Ranking with kernels in Fourier space

Risi Kondor (Caltech)
Marconi Barbosa (NICTA)
presented by Jonathan Huang (CMU)
Applications with rankings:

1. Recommendations
2. Elections
3. Sports Tournaments

Ranking:

\[ x_{i_1} \succ x_{i_2} \succ x_{i_3} \succ \ldots \succ x_{i_{n-1}} \succ x_{i_n} \]

Apples > Bananas > Pomegranate > Kiwi > Peach > ...

Hard to represent functions on \( n! \) rankings...
Kernel-based algorithms have many advantages for ranking:

1. Accommodate mixture of ranking types (full, partial, etc).
2. Representer theorem circumvents $n!$ size of symmetric group.
3. Rankings can be $x$ (inputs) or $y$ (outputs).
4. Variety of fast algorithms to choose from (SVM, GP, KDE, etc)

Disadvantage:
Kernel can be very expensive to evaluate
Total ranking:

\[ x_{i_1} \succ x_{i_2} \succ x_{i_3} \succ \ldots \succ x_{i_{n-1}} \succ x_{i_n} \]

Partial rankings (many types):

\[ x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k} \]

\[ x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k} \succ "the\ rest" \]

\[ x_{i_1} \succ \{x_{i_2}, x_{i_3}\} \succ \ldots \succ \{x_{i_{11}}, x_{i_{12}}\} \succ x_{i_{13}} \]

\[ x_{i_1} \succ x_{i_2}, \quad x_{j_1} \succ x_{j_2} \succ x_{j_3} \]

How do we compute the kernel between all of these?
Standard approach is to use an averaged kernel, e.g.

\[ K(x_{i_1} \succeq \ldots \succeq x_{i_k}, x'_{i_1} \succeq \ldots \succeq x'_{i_k}) = \sum_{\sigma'(i_1) > \ldots > \sigma'(i_k)} \sum_{\sigma(i_1) > \ldots > \sigma(i_k)} k(\sigma', \sigma) \]

Naively takes \( O((n-k)!^2) \) to compute!!!

Main result of paper: can be done in \( O((2k)^{2k+3}) \)

Notice: this is independent of \( n \).

In practice compute times are even better.
General theory of kernels on $S_n$

First, kernels on full rankings
Want a legitimate Mercer kernel $K$: Symmetric, Positive Definite (corresponding to inner product in some feature space)

Right-invariance

$\sigma(i) = j \iff$ item $i$ is ranked in position $n - j + 1$

$\Rightarrow k(\sigma', \sigma) = k(\sigma', \sigma)$

$\Rightarrow k(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$

$k$ is a pos. def. kernel $\iff \kappa$ is a pos. def. function

Kernel evaluations don’t depend on how the items are labeled

On real line, this is like kernels $K(x,y)$ which depend only on $|x-y|$
Diffusion kernels on full rankings

Theorem.
If $\Delta_{\sigma',\sigma} = q(\sigma'\sigma^{-1})$, then the diffusion kernel

$$k(\sigma', \sigma) = [e^{\beta \Delta}]_{\sigma', \sigma} = \kappa(\sigma'\sigma^{-1})$$

is right-invariant, and $\hat{k}(\lambda) = \exp(\beta \hat{q}(\lambda))$.

$$\Delta_{\sigma',\sigma} = \begin{cases} 
1 & \text{if } \sigma' = (i, i+1) \sigma \text{ for some } i \\
-(n-1) & \text{if } \sigma' = \sigma \\
0 & \text{otherwise}
\end{cases}$$

Main thing to know: diffusion kernel can be evaluated in closed form

Banana > Orange > Peach > Apricot > Fig > Grape
Banana > Orange > Apricot > Peach > Fig > Grape
Bochner’s theorem

• **For real numbers**: The kernel \( K(x,y) = k(|x-y|) \) is positive definite iff its Fourier transform is a nonnegative measure.

• **On the symmetric group**

  **Theorem.**
  The function \( \kappa : S_n \to \mathbb{R} \) is positive definite if and only if each of its Fourier components

  \[
  \hat{\kappa}(\lambda) = \sum_{\sigma \in S_n} \kappa(\sigma) \rho_\lambda(\sigma)
  \]

  is a positive definite matrix.

  [Kondor ’08, Fukumizu et. al., ‘08]
Computing the kernel fast (using Fourier theory)
Going back to the partial ranking kernel

\[
K(x_{i_1} \succ \ldots \succ x_{i_k}, x_{i'_1} \succ \ldots \succ x_{i'_k}) = \\
\sum_{\sigma'(i'_1) \succ \ldots \succ \sigma'(i'_k)} \sum_{\sigma(i_1) \succ \ldots \succ \sigma(i_k)} k(\sigma', \sigma) = *
\]

In group algebra language, letting

\[
A_{i_1, \ldots, i_k} = \sum_{\sigma(i_1) \succ \sigma(i_2) \succ \ldots \succ \sigma(i_k)} e_{\sigma}
\]

by the inverse Fourier transform

\[
* = k(A', A) = \langle A', \kappa \cdot A \rangle = \frac{1}{n!} \sum_{\lambda} d_{\lambda} \text{tr} \left[ \widehat{A}'(\lambda) ^\top \widehat{\kappa}(\lambda) \widehat{A}(\lambda) \right]
\]

