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…

$$K(\sigma_1,\sigma_2)$$

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):

$$x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k}$$

$$x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k} \succ \text{"the rest"}$$

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

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

How do we compute the kernel between all of these?

Standard approach is to use an averaged kernel, e.g.

Sum over all full rankings consistent with partial rankings

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

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 $\mathbb{S}_n$

First, kernels on full rankings

Want a legitimate Mercer kernel K:

Symmetric, Positive Definite

(corresponding to inner product in some feature space)

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

Right-invariance

$$\sigma(i) = j \quad \longleftrightarrow \quad \text{item } i \text{ is ranked in position } n-j+1$$

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

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

$$k \text{ is a pos. def. kernel} \quad \Longleftrightarrow \quad \kappa \text{ is a pos. def. function}$$

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 $\widehat{\kappa}(\lambda) = \exp(\beta\,\widehat{q}(\lambda))$.

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

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

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 $\kappa\colon \mathbb{S}_n \to \mathbb{R}$ is positive definite if and only if each of its Fourier components

$$\widehat{\kappa}(\lambda) = \sum_{\sigma \in \mathbb{S}_n} \kappa(\sigma)\, \rho_\lambda(\sigma)$$

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

$$K(x_{i_1} \succ \ldots \succ x_{i_k},\, x_{i'_1} \succ \ldots \succ x_{i'_k}) =$$

$$\sum_{\sigma'(i'_1)>\ldots>\sigma'(i'_k)} \;\; \sum_{\sigma(i_1)>\ldots>\sigma(i_k)} k(\sigma', \sigma) = *$$

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

In group algebra language, letting

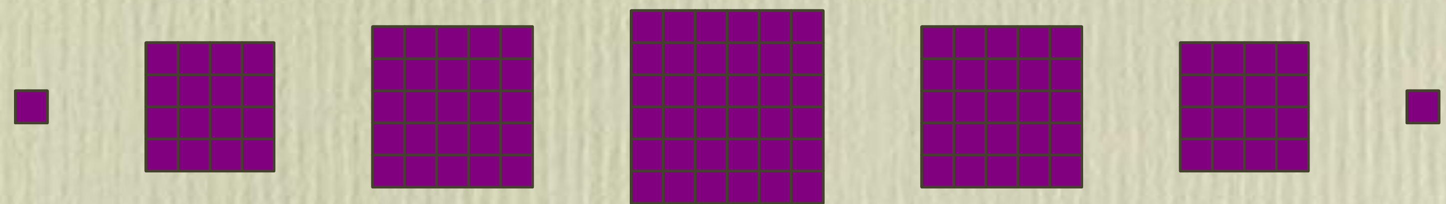
$$\boldsymbol{A}_{i_1,\ldots,i_k} = \sum_{\sigma(i_1)>\sigma(i_2)>\ldots>\sigma(i_k)} \boldsymbol{e}_\sigma$$

by the inverse Fourier transform

Convolution of kernel function against indicator functions

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

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

$$(f * g)(\sigma') = \sum_{\sigma \in \mathbb{S}_n} f(\sigma' \sigma^{-1})\, g(\sigma)$$

2. Group algebra products

$$(f * g)(\sigma) = (\boldsymbol{f}\boldsymbol{g})(\sigma)$$

3. Multiplication of Fourier matrices

$$\widehat{\boldsymbol{f}\boldsymbol{g}}(\lambda) = \widehat{\boldsymbol{f}}(\lambda) \cdot \widehat{\boldsymbol{g}}(\lambda)$$

