Learning from ranks, learning to rank

Jean-Philippe Vert
ML

https://codeburst.io/machine-learning-243cc92247a1
Supervised learning beyond binary classification

Functions a Deep Neural Network Can Learn

input

Pixels:

Audio:

“How hello are you?”

output

“lion”

“How cold is it outside?”

“Bonjour, comment allez-vous?”

Pixels:

“A blue and yellow train travelling down the tracks”

Slide from Jeff Dean
Beyond supervised learning

Un- and Self-supervised learning

Generative models

Reinforcement learning
Beyond images and strings

Graph convolutions

Nodes

Activation function

Nodes

Regularization, e.g., dropout

Graph convolutions

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

http://snap.stanford.edu/decagon
What if inputs or outputs are permutations?

- 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
- \(|S_N| = N!\)
Examples

- Ranking data

- Ranks extracted from data

(histogram equalization, quantile normalization...
Goals

1. Permutations as input:

\[ \sigma \in S_N \mapsto f_\theta(\sigma) \in \mathbb{R}^p \]

How to define / optimize \( f_\theta : S_N \rightarrow \mathbb{R}^p \)?
- SUQUAN (Le Morvan and Vert, 2017), Kendall (Jiao and Vert, 2015, 2017, 2018)

2. Permutations as intermediate / output:

\[ x \in \mathbb{R}^N \mapsto \sigma(x) \in S_N \mapsto f_\theta(\sigma(x)) \in \mathbb{R}^p \]

How to differentiate the ranking operator \( \sigma : \mathbb{R}^N \rightarrow S_N \)?
- Sinkhorn CDF (Cuturi et al., 2019)
Assume your data are permutations and you want to learn

\[ f : S_N \rightarrow \mathbb{R} \]

A solutions: embed \( S_N \) to a Euclidean (or Hilbert) space

\[ \Phi : S_N \rightarrow \mathbb{R}^p \]

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 $\Phi : S_N \rightarrow \mathbb{R}^p$?

- 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**

$$\forall \sigma_1, \sigma_2, \pi \in S_N, \quad K(\sigma_1 \pi, \sigma_2 \pi) = K(\sigma_1, \sigma_2)$$
Some attempts

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

$$\langle \Phi(\sigma), \Phi(\sigma') \rangle = \Tr(\Pi_\sigma \Pi_{\sigma'}^\top) = \Tr(\Pi_\sigma \Pi_{\sigma'}^{-1}) = \Tr(\Pi_\sigma \Pi_{\sigma'-1}) = \Tr(\Pi_{\sigma \sigma'-1})$$
Link with quantile normalization (QN)

- 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 $f$"

$$\left[\Psi_f(x)\right]_i = f_{\sigma(x)(i)} \iff \Psi_f(x) = \prod_{\sigma(x)} f$$
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.:

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

SUQUAN: jointly learn $f$ and the model:

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

![Box plots and scatter plot showing AUC for various distributions and methods.](image)
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 $\sigma, \sigma' \in S_N$:

- **Inner product**

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

  $n_c = \text{number of concordant pairs}$

- **Distance**

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

  $n_d = \text{number of discordant pairs}$
The Kendall kernel is

\[ K_T(\sigma, \sigma') = n_c(\sigma, \sigma') \]

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

- 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)
Average performance on 10 microarray classification problems (Jiao and Vert, 2017).
Permutation as intermediate / output?

- Ranking operator:
  \[ \text{rank}(-15, 2.3, 20, -2) = (4, 2, 1, 3) \]

- Main problem:
  \[ x \in \mathbb{R}^N \leftrightarrow \text{rank}(x) \in \mathbb{S}_N \text{ is not differentiable} \]
Permutation as intermediate / output?

- Ranking operator:
  \[ \text{rank}(-15, 2.3, 20, -2) = (4, 2, 1, 3) \]

- Main problem:
  \[ x \in \mathbb{R}^N \mapsto \text{rank}(x) \in S_N \text{ is not differentiable} \]

---------------------

**Differentiable Sorting using Optimal Transport:**

The Sinkhorn CDF and Quantile Operator

Marco Cuturi  Olivier Teboul  Jean-Philippe Vert

Google Research, Brain team
From optimal transport (OT) to \textit{rank}?

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

- Solving OT in 1D is done in \( O(n \ln n) \) with the \textit{rank} function
- If \( \nu \) is ordered, then the solution \( P \) is the permutation matrix of \( \xi \)
- We propose instead to solve (a differentiable variant of) OT in order to recover (a differentiable variant of) \textit{rank}
Differentiable OT

\[ P_\epsilon = \arg\min_{P \in U(a,b)} \langle P, C \rangle - \epsilon H(P) \]

**Algorithm 1: Sinkhorn**

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

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

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

![Accuracy graph](image)

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

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


Harmonic analysis on $S_N$

- A representation of $S_N$ is a matrix-valued function $\rho : S_N \rightarrow \mathbb{C}^{d_\rho \times d_\rho}$ 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 $\Lambda = \{\lambda \vdash N\}$\(^1\) is the set of partitions of $N$

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

---

\(^1\) $\lambda \vdash N$ iff $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 \geq \ldots \geq \lambda_r$ and $\sum_{i=1}^r \lambda_i = N$
An embedding $\Phi : S_N \rightarrow \mathbb{R}^p$ defines a right-invariant kernel $K(\sigma_1, \sigma_2) = \Phi(\sigma_1)^\top \Phi(\sigma_2)$ if and only there exists $\phi : S_N \rightarrow \mathbb{R}$ such that

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

and

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