24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Permutation Learning via Lehmer Codes

Aïssatou Diallo[1] **Markus Zopf**[1] **Johannes Fürnkranz**[2]

Abstract. Many machine learning problems require to learn to permute a set of objects. Notable applications include ranking or sorting. One of the difficulties of learning in such a combinatorial space is the definition of meaningful and differentiable distances and loss functions. Lehmer codes are an elegant way for mapping permutations into a vector space where the Euclidean distance between two codes corresponds to the Kendall tau distance between the corresponding rankings. This transformation, therefore, allows the use of a surrogate loss converting the original permutation learning problem into a surrogate prediction problem which is easier to optimize. To that end, learning to predict Lehmer codes allows a transformation of the inference problem from permutations matrices to row-stochastic matrices, thereby removing the constraints of row and columns to sum to one. This permits a straight-forward optimization of the permutation prediction problem. We demonstrate the effectiveness of this approach for object ranking problems by providing empirical results and comparing it to competitive baselines on different tasks.

1 INTRODUCTION

Permutation learning is a long-standing problem in many scientific fields. The need to deal with permutations arises in a variety of applications, such as ranking [13], information retrieval or multi-object tracking [6] for computer vision. The fundamental underlying problem is the correct identification of the permutation that allows to reconstruct the original sequence. Ranking, sorting and matching problems are classical examples of the necessity to perform inference over such a combinatorial space, and they can be both represented by permutations over the set of objects to sort. However, one of the main difficulties in dealing with such combinatorial objects is the factorial growth of the number of permutations with growing number of elements. Moreover, there are mutual exclusivity constraints associated with permutations. Finally, the space of all possible permutations of a set of objects is not smooth. In fact, permutations are discrete objects and thus it is not possible to compute straight marginals with gradient-based methods. Thus, exact solutions are intractable to find for most cases.

One strategy for working with combinatorial objects is to embed them into continuous spaces, in order to impose a structure and a topology so that conventional tools for inference can be used. Prior works have proposed different approaches to deal with this problem. For example, several authors [4, 6] derived an approximation of a general probability distribution with a restricted set of basis functions, performing inference in the Fourier domain, and using accompanying transformations to project back to the combinatorial space.

[1] Technische Universität Darmstadt, Germany, email: [diallo@aiphes, zopf@ke].tu-darmstadt.de
[2] Johannes Kepler Universität Linz, Austria, email: juffi@faw.jku.at

Another line of work investigated convex surrogates of the permutations themselves. The classical workflow for tackling the discrete optimization problem over the set of permutations of n elements is the following: (1) use permutation matrices to represent permutations, (2) as these matrices are discrete, gradient-based optimisation is not possible; hence use a relaxation in order to transform them into the nearest convex, surrogate, doubly stochastic matrix, i.e., into a matrix where all rows and all columns sum up to one. In particular, permutation matrices may be viewed as special cases of doubly stochastic matrices which contain only zero and ones.

In this paper, we propose a different approach for learning permutations, which differs from previous work [6] in that we transform the permutation matrix from the combinatorial space to the Hilbert space using Lehmer codes [9]. Unlike [7], where the use of Lehmer codes was proposed for label ranking, we employ them for the task of object ranking. Lehmer codes provide a bijective mapping from permutations onto sequences of integers. Moreover, Lehmer codes enjoy algorithmic advantages such as that the L_1 norm of the code represents the Kendall's tau distance between the permutation that generated the code and the lexicographic permutation, which is a commonly used metric for assessing the ordinal association between ranking data. Lehmer codes provide vector representations of permutations in which the coordinates are decoupled, i.e., the values of each coordinate are not restricted by the other coordinates. This mapping allows to transform the problem from learning a doubly stochastic matrix to an integer valued vector which enjoys great advantages over the direct optimization of permutation matrices.

The remainder of the paper is organized as follows: Section 2 lays the necessary mathematical background and defines the notation used throughout the paper, whereas Section 3 introduces prior work on permutation learning and embedding of permutations. Section 4 presents our primary contribution, motivating our theoretical choices. Finally, Section 5 presents experiments which compare our proposed approach against alternative methods, illustrating the benefits of Lehmer codes for the task of permutation learning.

2 BACKGROUND

In this section, we formally state the permutation learning problem as well as the notation used throughout the paper, which closely follows the literature [7, 11].

2.1 Notation

Given an ordered list of items indexed by $\{1, ..., N\}$, let us denote without loss of generality, π_{I_n} as the n-dimensional identity permutation vector. A permutation is a rearrangement of the elements of π_{I_n}, i.e a bijection from π_{I_n} to π_n, a generic permutation vector with $\pi_n \in \mathcal{P}^n$. $\mathcal{P}^n$ denotes the set of all permutations vector of length n in the symmetric group, and has the cardinality $n!$.

2.2 Learning problem

The permutation learning problem aims at learning a function $s : \mathcal{X} \rightarrow \mathcal{P}^n$ that maps a feature space $\mathcal{X}$ to the permutation space $\mathcal{P}^n$.

Loosely speaking, given a sequence of shuffled items, the goal is to recover their natural order as defined by some specific criterion c. Formally, the underlying problem is to predict the permutation matrix $P_i \in \{0,1\}^{n \times n}$, given a sequence of shuffled items $\tilde{X}_i$ such that P_i^{-1} recovers the original ordered sequence X_i.

