Supplements to “The Weighted Kendall and High-order Kernels for Permutations” (ICML 2018)

Yunlong Jiao\(^1\) Jean-Philippe Vert\(^2\)

Abstract

This is the supplements to the paper “The Weighted Kendall and High-order Kernels for Permutations” (ICML 2018).

1. Proofs of theorems

Proof of Theorem 2. The proof is constructive and the algorithm is summarized in Algorithm 1. C++/R implementatio available in the package kernrank at https://github.com/YunlongJiao/kernrank.

The algorithm can be decomposed into three parts. First, we compute \(\pi := \sigma'\sigma^{-1}\) by carrying out the inverse and composition of permutations, which can be done in linear time (Line 1). Due to the right-invariance of any concerning kernel, we have \(K(\sigma, \sigma') = \kappa(\pi)\) where \(\kappa\) is the corresponding p.d. function:

\[
\kappa^{U,k}_H(\pi) = \sum_{1 \leq i < j \leq n} I_{i \leq k} I_{j \leq k} I_{\pi(i) \leq k} I_{\pi(j) \leq k} \Pi_{\pi(i) < \pi(j)},
\]

\[
\kappa^{add}_U(\pi) = \sum_{1 \leq i < j \leq n} (u_i + u_j)(u_{\pi(i)} + u_{\pi(j)}) \Pi_{\pi(i) < \pi(j)},
\]

\[
\kappa^{mult}_U(\pi) = \sum_{1 \leq i < j \leq n} u_i u_j u_{\pi(i)} u_{\pi(j)} \Pi_{\pi(i) < \pi(j)},
\]

\[
\kappa^{avg}_W(\pi) = \sum_{1 \leq i < j \leq n} \frac{1}{n} \min \{i, \pi(i)\} \Pi_{\pi(i) < \pi(j)}.
\]

Second, we register a global variable \(s\) to record \(\kappa(\pi)\) (Line 2) and implement \(\kappa(\pi)\) in the function QUICKKAPPA (Lines 3–36). Finally, \(s\) is updated by calling the function QUICKKAPPA (Line 37) and then outputted by the algorithm.

---

\(^1\)University of Oxford, Oxford, UK \(^2\)MINES ParisTech & Institut Curie & Ecole Normale Supérieure, PSL Research University, Paris, France. Correspondence to: Jean-Philippe Vert <jean-philippe.vert@mines-paristech.fr>.


Algorithm 1 Top-\(k\), average and weighted Kendall kernel with additive or multiplicative weight

**input** permutations \(\sigma, \sigma'\), size \(n\), \(u\) for weighted Kendall kernel (optional), \(k\) for top-\(k\) Kendall kernel (optional)

1: \(\pi := \sigma'\sigma^{-1}\)
2: Initialize a global variable \(s := 0\) then define
3: function QUICKKAPPA(indices)
4: if length of indices > 1 then
5: \(\text{pivot} := \text{pick any element from indices}\)
6: \(\text{indhigh, indlow} := \text{two empty arrays}\)
7: \(\text{cnum, ctop, cmin, cwa, cwb, cw} := 0\)
8: for each \(i\) in indices do
9: if \(\pi(i) < \pi(\text{pivot})\) then
10: \(\text{Add } i \text{ to indlow}\)
11: \(\text{cnum} += 1\)
12: \(\text{cmin} += \min\{i, \pi(i)\}/n\)
13: \(\text{ctop} += \text{if } i \leq k \text{ and } \pi(i) \leq k \text{ then } 1 \text{ else } 0\)
14: \(\text{cwa} += u_i\)
15: \(\text{cwb} += u_{\pi(i)}\)
16: \(\text{cw} += u_i * u_{\pi(i)}\)
17: else
18: \(\text{Add } i \text{ to indhigh}\)
19: switch type of weighted Kendall kernel do
20: case STANDARD:
21: \(s += \text{cnum}\)
22: case TOP-k:
23: \(s += \text{if } i \leq k \text{ and } \pi(i) \leq k \text{ then } \text{ctop} \text{ else } 0\)
24: case AVERAGE:
25: \(s += \text{cmin}\)
26: case ADDITIVE WEIGHT:
27: \(s += \text{cwa} + \text{cwa} * u_{\pi(i)} + \text{cwb} * u_i + \text{cnum} * u_i * u_{\pi(i)}\)
28: case MULTIPLICATIVE WEIGHT:
29: \(s += \text{cw} * u_i * u_{\pi(i)}\)
30: end switch
31: end if
32: end for
33: QUICKKAPPA(indices)
34: QUICKKAPPA(indices)
35: end if
36: end function
37: Call QUICKKAPPA([1, n]) to update \(s\)

output \(K(\sigma, \sigma') = s\)
Central to the algorithm is the computation of $\kappa(\pi)$. It is based on an idea similar to a quicksort algorithm, where we recursively partition an array into two sub-arrays consisting of greater or smaller values according to a pivot, and cumulatively count the contributions between pairs of items with one in each sub-array. Specifically, suppose now $\pi$ is divided into two sub-arrays $\pi_{\text{ind high}}$ and $\pi_{\text{ind low}}$ where ranks in $\pi_{\text{ind high}}$ are all higher and those in $\pi_{\text{ind low}}$, now $\kappa(\pi)$ can be decomposed into

$$\kappa(\pi) = \kappa(\pi_{\text{ind high}}) + \kappa(\pi_{\text{ind low}}) + c(\pi_{\text{ind high}}, \pi_{\text{ind low}}),$$

where $c$ characterizes the weighted non-inversion number of $\pi$ restricted on pairs of items with one in each sub-array. The computation of $c(\pi_{\text{ind high}}, \pi_{\text{ind low}})$ depends on specific choice of weight and is depicted in the pseudo-code (Lines 19–30). Notably a single linear-time pass over $\pi$ is sufficient to compute $c(\pi_{\text{ind high}}, \pi_{\text{ind low}})$. By the analysis of deduction typically for a quicksort algorithm, the overall time complexity of our algorithm is on average $O(n \ln(n))$.

In particular, recall that the standard Kendall kernel is merely a special case of the weighted Kendall kernel with constant weight, and hence our algorithm provides an alternative to the efficient algorithm based on merge sort proposed by Knight (1966).

References