The skew spectrum of graphs

Risi Kondor
Gatsby Unit, UCL

with

Karsten Borgwardt
University of Cambridge
A = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
$q(A)$ is a graph invariant if it is invariant to relabeling.
poly(n) time computable complete set of invariants

Graph isomorphism problem

efficiently computable set of invariant features

Graph kernels, etc.
\[ f(\sigma)f \coloneqq [S_\sigma(n), R(n-1)] \]
\[
\begin{align*}
  f(\begin{array}{cccc}7 & 6 & 7 & 6 \\ ? & ? & ? & ? \end{array}) &= [A]_{1,2} \\
  f(\begin{array}{cccc}7 & ? & 6 & ? \\ ? & ? & ? & ? \end{array}) &= [A]_{1,3} \\
  f(\begin{array}{cccc}7 & ? & ? & 6 \\ ? & ? & ? & ? \end{array}) &= [A]_{1,4} \\
  f(\begin{array}{cccc}7 & ? & ? & 6 \\ ? & ? & ? & ? \end{array}) &= [A]_{2,3} \\
  f(\begin{array}{cccc}7 & ? & 6 & ? \\ ? & ? & ? & ? \end{array}) &= [A]_{2,4} \\
  \vdots & \quad \vdots \\
  \vdots & \quad \vdots 
\end{align*}
\]
Now if we permute the vertices by \( i \mapsto \pi(i) \) ....

\[
[A']_{\pi(i), \pi(j)} = [A]_{i,j}
\]
Now if we permute the vertices by $i \mapsto \pi(i)$ ....

$$[A']_{\pi(i), \pi(j)} = [A]_{i,j}$$

$$ff\left(\begin{array}{cccccc}
7 & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\end{array}\right) =$$

$$f\left(\begin{array}{cccccc}
7 & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast & \ast \\
\end{array}\right)$$

... in other words $f' (\pi \sigma) = f (\sigma)$.
... or $f' = f^\pi$, where

$$f^\pi(\sigma) = f(\pi^{-1}\sigma)$$

is the translate of $f$ by $\pi$. 
2. Non-commutative harmonic analysis and invariants
$G$ is a **group** if for any $x, y, z \in G$

1. $xy \in G$,
2. $x(yz) = (xy)z$,
3. there is an $e \in G$ such that $ex = xe = x$,
4. there is an $x^{-1} \in G$ such that $xx^{-1} = x^{-1}x = e$.

Permutations $\sigma: \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$ form a group called the **symmetric group**, denoted $S_n$. 
\[ \hat{f}(k) = \sum_{x=0}^{n-1} e^{-ikx} f(x) \]
\[ \hat{f}(\rho) = \sum_{x \in G} \rho(x) f(x) \]
\[ \rho(x) \rho(y) = \rho(xy) \]

\( \rho: G \rightarrow \mathbb{C}^{d \times d} \) is called a representation of \( G \).
Example

\[ \rho(e) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \]

\[ \rho((12)) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \]

\[ \rho((123)) = \begin{pmatrix} -1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{pmatrix} \]
Equivalence:

\[ \rho_1(x) = T^{-1} \rho_2(x) T \]

Reducibility:

\[ T^{-1} \rho(x) T = \begin{pmatrix} \rho_1(x) & 0 \\ 0 & \rho_2(x) \end{pmatrix} \]

\( \rho: \mathbb{G} \rightarrow \mathbb{C}^{d \times d} \) is called a representation of \( \mathbb{G} \).

A complete set of inequivalent irreducible unitary representations we denote \( \mathcal{R} \).
The **Fourier transform** on a group is

\[
\hat{f}(\rho) = \sum_{x \in G} f(x) \rho(x) \quad \rho \in \mathcal{R}
\]

