Permutation Classifiers

Xinrui Zhou(1), Concettina Guerra(1), Jarek Rossignac(1), Leo Rossignac-Milon(2)

(1) College of Computing, Georgia Institute of Technology, TSRB Building, 85 5th St NW, Atlanta, GA, USA
(2) Axon, 1100 Olive Way 1300, Seattle, WA 98101, USA

Abstract

We consider permutations of a given set of n different symbols. We are given two unordered training sets, T_1 and T_2, of such permutations that are each assumed to contain examples of permutations of the corresponding type, t_1 and t_2. Our goal is to train a classifier, $\mathcal{C}(\mathbf{q})$, by computing a statistical model from T_1 and T_2, which, when given a candidate permutation, $\mathbf{q}$, decides whether $\mathbf{q}$ is of type t_1 or t_2. We discuss two versions of this problem. The *ranking* version focuses on the order of the symbols. Our Separation Average Distance Matrix (SADiM) solution expands on previously proposed ranking aggregation formulations. The *grouping* version focuses on contiguity of symbols and hierarchical grouping. We propose and compare two solutions: (1) The Population Augmentation Ratio (PAR) solution computes a PQ-tree for each training set and uses a novel measure of distance between these and $\mathbf{q}$ that is based on ratios of population counts (i.e., of numbers of permutations explained by specific PQ-trees). (2) The Difference of Positions (DoP) solution is computationally less expensive than PAR and is independent of the absolute population counts. Although DoP does not have the simple statistical grounding of PAR, our experiments show that it is consistently effective.

Keywords: Permutations, Classification, Average, Distance

1. Introduction

1.1. Overall problem statement

In this paper, we discuss permutations (i.e, ordered sequences without omission or repetition) of any given set of n different symbols. In our examples, we use the first n letters of the alphabet and consider their lexicographic ordering (*abcd*) as the non-permuted, base (i.e., identity) permutation.

If $\mathbf{p}$ is such a permutation, we denote by $\mathbf{p}[i]$, for i in $[1,n]$, the symbol at position i in $\mathbf{p}$, and by $\mathbf{p}^{-1}[s]$ the position (i.e., integer in $[1,n]$) of symbol s in $\mathbf{p}$. For example, if $\mathbf{p} = cdab$, then $\mathbf{p}[3] = a$ and $\mathbf{p}^{-1}[a] = 3$.

Our goal is to develop and validate representations, algorithms, and mathematical formulae for the following problem. We are given two training sets, T_1 and T_2, of permutations. These training sets do not need to have the same number of permutations and need not be disjoint: i.e., one or more permutations may be included in both. Each training set contains permutations that, in some application-dependent sense, are examples taken from a much larger population of permutations. We seek a computational solution (an algorithm and an associated digital model) that "learns" a binary classifier, $\mathcal{C}$, such that, for any candidate permutation $\mathbf{q}$, $\mathcal{C}(\mathbf{q})$ returns a Boolean value indicating whether $\mathbf{q}$ is more likely to be of type t_1 (i.e., part of the population illustrated by T_1) or of type t_2. Furthermore, in order to reduce the computational cost of each classification query, we want its space and time computational complexity to be only a function of n, the number of symbols, but not of the number of permutations in the training sets, and certainly not in the size of the populations defined by the training sets. Hence, we focus on approaches that pre-compute and store a model of each population and possibly of measures of statistical correlations between populations. We then use the resulting model to evaluate $\mathcal{C}(\mathbf{q})$ for each query permutation, $\mathbf{q}$.

1.2. Discussion

In this subsection, we discuss the intellectual merit of the problem addressed here, the nature of our contributions, and their novelty. Their limitations and anticipated future work are discussed in the conclusion.

1.2.1. Merit

To appreciate the intellectual merit of the problem stated above, consider first the two populations (set of permutations) of types t_1 and of type t_2. Let us distinguish four situations: (1) the candidate permutation $\mathbf{q}$ is of type

t_1 but not t_2, (2) the reverse, (3) both, and (4) neither. Arguably, the first two cases are easier to address. But in many applications, the latter two are important and, for these, one may wish to qualify the answer with a confidence measure. The challenge may be exacerbated when one of the permutations is contained in the other or when they have drastically different sizes.