In such a setting, the cardinality of $\mathcal{P}$ is $n!$, which often is orders of magnitude larger than the number of training samples. This is one of the difficulties in dealing with combinatorial spaces such as $\mathcal{P}^n$. Moreover, permutations are represented as discrete points in the Euclidean space. This discrete nature typically prohibits gradient-based optimization, because these discrete solvers work by iteratively moving towards an optimum via stochastic updates along gradient directions. A large part of the dedicated literature proposed different relaxations of the permutation matrix, the best known of which involve the use of the Sinkhorn operator [18]. This operator is a procedure that takes as an input a non-negative square matrix and outputs a doubly-stochastic matrix by iteratively normalizing rows and columns[1].

Another approach consists of embedding the permutations into continuous spaces. An example of such line of work is given by [7], which proposes three different encodings suitable to act as embeddings for the task of label ranking. Among these encodings, Lehmer codes showed great algorithmic advantages, in particular, the fact that the encoding and decoding step have a cost of $\mathcal{O}(n)$.

2.3 Permutation matrices

As previously stated, there are several ways to represent permutations. In matrix theory, a permutation matrix is a square matrix with the peculiarity of having exactly a single unit value for each row and column and zero elsewhere. They are a natural and concise way to represent permutations. In fact, given an ordered sequence of elements represented as a vector π_{I_n}, it is possible to derive the permutation π_n from the permutation matrix $P_n \in \{0,1\}^{n \times n}$, simply by performing a vector-matrix multiplication:

$$\pi_n = P_n \pi_{I_n} \tag{1}$$

where P_n is obtained by swapping the rows of the identity matrix according to the desired permutation π_n. The set of the $n \times n$ permutation matrices has cardinality $n!$ and is the subset of the non-singular matrices in $\mathbb{R}^{n \times n}$. This set is closed under multiplication, which implies that $P^{-1} = P^T$.

A *row stochastic matrix* is a non-negative square matrix whose rows all sum up to one. Correspondingly, the transpose of such a matrix is said to be a *column stochastic matrix* because all the columns sum to one. If a matrix happen to be simultaneously row and column stochastic is said to be a *doubly stochastic matrix*. The characteristics of a doubly stochastic matrix A are:

$$A_{ij} \geq 0$$
$$A\mathbf{1} = \mathbf{1}$$
$$A^T\mathbf{1} = \mathbf{1}$$

where $\mathbf{1}$ is a column vector of ones.

Permutation matrices are a special case of doubly stochastic matrices. The Birkhoff-von Neumann theorem [10] states that any doubly stochastic matrix may be viewed as a convex combination of a finite number of permutation matrices. The set of $n \times n$ doubly stochastic matrices is the called the Birkhoff polytope (cf. Figure 1 (left)). A natural consequence that arises from this theorem is to consider doubly stochastic matrices as a convex relaxation of permutation matrices. However, it is difficult to learn matrices with this particular structure. The dedicated literature proposed to use Sinkhorn normalization [18, 5] to ensure that columns and rows sums to one. This procedure works by iteratively normalising rows and columns of the square matrix.

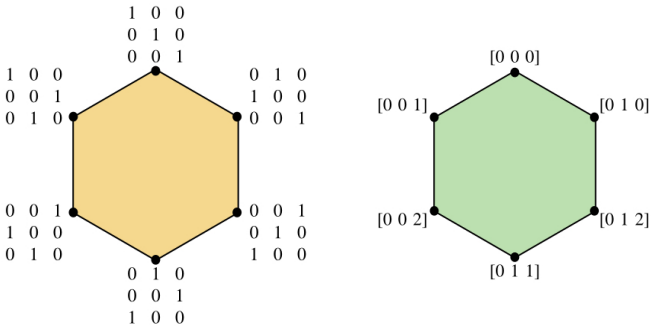


Figure 1: The Birkhoff (left) and Lehmer (right) polytopes of size 3. Permutations matrices are in the column representation.

2.4 Lehmer codes

Several works in theoretical computer science and discrete mathematics deal with effective ways to representing permutations. The *Lehmer code* [9] is a particular way to encode each possible permutation of a sequence of n numbers. Concretely, it is a bijection from every permutation $[\![N]\!]$ to a function $\phi : [\![N]\!] \rightarrow \{0, 1, \ldots, N-1\}$. The Lehmer code, also known as *inversion vector*, is a word of the form $c_\pi \in \mathcal{C}_N \triangleq \{0\} \times \{0,1\} \times \{0,2\} \times \cdots \times \{0, N-1\}$, where for $j = 1, ..., N$,

$$c_\pi(j) = \#\{i \in [\![N]\!] : i < j, \pi(i) > \pi(j)\} \tag{2}$$

Thus, the coordinate $c_\pi(j)$ represents the number of elements i with index smaller than j that are ranked higher than j in the permutation π. Consider the following example, which shows the canonical set of items e, a permutation π, and the corresponding Lehmer code c_π:

e	1	2	3	4	5	6	7	8	9
π	2	1	4	5	7	3	6	9	8
c_π	0	1	0	0	0	3	1	0	1

For example, the 6th digit of the Lehmer code $c_\pi(6) = 3$ because in π, three elements (4, 5, and 7) that appear to the left of the 6th element are larger than this element (3).