- Diaconis: Group representations in probability and statistics (1988)
- Clausen, Maslen, Rockmore, Healy, ...: FFTs
- Kondor, Howard and Jebara: Multi-object tracking with representations of the symmetric group (AISTATS, 2007)
- Huang, Guestrin and Guibas: Efficient inference for distributions on permutations (NIPS, 2007)
\[ \hat{f}^t(\rho) = \rho(t) \hat{f}(\rho) \]
The **power spectrum** of $f$ is the set of invariant matrices

$$\hat{a}(\rho) = \hat{f}(\rho)^\dagger \cdot \hat{f}(\rho)$$

$$\hat{a}^t(\rho) = (\rho(t)\hat{f}(\rho))^\dagger \cdot (\rho(t)\hat{f}(\rho)) = \hat{f}(\rho)^\dagger \cdot \hat{f}(\rho) = \hat{a}(\rho)$$
Kakarala’s non-commutative bispectrum is

\[ b(\rho_1, \rho_2) = C^\dagger \left( \hat{f}(\rho_1) \otimes \hat{f}(\rho_2) \right)^\dagger C \bigoplus_\rho \hat{f}(\rho) \]

where

\[ \rho_1(z) \otimes \rho_2(z) = C \left[ \bigoplus_\rho \rho(z) \right] C^\dagger \]

is the Clebsch-Gordan decomposition.

[Kakarala, 1992]
The *skew spectrum* is the unitarily equivalent, but easier to compute set of matrices

\[
\hat{q}_z(\rho) = \hat{r}_z(\rho)^\dagger \cdot \hat{f}(\rho)
\]

where

\[
r_z(x) = f(xz) f(x)
\]

[Kondor, 2007]
3. Back to graphs...
What we have so far:

1. \( f(\sigma) = [A]_{\sigma(n), \sigma(n-1)} \)

2. Under permuting the vertices \( f' = f^\pi \)

3. Our favorite invariant is the skew spectrum

\[
\hat{q}_\nu(\rho) = \hat{r}_\nu(\rho) \dagger \cdot \hat{f}(\rho) \quad r_\nu(\sigma) = f(\sigma \nu) f(\sigma)
\]

where

\[
\hat{f}(\rho) = \sum_{\sigma \in S_n} \rho(\sigma) f(\sigma)
\]

Far too expensive in this form!
\[
\begin{align*}
  f(\begin{array}{cccc}
    7 & 6 & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
  \end{array}) &= [A]_{1,2} \\
  f(\begin{array}{cccc}
    7 & \text{?} & 6 & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
  \end{array}) &= [A]_{1,3} \\
  f(\begin{array}{cccc}
    7 & \text{?} & \text{?} & 6 \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
  \end{array}) &= [A]_{1,4} \\
  \vdots & \vdots & \vdots & \vdots \\
  f(\begin{array}{cccc}
    \text{?} & 7 & 6 & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
  \end{array}) &= [A]_{2,3} \\
  f(\begin{array}{cccc}
    \text{?} & 7 & \text{?} & 6 \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
    \text{?} & \text{?} & \text{?} & \text{?} \\
  \end{array}) &= [A]_{2,4} \\
  \vdots & \vdots & \vdots & \vdots 
\end{align*}
\]
1. The $\nu$ index only has to extend over one representative from each $S_{n-2} \sigma S_{n-2}$ coset.

2. The $\hat{f}$ and $\hat{r}_\nu$ Fourier transforms are very sparse.
\[ \hat{r}_\nu(\rho)^\dagger \cdot \hat{f}(\rho) \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ \hat{f}(\begin{array}{c}
\end{array}) = \]

\[ d = 1 \]

\[ d = n - 1 \]

\[ d = n(n-3)/2 \]

\[ d = (n-1)(n-2)/2 \]

\[ d = n(n-1)(n-5)/6 \]
The answer is 49.

(and it’s computable in $O(n^3)$ time)
$S_n$
Bratelli diagram
$S_n$ob

A C++ library for fast Fourier transforms on the symmetric group.

author: Risi Kondor, Columbia University (risi@cs.columbia.edu)

Development version as of August 23, 2006 (unstable!):

Documentation: [ps][pdf]
C++ source code: [directory]
BiBTeX entry: [bib]
Entire package: [tar.gz]

ALL SOFTWARE ON THIS PAGE IS DISTRIBUTED UNDER THE TERMS OF THE GNU GENERAL PUBLIC LICENSE [site]

References:

3. K. L. Kueh, T. Olson, D. Rockmore and K. S. Tan: Nonlinear approximation theory on finite

http://www.cs.columbia.edu/~risi/SnOB
#include <vector>
#include "base.h"
#include "Matrix.hpp"
#include <iostream>
#include "Sn.hpp"
#include "SnFunction.hpp"
#include "StandardTableau.hpp"

using namespace std;

class Sn::FourierTransform: FiniteGroup::FourierTransform{

public:

    friend class Sn::Function;
    friend class Sn::Ftree;

