Differentiable Ranks and Sorting using Optimal Transport

Marco Cuturi

To be presented at NeurIPS 2019

Google AI Brain Team

O. Teboul

J.P. Vert

https://arxiv.org/abs/1905.11885

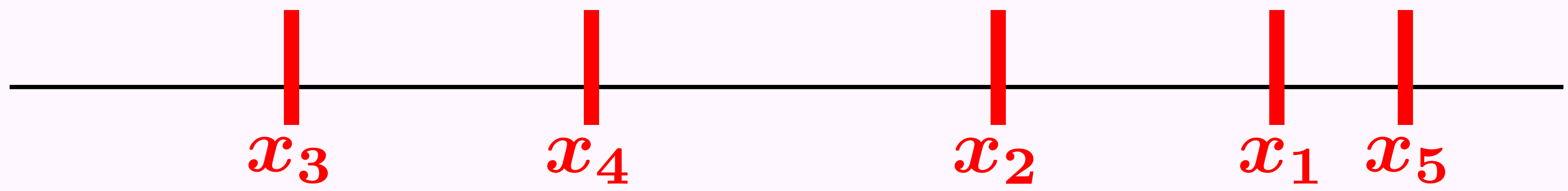
Sorting Permutations

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$

$$\sigma(\mathbf{x}) = (3, 4, 2, 1, 5)$$

Sorting Permutations

$$\mathbf{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}, \boldsymbol{x_5})$$



$$\sigma(\mathbf{x}) = (3, 4, 2, 1, 5)$$

Written as **permutation matrix**

$$\Pi_{\boldsymbol{\sigma}} = \mathrm{sp}\mathbf{1}\left((i, \boldsymbol{\sigma_i})_i\right) \qquad \Pi_{\sigma(\mathbf{x})} = \begin{bmatrix} & & 1 & & \\ & & & 1 & \\ & 1 & & & \\ 1 & & & & \\ & & & & 1 \end{bmatrix}$$

$$\Pi_{\sigma^{-1}} = \Pi_{\sigma}^{T}$$

Ranks and Sort operator

Ranking / Sorting

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$

$$\sigma(\mathbf{x}) = (3, 4, 2, 1, 5)$$

$$R\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \\ 5 \end{bmatrix} := \sigma(\mathbf{x})^{-1}$$

$$S\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} x_3 \\ x_4 \\ x_2 \\ x_1 \\ x_5 \end{bmatrix} := \mathbf{x}_{\sigma(\mathbf{x})}$$

Sorting as a Subroutine in ML

k-NN

(1) select neighbours
(2) majority vote

Least Quantile regression

minimize empirical
quantile of loss (not mean)

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

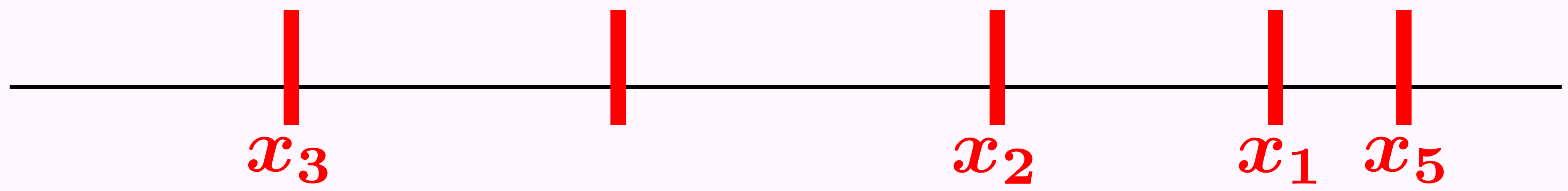
Rank-based statistics

data viewed as ranks

Non-differentiability of ranks and sort

Ranking / Sorting

$$\mathbf{x} = (x_1, x_2, x_3, x_4 + \tau, x_5)$$

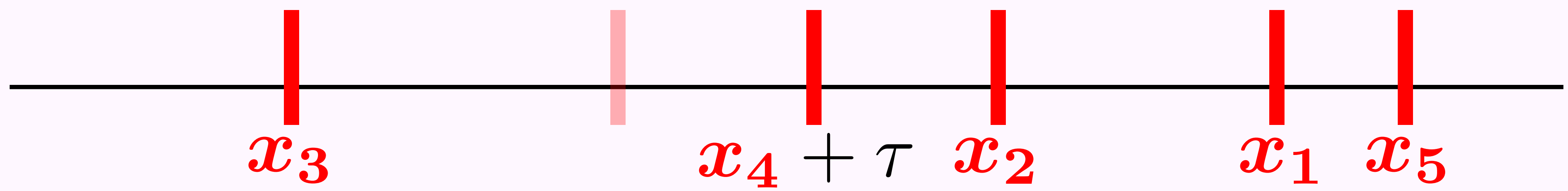


$$R\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 + \tau \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \\ 5 \end{bmatrix} \qquad S\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 + \tau \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} x_3 \\ x_4 + \tau \\ x_2 \\ x_1 \\ x_5 \end{bmatrix}$$

Non-differentiability of ranks and sort

Ranking / Sorting

$$\mathbf{x} = (x_1, x_2, x_3, x_4 + \tau, x_5)$$

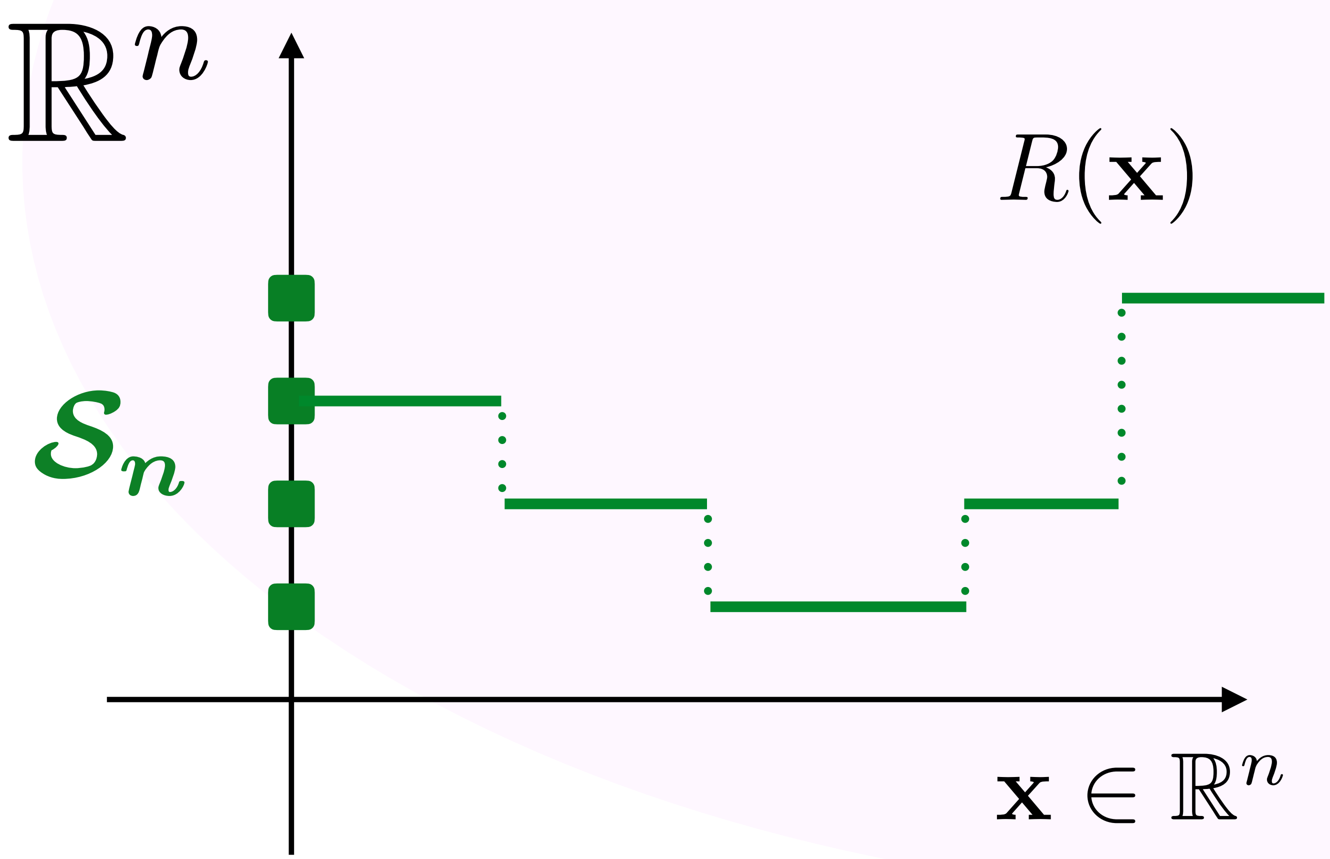


$$R\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 + \tau \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \\ 5 \end{bmatrix} \qquad S\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 + \tau \\ x_5 \end{bmatrix}\right) = \begin{bmatrix} x_3 \\ x_4 + \tau \\ x_2 \\ x_1 \\ x_5 \end{bmatrix}$$

Non-differentiability of ranks and sort

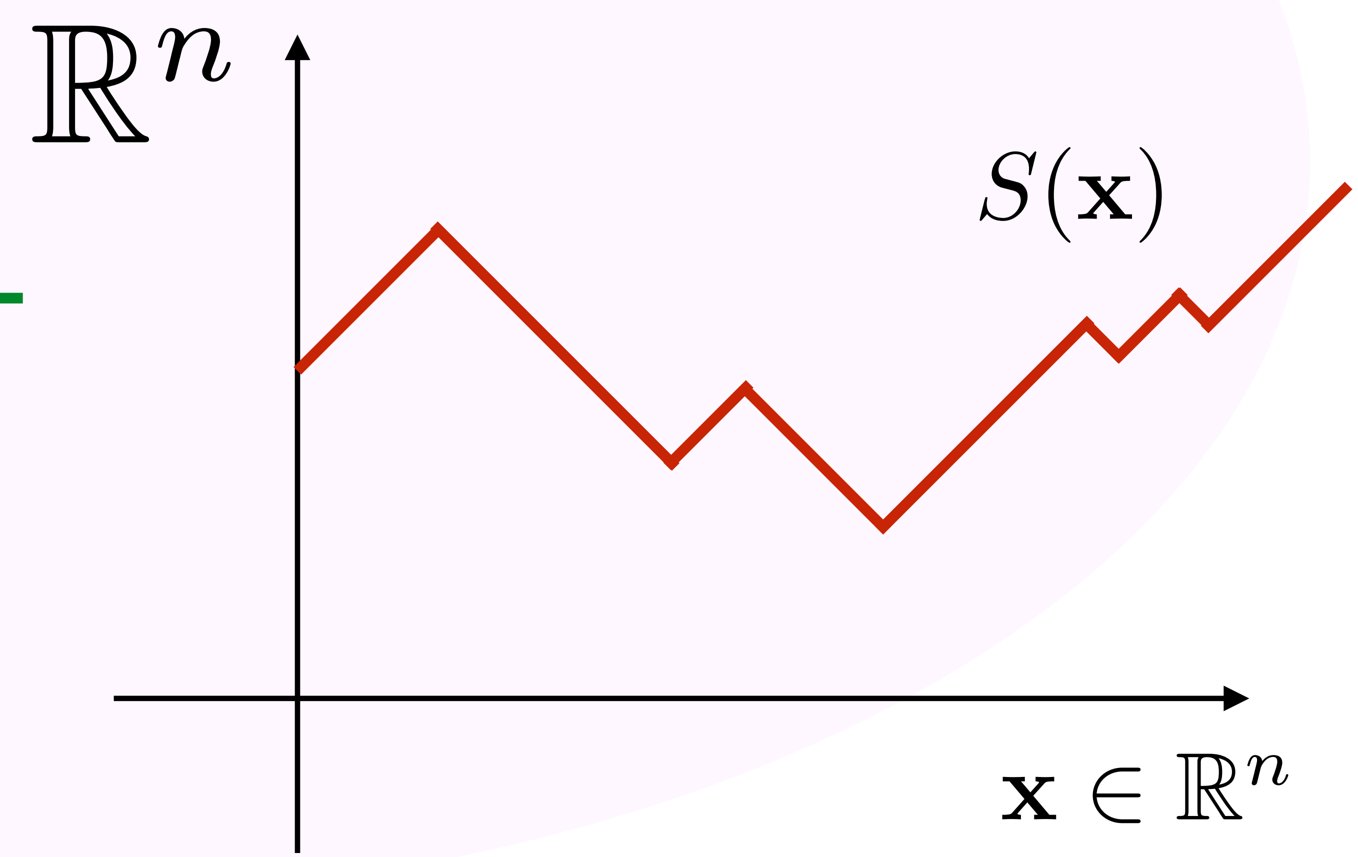
Ranking / Sorting

piecewise constant



$JR(\mathbf{x}) \in \{0, \pm\infty\}^{n \times n}$

piecewise linear



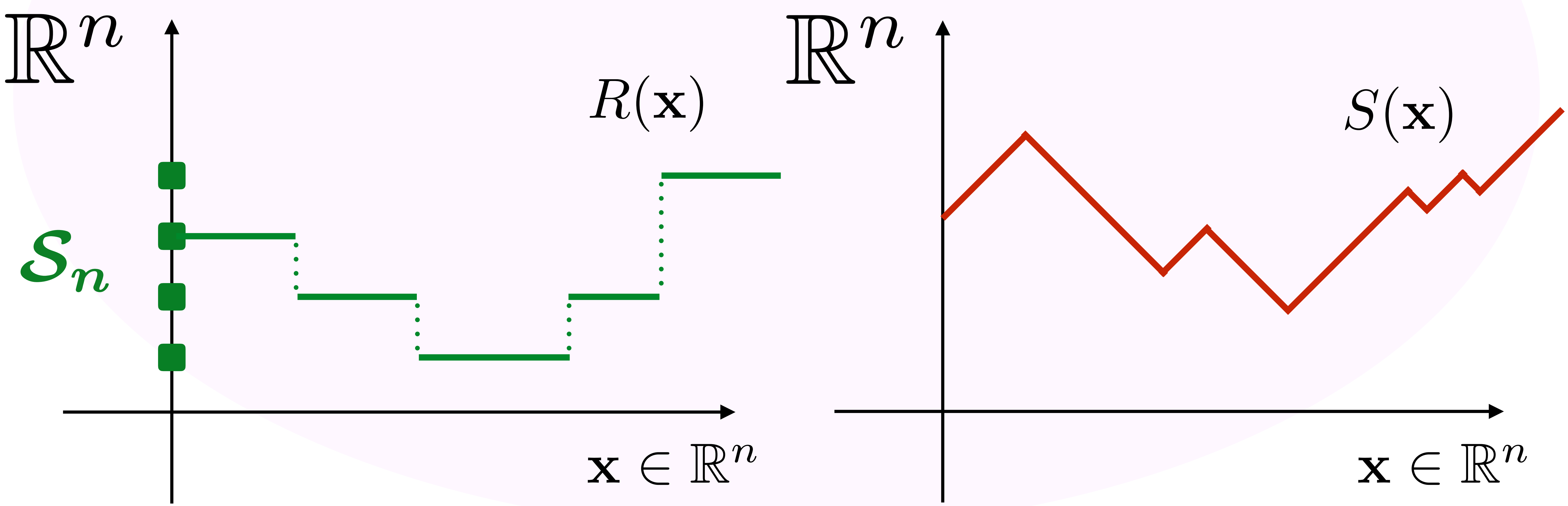
$JS(\mathbf{x}) = \Pi^T_{\sigma(\mathbf{x})}$ a.e.

Non-smoothness of ranks/sorts

- *k-NN* LMNN [Weinberger & Saul'06] Neural [Plötz & Roth'18]
- *Learning to rank*
  - **Pairwise losses:** Ranknet[Burges+'05], Lambdarank[Burges+'07], Rankboost [Freund+'03], BoltzRank [Volkovs & Zemel'09]
  - **Smoothed NDCG:** SoftRanks [Taylor+'08][Chapelle & Wu'09']
- *Multiclass classification* XE on Softmax activations, smoothed top-*k* losses [Boyd+'12][Berrada+'18] Focal loss [Lin+17]
- *Trimmed Regression* Combinatorial approaches

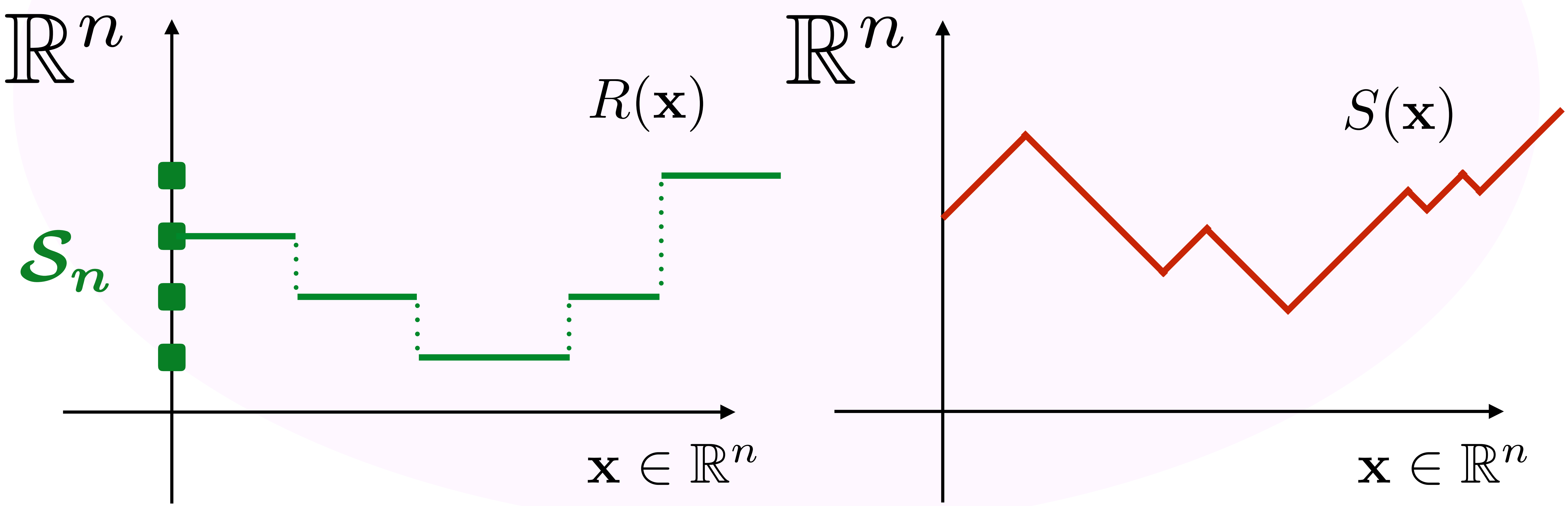
Soft Ranks and Sort Operators

Ranking / Sorting



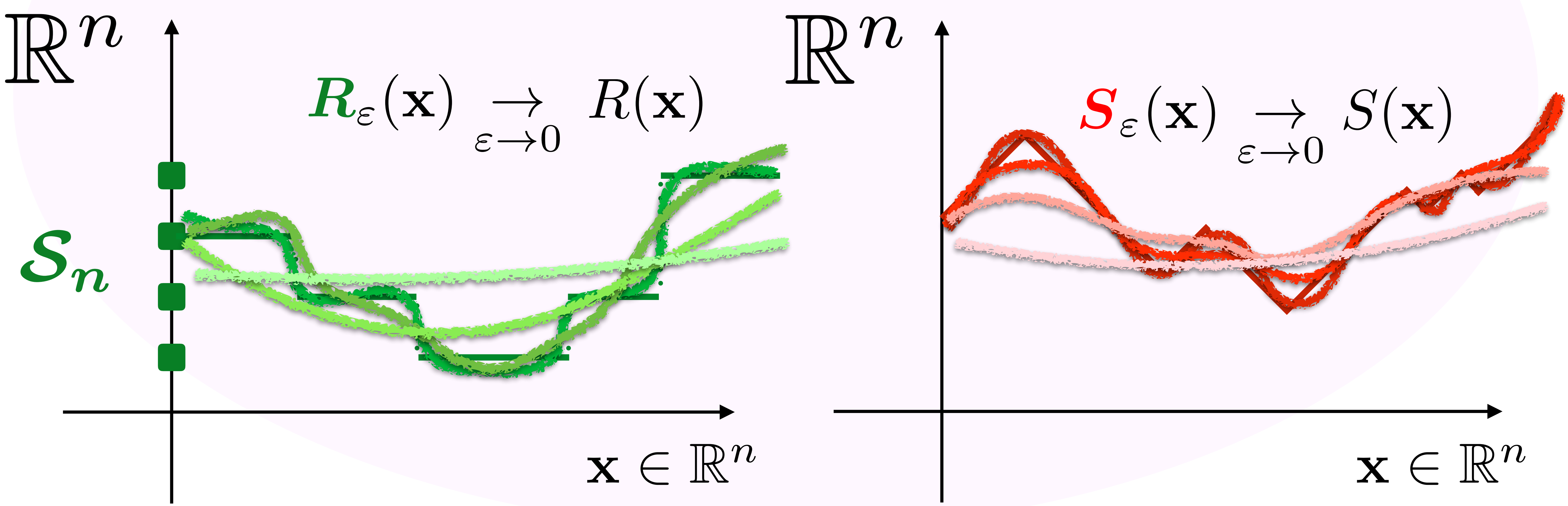
Soft Ranks and Sort Operators

Goal: define (programatically) everywhere differentiable approximation functions for *R/S*, **arbitrarily close** to the true *R/S* vector outputs



Soft Ranks and Sort Operators

Goal: define (programatically) everywhere differentiable approximation functions for *R*/*S*, **arbitrarily close** to the true *R*/*S* vector outputs

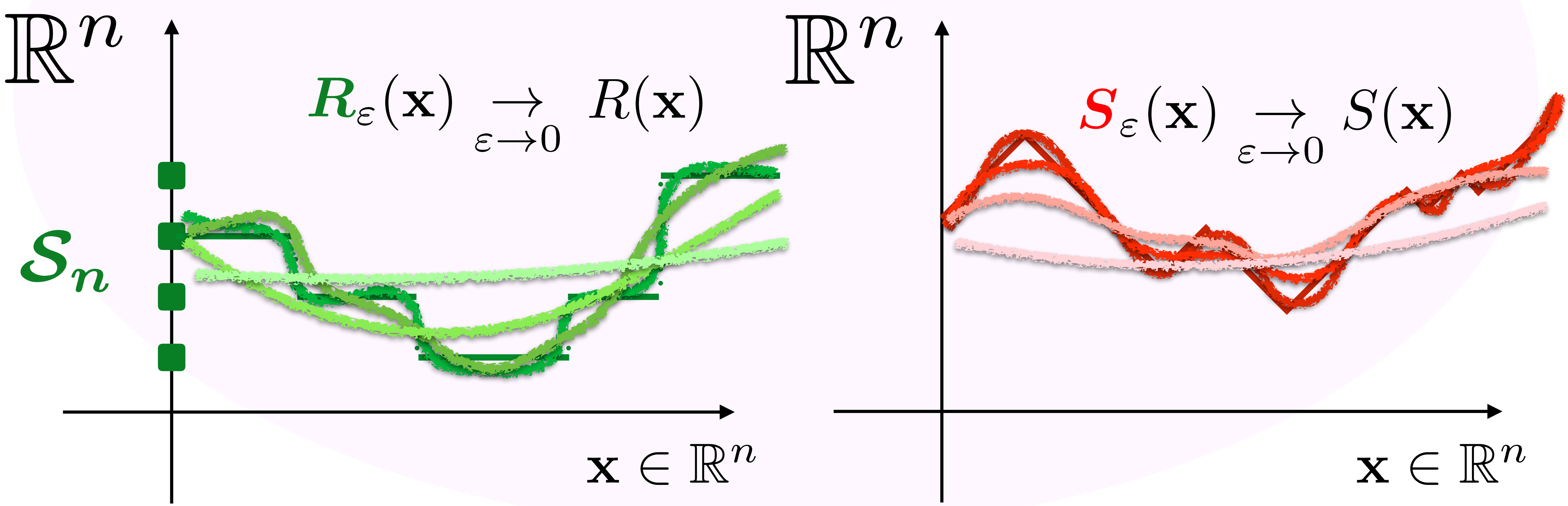


Soft Ranks and Sort Operators

Why?