Indicator function of permutations consistent with relative ranking of apples, oranges, …

Convolution of kernel function against indicator functions
Fourier transforms on rankings

- Interpretation:
  - 1st order: Orange is ranked best
  - 2nd order: Orange > Apple
  - 3rd order: Orange > Apple > Fig
Key mathematical idea is that the following are closely related:

1. **Convolution**

   \[ (f * g)(\sigma') = \sum_{\sigma \in S_n} f(\sigma' \sigma^{-1}) g(\sigma) \]

2. **Group algebra products**

   \[ (f * g)(\sigma) = (fg)(\sigma) \]

3. **Multiplication of Fourier matrices**

   \[ \hat{fg}(\lambda) = \hat{f}(\lambda) \cdot \hat{g}(\lambda) \]
Indicator function for rankings consistent with apple > banana

\[ A = \sum_{\sigma(i_1) > \sigma(i_2) > \ldots > \sigma(i_k)} e_k \]

Permutations consistent with a partial ranking can be factored!
Decomposition:

Sweep over interleavings of \{apple, banana\} into remaining items

Sweep over all permutations of remaining elements

Fix apple > banana

\[
\Pi_k^n \quad S_{n-k} \quad \pi_{i_1, i_2, \ldots, i_k}
\]

[“Riffled independence” in Huang et al NIPS 09]
More formally (using group algebra terminology)

\[ A = \sum_{\sigma(i_1) > \sigma(i_2) > \ldots > \sigma(i_k)} e_k = \prod_{k}^{n} S_n \pi_{i_1, \ldots, i_k} \]

To Fourier transform \( A \), multiply Fourier matrices of each term in the convolution.

Prop: Fourier matrices of \( S_{n-k} \) are zero beyond \( k^{th} \) order terms.

Corollary: Only need up to \( k^{th} \) order Fourier coefficients to evaluate kernel.
\[ k(A', A) = \langle \Pi_k^n S_{n-k} \pi_{i'_1, \ldots, i'_k}, \kappa \Pi_k^n S_{n-k} \pi_{i_1, \ldots, i_k} \rangle \]
\[ = \langle \Pi_k^{n^+} \kappa \Pi_k^n, S_{n-k} \pi_{i_1, \ldots, i_k} \pi_{i'_1, \ldots, i'_k}^{-1} S_{n-k} \rangle \]

\text{Proposition.}

The only non-zero elements of \( \hat{S}_{n-k} \) are \( [\hat{S}_{n-k}(\lambda)]_{t,t} \), where \( t \) is of the form

\[
\begin{array}{cccc}
1 & 2 & 3 & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\end{array}
\]

and \( 123 \cdots * \) denotes \( 1, 2, \ldots, n-k \).
Main Theorem.
The kernel between two partial rankings \( x_{i_1} < x_{i_2} < \ldots < x_{i_k} \) and \( x_{i'_1} < x_{i'_2} < \ldots < x_{i'_k} \) (or their top-\( k \) variants) can be computed in time \( O((2k)^{2k+3}) \).

<table>
<thead>
<tr>
<th>( k )</th>
<th>( k ) naive</th>
<th>( k ) our method</th>
<th>( k ) bound ((2k)^{2k+3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.0 ( \cdot ) 10(^7)</td>
<td>7</td>
<td>4096</td>
</tr>
<tr>
<td>3</td>
<td>1.8 ( \cdot ) 10(^{10})</td>
<td>34</td>
<td>1.7 ( \cdot ) 10(^6)</td>
</tr>
<tr>
<td>4</td>
<td>7.0 ( \cdot ) 10(^{12})</td>
<td>209</td>
<td>1.0 ( \cdot ) 10(^9)</td>
</tr>
<tr>
<td>5</td>
<td>4.2 ( \cdot ) 10(^{15})</td>
<td>1,546</td>
<td>1.0 ( \cdot ) 10(^{12})</td>
</tr>
<tr>
<td>6</td>
<td>2.6 ( \cdot ) 10(^{17})</td>
<td>13,327</td>
<td>1.3 ( \cdot ) 10(^{15})</td>
</tr>
<tr>
<td>7</td>
<td>2.6 ( \cdot ) 10(^{19})</td>
<td>130,922</td>
<td>2.2 ( \cdot ) 10(^{18})</td>
</tr>
</tbody>
</table>

Note that precomputations can be expensive. The method was implemented in \( S_{nob} \) and the paper contains preliminary experiments.
Conclusions

1. Kernel algorithms are a flexible framework for a variety of ranking tasks, but have not been used much in the past.

2. In most ranking problems $n$ is large, but $k$ is not that big.

3. To have any chance of computing the kernel in a reasonable amount of time, one must exploit the underlying algebra, as in this paper.

Special thanks to Dmitry Gavinsky for swapping slots with us.