The mapping function based on the Lehmer code is the following

$$\begin{aligned} \phi : \mathcal{Y} &\rightarrow \mathbb{R}^N \\ \pi &\rightarrow (c_\pi(i))_{i=1,\ldots,N} \end{aligned} \tag{3}$$

resulting in the Lehmer polytope illustrated in Figure 1 (right).

One of the key advantages of the Lehmer embedding is that the sum of the coordinates of the Lehmer vector corresponds to the number of inversions of π with respect to the identity permutation e, i.e., $\|c_\pi\|_1 = d_\tau(\pi, e)$. In the above example, $\|c_\pi\|_1 = 6$, which means that six swaps have to be made in order to transform π into e (move 1 one entry to the left, 3 three entries to the left, 6 one entry to the left, and 8 one entry to the left). Moreover, its coordinates are decoupled, for this reason the decoding step is trivial.

2.5 Surrogate least square loss minimization

The structured output approach states the original prediction problem as follows:

$$\underset{s:\mathcal{X}\to\mathcal{Y}}{\text{minimize}} \text{ with } \mathcal{E}(s) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(s(x), \pi) dP(x, \pi) \quad (4)$$

In the standard setting for structured prediction, the quality of the prediction $s(x)$ is measured by a cost function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, i.e., the loss incurred by predicting $s(x)$ instead of the correct output π is given by $\Delta(s(x), \pi)$.

The *surrogate least squares framework* allows to shift the problem from the original combinatorial space to the Hilbert space by invoking a function $g : \mathcal{X} \to \mathcal{F}$ and a surrogate cost function $L(g(x), \phi(\pi))$. Additionally, ϕ is the embedding function for the permutations into the Hilbert space. In this setting, the original problem becomes:

$$\underset{f:\mathcal{X}\to\mathcal{F}}{\text{minimize}} \text{ with } \mathcal{R}(g) = \int_{\mathcal{X}\times\mathcal{Y}} L(f(x), \phi(\pi)) dP(x, \pi) \quad (5)$$

And the surrogate loss is represented by:

$$L(f(x), \phi(\pi)) = \|g(x) - \phi(\pi)\|^2 \quad (6)$$

To summarize, the surrogate least square problem can be divided in two steps:

1. Step 1: Define the regularized empirical risk $\min_g \frac{1}{n} \sum_{i=1}^{n} \|g(x_i) - \phi(\pi_i)\|^2 + \Omega g$ to provide an estimator for the square surrogate risk. g^* is the solution of this equation.
2. Step 2: Solve the decoding problem for every x in $\mathcal{X}$ that provides a prediction in the original space $\mathcal{P}^n$: $\hat{s}(x) = \text{argmin}_{\pi\in\mathcal{P}^n} \|\phi(\pi) - g^*(x)\|^2$.

3 RELATED WORK

The problem of reasoning about permutations appears in several applications in different fields of the scientific community. Broadly speaking, the task of permutation learning consists of learning the underlying order for a collection of objects based on a predetermined criterion. In information retrieval, for example, a good model should select the documents and sort them according to the permutation that maximizes the given criterion. Another common example from computer vision, is the Jigsaw problem, consisting of reconstructing an image from a set of puzzle parts [17, 15] (cf. Figure 2). In natural language processing, the task of reordering sentence from a collection of shuffled sentences is a typical case involving sorting and permutation learning [8]. Or again, DNA/RNA modeling in biology or re-assembling relics in archaeology can be formulated as permutation learning problems [19].

Unfortunately, maximizing the marginal likelihood for problems that involve latent permutations is very difficult. The exploding cardinality of the combinatorial space is a factor that makes exact inference intractable. This is due to the fact that it is not possible to treat these problems as if they involved categorical latent variables since the computation of the partition function is intractable.

Moreover, permutations are discrete objects in the Euclidean space, hence gradient-based optimization methods are not adapted. In fact, these solvers start with an initial point and iteratively update it by making small steps towards an optimum. A number of works in the dedicated literature have considered approximating the non-differentiable parameterization of a permutation with a differentiable

Figure 2: Illustration of the jigsaw puzzle problem, which consists of identifying the underlying permutation that sorts the shuffled tiles in order to reconstruct the image. This is an example of permutation learning task.

relaxation through the Sinkhorn operator. This operator proceeds by iteratively normalizing the rows and the columns of output matrix, in order to obtain a doubly stochastic matrix. Permutation matrices being a special case of permutation matrices, and drawing from the Birkhoff's theorem, the double stochastic matrices obtained through the Sinkhorn operator can be seen as the result of the convex combination incorporating uncertainty around the rank of the j^{th} item. This methodology is, e.g. used in [1, 17]. The approach taken in [14] extends these cited works because it adds a Gumbel noise component to the output matrix before applying the Sinkhorn operator, leveraging a temperature parameter.

