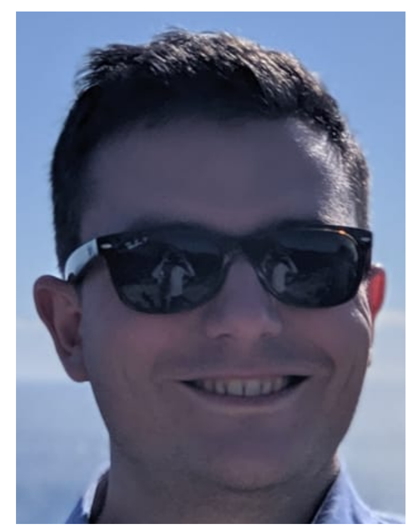
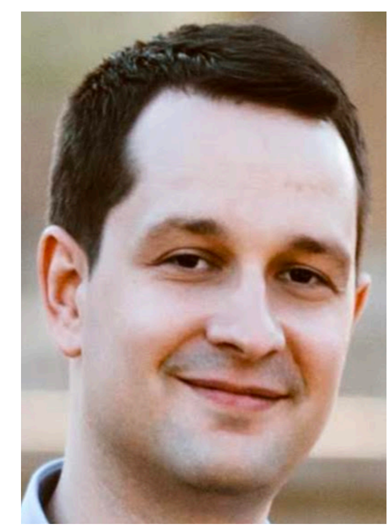
Fast differentiable sorting and ranking



M.Blondel O. Teboul Q. Berthet J. Djolonga

March 12th, 2020

Google Research

Background

Proposed method

Experimental results

Background

Proposed method

Experimental results

DL as Differentiable Programming

DL as Differentiable Programming

Deep learning increasingly synonymous with differentiable programming

Yann LeCun, 2018

"People are now building a **new kind of software** by assembling networks of parameterized **functional blocks** (including loops and conditionals) and by **training** them from examples using some form of gradient-based optimization."

DL as Differentiable Programming

Deep learning increasingly synonymous with differentiable programming

Yann LeCun, 2018

"People are now building a **new kind of software** by assembling networks of parameterized **functional blocks** (including loops and conditionals) and by **training** them from examples using some form of gradient-based optimization."

Many computer programming operations remain **poorly differentiable**

In this work, we focus on **sorting** and **ranking**.

Sorting as subroutine in ML

k-NN
(1) select neighbours
(2) majority vote

Trimmed regression
ignore large errors

Classifiers
select top-*k* activations

MoM estimators

Ranking / Sorting
$O(n \log n)$

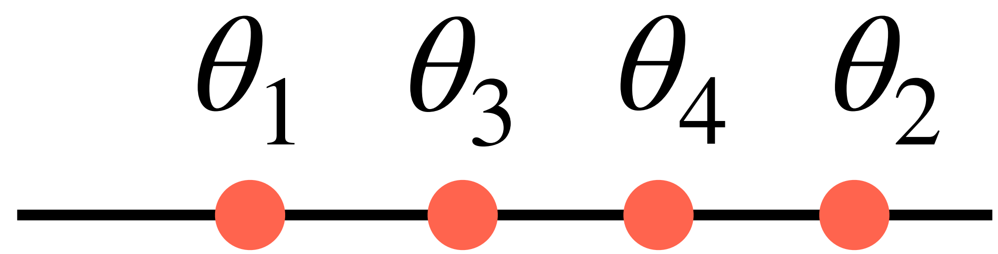
Learning to rank
NDCG loss and others

Descriptive statistics
Empirical distribution function
quantile normalization

Rank-based statistics
data viewed as ranks

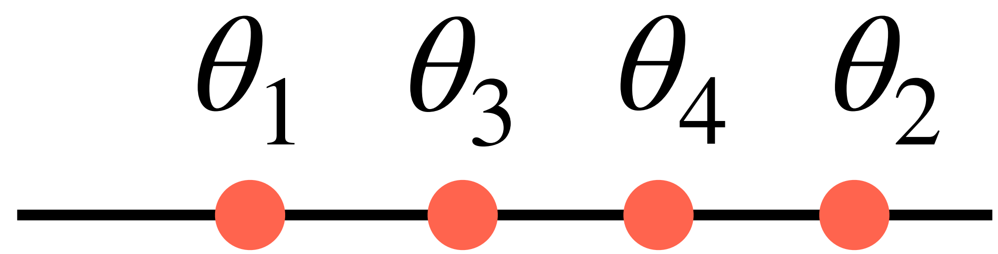
Slide credit: Marco Cuturi

Sorting



Argsort (decending) $\sigma(\theta) = (2,4,3,1)$

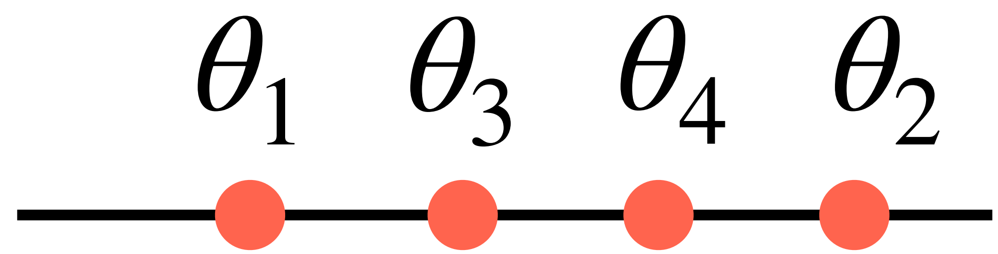
Sorting



Argsort (decending) $\sigma(\theta) = (2,4,3,1)$

Sort (descending) $s(\theta) \triangleq \theta_{\sigma(\theta)}$

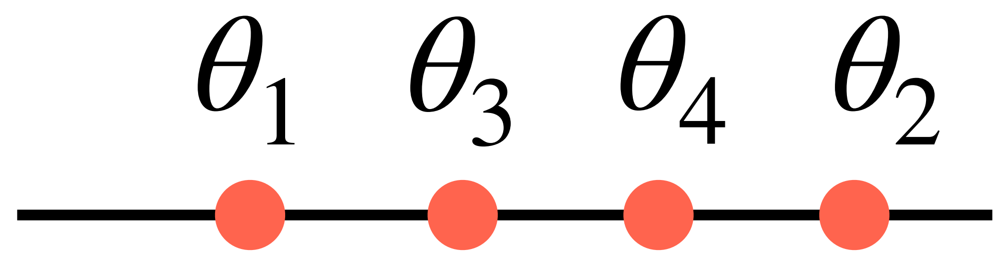
Sorting



Argsort (decending) $\sigma(\theta) = (2,4,3,1)$

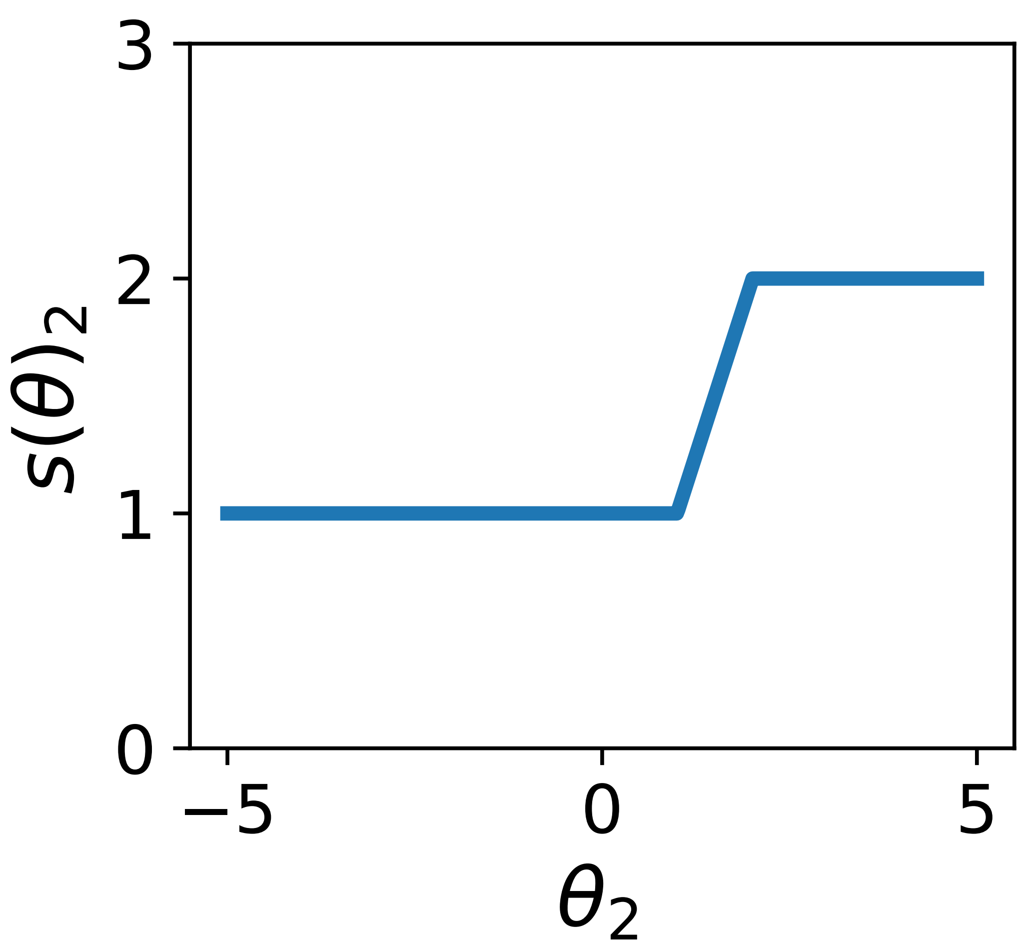
Sort (descending) $s(\theta) \triangleq \theta_{\sigma(\theta)} = (\theta_2, \theta_4, \theta_3, \theta_1)$

Sorting



Argsort (decending) $\sigma(\theta) = (2,4,3,1)$

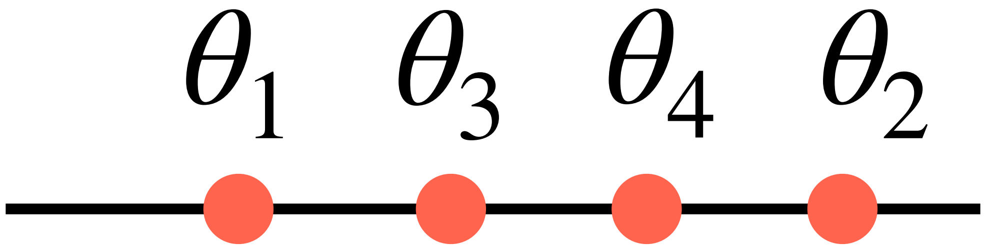
Sort (descending) $s(\theta) \triangleq \theta_{\sigma(\theta)} = (\theta_2, \theta_4, \theta_3, \theta_1)$



piecewise linear

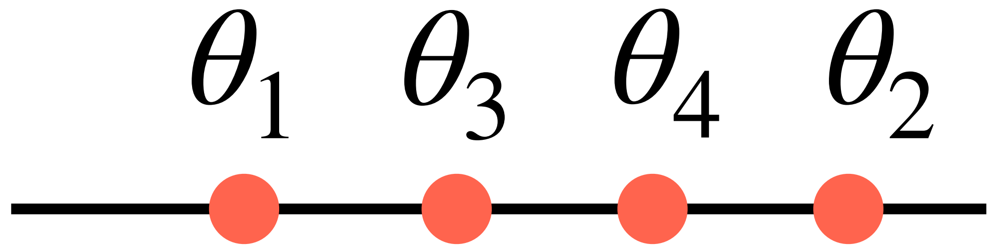
induces non-convexity

Ranking



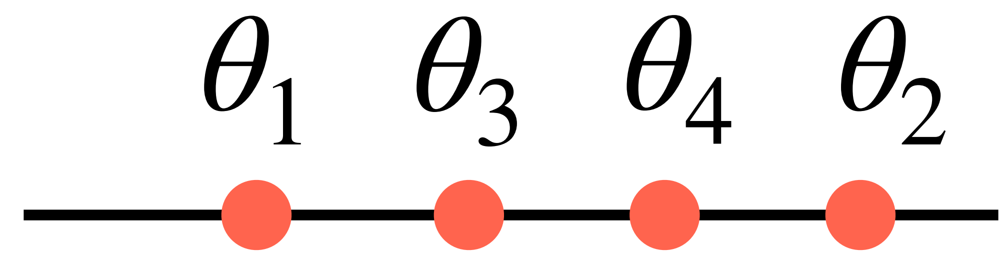
Ranks $$r(\theta) \triangleq \sigma^{-1}(\theta)$$