Now Train the way you test + ε

Next Constraints in real life are *relative* (fairness)



Related work

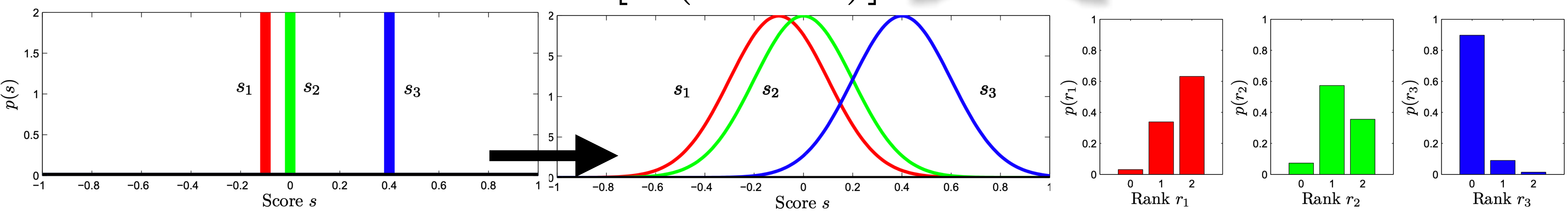
SoftRank: Optimising Non-Smooth Rank Metrics

Mike Taylor, John Guiver, Stephen Robertson, Tom Minka
WSDM 2008 | February 2008

$\mathbb{E}_{\boldsymbol{Z}}[R(\boldsymbol{x} + \boldsymbol{Z})]$

$O(n^3)$

Approximation with documented pathologies



A General Approximation Framework for Direct Optimization of Information Retrieval Measures

Tao Qin, Tie-Yan Liu, Hang Li
MSR-TR-2008-164 | November 2008

$O(n^2)$

$$R(\mathbf{x})_i = \sum_j \mathbf{1}_{x_i \geq x_j} \approx \sum_j \frac{1}{1 + e^{-(x_i - x_j)/\tau}}$$

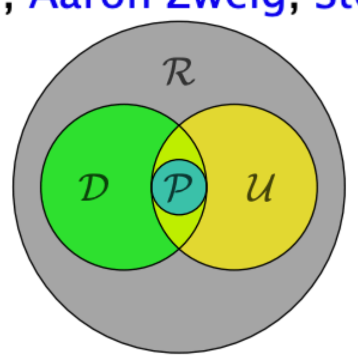
Stochastic Optimization of Sorting Networks via Continuous Relaxations

Aditya Grover, Eric Wang, Aaron Zweig, Stefano Ermon

Published as a conference paper at ICLR 2019

$O(n^2)$

$$\begin{pmatrix} 0 & 1/2 & 1/2 \\ 7/16 & 3/16 & 3/8 \\ 9/16 & 5/16 & 1/8 \end{pmatrix}$$



$$\begin{pmatrix} 3/8 & 1/8 & 1/2 \\ 3/4 & 1/4 & 0 \\ 1/4 & 1/2 & 1/4 \end{pmatrix}$$

Doubly stochastic

Unimodal row matrices

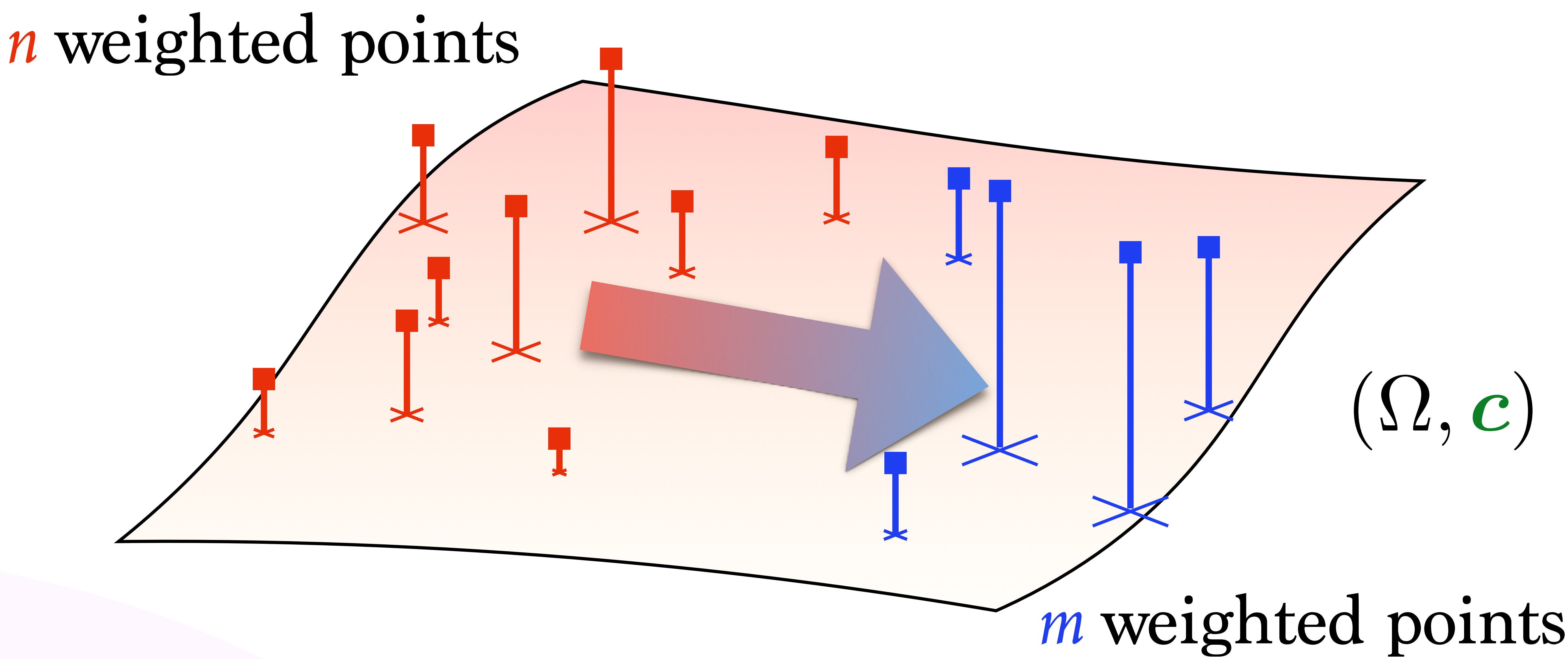
$$A_{\mathbf{s}}[i, j] = |s_i - s_j|.$$

$$\widehat{P}_{\mathtt{sort}(\mathbf{s})}[i, :](\tau) = \text{soft max}\left[((n + 1 - 2i)\mathbf{s} - A_{\mathbf{s}}\mathbb{1})/\tau\right]$$

Our Approach builds on OT

Optimal Transport

$O((n+m)nm \log(n+m))$

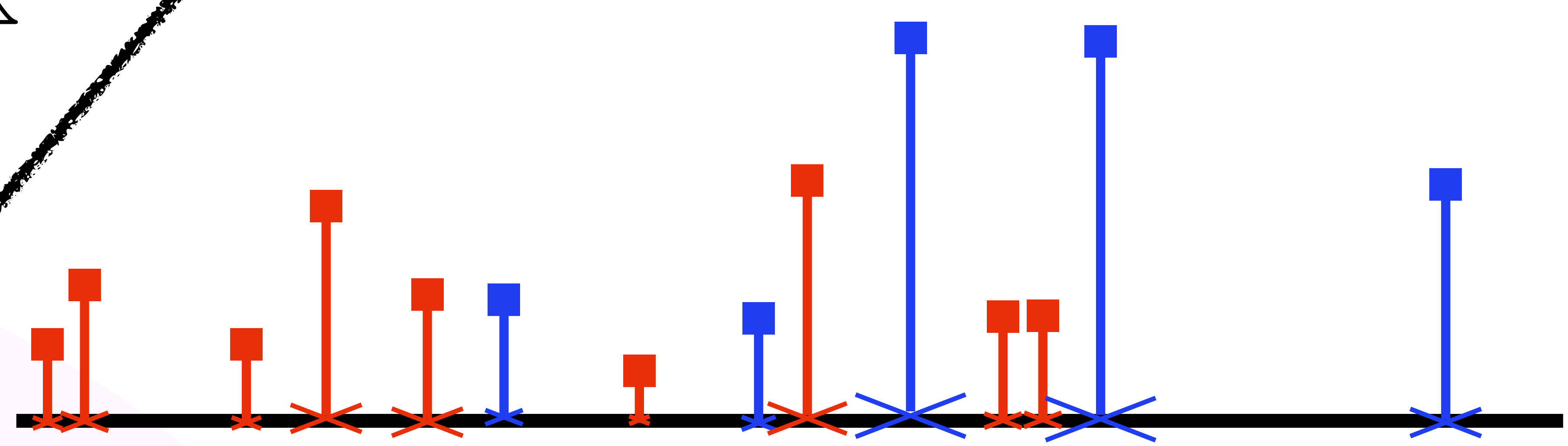


OT in 1D needs sorting

Optimal Transport

$O((n+m)nm\log(n+m))$

important to solve OT when $\Omega = \mathbb{R}$



$\Omega = \mathbb{R}$

g / Sorting

$O(n \log n)$

$+$

$O(m \log m)$

Our idea in one slide

Optimal Transport

$O((n+m)nm \log(n+m))$

important to solve OT when $\Omega = \mathbb{R}$

g / Sorting

$O(n \log n)$

$+$

$O(m \log m)$

Our idea in one slide

Optimal Transport

$O((n+m)nm \log(n+m))$

generalize both using OT (overkill!!)

g / Sorting

$O(n \log n)$

+

$O(m \log m)$

Our idea in one slide

Optimal Transport

$O((n+m)nm \log(n+m))$

generalize both using OT (overkill!!)

faster and differentiable variant.

g / Sorting

$O(n \log n)$

+

$O(m \log m)$

Regularized OT

Sinkhorn Algorithm

$O(nm)$

Our idea in one slide

Optimal Transport

$O((n+m)nm \log(n+m))$

generalize both using OT (overkill!!)

faster and differentiable variant.

Differentiable sorting in $O(nm)$

Regularized OT

Sinkhorn Algorithm

$O(nm)$

g / Sorting

$O(n \log n)$

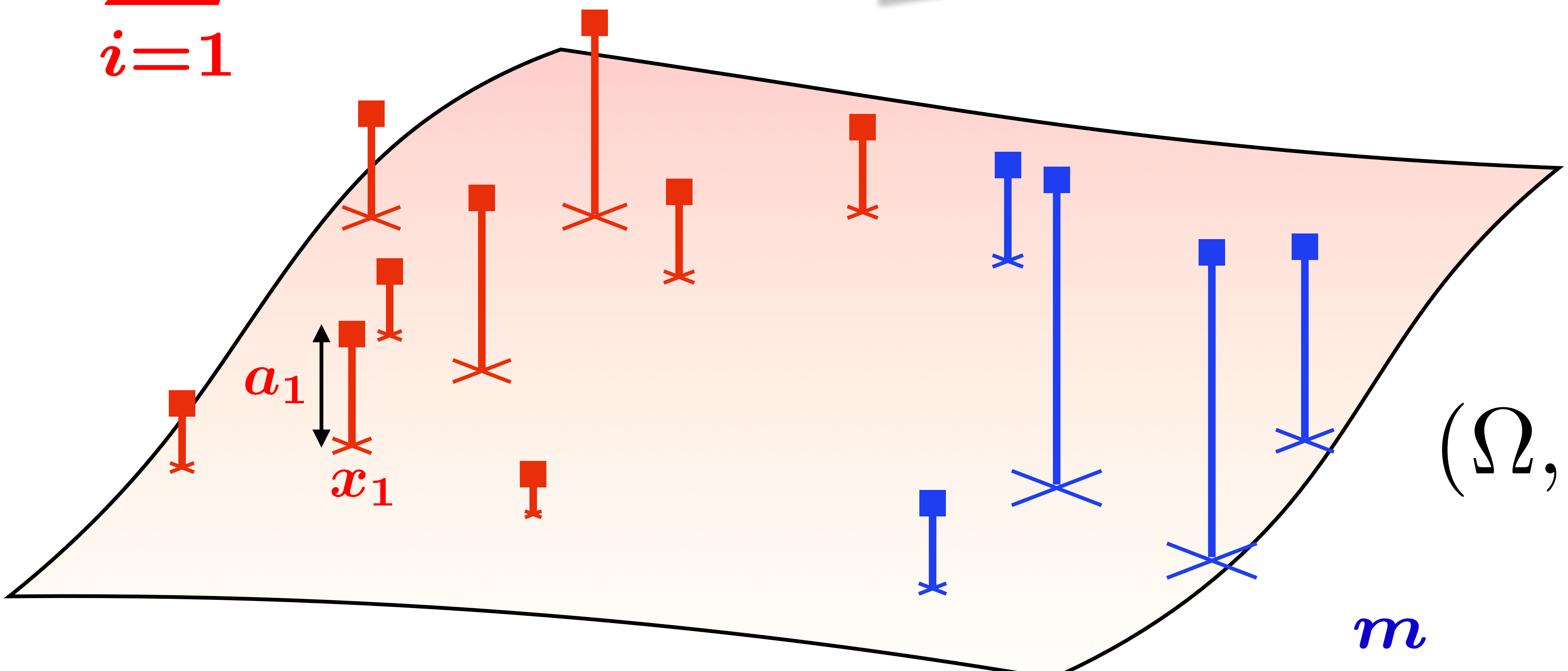
+

$O(m \log m)$

Discrete OT Problem

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

$$\sum_i a_i = \sum_j b_j = 1$$



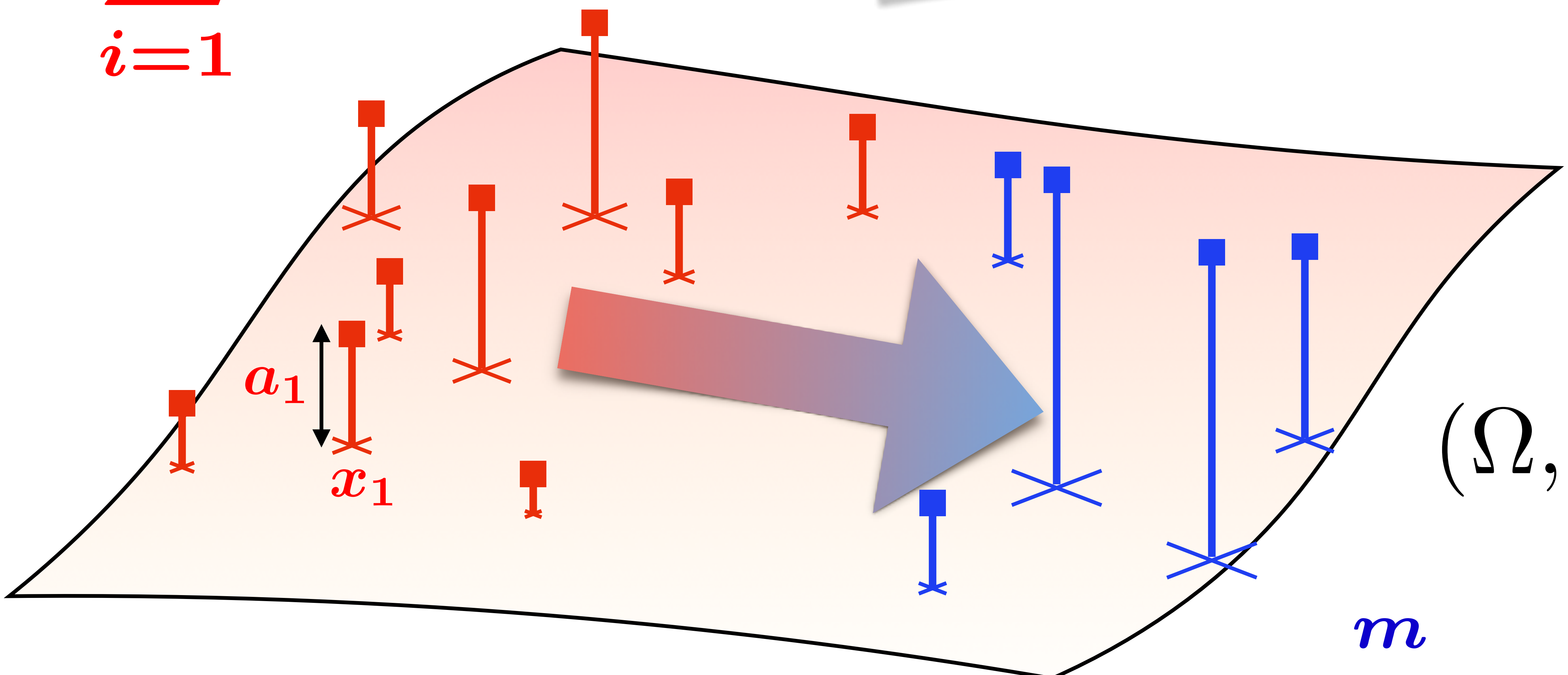
(Ω, c)

$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Discrete OT Problem

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

$\sum_i a_i = \sum_j b_j = 1$



$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

LP Formulation

Consider $\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{\delta}_{\boldsymbol{x}_i}$ and $\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b}_j \boldsymbol{\delta}_{\boldsymbol{y}_j}$.

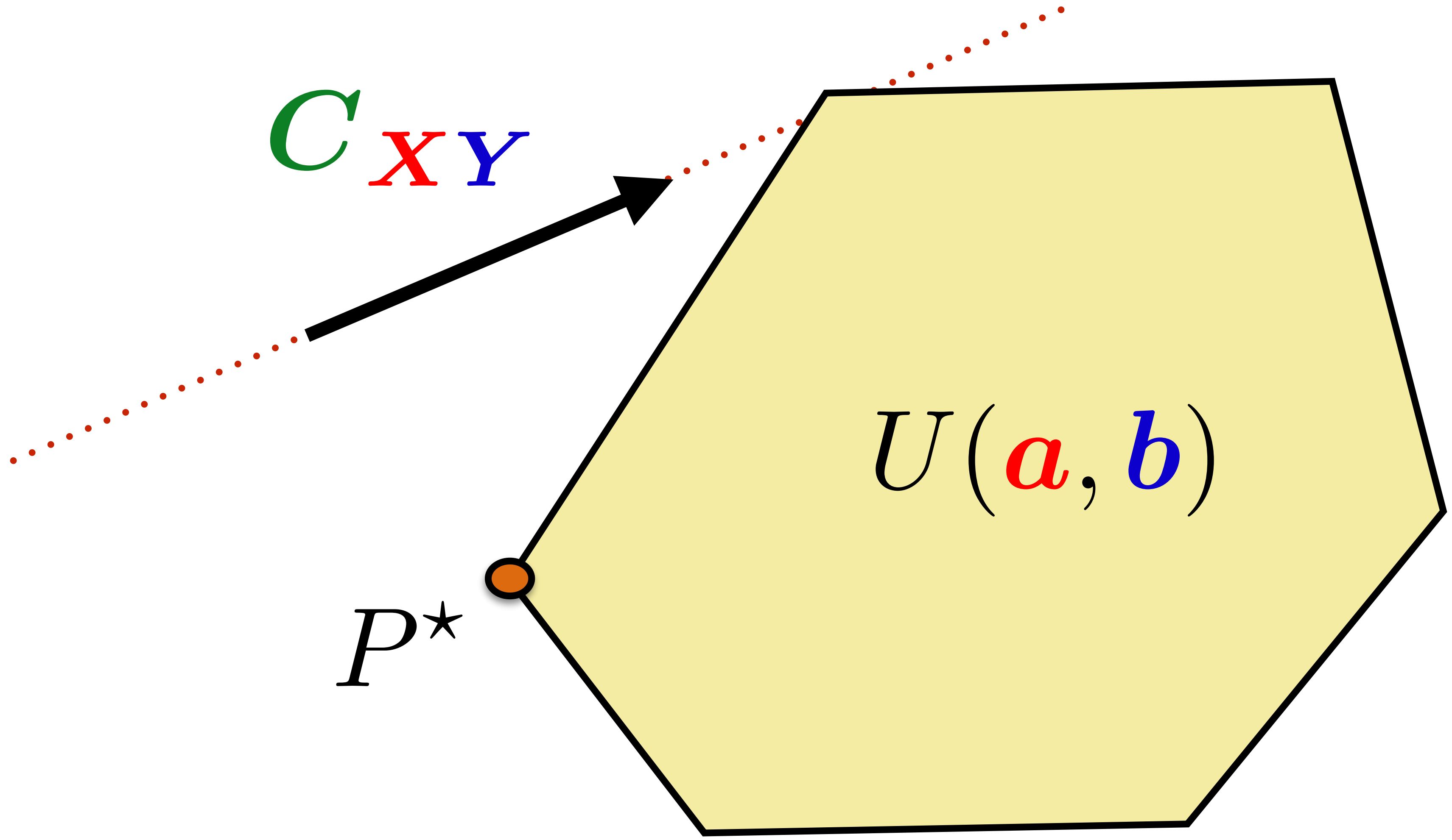
$$\boldsymbol{C}_{\boldsymbol{XY}} \stackrel{\text{def}}{=} [\boldsymbol{c}(\boldsymbol{x}_i, \boldsymbol{y}_j)]_{ij}$$

$$U(\boldsymbol{a}, \boldsymbol{b}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathbb{R}_{+}^{n \times m} | \boldsymbol{P}\mathbf{1}_m = \boldsymbol{a}, \boldsymbol{P}^T\mathbf{1}_n = \boldsymbol{b}\}$$

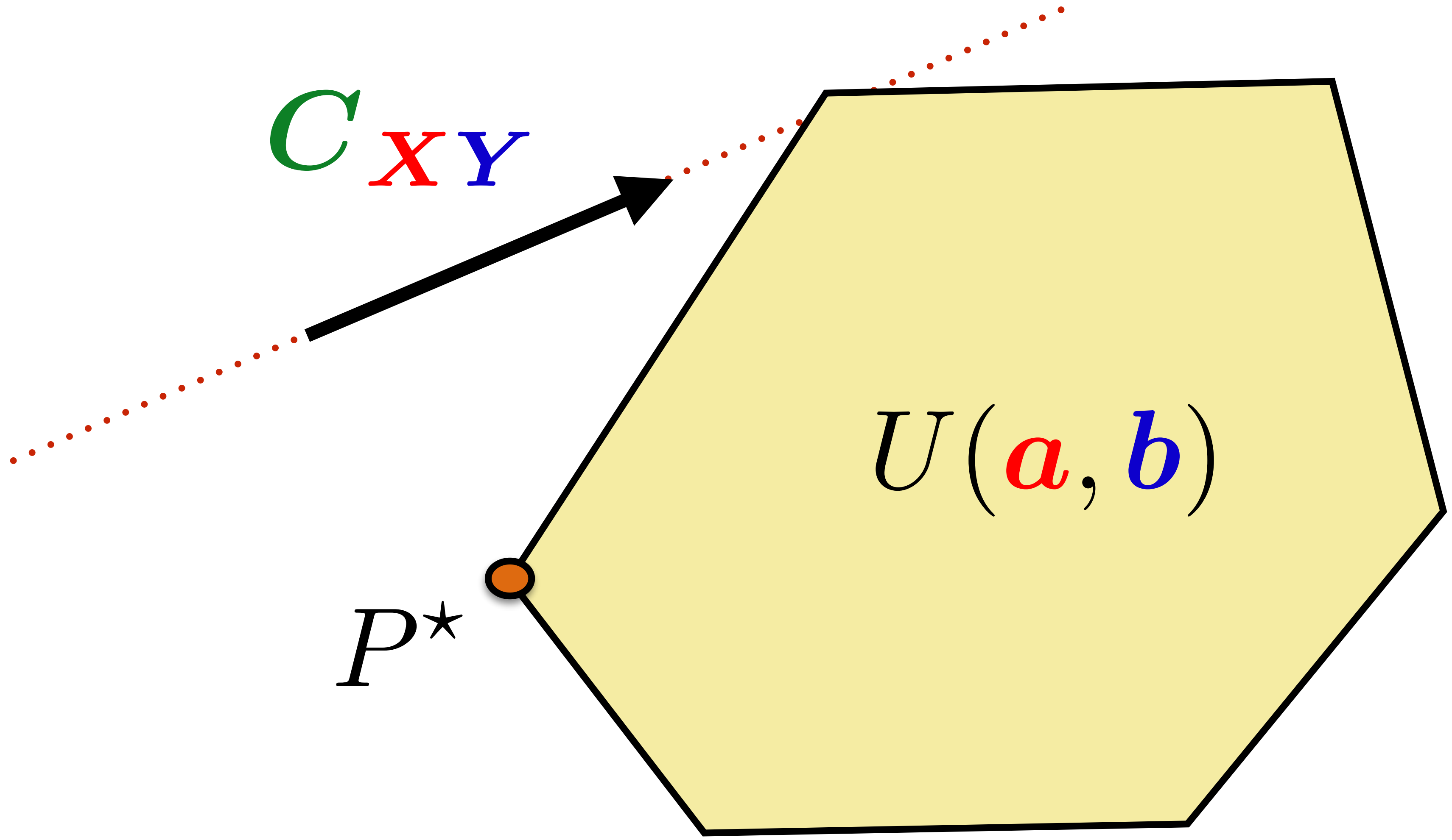
Def. Optimal Transport Problem

$$\min_{\boldsymbol{P} \in U(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{P}, \boldsymbol{C}_{\boldsymbol{XY}} \rangle$$

Computations



Computations



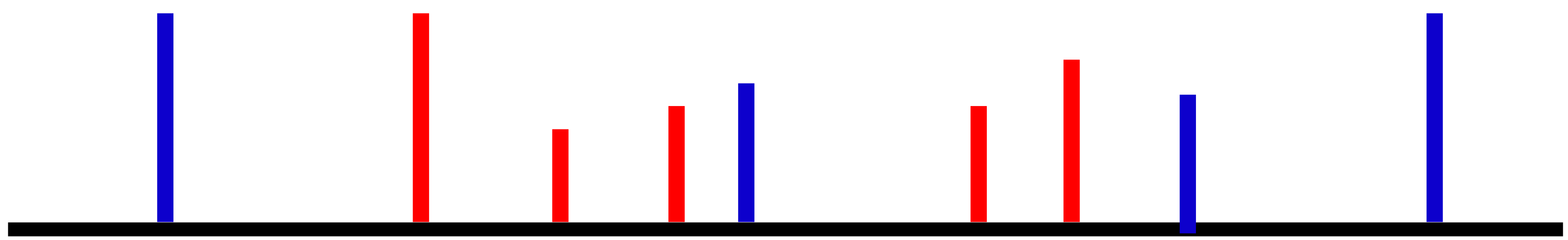
min cost flow solver used in practice.

$$O((n+m)nm\log(n+m))$$

Optimal Transport in 1D

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

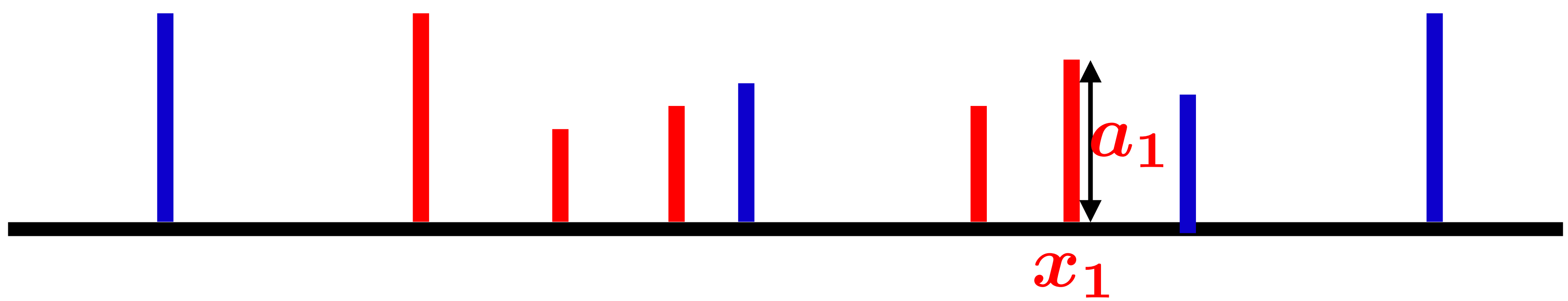


$\Omega = \mathbb{R}$

Optimal Transport in 1D

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

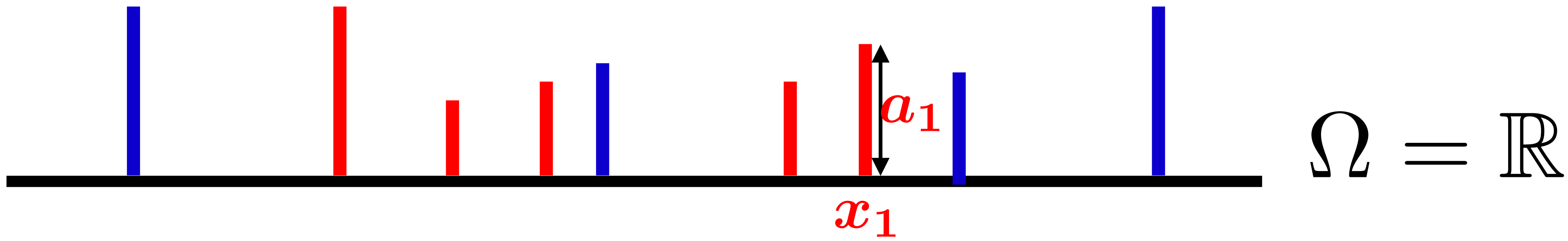
$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$



$$\Omega = \mathbb{R}$$

Optimal Transport in 1D

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \qquad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$



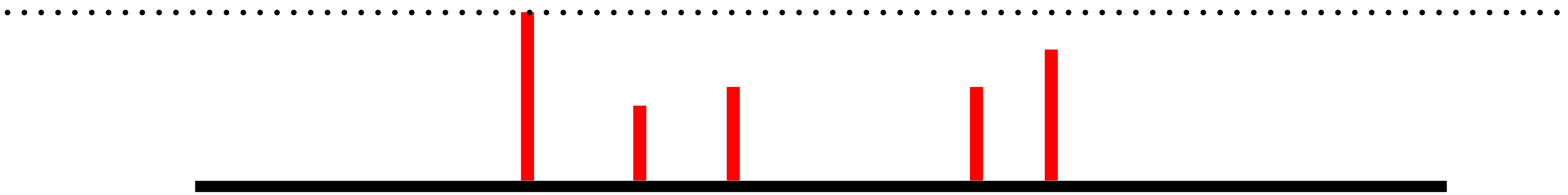
Assume...