Next, consider that we typically do not have a clean description of these two populations. Instead, we have a set of examples of each. Hence, our solution must somehow derive a model of each population from the corresponding training set of examples. In some applications, the problem may be exacerbated by random noise, which may be present both in the training set and in the query.

Finally, consider that the features used to discriminate between permutation types may vary from one application to another. In some applications, absolute order may be important. In others, adjacency or proximity between pairs or groups of symbols may be important.

Hence, we may seek a solution which, based on empirical evidence, appears to be useful for a particular application. But validating it may require having a ground truth benchmark. Or, we may seek a generic solution that is grounded on a clear formulation involving probabilities or statistics and try to argue that it may be useful for a broad class of applications.

1.2.2. Contributions

In this paper, we propose solutions that are designed to work for the four situations described above, including those where the query permutation belongs to both populations or to neither of them. We report tests on synthetic benchmarks with different degree of noise (random perturbations). We consider two versions of the problem: one for which the absolute order of the symbols is the key discriminator and the other one for which a hierarchy of groups of symbols or groups that always appear in a contiguous sequence (some always in the same order, other not) is the key discriminator and where, in a group, the order may be partial. Finally, we propose both a solution grounded in a probabilistic formulation and heuristics that, according to our experimental results, appear to better match our intuition.

1.2.3. Novelty

A fair amount of prior art has been focused on formulating and computing distances between permutations and on constructing abstract models that summarize or represent a population and may be computed from examples in a given training set. Our solutions are built upon these pioneering results. Specifically, we propose: (1) a new variant of the Kendall tau distance between permutations, (2) a novel statistical model of the central tendency and variability of permutations in a training set, (3) a novel classifier based on the above, (4) a formulation of the probability that a permutation is part of a population that is implicitly defined by a set of example permutations, (5) a novel classifier based on the above, (6) a heuristic model of the distance between two permutation sets that is based on statistics of position differences between all pairs of symbols, and (7) a novel classifier based on the above.

1.3. Two versions of the problem and outline of our solutions

In this paper, we discuss two versions of of the permutation classification problem: ranking and grouping. The solutions proposed here for each version may be considered orthogonal to each other (you may choose to use one or the other, depending on the application), or complementary (you may choose to use both and combine their results).

1.3.1. Ranking problem

The *ranking* version of the problem may be illustrated by the following simple example. To train our classifier, we ask men and women to rank a set of n movies. Then, given a new candidate ranking, $\mathbf{q}$, which is a permutation of some nominal list of the movies, we guess the gender of its author. We propose a solution that is formulated in terms of precedence relations (How often was movie A ranked before movie B?), as for instance defined by ranking aggregation formulations (7). We call our solution the *Separation Average Distance* (SAD).

1.3.2. Grouping problem

The *grouping* version of this problem may be illustrated by the following example. To train our classifier, we consider two tasks, t_1 or t_2, that each may be accomplished by executing the same set of different actions, but possibly in a different order. We ask a few experts to order these actions for task t_1 and different experts to order them for task t_2. We obtain two sets, T_1 and T_2, of permutations on some nominal task order. These experts may have organized actions carefully into groups (time-contiguous sequences, also called intervals), for example, to reduce the blocking of sparse resources, and may have ordered the actions in some of the groups to satisfy precedence constraints. They may even have conceived their own hierarchy of groups that each comprise a contiguous sequence of three or more sub-groups. Because of precedence constraints, the actions or sub-groups in some groups may have always to be executed in the same order. However, we only record the order in which they have arranged the actions. We do not have access to their groups or hierarchy. Then, given a new candidate sequence (permutation), $\mathbf{q}$, of these actions, we guess whether $\mathbf{q}$ was designed to accomplish task t_1 or t_2. We propose two solutions to the grouping problem.

1.3.3. PAR solution to the grouping problem

Our *Population Augmentation Ratio* (PAR) solution computes a PQ-tree for each training set. The PQ-tree, which was proposed in 1976 (4), defines a population: The set of all different permutations that can be generated by that PQ-tree. PAR builds upon a previously proposed measure of distance, which we use to define the distance between a candidate permutation $\mathbf{q}$ and a PQ-tree. It takes into account not only how many PQ-tree rules are violated by the candidate permutation $\mathbf{q}$, but also a measure of the probability that $\mathbf{q}$ belongs to the population of each one of the two PQ-trees computed for the two training sets. We define this probability in terms of relative change in population size resulting from adding $\mathbf{q}$ to each given training set. This relative change is small when the absolute population count is large. This bias may be viewed as a drawback of the PAR solution.

