National Institute of Advanced Industrial Science and Technology (AIST)


- 2500 scientists, 700 administrative staff, 5200 scientists from outside

- Since 2001
Computational Biology Research Center

- Odaiba, Tokyo

Developing Novel Methods and Tools for
- Genome Informatics
- Molecular Informatics
- Cellular Informatics

Diverse Collaboration with Companies and Universities
Koji Tsuda: Short Bio

1998 PhD in Kyoto Univ, Join ETL
2000 GMD FIRST (Berlin, Germany)
2001 Join CBRC/AIST
2003-2004, 2006-2008 Max Planck Institute for Biological Cybernetics, Tuebingen, Germany
2009 Back to CBRC, Machine Learning Group Leader
About this lecture

- How to extract knowledge from structured data
- Itemset mining, tree mining, graph mining
  - “Reverse Search Principle”
- Learning from structured data
- Kernels for structured data
Chapter 1 (Data Mining)

1. Structured Data in Biology
2. Itemset Mining
3. Closed Itemset Mining
4. Ordered Tree Mining
5. Unordered Tree Mining
5. Graph Mining
6. Dense Module Enumeration
Agenda 2 (Learning from Structured Data)

1. Preliminaries
2. Graph Clustering by EM
3. Graph Boosting
4. Regularization Paths in Graph Classification
5. Itemset Boosting for predicting HIV drug resistance
Agenda 3 (Kernel)

1. Kernel Method Revisited
2. Marginalized Kernels (Fisher Kernels)
3. Marginalized Graph Kernels
4. Weisfeiler-Lehman kernels
5. Reaction Graph kernels
6. Concluding Remark
Part 1: Structural Data in Biology
Biological Sequences

- **DNA sequences (A,C,G,T)**
  - Gene Finding, Splice Sites
- **RNA sequences (A,C,G,U)**
  - MicroRNA discovery, etc.
- **Amino acid sequences (20 symbols)**
  - Remote homolog detection, Fold recognition etc.
Structures hidden in sequences (I)

Exon/intron of DNA (Gene)
Structures hidden in sequences (II)

It is crucial to infer hidden structures and exploit them for classification.

Biological Graphs

RNA
Secondary Structure

Protein
3D Structures

Biological Graphs
Molecular graphs

**Structure:** Thiamine (Vitamin B₁)

- **Implicit hydrogens**
- **Explicit hydrogens**

**Molecular graph**

- Graph = a set of **dots** & **lines** (or nodes & edges)

- **Hydrogen**
- **Carbon**
- **Oxygen**
- **Nitrogen**
- **Sulfur**

- **Single bond**
- **Double bond**

**Abstraction**
Gene Expression Data

- Measurement of many mRNAs in the cell
- Rough estimate of amount of proteins
- Time-series or not
- Snapshot of the underlying dynamic system
Biological Networks

- Protein-protein physical interaction
- Metabolic networks
- Gene regulatory networks
- Network induced from sequence similarity

- Thousands of nodes (genes/proteins)
- 100,000s of edges (interactions)
Physical Interaction Network
Many possible prediction problems..

**Given Data**
- sequence
- structure
- expression
- phylogeny

**Predicted Property**
- **structure** (3D coordinates of the atoms)
- **function** (e.g., according to GO or MIPS)
- **interactions** (with other proteins, DNA or metabolites)
- **localization** (e.g., compartment)
Part 2: Itemset mining
Data Mining

- A formal study of efficient methods for extracting interesting rules and patterns from massive data

- Frequent itemset mining (Agrawal and Srikant 1994)

- Closed pattern mining

- Structured data mining (Sequence, Trees, and Graphs)
Frequent Itemset Mining

[Agrawal, Srikant, VLDB'94]

• Finding all "frequent" sets of elements (items) appearing \( \sigma \) times or more in a database

Minsup \( \sigma = 2 \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>O</td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>t3</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>t4</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>t5</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

Frequent sets
\( \emptyset, 1, 2, 3, 4, 12, 13, 14, 23, 24, 124 \)

\( X = \{2, 4\} \) appears three times, thus frequent

The itemset lattice \( (2^\Sigma, \subseteq) \)
Definitions: Database

- A set \( \Sigma = \{ 1, \ldots, n \} \) of items (elements)
- Transaction database
  - A set \( T = \{ t_1, \ldots, t_m \} \) of subsets of \( \Sigma \)
  - Each subset \( t \subseteq \Sigma \) is called a transaction

\( L = \{1, 2, 3, 4\} \)

Alphabet of items

<table>
<thead>
<tr>
<th>id</th>
<th>transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>1, 3</td>
</tr>
<tr>
<td>t2</td>
<td>2, 4</td>
</tr>
<tr>
<td>t3</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>t4</td>
<td>1, 2, 4</td>
</tr>
</tbody>
</table>
Definitions: Frequent sets

- Itemset $X$ appears in transaction $t$: $X \subseteq t$
- Occurrence of $X$ in database $T$:
  $$\text{Occ}(X, T) = \{ t \in T : X \subseteq t \}$$
- Frequency of $X$: $\text{Fr}(X, T) = |\text{Occ}(X, T)|$
- Minimum support (minsup): $0 \leq \sigma \leq |T|$
- $X$ is frequent in $T$ if $\text{Fr}(X, T) \geq \sigma$.

$I = \{1, 2, 3, 4\}$

**Alphabet of items**

**Transaction database**

<table>
<thead>
<tr>
<th>id</th>
<th>transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>1, 3</td>
</tr>
<tr>
<td>t2</td>
<td>2, 4</td>
</tr>
<tr>
<td>t3</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>t4</td>
<td>1, 2, 4</td>
</tr>
</tbody>
</table>

**Occurrences and frequencies of itemsets**

- $\text{Occ}(3, T) = \{t1, t3\}$
  - $\text{Fr}(3, T) = 2$
- $\text{Occ}(24, T) = \{t2, t3, t4\}$
  - $\text{Fr}(24, T) = 3$
Market Basket Data

- Popular application of itemset mining
- Business and Market data analysis

- Transaction Data of purchase
  - a transaction or a "basket"

<table>
<thead>
<tr>
<th>ID</th>
<th>Chips</th>
<th>Mustard</th>
<th>Sausage</th>
<th>Softdrink</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>002</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>003</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>004</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>005</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>006</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>007</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>008</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>009</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Meaning of the transaction 003
  "Customer 003 bought Chips and Sausage together in his basket"
DAG of itemsets: Hasse diagram

- Edge: Adding one item

```
empty
1 2 3 4
1,2 1,3 1,4 2,3 2,4 3,4
1,2,3 1,2,4 1,3,4 2,3,4
1,2,3,4
```
Need a tree to avoid duplication

Enumeration Tree by Lexicographical Order
Backtracking Algorithm: FP Growth etc.

- Monotonicity: Support only decreases
- Depth First Traversal, Prune if support < \( \sigma \)
Association Rule Mining

- Confidence: \( \frac{\text{Supp}(A \cup B)}{\text{Supp}(A)} \)
  - Probability of B, Given A

- What item is likely to be bought when A is bought

- Search: large support, confidence large

- Post-processing of itemset mining
Summary: Itemset mining

- Itemset mining is the simplest of all mining algorithms
- Need to maintain occurrence of each pattern in database
- Tree by lexicographical order is (implicitly) used
Part 3: Closed Itemset mining
Problem in Frequent Pattern Mining

- **Huge Number of frequent itemsets**
- **Hard to analyze**
- **Most of them are similar**

An input transaction database

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td></td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

minsup $\sigma = 2$

Huge number of frequent itemsets discovered in $T$

Frequent sets

mining
Solution: Closed Pattern Mining

- Find only closed patterns

- Observation: Most frequent itemset $X$ can be extended without changing occurrence by adding new elements

- def ([Pasquier et al., ICDT'99]). An itemset $X$ is a closed set if and only if there is no proper superset of $X$ with the same frequency (thus the same occurrence set).
Closed Pattern Mining

- A closed itemset is the maximal set among all itemsets with the same occurrences.
- Equivalence class \([X] = \{Y| \text{Occ}(X)=\text{Occ}(Y) \}\).

Database

<table>
<thead>
<tr>
<th>id</th>
<th>records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABCDE</td>
</tr>
<tr>
<td>2</td>
<td>AC</td>
</tr>
<tr>
<td>3</td>
<td>BE</td>
</tr>
<tr>
<td>4</td>
<td>BCE</td>
</tr>
</tbody>
</table>

Closed sets (maximal sets)

Closed sets (maximal sets)
Brute-force: Stupid Baseline

**ALGORITHM Bruteforce**
- First, generate all frequent itemsets
- Check them one by one via maximality test

**Maximality test** for each candidate frequent set $X$
- Add some element $e$ in $\Sigma$ to $X$
- If $\text{Freq}(X \cup \{e\})$ is properly less than $\text{Freq}(X)$ then reject $X$. 

•34
Brute Force

- **STEP1)** first, generate all frequent sets
Brute force

- STEP 1) first, generate all frequent sets
- STEP 2) make closedness test for each set

Database $T$

<table>
<thead>
<tr>
<th>id</th>
<th>records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A B C E$</td>
</tr>
<tr>
<td>2</td>
<td>$A C$</td>
</tr>
<tr>
<td>3</td>
<td>$B E$</td>
</tr>
<tr>
<td>4</td>
<td>$B C E$</td>
</tr>
</tbody>
</table>

All itemsets in $T$

- $[1,2]$  
- $[1,2,4]$  
- $[1,3,4]$  
- $[1,4]$  
- $[1,2,3,4]$  

Closed sets (maximal sets)

Equivalence class w.r.t. occurrences

Occurrence (set of ids)
Brute force

- **STEP 1)** first, generate all frequent sets
- **STEP 2)** make closedness test for each set
- **STEP 3)** finally, extract all closed sets

**Database** $T$

<table>
<thead>
<tr>
<th>id</th>
<th>records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C E</td>
</tr>
<tr>
<td>2</td>
<td>A C</td>
</tr>
<tr>
<td>3</td>
<td>B E</td>
</tr>
<tr>
<td>4</td>
<td>B C E</td>
</tr>
</tbody>
</table>

**All itemsets in** $T$

- $\emptyset$
- $\{1,2\}$
- $\{1,2,4\}$
- $\{1,3,4\}$
- $\{1,2,3,4\}$

**Closed sets (maximal sets)**

- $\{1\}$
- $\{1,2\}$
- $\{1,2,4\}$
- $\{1,3,4\}$
- $\{1,2,3,4\}$

**Occurrence (set of ids)**

- $\{1,2\}$

**Equivalence class w.r.t. occurrences**

- $\{1\}$
- $\{1,2\}$
- $\{1,2,4\}$
- $\{1,3,4\}$
- $\{1,2,3,4\}$

All closed sets are found!
Complexity of Enumeration Algorithms

- Number of patterns usually exponential to input size
- Delay: Time between two pattern outputs
- Brute-force is exponential delay w.r.t. pattern size
To achieve *linear* delay,

- Must jump from closed set to closed set
- How to define the search tree?
- Reverse search!

