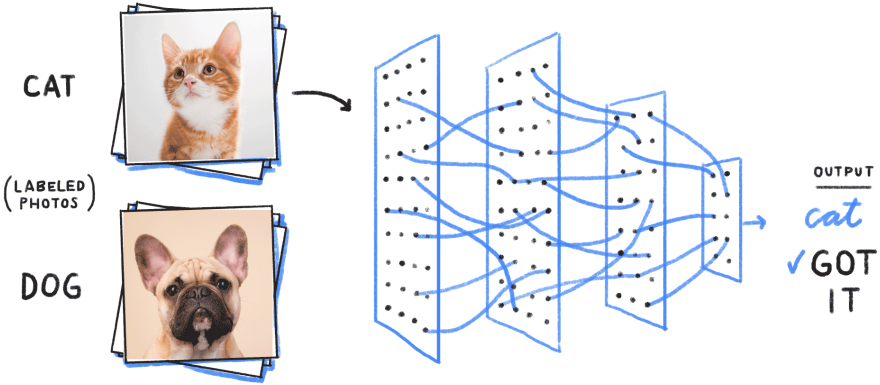
Learning from ranks, learning to rank

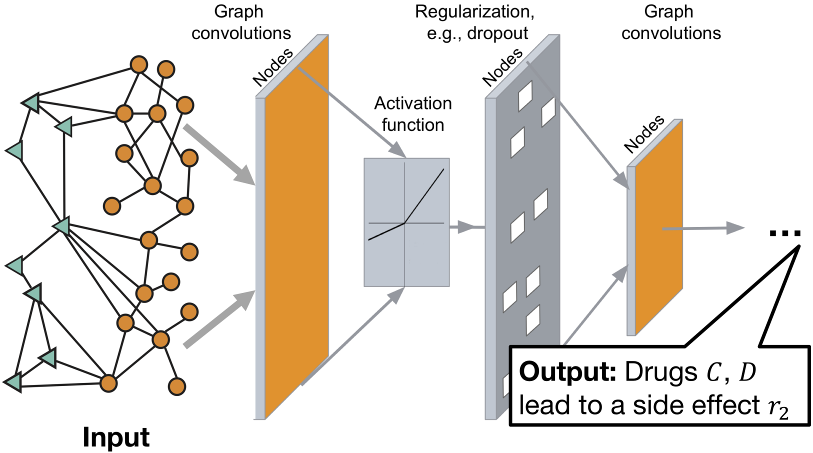
Jean-Philippe Vert

ML



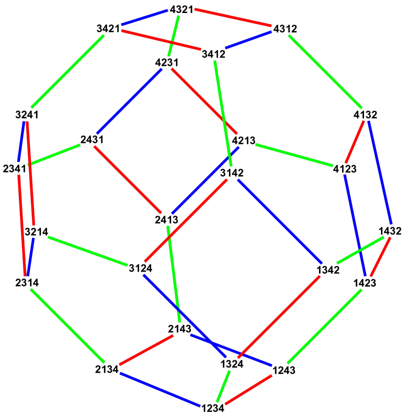
https://codeburst.io/machine-learning-243cc92247a1

Beyond images and strings



http://snap.stanford.edu/decagon

What if inputs or outputs are permutations?



- Permutation: a bijection

$$\sigma : [1, N] \to [1, N]$$

- $\sigma(i)$ = rank of item i
- Composition

$$(\sigma_1 \sigma_2)(i) = \sigma_1(\sigma_2(i))$$

- $\mathbb{S}_N$ the symmetric group
- $|\mathbb{S}_N| = N!$

Examples

- Rankings (as input or output)

- Discretization / normalization of continuous data

(histogram equalization, quantile normalization...)

Goals

1. Permutations as input / intermediate:

$$\sigma \in \mathbb{S}_N \mapsto f_\theta(\sigma) \in \mathbb{R}^p$$

   How to define / optimize $f_\theta : \mathbb{S}_N \to \mathbb{R}^p$?