    FourierTransform(const Sn& _group);
    FourierTransform(const Sn& _group, int dummy):group(_group),n(_group.n){};
    FourierTransform(const Sn& _group, const vector<Matrix<FIELD>*>* matrices);
    FourierTransform(const Function& f);
    ~FourierTransform();

    Function* iFFT() const;

    FIELD operator()(const StandardTableau& t1, const StandardTableau& t2) const;

    double norm2() const {double result; for(int i=0; i<matrix.size(); i++) result+=1; return result;}

    string str() const;

    vector<Matrix<FIELD>*>* matrix;

private:

    void fft(const Sn::Function& f, const int offset);
    void ifft(Sn::Function* target, const int _offset) const;

    const int n;
    const Sn* group;
};
Sn::Irreducible

 Represents an irreducible representation $\rho$, of $S_n$.

 Parent class: FiniteGroup::Irreducible

 CONSTRUCTORS

 Irreducible(Sn* G, Partition* lambda)
 Construct the irreducible representation of the symmetric group G corresponding to the partition lambda.

 MEMBER FUNCTIONS

 Matrix<FIELD> & rho(const Sn::Element* sigma)
 Returns $\rho(\sigma)$, the representation matrix of permutation sigma in Young's orthogonal representation.

 FIELD character(const Partition* mu)
 Returns $\chi(\mu)$, the character of this representation at permutations of cycle type $\mu$.

 void computeTableaux()
 Compute the standard tableaux of this irreducible if they have not already been computed. Because this is an expensive operation, it is postponed until some function is called (such as rho or character) which requires the tableaux of this particular irreducible. computeTableaux() is called automatically by these functions, and once the tableaux have been computed they are stored for the lifetime of the Irreducible.

 StandardTableau* tableau(const int t)
 Return a new standard tableau of index t. This works even if tableauV has not been computed.

 void computeYUR()
 Compute and store the coefficients (2.5) and (2.6) in Young's orthogonal representation for all adjacent transpositions $\tau$ and all tableau $t$ of shape $\lambda$. Because this is an expensive operation, these coefficients are not normally computed until they are demanded by functions such as rho or character. computeYUR() is called automatically by these functions, and once the tableaux have been computed they are stored for the lifetime of the Irreducible. computeYUR() also requires the tableaux, so it calls computeTableaux() if those have not been computed yet.

 void applyCycle(const int j, Matrix<FIELD>* M, int m)
 void applyCycle(const int j, Matrix<FIELD>* M, int m)
4. Experiments
For $n$ up to about 300, the skew spectrum can be computed in fractions of a second.

For small graphs ($n \sim 5$) it’s complete!

For $n \sim 100$ good for learning tasks.
<table>
<thead>
<tr>
<th></th>
<th>MUTAG</th>
<th>ENZYME</th>
<th>NCI1</th>
<th>NCI109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instances/classes</td>
<td>600/6</td>
<td>188/2</td>
<td>4110/2</td>
<td>4127/2</td>
</tr>
<tr>
<td>Max. number of nodes</td>
<td>28</td>
<td>126</td>
<td>111</td>
<td>111</td>
</tr>
<tr>
<td>Reduced skew spectrum</td>
<td>88.61 (0.21)</td>
<td>25.83 (0.34)</td>
<td><strong>62.72</strong> (0.05)</td>
<td><strong>62.62</strong> (0.03)</td>
</tr>
<tr>
<td>Random walk kernel</td>
<td>71.89 (0.66)</td>
<td>14.97 (0.28)</td>
<td>51.30 (0.23)</td>
<td>53.11 (0.11)</td>
</tr>
<tr>
<td>Shortest path kernel</td>
<td>81.28 (0.45)</td>
<td><strong>27.53</strong> (0.29)</td>
<td>61.66 (0.10)</td>
<td>62.35 (0.13)</td>
</tr>
</tbody>
</table>
Conclusions
• Reduced the problem of representing graphs to an abstract algebraic problem.

• Being restricted to a homogeneous space makes it easy to compute the skew spectrum but also collapses its size.

• Surprisingly, just 49 scalar invariants seem to be able enough to do the job (compressed sensing).

• Natural question: what about labeled graphs?