Along the same line of work, [3] proposed a continuous relaxation of the permutation learning problem, which involves transforming the permutation matrix into a unimodal row-stochastic matrix, that is to say a positive real square matrix where each row sums to one. The work relies on a temperature parameter which controls the degree of approximation to the real underlying permutation matrix. This approach allows to perform straight-through gradient optimization [2] which requires exact permutation matrices to evaluate the learning objective. Our approach relates to this work with respect to the form of the output of their network. In fact, we propose to predict the Lehmer code of the permutation rather than the permutation itself. The nature of this encoding is such that its differentiable matrix form is a row-stochastic matrix that is obtained applying the softmax operator row-wise on the output matrix. The use of Lehmer codes has been proposed before in [7] for the task of label ranking. The main difference to our work is that we tackle the task of object ranking. Moreover, we extend the surrogate least square prediction framework to the use of the cross-entropy loss.

It is also possible to deal with inference over combinatorial objects such as permutations by shifting the domain. For example, [4, 6] approximated distributions over permutations with the low-frequency Fourier components. This work states that it is natural to approximate smooth association distributions over the intractably large permutation space $\mathcal{P}^n$ by their first few Fourier matrices, very much analogous to the way how smooth periodic functions on $\mathbf{R}^n$ can be approximated by their first few Fourier components.

Another representative work along the same lines is [16], which proposed a continuous relaxation from permutation matrices to points on a hypersphere, and then use the von Mises-Fisher (vMF) distribution to model distributions on a sphere's surface. By doing this, they map the $n!$ permutation space to a $(n-1)^2$ space. Or again, Linderman et al [12], who relaxed permutations to points in the Birkhoff polytope and derived temperature-controlled densities such that, as the temperature goes to zero, the distribution converges to an atomic density on permutation matrices.

Besides exploring approximation with respect to permutation matrices, [7] proposed different embeddings suitable for permutation vectors for the task of label ranking. This work adopts a least square

surrogate loss approach to solve the structured output regression problem and for doing so, they propose three different embeddings for permutations, namely: Kemeny embeddings, Hamming embeddings and Lehmer embeddings. The authors went along to demonstrate theoretical guarantees and algorithmic complexity for the three vector representations.

We build upon this work by exploiting the Lehmer transformation and applying it to the underlying permutation problem. In this way, the original problem of finding good approximations for the permutation matrices becomes the matrix form of an integer-valued vector, where the coordinates are decoupled, which is one of the interesting properties of the Lehmer codes. In the following section, we will further explain the characteristics of the Lehmer codes and how we apply this transformation to the original permutation learning problem.

4 APPROACH

The permutation prediction task aims at predicting a permutation P_i which when applied to an ordered sequence X_i gives as a result a desired shuffled sequence $\tilde{X}$. More concretely, the permutation learning task takes as an input a set of shuffled items and outputs the permutation matrix that shuffled the original set.

In the following, we first formally define this problem, and then sketch our approach for tackling it.

4.1 Problem definition

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{M}$ be the dataset with $\tilde{x}_i \in \mathbb{R}^d$ the i-th sequence of shuffled objects and $y_i \subseteq \mathcal{P}$ the corresponding ground-truth permutation, where $\mathcal{P} \triangleq \{1, 2, ..., N\}$ is the set of all permutations of the indexed elements to sort.

For example, given a set of ordered sequences, a dataset of this type can be obtained on the fly by shuffling all sequences with random permutations. This is a method to obtain a great amount of training data with low computational cost.

The learning task is then to recover the original permutation from the random sequence, based on the characteristics of the randomly shuffled objects.

4.2 Principal solution approaches

As stated earlier, there are two ways to tackle the permutation learning problem. The first consists in treating the permutation, (or the permutation matrix) as it is, while the second approach transforms the permutation into a different representation. In other words, the first approach learns the following parameterized function:

$$g_\theta \ : \ \mathcal{X} \rightarrow \mathcal{P}$$
$$\text{minimize}_\theta \sum_{(\mathcal{X},\mathcal{P})\in\mathcal{D}} \Delta(P, f_\theta(\tilde{X})) + R(\theta)$$

where $\tilde{X}$ is the sequence of shuffled objects, P is a permutation, Δ is the loss, θ the parameters of the objective function and $R(\theta)$ a regularization term. Permutation matrices are discrete points in the Euclidean space, hence the object of (4.2) is actually a convex surrogate of the permutation matrix at hand, i.e., a doubly stochastic matrix.

The second approach, which is the object of this paper, learns the following function:

$$f_\theta \ : \ \mathcal{X} \rightarrow \phi(\mathcal{P})$$
$$\text{minimize}_\theta \sum_{(\mathcal{X},\mathcal{P})\in\mathcal{D}} \Delta(\phi(P), f_\theta(\tilde{X})) + R(\theta)$$

where ϕ is the Lehmer embedding function defined in (3). We chose to format the Lehmer code in matrix form, which is simply the one-hot encoding of the vector itself. In a nutshell, given a tuple consisting of shuffled items and a ground-truth permutation, our objective is to learn the Lehmer code of the ground-truth permutation. The encoded prediction is subsequently decoded in order to map it back to the original combinatorial space.

4.3 Permutation learning using Lehmer codes

To solve the permutation learning problem, we propose to leverage Lehmer codes to provide a surrogate representation of permutations. In fact, permutation matrices, as mentioned in section 2 are particular discrete elements with a specific structure where columns and rows sum to one. A convex relaxation of permutation matrices is represented by doubly stochastic matrices which can be interpreted as marginals over the matrices themselves.

