Differentiable ranking and sorting

Jean-Philippe Vert

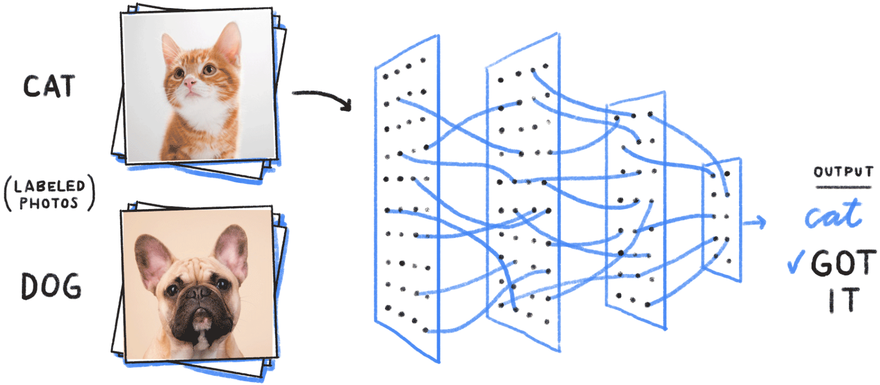
Google AI

MINES ParisTech

Outline

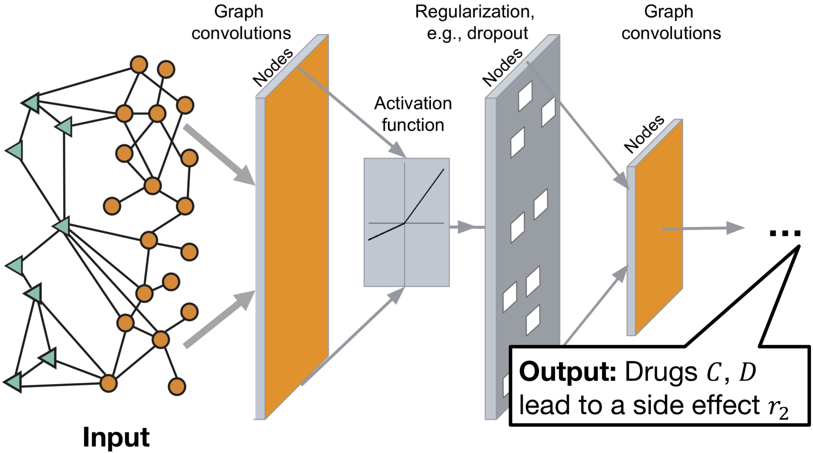
Outline

Differentiable programming



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

Going beyond vectors (strings, graphs...)



http://snap.stanford.edu/decagon

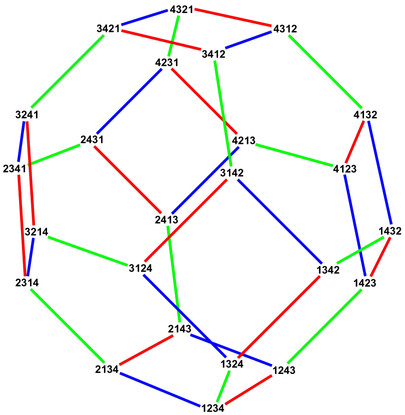
What about rankings / permutations?

- Some data are permutations (input, output)

- Some operations may involve ranking

(histogram equalization, quantile normalization...)

More formally



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

Goal

$$\begin{tikzcd} \mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p \end{tikzcd}$$

1. Embed
   - To define / optimize $f_\theta(\sigma) = g_\theta(\text{embed}(\sigma))$ for $\sigma \in \mathbb{S}_N$
   - E.g., σ given as input, or output
2. Differentiate
   - To define / optimize $h_\theta(x) = f_\theta(\text{argsort}(x))$ for $x \in \mathbb{R}^n$
   - E.g., normalization layer or rank-based loss

Outline

Argmax

$$\operatorname{argmax} \begin{pmatrix} 2.1 \\ -0.4 \\ 5.7 \end{pmatrix} = 3$$

is not differentiable because:

- As a function $\mathbb{R}^n \to [1, n]$, the output space is not continuous
- It is piecewise constant (ie, gradient=0 almost everywhere even if the output space was continuous, eg, $(1, n) \subset \mathbb{R}$)

Softmax

$$\text{softmax}_1 \begin{pmatrix} 2.1 \\ -0.4 \\ 5.7 \end{pmatrix} = \begin{pmatrix} 0.027 \\ 0.002 \\ 0.971 \end{pmatrix}$$

is a differentiable function $\mathbb{R}^n \to \mathbb{R}^n$, where

$$\text{softmax}_\epsilon(x)_i = \frac{e^{x_i/\epsilon}}{\sum_{j=1}^n e^{x_j/\epsilon}}$$