$\boldsymbol{c}(x, y) := \boldsymbol{h}(y - x)$, $\boldsymbol{h}$ is **convex** and ≥ 0.

...then OT boils down to sorting

Empirical Distribution Functions

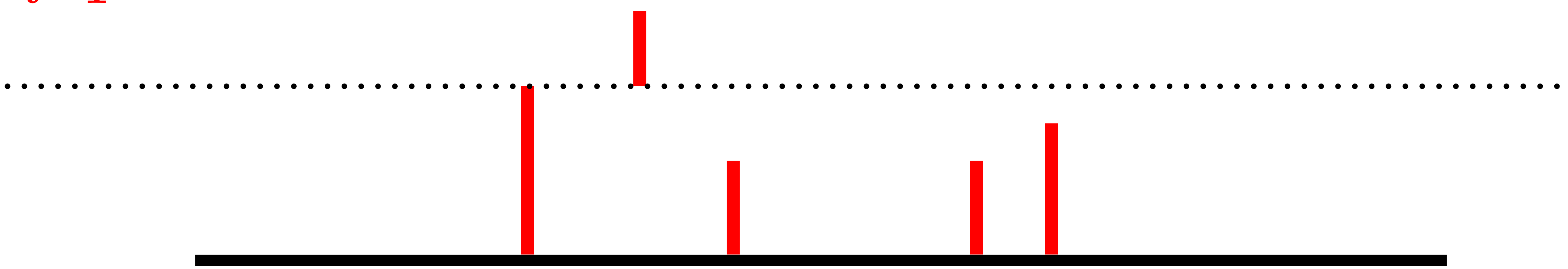
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$



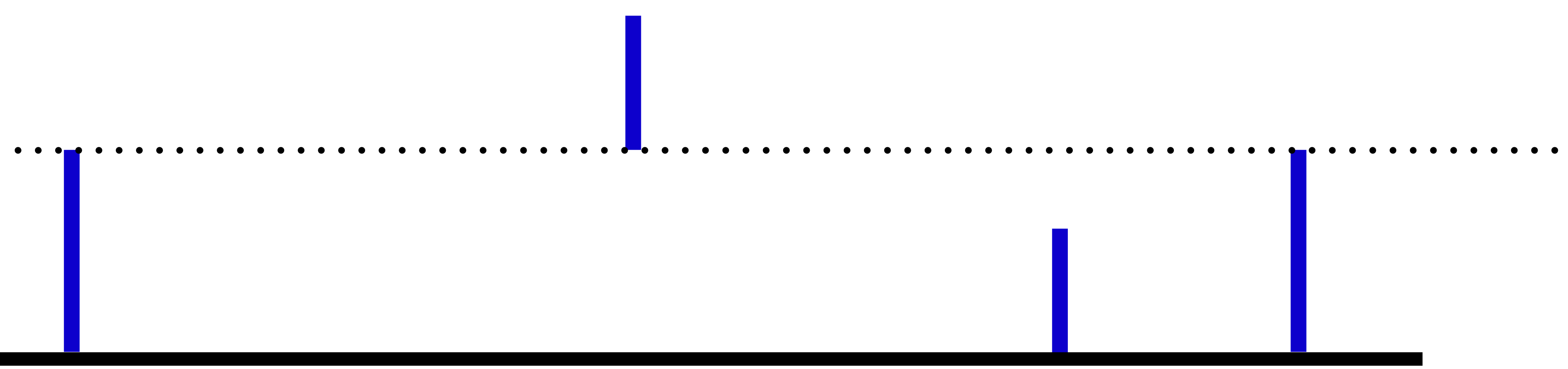
$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Empirical Distribution Functions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

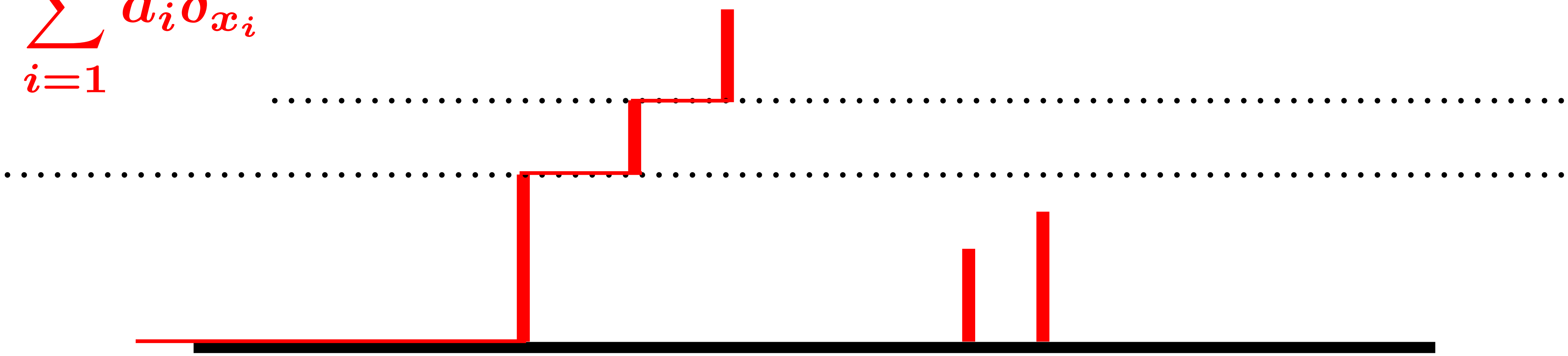


$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

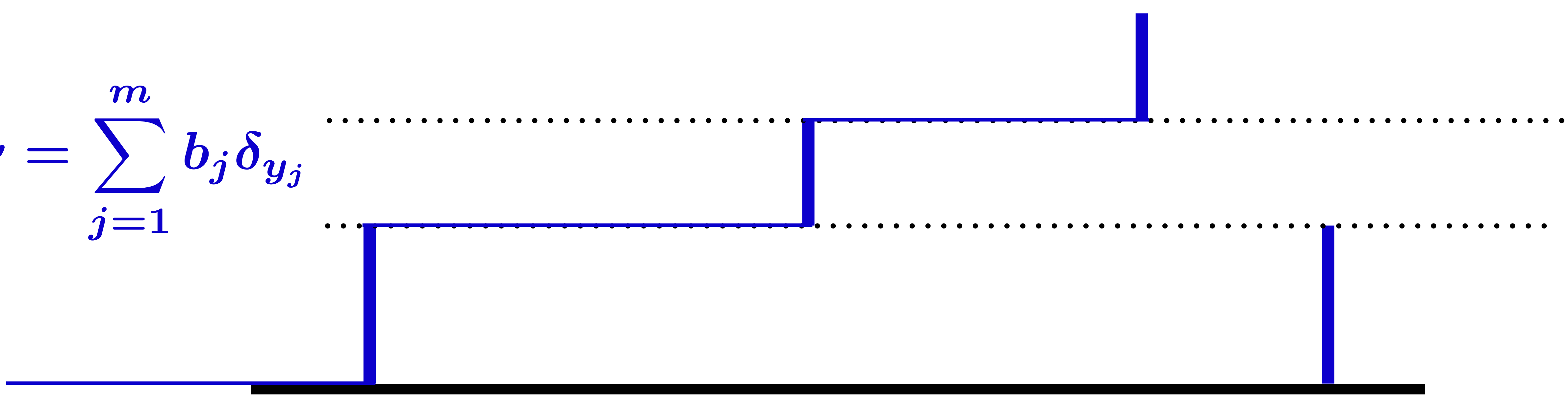


Empirical Distribution Functions

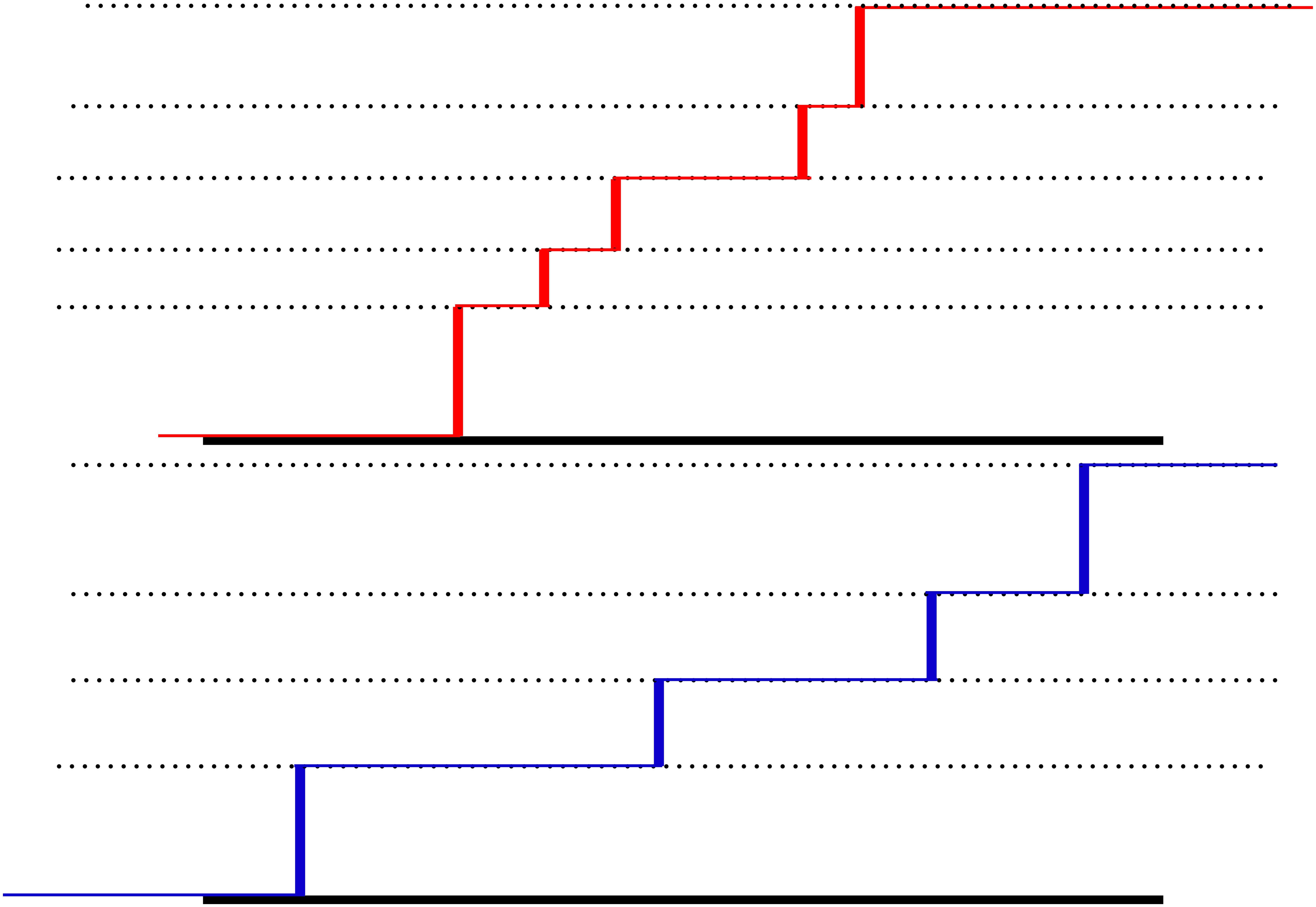
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$



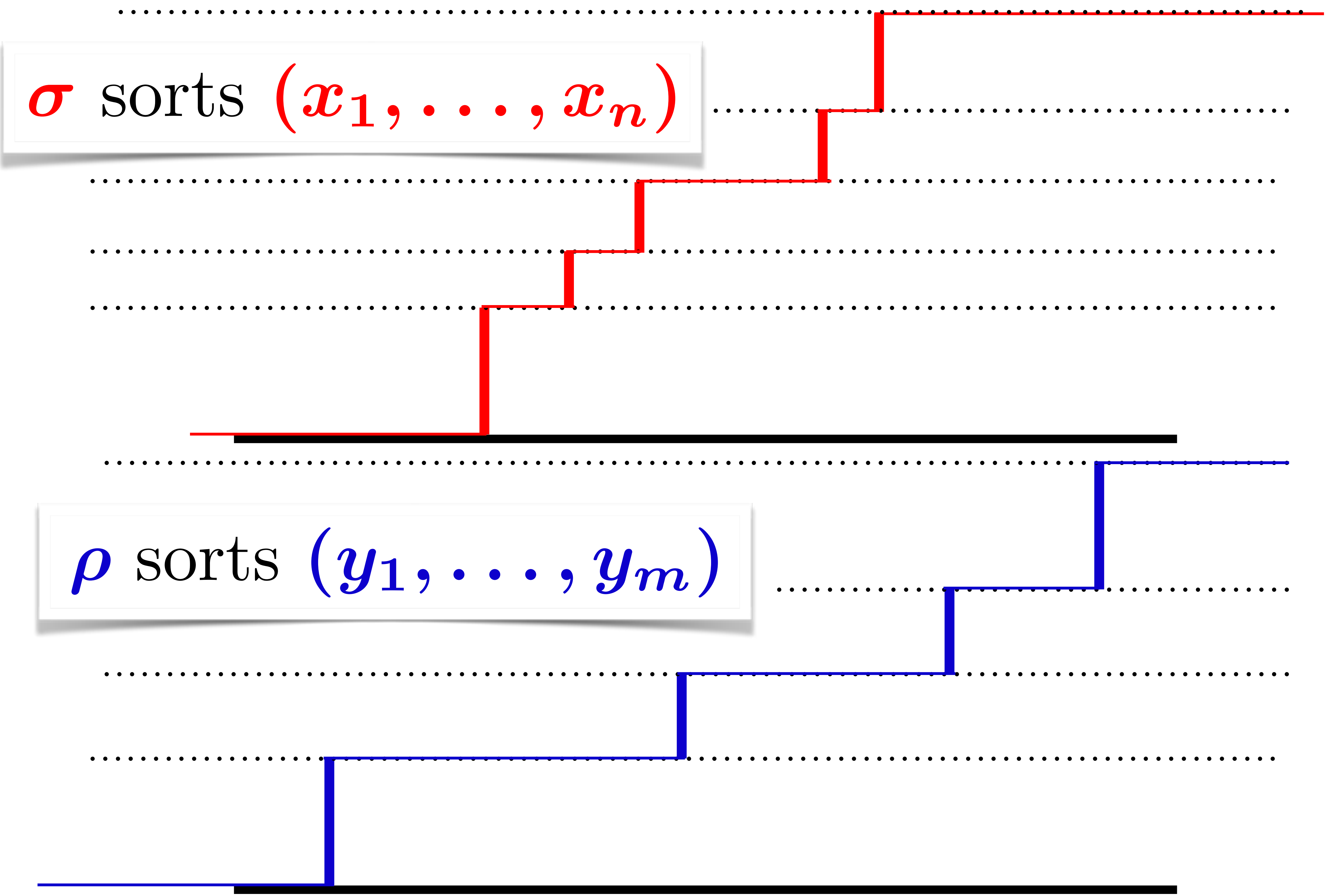
$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$



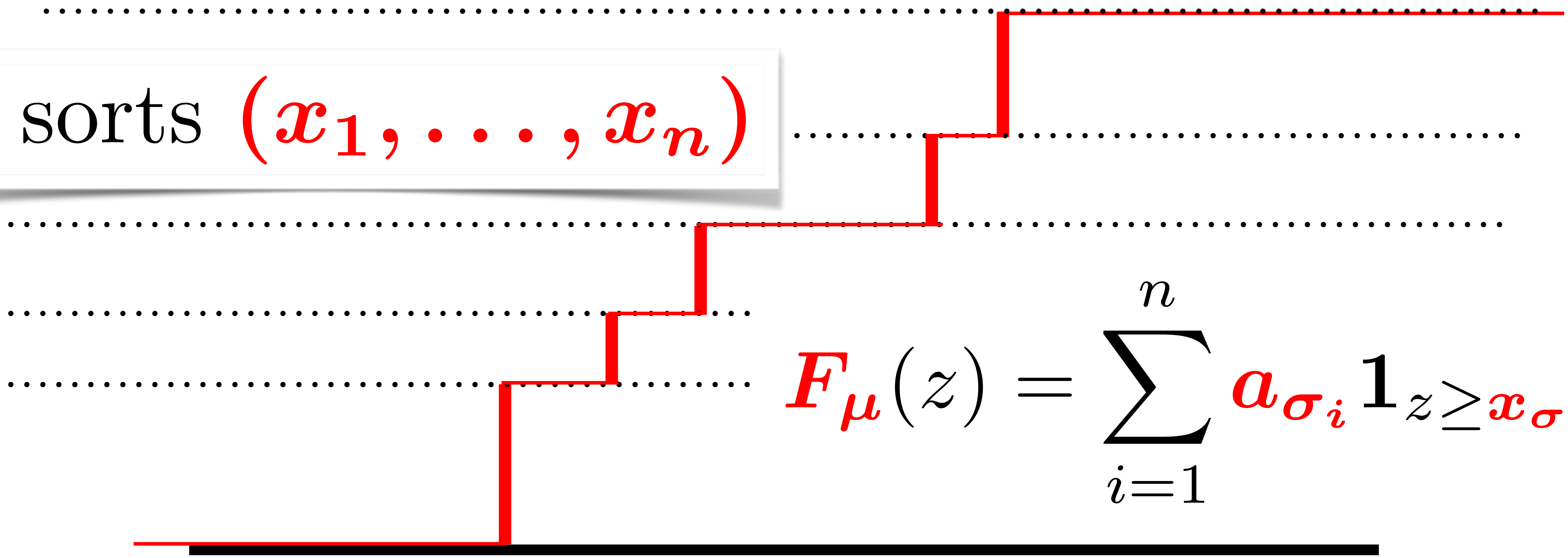
Empirical Distribution Functions



Empirical Distribution Functions

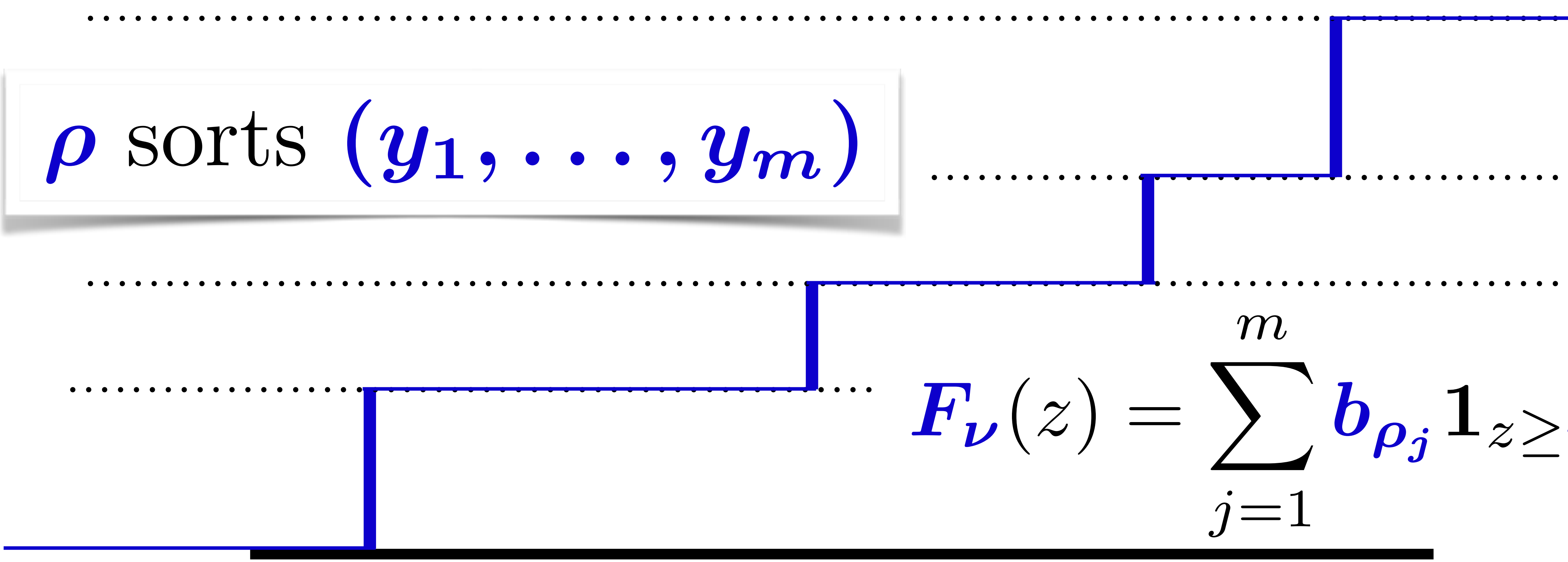


Empirical Distribution Functions



$\boldsymbol{\sigma}$ sorts $(\boldsymbol{x_1, \ldots, x_n})$

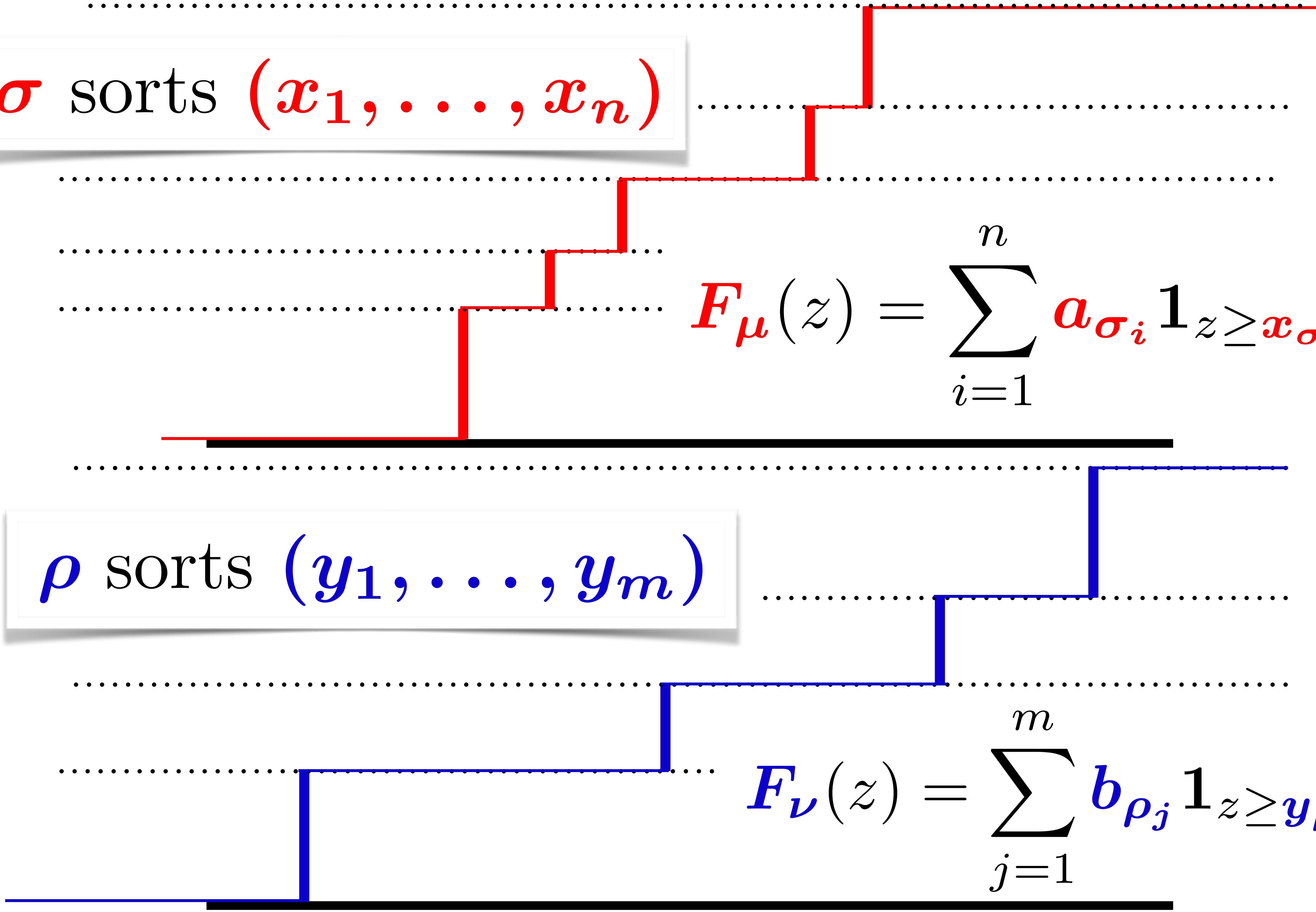
$$F_\mu(z) = \sum_{i=1}^{n} \boldsymbol{a_{\sigma_i}} \mathbf{1}_{z \geq \boldsymbol{x_{\sigma_i}}}$$



$\boldsymbol{\rho}$ sorts $(\boldsymbol{y_1, \ldots, y_m})$

$$F_\nu(z) = \sum_{j=1}^{m} \boldsymbol{b_{\rho_j}} \mathbf{1}_{z \geq \boldsymbol{y_{\rho_j}}}$$

Empirical Distributions to Quantiles



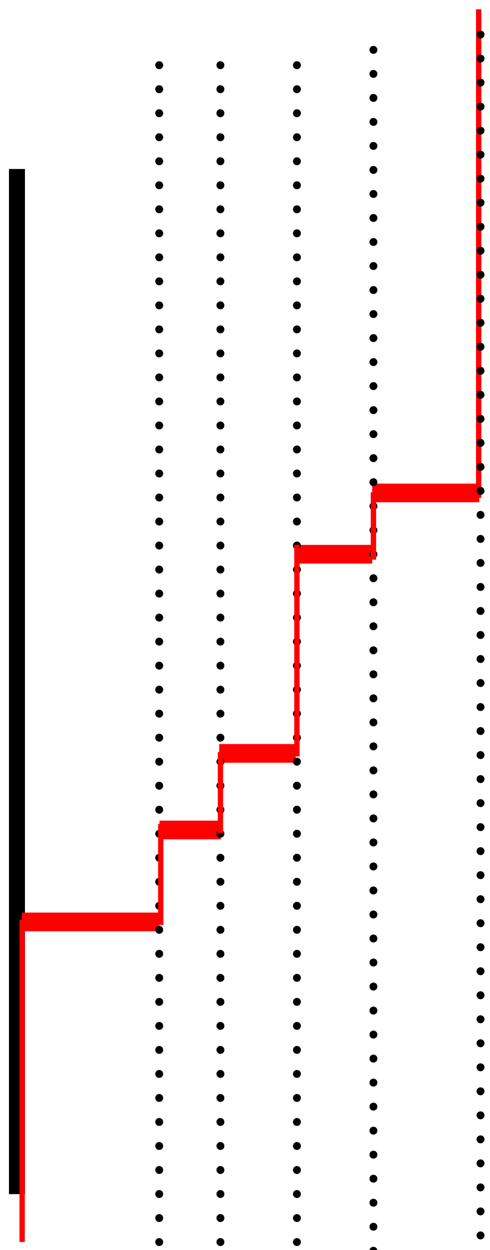
$\boldsymbol{\sigma}$ sorts $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$

$$\boldsymbol{F}_{\boldsymbol{\mu}}(z) = \sum_{i=1}^{n} \boldsymbol{a}_{\boldsymbol{\sigma}_i} \mathbf{1}_{z \geq \boldsymbol{x}_{\boldsymbol{\sigma}_i}}$$