   - SUQUAN (Le Morvan and Vert, 2017), Kendall (Jiao and Vert, 2015, 2017, 2018)

2. Permutations as intermediate / output:

$$x \in \mathbb{R}^N \mapsto \sigma(x) \in \mathbb{S}_N \mapsto f_\theta(\sigma(x)) \in \mathbb{R}^p$$

   How to differentiate the ranking operator $\sigma : \mathbb{R}^N \to \mathbb{S}_N$?

   - Sinkhorn ranking (Cuturi et al., 2019)

Permutations as inputs

- Assume your data are permutations and you want to learn

$$f : \mathbb{S}_N \to \mathbb{R}$$

- A solutions: embed $\mathbb{S}_N$ to a Euclidean (or Hilbert) space

$$\Phi : \mathbb{S}_N \to \mathbb{R}^p$$

  and learn a linear function:

$$f_\beta(\sigma) = \beta^\top \Phi(\sigma)$$

- The corresponding kernel is

$$K(\sigma_1, \sigma_2) = \Phi(\sigma_1)^\top \Phi(\sigma_2)$$

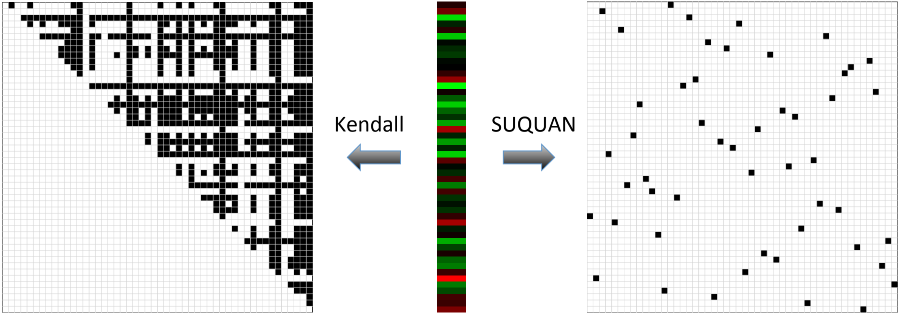
How to define the embedding $\Phi : \mathbb{S}_N \to \mathbb{R}^p$?

- Should encode interesting features
- Should lead to efficient algorithms

- Should be invariant to renaming of the items, i.e., the kernel should be right-invariant

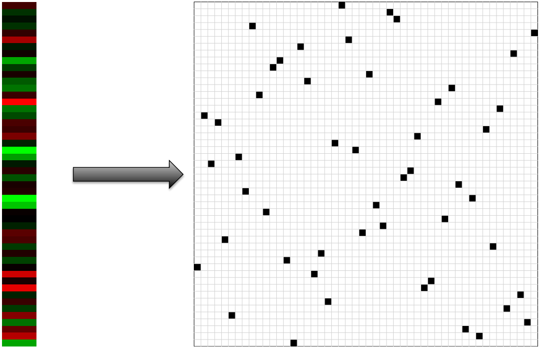
$$\forall \sigma_1, \sigma_2, \pi \in \mathbb{S}_N\,, \quad K(\sigma_1\pi, \sigma_2\pi) = K(\sigma_1, \sigma_2)$$

Some attempts



(Jiao and Vert, 2015, 2017, 2018; Le Morvan and Vert, 2017)

SUQUAN embedding (Le Morvan and Vert, 2017)



- Let $\Phi(\sigma) = \Pi_\sigma$ the permutation representation (Serres, 1977):

$$[\Pi_\sigma]_{ij} = \begin{cases} 1 & \text{if } \sigma(j) = i\,, \\ 0 & \text{otherwise.} \end{cases}$$

- Right invariant:

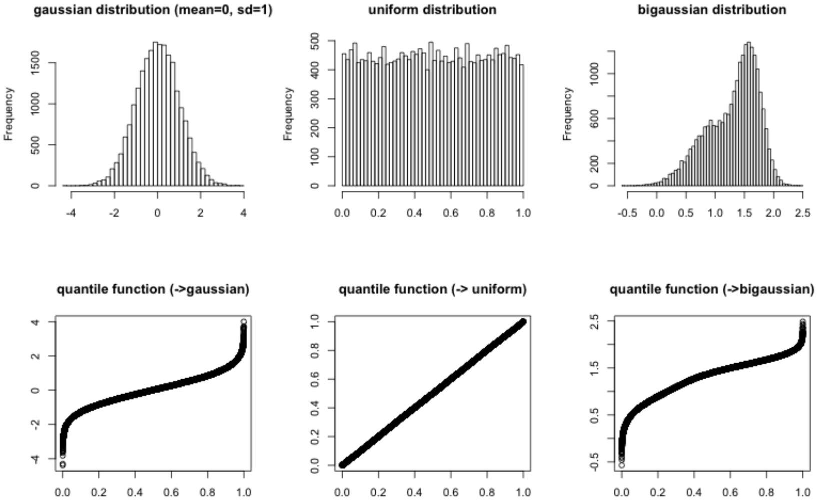
$$< \Phi(\sigma), \Phi(\sigma') >= \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'}^\top\right) = \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'}^{-1}\right) = \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'^{-1}}\right) = \mathrm{Tr}\left(\Pi_{\sigma\sigma'^{-1}}\right)$$

Link with quantile normalization (QN)

- Take $\sigma(x) = \mathtt{rank}(x)$ with $x \in \mathbb{R}^N$
- Fix a target quantile $f \in \mathbb{R}^n$
- "Keep the order of x, change the values to f"

$$[\Psi_f(x)]_i = f_{\sigma(x)(i)} \quad \Leftrightarrow \quad \Psi_f(x) = \Pi_{\sigma(x)} f$$

How to choose a "good" target distribution?



Supervised QN (SUQUAN)

Standard QN:

1. Fix f arbitrarily
2. QN all samples to get $\Psi_f(x_1), \ldots, \Psi_f(x_N)$
3. Learn a model on normalized data, e.g.:

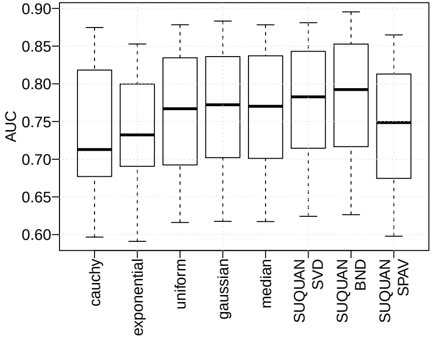
$$\min_{\theta} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_i \left(f_\theta(\Psi_f(x_i)) \right) \right\}$$

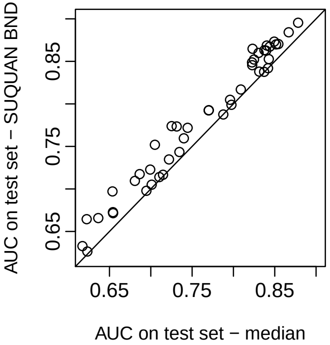
SUQUAN: jointly learn f and the model:

$$\min_{\theta, f} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_i \left(f_\theta(\Psi_f(x_i)) \right) \right\}$$

Experiments: CIFAR-10

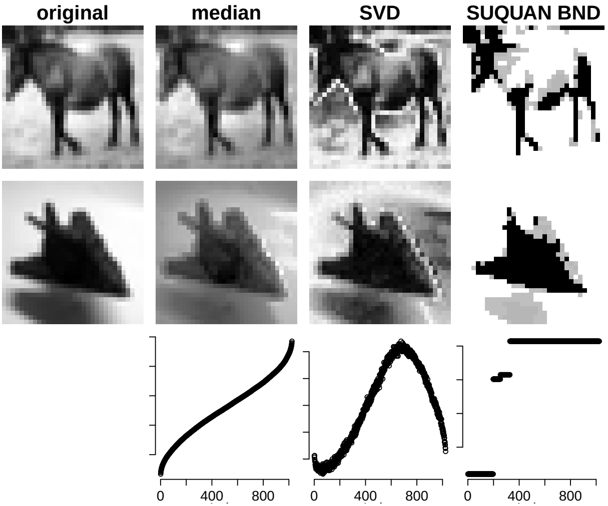
- Image classification into 10 classes (45 binary problems)
- $N = 5,000$ per class, $p = 1,024$ pixels
- Linear logistic regression on raw pixels