From Softmax to Argmax

$$\lim_{\epsilon \to 0} \text{softmax}_\epsilon \begin{pmatrix} 2.1 \\ -0.4 \\ 5.7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \Psi\,(3)\ ,$$

where $\Psi : [1, n] \to \mathbb{R}^n$ is the one-hot encoding. More generally,

$$\forall x \in \mathbb{R}^n, \quad \lim_{\epsilon \to 0} \text{softmax}_\epsilon(x) = \Psi(\operatorname{argmax}(x))$$

$$\begin{tikzcd}
\mathbb{R}^n \arrow[r, "\texttt{argmax}"] \arrow[rr, bend right, "\texttt{softmax}"'] & {[1, n]} \arrow[r, "\texttt{one-hot}"] & \mathbb{R}^n
\end{tikzcd}$$

From Argmax to Softmax (1): embedding

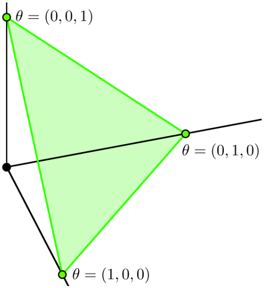
$$\mathbb{R}^n \xrightarrow{\texttt{argmax}} [1, n] \xrightarrow{\texttt{one-hot}} \mathbb{R}^n$$

Let the simplex

$$\Delta_{n-1} = \mathrm{conv}\left(\{\Psi(y) \,:\, y \in [1, n]\}\right)$$

Then we have a variational characterization (*exercice*):

$$\Psi(\mathrm{argmax}\,(x)) = \underset{z \in \Delta_{n-1}}{\mathrm{argmax}}\,(x^\top z)$$

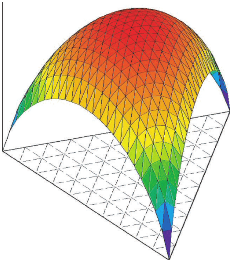


From Argmax to Softmax (2a): regularization

$$\begin{tikzcd} \mathbb{R}^n \arrow[r, "\texttt{argmax}"] \arrow[rr, bend right, "\texttt{softmax}"'] & {[1, n]} \arrow[r, "\texttt{one-hot}"] & \mathbb{R}^n \end{tikzcd}$$

Let the entropy $H(z) = -\sum_{i=1}^{n} z_i \ln(z_i)$ for $z_i \in \Delta_{n-1}$.
Then we have (*exercice*):

$$\mathrm{softmax}_{\epsilon}(x) = \underset{z \in \Delta_{n-1}}{\mathrm{argmax}} \left[x^{\top} z + \epsilon H(z) \right]$$

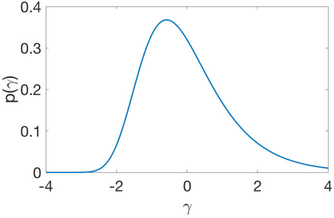


From Argmax to Softmax (2b): perturbation

$$\begin{tikzcd} \mathbb{R}^n \arrow[r, "\texttt{argmax}"] \arrow[rr, bend right, "\texttt{softmax}"'] & {[1,n]} \arrow[r, "\texttt{one-hot}"] & \mathbb{R}^n \end{tikzcd}$$

Let $G = (G_1, \ldots, G_n)$ be i.i.d. Gumbel(0,1) random variables[1] Then we have (*exercice*):

$$\mathrm{softmax}_\epsilon(x) = E \operatorname*{argmax}_{z \in \Delta_{n-1}} \left[(x + \epsilon G)^\top z \right]$$



[1] $G_i = -\ln(-ln(U_i))$ where $U_i \sim \mathrm{unif}(0, 1)$

From Argmax to Softmax: summary

$$\begin{tikzcd}
\mathbb{R}^n \arrow[r, "\texttt{argmax}"] \arrow[rr, bend right, "\texttt{softmax}"'] & {[1, n]} \arrow[r, "\texttt{one-hot}"] & \mathbb{R}^n
\end{tikzcd}$$

1. Embed, such that

$$\Psi(\operatorname{argmax}(x)) = \operatorname*{argmax}_{z \in \Delta_{n-1}} (x^\top z)$$

2. Regularize or perturb:

$$\text{softmax}_\epsilon(x) = \operatorname*{argmax}_{z \in \Delta_{n-1}} \left[x^\top z + \epsilon H(z) \right] = E \operatorname*{argmax}_{z \in \Delta_{n-1}} \left[x^\top (z + \epsilon G) \right]$$

Both lead to efficient (stochastic) Jacobian estimates.
Can we generalize this to other discrete operations, such as ranking?