However, learning a mapping to the set of doubly stochastic matrices, (i.e., to the Birkhoff polytope) is not easy because said matrices are difficult to parameterize. Instead, we propose to learn a matrix form of the Lehmer encoding of the permutations. As previously stated, Lehmer codes are permutation codes, which provide several computational advantages. One of them, particularly relevant for our goal, is the property of independence of relative ranks. That is to say that the coordinates of the Lehmer embedding are decoupled. Loosely speaking, a Lehmer code is a sequence of random variables, independently drawn from uniform distributions on $[N - i]$, where i defines the i-th position in the sequence. We motivate this choice by the fact that such representation doesn't have the constraints of double stochasticity proper of permutation matrices. Moreover, the disjoint nature of the coordinates of the Lehmer code allows to represent them as a simple concatenation of feature vectors of the elements of the sequence to sort. In other terms, Lehmer codes allow to map permutation matrices to row-stochastic matrices. This is an advantage because we can easily construct a surjective function using row-wise softmax function in order, followed by the decoding step in order to retrieve the original permutation.

4.4 Neural network training strategy

After encoding the training permutations, we perform end-to-end learning with gradient descent. The least surrogate loss framework provides an alternative way to standard empirical risk minimization. The goal of aforesaid framework is to directly find the model that best explains the training data within a determined hypothesis space.

All experiments rely on networks that have the same structure. First, a convolutional neural network shared among all elements is employed for the feature extraction step. We concatenate the feature vectors into a matrix form and feed this matrix to the network that will compute the final square matrix. Row-wise softmax is then applied in order to transform the output matrix into a row-stochastic matrix. A natural loss function for such setting is the binary cross entropy between the prediction and the ground-truth matrix, which we use for all baselines.

5 EXPERIMENTS

In this section, we report the results of various experiments comparing our proposed approach to previous works on learning permutations. We provide experimental results that allow us to assess the advantages of predicting Lehmer codes over the direct prediction of the permutation or the prediction of some convex surrogates of the permutation matrix.

We recall that we wish to predict the latent permutation in order to sort a collection of given shuffled objects according to a criterion c. With this objective in mind, we perform two experiments comparing the use of Lehmer codes to existing methods: In the first setting, we explore the capabilities of Lehmer codes for the task of solving *jigsaw puzzles*. This task consists of reconstructing an image from a sequence a tiles in order to recover the original spatial layout, as illustrated in Figure 2. The second experiment aims at *sorting* a set of random unsorted objects into a sorted list. We will present quantitative and qualitative results of reconstructed images and sorted sequences, but first start with a brief description of our experimental setup.

5.1 Experimental Setup

Baselines. As representatives of the state-of-the-art in permutation learning, we implemented the following baselines:

- *Vanilla row-stochastic*: This is the naive approach which consists in casting the permutation learning problem to an n^2 binary classification problem and optimizing the cross entropy in a row-wise fashion. It is worth noting that this approach does not take into account the geometry of permutation matrices and for this reason has several inefficiencies.
- *Sinkhorn* [1]: As discussed in Section 3, the Sinkhorn operator allows to transforms a non-negative square matrix into a doubly stochastic matrix by repeatedly normalizing rows and columns. The Sinkhorn networks interpret such matrices as marginals of the distribution over permutations.
- *Gumbel-Sinkhorn* [14]: This approach uses the Sinkhorn operator with the addition of Gumbel noise to obtain doubly stochastic matrices that define a latent distribution over the permutation matrices themselves. This allows to perform posterior inference on the distribution of permutation matrices.

All implemented baselines as well as our approach based on Lehmer code have a joint feature extraction layer in common, which is a deep convolutional neural network shared across all items in the sequence to sort (cf. Section 4.4). The aim is to map each concatenated image to intermediate representations in a feature space.

Evaluation metrics. We use three commonly used metrics for comparing our results obtained with our approach against the baselines [14]:

- *Kendall's tau coefficient* (k_τ): A distance function of interest focuses on transpositions. A transposition (i, j) is a swap of elements at positions i and j, with $i \neq j$. The smallest number of such transpositions denotes the Kendall's tau distance between σ and π, $d_\tau(\sigma, \pi)$, which is defined as

$$k_\tau(\sigma, \pi) = \#\{(i, j) : \pi(i) > \pi(j), \sigma(i) < \sigma(j)\} \qquad (7)$$

- *Proportion of Correctly identified permutations* (PC): This measure is useful to assess the capabilities of our approach to correctly identify entire permutations. This metric is particular relevant for permutation learning where the underlying goal is to correctly match elements to their correct position in the ordered sequence.
- *Proportion of Any Correctly identified element* (PAC): In addition to the proportion of entire permutations correctly found, another interesting metric is the proportion of individual items correctly ranked in the sequence. In combination with the previous metric, these two metrics evaluate the performance of permutation learning models with respect to accuracy.