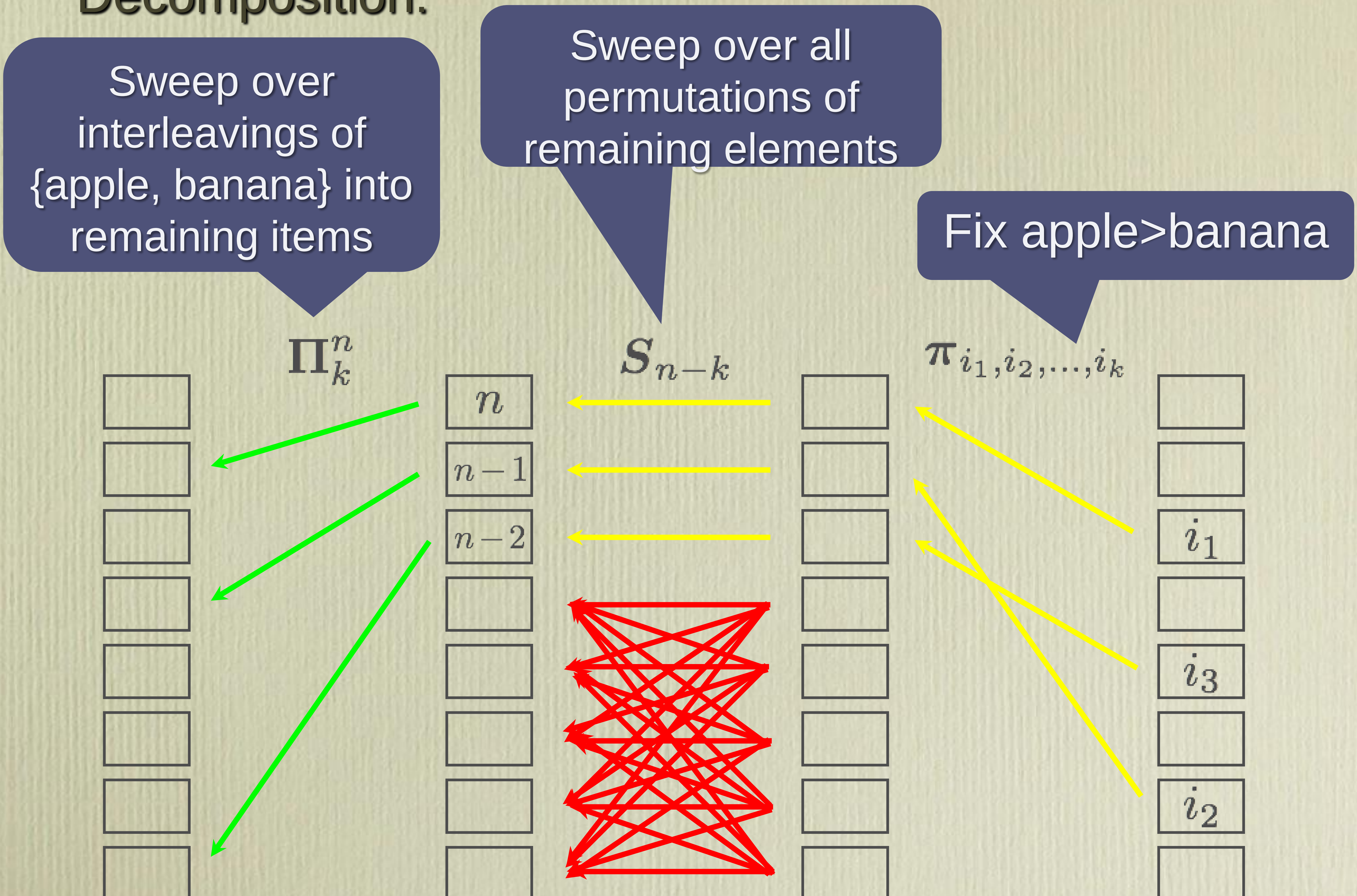
Indicator function for rankings consistent with apple>banana

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

A>B>C>D>E	A>C>B>D>E	C>A>D>B>E
A>B>E>D>C	A>E>B>D>C	E>A>D>B>C
A>B>D>C>E	A>D>B>C>E	D>A>C>B>E
A>B>C>E>D	A>C>B>E>D	C>A>E>B>D
A>B>E>C>D	A>E>B>C>D	E>A>C>B>D
A>B>D>E>C	A>D>B>E>C	D>A>E>B>C

Permutations consistent with a partial ranking can be factored!

Decomposition:



["Riffled independence" in Huang et al NIPS 09]

More formally (using group algebra terminology)

$$A = \sum_{\sigma(i_1)>\sigma(i_2)>\ldots>\sigma(i_k)} e_k = \Pi_k^n S_n \pi_{i_1,\ldots,i_k}$$

Convolution of indicator functions

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

$$k(\boldsymbol{A}', \boldsymbol{A}) = \left\langle \boldsymbol{\Pi}_k^n \, \boldsymbol{S}_{n-k} \, \boldsymbol{\pi}_{i'_1,\ldots,i'_k}, \; \boldsymbol{\kappa} \, \boldsymbol{\Pi}_k^n \, \boldsymbol{S}_{n-k} \, \boldsymbol{\pi}_{i_1,\ldots,i_k} \right\rangle$$

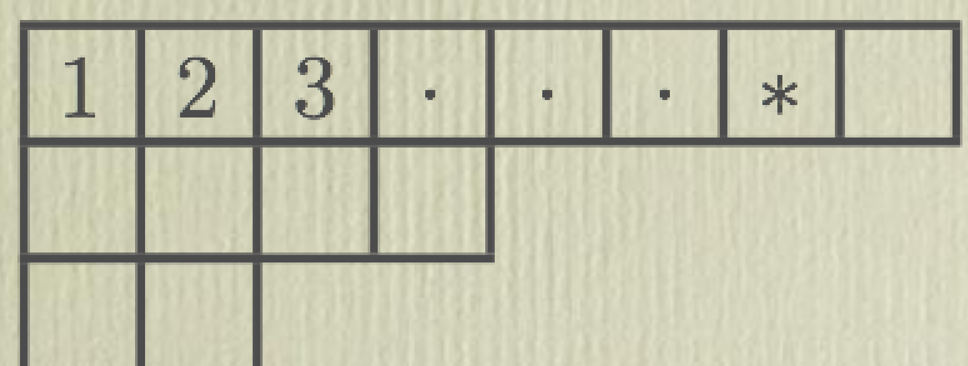
$$= \left\langle \boldsymbol{\Pi}_k^{n\dagger} \boldsymbol{\kappa} \, \boldsymbol{\Pi}_k^n, \; \boldsymbol{S}_{n-k} \, \boldsymbol{\pi}_{i_1,\ldots,i_k} \boldsymbol{\pi}^{-1}_{i'_1,\ldots,i'_k} \boldsymbol{S}_{n-k} \right\rangle$$

$O(k^k)$ matrices

$O(k^k)$ rows/columns in each

Proposition.

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



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 $O((2k)^{2k+3})$.

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

Note that precomputations can be expensive. The method was implemented in $\mathbb{S}_n$ob 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.