Outline

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

$$\begin{tikzcd}
\mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p
\end{tikzcd}$$

- Should encode interesting features
- Should lead to efficient algorithms

- Geometry should not change by arbitrary renaming of items, i.e.,

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

- Equivalently, the kernel should be translation-invariant

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

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

$$\begin{tikzcd} \mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p \end{tikzcd}$$

- Should encode interesting features
- Should lead to efficient algorithms

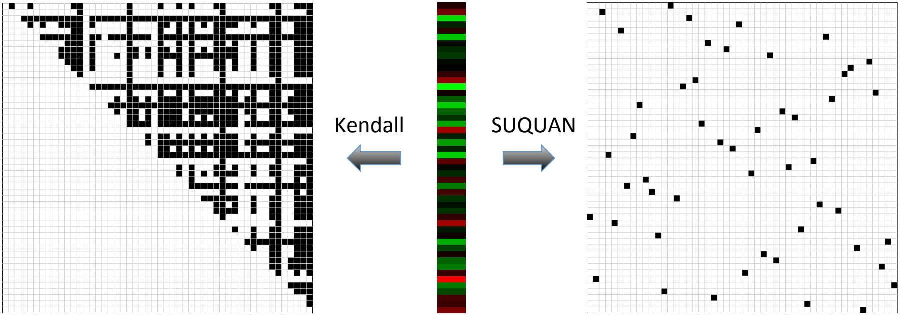
- Geometry should not change by arbitrary renaming of items, i.e.,

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

- Equivalently, the kernel should be translation-invariant

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

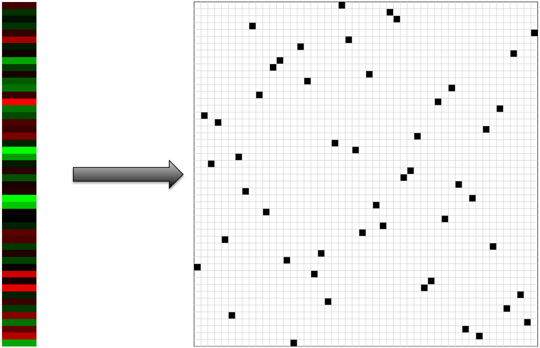
Some attempts



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

Outline

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:

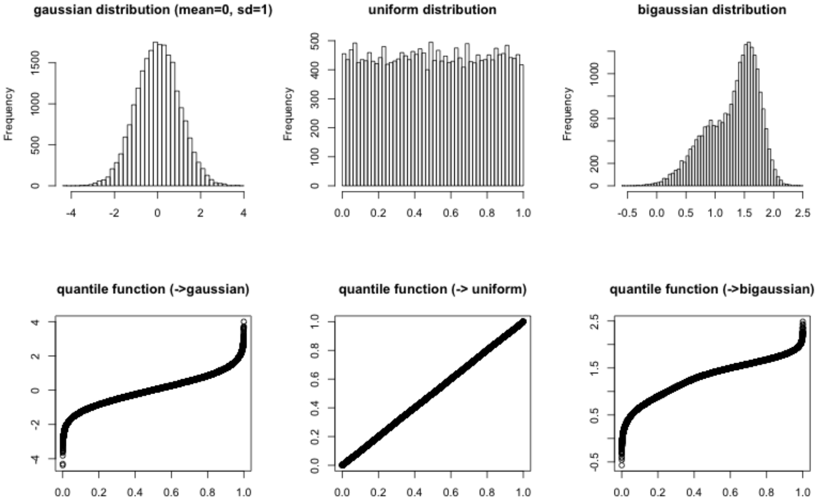
$$< \Phi(\sigma), \Phi(\sigma') >= \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'}^\top\right) = \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'}^{-1}\right) = \mathrm{Tr}\left(\Pi_\sigma \Pi_{\sigma'^{-1}}\right) = \mathrm{Tr}\left(\Pi_{\sigma\sigma'^{-1}}\right)$$

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"

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

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.:

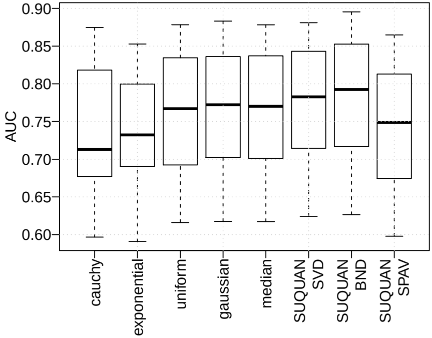
$$\min_{\theta} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_i \left(f_\theta(\Psi_f(x_i)) \right) \right\}$$

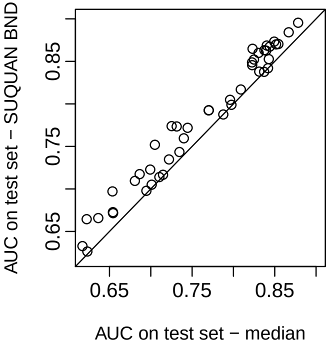
SUQUAN: jointly learn f and the model:

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

Experiments: CIFAR-10

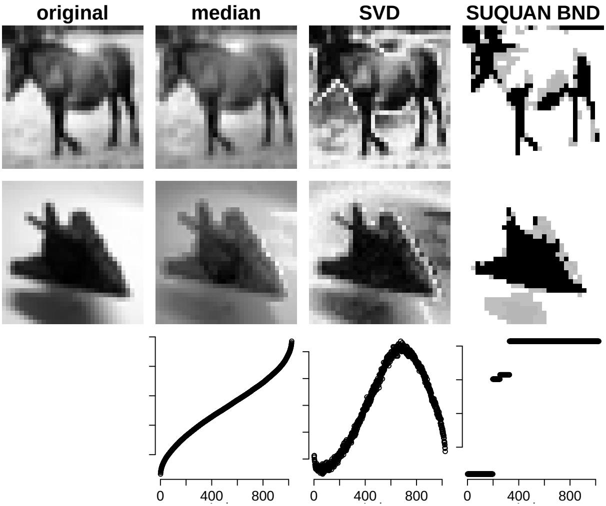
- Image classification into 10 classes (45 binary problems)
- $N = 5,000$ per class, $p = 1,024$ pixels
- Linear logistic regression on raw pixels



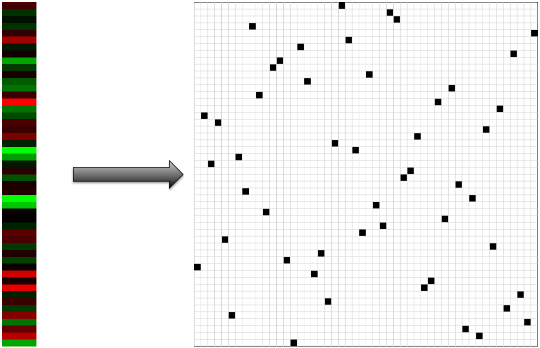


Experiments: CIFAR-10

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



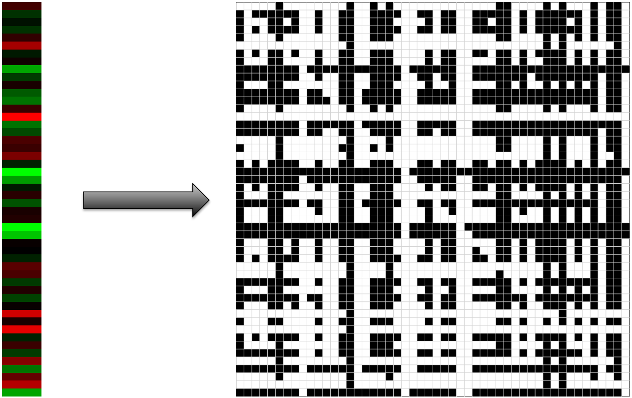
Limits of the SUQUAN embedding



- 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*"?

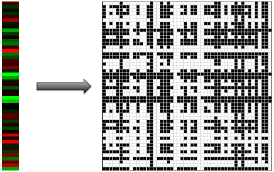
Outline

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

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

  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

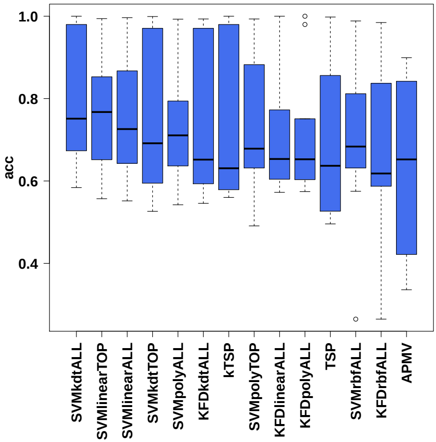
$$\forall \lambda \geq 0 \quad K_M^\lambda(\sigma, \sigma') = e^{-\lambda n_d(\sigma, \sigma')}$$

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

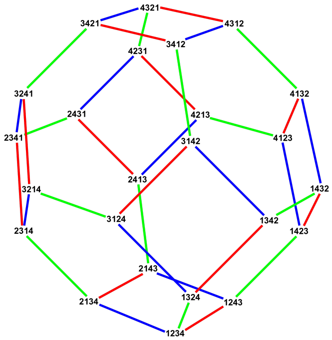
Kernel trick useful with few samples in large dimensions

Applications



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

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.

Outline

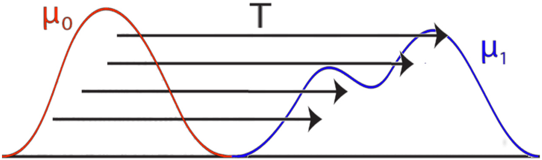
The problem

$$
\begin{tikzcd}
\mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p
\end{tikzcd}
$$

- $x \in \mathbb{R}^N \mapsto \texttt{argsort}(x) \in \mathbb{S}_N$ is piecewise constant
- Derivative a.e. equal to zero
- Same for $x \mapsto \Phi(\texttt{argsort}(x))$, for any embedding Φ

How to create a differentiable approximation to $\Phi(\texttt{argsort}(x))$?