1.3.4. DoP solution to the grouping problem

Our *Difference of Positions* (DoP) solution is computationally less expensive than PAR and is independent of the absolute population counts. Although DoP does not have the simple statistical grounding of PAR, our experiments show that it is consistently effective. DoP computes, for the candidate $\mathbf{q}$ and for the two training sets, the $n \times n$ matrix ($\mathrm{M}_{\mathrm{T}_1}$,$\mathrm{M}_{T_2}, \mathrm{M}_q$) of the average differences in position for each pair of symbols. The classification of $\mathbf{q}$ is based the weighted sums of the absolute value of the elements in $\mathrm{M}_{\mathrm{T}_1} - \mathrm{M}_q$ and of the elements in $\mathrm{M}_{\mathrm{T}_2} - \mathrm{M}_q$. The weights are the same for the two sums and are proportional to the overlap of the two distributions of the corresponding position differences used to compute $\mathrm{M}_{\mathrm{T}_1}$ and $\mathrm{M}_{\mathrm{T}_2}$.

2. Ranking Problem

The precedence relation plays a significant role in applications involving rankings. One popular measure of distance of two permutations based on precedence is the Kendall tau distance (KD). In the following, after a brief review of the literature on precedence-based distance measures, we describe a variant of the Kendall tau distance and its use for solving the ranking problem.

2.1. Prior art on distances between permutations

The Kendall tau distance counts the number of inversions occurring between two permutations $\mathbf{p}$ and $\mathbf{q}$, i.e. the number of pairs of symbols that appear in opposite order. Formally,

$$KD = |\{(x, y) : (\mathbf{p}^{-1}[x] < \mathbf{p}^{-1}[y]) \neq (\mathbf{q}^{-1}[x] < \mathbf{q}^{-1}[y])\}|$$

Interestingly, when one of the two permutations is the identity, $abc\cdots$, this measure corresponds to the number of swaps of pairs of adjacent symbols needed to sort the other permutation (13) and, as such, the KD is also referred to as the bubble-sort distance. Relations of the Kendall tau with other distances, such as Spearman footrule and Caley distance, have been extensively studied and are reviewed in (14).

In the context of rank aggregation, where the term *ranking* is used instead of *permutation*, the Kendall tau distance has been used to define a *consensus* ranking for a given set of rankings. In other words: to find a permutation that minimizes the sum of the distances from all permutations of the set (10; 7). Such minimization approach produces a result in accordance with the Concorced criterion (10): For a given pair of symbols (a, b), if a precedes b in the majority of rankings, then a should precede b in the consensus ranking. Finding a consensus ranking that minimizes the Kendall tau distance is known to be NP-hard (1; 7).

2.2. Proposed classification based on precedence relations

Given a training set T of permutations on n symbols, the precedence relation may be represented in terms of an $n \times n$ *separation average distance matrix* ($SADiM_T$). An entry of the matrix corresponds to a pair of symbols x and y and gives the ratio of the number of times symbol x is preferred over y in the set of permutations over the total number of permutations.

The *separation average matrix* ($SADiM_{\mathbf{q}}$) for a query permutation $\mathbf{q}$ is similarly defined: it is an $n \times n$ binary matrix, in which a coefficient value of 1 means that x precedes y in the $\mathbf{q}$ and a 0 means the reverse.

The *separation average distance* (SAD) of $\mathbf{q}$ from the set T is defined by the following formula:

$$SAD(\mathbf{q}, \mathrm{T}) = \sum_{x,y} (SADiM_T[x, y] - SADiM_q[x, y])^2$$

An interesting property of the separation average distance matrix is that it has a simple relation with the *Borda* score. The Borda score assigns a score of 0 to the last symbol of a permutation, a score of 1 to the second last and so on. The total Borda score of a symbol over a set of permutations is computed over all permutations of the set. It is easy to show that the sum of the entries of row x of the separation average distance matrix $SADiM_T$ is equal to the Borda score of x on T.