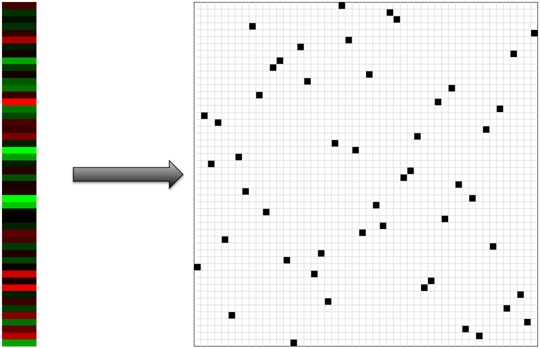


Experiments: CIFAR-10

- Example: horse vs. plane
- Different methods learn different quantile functions

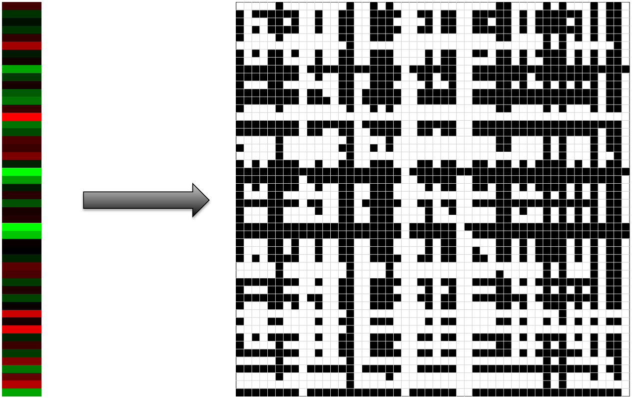


Limits of the SUQUAN embedding



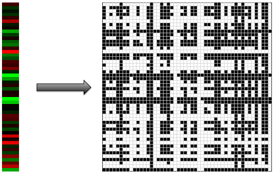
- Linear model on $\Phi(\sigma) = \Pi_\sigma \in \mathbb{R}^{N\times N}$
- Captures first-order information of the form "*i-th feature ranked at the j-th position*"
- What about higher-order information such as "*feature i larger than feature j*"?

The Kendall embedding (Jiao and Vert, 2015, 2017)



$$\Phi_{i,j}(\sigma) = \begin{cases} 1 & \text{if } \sigma(i) < \sigma(j)\,, \\ 0 & \text{otherwise.} \end{cases}$$

Geometry of the embedding



For any two permutations $\sigma, \sigma' \in \mathbb{S}_N$:

- Inner product

$$\Phi(\sigma)^\top \Phi(\sigma') = \sum_{1 \leq i \neq j \leq n} \mathbb{1}_{\sigma(i) < \sigma(j)} \mathbb{1}_{\sigma'(i) < \sigma'(j)} = n_c(\sigma, \sigma')$$

  n_c = number of concordant pairs

- Distance

$$\| \Phi(\sigma) - \Phi(\sigma') \|^2 = \sum_{1 \leq i,j \leq n} \left(\mathbb{1}_{\sigma(i) < \sigma(j)} - \mathbb{1}_{\sigma'(i) < \sigma'(j)} \right)^2 = 2 n_d(\sigma, \sigma')$$

  n_d = number of discordant pairs

Kendall and Mallows kernels

- The Kendall kernel is

$$K_\tau(\sigma, \sigma') = n_c(\sigma, \sigma')$$

- The Mallows kernel is

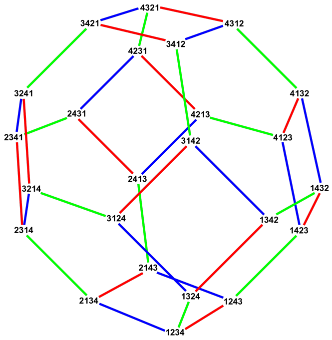
$$\forall \lambda \geq 0 \quad K_M^\lambda(\sigma, \sigma') = e^{-\lambda n_d(\sigma, \sigma')}$$

