Max-Rank: Efficient Multiple Testing for Conformal Prediction

Alexander Timans[1] Christoph-Nikolas Straehle[2] Kaspar Sakmann[2] Christian A. Naesseth[1] Eric Nalisnick[3]

[1] University of Amsterdam [2] Bosch [3] Johns Hopkins University

Motivation

Coverage guarantees for prediction sets constructed via Conformal Prediction should extend efficiently to multivariate applications, e.g. object detection (right).

- Problem: This requires accounting for multiple testing.
- Solution: `max-rank`, a simple permutation-based multiplicity correction exploiting positive dependencies.

Example: Multivariate Conformal Prediction

Goal: ensure coverage for all box coordinates jointly.

Multiple Testing Framework

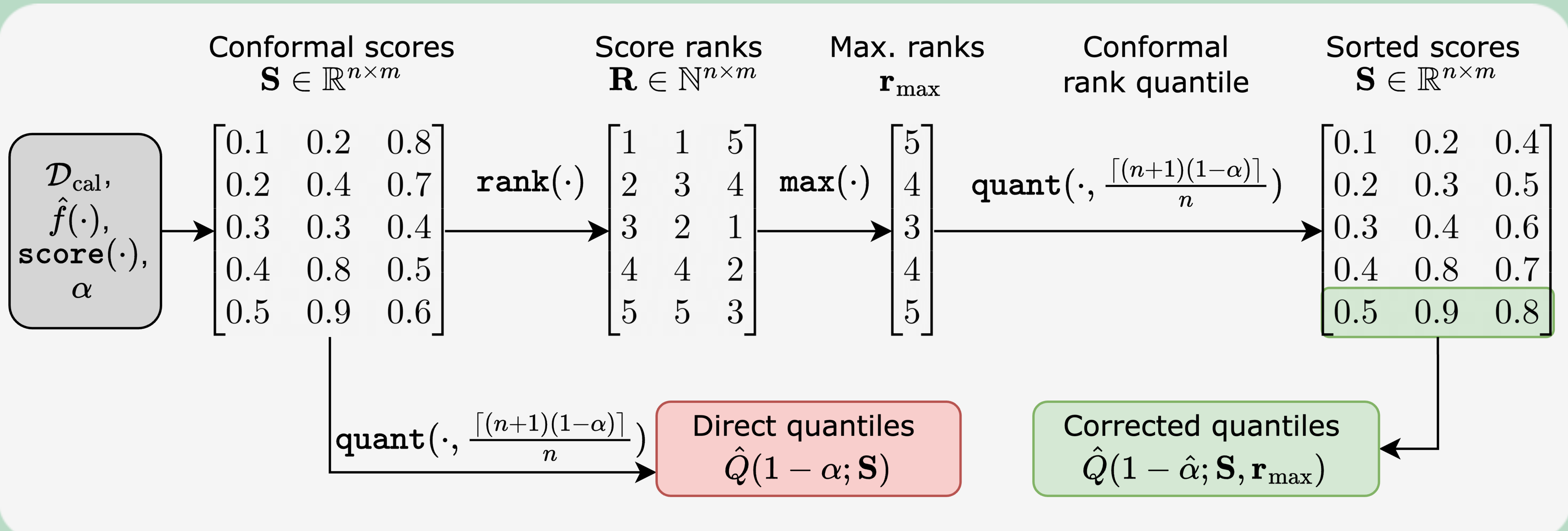
Conformal Prediction as Permutation Testing:

- INPUT: Calibration scores $S = \{s_1, \dots, s_n\}$, test sample (X_{n+1}, Y_{n+1})
- HYPOTHESIS: For candidate value $y \in \mathcal{Y}$ test the hypothesis pair $H_0 : Y_{n+1} = y, \ H_1 : Y_{n+1} \neq y$
- EVIDENCE: Compute a rank-based conformal p-value as
  $$\hat{P}_{n+1}(y; S) = \frac{|\{i = 1, \dots, n+1 : s_i \geq s_{n+1}\}|}{n+1}, s_{n+1} = \texttt{score}(\hat{f}(X_{n+1}), y)$$
- DECISION: Null rejection equates a prediction set exclusion, i.e.
  $$\hat{P}_{n+1}(y; S) > \alpha \Leftrightarrow y \in \hat{C}(X_{n+1}) \Leftrightarrow s_{n+1} \leq \hat{Q}(1-\alpha; S),$$
  $$\hat{P}_{n+1}(y; S) \leq \alpha \Leftrightarrow y \notin \hat{C}(X_{n+1}) \Leftrightarrow s_{n+1} > \hat{Q}(1-\alpha; S).$$

Multiple Testing Issue

- Multivariate Conformal Prediction as parallel testing causes multiplicity:
  $$\mathbb{P}\Big(\bigcap_{k=1}^{m} \big(Y_{n+1,k} \in \hat{C}(X_{n+1,k})\big)\Big) \geq 1 - m\alpha \overset{!}{\leq} 1 - \alpha.$$
- The Bonferroni correction, where $\alpha_{\text{Bonf}} = \alpha/m$, is inflexible and grows conservative under dependency (PRDS).