Ranking

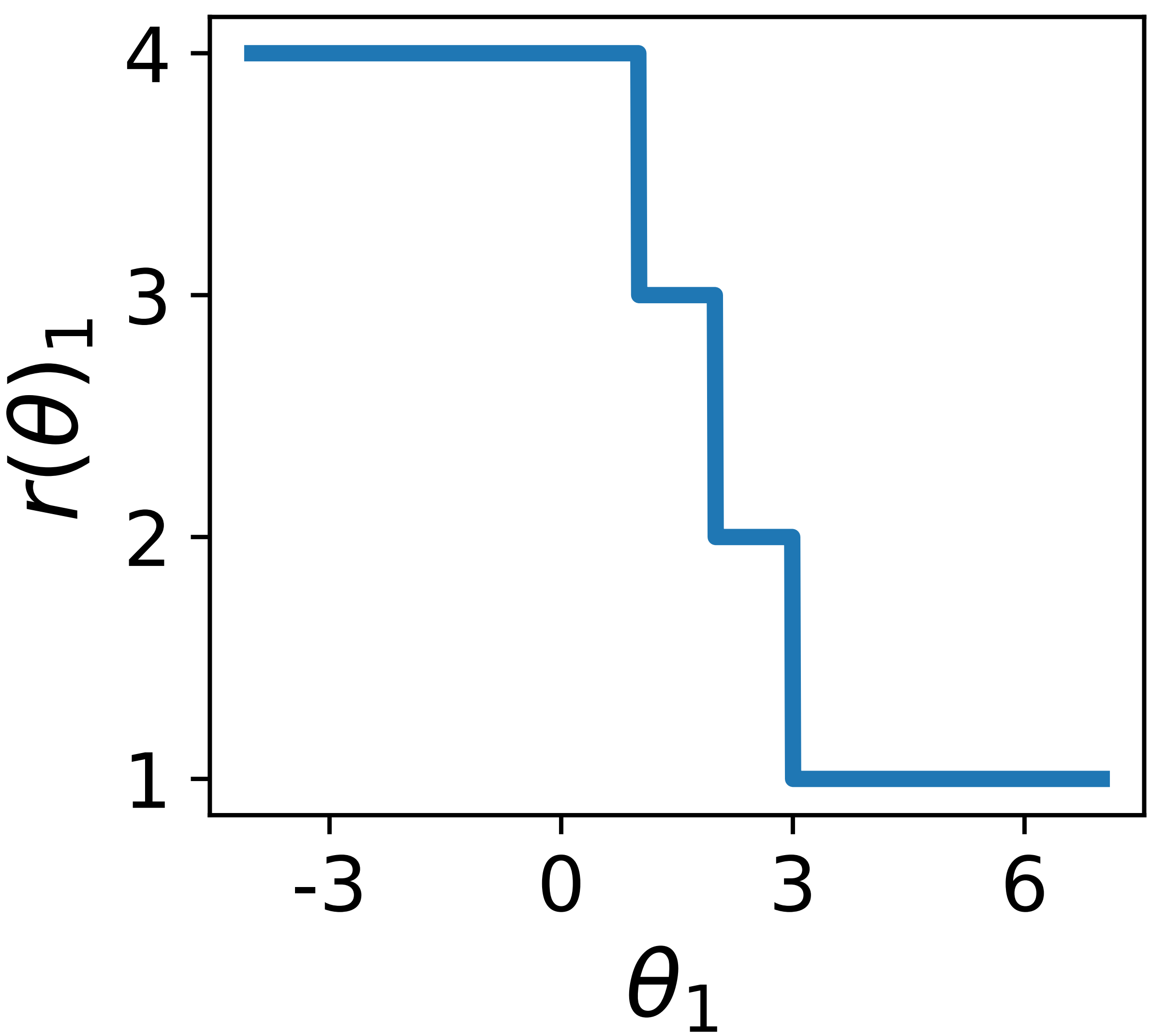


Ranks $$r(\theta) \triangleq \sigma^{-1}(\theta) = (4,1,3,2)$$

Ranking



Ranks $$r(\theta) \triangleq \sigma^{-1}(\theta) = (4,1,3,2)$$



discontinuous

piecewise constant

Related work on soft ranks

Soft ranks : differentiable proxies to "hard" ranks

Related work on soft ranks

Soft ranks : differentiable proxies to “hard” ranks

- Random perturbation technique to compute expected ranks in $O(n^3)$ time [Taylor et al., 2008]

Related work on soft ranks

Soft ranks : differentiable proxies to "hard" ranks

- Random perturbation technique to compute expected ranks in $O(n^3)$ time [Taylor et al., 2008]

- Using pairwise comparisons in $O(n^2)$ time [Qin et al., 2010]

$$r_i(\theta) \triangleq 1 + \sum_{i \neq j} \mathbf{1}[\theta_i < \theta_j]$$

Related work on soft ranks

Soft ranks : differentiable proxies to "hard" ranks

- Random perturbation technique to compute expected ranks in $O(n^3)$ time [Taylor et al., 2008]

- Using pairwise comparisons in $O(n^2)$ time [Qin et al., 2010]

$$r_i(\theta) \triangleq 1 + \sum_{i \neq j} \mathbf{1}[\theta_i < \theta_j]$$

- Regularized optimal transport approach and Sinkhorn in $O(T\, n^2)$ time [Cuturi et al., 2019]

Related work on soft ranks

Soft ranks : differentiable proxies to "hard" ranks

- Random perturbation technique to compute expected ranks in $O(n^3)$ time [Taylor et al., 2008]

- Using pairwise comparisons in $O(n^2)$ time [Qin et al., 2010]

$$r_i(\theta) \triangleq 1 + \sum_{i \neq j} \mathbf{1}[\theta_i < \theta_j]$$

- Regularized optimal transport approach and Sinkhorn in $O(T\ n^2)$ time [Cuturi et al., 2019]

None of these works achieves O(n log n) complexity

Background

Proposed method

Experimental results

Our proposal

Our proposal

- Differentiable (soft) relaxations of s(θ) and r(θ)

Our proposal

- Differentiable (soft) relaxations of s(θ) and r(θ)
- Two formulations: **L2** and Entropy regularised

Our proposal

- Differentiable (soft) relaxations of s(θ) and r(θ)
- Two formulations: **L2** and Entropy regularised
- “Convexification” effect

Our proposal

- Differentiable (soft) relaxations of s(θ) and r(θ)
- Two formulations: **L2** and Entropy regularised
- "Convexification" effect
- Exact computation in O(n log n) time (forward pass)

Our proposal

- Differentiable (soft) relaxations of s(θ) and r(θ)
- Two formulations: **L2** and Entropy regularised
- “Convexification” effect
- Exact computation in O(n log n) time (forward pass)
- Exact multiplication with the Jacobian in O(n) time without unrolling (backward pass)

Strategy outline

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

→ Turn LP into a projection onto convex polytopes

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

→ Turn LP into a projection onto convex polytopes

3. Derive algorithm for **computing** the projection

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

→ Turn LP into a projection onto convex polytopes

3. Derive algorithm for **computing** the projection

→ Ideally, the projection shoud be computable in the same cost as the original function…

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

→ Turn LP into a projection onto convex polytopes

3. Derive algorithm for **computing** the projection

→ Ideally, the projection shoud be computable in the same cost as the original function…

4. Derive algorithm for **differentiating** the projection

Strategy outline

1. Express s(θ) and r(θ) as **linear programs** (LP) over convex polytopes

→ Turn algorithmic function into an optimization problem

2. Introduce **regularization** in the LP

→ Turn LP into a projection onto convex polytopes

3. Derive algorithm for **computing** the projection

→ Ideally, the projection shoud be computable in the same cost as the original function…

4. Derive algorithm for **differentiating** the projection

→ Could be challenging (argmin differentiation problem)

Strategy outline

	Cuturi et al. [2019]	This work

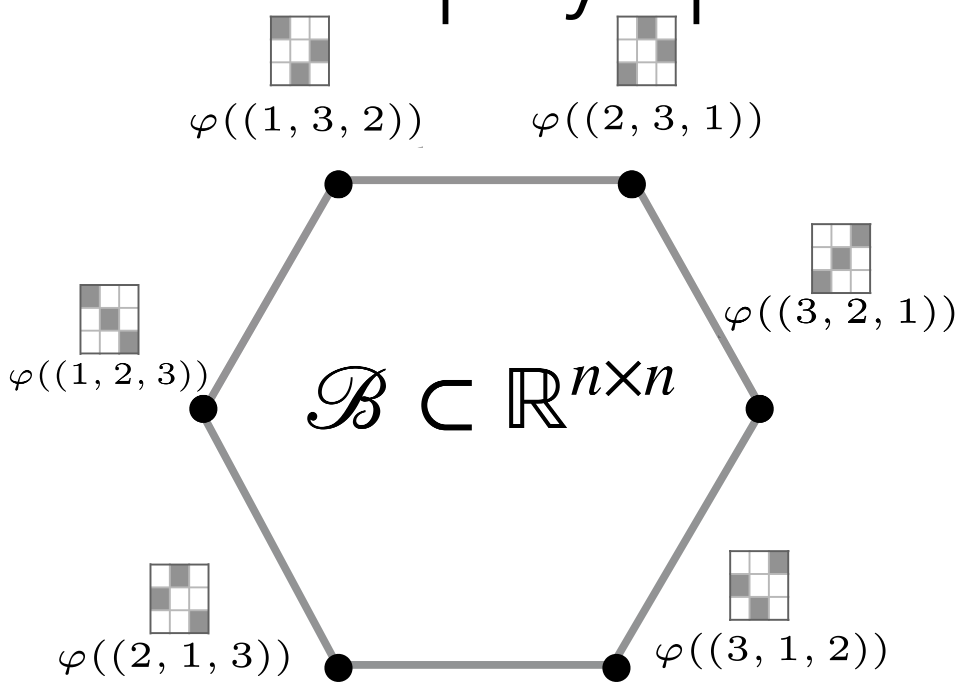
Strategy outline

Cuturi et al. [2019]