Theorem (Jiao and Vert, 2015, 2017)

The Kendall and Mallows kernels are positive definite right-invariant kernels and can be evaluated in $O(N \log N)$ time

Kernel trick useful with few samples in large dimensions

Remark



Cayley graph of $\mathbb{S}_4$

- Kondor and Barbarosa (2010) proposed the diffusion kernel on the Cayley graph of the symmetric group generated by adjacent transpositions.
- Computationally intensive ($O(N^{2N})$)
- Mallows kernel is written as

$$K_M^\lambda(\sigma, \sigma') = e^{-\lambda n_d(\sigma,\sigma')},$$

  where $n_d(\sigma, \sigma')$ is the shortest path distance on the Cayley graph.
- It can be computed in $O(N \log N)$
- Extension to weighted Kendall kernel (Jiao and Vert, 2018)

Remark

The SUQUAN and Kendall representations are two particular cases of the more general

Bochner's theorem

An embedding $\Phi : \mathbb{S}_N \to \mathbb{R}^p$ defines a right-invariant kernel $K(\sigma_1, \sigma_2) = \Phi(\sigma_1)^\top \Phi(\sigma_2)$ if and only there exists $\phi : \mathbb{S}_N \to \mathbb{R}$ such that

$$\forall \sigma_1, \sigma_2 \in \mathbb{S}_N \,, \quad K(\sigma_1, \sigma_2) = \phi(\sigma_2^{-1}\sigma_1)$$

and

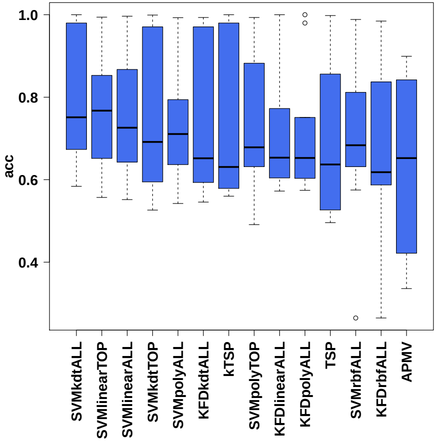
$$\forall \lambda \in \Lambda \,, \quad \hat{\phi}(\rho_\lambda) \succeq 0$$

where for any $f : \mathbb{S}_N \to \mathbb{R}$, the Fourier transform of f is

$$\forall \lambda \in \Lambda \,, \quad \hat{f}(\rho_\lambda) = \sum_{\sigma \in \mathbb{S}_N} f(\sigma) \rho_\lambda(\sigma)$$

with $\{\rho_\lambda \ : \ \lambda \in \Lambda\}$ the irreductible representations of the symmetric group.

Applications



Average performance on 10 microarray classification problems (Jiao and Vert, 2017).

Permutations as intermediate / output?

- Ranking operator:

$$\texttt{rank}(-15, 2.3, 20, -2) = (4, 2, 1, 3)$$

- Main problem:

$$x \in \mathbb{R}^N \mapsto \texttt{rank}(x) \in \mathbb{S}_N \quad \text{is not differentiable}$$

Permutations as intermediate / output?