Our classification based on SAD is defined as follows. If T_1 and T_2 are two training sets, the query $\mathbf{q}$ will be classified as follows:

$$\begin{cases} \text{type}\,\mathrm{t}_1 & \text{if } SAD(\mathbf{q}, \mathrm{T}_1) < SAD(\mathbf{q}, \mathrm{T}_2) \\ \text{type}\,\mathrm{t}_2 & \text{otherwise.} \end{cases} \tag{1}$$

3. Grouping problem: Population Augmentation Ratio (PAR)

The ranking version of the problem, discussed above, may be relevant to applications for which the absolute order of the symbols is of primary importance. In this section, we focus on the grouping version, for which adjacency and partial, relative order is important.

3.1. Groups

We start by defining a few terms. A *population* is a set of permutations of a given number, n, of symbols. We consider a set T of permutations that are assumed to be examples of some population $\mathbb{P}(\mathrm{T})$. To define $\mathbb{P}(\mathrm{T})$, we define a hierarchy of groups. A *group* of T is a set of symbols that appear as a contiguous group in all permutations in T. Note that, while contiguous, the symbols of a group may be listed in different orders in the various permutations in T. Clearly, the set of all n symbols is a group, which we call the *full group*. There is unique decomposition of each group into its *children*, i.e., subgroups, which are each either a symbol or a group, and so recursively.

We distinguish three types of groups.

Ordered: The children of an *O-group* appear always in the same order in all the permutations of T. We could decide to consider that each symbol is an O-group.

Reversible: The children of an *R-group* appear are present in only and exactly two different orders, one being the reverse of the other, in the permutations of T. An R-group has at least 2 elements.

Arbitrary: An *A-group* is a group that is neither ordered nor reversible. An A-group has at least three elements.

3.2. ORA-tree

The hierarchical decomposition of the full group into O-, R-, and A-groups may be represented by an *ORA-tree*, which has symbols as leaves and groups as internal nodes. The root of the tree is the full group.

The children of an O-node, which represents an O-group, are listed in the order in which they appear in all permutations of T. However, the children of R-nodes may be listed in one or the other valid orders and the children of an A-nodes may be listed in any order (even if that particular order is not used in any of the permutations in T).

Below, we explain how to compute the ORA-tree, ORA(T), of any set T of permutations.

Note that, given an ORA-tree, one can always compute a *summary set*, T', of only three permutations that yield the same ORA-tree. The justification of this claim and the key idea of this computation is that an A-group, such as 'abc' that has three children (symbols, not groups) should not be confused with an R-group or with an A-group that has non-symbol children. This is not possible if T' contains only two permutations: For example, listing in T' only '*abc*' and '*cba*' would be interpreted as an R-group with 3 children and listing only '*abc*' and '*bac*' would be interpreted as an O-group with two children, one of which, '*ab*', is an R-group. But listing '*abc*', '*cba*', and '*bca*' can only be interpreted as an A-group. From this example, we see that three permutations may be needed for T'. One can show that three are sufficient, because A-groups with more than three elements can be unambiguously captured in T' using only two summary permutations. For example, listing '*abcd*' and '*cadb*' defines an A-group.

A candidate permutation **p** is *valid* (with respect to T) if it respects the contiguity and order constraints of each node of the ORA-tree derived from T. We define the *population*, $\mathbb{P}(\mathrm{T})$, of T as the set of all valid permutations. Hence, when T' is the summary set of T, $\mathbb{P}(\mathrm{T}') = \mathbb{P}(\mathrm{T})$.

3.3. Incremental computation of the ORA-tree

The first permutation **p** yields an ORA-tree with a single O-node root that represents the full group. This is the most constrained version of T possible as it implies that the $\mathbb{P}(\mathrm{T})$ contains only **p**.

Then, for each consecutive permutation **p** in T, we perform a recursive (post-order) tree traversal and decide, for each node, N, of the current tree, whether **p** breaks any rule (ordering or grouping) encoded by N.

If grouping and ordering are respected by **p**, then all the children of N represent *valid* groups and we leave N untouched.

If all children report that their grouping is respected, but the ordering of the children is not respected by **p**, we re-label N as either an R-node or an A-node (an R-nodes is appropriate if N used to be an R-node or when it used to be an O-node and the elements in N appear in the exact opposite order in **p**).