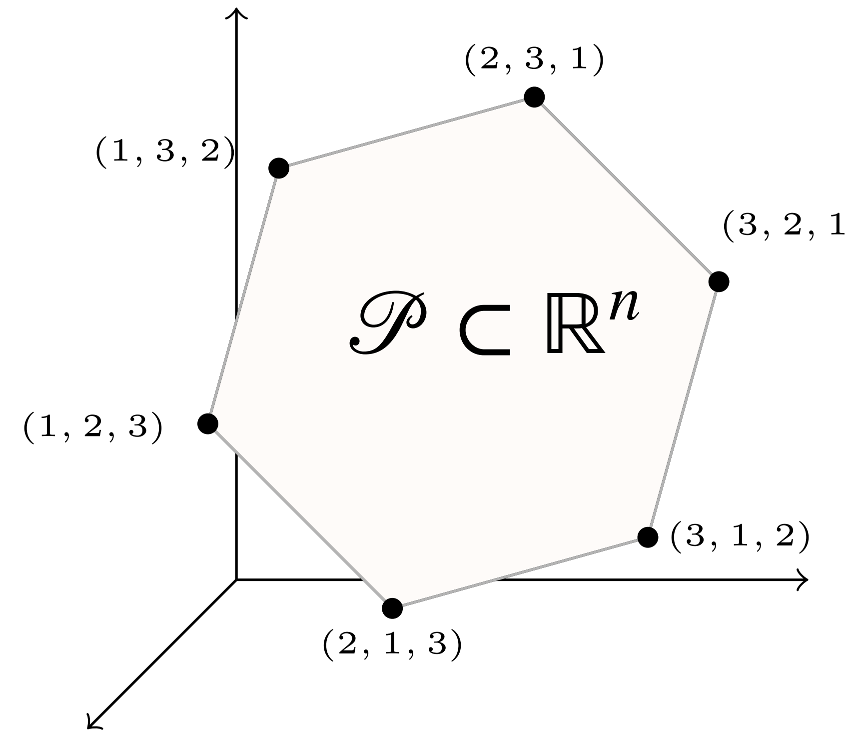
This work

1. LP

Birkhoff polytope

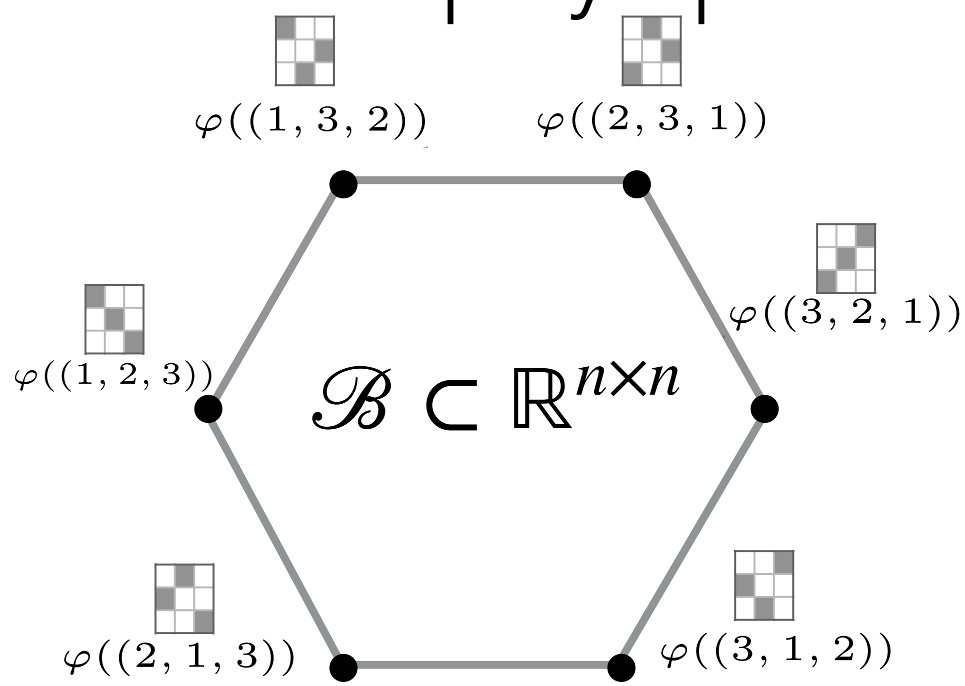


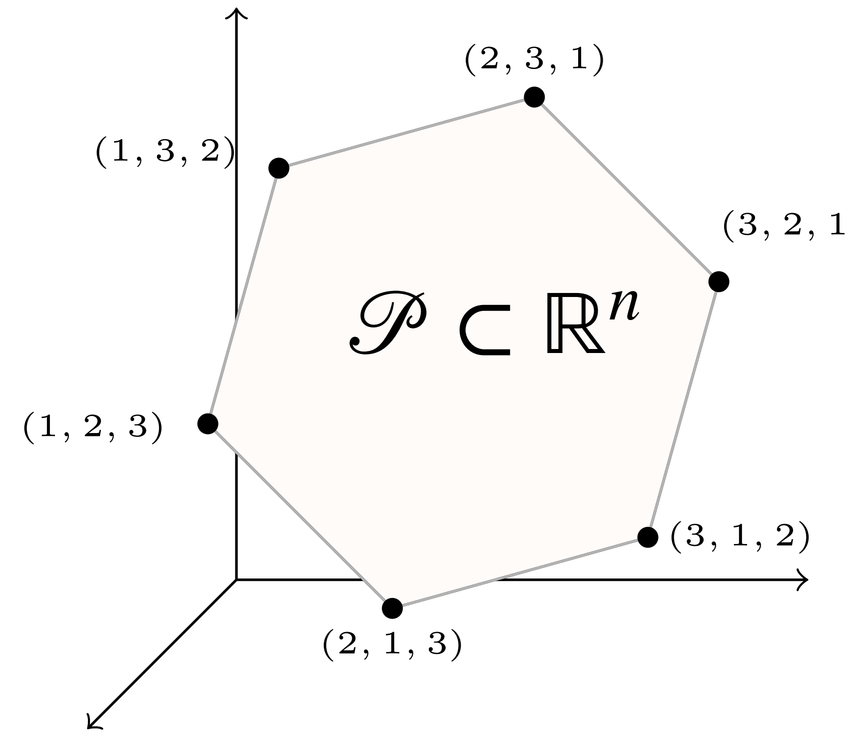
Permutahedron



Strategy outline

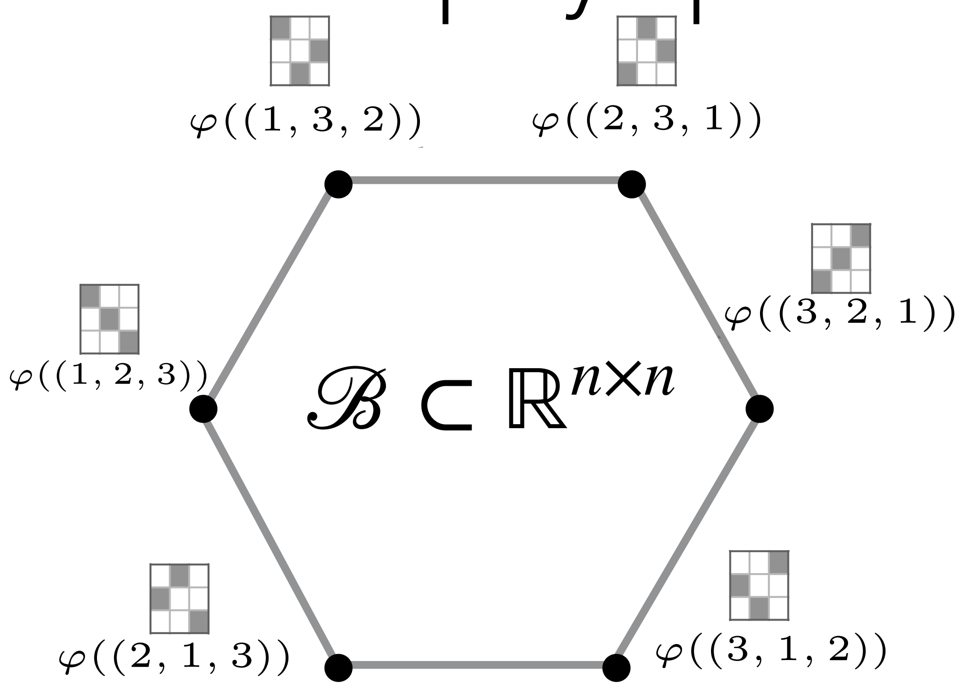
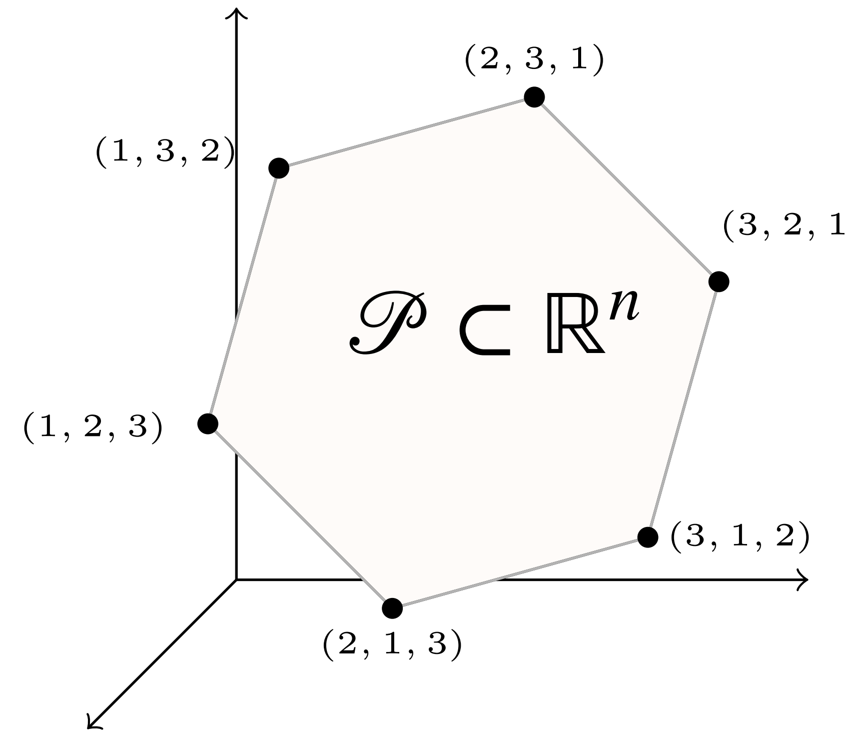
	Cuturi et al. [2019]	This work
1. LP	Birkhoff polytope	Permutahedron





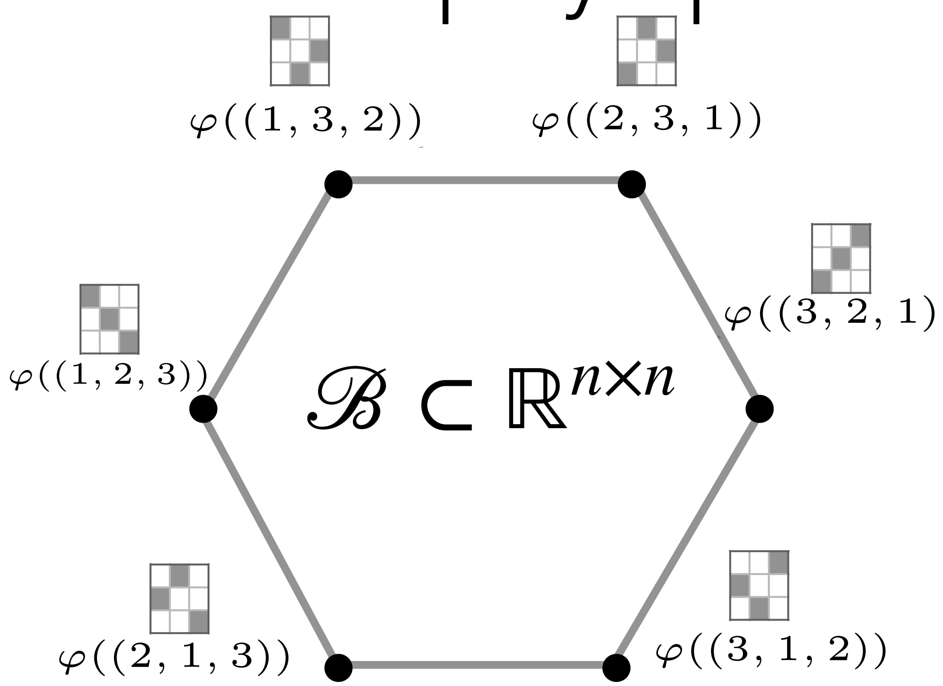
2. Regularization	Entropy	**L2** or Entropy

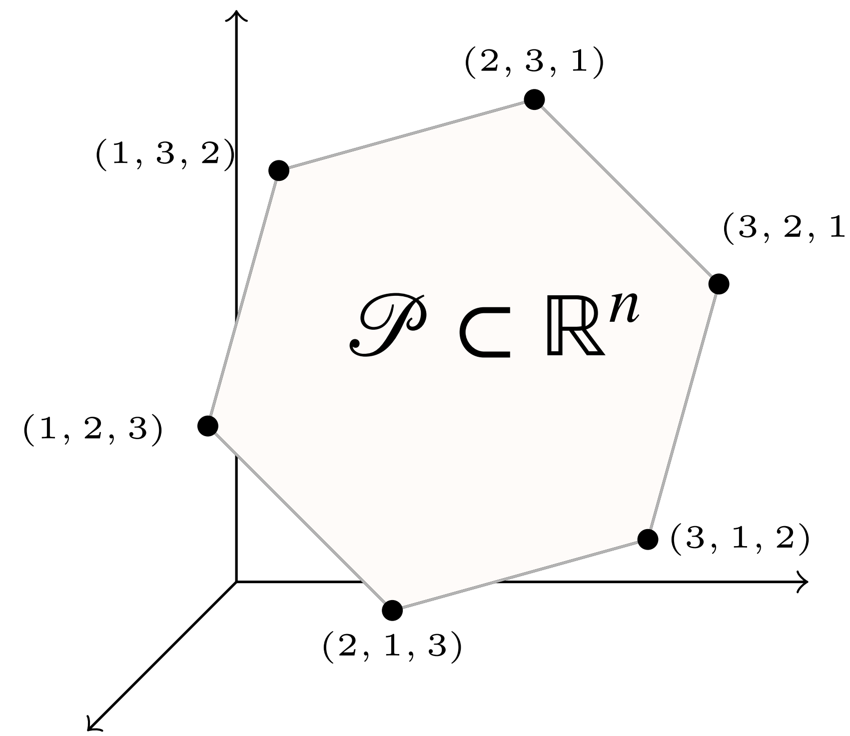
Strategy outline

	Cuturi et al. [2019]	This work
1. LP	Birkhoff polytope	Permutahedron
		
2. Regularization	Entropy	**L2** or Entropy
3. Computation	Sinkhorn	Pool Adjacent Violators (PAV)

Strategy outline

	Cuturi et al. [2019]	This work
1. LP	Birkhoff polytope	Permutahedron

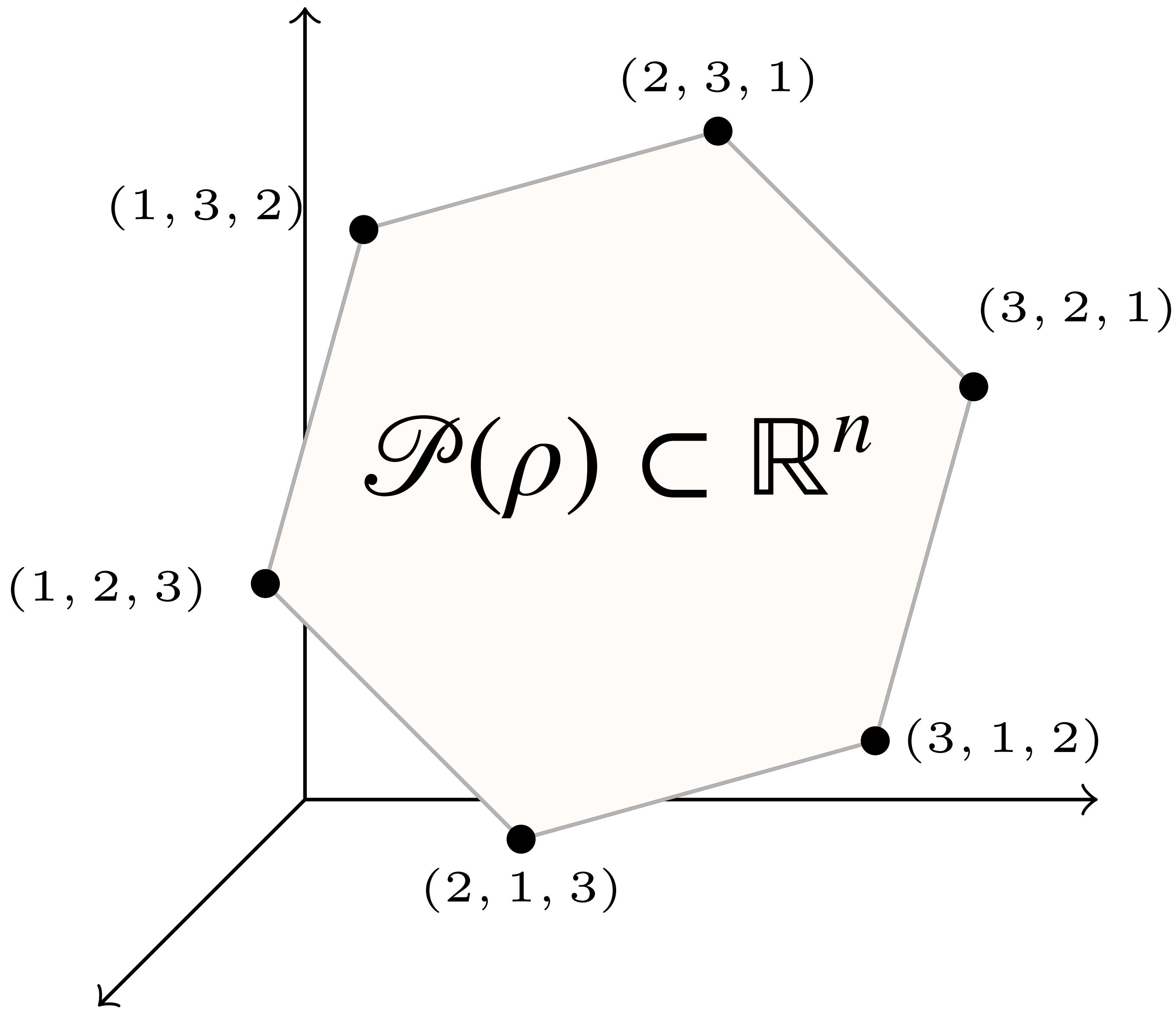




	Cuturi et al. [2019]	This work
2. Regularization	Entropy	**L2** or Entropy
3. Computation	Sinkhorn	Pool Adjacent Violators (PAV)
4. Differentiation	Backprop through Sinkhorn **iterates**	Differentiate PAV **solution**