*(Avis and Fukuda 1996)*
Reverse Search: It’s a must

- A general mathematical framework to design enumeration algorithms
- Can be used to prove the correctness of the algorithm
- Popular in computational geometry
- Data mining algorithms can be explained in remarkable simplicity
Often, search space comes as a DAG

- Naive Backtracking = Duplication
- Duplication check by Marking = Exponential Memory
- How to visit all nodes without duplication?
Reduction Map

- Mapping from a children to the parent
- Reduction map for closed itemset
  - Shrink the itemset until occurrence changes
  - Take “closure operation”
Closure Operation

- **closure(X)** of a set X:
  - Closed set computed by
  \[
  \text{closure}(X) = \bigcap \{ t \in T : X \subseteq t \}.
  \]
  (taking the intersection of all transactions in T that X occurs as subset)

闭包运算

- **closure(X)** of a set X:
  - 封闭集由
  \[
  \text{closure}(X) = \bigcap \{ t \in T : X \subseteq t \}.
  \]
  (取 T 中所有 X 作为子集的交易的交集)
Example of Closure Operation

- Non-closed itemset: (B, C)
- Occurrence: 1, 4
- Take Intersection of 1 and 4
  \[(A, B, C, E) \cap (B, C, E) = (B, C, E)\]
- This is closed itemset

```
<table>
<thead>
<tr>
<th>id</th>
<th>records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C E</td>
</tr>
<tr>
<td>2</td>
<td>A C</td>
</tr>
<tr>
<td>3</td>
<td>B E</td>
</tr>
<tr>
<td>4</td>
<td>B C E</td>
</tr>
</tbody>
</table>
```
By applying the reduction map to all nodes, enumeration tree is defined.

- But arrows are in reverse direction.
Children generation

- In backtracking, one has to generate all children of the current node.
- Inverse of reduction map:
  - Generate all children candidates
  - Apply reduction map to them
  - Remove if not coming back
Reverse Search Theorem

To prove the correctness, prove the following

- Reduction map is uniquely defined on all nodes
- By applying the reduction map repeatedly, one can reach the root node from any node
- Children generation is inverse of reduction map

Easy to check!
LCM = Linear Time Closed Sets Miner (Uno et al., 2003)

- Prefix Preserving Closure Extension
  - = Children generation from the reduction map
  - Linear Delay!
Closure Extension

- Repeat: Add an item and taking closure

Non-closed sets:

Closed sets:

\[ \text{closure}(X) \]

Step 1:

\[ Y = X \cup \{i\} \]

Step 2:

\[ Z = \text{closure}(X \cup \{i\}) \]

Start:

Closed set \( X \)

Add item \( i \)
Naïve Closure Extension: Duplication!

- closure extension

\[ T = \{1,2,5,6,7,9\}, \{1,2,7,8,9\}, \{1,2,7,9\}, \{2,7,9\}, \{2\} \]

\[ \varnothing \]

\[ \rightarrow \] DAG of closed itemsets

\[ \{1,2,7,8,9\} \rightarrow \{1,2,7,9\} \rightarrow \{1,2,5,6,7,9\} \rightarrow \{2,3,4,5\} \rightarrow \{2,5\} \rightarrow \{2\} \rightarrow \varnothing \]
Prefix Preserving Closure Extension

- Ensure any closed set is generated from a unique parent

**Def.** Closure tail of a closed itemset $P$

$\iff$ the minimum $j$ s.t. $\text{closure}(P \cap \{1,\ldots,j\}) = P$

**Def.** $H = \text{closure}(P \cup \{i\})$ is a PPC-extension of $P$

$\iff i > \text{closure tail}$ and

$H \cap \{1,\ldots,i-1\} = P \cap \{1,\ldots,i-1\}$
Enumeration tree by PPC extension

- closure extension $\Rightarrow$ DAG
- ppc extension $\Rightarrow$ tree

$T =$

1,2,5,6,7,9  
2,3,4,5  
1,2,7,8,9  
1,7,9  
2,7,9  
2

closure extension

ppc extension
Linear Delay in Pattern Size
(Uno, Uchida, Asai, Arimura, Discovery Science 2004)

**Theorem**: The algorithm LCM finds all frequent closed sets $X$ appearing in a collection of a transaction database $D$ in $O(lmn)$ time per closed set in the total size of $D$ without duplicates, where $l$ is the maximum length of transactions in $D$, and $n$ is the total size of $D$, $m$ is the size of pattern $X$.

Note: The output polynomial time complexity of Closed sets discovery is shown by [Makino et al. STACS2002]
Summary: Closed Itemset Mining

- Closure Extension: Jump from closed set to closed set
- LCM: Linear Delay
- Very fast in practice, too
  - Winner of FIMI’04 (Frequent Itemset Mining Implementation Workshop)
- Relation to clique enumeration (Arimura, Uno, SDM2009)
Part 4: Ordered Tree Mining
Frequent Ordered Tree Mining

- **Natural extension** of frequent itemset mining problem for trees
  - Finding all frequent substructure in a given collection of labeled trees
  - How to enumerate them without duplicates

- **Efficient DFS Algorithm**
  - FREQT [Asai, Arimura, SIAM DM2002]
  - TreeMiner [Zaki, ACM KDD2002]
  - Rightmost expansion technique
Labeled Ordered Trees

- Rooted:

- Ordered:
  - Siblings are ordered from left to right.

- Labeled
  - Each node has a label.

- A model of
  - HTML/XML
  - Hierarchical records
  - Dependency tree of natural language texts.
Matching between trees

Pattern tree $T$ matches a data tree $D$ ($T$ occurs in $D$)

There is a matching function $\phi$ from $T$ into $D$.

- $\phi$ is 1-to-1.
- $\phi$ preserves parent-child relation.
- $\phi$ preserves (indirect) sibling relation.
- $\phi$ preserves labels.
Frequency of a pattern tree

- A root occurrence of pattern T:
  - The node to which the root of T maps by a matching function
  - The frequency $fr(T) = \#\text{root occurrences of } T \text{ in } D$

Root occurrence list
$\text{Occ}_D(T) = \{2, 8\}$
Frequent Tree Mining Problem

- **Given:** a collection $S$ of labeled ordered trees and a minimum frequency threshold $\sigma$
- **Task:** Discover all frequent ordered trees in $S$ (with frequency no less than $\sigma$) without duplicates

- A minimum frequency threshold (min-sup) $s = 50\%$
Key: How to enumerate ordered trees without duplicates?

- **A naive algorithm**
  - Starting from the smallest tree
  - Grow a pattern tree by adding a new node one by one

- **Drawbacks**
  - Exponentially many different ways to generate the same pattern tree
  - Explicit duplication test needed

- **How to overcome this difficulty?**
An idea: DFS Code of Ordered Tree

- Depth-label sequence in the preorder traversal (depth first search)
  \[ S = ((d(v_1), l(v_1)), \ldots, (d(v_k), l(v_k))) \]

<table>
<thead>
<tr>
<th>depth</th>
<th>id</th>
<th>seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0A</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1B</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2A</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3C</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2B</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1B</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>2C</td>
</tr>
</tbody>
</table>

DFS code
Rightmost expansion

- Extending the DFS Code = Attaching a new node on the rightmost branch 
  \((d_1,l_1),..., (d_n,l_n), (d_{n+1},l_{n+1})\)
Searching frequent ordered trees

- Enumerate all frequent ordered trees by backtracking
- Tree extended only by rightmost extension = No duplication
Summary: Ordered tree mining

- Convert tree to a string (DFS Code)
- Adding element to the code = Rightmost extension
- It was relatively easy because nodes are ordered
  - How about unordered case?
Part 5: Unordered Tree Mining
Frequent Unordered Tree Mining

• Unordered trees: Non-trivial subclass of general graphs
• Problem: Exponentially many isomorphic trees
• Efficient DFS Algorithm
  – Unot [Asai, Arimura, DS'03]
  – NK [Nijssen, Kok, MGTS’03]
Given ordering among siblings, depth-first search (DFS) code is defined:

\[ \text{Code}(T) = ((\text{depth}(v_1), \text{label}(v_1)), \ldots, (\text{depth}(v_k), \text{label}(v_k))) \]

Code(T) = (0A,1B,2A,3C,2B,1B,2C)
Canonical representation

Ordered tree $T$ with lexicographically maximum code

$T_1$  
A  
B  
B  
A  
B  
C  

$T_2$  
A  
B  
B  
A  
B  
C  

$T_3$  
A  
B  
C  
A  
B  
C  

$T_4$  
A  
B  
C  
B  
A  
C  

(0A, 1B, 2A, 3C, 2B, 1B, 2C)  
(0A, 1B, 2B, 2A, 3C, 1B, 2C)  
(0A, 1B, 2C, 1B, 2A, 3C, 2B)  
(0A, 1B, 2C, 1B, 2B, 2A, 3C)
Left Heavy Condition (Nakano and Uno, 2002)

- $T(v)$: subtree rooted on $v$
- Ordered tree is canonical \textit{if and only if} $\text{Code}(T(v_1)) \geq \text{Code}(T(v_2))$
  for any pair of sibling nodes $v_1$ (left) and $v_2$ (right)
Reduction Map

- How to define parent from child in the enumeration tree
- Generate canonical tree of size $k-1$ from canonical tree of size $k$
- Remove the last element of DFS Code

Code(T) = (0A,1B,2A,3C,2B,1B,2C)
Children Generation

- Generate children candidates by rightmost extension
- Check the maximality of candidate based on left heavy property
- Discard if not maximal
Maximality Check by Left Heavy Property

- Code of left subtree must be larger than that of right subtree
- Check only rightmost sibling and second rightmost sibling
Complexity of UNOT

- Delay per pattern $O(kb^2 mn)$
- $k$: pattern size
- $b$: branching factor of the data tree
- $m$: size of data tree
- $n$: database size
Summary: Mining Unordered Tree

- The following three elements are necessary for a mining algorithm:
  - Canonical Representation
  - Reduction Map
  - Children Generation including Maximality Check

