Differentiable ranking and sorting

Jean-Philippe Vert
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
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
Differentiable programming

https://codeburst.io/machine-learning-243cc92247a1
Going beyond vectors (strings, graphs...)

Output: Drugs $C, D$ lead to a side effect $r_2$

http://snap.stanford.edu/decagon
What about rankings / permutations?

- Some data are permutations (input, output)
  
  ![Bottles](image1.png)

- Some operations may involve ranking
  
  ![Swimmers](image2.png)

(histogram equalization, quantile normalization...
More formally

- Permutation: a bijection
  \[ \sigma : [1, N] \rightarrow [1, N] \]
- \( \sigma(i) = \) rank of item \( i \)
- Composition
  \[ (\sigma_1 \sigma_2)(i) = \sigma_1(\sigma_2(i)) \]
- \( S_N \) the symmetric group
- \( |S_N| = N! \)
1. **Embed**
   - To define / optimize $f_\theta(\sigma) = g_\theta(\text{embed}(\sigma))$ for $\sigma \in S_N$
   - E.g., $\sigma$ 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
Motivation

Warm-up: from argmax to softmax

Embed
- SUQUAN embedding
- Kendall embedding

Differentiate

Extensions
- Differentiable quantiles
- Matrix factorization with quantile normalization
- Smoothing by regularization
- Smoothing by perturbation
- Extensions to other discrete problems

Conclusion
Argmax

\[
\text{argmax } \begin{pmatrix}
2.1 \\
-0.4 \\
5.7
\end{pmatrix} = 3
\]

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 \( \mathbb{R}^n \rightarrow \mathbb{R}^n \), 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,

\[
\forall x \in \mathbb{R}^n, \lim_{\epsilon \to 0} \text{softmax}_\epsilon(x) = \Psi(\text{argmax}(x))
\]
From Argmax to Softmax (1): embedding

Let the simplex

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

Then we have a variational characterization (exercise):

$$\Psi(\text{argmax} (x)) = \text{argmax} \ (x^\top z)$$

$$\quad z \in \Delta_{n-1}$$
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):

$$\text{softmax}_\epsilon(x) = \arg\max_{z \in \Delta_{n-1}} \left[ x^\top z + \epsilon H(z) \right]$$
Let $G = (G_1, \ldots, G_n)$ be i.i.d. Gumbel(0,1) random variables. Then we have (exercice):
\[
\text{softmax}_\epsilon(x) = E \arg\max_{z \in \Delta_{n-1}} [(x + \epsilon G) \top z]
\]

$G_i = -\ln(-\ln(U_i))$ where $U_i \sim \text{unif}(0, 1)$
From Argmax to Softmax: summary

1. **Embed**, such that

\[
\Psi(\text{argmax}(x)) = \arg\max_{z \in \Delta_{n-1}} (x^\top z)
\]

2. **Regularize** or perturb:

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

Both lead to efficient (stochastic) Jacobian estimates. Can we generalize this to other discrete operations, such as ranking?
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
How to define an embedding $\Phi : S_N \rightarrow \mathbb{R}^p$?

- Should encode **interesting features**
- Should lead to **efficient algorithms**

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

$$\forall \sigma_1, \sigma_2, \pi \in S_N, \quad \|\Phi(\sigma_1 \pi) - \Phi(\sigma_2 \pi)\| = \|\Phi(\sigma_1) - \Phi(\sigma_2)\|$$

Equivalently, the kernel should be **translation-invariant**

$$\forall \sigma_1, \sigma_2 \in S_N, \quad K(\sigma_1, \sigma_2) = \langle \Phi(\sigma_1), \Phi(\sigma_2) \rangle = \kappa(\sigma_1 \sigma_2^{-1})$$
How to define an embedding $\Phi : \mathbb{S}_N \rightarrow \mathbb{R}^p$?

- 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

$$\forall \sigma_1, \sigma_2 \in \mathbb{S}_N, \quad K(\sigma_1, \sigma_2) = \langle \Phi(\sigma_1), \Phi(\sigma_2) \rangle = \kappa(\sigma_1 \sigma_2^{-1})$$
Some attempts

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

$$< \Phi(\sigma), \Phi(\sigma') > = \text{Tr} \left( \Pi_\sigma \Pi_{\sigma'}^\top \right) = \text{Tr} \left( \Pi_\sigma \Pi_{\sigma'}^{-1} \right) = \text{Tr} \left( \Pi_\sigma \Pi_{\sigma'_{-1}} \right) = \text{Tr} \left( \Pi_{\sigma\sigma'_{-1}} \right)$$
Take $\sigma(x) = \text{rank}(x)$ with $x \in \mathbb{R}^N$

Fix a target quantile $f \in \mathbb{R}^n$

"Keep the order of $x$, change the values to $f$"

$$[\Psi_f(x)]_i = f_{\sigma(x)(i)} \iff \Psi_f(x) = \prod_{\sigma(x)}^T f$$
How to choose a "good" target distribution?