- Ranking operator:

$$\texttt{rank}(-15, 2.3, 20, -2) = (4, 2, 1, 3)$$

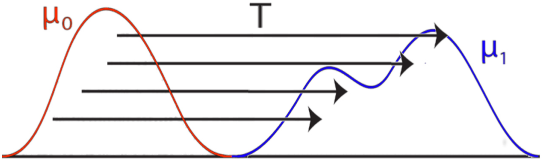
- Main problem:

$$x \in \mathbb{R}^N \mapsto \texttt{rank}(x) \in \mathbb{S}_N \quad \text{is not differentiable}$$

Differentiable Ranks and Sorting using Optimal Transport

Marco Cuturi **Olivier Teboul** **Jean-Philippe Vert**
Google Research, Brain Team
{cuturi,oliviert,jpvert}@google.com

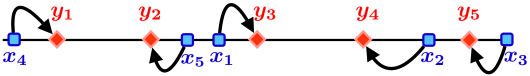
From optimal transport (OT) to `rank`



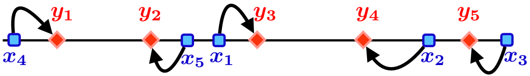
- For $\xi = \sum_{i=1}^{n} a_i \delta_{x_i}$ and $\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$ where $x_i, y_j \in \mathbb{R}$:

$$\mathrm{OT}_c(\xi, \nu) \overset{\text{def.}}{=} \min_{P \in U(\mathbf{a}\,\mathbf{b})} \langle P, C_{\mathbf{xy}} \rangle, \text{ where } U(\mathbf{a}, \mathbf{b}) \overset{\text{def.}}{=} \{P \in \mathbb{R}_+^{n \times m} | P\mathbf{1}_m = \mathbf{a}, P^T \mathbf{1}_n = \mathbf{b}\}$$

- For a cost $C(x_i, y_j)$ with $C \in C^2(\mathbb{R}^2)$ and $\partial^2 C / \partial x \partial y > 0$, solving OT is done in $O(n \ln n)$ with the `rank` function by sorting x and y
- If ν is ordered, then the solution P is the permutation matrix of ξ



Differentiable OT



$$P_\epsilon = \underset{P \in U(a,b)}{\operatorname{argmin}} < P, C > -\epsilon H(P)$$

Algorithm 1: Sinkhorn

Inputs: $\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{y}, \varepsilon, \ell$

$K \leftarrow e^{-C_{\mathbf{xy}}/\varepsilon}, \mathbf{u}_0 = \mathbf{1}_n$;

for $t \leftarrow 0$ **to** $\ell - 1$ **do**

$\quad \mathbf{v}_{t+1} \leftarrow \mathbf{b}/K^T \mathbf{u}_t$

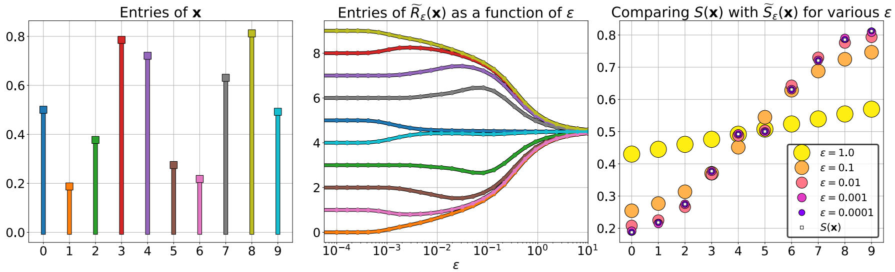
$\quad \mathbf{u}_{t+1} \leftarrow \mathbf{a}/K \mathbf{v}_{t+1}$

end

Result: $\mathbf{u}_\ell, K, \mathbf{v}_\ell$

- $P = \operatorname{diag}(u_\ell) K \operatorname{diag}(v_\ell)$ is the differentiable approximate permutation matrix of the input vector x
- Complexity $O(nm\ell)$, GPU-friendly

Soft ranks and sort sort



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

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

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

until $\max_s \Delta \left(\exp\left(C_{\mathbf{x}_s\mathbf{y}}^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right) \mathbf{1}_n, \mathbf{b}\right) < \eta;$