- Backtracking on the resulting enumeration tree
Part 6: Graph Mining
Frequent Subgraph Mining

Graph Database

Enumerate all subgraphs occurring more than 3 times

Patterns
Gspan (Yan and Han, 2002)

• Most widely used graph mining algorithm
• Can be interpreted with reverse search principle
  – Canonical representation?
  – Reduction map?
  – Children generation?
DFS Code for Graph

• Depth first search and preorder node labeling
• (src, dest, src_label, edge_label, dest_label)
• Some edges not traversed
  – backward edge (dest < src)
• Elements sorted in the code

{(0,1,A,a,A), (1,2,A,a,B), (2,0,B,a,A), (2,3,B,b,A)}
Canonical Representation

• Multiple DFS codes: different starting point and children ordering
• Minimum DFS Code: Lexicographically Minimum

((0,1),A,a,A), ((0,2),A,a,A)  ((0,1),A,a,A), ((1,2),A,a,A)
Reduction Map

• Removing the tail of minimum DFS code preserves minimality

min DFS:
\{e1, e2, e3, e4\}  \{e1, e2, e3\}
Children Generation

- Create candidates by adding an element to DFS code
- Check if each candidate is minimum
- If not, remove it
Minimality Check

- Reconstruct the graph from DFS Code
- Derive the minimum DFS Code by trying all DFSs
  - Speed up by traversing minimal label only
- If the minimal code is not identical to the original, prune it

\{(0,1,A,a,A), (1,2,A,a,B), (2,0,B,a,A), (2,3,B,b,A)\}

![Graph Diagram]

• 83
Summary: Graph Mining

• gSpan is a typical example of reverse search
• Not explained: Closed tree mining, Closed Graph mining
• Delay exponential to pattern size
  – It cannot be avoided due to NP-hardness of graph isomorphism
  – Yet it scales to millions for sparse molecular graphs
• Applications covered in next chapter
Part7: Dense Module Enumeration
Biological Motivation

- Most cellular processes performed by multi-component protein complexes
- Increasing amount of experimental protein interaction data available
- Our approach
  - Predict complexes (modules) from protein interaction network
  - Exploit additional information given by gene expression data, evolutionary conservation, phenotypic profiles etc.
Protein interaction networks

- Node: Proteins
- Edge: Physical interaction of two proteins

- Challenge 1: False negative edges
  - Go beyond clique search!
- Challenge 2: False positive edges
  - Assign confidence scores to edges

- Find node sets with high density of high confidence edges
Module Discovery

- Previous work
  - Clique percolation [Palla et al., 2005]
  - Partitioning
    - Hierarchical clustering [Girvan and Newman, 2001]
    - Flow Simulation [Krogan et al., 2006]
    - Spectral methods [Newman, 2006]
  - Heuristic Local Search [Bader and Hogue, 2003]

- Our approach
  - Exhaustively enumerate all dense subgraphs efficiently
Motivation for Enumeration Approach

- Detects overlapping modules
- Allows to specify minimum density for outcoming modules
- Outputs all modules satisfying the density threshold
Differential Expression Criterion

- Incorporation of gene expression
  - Presence of proteins depends on cell type

- Additional Criterion for modules
  - $e_1$: Num of conditions where whole module expressed
  - $e_0$: Num of conditions where whole module not expressed
  - Fix minimum values for both quantities
Problem Formalization

- Interaction network: \( G = (V, E(V)) \)
- Edge weights: \( 0 \leq w(\{u, v\}) \leq 1 \)
- Density of \( U \subset V \):

\[
d(U) = \frac{\sum_{\{u,v\} \in E(U)} w(\{u, v\})}{|U|(|U| - 1)/2}
\]

- Find all \( U \subset V \) with \( d(U) \geq \theta \), \( e_1(U) \geq n_1 \), and \( e_0(U) \geq n_0 \)
Typical Enumeration Algorithms

- Itemset mining, graph mining etc.
- Enumerate all entities whose frequency $\geq 10$
- Set up a search tree
- Tree Pruning by anti-monotonicity
  - An entity’s frequency is always smaller than that of sub-entity

Not generated
Network Example

Density of Modules

- $1,2,3 \mid 0.5$
- $1,3,4 \mid 0.9$
- $2,3,4 \mid 0.5$
- $1,2,4 \mid 0.3$
Graph-shaped Search Space of Modules
Choosing a search tree

- For efficient search, a search tree is needed
- There are many possible search trees
- Default: Lexicographical ordering
Density is not a monotonic criterion

- Subset of dense set is not necessarily dense
- Density does not decrease monotonically on a path
  - Pruning Impossible
Question

- Is it possible to make a search tree such that density decreases monotonically?
Question

- Is it possible to make a search tree such that density decreases monotonically?

- YES!
  - Use Reverse Search (Avis and Fukuda 1993)
Reverse search (Avis and Fukuda, 1993)

- Specify a search tree in the graph-shaped search space
- Reduction Mapping
  - Rule to generate a parent from a child
  - Remove the node with the smallest degree
  - Density always increase by the removal
Search Tree is uniquely specified by the reduction mapping

- Condition: Every node should converge to the root node by applying the reduction mapping repeatedly
Enumeration algorithm by reverse search

- A set of children is generated from a parent node
- Try every possible children, and choose the ones satisfying the reduction mapping
- Prune if no children exist
Constraint Integration

- Differential expression constraint
  \[ e_1(U) \geq n_1, \quad e_0(U) \geq n_0 \]

- Monotonicity: \( e_0 \) and \( e_1 \) decrease with extension of \( U \)

- Can be used for extra pruning without difficulty

<table>
<thead>
<tr>
<th>Module</th>
<th>( e_1 )</th>
<th>( e_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCDE</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ABCDEF</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Statistical Significance of a module

- $k$: The number of nodes in the module
- $\rho$: Density of the module
- $m_k(\rho)$: The number of modules of size $k$ with density at least $\rho$
- Probability of random selection making a denser module (p-value)

\[
p = \frac{m_k(\rho)}{\binom{n}{k}}
\]
Benchmarking in yeast complex discovery

- Combined interactions from CYGD-Mpact and DIP
- Interactions among 3559 nodes
- Confidence weights on edges due to (Jansen, 2003)
- Methods in comparison
  - Clique detection (Clique)
  - Clique Parcolation Method (CPM)
  - Markov Clustering
- Modules compared with MIPS complexes
Evolutionary Conserved Yeast Modules

- Use ortholog profiles (10 species, InParanoid)
- Density >= 50%, at least three orthologs
- 1917 modules in 30 minutes
- Recovered evolutionary conserved complexes
  - 20S proteasome
  - 19S/22S regulator
  - COPI vesicle coat complex
  - DNA polymerase I and II subunits
  - Translation initiation factor eIF2B complex
- They could not be recovered by simple DME due to low density
MIPS Complexes discovered by DME (Conserved in Evolution)
Phenotype-associated yeast modules

- Use growth phenotypic profiles (21 conditions, Dudley et al, 2005)
- Growth defect in at least one condition
- Each of the 13 highest ranking modules covers the large subunit of mitochondrial ribosome
  - Found additional protein, Mhr1
- Exactly recovered the nucleoplasmic THO complex (Hpr1, Mft1, Rlr1, Thp2)
  - Transcription elongation, hyperrecombination
  - Growth defect under ethanol
Mhr1: involved in homologous recombination of the mitochondrial genome
Human Settings

- Tissue-specific gene expression data (Su et al., 2004)
  - 79 different tissues
- Consistently expressed in 3 tissues, not in 10 tissues
- 7763 proteins, density >= 35%, 5 minutes
- 1021 maximal modules
- MIPS human complex database (Ruepp et al., to appear)
Human-expression result

- Around MCM complex, we found inter-complex relationships with ORC, CDC7, Toposome, PLK1 protein
- Module Uqcrc1, Uqcrc2, Uqcrb, Cyc1 (lg p = -13)
  - No overlap with MIPS
  - Ubiquinol-cytochrome c reductase complex
- SCF E3 ubiquitin ligase complex: Mark protein for degradation
  - 5 different modules with different tissue specificity
  - Peripheral proteins: Substrate recognition particles
  - Target proteins are selected in a tissue specific manner!
  - Natural Killer cells have all particles
High ranking modules around the MCM complex

Expressed in bone mallow cells
Not expressed in brain, liver, kidney etc.
Tissue Specific organization of the SCF ligase complex
Summary: Dense Module Enumeration

- Novel module enumeration algorithm based on reverse search
- Combination with other information sources
- Statistical significance of dense modules
- Successfully applied to
  - yeast/human protein interaction networks
Reference


Agenda 2 (Learning from Structured Data)

1. Preliminaries
2. Graph Clustering by EM
3. Graph Boosting
4. Regularization Paths in Graph Classification
5. Itemset Boosting for predicting HIV drug resistance
Part 1: Preliminaries
Clustering Graphs
Graph Regression

Training

\((?, -0.2)\)

\((?, 0.7)\)

\((?, -0.5)\)

Test

\((?, ?, \text{?})\)
Substructure Representation

- 0/1 vector of pattern indicators
- Huge dimensionality!
- Need Graph Mining for selecting features

(patterns)

(0, ..., 0, 1, 0, ..., 0, 1, 0, ...)
Graph Mining

- Frequent Substructure Mining
  - Enumerate all patterns occurred in at least m graphs

\[
S_{freq} = \{k \mid \sum_{i=1}^{n} x_{ik} \geq m\}. 
\]

\(x_{ik} \in \{0, 1\}\) : Indicator of pattern k in graph i

Support(k): # of occurrence of pattern k
Enumeration on Tree-shaped Search Space

- Each node has a pattern
- Generate nodes from the root:
  - Add an edge at each step
Tree Pruning

Anti-monotonicity:

$g \subseteq g' \Rightarrow \text{support}(g) \geq \text{support}(g')$

If support(g) < m, stop exploring!
Gspan (Yan and Han, 2002)

- Efficient Frequent Substructure Mining Method
- DFS Code
  - Efficient detection of isomorphic patterns
Depth First Search (DFS) Code

A labeled graph \( G \)

DFS Code Tree on \( G \)

- Non-minimum DFS code. Prune it.
Discriminative patterns

\[ w_i > 0: \text{positive class} \]
\[ w_i < 0: \text{negative class} \]

Weighted Substructure Mining

Patterns with large frequency difference

Not Anti-Monotonic: Use a bound

\[
S_w = \left\{ k \mid \left| \sum_{i=1}^{n} w_i (2x_{ik} - 1) \right| \geq \tau \right\},
\]
Multiclass version