- Gaussian distribution (mean=0, sd=1)
- Uniform distribution
- Bigaussian distribution

Quantile functions:
- Gaussian
- Uniform
- Bigaussian
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.:

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

SUQUAN: jointly learn $f$ and the model:

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

![Chart showing AUC comparisons]

AUC on test set − median
AUC on test set − SUQUAN BND

0.60 0.65 0.70 0.75 0.80 0.85

AUC

cauchy
exponential
uniform
gaussian
median
SUQUAN
SVD
SUQUAN
BND
SUQUAN
SP AV
Experiments: CIFAR-10

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

![Images of original, median, SVD, and SUQUAN BND methods with corresponding graphs showing 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 "$i$ feature $i$ larger than feature $j$"?
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
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**
  
  $$
  \Phi(\sigma)^\top \Phi(\sigma') = \sum_{1 \leq i \neq j \leq n} \mathbf{1}_{\sigma(i) < \sigma(j)} \mathbf{1}_{\sigma'(i) < \sigma'(j)} = n_c(\sigma, \sigma')
  $$

  \(n_c = \text{number of concordant pairs}\)

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

  \(n_d = \text{number of discordant pairs}\)
Kendall and Mallows kernels

- The Kendall kernel is
  \[ K_T(\sigma, \sigma') = n_c(\sigma, \sigma') \]

- The Mallows kernel is
  \[ \forall \lambda \geq 0 \quad K^\lambda_M(\sigma, \sigma') = e^{-\lambda n_d(\sigma, \sigma')} \]

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

- 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
  \[
  K^\lambda_M(\sigma, \sigma') = e^{-\lambda n_d(\sigma, \sigma')},
  \]
  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 $\Phi : S_N \to \mathbb{R}^p$ defines a right-invariant kernel $K(\sigma_1, \sigma_2) = \Phi(\sigma_1)^\top \Phi(\sigma_2)$ if and only there exists $\phi : S_N \to \mathbb{R}$ such that

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

and

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

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

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

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

\[ \mathbb{R}^N \xrightarrow{\text{argsort}} S_N \xrightarrow{\text{embed}} \mathbb{R}^p \]

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

How to create a differentiable approximation to \( \Phi(\text{argsort}(x)) \)?
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 $B_n = \{ P \in \mathbb{R}^{n \times n}_+ \mid P1_n = P^\top 1_n = 1_n \}$ 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 \( \partial^2 h/\partial x \partial y > 0 \) (e.g., \( h(a, b) = (b - a)^2 \)).

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

\[
\argmin_{P \in B_n} \langle P, C(x) \rangle = \Pi_{\sigma(x)}.
\]
Entropic regularization (Cuturi et al., 2019)

\[ P_\varepsilon(x) = \arg\min_{P \in B_n} \langle P, C(x) \rangle - \varepsilon H(P) \]

**Algorithm 1: Sinkhorn**

**Inputs:** \( a, b, x, y, \varepsilon, \ell \)

\( K \leftarrow e^{-C_{xy}/\varepsilon}, u_0 = 1_n; \)

**for** \( t \leftarrow 0 \) **to** \( \ell - 1 \) **do**

\[ v_{t+1} \leftarrow b/K^T u_t \]

\[ u_{t+1} \leftarrow a/K v_{t+1} \]

**end**

**Result:** \( u_\ell, K, v_\ell \)

- \( P = \text{diag}(u_\ell)K\text{diag}(v_\ell) \) is the **differentiable** approximate permutation matrix of the input vector \( x \)

- Complexity \( O(nm\ell) \), GPU-friendly
\( \nabla P_\varepsilon (x) \) can be computed by
- Automatic differentiation of Sinkhorn iterations (Cuturi et al., 2019)
- Implicit differentiation (Cuturi et al., 2020)
\[ S_\varepsilon(x) = P_\varepsilon(x)x \quad R_\varepsilon(x) = P_\varepsilon(x)^\top (1, 2, \ldots, n)^\top \]

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)
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

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
Parallelization. The Sinkhorn computations laid out in Algorithm 1 imply the application of kernels $K$ or $K^T$ to vectors $v$ and $w$, which is designed to "capture" whatever values may lie close to that quantile. This choice results in $m = 3$, with

$$
\mathcal{O}_m(x) = \int y \cdot \mu(x) \, dx
$$

E.g., $x = \{3, x, x\}$, with

$$
\mathcal{O}_m(x) = \frac{1}{m} \sum_{i=1}^{m} \log(x_i)
$$

which improves the stability of summing exponentials [28, §4.4].