Permutahedron

$$\mathscr{P}(w) \triangleq conv(\{w_\sigma : \sigma \in \Sigma\}) \subset \mathbb{R}^n$$



$$\rho \triangleq (n, n-1, \ldots, 1)$$

Step 1: linear programming formulations

Step 1: linear programming formulations

Proposition

$$s(\theta) = \arg\max_{y \in \mathcal{P}(\theta)} \langle y, \rho \rangle$$

$$\rho \triangleq (n, n-1, \ldots, 1)$$

Step 1: linear programming formulations

Proposition

$$s(\theta) = \arg\max_{y \in \mathscr{P}(\theta)} \langle y, \rho \rangle$$

$$r(\theta) = \arg\max_{y \in \mathscr{P}(\rho)} \langle y, -\theta \rangle$$

$$\rho \triangleq (n, n-1, \ldots, 1)$$

Proof of the first claim

$$\rho_n > \rho_{n-1} > \ldots > 1 \Rightarrow \sigma(\theta) = \arg\max_{\sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

Proof of the first claim

$$\rho_n > \rho_{n-1} > \ldots > 1 \Rightarrow \sigma(\theta) = \arg\max_{\sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$s(\theta) \triangleq \theta_{\sigma(\theta)}$$

Proof of the first claim

$$\rho_n > \rho_{n-1} > \ldots > 1 \Rightarrow \sigma(\theta) = \arg\max_{\sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$s(\theta) \triangleq \theta_{\sigma(\theta)}$$

$$= \arg \max_{\theta_\sigma :\ \sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

Proof of the first claim

$$\rho_n > \rho_{n-1} > \ldots > 1 \Rightarrow \sigma(\theta) = \arg\max_{\sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$s(\theta) \triangleq \theta_{\sigma(\theta)}$$

$$= \arg\max_{\theta_\sigma:\ \sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$= \arg\max_{y \in \Sigma(\theta)} \langle y, \rho \rangle$$

Proof of the first claim

$$\rho_n > \rho_{n-1} > \ldots > 1 \Rightarrow \sigma(\theta) = \arg\max_{\sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$s(\theta) \triangleq \theta_{\sigma(\theta)}$$

$$= \arg\max_{\theta_\sigma :\, \sigma \in \Sigma} \langle \theta_\sigma, \rho \rangle$$

$$= \arg\max_{y \in \Sigma(\theta)} \langle y, \rho \rangle$$

$$= \arg\max_{y \in \mathscr{P}(\theta)} \langle y, \rho \rangle$$

Step 2: introducing regularization

Step 2: introducing regularization

Quadratic regularization $Q(y) \triangleq \frac{1}{2}\|y\|^2$

$$P_Q(z, w) \triangleq \arg\max_{y \in \mathscr{P}(w)} \langle y, z \rangle - Q(y)$$

Step 2: introducing regularization

Quadratic regularization $Q(y) \triangleq \frac{1}{2}\|y\|^2$

$$P_Q(z, w) \triangleq \arg\max_{y \in \mathscr{P}(w)} \langle y, z \rangle - Q(y) = \arg\min_{y \in \mathscr{P}(w)} \|y - z\|^2$$

Step 2: introducing regularization

Quadratic regularization $Q(y) \triangleq \frac{1}{2}\|y\|^2$

$$P_Q(z, w) \triangleq \arg\max_{y \in \mathscr{P}(w)} \langle y, z \rangle - Q(y) = \arg\min_{y \in \mathscr{P}(w)} \|y - z\|^2$$

Definition

$$s_{\varepsilon Q}(\theta) \triangleq P_{\varepsilon Q}(\rho, \theta) = P_Q(\rho/\varepsilon, \theta)$$

Step 2: introducing regularization

Quadratic regularization $Q(y) \triangleq \frac{1}{2}\|y\|^2$

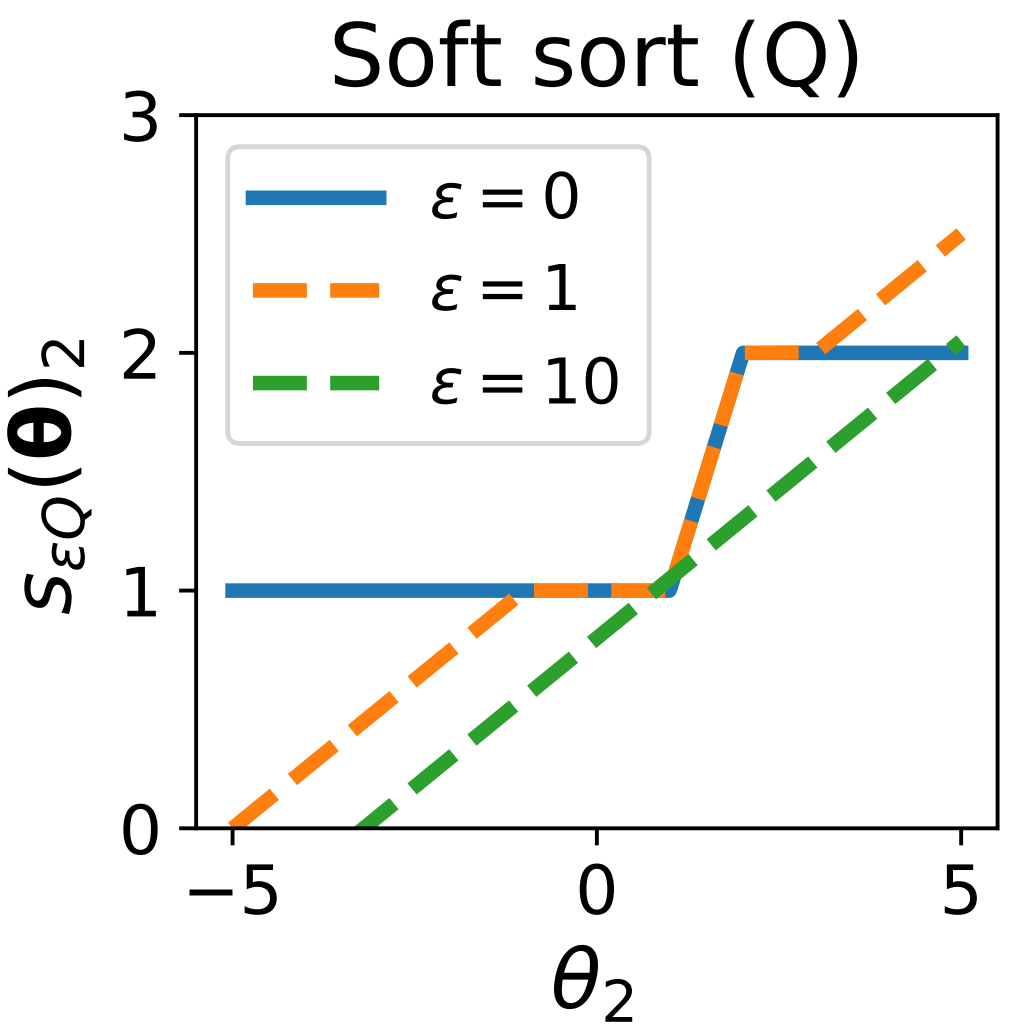
$$P_Q(z, w) \triangleq \arg\max_{y \in \mathscr{P}(w)} \langle y, z \rangle - Q(y) = \arg\min_{y \in \mathscr{P}(w)} \|y - z\|^2$$

Definition

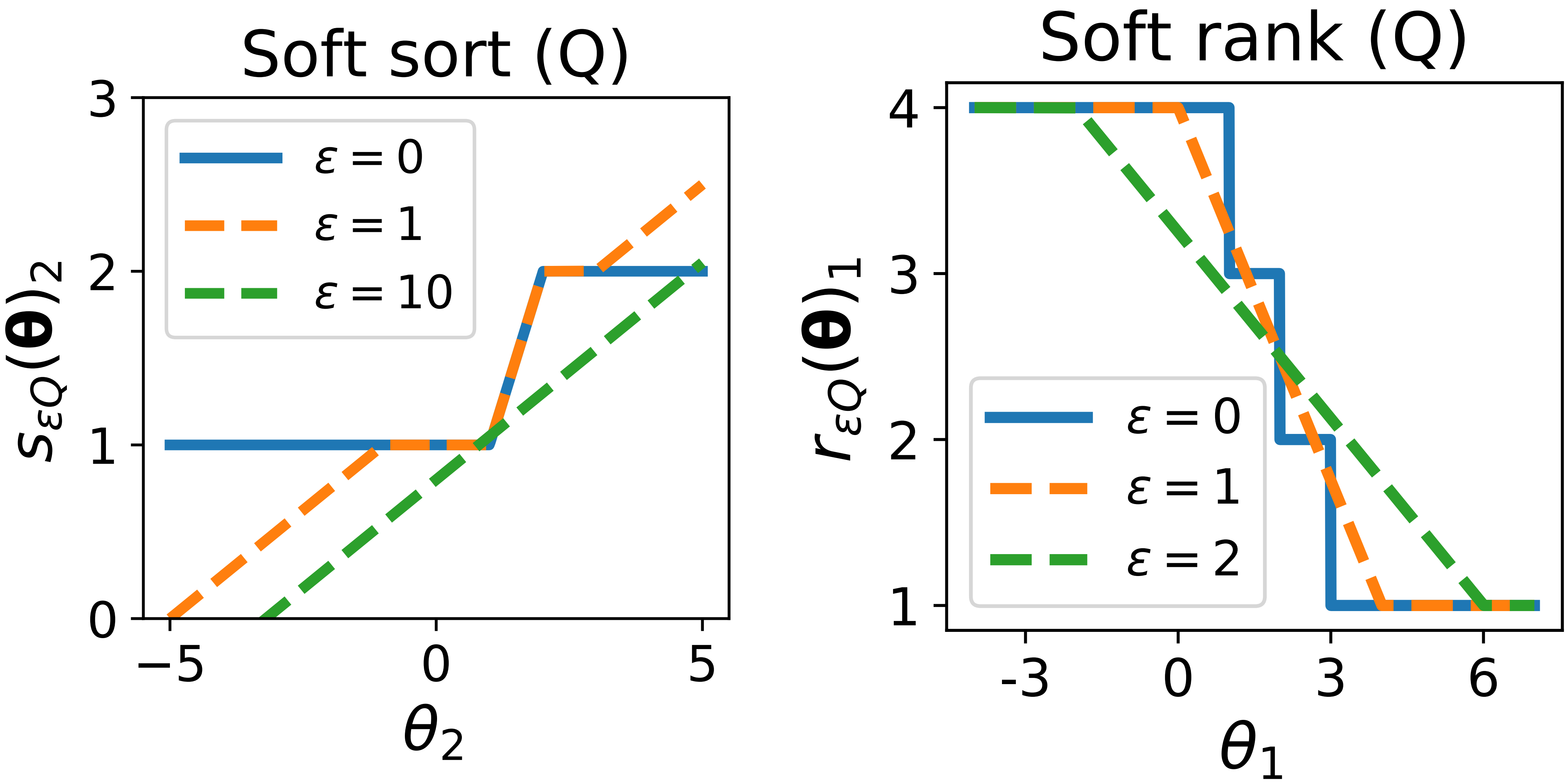
$$s_{\varepsilon Q}(\theta) \triangleq P_{\varepsilon Q}(\rho, \theta) = P_Q(\rho/\varepsilon, \theta)$$

$$r_{\varepsilon Q}(\theta) \triangleq P_{\varepsilon Q}(-\theta, \rho) = P_Q(-\theta/\varepsilon, \rho)$$