- Multiple weight vectors
  - \( w_{\ell i} > 0 \) (graph \( i \) belongs to class \( \ell \))
  - \( w_{\ell i} < 0 \) (otherwise)

- Search patterns overrepresented in a class

\[
S_W = \left\{ k \mid \max_{\ell=1,\ldots,c} \left| \sum_{i=1}^{n} w_{\ell i} (2x_{i k} - 1) \right| - \tau_\ell \geq 0 \right\}.
\]
Basic Bound

• $x_{ij}$: Occurrence of pattern j

• If k is supergraph of pattern j,

$$\left| \sum_{i=1}^{n} w_{li} (2x_{ik} - 1) \right| \leq \gamma_{\ell}$$

$$\gamma_{\ell} = \max(\gamma_{\ell}^+, \gamma_{\ell}^-)$$

$$\gamma_{\ell}^+ = 2 \sum \{i | w_{li} \geq 0, x_{ij} = 1\} |w_{li}| - \sum_{i=1}^{n} w_{li}$$

$$\gamma_{\ell}^- = 2 \sum \{i | w_{li} < 0, x_{ij} = 1\} |w_{li}| + \sum_{i=1}^{n} w_{li}$$
Pruning Condition

Summarizing the bound for all classes,

$$\max_{\ell} \left| \sum_{i=1}^{n} w_{\ell i} (2x_{i\ell} - 1) \right| - \tau_{\ell} \leq \max_{\ell} (\gamma_{\ell} - \tau_{\ell})$$

If it is negative, the search tree can be pruned safely
Summary: Preliminaries

- Various graph learning problems
  - Supervised/Unsupervised

- Discovery of salient features by graph mining

- Actual speed depends on the data
  - Faster for:
    - Sparse graphs
    - Diverse kinds of labels
Part 2: EM-based clustering of graphs
EM-based graph clustering

Motivation

- Learning a mixture model in the feature space of patterns
- Basis for more complex probabilistic inference

L1 regularization & Graph Mining

E-step -> Mining -> M-step
Probabilistic Model

Binomial Mixture

\[ p(\mathbf{x}|\Theta) = \sum_{\ell=1}^{c} \alpha_\ell p_\ell(\mathbf{x}|\theta_\ell) \]

Each Component

\[ p_\ell(\mathbf{x}|\theta_\ell) = \prod_{k=1}^{d} \frac{\exp(\theta_{\ell k}x_k)}{1 + \exp(\theta_{\ell k})}. \]

$\mathbf{x}$: Feature vector of a graph (0 or 1)

$\alpha_\ell$: Mixing weight for cluster $\ell$

$\theta_\ell$: Parameter vector for cluster $\ell$
Ordinary EM algorithm

- Maximizing the log likelihood

$$\arg\max_{\Theta} \sum_{i=1}^{n} \log \sum_{\ell=1}^{c} \alpha_{\ell} p_{\ell}(x_i | \theta_{\ell}).$$

- E-step: Get posterior
  $$r_{\ell i} = p(y = \ell | x_i)$$

- M-step: Estimate $$\theta_{\ell}$$ using posterior probs.

- Both are computationally prohibitive (!)
Regularization

- **L1-Regularized log likelihood**
  \[ \frac{1}{n} \sum_{i=1}^{n} \log \sum_{\ell=1}^{c} \alpha_{\ell} p_{\ell}(x_i | \theta_{\ell}) - \lambda \sum_{\ell=1}^{c} \sum_{k=1}^{d} |\theta_{\ell k} - \theta_{0k}| \]

- **Baseline constant** \( \theta_0 \)
  - ML parameter estimate using single binomial distribution
  \[ \theta_{0k} = \log \eta_{0k} - \log(1 - \theta_{0k}) \quad \eta_{0k} = \frac{1}{n} \sum_{i} x_{ik} \]

- **In solution, most parameters exactly equal to constants**
E-step

Active pattern

\[ F = \{ k \mid \text{there exists } \ell \text{ such that } \theta_{\ell k} \neq \theta_{0k} \}. \]

E-step computed only with active patterns (computable!)

\[ p(y = \ell | x) = \frac{\alpha_{\ell} \prod_{k \in F} p_{\ell k}(x_k | \theta_{\ell k})}{\sum_{\ell} \alpha_{\ell} \prod_{k \in F} p_{\ell k}(x_k | \theta_{\ell k})}. \]
M-step

- Putative cluster assignment $r_{\ell i}$
- Each parameter is solved separately

\[
\min_{\theta_{\ell k}} \left( -\frac{1}{n} \sum_i r_{\ell i} \log p(x_{i|k}|\theta_{\ell k}) + \lambda |\theta_{\ell k} - \theta_{0k}| \right).
\]

Naïve way:
- solve it for all params and identify active patterns

Use graph mining to find active patterns
Solution

\[ \lambda_\ell = \frac{\lambda n}{\sum_j r_{\ell j}} \]

\[ \theta_{\ell k} = \begin{cases} 
\log \frac{\eta_{\ell k} - \lambda_\ell}{1 - (\eta_{\ell k} - \lambda_\ell)} & (\eta_{\ell k} \geq \eta_{0k} + \lambda_\ell) \\
\theta_{0k} & (\eta_{0k} - \lambda_\ell \leq \eta_{\ell k} \leq \eta_{0k} + \lambda_\ell) \\
\log \frac{\eta_{\ell k} + \lambda_\ell}{1 - (\eta_{\ell k} + \lambda_\ell)} & (\eta_{\ell k} \leq \eta_{0k} - \lambda_\ell). 
\end{cases} \]

**Occurrence probability in a cluster**

\[ \eta_{\ell k} = \sum_i r_{\ell i} x_{i k} / \sum_j r_{\ell j} \]

**Overall occurrence probability**

\[ \eta_{0k} = \frac{1}{n} \sum_i x_{i k} \]
\[
\theta_{\ell k} = \begin{cases} 
\log \frac{\eta_{\ell k} - \lambda_{\ell}}{1 - (\eta_{\ell k} - \lambda_{\ell})} & (\eta_{\ell k} \geq \eta_{0k} + \lambda_{\ell}). \\
\theta_{0k} & (\eta_{0k} - \lambda_{\ell} \leq \eta_{\ell k} \leq \eta_{0k} + \lambda_{\ell}) \\
\log \frac{\eta_{\ell k} + \lambda_{\ell}}{1 - (\eta_{\ell k} + \lambda_{\ell})} & (\eta_{\ell k} \leq \eta_{0k} - \lambda_{\ell}).
\end{cases}
\]
Important Observation

For active pattern $k$, the occurrence probability in a graph cluster is significantly different from the average

\[ \theta_{\ell k} \neq \theta_{0k} \iff |\eta_{\ell k} - \eta_{0k}| \geq \lambda_{\ell} \]
Mining for Active Patterns

Active pattern

\[ F = \{ k \mid \text{there exists } \ell \text{ such that } \theta_{\ell k} \neq \theta_{0k} \}. \]

Equivalently written as

\[ F = \{ k \mid \max_{\ell=1,\ldots,c} \left| \sum_{i=1}^{n} w_{\ell i} (2x_{i k} - 1) \right| - 2\lambda_{\ell} \geq 0 \}. \]

F can be found by graph mining! (multiclass)
Experiments: RNA graphs

- Stem as a node
- Secondary structure by RNAfold
- 0/1 Vertex label (self loop or not)
Clustering RNA graphs

Three Rfam families
- Intron GP I (Int, 30 graphs)
- SSU rRNA 5 (SSU, 50 graphs)
- RNase bact a (RNase, 50 graphs)

Three bipartition problems
- Results evaluated by ROC scores (Area under the ROC curve)
Examples of RNA Graphs
## ROC Scores

<table>
<thead>
<tr>
<th></th>
<th>Int-SSU</th>
<th>Int-RNase</th>
<th>SSU-RNase</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGK</td>
<td>0.748</td>
<td>0.531</td>
<td>0.878</td>
</tr>
<tr>
<td>Spec</td>
<td>0.550</td>
<td>0.573</td>
<td>0.848</td>
</tr>
<tr>
<td>$\lambda = 0.01$</td>
<td>0.824</td>
<td>0.921</td>
<td>0.863</td>
</tr>
<tr>
<td>$\lambda = 0.02$</td>
<td>0.821</td>
<td>0.920</td>
<td>0.862</td>
</tr>
<tr>
<td>$\lambda = 0.03$</td>
<td>0.825</td>
<td><strong>0.948</strong></td>
<td>0.843</td>
</tr>
<tr>
<td>$\lambda = 0.04$</td>
<td>0.832</td>
<td>0.947</td>
<td>0.825</td>
</tr>
<tr>
<td>$\lambda = 0.06$</td>
<td>0.831</td>
<td>0.941</td>
<td>0.782</td>
</tr>
<tr>
<td>$\lambda = 0.08$</td>
<td><strong>0.845</strong></td>
<td>0.941</td>
<td>0.787</td>
</tr>
<tr>
<td>$\lambda = 0.10$</td>
<td>0.815</td>
<td>0.927</td>
<td>0.786</td>
</tr>
</tbody>
</table>
## No of Patterns & Time

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>Int-SSU</th>
<th>Int-RNase</th>
<th>SSU-RNase</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda = 0.01 )</td>
<td>12505 (71s)</td>
<td>14366 (77s)</td>
<td>17934 (102s)</td>
</tr>
<tr>
<td>( \lambda = 0.02 )</td>
<td>12596 (75s)</td>
<td>10988 (65s)</td>
<td>11025 (76s)</td>
</tr>
<tr>
<td>( \lambda = 0.03 )</td>
<td>9799 (66s)</td>
<td>7632 (52s)</td>
<td>8875 (73s)</td>
</tr>
<tr>
<td>( \lambda = 0.04 )</td>
<td>6904 (57s)</td>
<td>5924 (45s)</td>
<td>6925 (67s)</td>
</tr>
<tr>
<td>( \lambda = 0.06 )</td>
<td>5093 (47s)</td>
<td>4305 (37s)</td>
<td>5230 (58s)</td>
</tr>
<tr>
<td>( \lambda = 0.08 )</td>
<td>4065 (42s)</td>
<td>3001 (32s)</td>
<td>3896 (50s)</td>
</tr>
<tr>
<td>( \lambda = 0.10 )</td>
<td>3245 (37s)</td>
<td>2074 (26s)</td>
<td>2923 (44s)</td>
</tr>
</tbody>
</table>
Found Patterns
Summary (graph EM)