The choices that we have made are summarized in Alg. 2, but we believe there are opportunities to further Sinkhorn's algorithm, an important result to stabilize the transport matrices $\Pi$ inherited from OT.

- The freedom to choose a non-uniform target measure is important to standardize input vectors to be the regular grid on $[0, 1]^d$.
- Any nonnegative squashing function is critical to standardize input vectors to be the regular grid on $[0, 1]^d$.
- To illustrate the flexibility offered by this smooth approximation, we cast Sinkhorn iteration to the tilted distribution, with weights $\tau = 30\%$.
- The map $\mathcal{O}_m(x)$ is such that $\mathcal{O}_m(x)$ varies under $\tau$ and rescales the entries of $\mathcal{O}_m(x)$.
- Any nonnegative squashing function, such as $\mathcal{O}_m(x)$, can be obtained by squashings into $\mathcal{O}_m(x)$.
- The map $\mathcal{O}_m(x)$ is calculated as

$$
\mathcal{O}_m(x) = \frac{1}{m} \sum_{i=1}^{m} \log(x_i)
$$

when using small regularization strengths, we recommend to cast Sinkhorn iteration into the log-domain by considering the following stabilized iterations for each pair of vectors $v$ and $w$.

$$
\Pi_{ij} = \frac{1}{m} \sum_{i=1}^{m} \log(v_i) + \log(w_j)
$$

This smooth approximation can be obtained by squashings into $\mathcal{O}_m(x)$, such as $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, or a logistic map. We also notice in our experiments that it is important to standardize input vectors to be the regular grid on $[0, 1]^d$.

- When using small regularization strengths, we recommend to cast Sinkhorn iteration into the log-domain by considering the following stabilized iterations for each pair of vectors $v$ and $w$.

$$
\Pi_{ij} = \frac{1}{m} \sum_{i=1}^{m} \log(v_i) + \log(w_j)
$$

This smooth approximation can be obtained by squashings into $\mathcal{O}_m(x)$, such as $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, or a logistic map. We also notice in our experiments that it is important to standardize input vectors to be the regular grid on $[0, 1]^d$.

- When using small regularization strengths, we recommend to cast Sinkhorn iteration into the log-domain by considering the following stabilized iterations for each pair of vectors $v$ and $w$.

$$
\Pi_{ij} = \frac{1}{m} \sum_{i=1}^{m} \log(v_i) + \log(w_j)
$$

This smooth approximation can be obtained by squashings into $\mathcal{O}_m(x)$, such as $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, or a logistic map. We also notice in our experiments that it is important to standardize input vectors to be the regular grid on $[0, 1]^d$.

- When using small regularization strengths, we recommend to cast Sinkhorn iteration into the log-domain by considering the following stabilized iterations for each pair of vectors $v$ and $w$.

$$
\Pi_{ij} = \frac{1}{m} \sum_{i=1}^{m} \log(v_i) + \log(w_j)
$$

This smooth approximation can be obtained by squashings into $\mathcal{O}_m(x)$, such as $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, $\mathcal{O}_m(x)$, or a logistic map. We also notice in our experiments that it is important to standardize input vectors to be the regular grid on $[0, 1]^d$.
Application: soft top-$k$ loss

\[
\text{S-top-}k\text{-loss}(f_{\theta}(\omega_0), l_0) = \mathcal{J}_k \left( 1 - \left( \widehat{F}^{\ell} \left( \frac{1}{L}, f_{\theta}(\omega); \frac{1}{m}, y \right) \right)_{l_0} \right)
\]

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

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
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)

\[
\min_{\nu,U,V} \Delta(X, T_\nu(UV))
\]

\[X \approx UV\]

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

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

with $$\Phi : S_N \rightarrow \mathbb{R}^{n \times n}$$ the SUQUAN embedding

Smooth by entropy regularization

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

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

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

Smooth by regularization

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

with $\Omega : Z \to \mathbb{R}$ a regularization that makes $h_\epsilon$ 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

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
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

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

- It is obtained from the argmax by entropic regularization

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

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

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

  where $U$ is a random variable following the Gumbel distribution.
Smoothing by perturbation (Berthet et al., 2020)

\[ h(x) = \arg\min_{z \in C} \langle z, \Psi(x) \rangle \]

\[ h_\epsilon(x) = E \left[ \arg\min_{z \in C} \langle z, \Psi(x + \epsilon U) \rangle \right] \]

- Generalization of the “Gumbel trick” for soft-max (Gumbel, 1954)
- \( h_\epsilon \) is differentiable if the density of \( U \) is smooth (e.g. normal)
- **Stochastic gradient of** \( h_\epsilon \) **can be computed efficiently**
- Fast gradient for Fenchel-Young losses (Blondel et al., 2019)
- Sometimes equivalent to smoothing by regularization
We analyze the impact of the algorithmic parameters on optimization and generalization abilities. We exhibit the final loss and accuracy for different number of perturbations in the doubly stochastic gradient ($M = 1, 1000$). We highlight the importance of the temperature parameter on the algorithm (see Figure 2, right). Very high or low temperature degrades the ability to train and to generalize to test data, by lack of smoothing or loss of information about $\mathbf{x}$. We also observe that our framework is very robust to the choice of $\tau$, demonstrating it adaptivity.