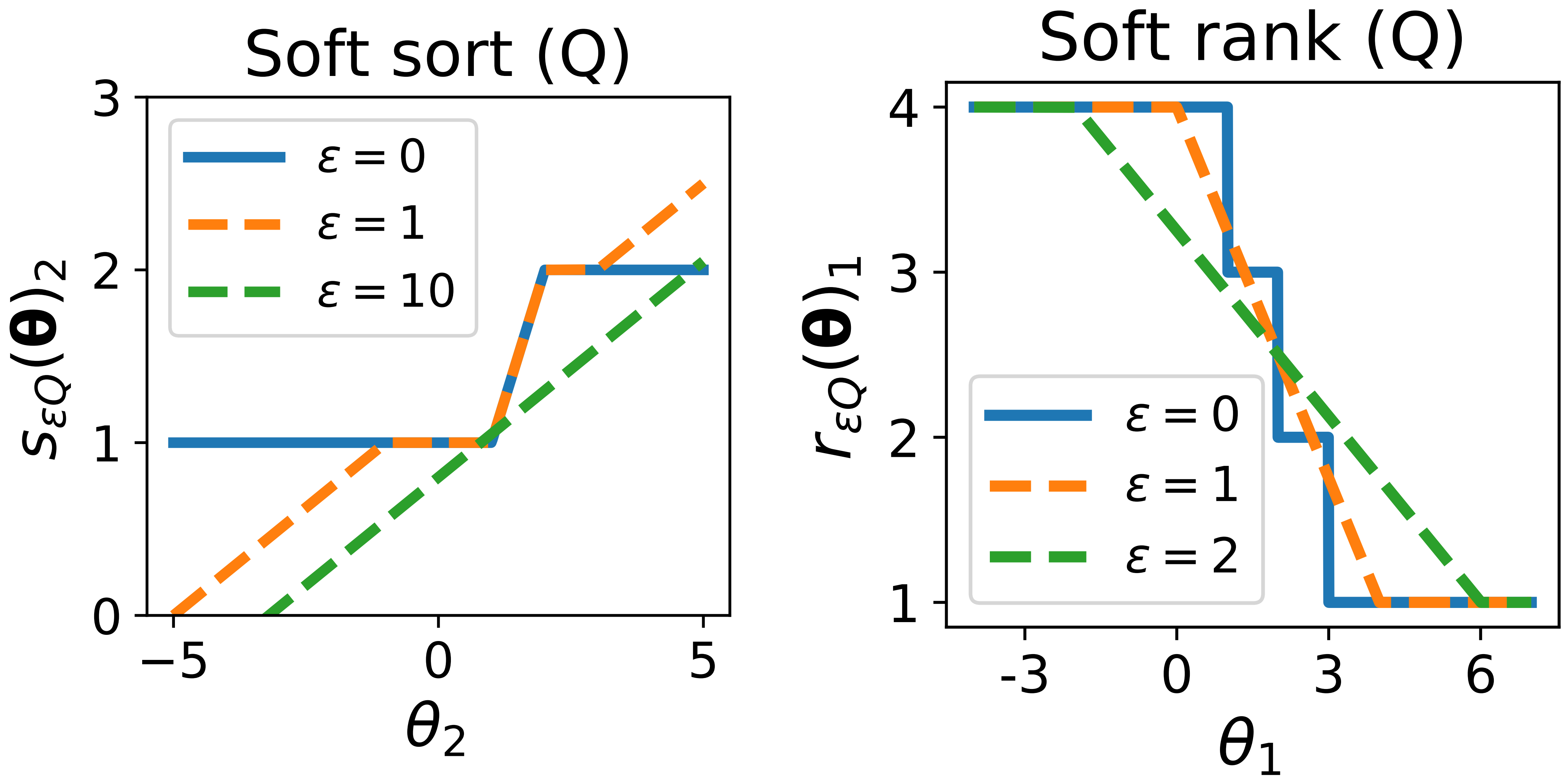
Continuity and differentiability



Continuity and differentiability



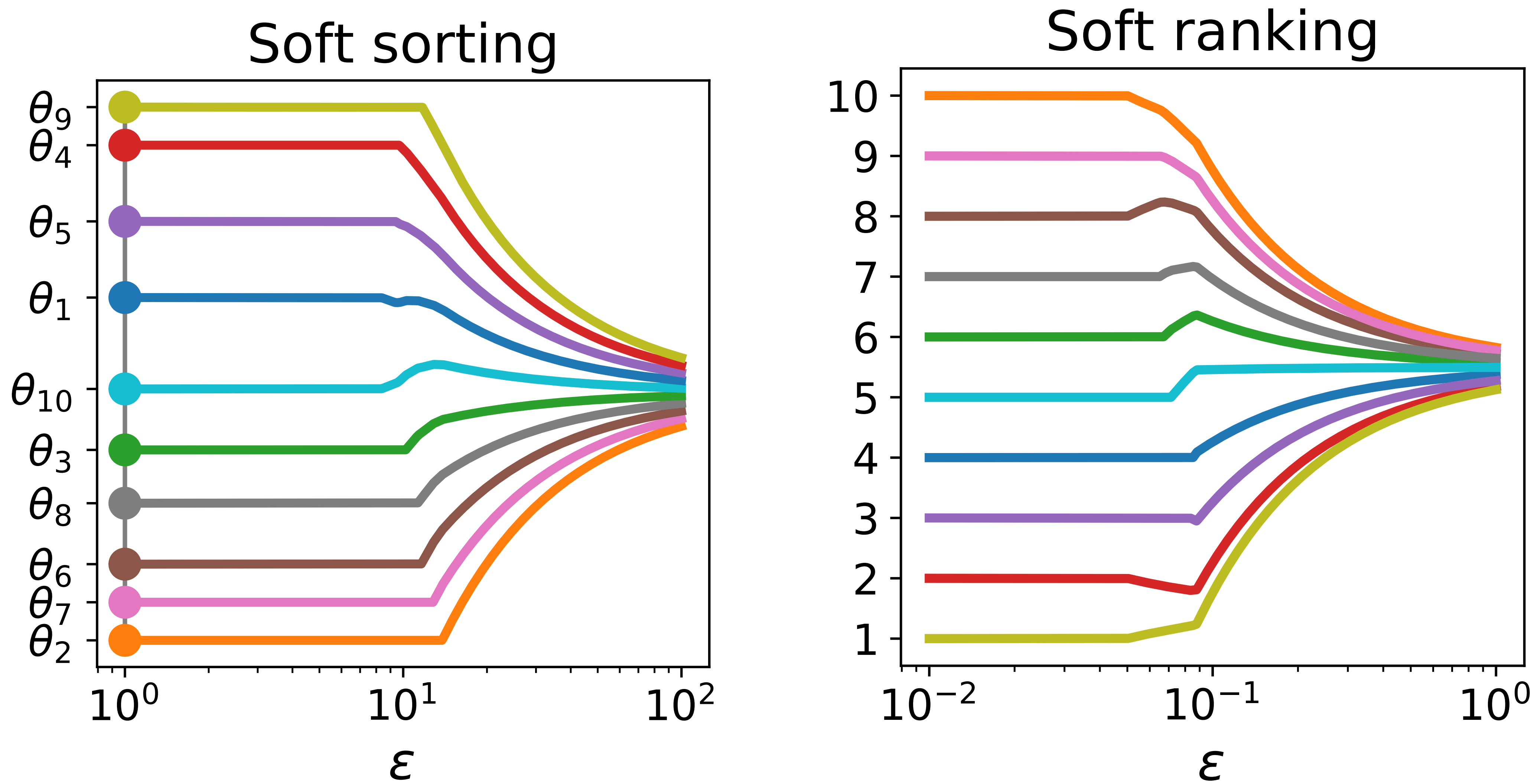
Continuity and differentiability



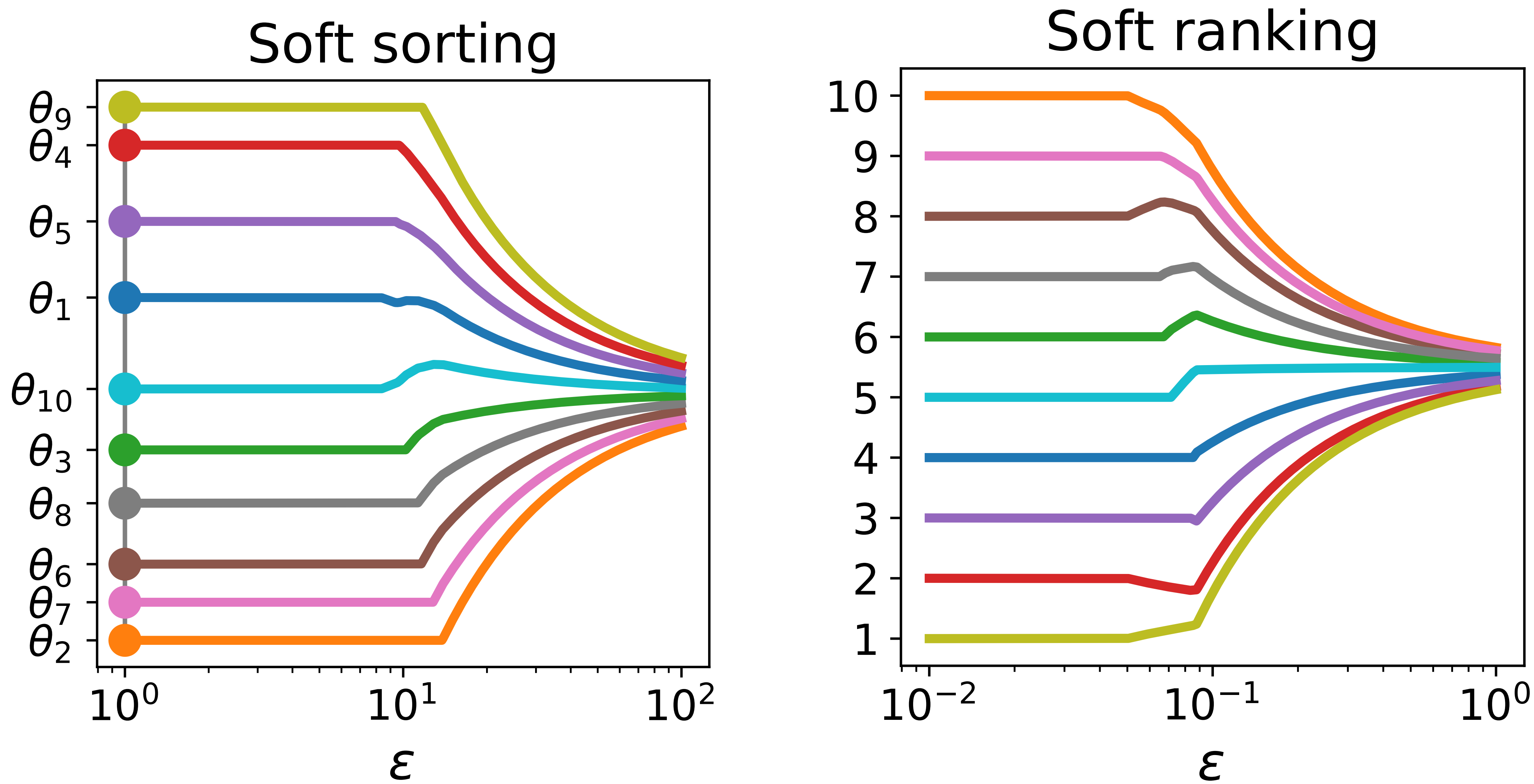
Properties

s_Q and r_Q are 1-Lipchitz continuous and differentiable almost everywhere.