If at least one child of N reports that its grouping is not respected, we attempt to "repair" the problem by rearranging the groups in N. If this is impossible, we report the incorrect grouping problem to the parent of N, which will either resolve the problem, or pass it on to its parent, and so on.

An implementation of this algorithm is provided in (12).

3.4. PAR measure and classification

Assume that we are given two training sets, T_1 and T_2, and have computed their ORA-tree. We are given a candidate permutation **q**. We want to assess whether **q** is more likely to be of type t_1 or t_2.

We could do so by counting, for each ORA-tree, how many constraints are violated (not satisfied) by **q**. **q** violates the constraint associated with a node N of the tree if the children of N are not each corresponding to a contiguous interval of symbols in **q** or if the order in which these intervals appear in **q** does not correspond to the orders defined by the node-type of

N. We could also consider giving different weights to the node, depending on their type and size.

We propose to do the above by considering the relative growths of the population size that results from adding $\mathbf{q}$ to T_1 and to T_2. This formulation provides a simple and unifying measure that takes into account all the violations suggested above and gives them a weight that is proportional to their impact on population growth.

Let $|\mathbb{P}|$ denote the size (i.e., permutation count) of population $\mathbb{P}$. Also, let $\mathrm{T}+\mathbf{q}$ denote the set of permutations that is the union of $\mathbf{q}$ and of all permutations in T.

We propose to measure the "distance" between $\mathbf{q}$ and T using a new measure, which we call the *Population Augmentation Ratio* (PAR). It is defined by $\log(|\mathbb{P}(\mathrm{T}+\mathbf{q})|/|\mathbb{P}(\mathrm{T})|)$. Note that this distance is zero when $\mathbf{q}$ is in $\mathbb{P}(\mathrm{T})$ and that its maximal value is $\log(n!/|\mathbb{P}(\mathrm{T}))|$.

As an example, consider a set T of permutations on n symbols with four non-overlapping groups of length 5, 4, 3, and 2. A query permutation can share any combination of such groups with T or no group at all. Figure 1 lists all such combinations and the corresponding PAR values for n = 100. The rows are sorted by increasing values of PAR.

We classify $\mathbf{q}$ as being of type t_1 when $\log(|\mathbb{P}(\mathrm{T}_1+\mathbf{q})|/|\mathbb{P}(\mathrm{T}_1)|) < \log(|\mathbb{P}(\mathrm{T}_2+\mathbf{q})|/|\mathbb{P}(\mathrm{T}_2)|)$. Note that, in some cases, $\mathbf{q}$ may agree with more groups of T_2 than of T_1, and still be classified as being of type t_1.

3.5. Practical implementation of the PAR classification

Except for toy cases where n is small, we cannot use floats or doubles to compute $|\mathbb{P}(\mathrm{T}+\mathbf{q})|/|\mathbb{P}(\mathrm{T})|$ with sufficient accuracy to support precise classification. Hence, we use the following approach. Note that $\log(|\mathbb{P}(\mathrm{T}+\mathbf{q})|/|\mathbb{P}(\mathrm{T})|)$ can be written as $\log(|\mathbb{P}(\mathrm{T}+\mathbf{q})|) - \log(|\mathbb{P}(\mathrm{T}_1)|)$. Therefore, our classification test can be rearranged into the following form: we classify $\mathbf{q}$ as being of type t_1 when

$$\log(|\mathbb{P}(\mathrm{T}_1+\mathbf{q})||\mathbb{P}(\mathrm{T}_2)|) < \log(|\mathbb{P}(\mathrm{T}_2+\mathbf{q})||\mathbb{P}(\mathrm{T}_1)|). \qquad (2)$$

Now, notice that $\mathbb{P}(\mathrm{T}_1+\mathbf{q})\mathbb{P}(\mathrm{T}_2)$ is the product, $\prod_{k=2}^{n} k^{E[k]}$, of non-negative integers k for k in $[2,n]$, each raised to an integer power, which we denote by E[k].