Our proposal: `max-rank`



Five lines of code!

```
import numpy as np

alpha = 0.1; n = 1000; m = 5; corr = 1.0 # params

# simulate correlated score matrix
cov = np.full((m, m), corr)
np.fill_diagonal(cov, 1)
S = np.random.multivariate_normal(np.zeros(m), cov, n)

# max-rank correction
R = np.argsort(np.argsort(S, axis=0), axis=0) # column-wise rank matrix
r_max = np.max(R, axis=1) # row-wise max-rank vec(r)_max
q = np.ceil((1 - alpha) * (n + 1)) / n # conformal target coverage level
rank_q = np.quantile(r_max, q, axis=0, method="higher") # quantile r_max
adj_quantile = np.sort(S, axis=0)[rank_q] # final corrected quantiles
```

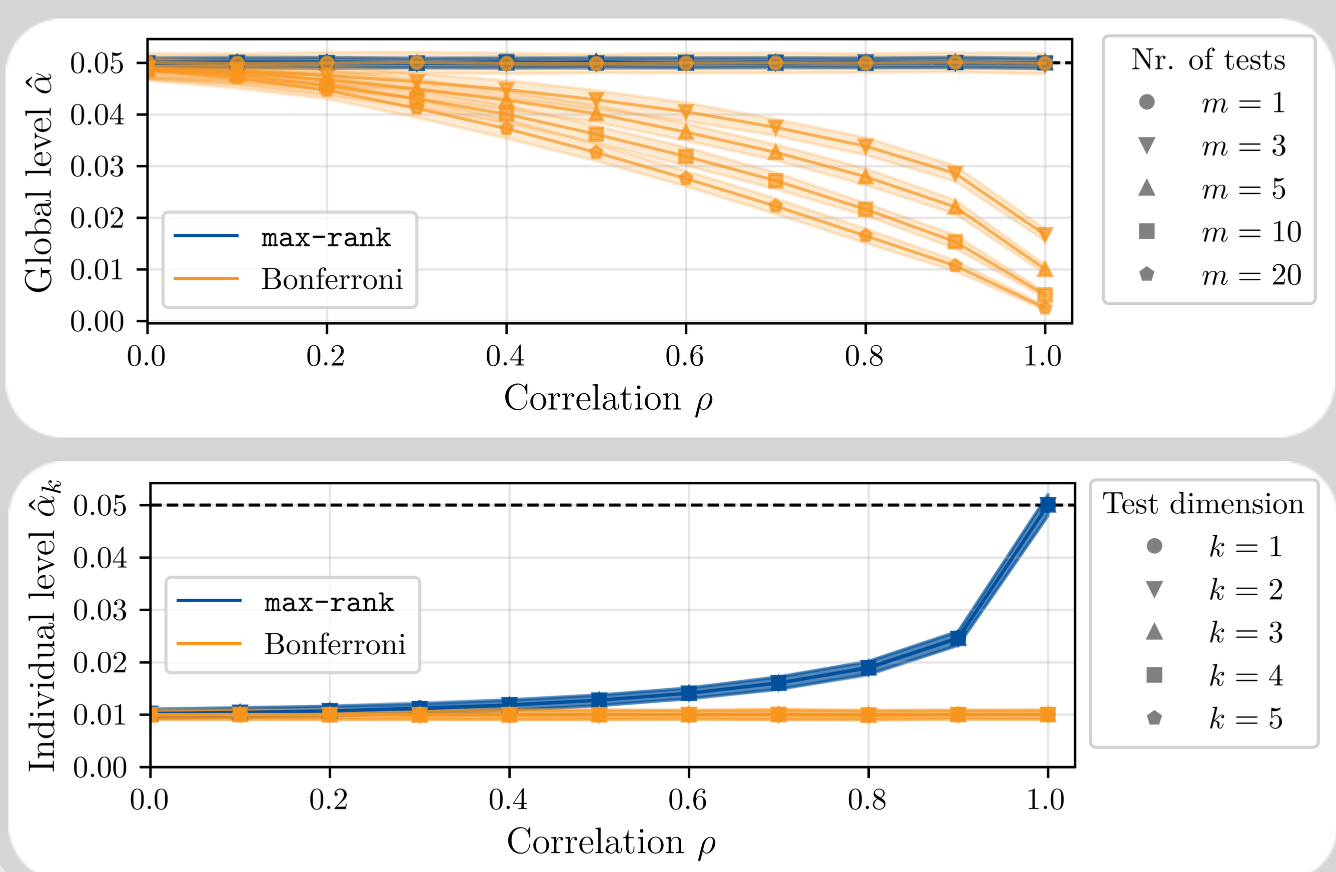
Key operation: rank sorting, $\mathcal{O}(n \log n \cdot m)$