Effect of regularization strength ε



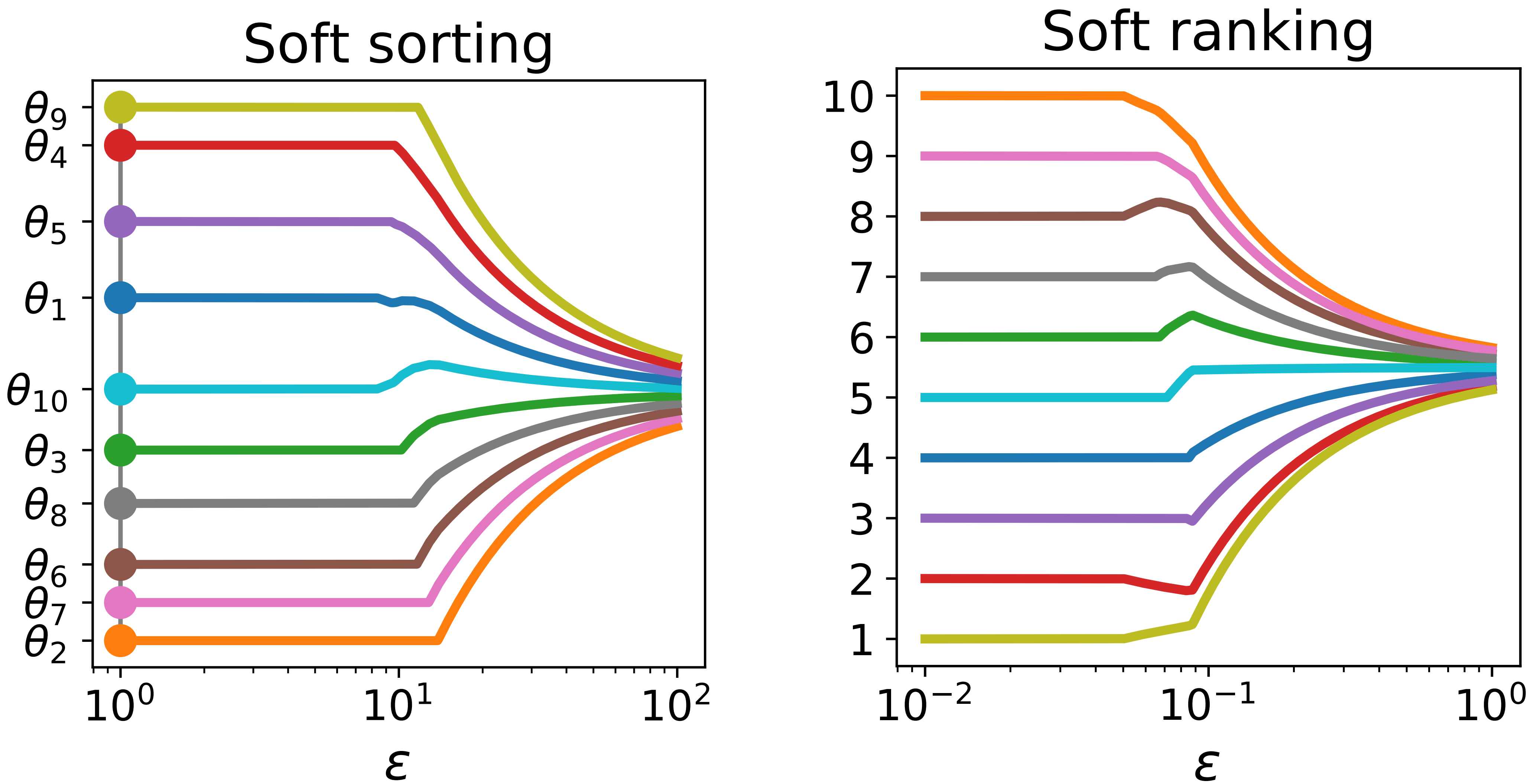
Effect of regularization strength ε



Properties

Converge to hard version when $\varepsilon \leq \varepsilon_{min}$

Effect of regularization strength ε

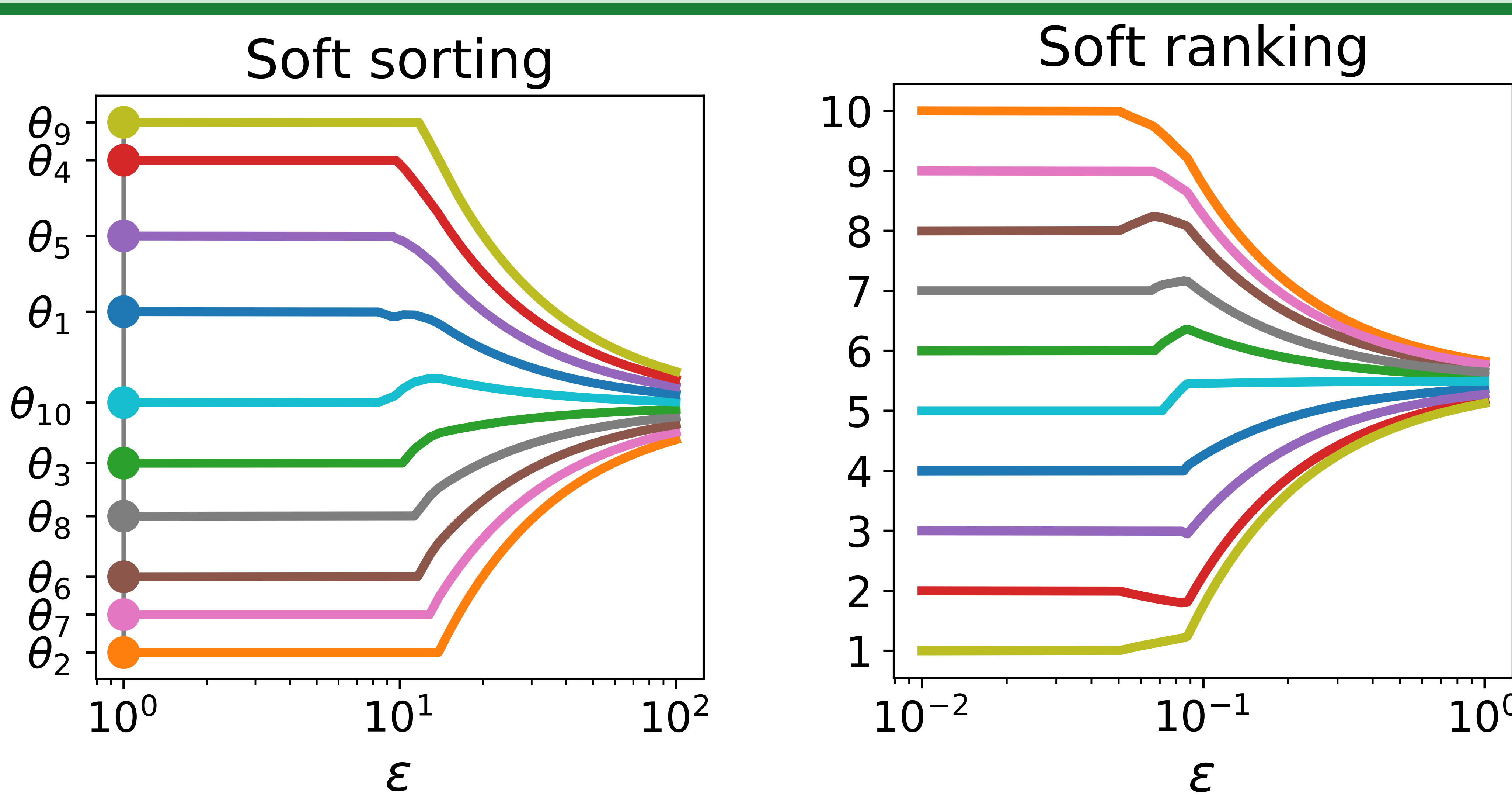


Properties

Converge to hard version when $\varepsilon \leq \varepsilon_{min}$

Collapse to a mean when $\varepsilon \to \infty$

Effect of regularization strength ε



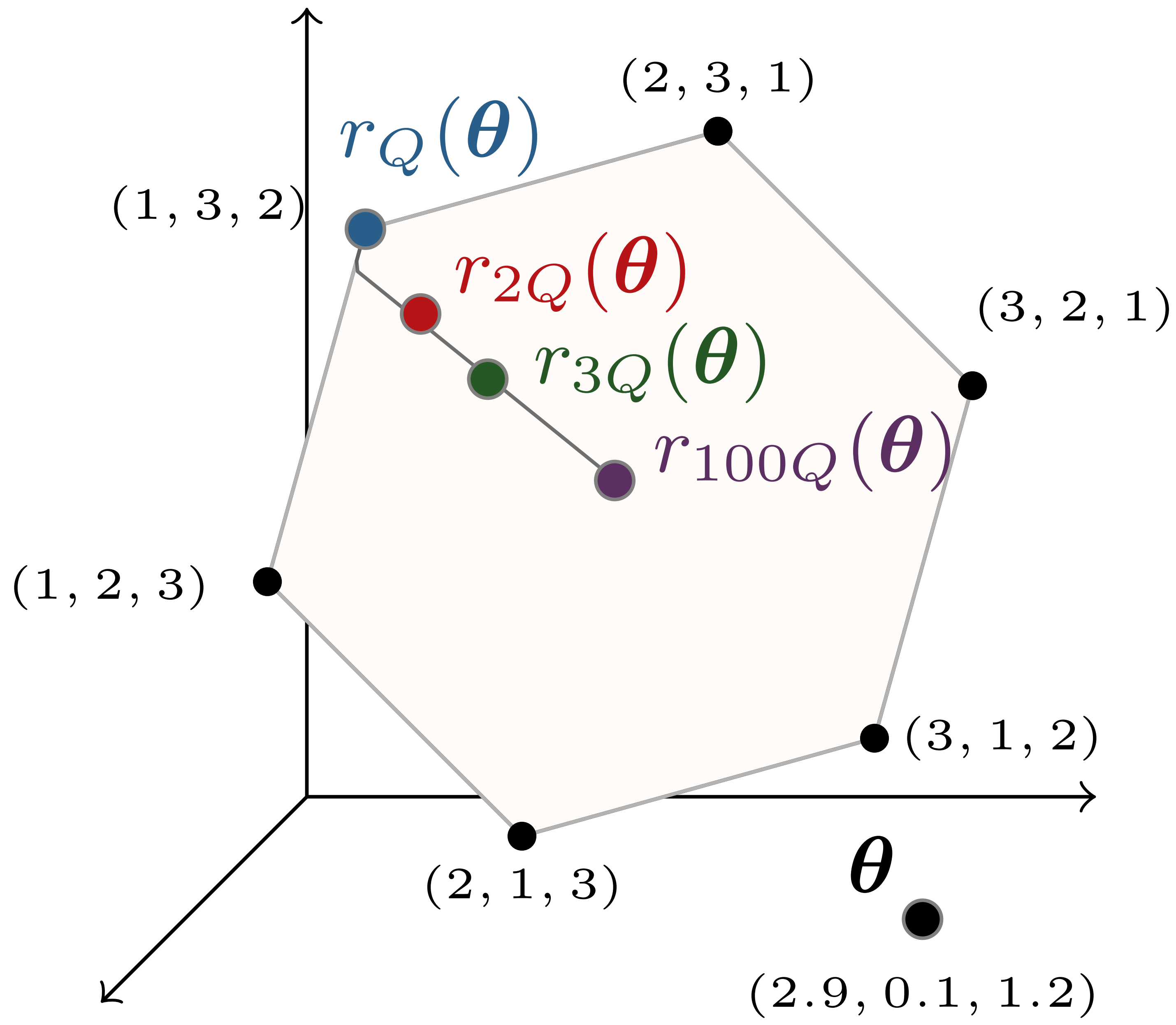
Properties

Converge to hard version when $\varepsilon \leq \varepsilon_{min}$

Collapse to a mean when $\varepsilon \to \infty$

Order preserving (paths don't cross)

Regularization path



Collapse to a mean(ρ)**1** when $\varepsilon \to \infty$

Step 3: Computation

Step 3: Computation

Reduction to isotonic regression

Proposition

$$P_Q(z, w) = z - v_Q(z_{\sigma(z)}, w)_{\sigma^{-1}(z)}$$

$$v_Q(s, w) \triangleq \arg \min_{v_1 \geq \ldots \geq v_n} \|v - (s - w)\|^2$$

Total time cost: O(n log n)

e.g. [Negrignho & Martins, 2014; Lim & Wright 2016]

Step 3: Computation

Reduction to isotonic regression

Proposition

$$P_Q(z, w) = z - v_Q(z_{\sigma(z)}, w)_{\sigma^{-1}(z)}$$

$$v_Q(s, w) \triangleq \arg \min_{v_1 \geq \ldots \geq v_n} \|v - (s - w)\|^2$$

dual solution

Total time cost: O(n log n)

e.g. [Negrignho & Martins, 2014; Lim & Wright 2016]

Step 3: Computation

Reduction to isotonic regression

primal dual relation

Proposition

$$P_Q(z, w) = z - v_Q(z_{\sigma(z)}, w)_{\sigma^{-1}(z)}$$

$$v_Q(s, w) \triangleq \arg \min_{v_1 \geq \ldots \geq v_n} \|v - (s - w)\|^2$$

dual solution

Total time cost: O(n log n)

e.g. [Negrignho & Martins, 2014; Lim & Wright 2016]

Step 3: Computation

Boils down to solving $$v^\star = \arg\min_{v_1 \geq \ldots \geq v_n} \|v - u\|^2$$ u = s - w

[Best, 2000]

Step 3: Computation

Boils down to solving $$v^\star = \arg \min_{v_1 \geq \ldots \geq v_n} \|v - u\|^2$$ u = s - w

Pool Adjacent Violators (PAV): Finds a partition $\mathscr{B}_1, \ldots, \mathscr{B}_m$ by repeatedly splitting coordinates. The worst-case cost is O(n).

[Best, 2000]

Step 3: Computation

Boils down to solving $v^\star = \arg\min_{v_1 \geq \ldots \geq v_n} \|v - u\|^2$ u = s - w

Pool Adjacent Violators (PAV): Finds a partition $\mathscr{B}_1, \ldots, \mathscr{B}_m$ by repeatedly splitting coordinates. The worst-case cost is O(n).