- Substructure representation is better than paths
- Probabilistic inference helped by graph mining
- Extension to Dirichlet mixture model
  - Reported in Tsuda et al., SDM 2008
- Possible extension
  - Graph PCA, LFD, CCA
  - Semi-supervised learning
Part 3: Graph Boosting
Graph classification problem in chemoinformatics

- Known as SAR problem in chemical informatics
  - (Quantitative) Structure-Activity Analysis

- Given a graph, predict a class-label (+1 or -1)
  - Typically, features (descriptors) are given
  - e.g., Dragon Descriptors, JOELIB2
### SAR with conventional descriptors

<table>
<thead>
<tr>
<th>#atoms</th>
<th>#bonds</th>
<th>#rings</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Chemical Structure" /></td>
<td>22</td>
<td>25</td>
<td>+1</td>
</tr>
<tr>
<td><img src="image2.png" alt="Chemical Structure" /></td>
<td>20</td>
<td>21</td>
<td>+1</td>
</tr>
<tr>
<td><img src="image3.png" alt="Chemical Structure" /></td>
<td>23</td>
<td>24</td>
<td>+1</td>
</tr>
<tr>
<td><img src="image4.png" alt="Chemical Structure" /></td>
<td>11</td>
<td>11</td>
<td>-1</td>
</tr>
<tr>
<td><img src="image5.png" alt="Chemical Structure" /></td>
<td>21</td>
<td>22</td>
<td>-1</td>
</tr>
</tbody>
</table>
Motivation of Graph Boosting

- Descriptors are not always available
- New features by obtaining informative patterns (i.e., subgraphs)
- Greedy pattern discovery by Boosting + gSpan

Linear Programming (LP) Boosting
- Reduce the number of graph mining calls
- Faster than AdaBoost

Accurate prediction & interpretable results
Molecule as a labeled graph
## SAR with patterns

<table>
<thead>
<tr>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

\[ f = \alpha_1 \left( \begin{array}{c} \text{C} \\ \text{Cl} \end{array} \right) + \alpha_2 \left( \begin{array}{c} \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \end{array} \right) + \alpha_3 \left( \begin{array}{c} \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \\ \text{C} \end{array} \right) + \ldots \]
Sparse classification in a very high dimensional space

**G**: all possible patterns (intractably large)

|G|-dimensional feature vector $\mathbf{x}$ for a molecule

Linear Classifier

$$f(\mathbf{x}) = \sum_{j=1}^{d} \alpha_j x_j$$

Use L1 regularizer to have sparse $\alpha$

Select a tractable number of patterns
Problem formulation

\[
\min_{\alpha, \xi} \quad \|\alpha\|_1 + C \sum_{n=1}^{\ell} \xi_n \\
s.t. \quad y_n \alpha^\top x_n \geq 1 - \xi_n, \quad \xi_n \geq 0, \quad n = 1, \ldots, \ell
\]

Sum of hinge loss and L1 regularizer

\(\{x_n, y_n\}^\ell_{n=1} : \) Training examples

\(\xi: \) slack variables
Dual LP

Primal: Huge number of weight variables
Dual: Huge number of constraints

Dual problem

\[
\begin{align*}
\max_{u} & \quad \sum_{n=1}^{\ell} u_i \\
\text{s.t.} & \quad \sum_{n=1}^{\ell} u_n y_n x_{ni} \leq 1, \quad i = 1, \ldots, d \\
& \quad 0 \leq u_n \leq C, \quad n = 1, \ldots, \ell
\end{align*}
\]
Column Generation Algorithm for LP Boost (Demiriz et al., 2002)

- Start from the dual with no constraints
- Add the most violated constraint each time
- Guaranteed to converge

Constraint Matrix

Used Part
Finding the most violated constraint

Constraint for a pattern (shown again)

\[ \sum_{n=1}^{\ell} u_n y_n x_{ni} \leq 1 \]

Finding the most violated one

\[ \text{argmax}_i \sum_{n=1}^{\ell} u_n y_n x_{ni} \]

Searched by weighted substructure mining
Algorithm Overview

- **Iteration**
  - Find a new pattern by graph mining with weight $u$
  - If all constraints are satisfied, break
  - Add a new constraint
  - Update $u$ by solving the dual problem

- **Return**
  - Convert dual solution to obtain primal solution $\alpha$
## Experimental Settings

### Classification and Regression Datasets

<table>
<thead>
<tr>
<th></th>
<th># data</th>
<th># positives</th>
<th># negatives</th>
<th>avg. atoms</th>
<th>avg. bonds</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPDB</td>
<td>684</td>
<td>341</td>
<td>343</td>
<td>14.1</td>
<td>14.6</td>
</tr>
<tr>
<td>CAS</td>
<td>4337</td>
<td>2401</td>
<td>1936</td>
<td>29.9</td>
<td>30.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># data</th>
<th>avg. atoms</th>
<th>avg. bonds</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>146</td>
<td>19.5</td>
<td>21.1</td>
</tr>
<tr>
<td>ER</td>
<td>131</td>
<td>19.2</td>
<td>20.7</td>
</tr>
<tr>
<td>ES</td>
<td>59</td>
<td>18.2</td>
<td>19.7</td>
</tr>
</tbody>
</table>
Table 2 Classification performance obtained by 10-fold cross validation in two datasets measured by the accuracy (ACC) and the area under the ROC curve (AUC). We obtained the results of MGK and gBoost from our implementations, but the other results are quoted from literature. The best results are highlighted in bold fonts.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>-</td>
<td>78</td>
<td>75.96</td>
<td>76.5</td>
<td>78.8</td>
</tr>
<tr>
<td>AUC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.756</td>
<td>0.854</td>
</tr>
<tr>
<td>CAS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>79</td>
<td>-</td>
<td>80.14</td>
<td>77.1</td>
<td>82.5</td>
</tr>
<tr>
<td>AUC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.763</td>
<td>0.889</td>
</tr>
</tbody>
</table>
Regression Results

Table 3  Regression performance obtained by leave-one-out cross validation in three assays from the EDKB evaluated by mean absolute error (MAE), root mean squared error (RMSE), and $Q^2$. Note that for MAE and RMSE, lower values indicate better prediction, which is vice versa for $Q^2$. We obtained the results of MGK and gBoost from our implementations, but the other results are quoted from literature. The best results are highlighted in bold fonts.

<table>
<thead>
<tr>
<th>Measure</th>
<th>CoMFA [14,37]</th>
<th>MGK [18]</th>
<th>gBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td>-</td>
<td>0.229</td>
<td>0.176</td>
</tr>
<tr>
<td>RMSE</td>
<td>-</td>
<td>0.335</td>
<td>0.232</td>
</tr>
<tr>
<td>$Q^2$</td>
<td>0.571</td>
<td>0.346</td>
<td>0.682</td>
</tr>
<tr>
<td>ER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td>-</td>
<td>0.320</td>
<td>0.307</td>
</tr>
<tr>
<td>RMSE</td>
<td>-</td>
<td>0.427</td>
<td>0.393</td>
</tr>
<tr>
<td>$Q^2$</td>
<td>0.660</td>
<td>0.267</td>
<td>0.378</td>
</tr>
<tr>
<td>ES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td>-</td>
<td>0.322</td>
<td>0.249</td>
</tr>
<tr>
<td>RMSE</td>
<td>-</td>
<td>0.413</td>
<td>0.362</td>
</tr>
<tr>
<td>$Q^2$</td>
<td>-</td>
<td>0.522</td>
<td>0.632</td>
</tr>
</tbody>
</table>
Extracted patterns from CPDB

![Chemical structures and patterns extracted from CPDB.](image)
Memory Usage

![Graph showing memory usage with two lines, one labeled Naive and the other labeled Progressive. The y-axis represents tree size with values from $10^0$ to $10^7$, and the x-axis represents maxpat with values from 0 to 16.](image)
Runtime

![Runtime Graph](image-url)
Comparison with AdaBoost
Summary (Graph Boosting)

- Graph Boosting simultaneously generate patterns and learn their weights
- Finite convergence by column generation
- Interpretable by chemists.
- Flexible constraints and speed-up by LP.
Part 4: Entire Regularization Path
Overview

Entire regularization paths
- LARS-LASSO (Efron et al., 2004), L1SVM
- Forward selection of features
- Trace the solution trajectory of L1-regularized learning

Path following algorithm for graph data
- Feature search -> pattern search
- Branch-and-bound algorithm
- DFS code tree, New Bound
Path Following Algorithms

- **LASSO regression**
  \[ \beta(\lambda) = \arg\min_{\beta} L(y, X\beta) + \lambda \|\beta\|_1. \]

- Follow the complete trajectory of \( \beta(\lambda) \)
  - \( \lambda \): Infinity to Zero

- **Active feature set** \( A \)
  - Features corresponding to nonzero weights
Piecewise Linear Path

At a turning point,

- A new feature included into the active set,
or
- An existing feature excluded from the active set
Practical Merit of Path Following

- Cross validation by grid search
  - Has to solve QP many times
  - Especially time-consuming for graph data

- Path following does not include QP
- Determine the CV-optimal regularization parameter in the finest precision
Pseudo code of path following

Set initial point $\beta$ and direction $\gamma$

Do

- $d_1 =$ Step size if next event is inclusion
- $d_2 =$ Step size if next event is exclusion
- $d = \min(d_1,d_2)$
- $\beta = \beta + d\gamma$
- Update the active feature set
- Set the next direction $\gamma$

Until all features are included
Feature space of patterns

- Graph training data: $\mathcal{G} = \{G_i\}_{i=1}^n$
- Set of all subgraphs (patterns): $\mathcal{T}$
- Each graph is represented as:

$$\mathbf{x}_i = (x_{it})_{t \in \mathcal{T}}, \quad x_{it} = I(t \subseteq G_i)$$

$$(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots)$$
Main Search problem

Step size if pattern $t$ is included next

$$d_t = \min \left\{ \frac{\rho_0 - \sum_i w_i x_{it}}{\eta_0 - \sum_i v_i x_{it}}, \frac{\rho_0 + \sum_i w_i x_{it}}{\eta_0 + \sum_i v_i x_{it}} \right\}.$$ 

$w_i, v_i, \rho_0, \eta_0$ : constants computed from active set

Find pattern $t \in T$ that minimizes $d_t$
Tree-shaped Search Space

- Each node has a pattern
- Generate nodes from the root:
  - Add an edge at each step
Tree Pruning

If it is guaranteed that the optimal pattern is not in the downstream, the search tree can be pruned.
Theorem (Pruning condition)