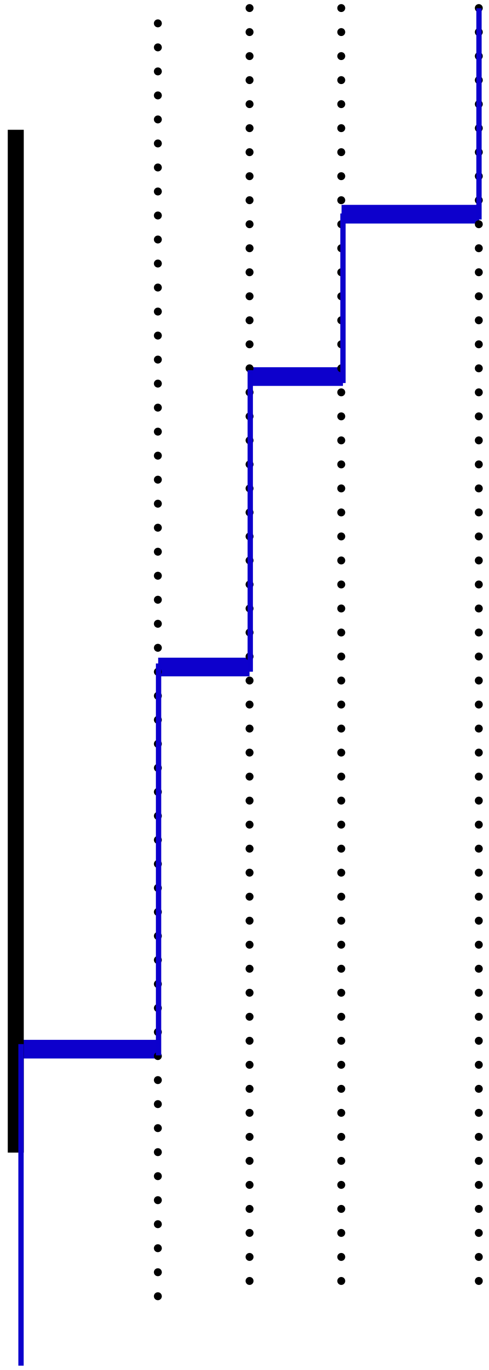
$\boldsymbol{\rho}$ sorts $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$

$$\boldsymbol{F}_{\boldsymbol{\nu}}(z) = \sum_{j=1}^{m} \boldsymbol{b}_{\boldsymbol{\rho}_j} \mathbf{1}_{z \geq \boldsymbol{y}_{\boldsymbol{\rho}_j}}$$

Empirical Distributions to Quantiles

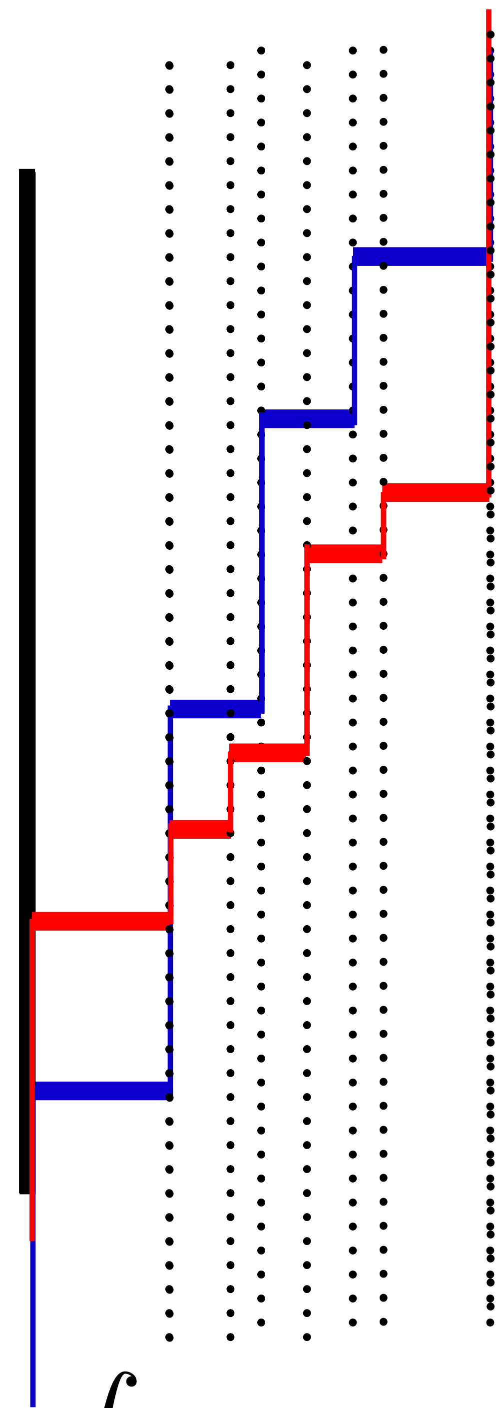


F_μ^{-1}



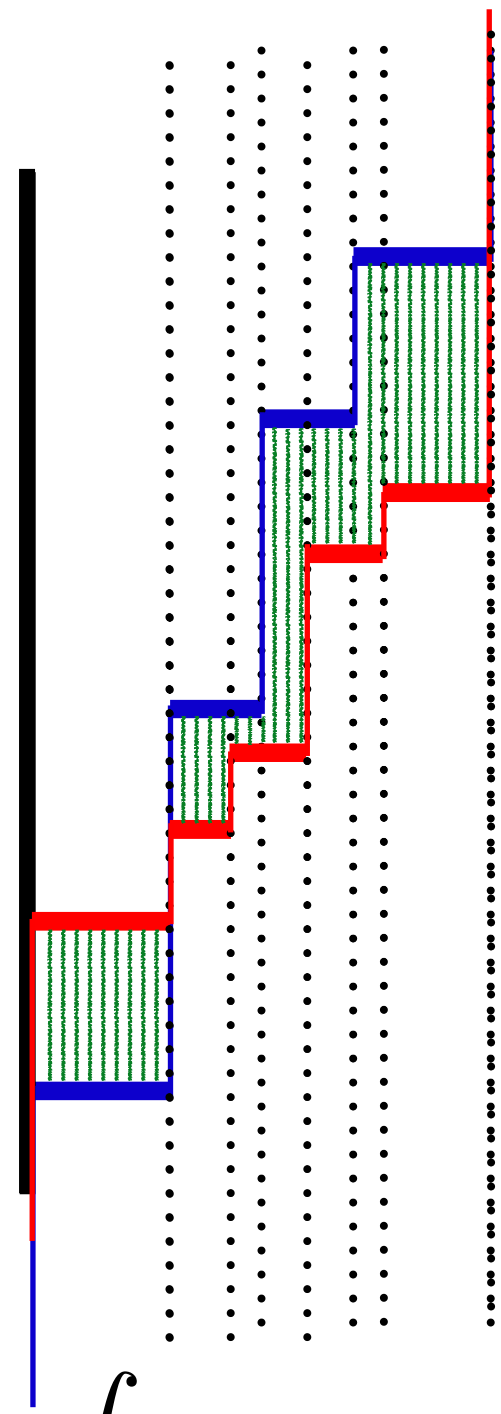
F_ν^{-1}

OT = Comparing Quantiles



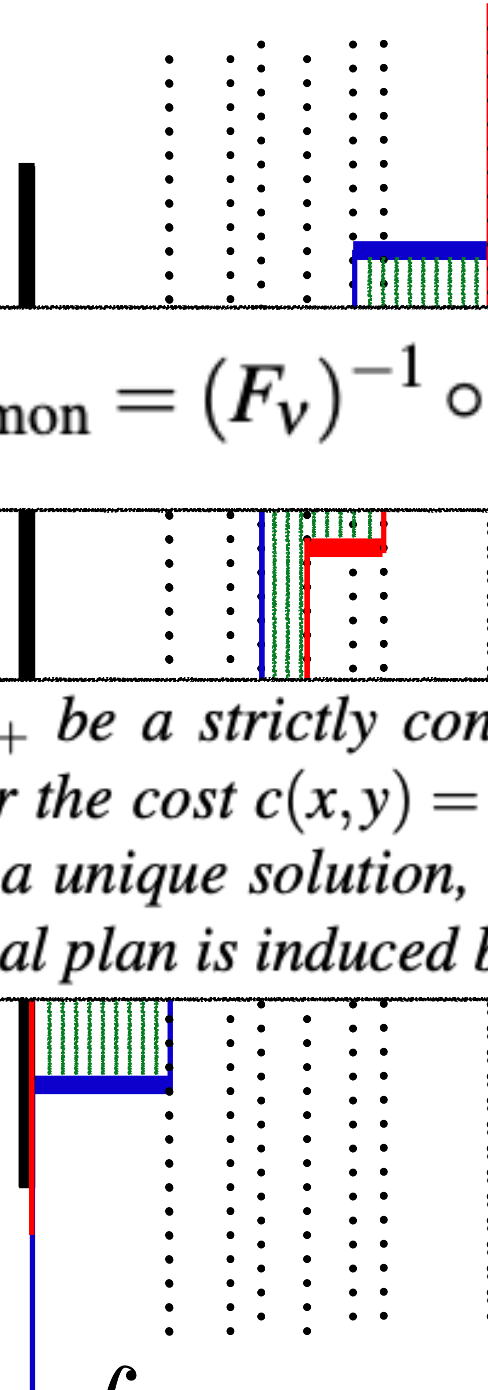
$$\min_{\boldsymbol{P}\in U(\boldsymbol{a},\boldsymbol{b})}\langle \boldsymbol{P}, \boldsymbol{C}_{\boldsymbol{X}\boldsymbol{Y}}\rangle = \int_{u\in[0,1]} \boldsymbol{h}(\boldsymbol{F}_{\boldsymbol{\nu}}^{-1}(u) - \boldsymbol{F}_{\boldsymbol{\mu}}^{-1}(u))du$$

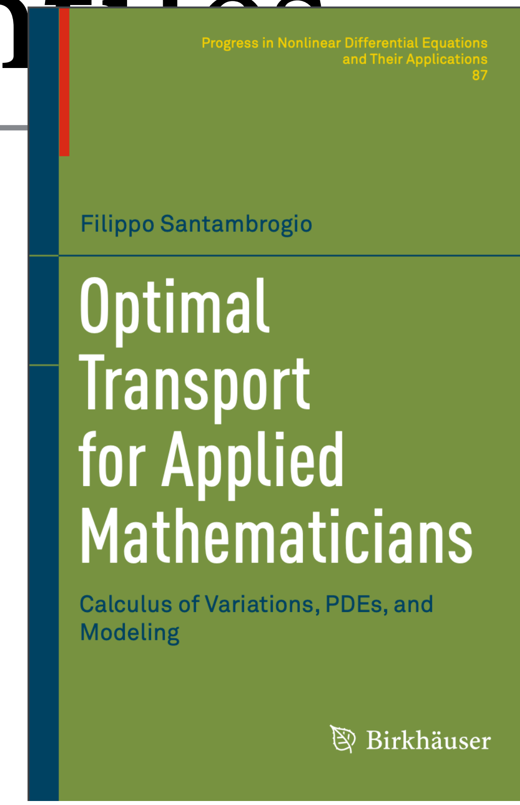
OT = Comparing Quantiles



$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \int_{u\in[0,1]} h(F_\nu^{-1}(u) - F_\mu^{-1}(u))du$$

OT = Comparing Quantiles





$$\mathrm{T}_{\mathrm{mon}} = (F_\nu)^{-1} \circ F_\mu .$$

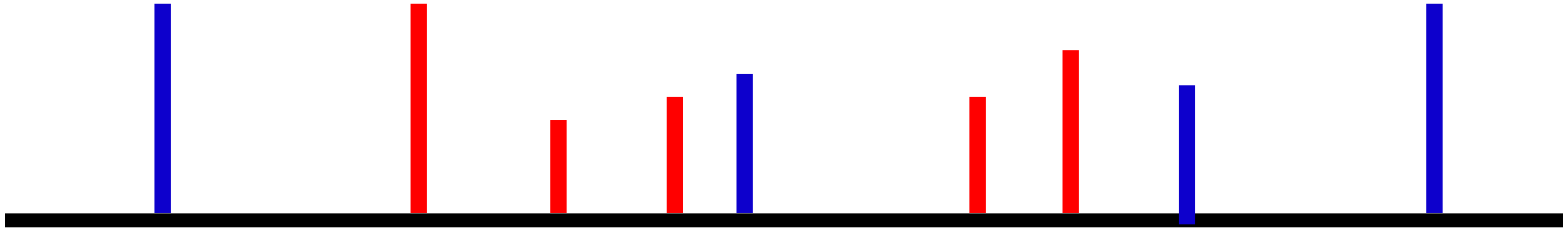
Theorem 2.9. *Let $h : \mathbb{R} \to \mathbb{R}_+$ be a strictly convex function, and $\mu, \nu \in \mathscr{P}(\mathbb{R})$ be probability measures. Consider the cost $c(x,y) = h(y-x)$ and suppose that* (KP) *has a finite value. Then,* (KP) *has a unique solution, which is given by γ_{mon}. In the case where μ is atomless, this optimal plan is induced by the map $\mathrm{T}_{\mathrm{mon}}$.*

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \int_{u \in [0,1]} h(F_\nu^{-1}(u) - F_\mu^{-1}(u))du$$

More Explicit Link to Sorting

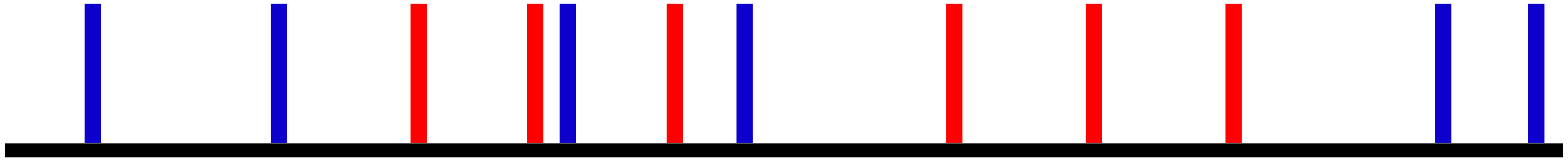
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$$

$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$



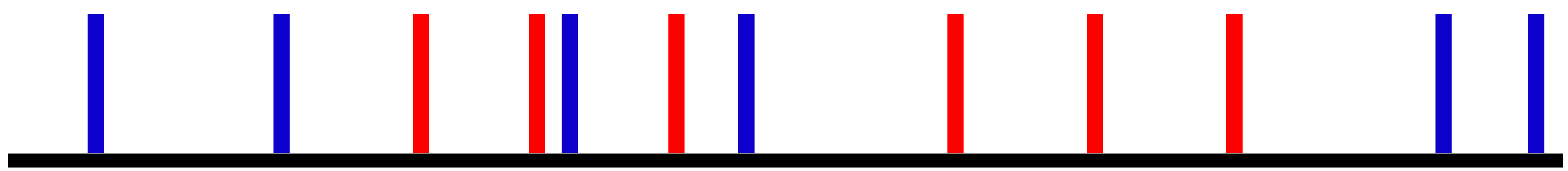
Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \qquad \nu = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}$$



Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \qquad \nu = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}$$

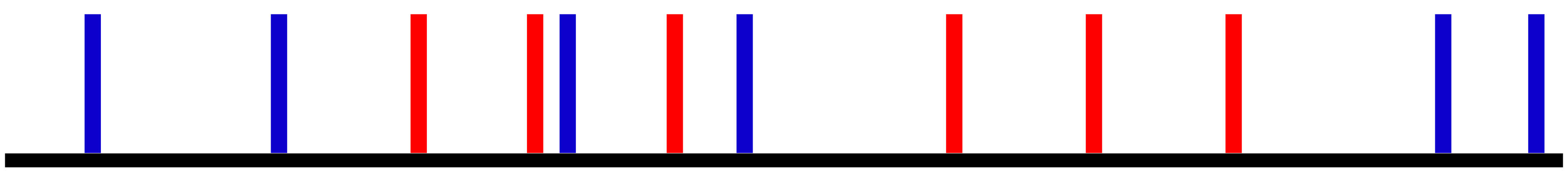


σ sorts $(x_1, \ldots, x_n)$

ρ sorts $(y_1, \ldots, y_m)$

Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$



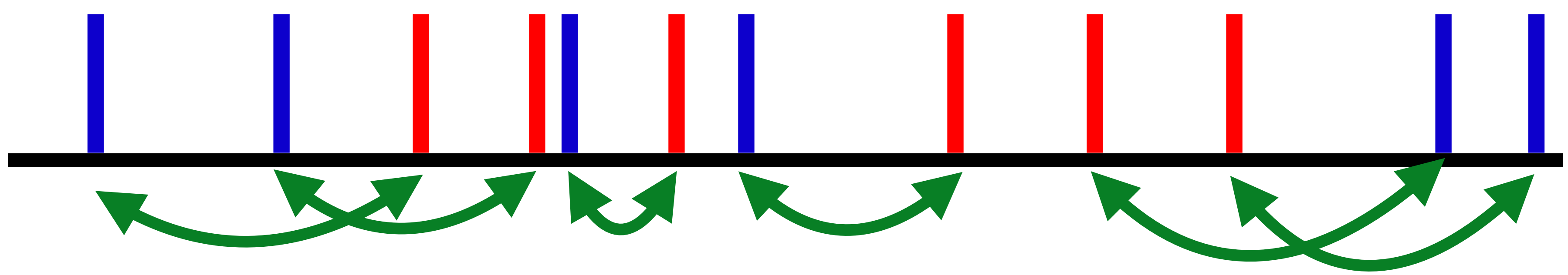
σ sorts $(x_1, \dots, x_n)$

ρ sorts $(y_1, \dots, y_m)$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \sum_{i=1}^{n} h(y_{\rho_i} - x_{\sigma_i})$$

Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$



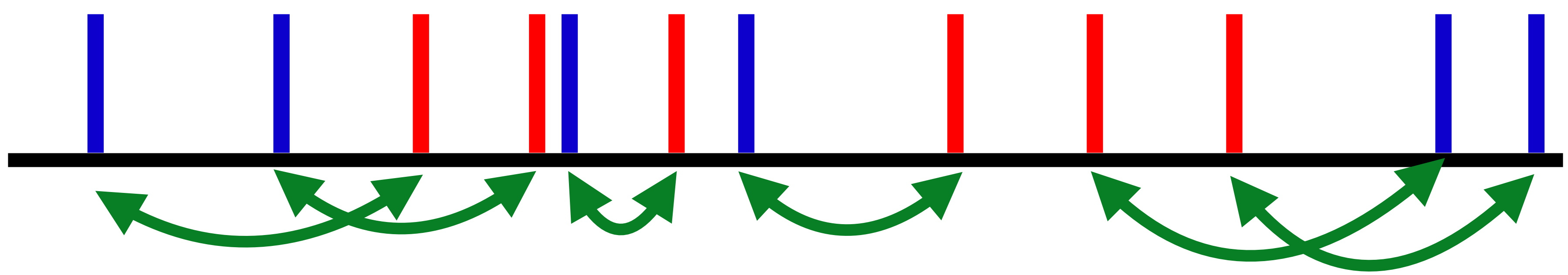
σ sorts $(x_1, \dots, x_n)$

ρ sorts $(y_1, \dots, y_m)$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \sum_{i=1}^{n} h(y_{\rho_i} - x_{\sigma_i})$$

Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$



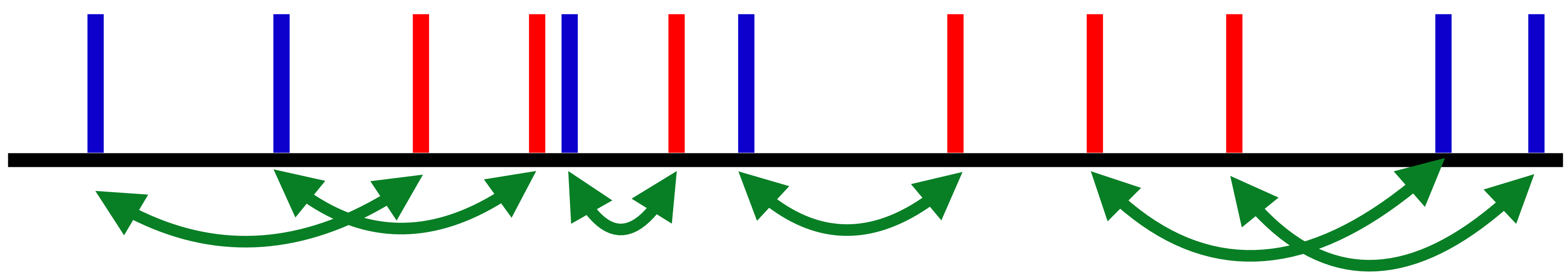
σ sorts $(x_1, \dots, x_n)$

ρ sorts $(y_1, \dots, y_m)$

$$P^\star = \frac{1}{n}\mathrm{sp}\mathbf{1}\left((\sigma_i, \rho_i)_i\right)$$

Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$



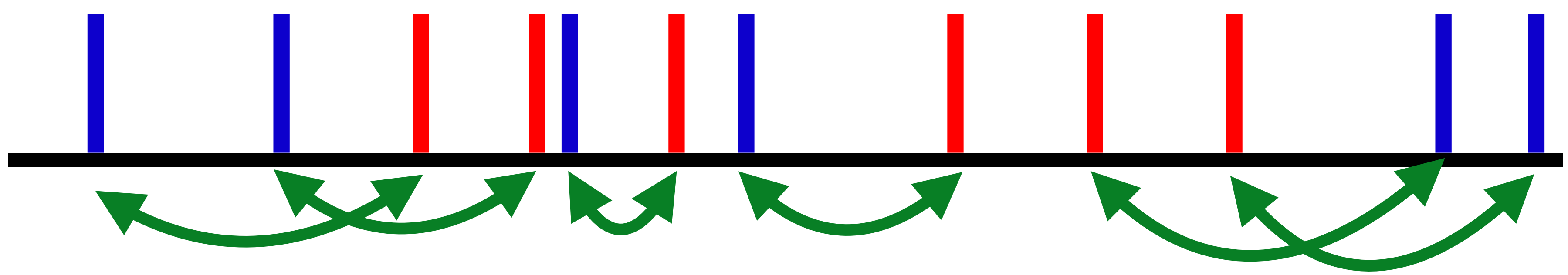
σ sorts $(x_1, \ldots, x_n)$

ρ sorts $(y_1, \ldots, y_m)$

$$P^\star = \frac{1}{n}\mathrm{sp}\mathbf{1}\left(\left(\sigma_i, \rho_i\right)_i\right)$$

Link to Sorting: Uniform Measures

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \qquad \nu = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}$$



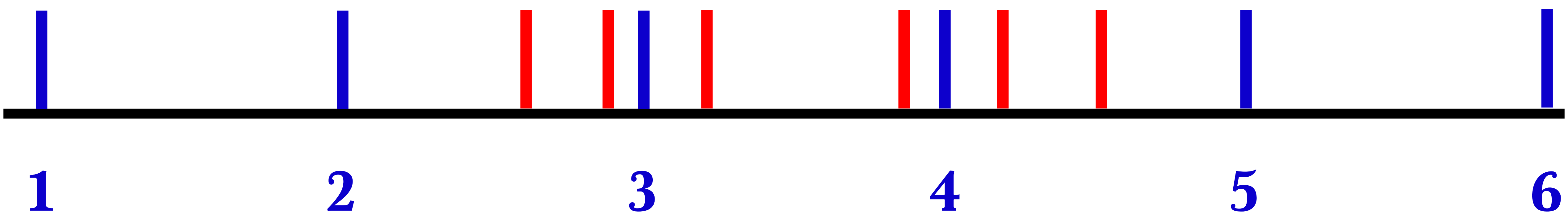
σ sorts $(x_1, \ldots, x_n)$

ρ sorts $(y_1, \ldots, y_m)$

$$P^\star = \frac{1}{n}\mathrm{sp}\mathbf{1}\left(\left(\sigma_i, \rho_i\right)_i\right)$$

OT towards a *sorted* sequence

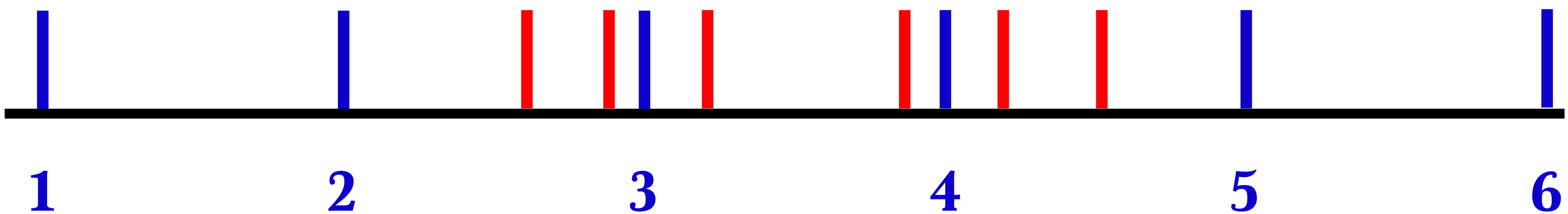
$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_i$$



σ sorts $(x_1, \ldots, x_n)$

OT towards a *sorted* sequence

$$\mu = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n} \delta_i$$

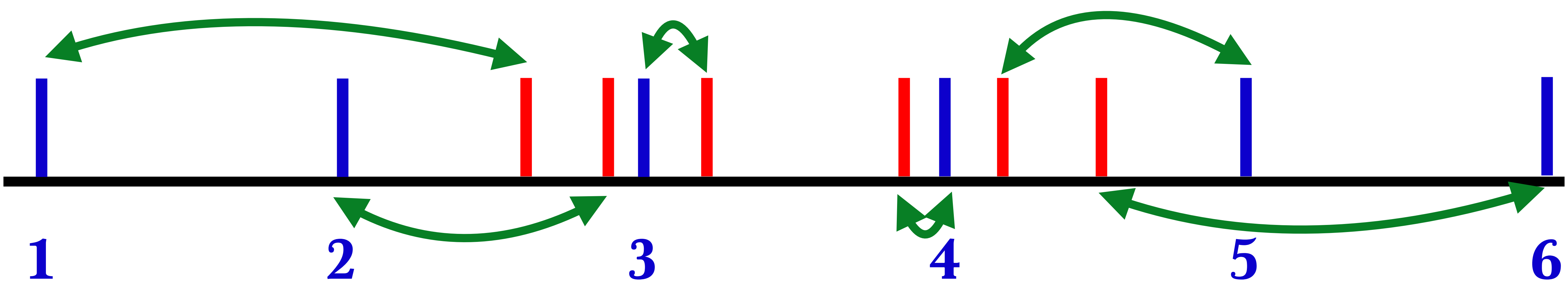


σ sorts $(x_1, \dots, x_n)$ $\qquad \rho = \mathrm{Id}$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \sum_{i=1}^{n} h(i - x_{\sigma_i})$$