Figure 3: Comparison on 21 datasets, for our proposed perturbed Fenchel-Young loss, a squared loss and the blackbox loss of Vlastelica et al. (2019). Points above the diagonal are datasets where our loss performs better.

5.2 Perturbed label ranking

We consider label ranking tasks, where each $y_i$ is a label permutation for features $x_i$. We minimize the weights of an affine model $g_w$ (i.e., $\mathbf{x}_i = g_w$) using our perturbed Fenchel-Young loss, as simple squared loss and the recently-proposed blackbox loss of Vlastelica et al. (2019). Note that our loss is convex in $\mathbf{x}$ and enjoys unbiased gradients, while (Vlastelica et al., 2019) uses a non-convex loss with gradient proxies. We use the same 21 datasets as in (Hüllermeier et al., 2008; Cheng et al., 2009).

- 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

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
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:

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

$$h_{\epsilon}^{\text{Perturbation}}(x) = E \left[ \arg \min_{z \in \mathcal{Z}} < z, \Psi(x + \epsilon U) > \right]$$
Example: shortest path (Berthet et al., 2020)

- Take a graph $G = (V, E)$, positive edge costs $c \in \mathbb{R}^E_+$, polytope
  \[ C = \left\{ y \in \mathbb{R}^E_+ : \forall i \in V, (\mathbf{1}_{i\rightarrow} - \mathbf{1}_{i\rightarrow})^\top y = \delta_{i=s} - \delta_{i=t} \right\} \]

- Shortest path (computable in $O(|E|)$ by DP) solves
  \[ y^* = \arg\min_{y \in C} < c, y > \]

- Differentiable shortest path (w.r.t $c$):
  \[ y_{\epsilon}^* = E \left[ \arg\min_{y \in C} < c + \epsilon U, y > \right] \]

Figure 5: In the shortest path experiment, training features are images. Shortest paths are computed based on terrain costs, hidden to the network. Training responses are shortest paths based on this cost.

Following Vlastelica et al. (2019), we train a network whose first five layers are those of ResNet18 for the Fenchel-Young loss between the predicted costs $\mathbf{x}_i^g(w,p,x,q)$ and the shortest path $y_i$. We optimize over $50$ epochs with batches of size $70$, temperature $1$ and $M = 1$ (single perturbation).

We are able, only after a few epochs, to generalize very well, and to accurately predict the shortest path on the test data. We compare our method to two baselines, from (Vlastelica et al., 2019):

- training the same network with their proposed blackbox loss and with a squared loss.

We show two metrics: perfect accuracy percentage and cost ratio to optimal path (see Figure 6); full implementation details are in Appendix C.3.

Figure 6: Accuracy of the predicted path for several methods during training.

Left. Percentage of test instances where the predicted path is optimal.
Right. Ratio of costs between the predicted path and the actual shortest path – without the squared loss baseline as it does not yield valid paths.

6 Conclusion

Despite a large body of work on perturbations techniques for machine learning, most existing works focused on approximating sampling, log-partitions and expectations under the Gibbs distribution. Together with novel theoretical insights, we propose to use a general perturbation framework to differentiate through, not only a max, but also an argmax, without ad-hoc modification of the underlying solver. In addition, by defining an equivalent regularizer $\Phi(w)$, we show how to construct Fenchel-Young losses and propose a doubly stochastic scheme, enabling learning in various tasks, and validate on experiments its ease of 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)
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
Conclusion

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


Harmonic analysis on $S_N$

- A representation of $S_N$ is a matrix-valued function $\rho : S_N \to \mathbb{C}^{d_\rho \times d_\rho}$ such that
  $$\forall \sigma_1, \sigma_2 \in S_N, \quad \rho(\sigma_1 \sigma_2) = \rho(\sigma_1) \rho(\sigma_2)$$

- A representation is irreductible (irrep) if it is not equivalent to the direct sum of two other representations

- $S_N$ 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 $f : S_N \to \mathbb{R}$, the Fourier transform of $f$ is
  $$\forall \lambda \in \Lambda, \quad \hat{f}(\rho_\lambda) = \sum_{\sigma \in S_N} f(\sigma) \rho_\lambda(\sigma)$$

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

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

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

and

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