Optimal transport (OT)

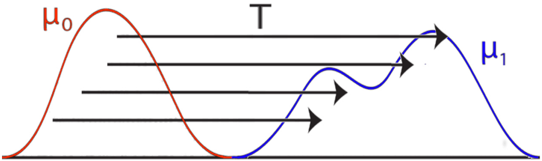


Given a cost matrix $C \in \mathbb{R}^{n \times n}$ (where C_{ij} is the cost of moving the i-th point to the j-th location), OT solves

$$\min_{P \in B_n} \langle P, C \rangle$$

where $B_n = \left\{ P \in \mathbb{R}_+^{n \times n} \,|\, P\mathbf{1}_n = P^\top \mathbf{1}_n = \mathbf{1}_n \right\}$ is the Birkhoff polytope.

Variational formulation of SUQUAN embedding



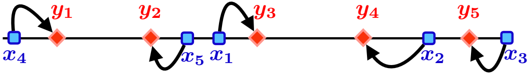
Lemma

Take

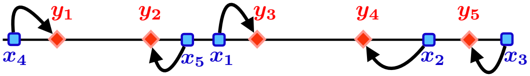
- $y \in \mathbb{R}^n$ *with* $y_1 < \ldots < y_n$,
- $h \in C^2(\mathbb{R}^2)$ *and* $\partial^2 h / \partial x \partial y > 0$ *(eg,* $h(a,b) = (b-a)^2$*).*

For any $x \in \mathbb{R}^n$*, let* $C(x) \in \mathbb{R}^{n \times n}$ *given by* $C(x)_{ij} = h(y_i, x_j)$*. Then*

$$\operatorname*{argmin}_{P \in B_n} \langle P, C(x) \rangle = \Pi_{\sigma(x)} \,.$$



Entropic regularization (Cuturi et al., 2019)



$$P_\epsilon(x) = \underset{P \in B_n}{\operatorname{argmin}} \langle P, C(x) \rangle - \epsilon H(P)$$

Algorithm 1: Sinkhorn

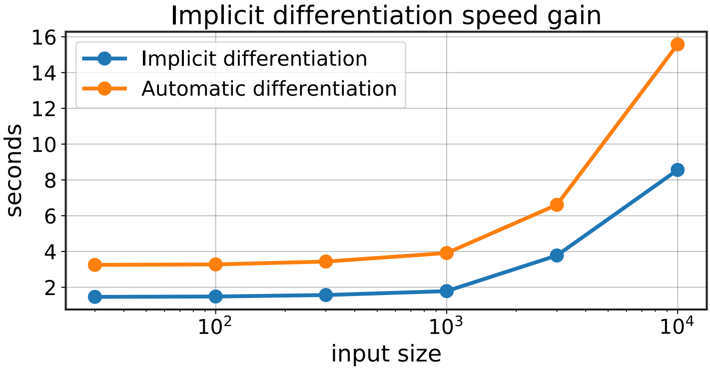
Inputs: $\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{y}, \varepsilon, \ell$
$K \leftarrow e^{-C_{\mathbf{xy}}/\varepsilon}, \mathbf{u}_0 = \mathbf{1}_n;$
for $t \leftarrow 0$ **to** $\ell - 1$ **do**
  $\mathbf{v}_{t+1} \leftarrow \mathbf{b}/K^T \mathbf{u}_t$
  $\mathbf{u}_{t+1} \leftarrow \mathbf{a}/K \mathbf{v}_{t+1}$
end
Result: $\mathbf{u}_\ell, K, \mathbf{v}_\ell$

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

Derivatives

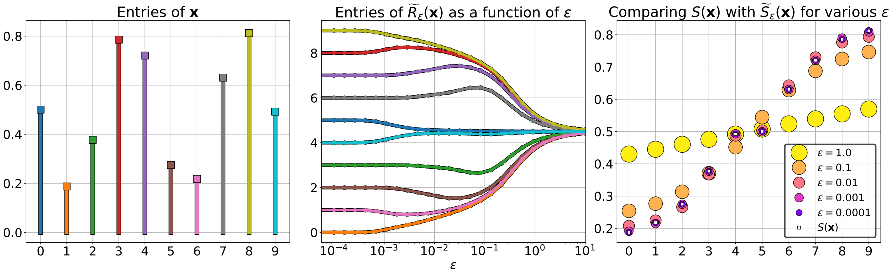
$\nabla P_\epsilon(x)$ can be computed by

- Automatic differentiation of Sinkhorn iterations (Cuturi et al., 2019)
- Implicit differentiation (Cuturi et al., 2020)



Differentiable `sort`, `argsort` and `rank`

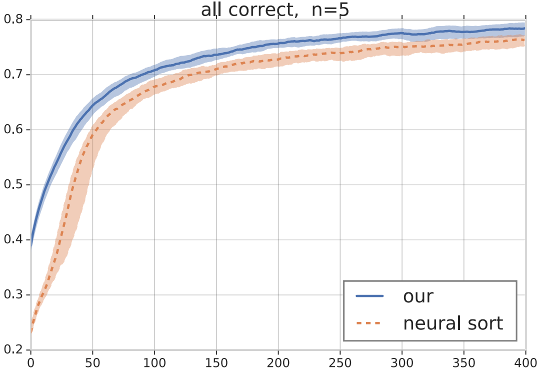
$$S_\epsilon(x) = P_\epsilon(x)x \qquad R_\epsilon(x) = P_\epsilon(x)^\top(1, 2, \ldots, n)^\top$$