OT towards a *sorted* sequence

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{i}$$



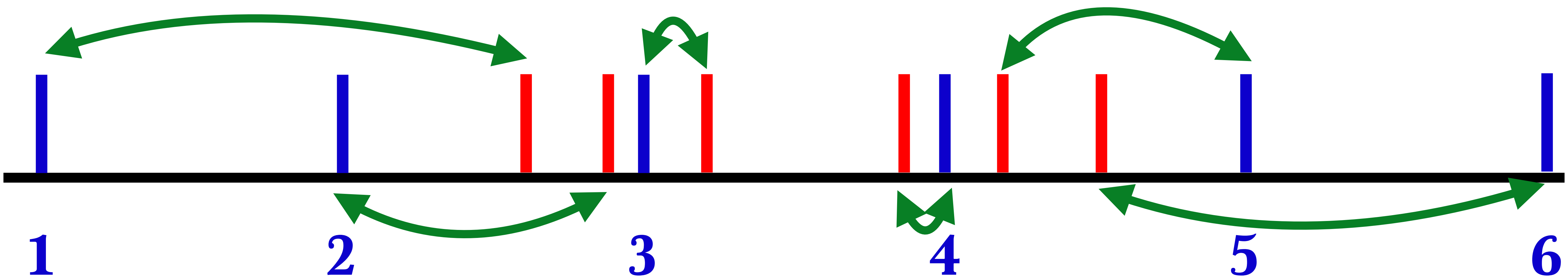
σ sorts $(x_1, \dots, x_n)$

$\rho = \mathrm{Id}$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle = \sum_{i=1}^{n} h(i - x_{\sigma_i})$$

OT towards a *sorted* sequence

$$\mu = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n} \delta_i$$



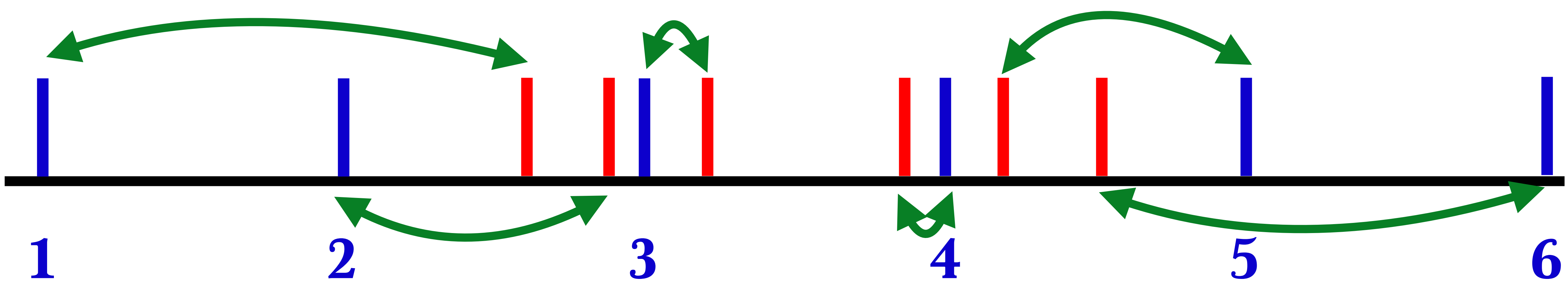
σ sorts $(x_1, \ldots, x_n)$

$\rho = \mathrm{Id}$

$$P^\star = \frac{1}{n}\mathrm{sp}\mathbf{1}\left(\left(\sigma_i, i\right)_i\right)$$

OT towards a *sorted* sequence

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \qquad \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{i}$$

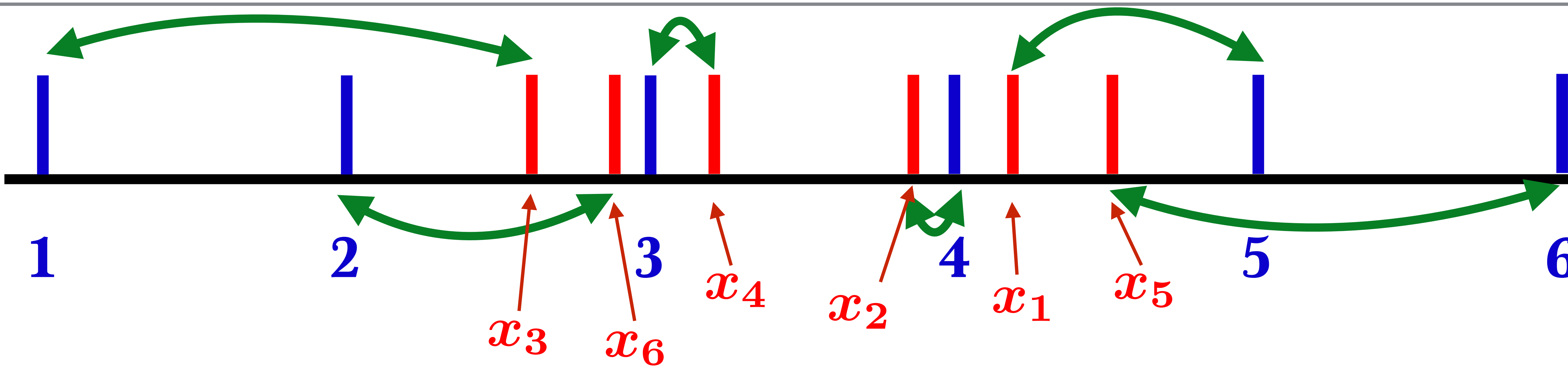


σ sorts $(x_1, \ldots, x_n)$

$\rho = \mathrm{Id}$

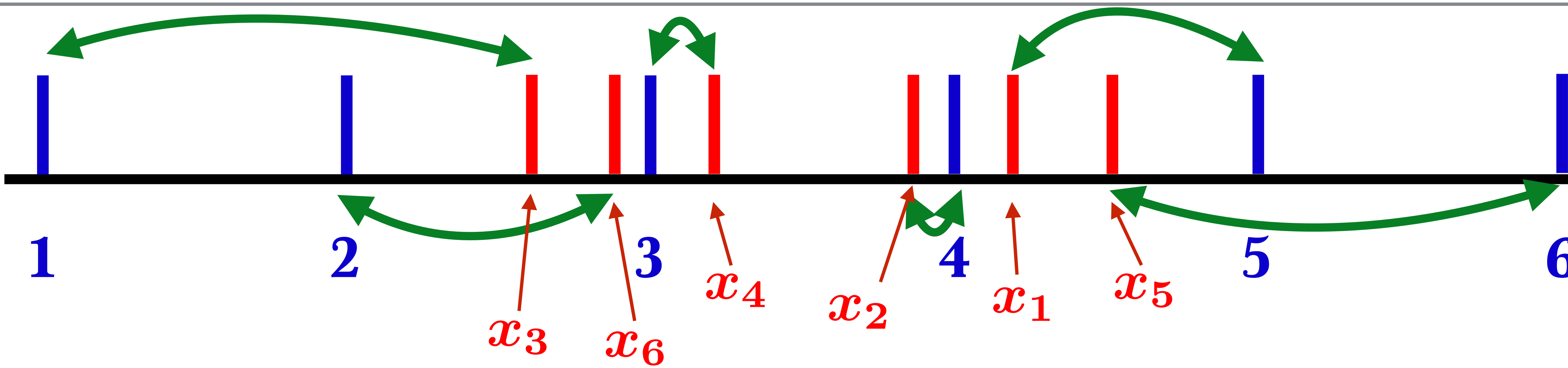
$$P^\star = \frac{1}{n}\mathrm{sp}\mathbf{1}\left(\left(\sigma_i, i\right)_i\right) = \frac{1}{n}\Pi_{\sigma}^{T}$$

OT towards a *sorted* sequence



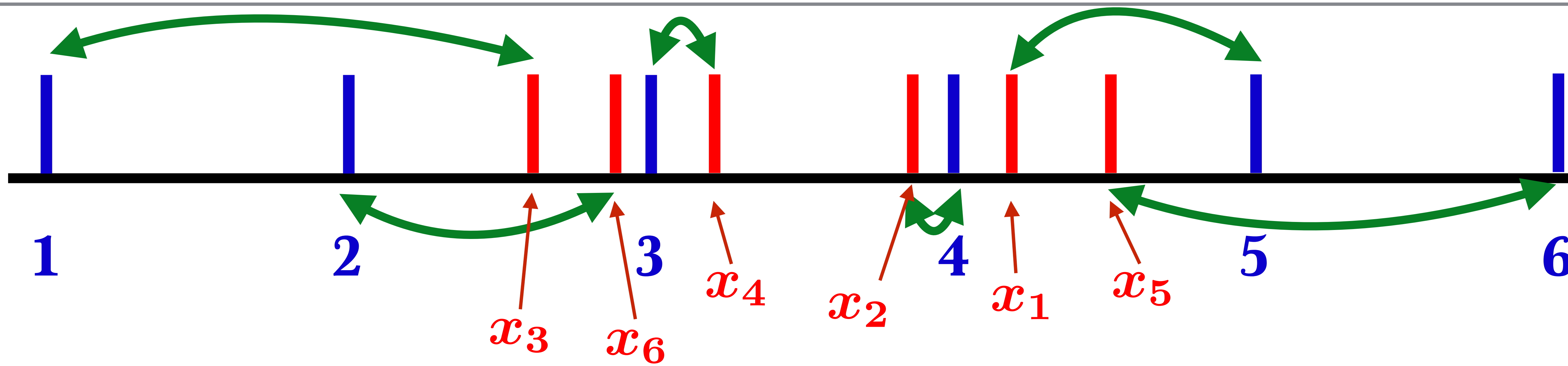
$$P^\star = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} & & & & \frac{1}{n} & \\ & & & \frac{1}{n} & & \\ \frac{1}{n} & & & & & \\ & & \frac{1}{n} & & & \\ & & & & & \frac{1}{n} \\ & \frac{1}{n} & & & & \end{bmatrix} \end{array} = \frac{1}{n}\Pi^T_{\sigma(\mathbf{x})} = \frac{1}{n}\Pi_{\sigma(\mathbf{x})^{-1}}$$

OT towards a *sorted* sequence = sorting



$$P^\star = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} & & & & \frac{1}{n} & \\ & & & \frac{1}{n} & & \\ \frac{1}{n} & & & & & \\ & & \frac{1}{n} & & & \\ & & & & & \frac{1}{n} \\ & \frac{1}{n} & & & & \end{bmatrix} \end{array}$$

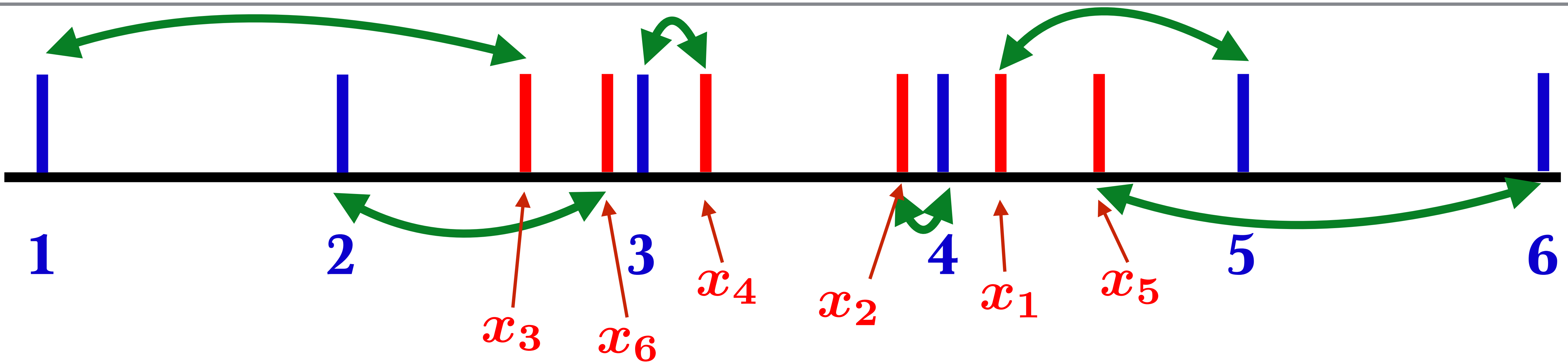
OT towards a *sorted* sequence = sorting



$$P^\star = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} & & & & \frac{1}{n} & \\ & & & \frac{1}{n} & & \\ \frac{1}{n} & & & & & \\ & & \frac{1}{n} & & & \\ & & & & & \frac{1}{n} \\ & \frac{1}{n} & & & & \end{bmatrix} \end{array}$$

$$R(\mathbf{x}) = \sigma(\mathbf{x})^{-1} = n^2 P^\star \begin{bmatrix} 1/6 \\ 2/6 \\ 3/6 \\ 4/6 \\ 5/6 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \\ 3 \\ 6 \\ 2 \end{bmatrix}$$

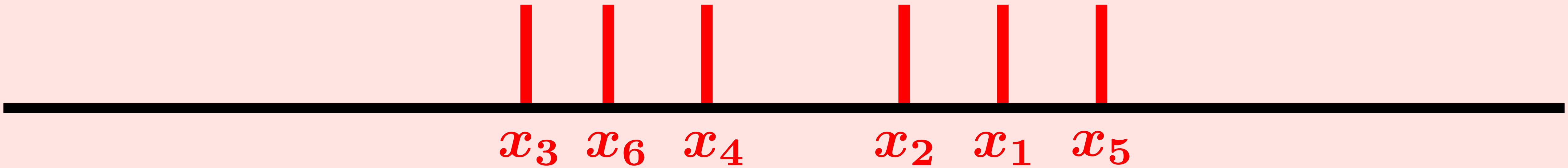
OT towards a *sorted* sequence = sorting

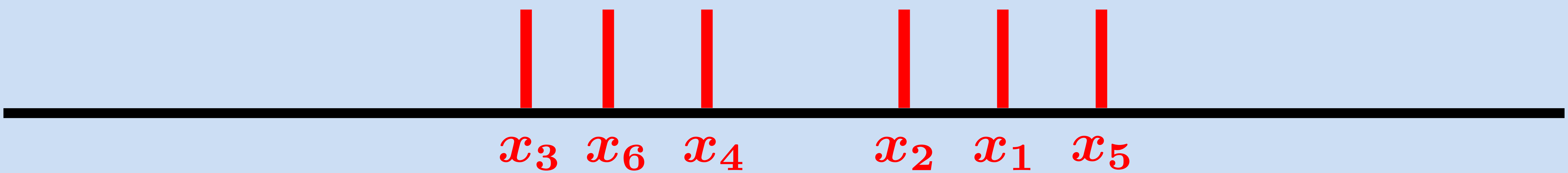


$$P^\star = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} & & & & \frac{1}{n} & \\ & & & \frac{1}{n} & & \\ \frac{1}{n} & & & & & \\ & & \frac{1}{n} & & & \\ & & & & & \frac{1}{n} \\ & \frac{1}{n} & & & & \end{bmatrix} \end{array}$$

$$R(\mathbf{x}) = \sigma(\mathbf{x})^{-1} = n^2 P^\star \begin{bmatrix} 1/6 \\ 2/6 \\ 3/6 \\ 4/6 \\ 5/6 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \\ 3 \\ 6 \\ 2 \end{bmatrix} \qquad S(\mathbf{x}) = n(P^\star)^T \mathbf{x} = \begin{bmatrix} x_3 \\ x_6 \\ x_4 \\ x_2 \\ x_1 \\ x_5 \end{bmatrix}$$

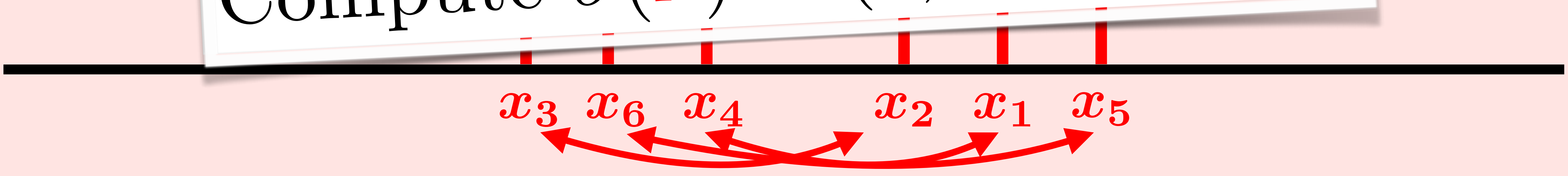
Two ways to rank and sort

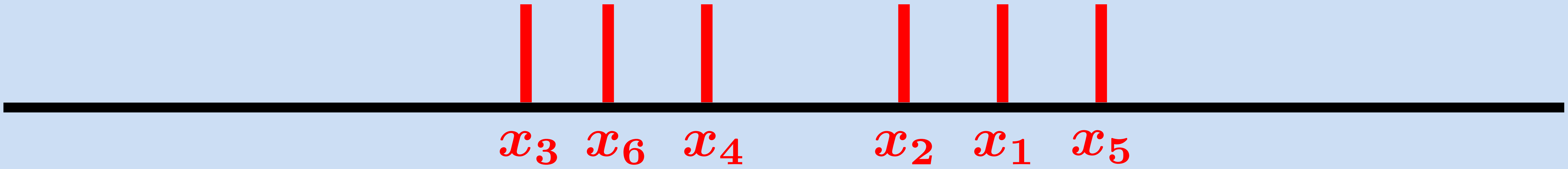




Two ways to rank and sort

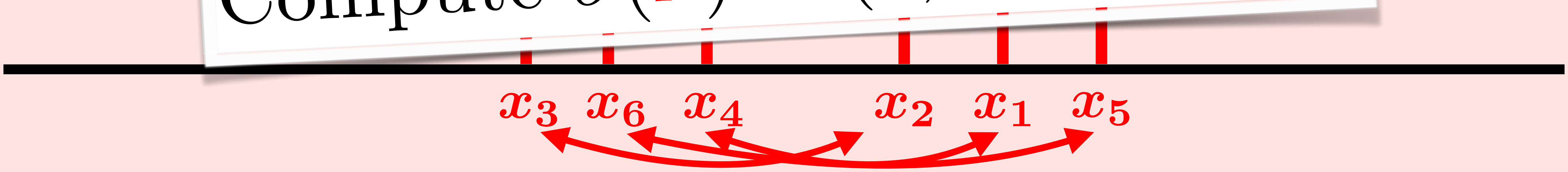
Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



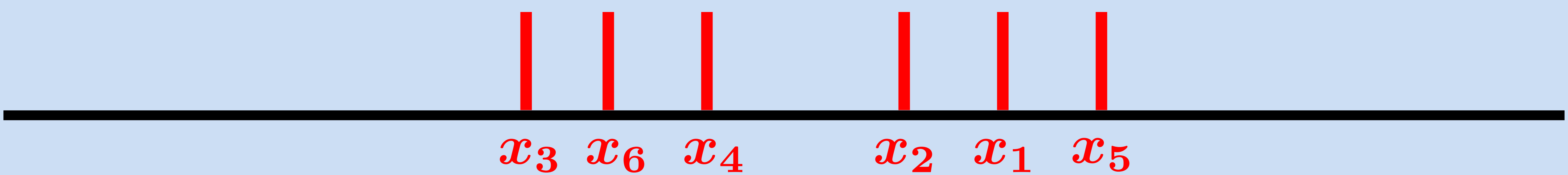


Two ways to rank and sort

Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,

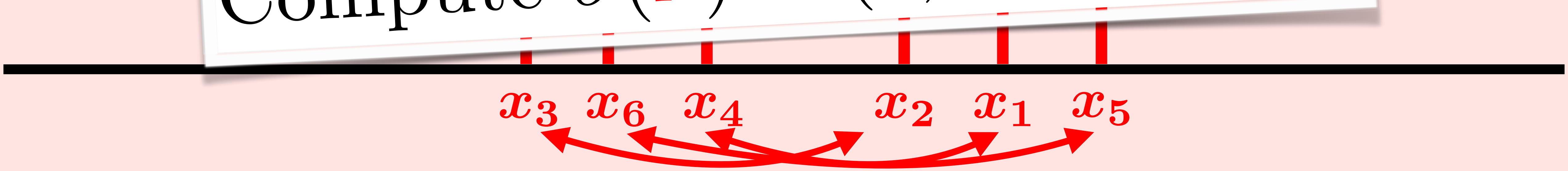


$$R(\mathbf{x}) = (\mathbf{5}, \mathbf{4}, \mathbf{1}, \mathbf{3}, \mathbf{6}, \mathbf{2}),\ S(\mathbf{x}) = (\boldsymbol{x_3}, \boldsymbol{x_6}, \boldsymbol{x_4}, \boldsymbol{x_2}, \boldsymbol{x_1}, \boldsymbol{x_5})$$



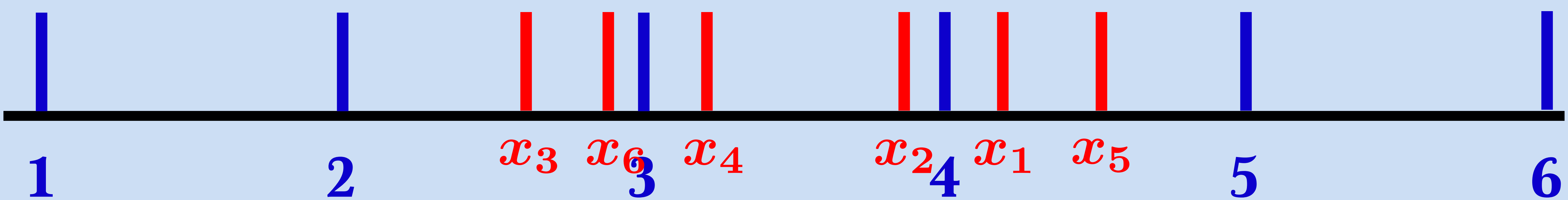
Two ways to rank and sort

Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



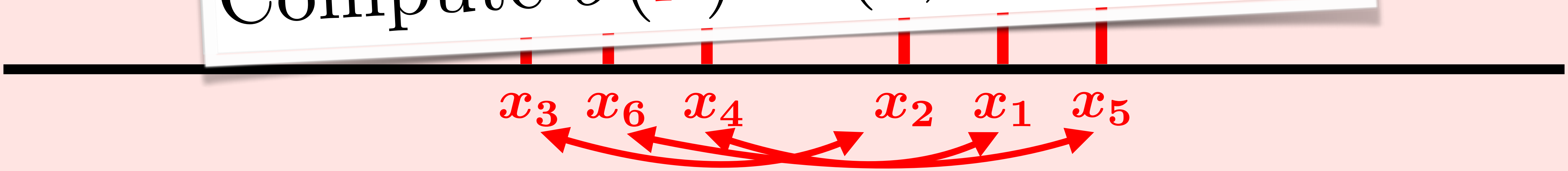
$R(\mathbf{x}) = (\mathbf{5}, \mathbf{4}, \mathbf{1}, \mathbf{3}, \mathbf{6}, \mathbf{2}),\ S(\mathbf{x}) = (\boldsymbol{x_3}, \boldsymbol{x_6}, \boldsymbol{x_4}, \boldsymbol{x_2}, \boldsymbol{x_1}, \boldsymbol{x_5})$

... *Set first* **Milestones** *in the race* ...