5.2 Jigsaw Puzzles

Our first experiment concerns the task of solving jigsaw puzzles which is a complex scenario to test the capabilities of the proposed approach in the context of convolutional neural networks. The goal is to reconstruct an image from a collection of shuffled pieces of the same image (see also Figure 2). In this particular case, the input consists of N non-overlapping, equally-sized pieces in the same orientation as the original image. Specifically, given a a puzzle of size $H \times W$, the input is a sequence of $N = H \times W$:

$$X = (x_1, x_2, ..., x_N) = \pi(1, 2, ..., N)$$

such that x_i is the index of the piece of puzzle. The objective is to learn a function $\pi(X)$ which takes the input sequence X and returns a permutation:

$$\pi(X) = \sigma(1, 2, ..., N)$$

that represents the order in which the input image pieces should be sorted to reconstruct the original image. Figure 3 illustrates the task.

Figure 3: Illustration of the jigsaw puzzle task. Given an image of shuffled pieces, the task is to recover the permutation that will recover the original spatial layout.

In this particular case, the latent ranking criterion c is the reconstruction of the original spatial layout, i.e., the first rank should be assigned to the image part that constitutes the upper left corner of the image, and last rank to the lower right corner. A good model for the task of solving jigsaw puzzles has to be able to identify individual tiles, and compare them against each other in order to determine their correct relative position in the sequence.

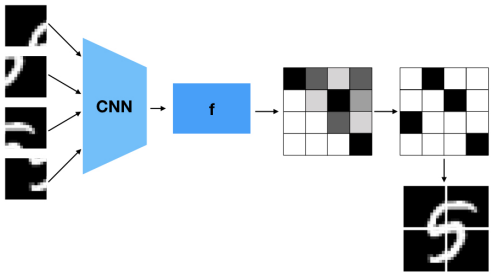
We evaluate our approach using two benchmark image datasets, MNIST and CIFAR10, the second one being more challenging due to the high multimodality and the lack of a clear sequential structure that generalizes over images. Nonetheless, we will see that our approach based on Lehmer codes outperforms the baseline models.

Setting For both datasets, we first normalize the inputs to have zero mean and standard deviation one. MNIST is composed of images of dimensions 28×28 pixels, whereas CIFAR10 images have 32×32 pixels. For cases in which the image is not exactly divisible by the number of tiles, we pad the original image to the next larger size that allows such a partition. Finally, the tiles are arranged into a sequence and randomly shuffled.

Table 1: Jigsaw puzzle results. We evaluate Kendall's tau coefficient, the proportion of correctly identified permutations (PC) and the proportion of individual elements correctly ranked (PAC). We compare our results to [14] for the available datasets (MNIST) and to a Sinkhorn based network as in [1] which doesn't need fine-tuning additional parameters. Randomly sorted elements have a PAC score of $(n-1)/n$.

		MNIST				CIFAR10			
		2x2	3x3	4x4	5x5	2x2	3x3	4x4	5x5
k_τ	Sinkhorn	1.0	0.82	0.42	0.31	0.82	0.72	0.35	0.21
	Gumbel Sinkhorn	1.0	0.83	0.43	0.39	–	–	–	–
	Lehmer code	**1.0**	**0.85**	**0.50**	**0.39**	**0.99**	**0.96**	**0.73**	**0.69**
PC	Sinkhorn	1.0	0.89	0.0	0.0	0.74	0.11	0.0	0.0
	Gumbel Sinkhorn	1.0	**0.91**	0.0	0.0	–	–	–	–
	Lehmer code	1.0	0.84	0.0	0.0	**0.99**	**0.75**	0.0	0.0
PAC	Sinkhorn	1.0	0.69	0.44	0.37	0.84	0.50	0.18	0.06
	Gumbel Sinkhorn	1.0	**0.97**	**0.45**	**0.45**	–	–	–	–
	Lehmer code	1.0	0.66	0.41	0.11	**0.99**	**0.92**	**0.34**	**0.18**

Implementation details Following previous works, we process each tile with a 5×5 convolutional network, with padding and stride 1 and 2×2 max pooling. This step is to obtain intermediate representations for each tile which are then fed to our network to output a square matrix. Figure 4 summarizes the network architecture used for this task.



Conv[Kernel: 5x5, Stride: 1, Output: 28x28x32, Act.: Relu]
→ Pool[Stride: 2, Output: 14x14x32]
→ FC[Units: 64, Activation: Relu]

Figure 4: Network architecture for the jigsaw puzzle task. It consists of an CNN for feature extraction of each tile and a fully connected network that maps each vector to a hidden dimension of 64.

Evaluation Measures Following the work of [14], evaluation on the test data is assessed with different metrics: (i) Kendall tau coefficient, which is a measure of rank correlation on ranked data, (ii) proportion of individual tile ranks correctly assigned and (iii) the proportion of permutations correctly identified. We train the model with Adam optimizer at a learning rate of $1e^{-3}$ and a batch size of 32. The CNN maps each tile to a feature vector of dimension $d = 64$ for both datasets.

Results As stated earlier, we compare the results obtained from our approach with two alternative methods to optimize permutations. The results are presented in Table 1. These two methods, namely *Sinkhorn* and *Gumbel-Sinkhorn* are designed particularly for the task of recovering permutation matrices for matchings. On the other hand, our method is based on Lehmer codes which by nature are designed to minimize the Kendall tau's correlation. This explanation is highlighted by the obtained results, in fact we can notice that our approach outperforms the baselines for all n with respect to the Kendall's tau metric. The lower performance of our approach shown in Table 1 compared to the Sinkhorn baselines is due to the peculiarity of Lehmer codes. In fact, the coordinates of the encoding are disjointed so an error in the prediction is actually an error in estimating the relative position of the elements with index bigger than that particular coordinate. This characteristic is particularly desirable for the task of ranking but is sub-optimal for matching.