- Traversed up to pattern $t$
- $d^*_t$: Minimum value so far
- No better pattern in the downstream, if

$$b_w + d^*_t b_v < |\rho_0| - d^*_t |\eta_0|.$$ 

where

$$b_w = \max \left\{ \sum_{w_i < 0} |w_i| x_{it}, \sum_{w_i > 0} |w_i| x_{it} \right\}.$$
Initial Experiments

- EDKB Estrogen receptor database
  - 131 training graphs (chemical compounds)

- Computation Time: 4 sec/search
  - Pattern size limitation: 10 edges
Solution Path
Events
Summary: Regularization Path

- Path following implemented for graph data
- Pattern search by graph mining
- Classification: To do
- Combination with item set mining
Part 5: Itemset Boosting for predicting HIV drug resistance
Life cycle of an HIV Virus

Drug Target
Approved HIV Drugs

- 8 Protease inhibitors (PI)
- 8 Nucleoside/nucleotide reverse transcriptase inhibitors (NRTI)
- 3 Non-nucleoside reverse transcriptase inhibitors (NNRTI)
- 1 Fusion inhibitor
Drug resistance of HIV

- Exposure to a drug causes mutations in HIV’s genes
- As a result, HIV gains resistance against the drug
- Cost of identifying the genotypes of HIV in a patient is relatively cheap
- Predict the drug resistance from HIV’s genotypes!
  - Effective Pharmacotherapy for individuals
Drug resistance prediction problem as regression

- **Input:** Set of mutations in a target protein
  - 41L: Amino acid at position 41 changed to L
- **Output:** Resistance against a drug (fold change)

\[(40F, 41L, 43E, 210W, 211K, 215Y) \rightarrow 0.8\]
\[(43Q, 75M, 122E, 210W, 211K) \rightarrow 12.8\]
\[(75I, 77L, 116Y, 151M, 184V) \rightarrow ?\]
Simple vs Complex Genotypic Features

• Simple genotypic features
  \[(0, 1, 0, 1, 0, 0, 0, 1, \ldots)\]
  41L, 62V, F116Y

• Complex genotypic features
  \[(0, 1, 0, 1, 0, 0, 0, 1, \ldots)\]
  77L, 116Y, 103N, 210W, 215Y
Linear Regression on Simple Genotypic Features (Rhee et al., PNAS 2006)

- Mutation associations not discovered
Complex Genotypic Features

- 0/1 vector of pattern indicators
- Huge dimensionality!
- Need itemset mining for selecting features
- Selection of salient features

\[(0, 1, 0, 1, 0, 0, 0, 1, \ldots)\]

Patterns:
- 77L, 116Y
- 103N, 210W, 215Y
- 77N, 116W, 215Y
Other methods

- Nonlinear SVM
  - Not interpretable
  - High accuracy
- Decision trees
  - Interpretable
  - Low accuracy
Motivation of Itemset Boosting

- Impossible to maintain all complex features
- Greedy feature discovery by Boosting + itemset mining
- Quadratic Programming (QP) Boosting
  - Reduce the number of itemset mining calls
  - Faster than AdaBoost
- Accurate prediction & interpretable results
Sparse classification in a very high dimensional space

G: all possible patterns (intractably large)

|G|-dimensional feature vector \( \mathbf{x} \)

Linear Classifier

\[
f(\mathbf{x}) = \sum_{j=1}^{d} \alpha_j x_j
\]

Use L1 regularizer to have sparse \( \alpha \) (LASSO)

Select a tractable number of patterns
Problem formulation: Quadratic Programming

\[
\min_{\alpha, \xi} \quad \|\alpha\|_1 + \frac{C}{2} \sum_{n=1}^{\ell} \xi_n^2
\]

s.t. \quad |y_n - \alpha^\top x_n| \leq \xi_n, \quad \xi_n \geq 0, \quad n = 1, \ldots, \ell

Sum of squared loss and L1 regularizer

\{x_n, y_n\}_{n=1}^{\ell} : \text{Training examples}

\xi: \text{slack variables}
Dual QP

Primal: Huge number of weight variables
Dual: Huge number of constraints

\[
\min_u \quad \frac{1}{2C} \sum_{n=1}^\ell u_n^2 - \sum_{n=1}^\ell y_n u_n \\
\text{s.t.} \quad -1 \leq \sum_{n=1}^\ell u_n x_{ni} \leq 1, \quad i = 1, \ldots, d \\
\sum_{n=1}^\ell u_i = 0
\]
Column Generation Algorithm for QP Boost (Demiriz et al., 2002)

- Start from the dual with no constraints
- Add the most violated constraint each time
- Guaranteed to converge
Finding the most violated constraint

Constraint for a pattern (shown again)

\[ \left| \sum_{n=1}^{\ell} u_n x_{ni} \right| \leq 1 \]

Finding the most violated one

\[ \text{argmax}_i \left| \sum_{n=1}^{\ell} u_n x_{ni} \right| \]

Searched by weighted itemset mining
Algorithm Overview

Iteration

- Find a new pattern by graph mining with weight $u$
- If all constraints are satisfied, break
- Add a new constraint
- Update $u$ by solving the dual problem

Return

- Convert dual solution to obtain primal solution $\alpha$
Experimental settings

• Three classes of drugs
  – NRTI (Nucleotide Reverse Transcriptase Inhibitors)
  – PI (Protease Inhibitors)
  – NNRTI (Nonnucleotide Reverse Transcriptase Inhibitors)

• 5fold cross validation
  – Linear SVM, Ridge regression, Lars
  – Nonlinear SVM, iBoost
# Regression results

<table>
<thead>
<tr>
<th>Drug</th>
<th># isolates</th>
<th>Linear Methods</th>
<th></th>
<th></th>
<th>Nonlinear Methods</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>iBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRTI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamivudine (3TC)</td>
<td>633</td>
<td>0.913 0.753 0.93</td>
<td>0.927 0.876 0.306 0.934 0.608</td>
<td>0.940</td>
<td>Abacavir (ABC)</td>
<td>628</td>
<td>0.731 0.585 0.79</td>
<td>0.772 0.720 0.256 0.745 0.614</td>
<td>0.801</td>
<td>Zidovudine (AZT)</td>
<td>630</td>
<td>0.751 0.72 0.65</td>
<td>0.797 0.760 0.240 0.754 0.515</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.737 0.647 0.708</td>
<td>0.744 0.692 0.211 0.747 0.502</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNRTI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delavirdine (DLV)</td>
<td>732</td>
<td>0.799 0.794 0.79</td>
<td>0.729 0.684 0.189 0.802 0.323</td>
<td>0.771</td>
<td>Efavirenz (EFV)</td>
<td>734</td>
<td>0.793 0.772 0.85</td>
<td>0.730 0.640 0.170 0.797 0.205</td>
<td>0.771</td>
<td>Nevirapine (NVP)</td>
<td>746</td>
<td>0.757 0.719 0.79</td>
<td>0.704 0.592 0.166 0.765 0.181</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.783 0.762 0.810</td>
<td>0.721 0.639 0.175 0.788 0.236</td>
<td>0.774</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amprenavir (APV)</td>
<td>768</td>
<td>0.819 0.749 0.81</td>
<td>0.756 0.697 0.483 0.82 0.576</td>
<td>0.802</td>
<td>Atazanavir (ATV)</td>
<td>329</td>
<td>0.724 0.594 0.76</td>
<td>0.731 0.654 0.297 0.739 0.450</td>
<td>0.701</td>
<td>Indinavir (IDV)</td>
<td>827</td>
<td>0.830 0.710 0.81</td>
<td>0.819 0.775 0.574 0.844 0.710</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.824 0.712 0.824</td>
<td>0.798 0.738 0.504 0.836 0.631</td>
<td>0.810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accuracy: \[ r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n} \sum_{i=1}^{n} y_i)^2}. \]
Accuracy Summary

• NRTIs: iBoost performed best
• PIs:
  – Nonlinear methods were better than linear
  – SVMs were slightly better (non-significant)
• NNRTIs
  – Linear methods were better
  – Combination is not necessary
NRTI Drugs
Known mutation associations in RT

Red: Thymidine-associated Mutations (TAM)

Blue: Q151M Complex
- 75I, 77L, 116Y, 151M, 65R, 74V, 184I/V
PI Drugs

APV (53.2%)

ATV (59.0%)

IDV (42.7%)

LPV (45.8%)

NFV (51.4%)

RTV (54.6%)

SQV (48.5%)
NNRTI Drugs
(almost no combination found)
Computation Time of iBoost

- Training time for 3TC
- 507 isolates with 371 mutations on average
- QP time longer than mining time
Summary (HIV)

• Itemset Boosting for finding mutation associations
• Good accuracy for NRTIs
• Our complex features re-discover known mutation clusters
• Broad applications
  – Multiple SNP analysis, RNAi efficacy prediction
  – Motif combination, Flu mutation analysis
  – P53 mutation analysis
Reference

Agenda 3 (Kernel)

1. Kernel Method Revisited
2. Marginalized Kernels (Fisher Kernels)
3. Marginalized Graph Kernels
4. Weisfeiler-Lehman kernels
5. Reaction Graph kernels
6. Concluding Remark
Part 1: Kernel Method Revisited
In Kernel-based learning algorithms, problem solving is now decoupled into:

- A general purpose learning algorithm (e.g. SVM, PCA, …) – Often linear algorithm
- A problem specific kernel
Current Synthesis

Modularity and re-usability

- Same kernel, different learning algorithms
- Different kernels, same learning algorithms
Kernel Methods: intuitive idea

- Find a mapping $\phi$ such that, in the new space, problem solving is linear.
- Kernel represents the similarity between two objects, defined as the dot-product in this new vector space.
- But the mapping is left implicit.
- Easy generalization of a lot of dot-product-based learning algorithms.
Kernel Methods: the mapping

Original Space $\phi$ Feature (Vector) Space
A kernel $k(x,y)$
- is a similarity measure
- defined by an implicit mapping $\phi$,
- such that: $k(x,y) = \phi(x) \cdot \phi(y)$

This similarity measure implies:
- Invariance or other a priori knowledge
- The class of functions the solution is taken from
- Possibly infinite dimension (hypothesis space for learning)
- … but still computational efficiency when computing $k(x,y)$
Kernel Trick

- Generalizes (nonlinearly) algorithms in clustering, classification, density estimation.
  - When these algorithms are dot-product based, by replacing the dot product \( x \cdot y \) by \( k(x,y) = \phi(x) \cdot \phi(y) \)
  - When these algorithms are distance-based, by replacing \( d(x,y) \) by \( k(x,x) + k(y,y) - 2k(x,y) \)

- Freedom of choosing \( \phi \) implies a large variety of learning algorithms
Valid Kernels

Theorem: \( k(x,y) \) is a valid kernel if \( k \) is positive definite and symmetric (Mercer Kernel)

- A function is P.D. if
  \[
  \int K(x, y) f(x) f(y) dx dy \geq 0 \quad \forall f \in L_2
  \]

- In other words, the Gram matrix \( K \) (whose elements are \( k(x_i, x_j) \)) must be positive definite for all \( x_i, x_j \) of the input space

- One possible choice of \( \phi(x) \): \( k(\cdot, x) \) (maps a point \( x \) to a function \( k(\cdot, x) \rightarrow \text{feature space with infinite dimension!} \))
How to build new kernels

Kernel combinations, preserving validity:

\[ K(x,y) = \lambda K_1(x,y) + (1 - \lambda) K_2(x,y) \quad 0 \leq \lambda \leq 1 \]
\[ K(x,y) = a.K_1(x,y) \quad a > 0 \]
\[ K(x,y) = K_1(x,y).K_2(x,y) \]
\[ K(x,y) = f(x).f(y) \quad f \text{ is real-valued function} \]
\[ K(x,y) = K_3(\phi(x),\phi(y)) \]
\[ K(x,y) = x'Py \quad P \text{ symmetric definite positive} \]
\[ K(x,y) = \frac{K_1(x,y)}{\sqrt{K_1(x,x)} \sqrt{K_1(y,y)}} \]
Strategies of Design

Convolution Kernels: text is a recursively-defined data structure. How to build “global” kernels form local (atomic level) kernels?