<u>Ex:</u>

n=6

$$\mathscr{B}_1 = \{1,2\} \qquad v_1^\star = v_2^\star = mean(u_1, u_2)$$

$$\mathscr{B}_2 = \{3\} \qquad v_3^\star = mean(u_3) = u_3$$

$$\mathscr{B}_3 = \{4,5,6\} \qquad v_4^\star = v_5^\star = v_6^\star = mean(u_4, u_5, u_6)$$

[Best, 2000]

Step 4: Differentiation

See also [Djolonga & Krause, 2017]

Step 4: Differentiation

Differentiate $v_Q(s, w) = \arg \min_{v_1 \geq \ldots \geq v_n} \|v - (s - w)\|^2$ w.r.t. s and w

See also [Djolonga & Krause, 2017]

Step 4: Differentiation

Differentiate $v_Q(s,w) = \arg\min_{v_1 \geq \ldots \geq v_n} \|v - (s-w)\|^2$ w.r.t. s and w

Proposition

$$\frac{\partial v_Q(s,w)}{\partial s} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_m \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$\mathbf{B}_j \triangleq \mathbf{1}/|\mathscr{B}_j| \in \mathbf{R}^{|\mathscr{B}_j| \times |\mathscr{B}_j|}, \quad j \in [m]$$

See also [Djolonga & Krause, 2017]

Step 4: Differentiation

Differentiate $P_Q(z, w)$ w.r.t. z and w

Step 4: Differentiation

Differentiate $P_Q(z, w)$ w.r.t. z and w

Proposition

$$\frac{\partial P_Q(z, w)}{\partial z} = J_Q(z_{\sigma(z)}, w)_{\sigma^{-1}(z)}$$

$$J_Q(s, w) \triangleq I - \frac{\partial v_Q(s, w)}{\partial s}$$

Multiplication with the Jacobian in O(n) time and space (see paper)

Background

Proposed method

Experimental results

Robust regression

Robust regression

Least squares (LS)

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \ell_i(w)$$

i^th loss

$$\ell_i(w) \triangleq \frac{1}{2}(y_i - g_w(x_i))^2$$

Robust regression

Least squares (LS)

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell_i(w)$$

ith loss

$$\ell_i(w) \triangleq \frac{1}{2}(y_i - g_w(x_i))^2$$

Soft Least trimmed squares (SLTS)

$$\min_w \frac{1}{n-k} \sum_{i=k+1}^{n} \ell_i^{\varepsilon}(w)$$

ith "soft sorted" loss

$$\ell_i^{\varepsilon}(w) \triangleq [s_{\varepsilon Q}(\ell(w))]_i$$

Robust regression

Least squares (LS)

ith loss

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell_i(w) \qquad \ell_i(w) \triangleq \frac{1}{2}(y_i - g_w(x_i))^2$$

Soft Least trimmed squares (SLTS)

ith "soft sorted" loss

$$\min_w \frac{1}{n-k} \sum_{i=k+1}^{n} \ell_i^{\varepsilon}(w) \qquad \ell_i^{\varepsilon}(w) \triangleq [s_{\varepsilon Q}(\ell(w))]_i$$

$\varepsilon \to 0 \quad SLTS \to LTS$

Robust regression

Least squares (LS)

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell_i(w)$$

ith loss

$$\ell_i(w) \triangleq \frac{1}{2}(y_i - g_w(x_i))^2$$

Soft Least trimmed squares (SLTS)

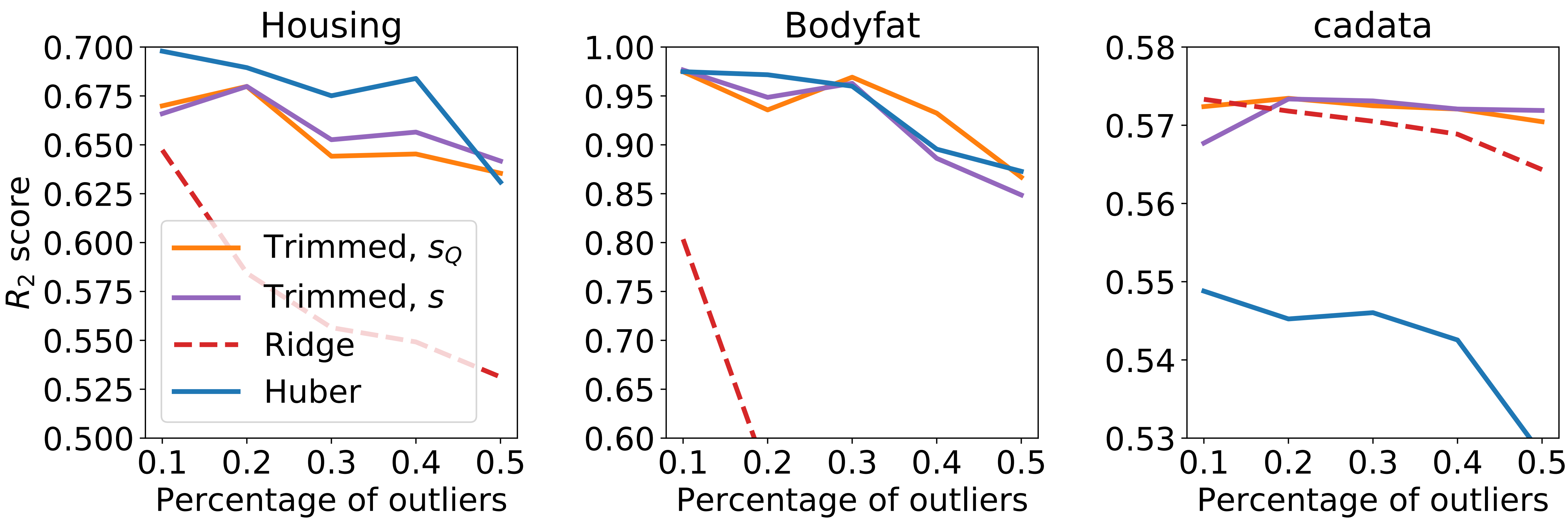
$$\min_w \frac{1}{n-k} \sum_{i=k+1}^{n} \ell_i^{\varepsilon}(w)$$

ith "soft sorted" loss

$$\ell_i^{\varepsilon}(w) \triangleq [s_{\varepsilon Q}(\ell(w))]_i$$

$\varepsilon \to 0 \quad SLTS \to LTS \qquad \varepsilon \to \infty \quad SLTS \to LS$

Robust regression



Evaluation: 10-fold CV

Hyper-parameter selection: 5-fold CV

Top-k classification

$$\ell : [n] \times \mathbb{R}^n \to \mathbb{R}_+$$

Cuturi et al. [2019]

Ground truth

Predicted soft ranks

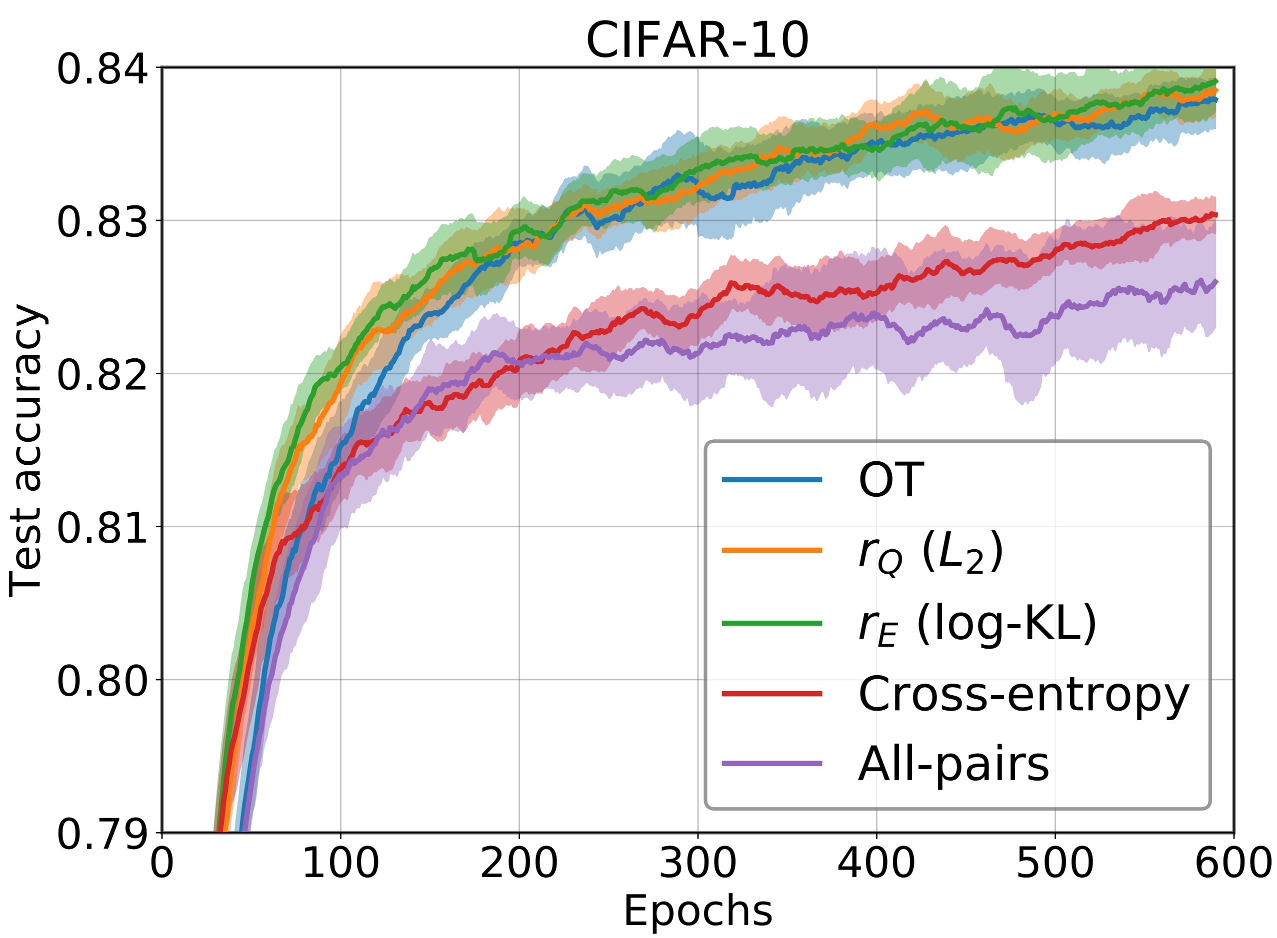
Top-k classification

$$\ell : [n] \times \mathbb{R}^n \to \mathbb{R}_+$$

Cuturi et al. [2019]