Moreover, the gap of performance between the datasets is of particular interest. In fact, for a growing number of tiles, the jigsaw puzzle becomes ill-posed in particular for the MNIST dataset. In fact, when n increases, there are many black tiles that become indistinguishable between each other. We can notice that the low accuracy problem is not present CIFAR10 dataset, even though it is a more challenging dataset than MNIST. This confirms our hypothesis that learning a matrix form of the Lehmer code allows to learn permutations with the objective of sorting according to the spatial layout.

5.3 Sorting Sequences of Handwritten Digits

For this particular experiment, we adapt the MNIST dataset in order to create sequences of random handwritten digits. More specifically, the dataset we used for this particular experiment is a set of multi-digit images obtained by concatenating 4 random MNIST images, and the task is to sort resulting set of 4-digit numbers. Figure 5 is an illustration of the task of sorting handwritten multi-digits. Training, validation and test sets are then obtained by random sampling of multi-MNIST images.

Figure 5: Illustration of the task of sorting a sequence of multi handwritten MNIST digits. The goal is to recover the ascending order of the items of each sequence.

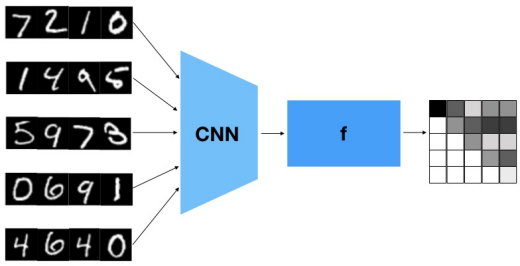
Given a sequence of n multi-MNIST images, the task is to predict the permutation such that the sequence of images is sorted in ascending order. The supervision signal comes from the ground truth permutation. This task is more challenging than sorting scalar numbers. In fact, in order to perform well on this task, a model has to learn meaningful semantics of the high-dimensional elements and learn the structure of the elements in order to be able to sort the multi-digits.

Table 2: Average performance for sorting handwritten digits. Shown are Kendall's tau coefficients as well as the proportions of fully correctly predicted permutations and the proportion of individual objects of the sequence whose position was correctly predicted.

	Model	$n = 3$	$n = 5$	$n = 7$	$n = 9$	$n = 15$
k_τ	Vanilla RS	0.75	0.58	0.28	0.23	0.11
	Sinkhorn	0.56	0.47	0.39	0.32	0.21
	Gumbel-Sinkhorn	0.56	0.46	0.40	0.27	0.19
	Lehmer code	**0.80**	**0.62**	**0.49**	**0.38**	**0.25**
PC	Vanilla RS	0.467	0.093	0.009	0.0	0.0
	Sinkhorn	0.462	0.038	0.001	0.0	0.0
	Gumbel Sinkhorn	0.484	0.033	0.001	0.0	0.0
	Lehmer code	**0.809**	**0.379**	**0.112**	**0.013**	0.0
PAC	Vanilla RS	0.801	0.603	0.492	0.113	0.067
	Sinkhorn	0.561	0.293	0.197	0.143	0.078
	Gumbel Sinkhorn	0.575	0.295	0.189	0.146	0.078
	Lehmer code	**0.869**	**0.676**	**0.511**	**0.367**	**0.155**

The learning process is supervised by the ground-truth permutation. It is worth mentioning that this is a weak signal for this task. In fact, ideally, the model should be able to learn to separate the individual digits of each element of the sequence of multi-MNIST images. Then, it should rank the digits in order to retrieve the underlying correct permutation, while lacking the image labels available when optimizing for the classification task.

Implementation Details As in the previous experiment, we compare against the Vanilla row-stochastic model, and the two variants of the Sinkhorn networks. We slightly modified the configuration of the previous network in order to adapt to the complexity of this experiment. In this case, the architecture is composed by stacking two convolutional networks of the same type used in the jigsaw experiment. We follow the common practice of normalizing to 0 mean and standard deviation equal to 1 for every individual MNIST image before concatenation. Thus, the sequence of input images has dimension $n \times 4 \times 28$, i.e., each element of the sequence is a 112×28 image. Figure 6 summarizes the network architecture for this task.



Conv[Kernel: 5x5, Stride: 1, Output: 140x28x32, Act.: Relu]
→ Pool[Stride: 2, Output: 70x14x32]
→ Conv[Kernel: 5x5, Stride: 1, Output: 70x14x64, Ac.: Relu]
→ Pool[Stride: 2, Output: 35x7x64]
→ FC[Units: 64, Activation: Relu]

Figure 6: Network architecture for the handwritten digits sorting task.

Evaluation Measures In order to assess the performance of our model, we compute the same metrics as in the jigsaw puzzle experiment. The proportion of individual ranks correctly identified and permutations correctly identified constitutes good metrics to assess the accuracy of the sorting task. Moreover, we again compare models based on the Kendall tau coefficient, which is the the optimal metric to assess the capabilities of a model to learn an ordinal association between objects.

