data
  - Aircraft Safety and Reporting System

- **Web topology and search**

- **Social network analysis**

...
Summary

- Multi-relational data mining and learning
- Graph-based relational learning
  - Discovery
  - Clustering
  - Graph grammar learning
  - Supervised learning
Future Directions

- Efficient graph-based learning from incremental streaming data
- Supervised graphs
  - All examples in one, connected graph
- Graph-based anomaly detection
- Improved scalability
  - Graph and subgraph isomorphism
Further Information

• SUBDUE Project
  • http://www.subdue.org