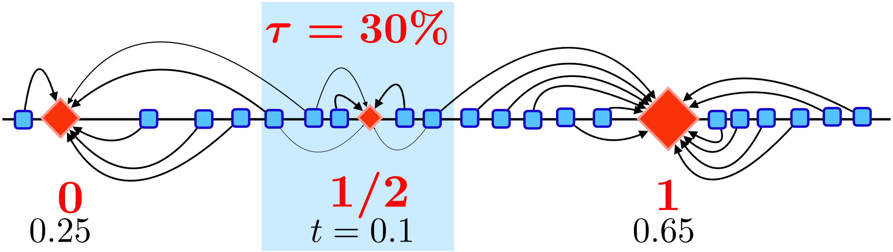
$\forall s, \widetilde{R}_\varepsilon(\mathbf{x}_s) \leftarrow \mathbf{a}_s^{-1} \circ \exp\left(C_{\mathbf{x}_s\mathbf{y}} - \boldsymbol{\alpha}_s \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}_s^T\right) \overline{\mathbf{b}},$

$\forall s, \widetilde{S}_\varepsilon(\mathbf{x}_s) \leftarrow \mathbf{b}_s^{-1} \circ \exp\left(C_{\mathbf{x}_s\mathbf{y}}^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right) \mathbf{x}_s.$

Result: $\left(\widetilde{R}_\varepsilon(\mathbf{x}_s), \widetilde{S}_\varepsilon(\mathbf{x}_s)\right)_s.$

Soft quantization and soft quantiles

- Take $m = 3$, $\mathbf{y} = (0, 0.5, 1)$, $\mathbf{b} = (\tau - t/2, t, \tau + t/2)$
- Overall complexity $O(n\ell)$



https://github.com/google-research/google-research/tree/master/soft_sort

Application: soft top-k loss

$$\text{S-top-}k\text{-loss}(f_\theta(\omega_0), l_0) = J_k\left(1 - \left(\widetilde{F}^\ell\left(\frac{\mathbf{1}_L}{L}, f_\theta(\omega); \frac{\mathbf{1}_m}{m}, \mathbf{y}\right)\right)_{l_0}\right)$$

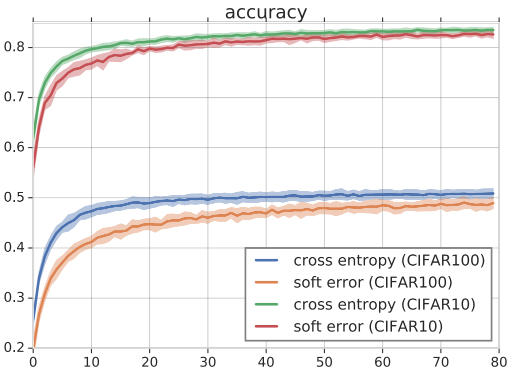
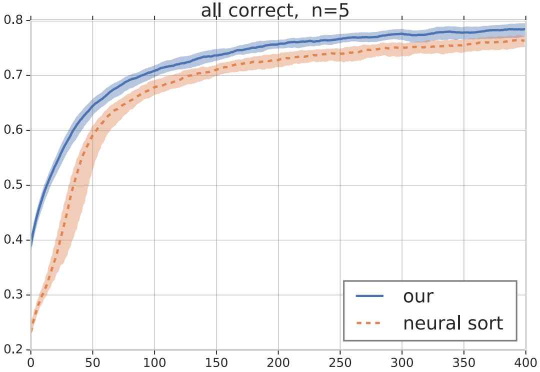


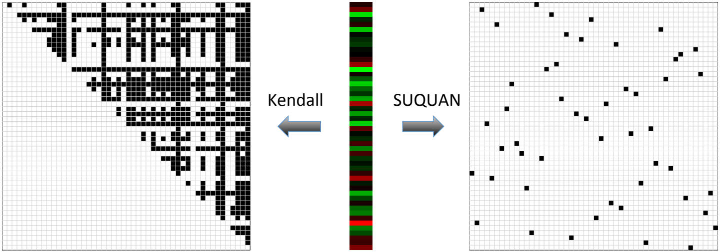
Figure 4: Error bars for test accuracy curves on CIFAR-100 and CIFAR-10 using the same network (averages over 12 runs).