Results The results are shown in Table 2. For this particular task, we can observe that the Lehmer code based approach consistently outperforms the baselines with respect to the Kendall's tau metric. Moreover, in many cases, we can also see a considerable improvement for the PC and the PAC metrics. It is, however, worth mentioning, that the Vanilla RS baseline performs well in ranking individual elements but the same performance is not found for predicting the overall permutation. This experiment is more related to the ranking task rather than the sorting task, hence the improvement over the baselines across all metrics for most cases of sequences with increasing length.

6 CONCLUSION

In this paper, we have proposed a new way to perform optimization for the task of permutation learning. We present an alternative methodology to obtain encode permutations which is based on Lehmer codes, which have previously been proposed for label ranking tasks. framing the problem in this way allows to optimize for the objective of minimizing the Kendall tau rank correlation, which is an ordinal metric for assessing the association between permutations. Moreover, Lehmer codes come with the natural advantage of not relying on the hard constraint of doubly stochastic matrices in which columns and rows have to sum to one, but can optimize on simpler row-stochastic matrices. We motivated our choices on two challenging experiments regarding sorting and matching complex items (images), namely solving jigsaw puzzles and sorting multi-MNIST digits, where we generally outperformed competitive models based on approximations of permutations matrices, in particular with respect to Kendall's tau. In future work, we would like to explore the use of Lehmer codes in more complex scenarios, such as variational inference permutation.

Acknowledgments

We thank Eneldo Loza Mencía for valuable discussion and feedback. This work has been supported by the German Research Foundation as part of the Research Training Group *Adaptive Preparation of Information from Heterogeneous Sources* (AIPHES) under grant No. GRK 1994/1.

REFERENCES

[1] Ryan Prescott Adams and Richard S Zemel, 'Ranking via Sinkhorn propagation', *arXiv preprint arXiv:1106.1925*, (2011).
[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, 'Estimating or propagating gradients through stochastic neurons for conditional computation', *arXiv preprint arXiv:1308.3432*, (2013).
[3] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon, 'Stochastic optimization of sorting networks via continuous relaxations', in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, (2019).
[4] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas, 'Fourier theoretic probabilistic inference over permutations', *Journal of Machine Learning Research*, **10**(May), 997–1070, (2009).
[5] Philip A Knight, 'The Sinkhorn–Knopp algorithm: convergence and applications', *SIAM Journal on Matrix Analysis and Applications*, **30**(1), 261–275, (2008).
[6] Risi Kondor, Andrew Howard, and Tony Jebara, 'Multi-object tracking with representations of the symmetric group', in *Artificial Intelligence and Statistics*, pp. 211–218, (2007).
[7] Anna Korba, Alexandre Garcia, and Florence d'Alché Buc, 'A structured prediction approach for label ranking', in *Advances in Neural Information Processing Systems*, pp. 8994–9004, (2018).
[8] Mirella Lapata, 'Probabilistic text structuring: Experiments with sentence ordering', in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 545–552. Association for Computational Linguistics, (2003).
[9] Derrick Henry Lehmer, 'Teaching combinatorial tricks to a computer', in *Combinatorial Analysis*, eds., Richard Bellman and Marshall Hall Jr., volume 10 of *Proceedings of the Symposium on Applied Mathematics*, pp. 179—193. American Mathematical Society, (1960).
[10] JL Lewandowski, CL Liu, and Jane W.-S. Liu, 'An algorithmic proof of a generalization of the Birkhoff–von Neumann theorem', *Journal of Algorithms*, **7**(3), 323–330, (1986).
[11] Pan Li, Arya Mazumdar, and Olgica Milenkovic, 'Efficient rank aggregation via lehmer codes', in *Artificial Intelligence and Statistics*, pp. 450–459, (2017).
[12] Scott W. Linderman, Gonzalo E. Mena, Hal Cooper, Liam Paninski, and John P. Cunningham, 'Reparameterizing the Birkhoff polytope for variational permutation inference', in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AI-STATS)*, eds., Amos J. Storkey and Fernando Pérez-Cruz, pp. 1618–1627, Lanzarote, Canary Islands, (2018). PMLR.
[13] Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff A Bilmes, 'Consensus ranking under the exponential model', *arXiv preprint arXiv:1206.5265*, (2012).
[14] Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek, 'Learning latent permutations with Gumbel-Sinkhorn networks', in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, (2018).
[15] Mehdi Noroozi and Paolo Favaro, 'Unsupervised learning of visual representations by solving jigsaw puzzles', in *European Conference on Computer Vision*, pp. 69–84. Springer, (2016).
[16] Sergey Plis, Stephen McCracken, Terran Lane, and Vince Calhoun, 'Directional statistics on permutations', in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 600–608, (2011).
[17] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould, 'DeepPermNet: Visual permutation learning', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3949–3957, (2017).
[18] Richard Sinkhorn and Paul Knopp, 'Concerning nonnegative matrices and doubly stochastic matrices', *Pacific Journal of Mathematics*, **21**(2), 343–348, (1967).
[19] Robert Willenbring, 'RNA secondary structure, permutations, and statistics', *Discrete Applied Mathematics*, **157**(7), 1607–1614, (2009).