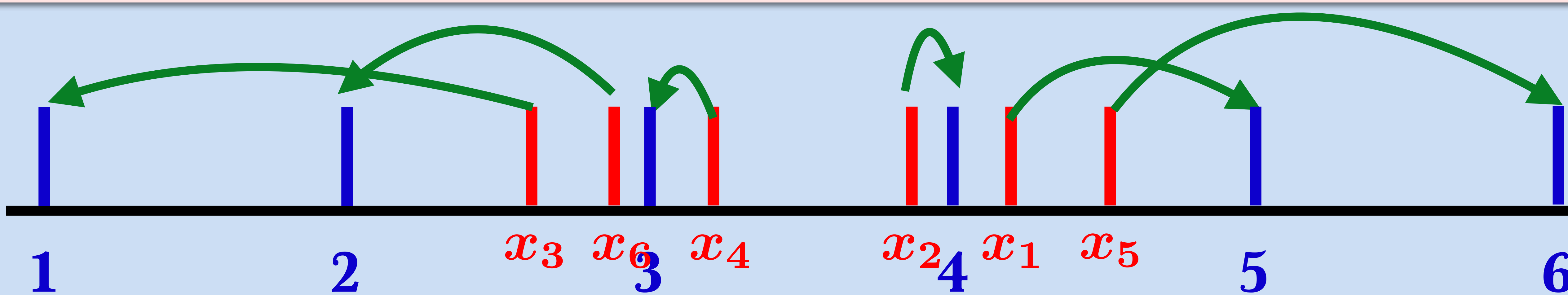


Two ways to rank and sort

Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



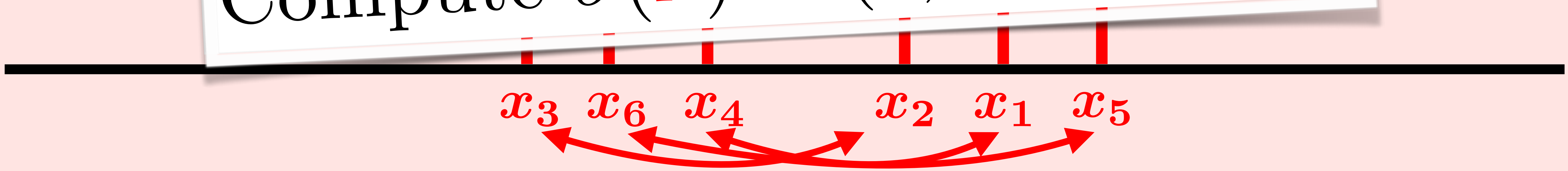
$R(\mathbf{x}) = (5, 4, 1, 3, 6, 2)$, $S(\mathbf{x}) = (x_3, x_6, x_4, x_2, x_1, x_5)$



Compute OT solution $P^\star$,

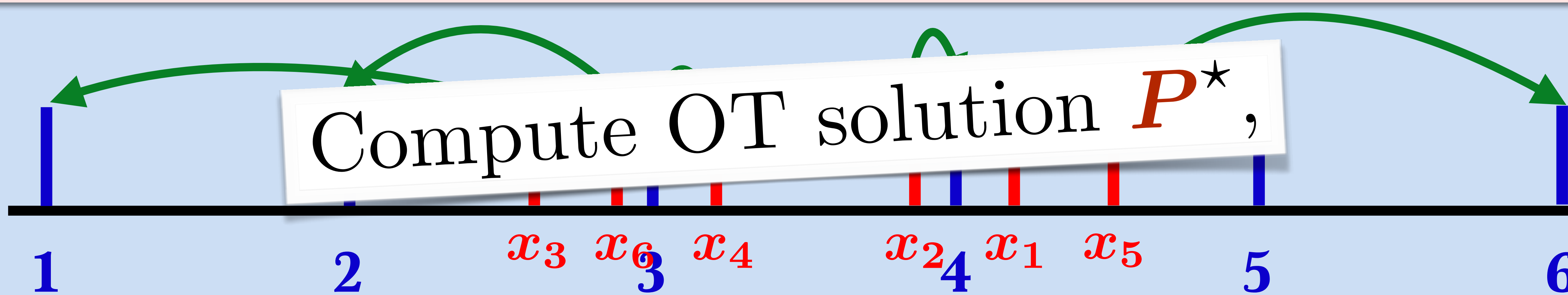
Two ways to rank and sort

Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



$$R(\mathbf{x}) = (\mathbf{5}, \mathbf{4}, \mathbf{1}, \mathbf{3}, \mathbf{6}, \mathbf{2}),\ S(\mathbf{x}) = (\boldsymbol{x_3}, \boldsymbol{x_6}, \boldsymbol{x_4}, \boldsymbol{x_2}, \boldsymbol{x_1}, \boldsymbol{x_5})$$

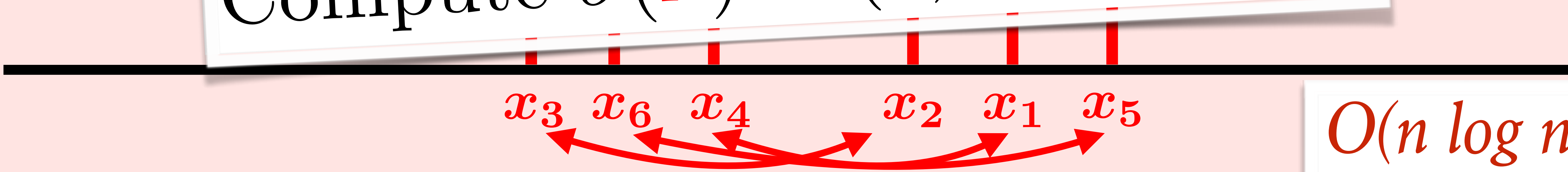
Compute OT solution $\boldsymbol{P}^\star$,



$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1/6} \\ \vdots \\ \mathbf{6/6} \end{bmatrix},\ S(\mathbf{x}) = n(\boldsymbol{P}^\star)^T \mathbf{x}$$

Two ways to rank and sort

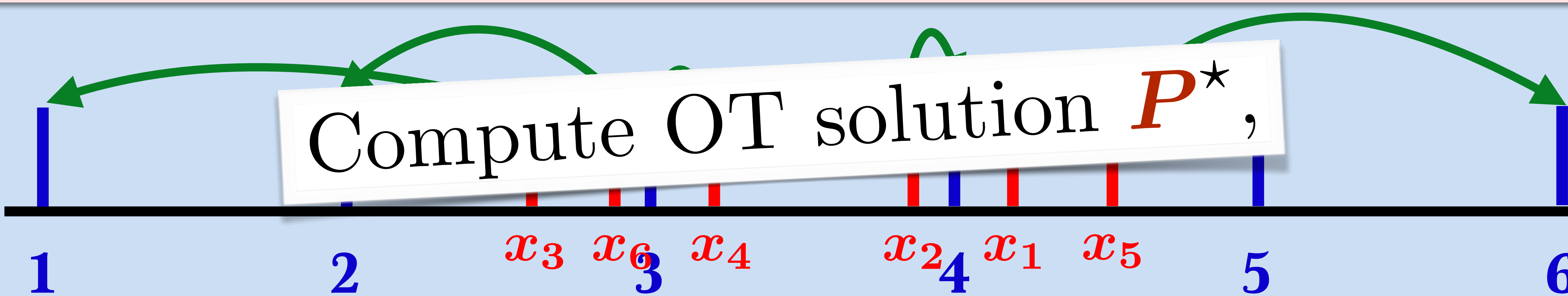
Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



$O(n \log n)$

$$R(\mathbf{x}) = (\mathbf{5}, \mathbf{4}, \mathbf{1}, \mathbf{3}, \mathbf{6}, \mathbf{2}),\ S(\mathbf{x}) = (\boldsymbol{x_3}, \boldsymbol{x_6}, \boldsymbol{x_4}, \boldsymbol{x_2}, \boldsymbol{x_1}, \boldsymbol{x_5})$$

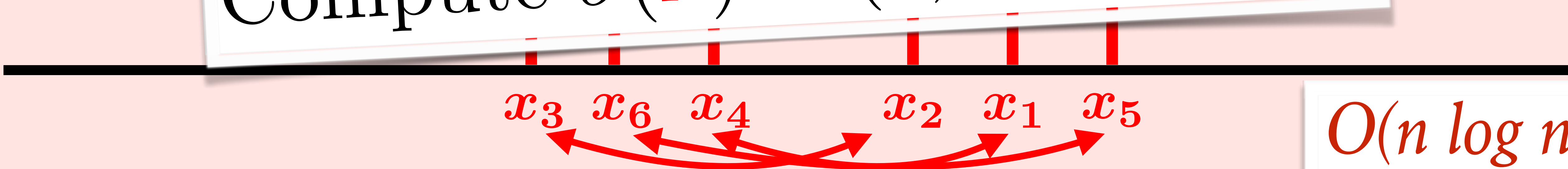
Compute OT solution $\boldsymbol{P}^\star$,



$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1/6} \\ \vdots \\ \mathbf{6/6} \end{bmatrix},\ S(\mathbf{x}) = n(\boldsymbol{P}^\star)^T \mathbf{x}$$

Two ways to rank and sort

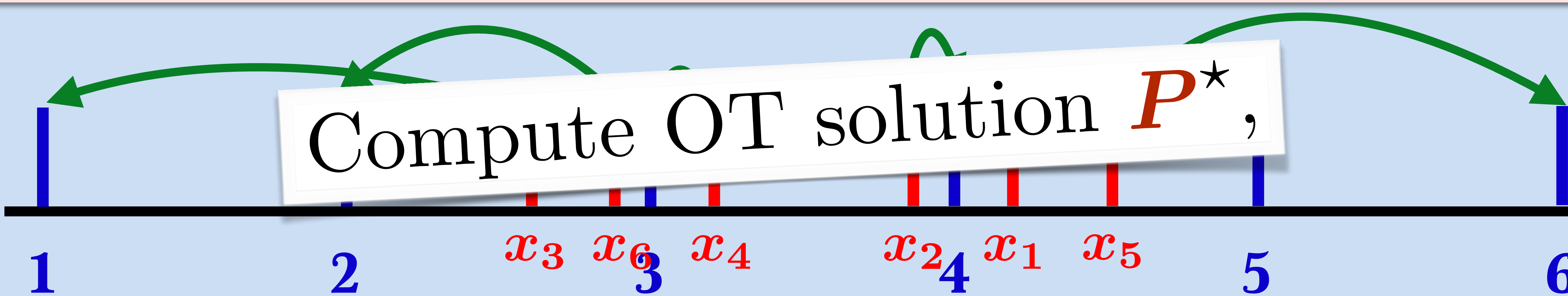
Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



$O(n \log n)$

$$R(\mathbf{x}) = (5, 4, 1, 3, 6, 2),\ S(\mathbf{x}) = (x_3, x_6, x_4, x_2, x_1, x_5)$$

Compute OT solution $P^\star$,

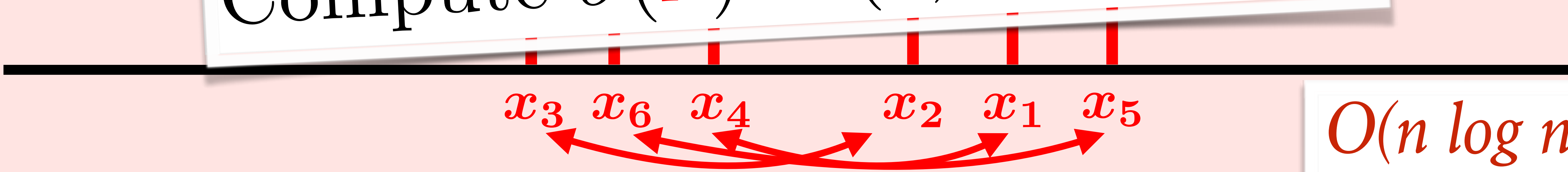


$$R(\mathbf{x}) = n^2 P^\star \begin{bmatrix} 1/6 \\ \vdots \\ 6/6 \end{bmatrix},\ S(\mathbf{x}) = n(P^\star)^T \mathbf{x}$$

$O(n^3 \log n)$

Two ways to rank and sort

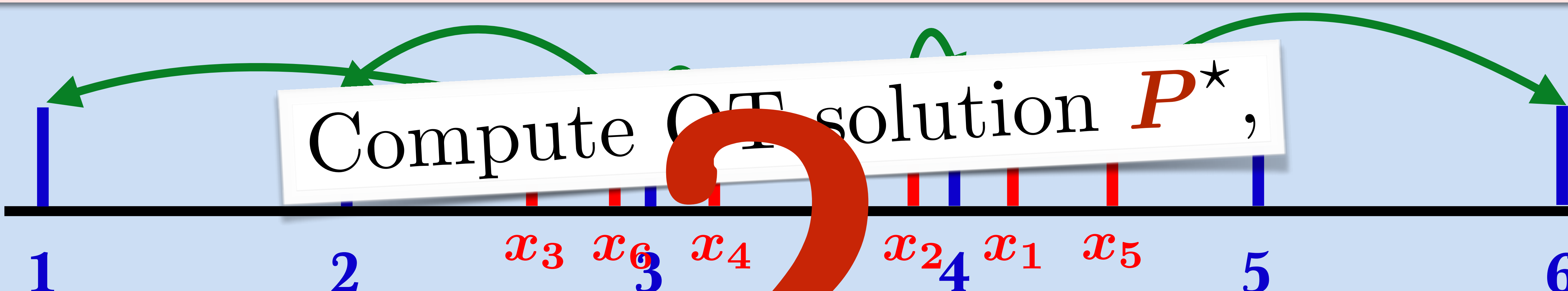
Compute $\sigma(\mathbf{x}) = (3, 6, 4, 2, 1, 5)$,



$O(n \log n)$

$$R(\mathbf{x}) = (\mathbf{5}, \mathbf{4}, \mathbf{1}, \mathbf{3}, \mathbf{6}, \mathbf{2}),\ S(\mathbf{x}) = (\boldsymbol{x_3}, \boldsymbol{x_6}, \boldsymbol{x_4}, \boldsymbol{x_2}, \boldsymbol{x_1}, \boldsymbol{x_5})$$

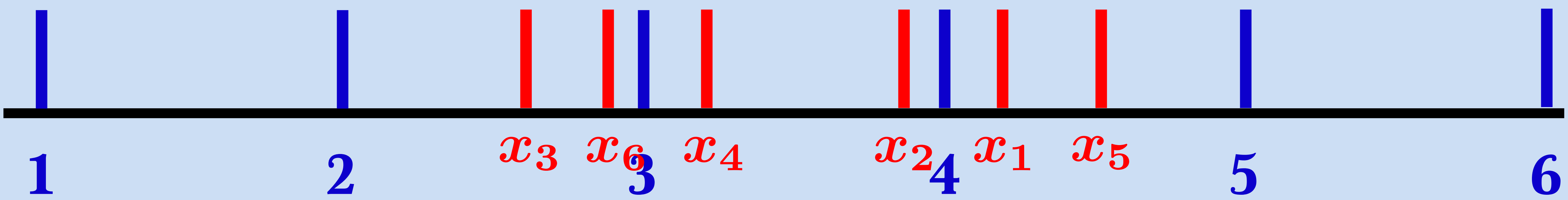
Compute OT solution $\boldsymbol{P}^\star$,



$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} 1/6 \\ \vdots \\ 6/6 \end{bmatrix},\ S(\mathbf{x}) = n(\boldsymbol{P}^\star)^T \mathbf{x}$$

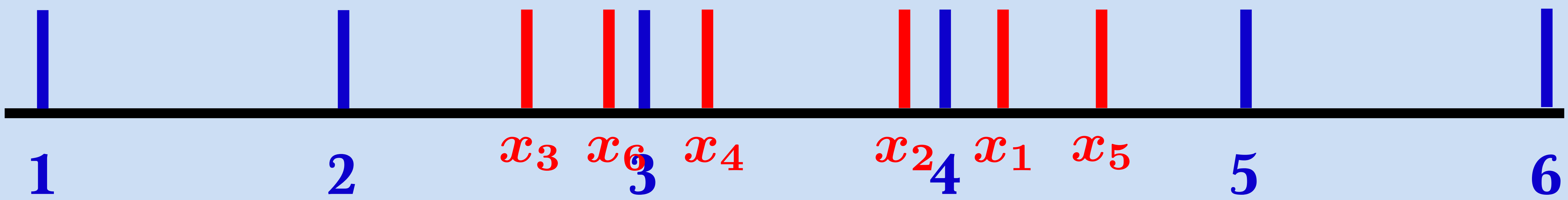
$O(n^3 \log n)$

Generalized Sorting and Ranking

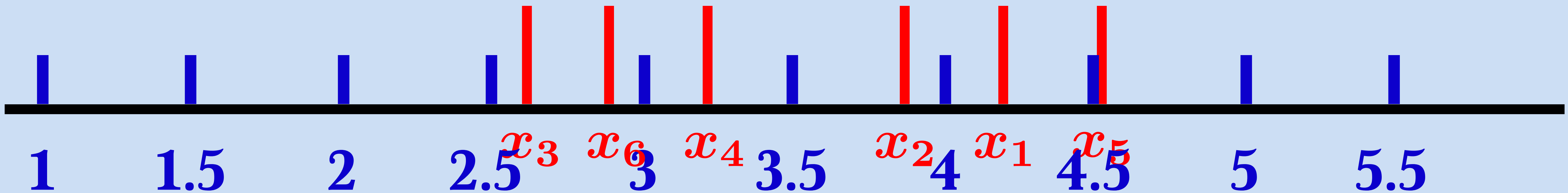


Generalized Sorting and Ranking

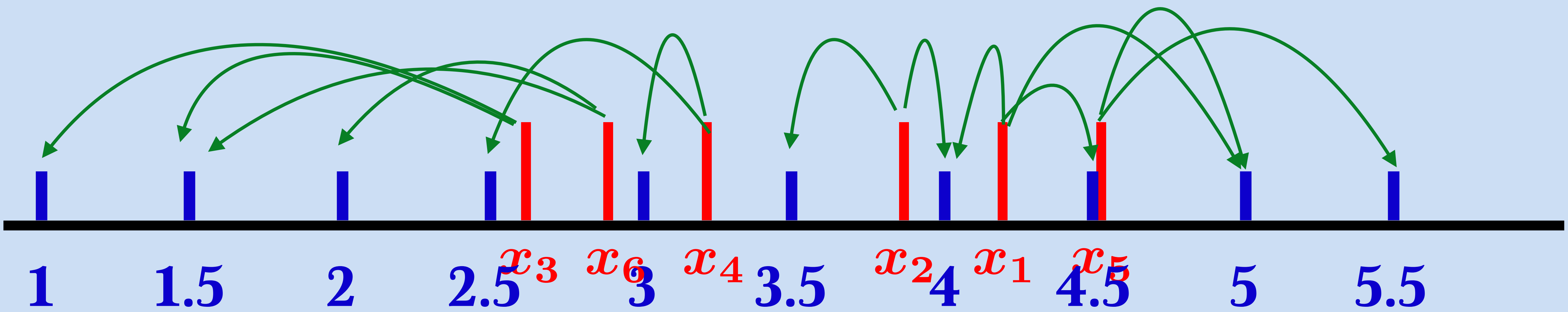
Interesting possibility: set a different number m *of* **Milestones**



Generalized Sorting and Ranking



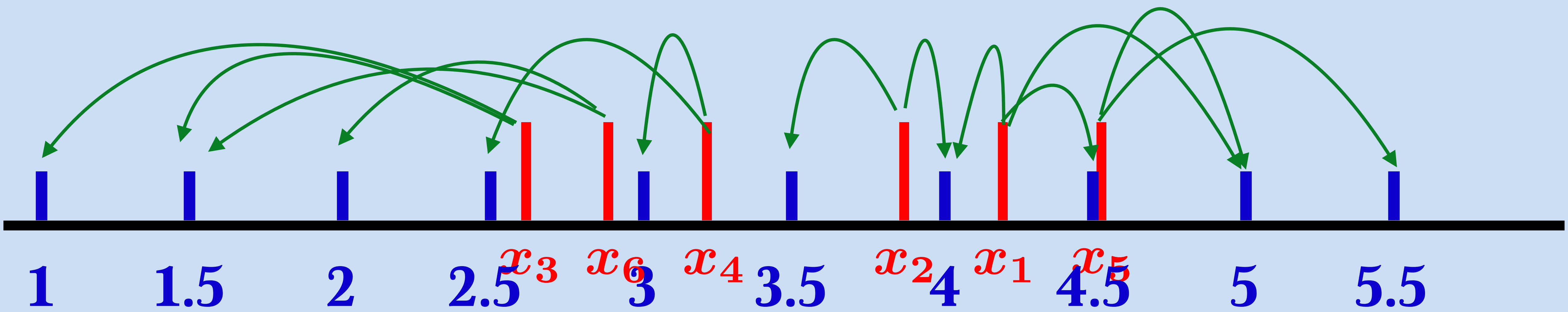
Generalized Sorting and Ranking



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1}/\boldsymbol{m} \\ \vdots \\ \boldsymbol{m}/\boldsymbol{m} \end{bmatrix}, \; S(\mathbf{x}) = \boldsymbol{m}(\boldsymbol{P}^\star)^T \mathbf{x}$$

Generalized Sorting and Ranking

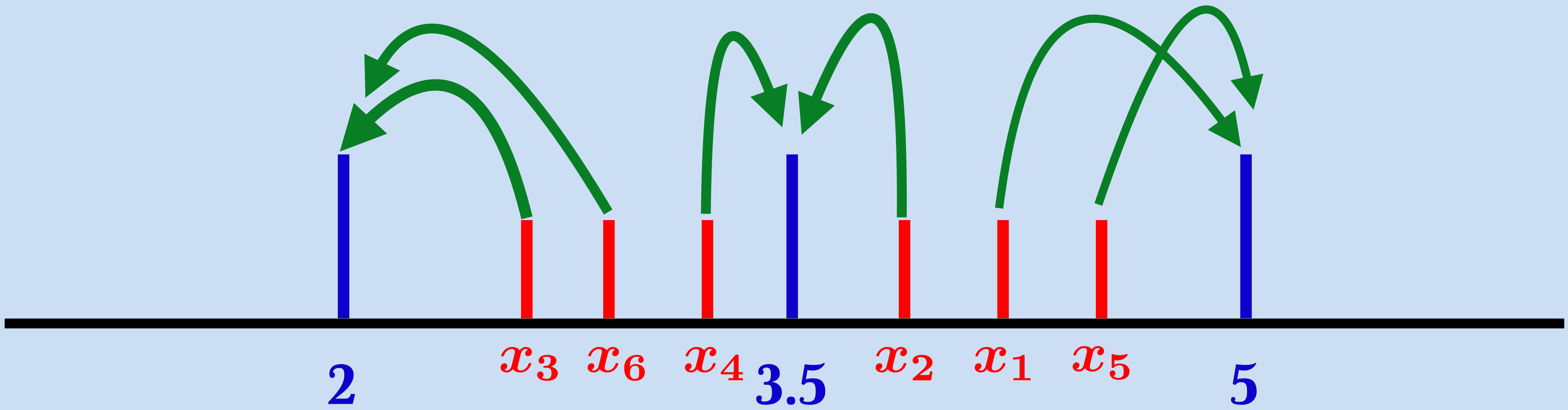


Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1}/\boldsymbol{m} \\ \vdots \\ \boldsymbol{m}/\boldsymbol{m} \end{bmatrix}, \; S(\mathbf{x}) = \boldsymbol{m}(\boldsymbol{P}^\star)^T \mathbf{x}$$

$\in \mathbb{R}^{\boldsymbol{n}}$ $\in \mathbb{R}^{\boldsymbol{m}}$

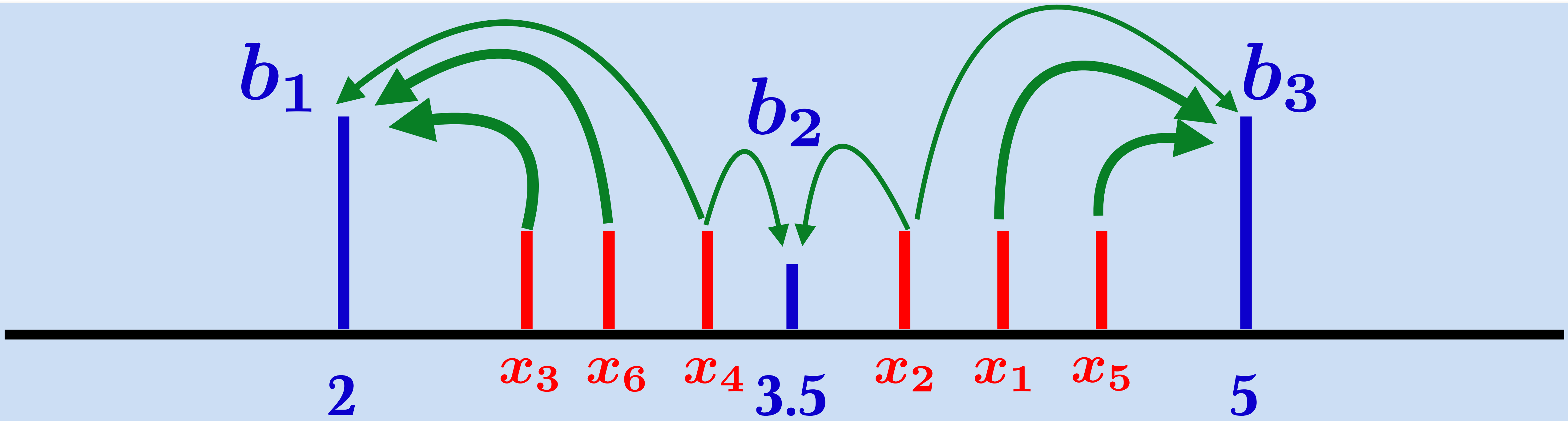
Less Milestones



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1}/\boldsymbol{m} \\ \vdots \\ \boldsymbol{m}/\boldsymbol{m} \end{bmatrix}, \; S(\mathbf{x}) = \boldsymbol{m}(\boldsymbol{P}^\star)^T \mathbf{x}$$

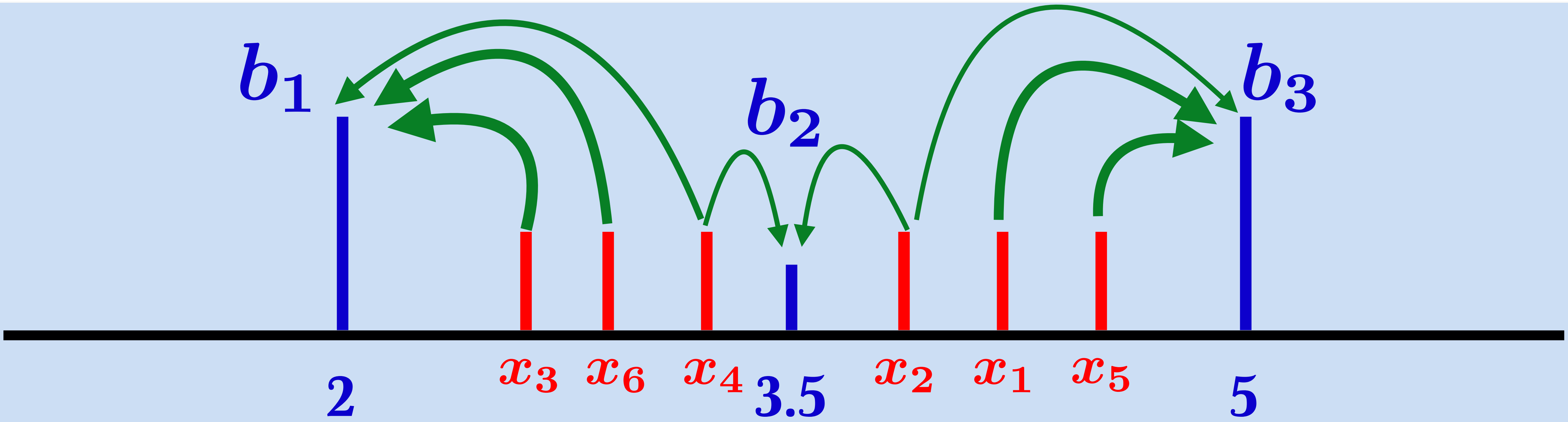
Weighted Milestones



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} 1/\boldsymbol{m} \\ \vdots \\ \boldsymbol{m}/\boldsymbol{m} \end{bmatrix}, \; S(\mathbf{x}) = \boldsymbol{m}(\boldsymbol{P}^\star)^T \mathbf{x}$$

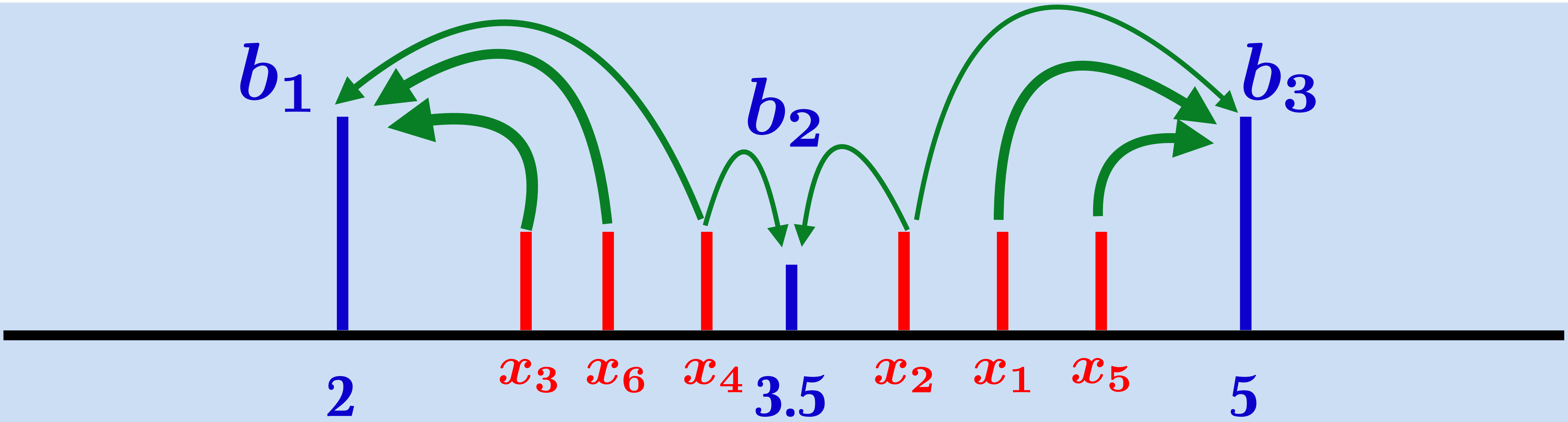
Weighted Milestones



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \begin{bmatrix} \mathbf{1}/\boldsymbol{m} \\ \vdots \\ \boldsymbol{m}/\boldsymbol{m} \end{bmatrix}, \; S(\mathbf{x}) = \boldsymbol{m}(\boldsymbol{P}^\star)^T \mathbf{x}$$