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

Application: learning to rank

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



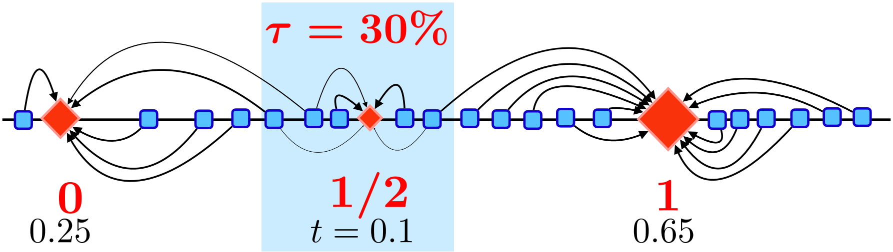
Outline

1. Motivation
2. Warm-up: from argmax to softmax
3. Embed
   - SUQUAN embedding
   - Kendall embedding
4. Differentiate
5. Extensions
   - Differentiable quantiles
   - Matrix factorization with quantile normalization
   - Smoothing by regularization
   - Smoothing by perturbation
   - Extensions to other discrete problems
6. Conclusion

Outline

Soft quantization and soft quantiles

- Take $C(x) \in \mathbb{R}^{m\times n}$ with $m < n$ and
  $B_n = \left\{P \in \mathbb{R}_+^{m\times n} \,|\, P\mathbf{1}_n = \mathbf{b}\,, P^\top \mathbf{1}_m = \mathbf{1}_n\right\}$
- E.g., $m = 3$, $\mathbf{y} = (0, 0.5, 1)$, $\mathbf{b} = (\tau - t/2, t, \tau + t/2)$
- Overall complexity $O(nm\ell)$



Application: soft top-k loss

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

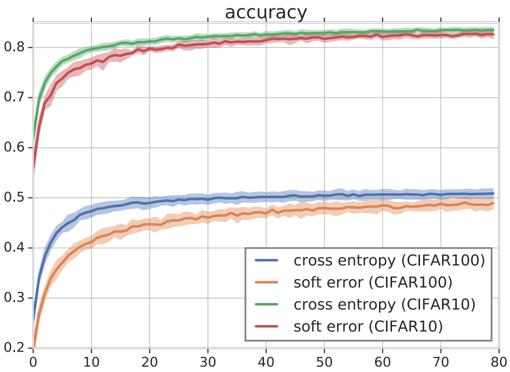
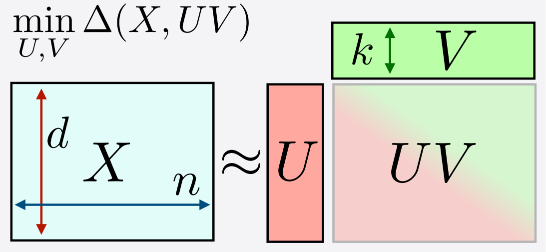


Figure 4: Error bars for test accuracy curves on CIFAR-100 and CIFAR-10 using the same network (averages over 12 runs).

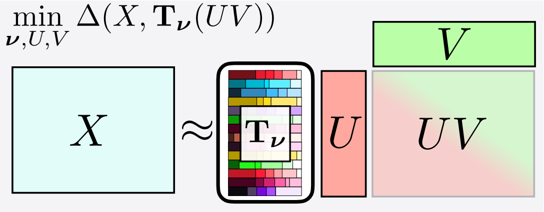
Outline

Nonnegative Matrix Factorization (NMF)

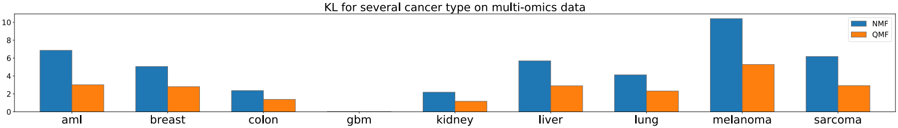


- Useful to decompose a signal as a superposition of basic elements
  - e.g., images, text, genomics...
- But one usually pre-process X so that it resembles a low-rank matrix
  - e.g., tf-idf, log-transform, quantile normalization etc...
- Can we jointly learn U, V and the pre-processing of X?