Generative model-based kernels: the “topology” of the problem will be translated into a kernel function
Family of kernels

- **Kernels for biological sequences**
  - Spectrum kernel
  - Marginalized kernel
  - Profile kernel
  - Local alignment kernel

- **Tree Kernels**
  - Kernel for phylogenetic profiles
  - Kernel for natural language
  - Kernel for RNA sequences
Family of kernels

- **Kernels for nodes in a network**
  - Diffusion kernel
  - Locally constrained diffusion kernel

- **Graph Kernels**
  - Marginalized Graph Kernels
  - MGK without tottering
  - Acyclic Pattern Kernels
  - Shortest Path Kernel
  - Weisfeiler-Lehman Kernel
Weak points of kernel methods

- **Not Interpretable**
  - Not sure which features are used
  - -> Graph Mining, Boosting

- **Dense kernel matrices: Slow**
  - Take $O(n^3)$ time for manipulation
  - -> Semi-supervised learning
Part 2. Marginalized kernels
Biological Sequences: Classification Tasks

- **DNA sequences** (A,C,G,T)
  - Gene Finding, Splice Sites

- **RNA sequences** (A,C,G,U)
  - MicroRNA discovery, Classification into Rfam families

- **Amino Acid Sequences** (20 symbols)
  - Remote Homolog Detection, Fold recognition
Kernels for Sequences

Similarity between sequences of different lengths

ACGGTTCAA

ATATCGCGGGAA

Later combined with SVMs and other kernel methods
Count Kernel

Inner product between symbol counts

Extension: Spectrum kernels (Leslie et al., 2002)

- Counts the number of k-mers (k-grams) efficiently

Not good for sequences with frequent context change
- E.g., coding/non-coding regions in DNA
Hidden Markov Models for Estimating Context

- **Visible Variable** $\mathbf{x} = (x_1, \ldots, x_m)$: Symbol Sequence
- **Hidden Variable** $\mathbf{h} = (h_1, \ldots, h_m)$: Context
- HMM can estimate the posterior probability of hidden variables $p(h|x)$ from data

\[ h: \quad 1 \quad 2 \quad 2 \quad 1 \quad 2 \quad 2 \quad 1 \quad 2 \quad 2 \]
\[ x: \quad A \quad C \quad G \quad G \quad T \quad T \quad C \quad A \quad A \]
Marginalized kernels

- Design a joint kernel $K_z(z, z')$ for combined $z = (x, h)$
  - Hidden variable is not usually available
  - Take expectation with respect to the hidden variable

- The marginalized kernel for visible variables

$$K(x, x') = \sum_{h \in \mathcal{H}} \sum_{h' \in \mathcal{H}} p(h|x)p(h'|x')K_z(z, z')$$
Designing a joint kernel for sequences

- Symbols are counted separately in each context

  \[ h: \ 1 \ 2 \ 2 \ 1 \ 2 \ 2 \ 1 \ 2 \ 2 \ \quad (A,1) = 1 \quad (C,1) = 1 \quad (G,1) = 1 \quad (T,1) = 0 \]
  \[ x: \ A \ C \ G \ G \ T \ T \ C \ A \ A \ \quad (A,2) = 2 \quad (C,2) = 1 \quad (G,2) = 1 \quad (T,2) = 2 \]

- \[ c_{k\ell}(z) \] : count of a combined symbol \((k,l)\)

- Joint kernel: count kernel with context information

  \[ K_z(z, z') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} c_{k\ell}(z)c_{k\ell}(z') \]
Marginalization of the joint kernel

\[ K_z(z, z') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} c_{k\ell}(z) c_{k\ell}(z') \]

Marginalized count kernel

\[ K(x, x') = \sum_h \sum_{h'} p(h|x) p(h'|x') K_z(z, z') \]

\[ = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \gamma_{k\ell}(x) \gamma_{k\ell}(x') \]

\( \gamma_{k\ell} \) is a marginalized count \( \gamma_{k\ell}(x) = \sum_h p(h|x) c_{k\ell}(z) \)
Computing Marginalized Counts from HMM

Marginalized count is described as

$$\gamma_{kl}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{h_i=1}^{n_h} p(h_i|\mathbf{x}) I(x_i = k, h_i = \ell).$$

Posterior probability of $i$-th hidden variable is efficiently computed as

$$p(h_i = k|\mathbf{x}) = \frac{f_k(i)b_k(i)}{p(\mathbf{x})}$$

$f_k(i)$: forward variable, $b_k(i)$: backward variable
2nd order marginalized count kernel

If adjacent relations between symbols have essential meanings, the count kernel is obviously not sufficient.

2nd order marginalized count kernel

- 4 neighboring symbols (i.e. 2 visible and 2 hidden) are combined and counted

h: 1 2 2 1 2 2 1 2 2
x: A C G G T T T C A A
Fisher Kernel

Probabilistic mode \( p(x | \hat{\theta}) \), \( x \in \mathcal{X} \)

\( \hat{\theta} \in \mathbb{R}^r \) is a parameter vector obtained from training samples.

Fisher Kernel

- Mapping to a feature vector (Fisher score vector)

\[
s(x, \hat{\theta}) = \left( \frac{\partial \log p(x | \hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \log p(x | \hat{\theta})}{\partial \theta_p} \right)
\]

- Inner product of Fisher scores

\[
K_f(x, x') = s(x, \hat{\theta})^\top Z^{-1}(\hat{\theta}) s(x', \hat{\theta})
\]

Z: Fisher information matrix
Fisher Kernel from HMM

- Derive FK from HMM (Jaakkola et al. 2000)
  - Derivative for emission probabilities only
  - No Fisher information matrix
- FK from HMM is a special case of marginalized kernels
  - Counts are centralized and weighted

\[
K_f(x, x') = \sum_h \sum_{h'} p(h|x)p(h'|x')K_{fz}(z, z')
\]

\[
K_{fz}(z, z') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \frac{1}{\epsilon_{\ell k}} (c_{k \ell}(z) - \bar{c}_{k \ell}(z))(c_{k \ell}(z') - \bar{c}_{k \ell}(z'))
\]
Difference between FK and Marginalized Kernels

FK: Probabilistic model determines the joint kernel and the posterior probabilities

MK: You can determine them separately
   ■ More flexible design!
Protein clustering experiment

- 84 proteins containing five classes
  - gyrB proteins from five bacteria species

Clustering methods
- HMM + {FK, MCK1, MCK2} + K-Means

Evaluation
- Adjusted Rand Index (ARI)
Kernel Matrices

Ideal

FK

MCK1

MCK2
Clustering Evaluation

![Graph showing Clustering Evaluation](image-url)

- **ARI** values are plotted against the number of HMM States.
- Three different algorithms are compared: FK, MCK1, and MCK2.
- FK shows a relatively stable performance across different states.
- MCK1 and MCK2 exhibit a more dynamic response, with MCK1 peaking at around 6 states and MCK2 showing a decline in performance with increasing states.
Applications since then..

- Marginalized Graph Kernels (Kashima et al., ICML 2003)
- Sensor networks (Nyugen et al., ICML 2004)
- Labeling of structured data (Kashima et al., ICML 2004)
- Robotics (Shimosaka et al., ICRA 2005)
- Kernels for Promoter Regions (Vert et al., NIPS 2005)
- Web data (Zhao et al., WWW 2006)
Summary (Marginalized Kernels)

- General Framework for using generative model for defining kernels
- Fisher kernel as a special case
- Broad applications
Part 3 Marginalized Graph Kernels
Motivations for graph analysis

- Existing methods assume "tables"

- Structured data beyond this framework → New methods for analysis
Graphs...
Graph Structures in Biology

DNA Sequence

3' A C G C

RNA

UA
CG
CG
U U U U

Compounds

H C C O H

Texts in literature

Amitriptyline inhibits adenosine uptake
Marginalized Graph Kernels

(Kashima, Tsuda, Inokuchi, ICML 2003)

- Going to define the kernel function $K(G, G')$
- Both vertex and edges are labeled
Label path

- Sequence of vertex and edge labels
  \[ h = (A, e, A, d, D, a, B, c, D) \]
- Generated by random walking
- Uniform initial, transition, terminal probabilities
## Path-probability vector

| Label path $h$ | Probability $p(h|G)$ |
|----------------|----------------------|
| AaA            | 0.001                |
|                |                      |
| AcDbE          | 0.0000003            |
|                |                      |
| AeAdDaBcD      | 0.000000007          |
|                |                      |
Kernel definition

Kernels for paths

\[
K(h, h') = \begin{cases} 
0 & (|h| \neq |h'|) \\
\prod_{i=1}^{\ell} k_v(h_i, h'_i) & (|h| = |h'|)
\end{cases}
\]

Take expectation over all possible paths!