Ground truth

Predicted soft ranks



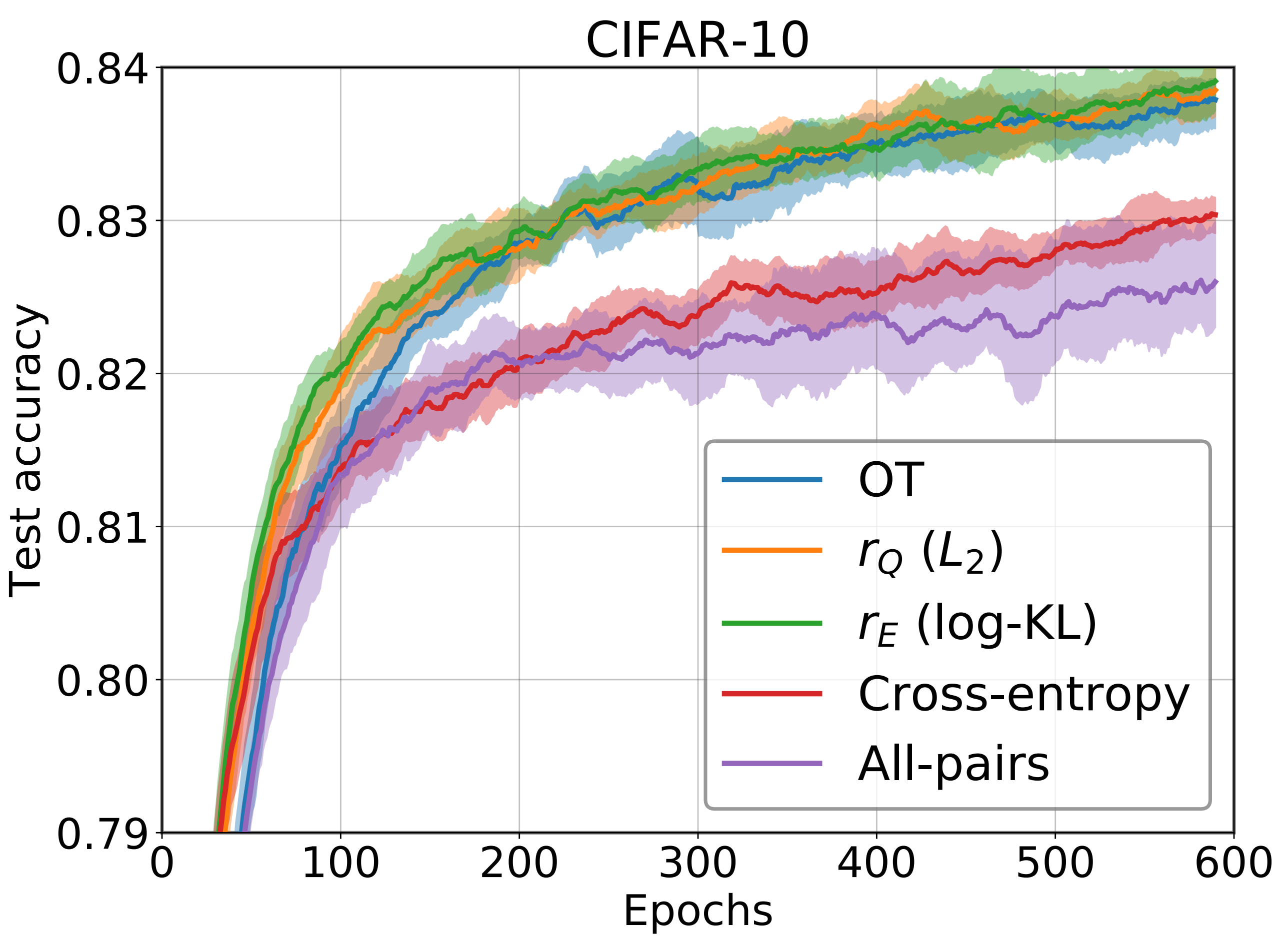
Top-k classification

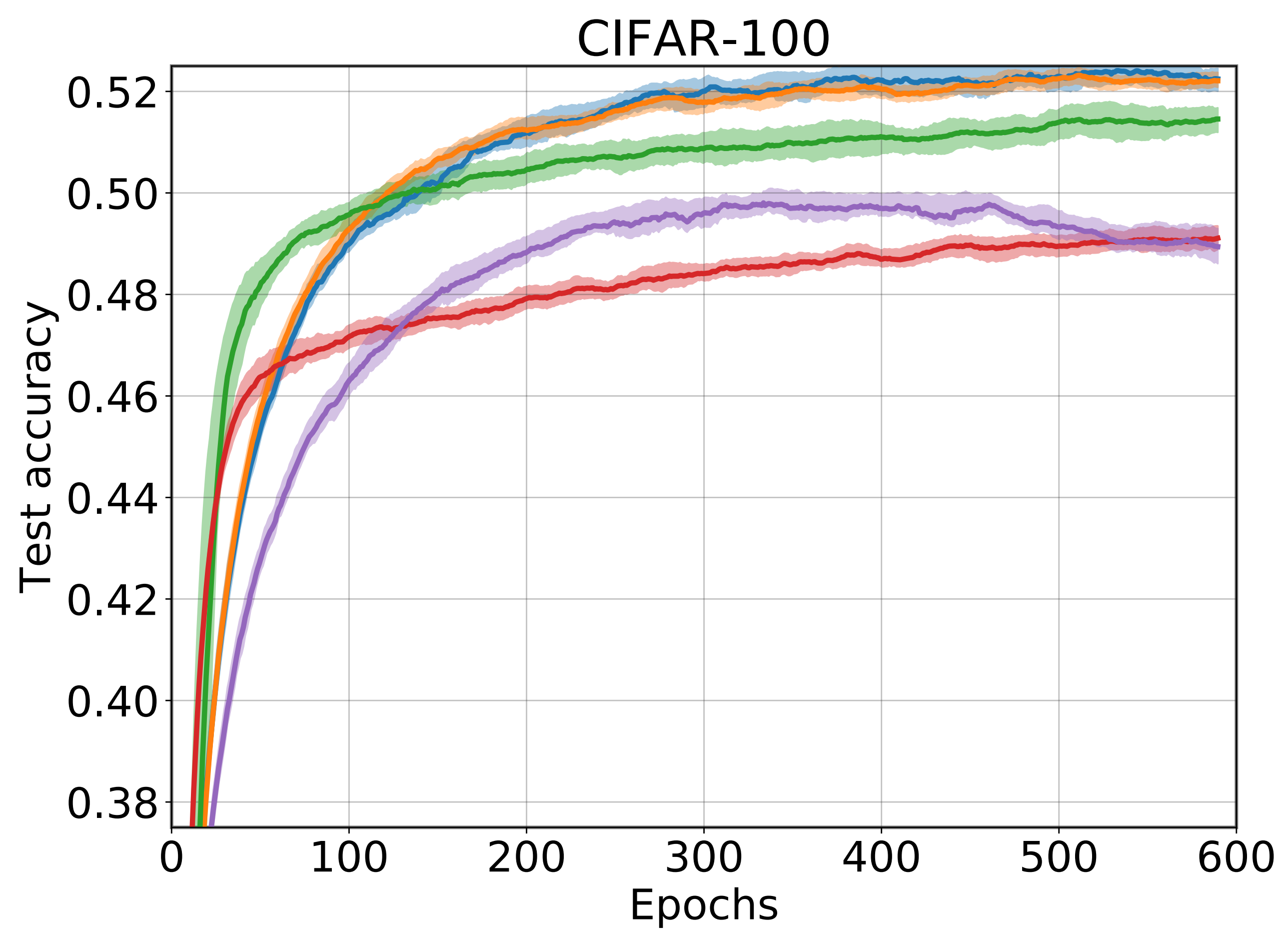
$$\ell : [n] \times \mathbb{R}^n \to \mathbb{R}_+$$

Cuturi et al. [2019]

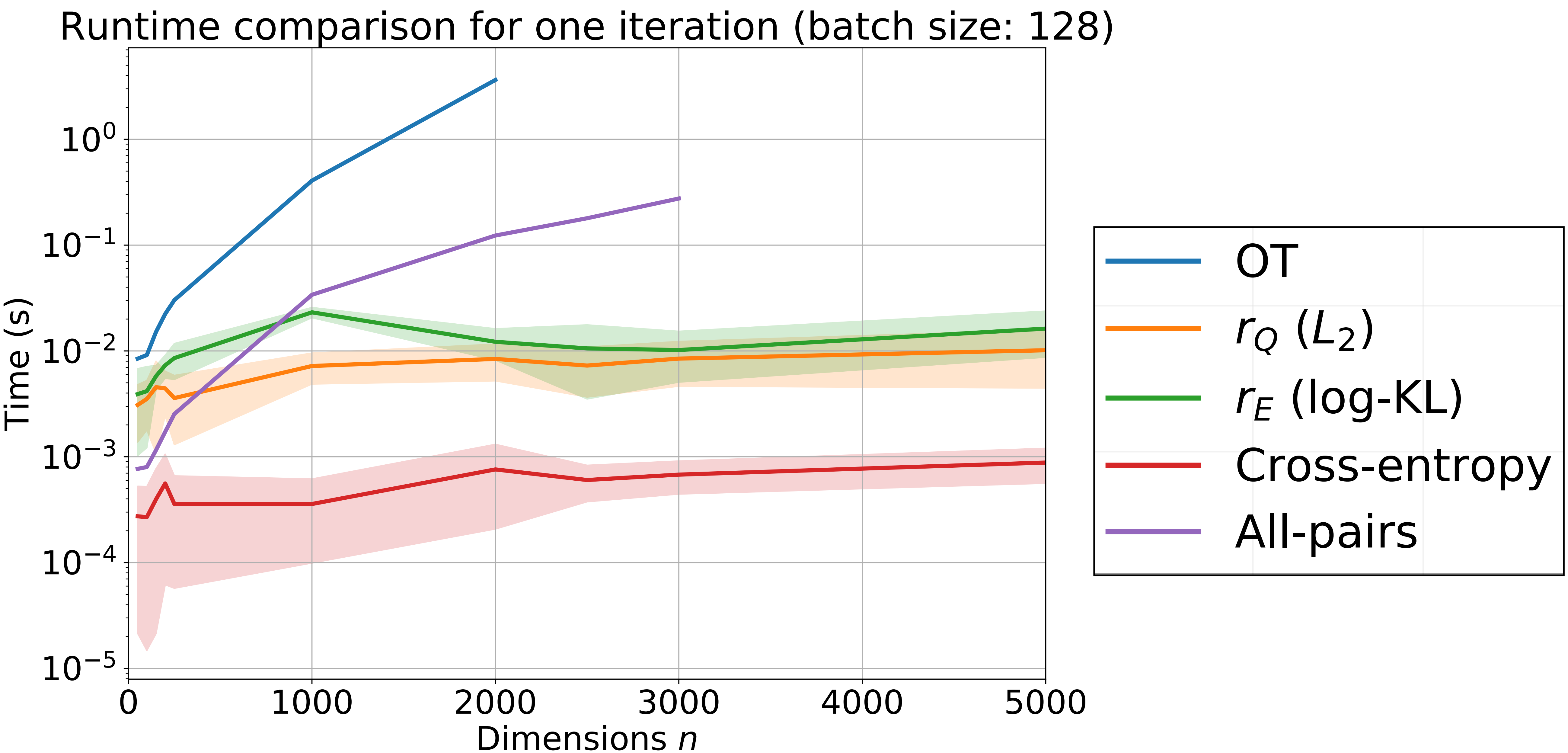
Ground truth

Predicted soft ranks





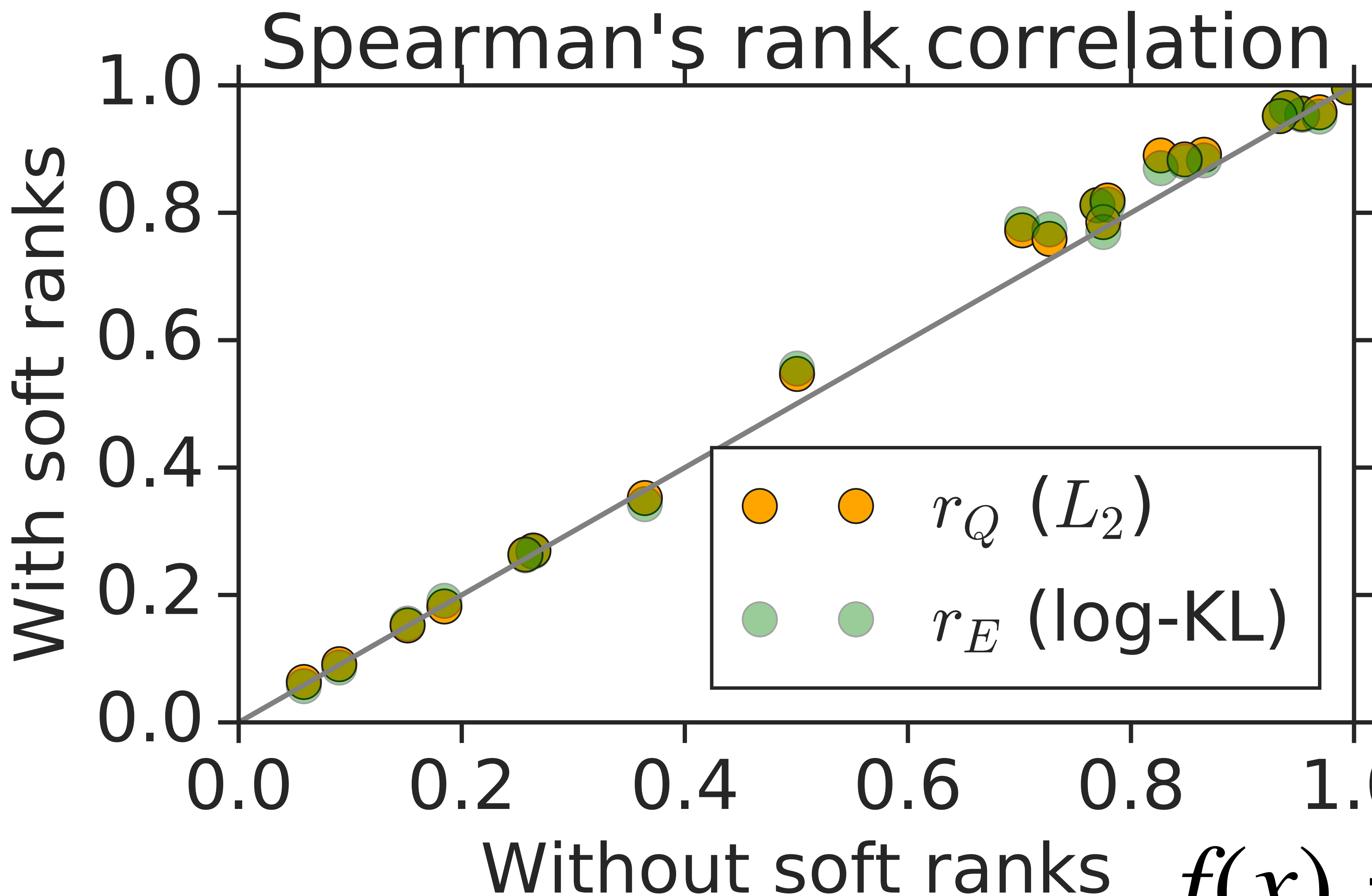
Speed benchmark



Label ranking experiment

$$\ell_i \triangleq \frac{1}{2}\|y_i - f(x_i)\|^2 \qquad y_i \in \Sigma$$

$$f(x) = r_Q(g(x))$$



$$f(x) = g(x)$$

Comparison on **21** datasets, 5-fold CV

Summary

Summary

- We proposed sorting and ranking relaxations with O(n log n) computation and O(n) differentiation

Summary

- We proposed sorting and ranking relaxations with O(n log n) computation and O(n) differentiation

- Key techniques: projections onto the permutahedron and reduction to isotonic optimization

Summary

- We proposed sorting and ranking relaxations with O(n log n) computation and O(n) differentiation

- Key techniques: projections onto the permutahedron and reduction to isotonic optimization

- Applications to least trimmed squares, top-k classification and label ranking

Summary

- We proposed sorting and ranking relaxations with O(n log n) computation and O(n) differentiation
- Key techniques: projections onto the permutahedron and reduction to isotonic optimization
- Applications to least trimmed squares, top-k classification and label ranking

Preprint: Fast Differentiable Sorting and Ranking [arXiv:2002.08871]

Code: coming soon!