NMF with quantile normalization (Cuturi et al., 2020)



- T_ν is a *soft-quantile normalization* operator, applied row-wise, differentiable w.r.t. the input row and the target quantiles (generalization of SUQUAN)
- T_ν is optimized jointly with the low-rank matrix *UV*
- Application: multiomics data integration for cancer stratification



Outline

What we have done

$$\begin{tikzcd} \mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p \end{tikzcd}$$

1. Variational formulation through OT:

$$\Phi(\texttt{argsort}(x)) = \arg\min_{z \in B_n} < z, C(x) >$$

   with $\Phi : \mathbb{S}_N \to \mathbb{R}^{n \times n}$ the SUQUAN embedding

2. Smooth by entropy regularization

$$P_\epsilon(x) = \arg\min_{z \in B_n} [< z, C(x) > -\epsilon H(z)]$$

More general differentiable `argsort`

$$\begin{tikzcd} \mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p \end{tikzcd}$$

1. Variational formulation: For a given embedding $\Phi : \mathbb{S}_N \to \mathbb{R}^p$, find $\mathcal{Z} \subset \mathbb{R}^p$ and $\Psi : \mathbb{R}^N \to \mathbb{R}^p$ such that

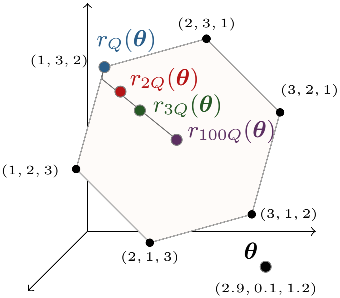
$$\Phi(\texttt{argsort}(x)) = \arg\min_{z \in \mathcal{Z}} < z, \Psi(x) >$$

2. Smooth by regularization

$$h_\epsilon(x) = \arg\min_{z \in \mathcal{Z}} [< z, \Psi(x) > + \epsilon\Omega(z)]$$

with $\Omega : \mathcal{Z} \to \mathbb{R}$ a regularization that makes h_ϵ differentiable

Example: FastSoftSort (Blondel et al., 2020)



- Embed to the permutahedron ($\mathcal{P}_N = B_N \times (1, 2, \ldots, N)^\top$)
- Ranking: $h(x) = \arg\min_{z \in \mathcal{P}_N} < x, z >$
- Regularization by negative entropy or Euclidean norm
- Fast $O(n \log(n))$ algorithm using isotonic regression to project onto the permutahedron and compute $h_\epsilon(x)$ and $\nabla h_\epsilon(x)$

`https://github.com/google-research/fast-soft-sort`

Outline

Reminder: the "Gumbel trick" for soft-max

- For $x \in \mathbb{R}^N$, the one-hot encoded argmax of x is

$$\arg\min_{z \in \Delta^{N-1}} < z, -x >$$

- The soft-max of x is

$$\mathrm{softmax}(x) = e^x / (\sum_i e^{x_i})$$

- It is obtained from the argmax by entropic regularization

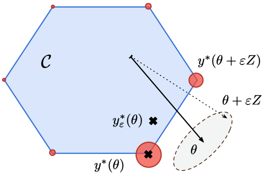
$$\mathrm{softmax}(x) = \arg\min_{z \in \Delta^{N-1}} [< z, -x > -H(z)]$$

- It is also equal to (Gumbel, 1954):

$$\mathrm{softmax}(x) = E\left[\arg\min_{z \in \Delta^{N-1}} < z, -(x + U) >\right]$$

where U is a random variable following the Gumbel distribution.

Smoothing by perturbation (Berthet et al., 2020)

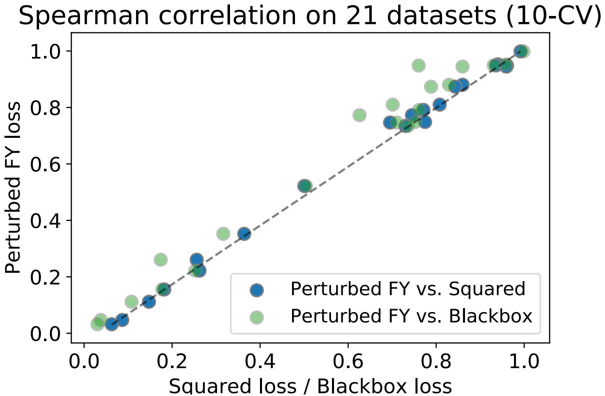


$$h(x) = \arg\min_{z \in \mathcal{Z}} < z, \Psi(x) >$$

$$h_\epsilon(x) = E\left[\arg\min_{z \in \mathcal{Z}} < z, \Psi(x + \epsilon U) >\right]$$

- Generalization of the “Gumbel trick” for soft-max (Gumbel, 1954)
- h_ϵ is differentiable if the density of U is smooth (e.g. normal)
- Stochastic gradient of h_ϵ can be computed efficiently
- Fast gradient for Fenchel-Young losses (Blondel et al., 2019)
- Sometimes equivalent to smoothing by regularization