Marginalized kernels for graphs

\[
K(G, G') = \sum_{h} \sum_{h'} p(h|G)p(h'|G')K(h, h')
\]
Computation

- $S_v(x)$: Set of paths ending at $v$
- $K_v$: Kernel computed from the paths ending at $(v, v')$

\[ K_v(v, v') = \sum_{s \in S_v(x)} \sum_{s' \in S_{v'}(x')} \lambda^{||s||} \lambda^{||s'||} K_S(s, s') \]

- $K_v$ is written recursively

\[ K_v(v, v') = \lambda^2 I(v = v')(1 + \sum_{\tilde{v} \in A(v)} \sum_{\tilde{v}' \in A(v')} \lambda^2 K_v(\tilde{v}, \tilde{v}')) \]

- Kernel computed by solving linear equations (polynomial time)

Transition probability: $\lambda$
Initial and terminal: omitted
Graph Kernel Applications

- Chemical Compounds (Mahe et al., 2005)
- Protein 3D structures (Borgwardt et al, 2005)
- RNA graphs (Karklin et al., 2005)
- Pedestrian detection
- Signal Processing
Predicting Mutagenicity

MUTAG benchmark dataset

- Mutation of Salmonella typhimurium
- 125 positive data (effective for mutations)
- 63 negative data (not effective for mutations)


<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lin.Reg</td>
<td>66.7%</td>
</tr>
<tr>
<td>Lin.Reg+</td>
<td>71.8%</td>
</tr>
<tr>
<td>DT</td>
<td>83.3%</td>
</tr>
<tr>
<td>NN</td>
<td>69.0%</td>
</tr>
<tr>
<td>Progol1</td>
<td>85.7%</td>
</tr>
<tr>
<td>Progol2</td>
<td>83.3%</td>
</tr>
</tbody>
</table>

Note: Lin.Reg (Linear Regression), DT (Decision Tree), NN (Neural Network), and Progol1/2 (Inductive Logic Programming): ref 19.

Classification of Protein 3D structures

- Graphs for protein 3D structures
  - Node: Secondary structure elements
  - Edge: Distance of two elements
- Calculate the similarity by graph kernels

Borgwardt et al. “Protein function prediction via graph kernels”, ISMB2005

protein data → secondary structure elements → sequence → structure
Classification of proteins:
Accuracy

<table>
<thead>
<tr>
<th>Kernel type</th>
<th>Accuracy</th>
<th>St. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector kernel</td>
<td>76.86</td>
<td>1.23</td>
</tr>
<tr>
<td>Optimized vector kernel</td>
<td>80.17</td>
<td>1.24</td>
</tr>
<tr>
<td>Graph kernel</td>
<td>77.30</td>
<td>1.20</td>
</tr>
<tr>
<td>Graph kernel without structure</td>
<td>72.33</td>
<td>5.32</td>
</tr>
<tr>
<td>Graph kernel with global info</td>
<td>84.04</td>
<td>3.33</td>
</tr>
<tr>
<td>DALI classifier</td>
<td>75.07</td>
<td>4.58</td>
</tr>
</tbody>
</table>

Table 1. Accuracy of prediction of functional class of enzymes and non-enzymes in 10-fold cross-validation with C-SVM. The first two results are the results obtained by Dobson and Doig (2003). ”Graph kernel” is our protein kernel defined as in Section 2.3, ”Graph kernel without structure” is the same kernel but on protein models without structural edges, ”Graph kernel with global info” is our protein graph kernel plus additional global node labels. ”DALI classifier” is a Nearest Neighbor Classifier on DALI Z-scores.
Pedestrian detection in images
(F. Suard et al., 2005)
Classifying RNA graphs (Y. Karklin et al., 2005)
Strong points of MGK

Polynomial time computation $O(n^3)$

Positive definite kernel

- Support Vector Machines
- Kernel PCA
- Kernel CCA
- And so on...
Drawbacks of graph kernels

- **Global similarity measure**
  - Fails to capture subtle differences
  - Long paths suppressed
- **Results not interpretable**
- **Structural features ignored (e.g. loops)**
  - No labels -> kernel always 1
Part 4. Weisfeiler Lehman kernel
Convert a graph into a set of words

i) Make a label set of adjacent vertices  
   ex) \{E,A,D\}

ii) Sort  
    ex) A,D,E

iii) Add the vertex label as a prefix  
     ex) B,A,D,E

iv) Map the label sequence to a unique value  
    ex) B,A,D,E→R

v) Assign the value as the new vertex label

Bag-of-words  
{A,B,D,E,…,R,…}
Given labeled graphs $G$ and $G'$

1st iteration
Result of steps 1 and 2: multiset-label determination and sorting

1st iteration
Result of step 3: label compression

1st iteration
Result of step 4: relabeling
<table>
<thead>
<tr>
<th>Dataset</th>
<th>MUTAG</th>
<th>NCI1</th>
<th>NCI109</th>
<th>D &amp; D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum # nodes</td>
<td>28</td>
<td>111</td>
<td>111</td>
<td>5748</td>
</tr>
<tr>
<td>Average # nodes</td>
<td>17.93</td>
<td>29.87</td>
<td>29.68</td>
<td>284.32</td>
</tr>
<tr>
<td># labels</td>
<td>7</td>
<td>37</td>
<td>54</td>
<td>89</td>
</tr>
<tr>
<td>Number of graphs</td>
<td>188</td>
<td>100</td>
<td>4110</td>
<td>100</td>
</tr>
<tr>
<td>Weisfeiler-Lehman</td>
<td>6”</td>
<td>5”</td>
<td>7’20”</td>
<td>5”</td>
</tr>
<tr>
<td>Ramon &amp; Gärtnner</td>
<td>40’6”</td>
<td>25’9”</td>
<td>29 days*</td>
<td>26’40”</td>
</tr>
<tr>
<td>Graphlet count</td>
<td>3”</td>
<td>2”</td>
<td>1’27”</td>
<td>2”</td>
</tr>
<tr>
<td>Random walk</td>
<td>12”</td>
<td>58’30”</td>
<td>68 days*</td>
<td>2h 9’41”</td>
</tr>
<tr>
<td>Shortest path</td>
<td>2”</td>
<td>3”</td>
<td>4’38”</td>
<td>3”</td>
</tr>
</tbody>
</table>

---: did not finish in 2 days, * = extrapolated.

Table 2: CPU runtime for kernel computation on graph classification benchmark datasets
Part 5. Reaction Graph Kernels
KEGG lysine degradation pathway
Missing enzymes in metabolic networks

• Many enzymatic reactions whose substrate and product are known, but the enzyme involved is unknown.

• Need to assign Enzymatic Classification numbers.
EC number

- EC (Enzymatic Classification) number is a hierarchical categorization of
  - Enzymes
  - Enzymatic reactions

EC 1.3.3.-

- class
- subclass
- subsubclass
Task

Given a pair of substrate and product as a query, find similar reactions in the database.

Query

Result of Retrieval

Similarity measure of reactions is necessary.
Reaction Graph

- Represent enzymatic reaction as reaction graph
  - Node: Molecules
  - Edge: chemical relation of molecules (main, leave, co-factor, transferase, ligase)

- Reaction graph kernel: Similarity measure of reaction graphs
  - Molecule = Graph of atoms
  - Reaction graph has ‘graph of graphs’ structure
  - Extension of existing walk-based graph kernel (Kashima et al., 2003)
Main Substrate + \( \text{H}_2\text{O} \) ➔ 2 Main Products

Main Substrate: \( \text{H}_3\text{C} \)

2 Main Products: \( \text{CH}_3 \), \( \text{OH} \), \( \text{OH} \), \( \text{CO} \)

C01479
C00001
C00729
C01456
Reaction graph kernels (RGK)

- Two-layered kernels on graphs of graphs
  - Node kernel = walk-based graph kernel of molecules
  - Edge kernel = delta kernel of labels
- “main”, “leave”, “cofactor”, “transferase”, “ligase”
Simplified Settings

• Query might not come in the complete form
• Remove some edges in the database entries

RPAIR

Use only reactant edges (main, leave)

main-only

Use “main” only
Automatic classification of enzymatic reactions in KEGG

• KEGG/REACTION database
  • 4610 reactions with known EC number
  • 6 classes, 50 subclasses, 124 subsubclasses

• Construct nearest neighbor classifier based on the reaction graph kernel
  • Three different levels: class, subclass, subsubclass
  • Three kernel versions: full, RPAIR, main-only
• Measure leave-one-out classification error
• As expected, classification is easier for upper categories, but difficult for lower categories such as subsubclass.

• The order of accuracy (full-edge > RPAIR > main-pair) suggests that detailed edge information contributes to further accuracy.
Predicting unannotated reactions in plant secondary metabolism

- KEGG pathway “Biosynthesis of Secondary Metabolites”
- Out of unannotated 56 reactions, we have manually assigned ECs of 36 reactions under chemists’ guidance
- RGK’s accuracy was better than e-zyrne. (50% improvement for the top candidate)
Case 1: EC 3.1.1

Structures of C02046 and C01479 are almost same except structural isomerism. A hydrolysis occur at a carboxylic-ester bond.

<table>
<thead>
<tr>
<th>method</th>
<th>rank1</th>
<th>rank2</th>
<th>rank3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGK</td>
<td>3.1.1</td>
<td>1.14.11</td>
<td>1.14.11</td>
</tr>
<tr>
<td>e-zyme</td>
<td>6.1.1</td>
<td>3.1.1</td>
<td>NA</td>
</tr>
</tbody>
</table>
After removing a methyl group of C06175 (2.1.1), C01735 will be produced by oxidation of CH-OH group (1.1.1). In the second reactions, enzymes usually use NAD+/NADP+ as an acceptor.

<table>
<thead>
<tr>
<th>method</th>
<th>rank1</th>
<th>rank2</th>
<th>rank3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGK</td>
<td>1.1.1</td>
<td>1.1.1</td>
<td>1.1.1</td>
</tr>
<tr>
<td>e-enzyme</td>
<td>2.1.1</td>
<td>1.13.12</td>
<td>1.14.14</td>
</tr>
</tbody>
</table>
This reaction is thought to be a set of 5 reactions by analogy with the pathway from C00423 => C01772 => C05158 => C05839 => C05838, and to C05851. The last reaction is spontaneous and not enzymatic.
A difficult case

No annotation

A very similar reaction (below) is found by manual inspection

Multi-step reaction is difficult to analyze. We don’t know how many steps are hidden between given substrate and product.
Concluding Remarks: New Frontier

- Developing Algorithms for Learning from Graphs
- Taming Combinatorial Explosion
  - Recursive Fixed Point Iteration: Graph Kernels
  - Statistical Pruning in Search Tree: Graph Boosting
  - Hashing to a set of words: WL Kernel
- New ideas are necessary to get beyond the current level of speed and prediction accuracy
- Deeper integration of learning and mining

THANK YOU VERY MUCH!!
Reference