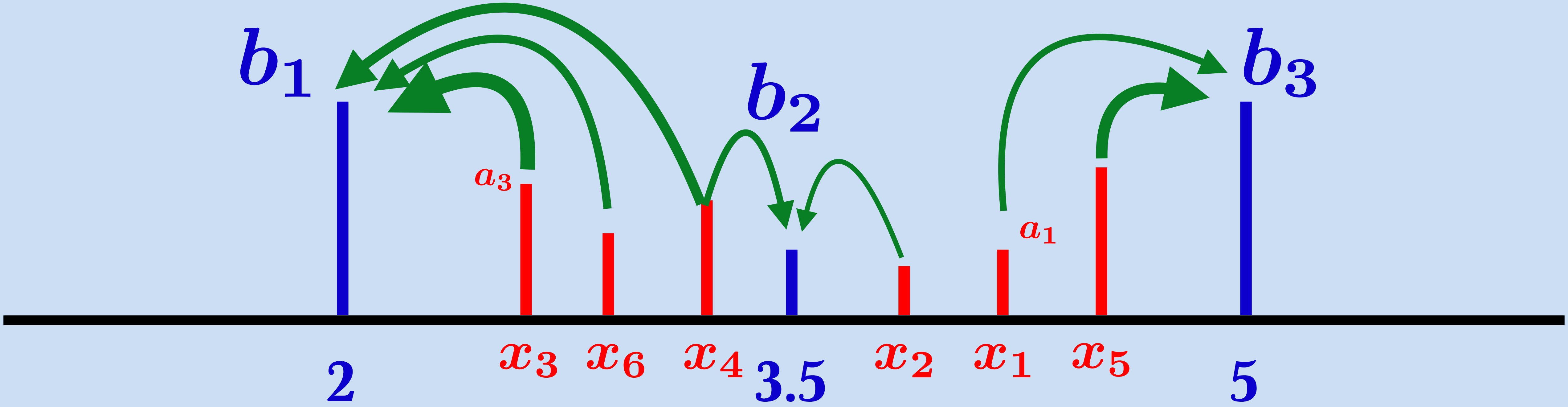
Weighted Milestones



Compute $n \times m$ OT solution $P^\star$,

$$R(\mathbf{x}) = n^2 P^\star \mathrm{cs}(b),\ S(\mathbf{x}) = b^{-1} \circ (P^\star)^T \mathbf{x}$$

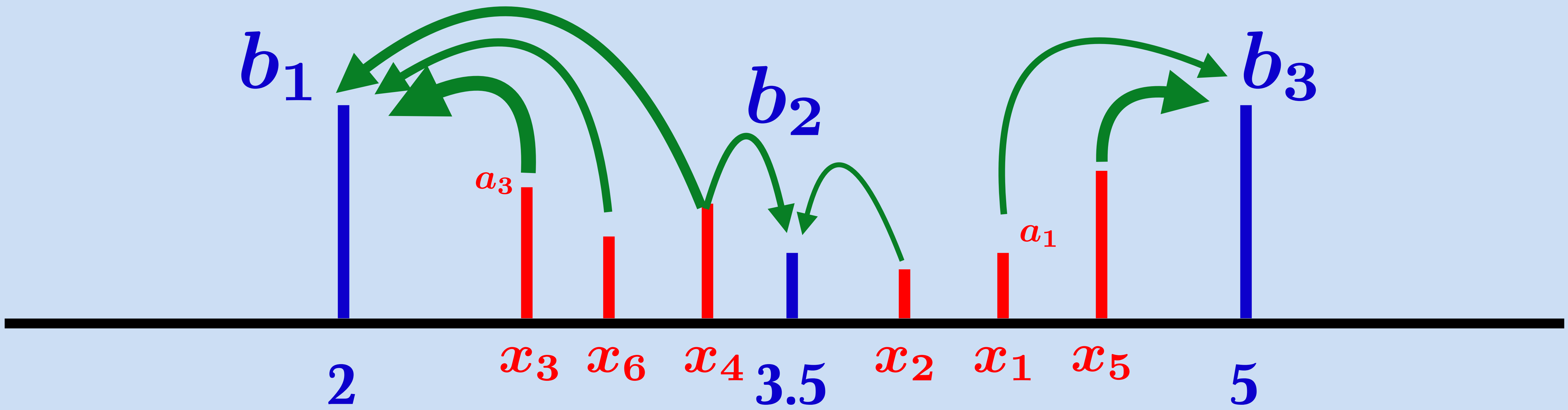
Weighted Inputs and Milestones



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\ S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

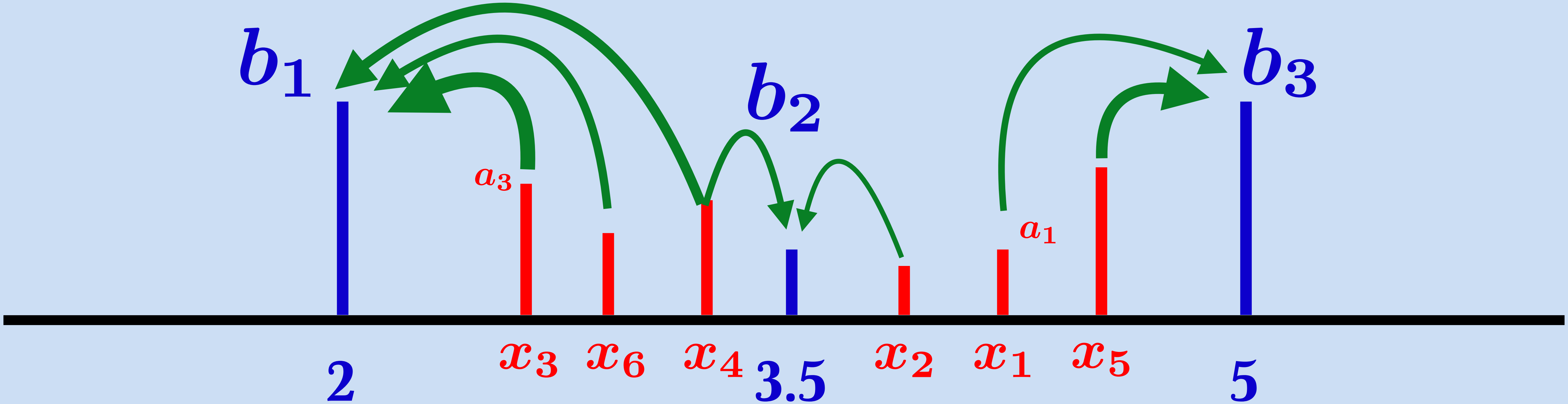
Weighted Inputs and Milestones



Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = n^2 \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\ S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

Weighted Inputs and Milestones



Compute $n \times m$ OT solution $P^\star$,

$$R(\mathbf{x}) = n\, a^{-1} \circ P^\star \mathrm{cs}(b),\; S(\mathbf{x}) = b^{-1} \circ (P^\star)^T \mathbf{x}$$

Weighted Inputs and Milestones

Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = \boldsymbol{n}\,\boldsymbol{a}^{-1} \circ \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\; S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

Weighted Inputs and Milestones

Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = \boldsymbol{n}\,\boldsymbol{a}^{-1} \circ \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\ S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

All issues do remain however !

(1) Still not differentiable *w.r.t* inputs

(2) Still very costly generalisation

Weighted Inputs and Milestones

Compute $n \times m$ OT solution $P^\star$,

$$R(\mathbf{x}) = n\, a^{-1} \circ P^\star \mathrm{cs}(b),\; S(\mathbf{x}) = b^{-1} \circ (P^\star)^T \mathbf{x}$$

All issues do remain however !

(1) Still not differentiable *w.r.t* inputs

(2) Still very costly generalisation

Optimal Transport

$O((n+m)nm \log(n+m))$

generalize both using OT (overkill!!)

Ranking / Sorting

Weighted Inputs and Milestones

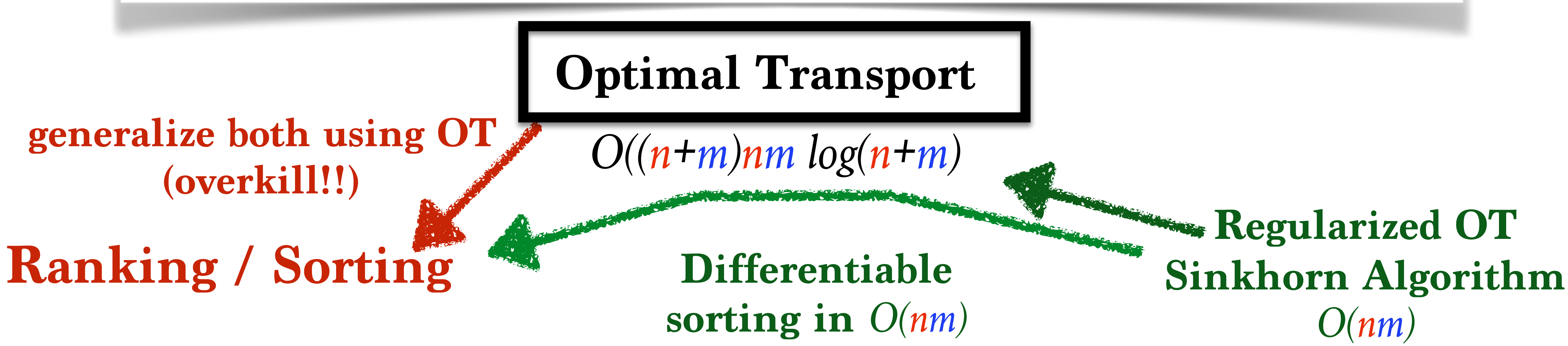
Compute $n \times m$ OT solution $P^\star$,

$$R(\mathbf{x}) = n\, a^{-1} \circ P^\star \mathrm{cs}(b),\; S(\mathbf{x}) = b^{-1} \circ (P^\star)^T \mathbf{x}$$

All issues do remain however !

(1) Still not differentiable *w.r.t* inputs

(2) Still very costly generalisation



Entropic Regularization [Wilson'68]

Def. Regularized OT, $\varepsilon \geq 0$

$$\min_{\mathbf{P} \in U(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{C}_{\mathbf{XY}} \rangle - \varepsilon E(\mathbf{P})$$

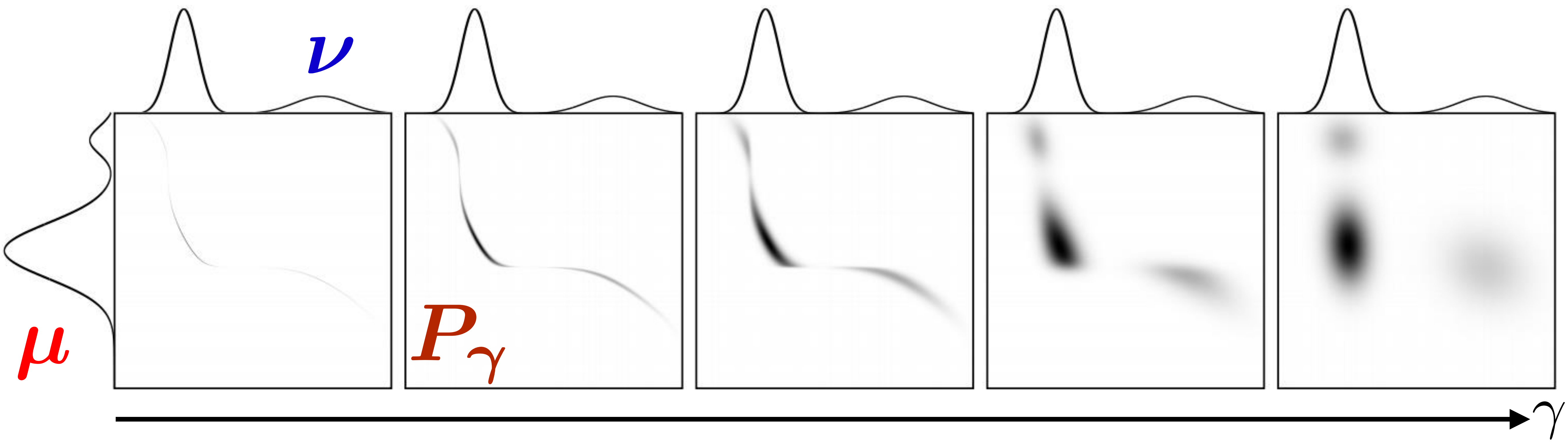
$$E(P) \stackrel{\text{def}}{=} -\sum_{i,j=1}^{nm} P_{ij}(\log P_{ij} - 1)$$

Note: Unique optimal solution thanks to strong concavity of entropy

Entropic Regularization [Wilson'68]

Def. Regularized OT, $\varepsilon \geq 0$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle - \varepsilon E(P)$$

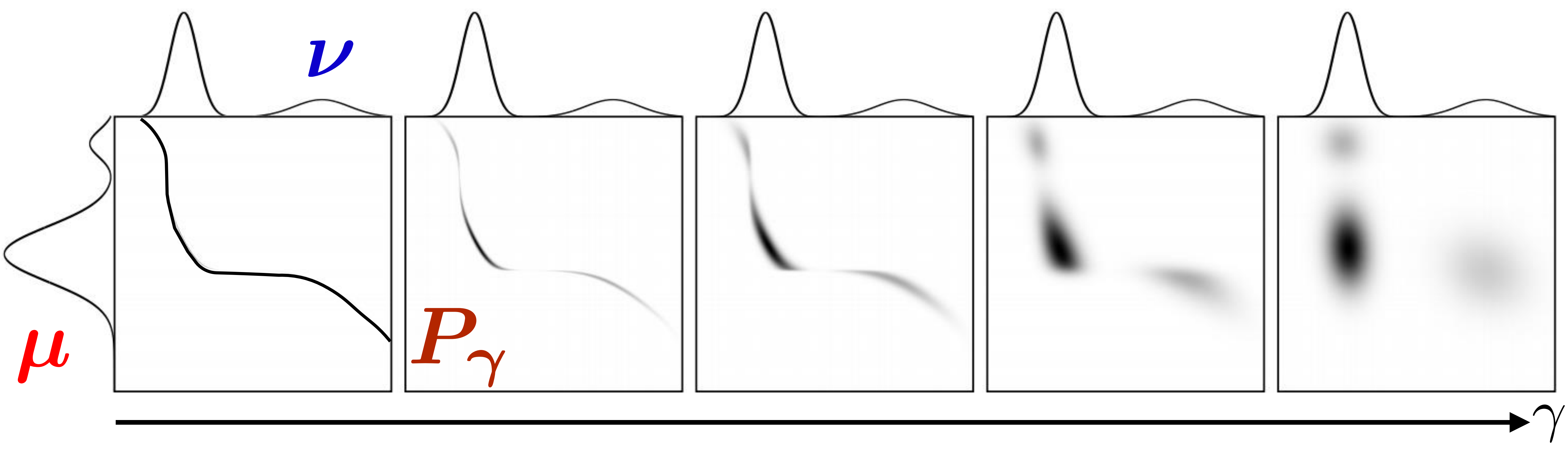


Note: Unique optimal solution thanks to strong concavity of entropy

Entropic Regularization [Wilson'68]

Def. Regularized OT, $\varepsilon \geq 0$

$$\min_{P \in U(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{C}_{\mathbf{XY}} \rangle - \varepsilon E(\mathbf{P})$$

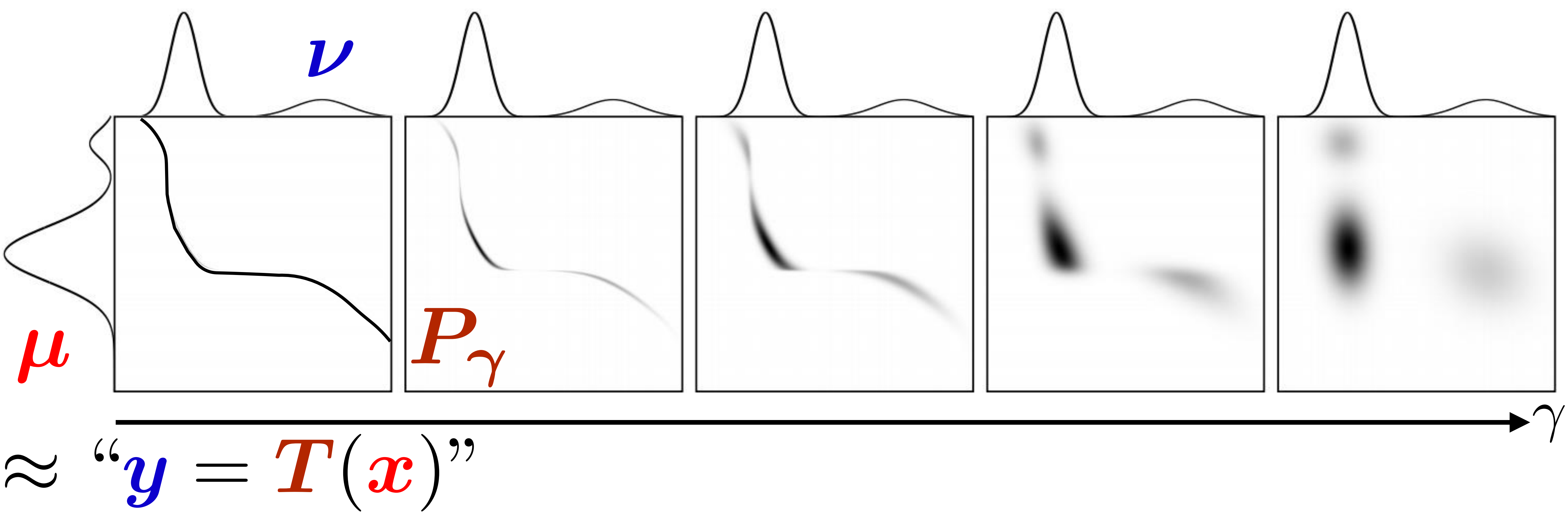


Note: Unique optimal solution thanks to strong concavity of entropy

Entropic Regularization [Wilson'68]

Def. Regularized OT, $\varepsilon \geq 0$

$$\min_{P \in U(a,b)} \langle P, C_{XY} \rangle - \varepsilon E(P)$$



Note: Unique optimal solution thanks to strong concavity of entropy

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})}{\text{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}^n_+, \boldsymbol{v} \in \mathbb{R}^m_+$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \overset{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}^n_+, \boldsymbol{v} \in \mathbb{R}^m_+$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \overset{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

$$L(P,\alpha,\beta) = \sum_{ij} P_{ij} C_{ij} + \boldsymbol{\varepsilon} P_{ij}(\log P_{ij} - 1) + \alpha^T(P\mathbf{1} - \boldsymbol{a}) + \beta^T(P^T\mathbf{1} - \boldsymbol{b})$$

$$\partial L / \partial P_{ij} = C_{ij} + \boldsymbol{\varepsilon} \log P_{ij} + \alpha_i + \beta_j$$

$$(\boldsymbol{\partial L/\partial P_{ij}} = 0) \Rightarrow P_{ij} = \boldsymbol{e}^{\frac{\boldsymbol{\alpha_i}}{\boldsymbol{\varepsilon}}} e^{-\frac{C_{ij}}{\boldsymbol{\varepsilon}}} \boldsymbol{e}^{\frac{\boldsymbol{\beta_j}}{\boldsymbol{\varepsilon}}} = \boldsymbol{u_i} K_{ij} \boldsymbol{v_j}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

$$P_\gamma \in U(\boldsymbol{a}, \boldsymbol{b}) \Leftrightarrow \begin{cases} \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}) \mathbf{1}_m & = \boldsymbol{a} \\ \mathbf{diag}(\boldsymbol{v}) K^T \mathbf{diag}(\boldsymbol{u}) \mathbf{1}_n & = \boldsymbol{b} \end{cases}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \underset{\boldsymbol{P}\in U(\boldsymbol{a},\boldsymbol{b})}{\text{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists ! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

$$P_\gamma \in U(\boldsymbol{a}, \boldsymbol{b}) \Leftrightarrow \begin{cases} \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}) \mathbf{1}_m & = \boldsymbol{a} \\ \mathbf{diag}(\boldsymbol{v}) K^T \underbrace{\mathbf{diag}(\boldsymbol{u}) \mathbf{1}_n}_{\boldsymbol{u}} & = \boldsymbol{b} \end{cases}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \underset{\boldsymbol{P}\in U(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}^n_+, \boldsymbol{v} \in \mathbb{R}^m_+$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

$$P_\gamma \in U(\boldsymbol{a}, \boldsymbol{b}) \Leftrightarrow \begin{cases} \mathbf{diag}(\boldsymbol{u}) K \overbrace{\mathbf{diag}(\boldsymbol{v}) \mathbf{1}_m}^{\boldsymbol{v}} & = \boldsymbol{a} \\ \mathbf{diag}(\boldsymbol{v}) K^T \underbrace{\mathbf{diag}(\boldsymbol{u}) \mathbf{1}_n}_{\boldsymbol{u}} & = \boldsymbol{b} \end{cases}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{a}, \boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C}_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists ! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\mathbf{XY}}/\boldsymbol{\varepsilon}}$$

$$P_\gamma \in U(\boldsymbol{a}, \boldsymbol{b}) \Leftrightarrow \begin{cases} \mathbf{diag}(\boldsymbol{u}) K \boldsymbol{v} & = \boldsymbol{a} \\ \mathbf{diag}(\boldsymbol{v}) K^T \boldsymbol{u} & = \boldsymbol{b} \end{cases}$$

Fast & Scalable Algorithm

$$\textbf{Prop.} \text{ If } \boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})} \langle \boldsymbol{P}, \boldsymbol{C}_{\boldsymbol{XY}} \rangle - \gamma E(\boldsymbol{P})$$

then $\exists! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\boldsymbol{XY}}/\boldsymbol{\varepsilon}}$$

$$P_\gamma \in U(\boldsymbol{a}, \boldsymbol{b}) \Leftrightarrow \begin{cases} \boldsymbol{u} \odot K \boldsymbol{v} = \boldsymbol{a} \\ \boldsymbol{v} \odot K^T \boldsymbol{u} = \boldsymbol{b} \end{cases}$$

Fast & Scalable Algorithm

Prop. If $\boldsymbol{P}_{\boldsymbol{\varepsilon}} \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})} \langle \boldsymbol{P}, \boldsymbol{C}_{\boldsymbol{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}^n_+, \boldsymbol{v} \in \mathbb{R}^m_+$, such that

$$\boldsymbol{P}_{\boldsymbol{\varepsilon}} = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-\boldsymbol{C}_{\boldsymbol{XY}}/\boldsymbol{\varepsilon}}$$

Sinkhorn's Algorithm : Repeat

1. $\boldsymbol{u} = \boldsymbol{a} / K\boldsymbol{v}$
2. $\boldsymbol{v} = \boldsymbol{b} / K^T\boldsymbol{u}$

Fast & Scalable Algorithm

Sinkhorn's Algorithm : repeat

1. $\boldsymbol{u} = \boldsymbol{a}/K\boldsymbol{v}$
2. $\boldsymbol{v} = \boldsymbol{b}/K^T\boldsymbol{u}$

- **[Sinkhorn'64]** proved first convergence
- Speed: **[Lorenz'89]** proved linear convergence, see **[Altschuler'17]** **[Dvurechensky'18]** for new results.
- $O(nm)$ complexity, GPGPU parallel **[C'13]** .
- $O(n \log n)$ using convolutions **[Solomon+'15]**
- $O(nk)$ using clever low rank K **[Altschuler+'18/19]**

Sinkhorn

Def. For $L \geq 1$, define $K \stackrel{\text{def}}{=} e^{-C_{\boldsymbol{X}\boldsymbol{Y}}/\varepsilon}$

$$\boldsymbol{P_L} \stackrel{\text{def}}{=} \mathbf{diag}(\boldsymbol{u_L}) K \mathbf{diag}(\boldsymbol{v_L}), \text{ where}$$

$$\boldsymbol{v_0} = \mathbf{1}_m; l \geq 0, \boldsymbol{u_l} \stackrel{\text{def}}{=} \boldsymbol{a}/K\boldsymbol{v_l}, \boldsymbol{v_{l+1}} \stackrel{\text{def}}{=} \boldsymbol{b}/K^T\boldsymbol{u_l}.$$

Prop. $\frac{\partial \boldsymbol{P_L}}{\partial \boldsymbol{X}}, \frac{\partial \boldsymbol{P_L}}{\partial \boldsymbol{a}}$ can be computed recursively, in $O(L)$ kernel $K \times$ vector products.