Results

- Provably better than Bonferroni (Prop. 2)
- Closely related to Westfall & Young (1993)

Bonferroni Comparison

- Exploits existing positive dependencies



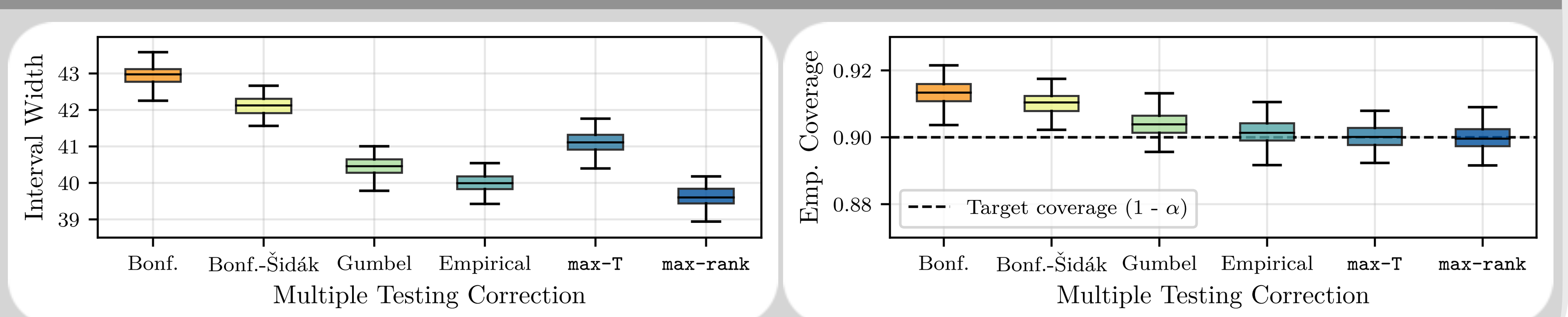
Multi-target Regression

- Efficient even as dimensions increase

	scpf ($m = 3$)		rf1 ($m = 8$)		scm1d ($m = 16$)	
Correction	Coverage	Interval Width	Coverage	Interval Width	Coverage	Interval Width
No correction	0.83 ± 0.04	18.89 ± 3.62	0.50 ± 0.01	1.88 ± 0.05	0.50 ± 0.02	243.87 ± 4.50
Bonferroni	0.95 ± 0.02	46.73 ± 13.39	0.92 ± 0.01	7.18 ± 0.44	0.96 ± 0.01	793.70 ± 50.65
Bonf.-Šidák	0.95 ± 0.02	46.73 ± 13.39	0.92 ± 0.01	6.98 ± 0.43	0.96 ± 0.01	793.70 ± 50.65
Gumbel Copula	0.90 ± 0.03	$\mathbf{30.15 \pm 5.53}$	0.91 ± 0.01	6.55 ± 0.39	0.94 ± 0.01	697.89 ± 36.48
Emp. Copula	0.92 ± 0.03	34.38 ± 6.74	0.91 ± 0.01	6.32 ± 0.31	0.91 ± 0.01	579.21 ± 15.43
`max-T`	0.91 ± 0.03	53.77 ± 10.63	0.90 ± 0.01	6.76 ± 0.24	0.90 ± 0.01	581.93 ± 14.14
`max-rank` (Ours)	0.91 ± 0.03	32.28 ± 6.26	0.90 ± 0.01	$\mathbf{6.08 \pm 0.29}$	0.90 ± 0.01	$\mathbf{564.01 \pm 13.79}$

Object Detection

- Efficient also for more complex settings



References

- Westfall & Young (1993). Resampling-based multiple testing (Wiley & Sons)
- Benjamini & Yekutieli (2001). Control of the FDR in multiple testing (Ann. S.)
- Bates et al. (2022). Testing for outliers with conformal p-values (Ann. S.)
- Messoudi et al. (2021). Copula-based conformal prediction for multi-target regression (Pattern Recognition)
- Timans et al. (2024). Two-step Conformal Prediction (ECCV)