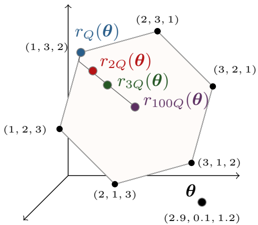
Experiments

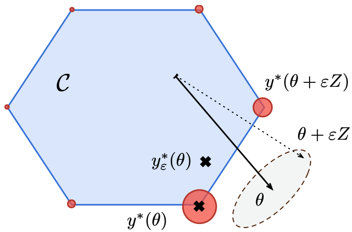


- 21 datasets (simulations and biology) of ranking prediction
- Compare perturbed FY loss on the permutahedron, with squared loss and Blackbox loss of Vlastelica et al. (2020)

Outline

General setting





Given a non-smooth (discrete) mapping $h : \mathbb{R}^n \to \mathbb{R}^p$ with a variational form:

$$h(x) = \left[\arg\min_{z \in \mathcal{Z}} < z, \Psi(x) >\right]$$

create differentiable versions:

$$h_\epsilon^{\text{Regularization}}(x) = \arg\min_{z \in \mathcal{Z}} [< z, \Psi(x) > -\epsilon\Omega(z)]$$

$$h_\epsilon^{\text{Perturbation}}(x) = E\left[\arg\min_{z \in \mathcal{Z}} < z, \Psi(x + \epsilon U) >\right]$$

Example: shortest path (Berthet et al., 2020)

- Take a graph $G = (V, E)$, positive edge costs $c \in \mathbb{R}_+^E$, polytope

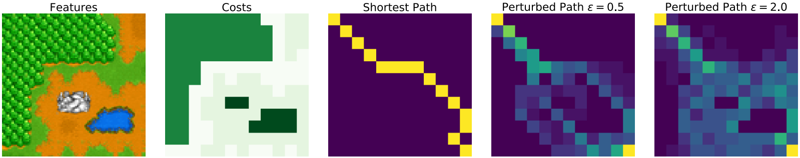
$$\mathcal{C} = \left\{ y \in \mathbb{R}_+^E \;:\; \forall i \in V \,, (\mathbf{1}_{\to i} - \mathbf{1}_{i \to})^\top y = \delta_{i=s} - \delta_{i=t} \right\}$$

- Shortest path (computable in $O(|E|)$ by DP) solves

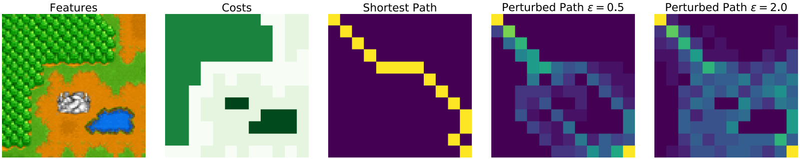
$$y^* = \arg\min_{y \in \mathcal{C}} < c, y >$$

- Differentiable shortest path (w.r.t c):

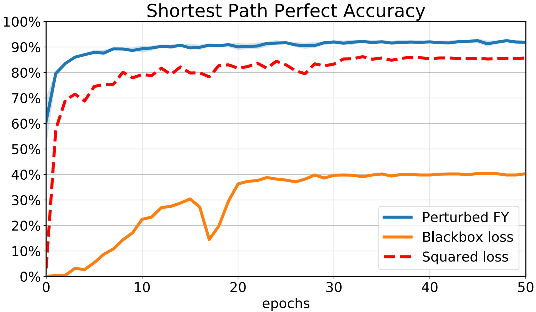
$$y^*_\epsilon = E \left[\arg\min_{y \in \mathcal{C}} < c + \epsilon U, y > \right]$$



(Toy) application



- Given 10k images of Warcraft terrains with 12x12 patches, and a shortest-path from upper-left to lower-right..
- Learn the cost of a patch (with a ResNet18 deep CNN)



Outline

Conclusion

$$\begin{tikzcd} \mathbb{R}^N \arrow[r, "\texttt{argsort}"] \arrow[rr, bend right, "\texttt{differentiate}"'] & \mathbb{S}_N \arrow[r, "\texttt{embed}"] & \mathbb{R}^p \end{tikzcd}$$

- Machine learning beyond vectors, strings and graphs
- Different embeddings of the symmetric group
- Differentiable sorting and ranking through regularization and perturbation
- Can be generalized to other discrete operations

THANK YOU!

References

Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, and F. Bach. Learning with differentiable perturbed optimizers. Technical Report 2002.08676, arXiv, 2020.

M. Blondel, A. F. T. Martins, and V. Niculae. Learning with Fenchel-Young losses. Technical Report 1901.02324, arXiv, 2019.

M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

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.

M. Cuturi, O. Teboul, J. Niles-Weed, and J.-P. Vert. Supervised quantile normalization for low-rank matrix approximation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

E. J. Gumbel. *Statistical theory of extreme values and some practical applications;: A series of lectures*. Number 33. US Govt. Print. Office, 1954.

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.

References (cont.)

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.

M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

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 $\Lambda = \{\lambda \vdash N\}$[2] is the set of partitions of N
- For any $f : \mathbb{S}_N \to \mathbb{R}$, the Fourier transform of f is

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

[2] $\lambda \vdash N$ 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

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