Sinkhorn Ranks and Sorts

Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = \boldsymbol{n}\,\boldsymbol{a}^{-1} \circ \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\; S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

Sinkhorn Ranks and Sorts

Compute $\boldsymbol{n} \times \boldsymbol{m}$ OT solution $\boldsymbol{P}^\star$,

$$R(\mathbf{x}) = \boldsymbol{n}\,\boldsymbol{a}^{-1} \circ \boldsymbol{P}^\star \mathrm{cs}(\boldsymbol{b}),\ S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ (\boldsymbol{P}^\star)^T \mathbf{x}$$

Set $\boldsymbol{\varepsilon}$, compute $\boldsymbol{u}_{\boldsymbol{L}}, \boldsymbol{v}_{\boldsymbol{L}}$ in $\boldsymbol{nmL}$ ops

$$R(\mathbf{x}) = \boldsymbol{n}\,\boldsymbol{a}^{-1} \circ \boldsymbol{u}_{\boldsymbol{L}} \circ K \boldsymbol{v}_{\boldsymbol{L}} \circ \mathrm{cs}(\boldsymbol{b})$$

$$S(\mathbf{x}) = \boldsymbol{b}^{-1} \circ \boldsymbol{v}_{\boldsymbol{L}} \circ K^T \boldsymbol{u}_{\boldsymbol{L}} \circ \mathbf{x}$$

Sinkhorn Ranks and Sorts

Compute $n \times m$ OT solution $P^\star$,

$$R(\mathbf{x}) = n\, a^{-1} \circ P^\star \mathrm{cs}(b),\; S(\mathbf{x}) = b^{-1} \circ (P^\star)^T \mathbf{x}$$

Set ε, compute u_L, v_L in nmL ops

$$R(\mathbf{x}) = n\, a^{-1} \circ u_L \circ \; K v_L \circ \mathrm{cs}(b)$$

$$S(\mathbf{x}) = b^{-1} \circ v_L \circ \; K^T u_L \circ \mathbf{x}$$

Ranking / Sorting ← **Differentiable sorting in $O(nm)$** ← **Regularized OT Sinkhorn Algorithm**

The Devil is in the Details

Algorithm 2: Sinkhorn Ranks/Sorts

Inputs: $(\mathbf{a}_s, \mathbf{x}_s)_s \in (\Sigma_n \times \mathbb{R}^n)^S, (\mathbf{b}, \mathbf{y}) \in \Sigma_m \times \mathbb{O}_m, h, \varepsilon, \eta, \widetilde{g}$.
$\forall s, \widetilde{\mathbf{x}}_s = \widetilde{g}(\mathbf{x}_s)$, $C_s = [h(y_j - (\widetilde{\mathbf{x}}_s)_i)]_{ij}$, $\boldsymbol{\alpha}_s = \mathbf{0}_n, \boldsymbol{\beta}_s = \mathbf{0}_m$.
repeat

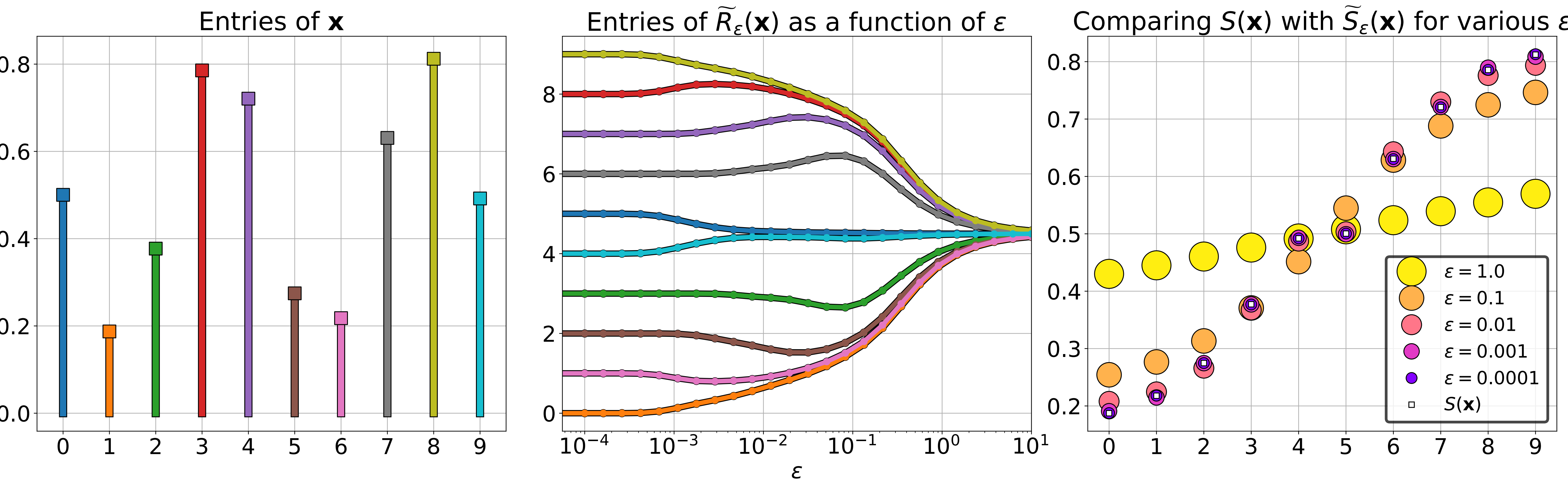
$\quad \forall s, \boldsymbol{\beta}_s \leftarrow \varepsilon \log \mathbf{b}_s + \min_\varepsilon \left(C_s^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right) + \boldsymbol{\beta}_s$

$\quad \forall s, \boldsymbol{\alpha}_s \leftarrow \varepsilon \log \mathbf{a}_s + \min_\varepsilon \left(C_s - \boldsymbol{\alpha}_s \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}_s^T\right) + \boldsymbol{\alpha}_s$

until $\max_s \Delta\left(\exp\left(C_{\mathbf{x}_s\mathbf{y}}^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right)\mathbf{1}_n, \mathbf{b}\right) < \eta$;
$\forall s, \widetilde{R}_\varepsilon(\mathbf{x}_s) \leftarrow \mathbf{a}_s^{-1} \circ \exp\left(C_{\mathbf{x}_s\mathbf{y}} - \boldsymbol{\alpha}_s \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}_s^T\right) \overline{\mathbf{b}}$,
$\forall s, \widetilde{S}_\varepsilon(\mathbf{x}_s) \leftarrow \mathbf{b}_s^{-1} \circ \exp\left(C_{\mathbf{x}_s\mathbf{y}}^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right) \mathbf{x}_s$.
Result: $\left(\widetilde{R}_\varepsilon(\mathbf{x}_s), \widetilde{S}_\varepsilon(\mathbf{x}_s)\right)_s$.

$$\tilde{g} : \mathbf{x} \mapsto g\left(\frac{\mathbf{x} - (\mathbf{x}^T \mathbf{1}_n)\mathbf{1}_n}{\frac{1}{\sqrt{n}} \|\mathbf{x} - (\mathbf{x}^T \mathbf{1}_n)\mathbf{1}_n\|_2}\right).$$

Sinkhorn Sort, Ranks, Quantiles



Applications

- **Soft-ranks** can tell differentiably if a point is close to desired rank among its peers.
- We use this directly in ML, to replace Softmax/KL losses by a differentiable approx to 0/1 or more generally *top-k* loss among L labels.

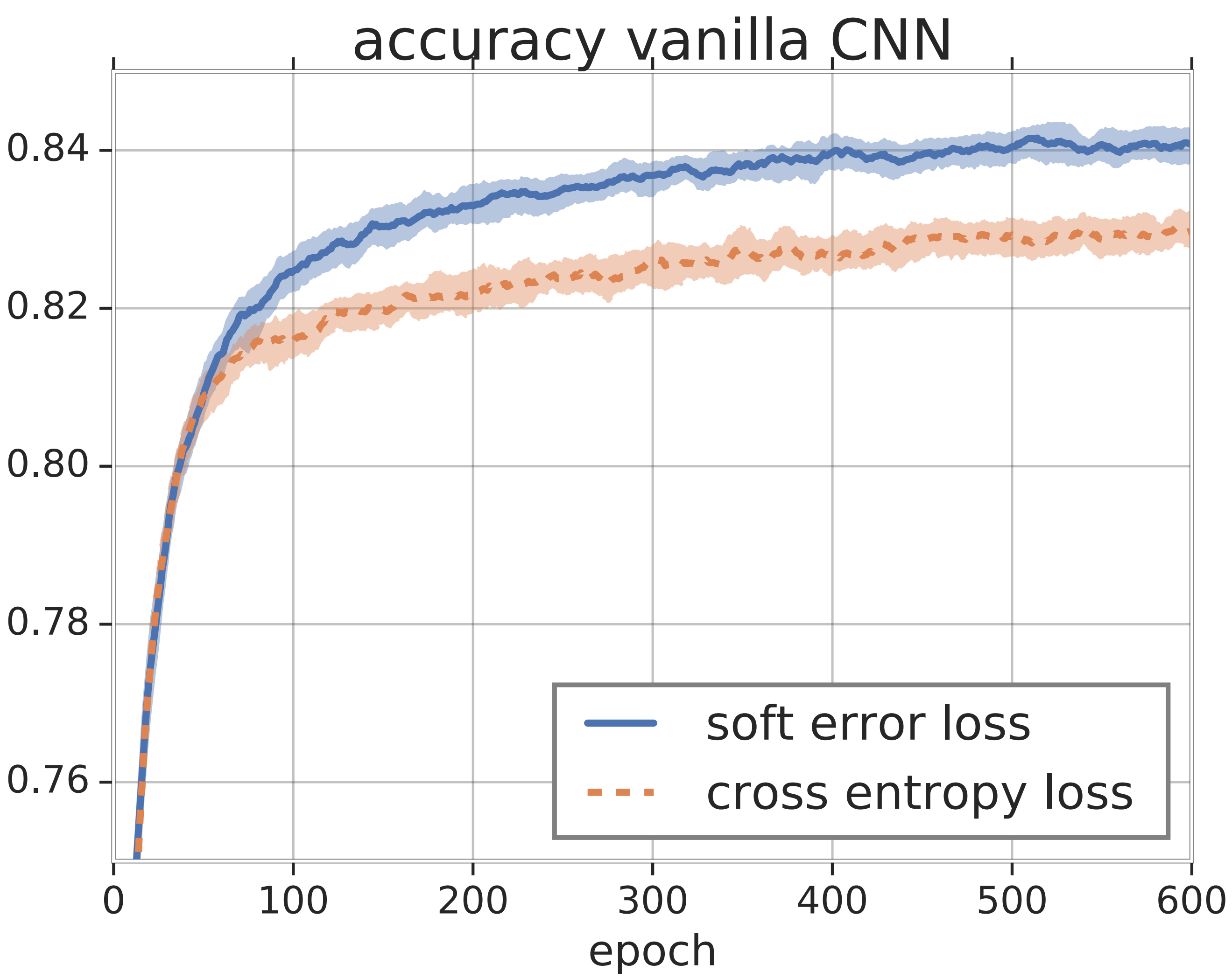
$$\mathcal{L}_{0/1}(f_\theta(\omega), l) = H\left(L - [R(f_\theta(\omega))]_l\right)$$

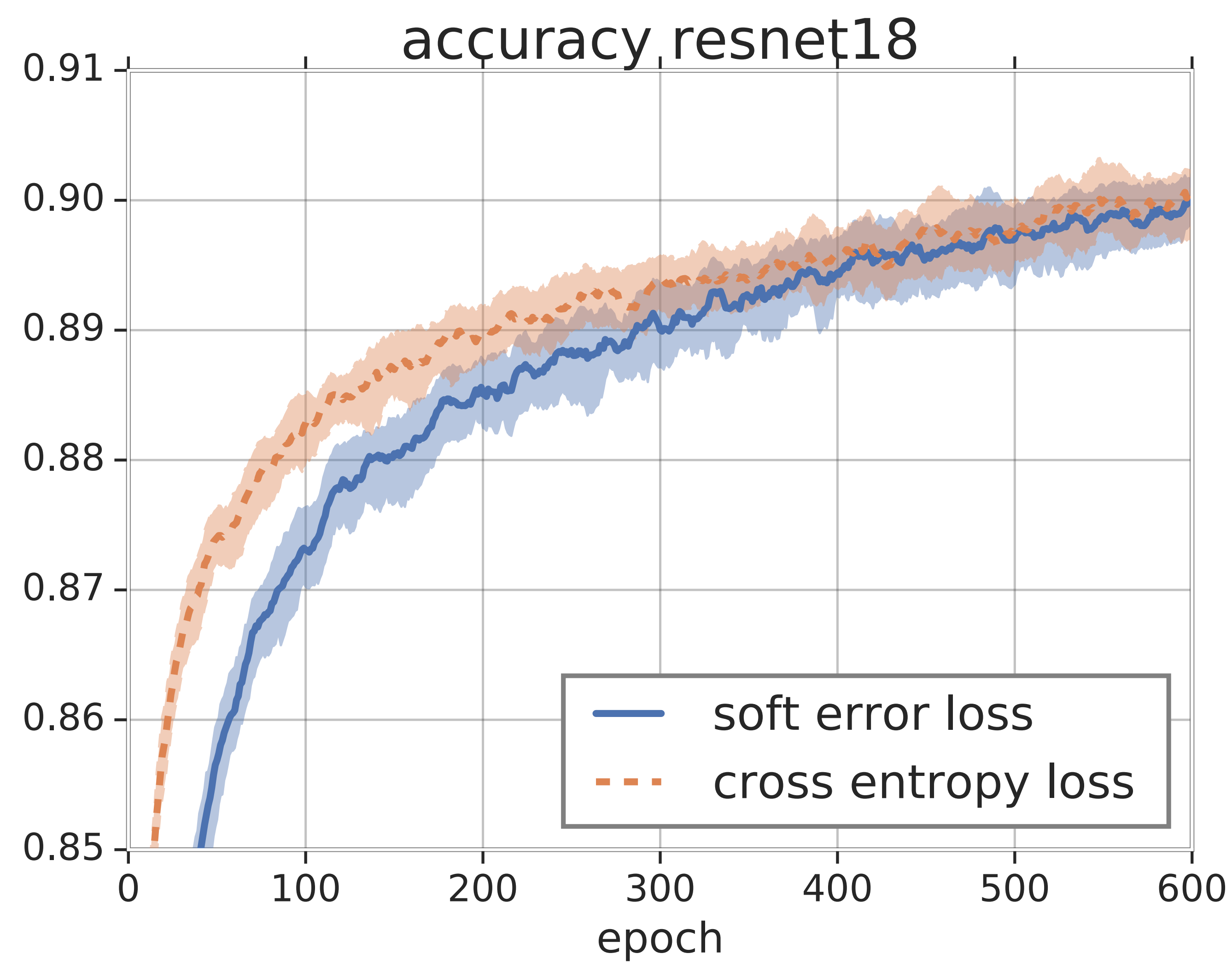
Heaviside

$$\widetilde{\mathcal{L}}_{k,\varepsilon}(f_\theta(\omega), l) = J_k\left(L - \left[\widetilde{R}_\varepsilon\left(\frac{\mathbf{1}_L}{L}, f_\theta(\omega); \frac{\mathbf{1}_L}{L}, \frac{\overline{\mathbf{1}}_L}{L}, h\right)\right]_l\right)$$

Any soft/continuous H like function.

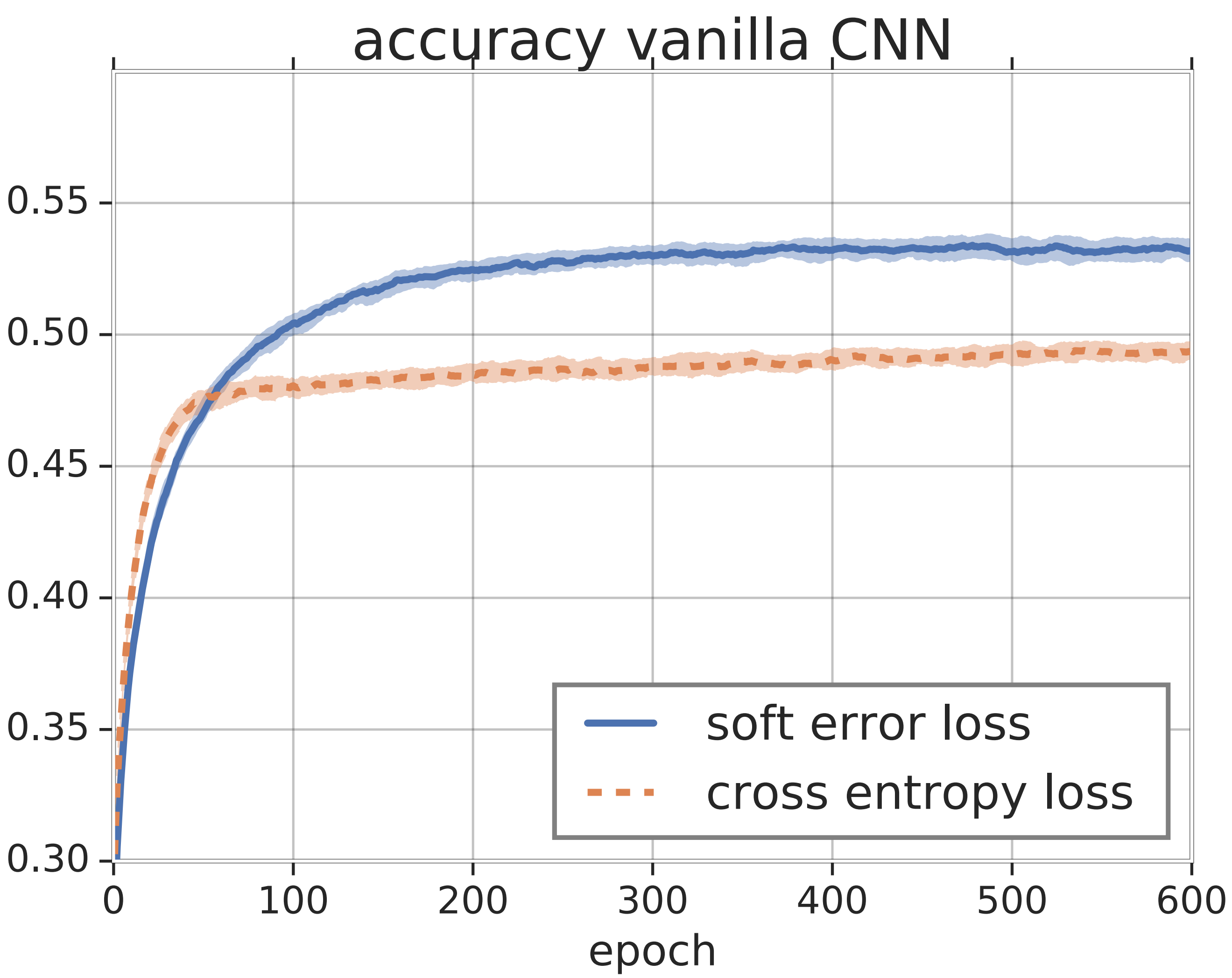
CIFAR-10



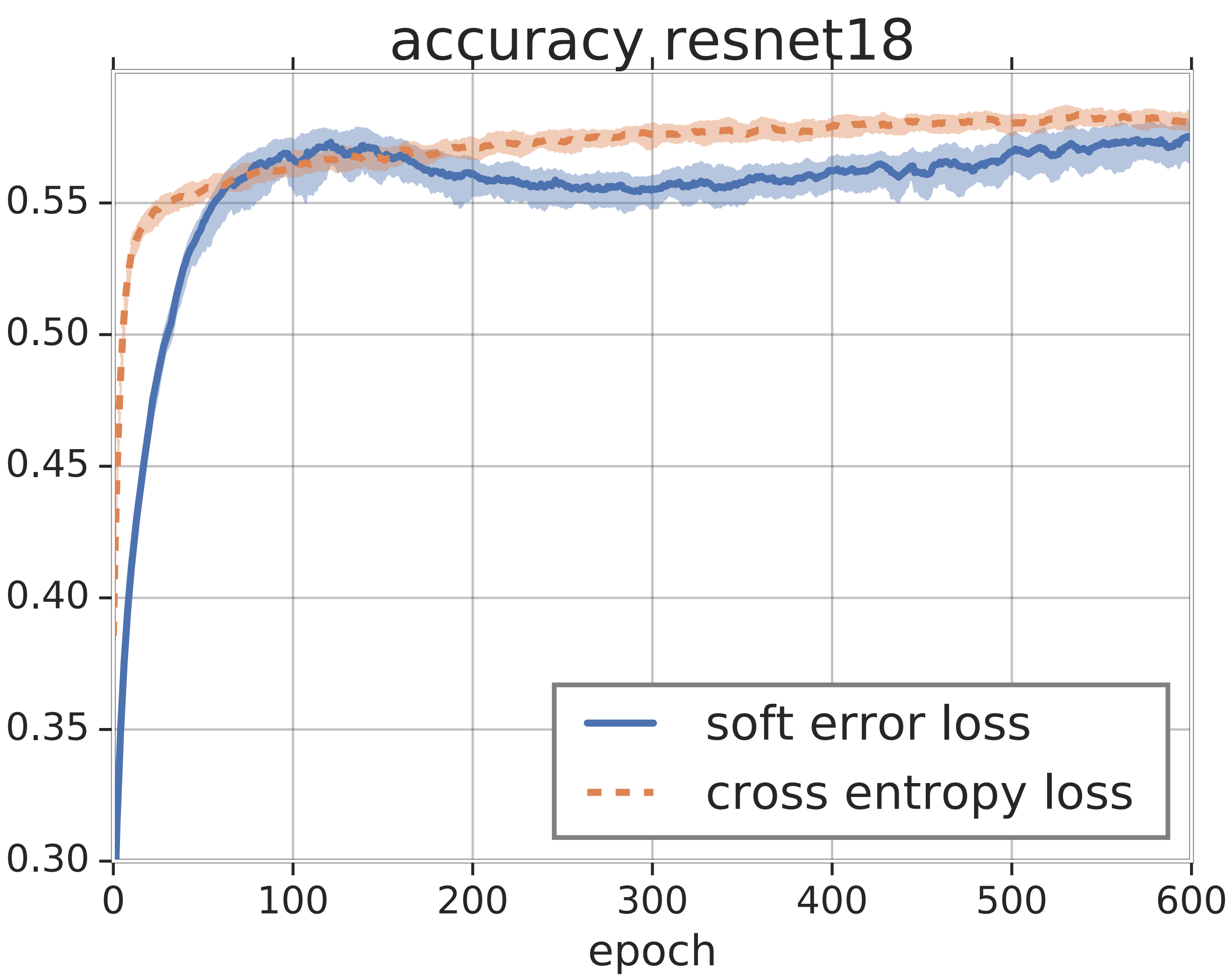


accuracy vanilla CNN
accuracy resnet18

CIFAR-100

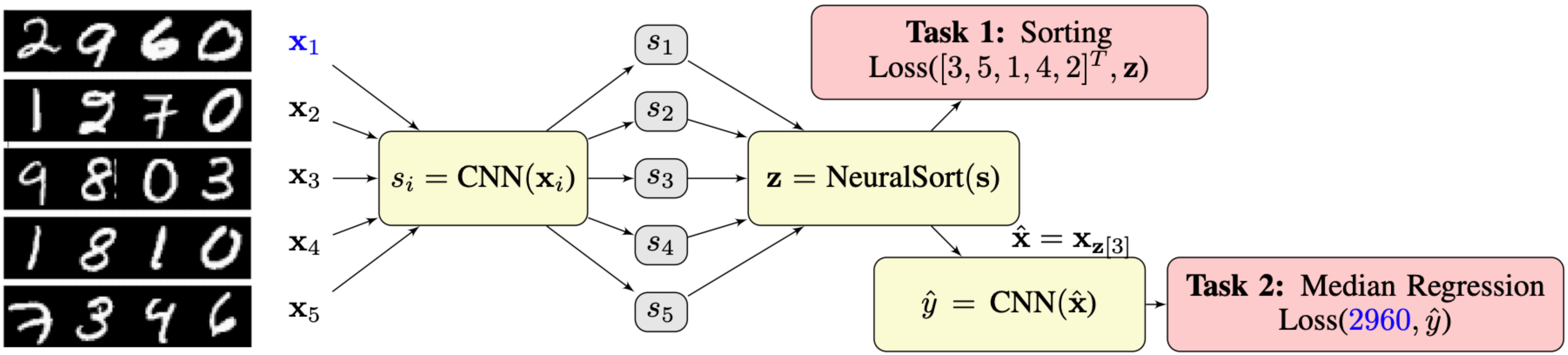


accuracy vanilla CNN
0.55
0.50
0.45
0.40
0.35
0.30
0
100
200
300
400
500
600
epoch
soft error loss
cross entropy loss

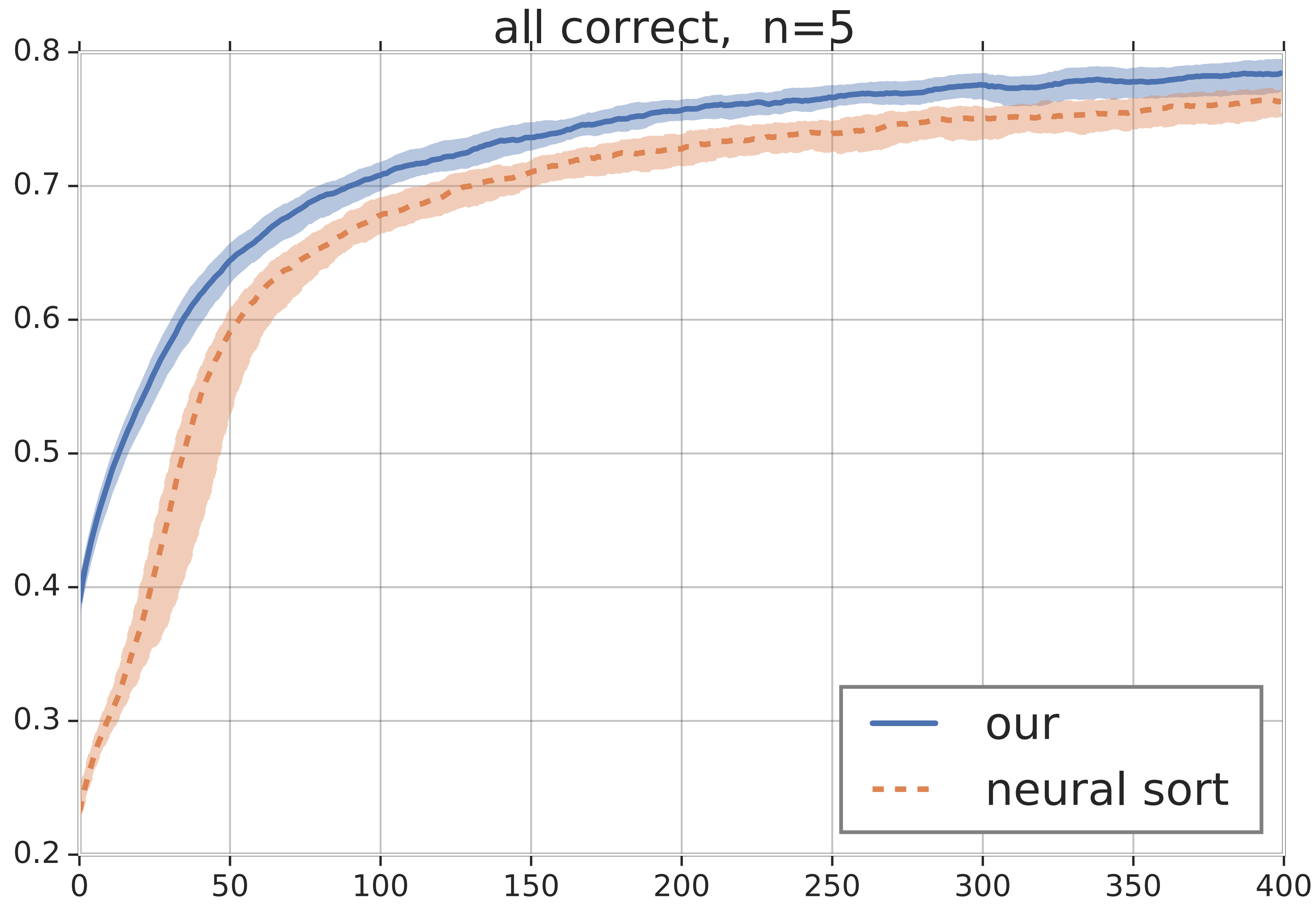


accuracy resnet18
0.55
0.50
0.45
0.40
0.35
0.30
0
100
200
300
400
500
600
epoch
soft error loss
cross entropy loss

Comparison with Neuralsort



Comparison with Neuralsort



algorithm	n=3	n=5	n=7	n=9	n=15
Stochastic NeuralSort	0.920 (0.946)	0.790 (0.907)	0.636 (0.873)	0.452 (0.829)	0.122 (0.734)
Deterministic NeuralSort	0.919 (0.945)	0.777 (0.901)	0.610 (0.862)	0.434 (0.824)	0.097 (0.716)
Our	**0.928 (0.950)**	**0.811 (0.917)**	**0.656 (0.882)**	**0.497 (0.847)**	**0.126 (0.742)**

What I could not talk about...

- Several open choices: h, ε, L, $\mathbf{y}$, $\mathbf{b}$
- They at all boil to down to sorting *in the limit*, as long as h convex, yet shape differentiability