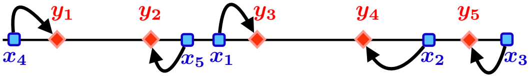
Application: learning to sort

- Task: Sort 5 numbers between 0000 and 9999 (concatenation of MNIST digits) (Grover et al, 2019)



Conclusion





- Machine learning beyond vectors, strings and graphs
- Different embeddings of the symmetric group
- Differentiable sorting and ranking
- Scalability? Robustness to adversarial attacks? Theoretical properties?

MERCI!

References

M. Cuturi, O. Teboul, and J.-P. Vert. Differentiable sorting using optimal transport: the Sinkhorn CDF and quantile operator. In *Adv. Neural. Inform. Process Syst. 31*, 2019.

Y. Jiao and J.-P. Vert. The Kendall and Mallows kernels for permutations. In *Proceedings of The 32nd International Conference on Machine Learning*, volume 37 of *JMLR:W&CP*, pages 1935–1944, 2015. URL `http://jmlr.org/proceedings/papers/v37/jiao15.html`.

Y. Jiao and J.-P. Vert. The Kendall and Mallows kernels for permutations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. doi: 10.1109/TPAMI.2017.2719680. URL `http://dx.doi.org/10.1109/TPAMI.2017.2719680`.

Y. Jiao and J.-P. Vert. The weighted kendall and high-order kernels for permutations. Technical Report 1802.08526, arXiv, 2018.

M. Le Morvan and J.-P. Vert. Supervised quantile normalisation. Technical Report 1706.00244, arXiv, 2017.

J.-P. Serres. *Linear Representations of Finite Groups*. Graduate Texts in Mathematics. Springer-Verlag New York, 1977. doi: 10.1007/978-1-4684-9458-7. URL `http://dx.doi.org/10.1007/978-1-4684-9458-7`.

Harmonic analysis on $\mathbb{S}_N$

- A representation of $\mathbb{S}_N$ is a matrix-valued function $\rho : \mathbb{S}_N \to \mathbb{C}^{d_\rho \times d_\rho}$ such that
$$\forall \sigma_1, \sigma_2 \in \mathbb{S}_N\,, \quad \rho(\sigma_1\sigma_2) = \rho(\sigma_1)\rho(\sigma_2)$$
- A representation is irreductible (irrep) if it is not equivalent to the direct sum of two other representations
- $\mathbb{S}_N$ has a finite number of irreps $\{\rho_\lambda \;:\; \lambda \in \Lambda\}$ where $\Lambda = \{\lambda \vdash N\}$[1] is the set of partitions of N
- For any $f : \mathbb{S}_N \to \mathbb{R}$, the Fourier transform of f is
$$\forall \lambda \in \Lambda\,, \quad \hat{f}(\rho_\lambda) = \sum_{\sigma \in \mathbb{S}_N} f(\sigma)\rho_\lambda(\sigma)$$

[1] $\lambda \vdash N$ iff $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 \geq \ldots \geq \lambda_r$ and $\sum_{i=1}^{r} \lambda_i = N$

Right-invariant kernels

Bochner's theorem

An embedding $\Phi : \mathbb{S}_N \to \mathbb{R}^p$ defines a right-invariant kernel $K(\sigma_1, \sigma_2) = \Phi(\sigma_1)^\top \Phi(\sigma_2)$ if and only there exists $\phi : \mathbb{S}_N \to \mathbb{R}$ such that

$$\forall \sigma_1, \sigma_2 \in \mathbb{S}_N \,, \quad K(\sigma_1, \sigma_2) = \phi(\sigma_2^{-1} \sigma_1)$$

and

$$\forall \lambda \in \Lambda \,, \quad \hat{\phi}(\rho_\lambda) \succeq 0$$