Hence, $\log(|\mathbb{P}(T_1+\mathbf{q})||\mathbb{P}(\mathrm{T}_2))|$ is the sum, $\sum_{k=2}^{n} E[k]log(k)$. Therefore, our strategy is to compute the entries $E_L[k]$ for the left-side of the inequality (2) and the entries $E_R[k]$ for the right-side.

To compute the table $E_L[]$ of exponents for the left side of the inequality (2), which corresponds to $|\mathbb{P}(\mathrm{T}_1 + \mathbf{q})||\mathbb{P}(\mathrm{T}_2))|$, we first initialize its elements to 0. Then, we traverse the trees of $\mathrm{T}_1 + \mathbf{q}$ and of T_2 via a standard recursive traversal. At an O-node, we do not change any exponent. At an R-node, we increment the exponent $E_L[2]$, because there are two possibility for ordering such a node. At an A-node of m children, we increment the exponent $E_L[k]$, for all values of k in $[2, m]$.

To compute the table $E_R[]$ of exponents for $|\mathbb{P}(\mathrm{T}_2{+}\mathbf{q})||\mathbb{P}(\mathrm{T}_1))|$, we proceed similarly.

The ORA-trees for $\mathrm{T}_1 + \mathbf{q}$ and for $\mathrm{T}_2 + \mathbf{q}$ are computed by using one iteration of the incremental computation of an ORA-tree, we described above.

We simplify both sides of the inequality (2). To do so, we reduce corresponding exponents on both sides by the same amount so that, for any k, $log(k)$ appears only on one side. For example, $5\log(3) + 4\log(4) < 8\log(3) + 2\log(4)$ is simplified to $2\log(4) < 3\log(3)$.

Finally, we evaluate the left and right expressions and compare them.

An interesting alternative for using a non-incremental construction of the ORA-trees of $\mathrm{T} + \mathbf{q}$ for various candidate permutations $\mathbf{q}$ is to (1) compute the ORA-tree of T, (2) compute the 3 permutations of the summary set T' of that tree, and (3), construct the ORA-tree of the 4 permutations in $\mathrm{T}' + \mathbf{q}$.

3.6. Extension to more than two training sets

A possible extension of the PAR solution proposed here is to consider several training sets, T_i. To estimate the type, t_i, of a candidate permutation, $\mathbf{q}$, one could identify the smallest of the distances $\log(|\mathbb{P}(\mathrm{T}_i + \mathbf{q})|/|\mathbb{P}(\mathrm{T}_i)|)$.

3.7. Prior Art

In this subsection, we briefly review closesly related prior art.

3.7.1. PQ-trees

The ORA-tree is a simple extension of the previously proposed PQ-tree (4), in which A-nodes are called *P-nodes* and in which O-nodes and R-nodes are not distinguished and both called *Q-nodes*. Efficient algorithms for finding P- and Q-groups have been proposed in (2; 6; 8; 16). The PQ-tree may be trivially obtained from an ORA-tree, but doing so may correspond to a loss of information (1 bit per Q-node) and may increase the corresponding population significantly. The latter observation has implications one the PAR distance and classification test proposed above.

The original PQ-tree paper (4) includes an incremental construction in which contiguity constraints (groups) are added one-at-a-time. Hence, it cannot be used to provide the incremental construction, one-permutation-at-a-time, proposed above.

Applications of PQ-trees in computational biology have been considered in (11). The PC-tree variant of a PQ-tree, in which Q-nodes corresponds to reversible (clockwise or counterclockwise) cyclic orderings of children have been proposed in (15) to simplify the testing of the planarity of undirected graphs.

3.7.2. Adjacency distance

The adjacency distance is perhaps the simplest measure that takes into account the proximity of symbols. Given two permutations **p** and **q**, it is based on the adjacency relation between symbols and counts the number of times a pair of symbols is adjacent in one permutation, but not in the other. It is defined as:

$$AD = n - 1 - adj_{\mathbf{p},\mathbf{q}}$$

where n is the size of the permutations and $adj_{p,q}$ is the number of pairs that are adjacent in both **p** and **q** (13). AD is an integer between 0 and n-1.

3.7.3. Interval distance

The interval distance between two sets of permutations, was proposed in (2). Basically, the interval distance counts the number of groups, in the paper referred to as common intervals, that are lost in each of the two sets when the sets are merged. More precisely, let T_1 and T_2 be two sets of permutations on n symbols, with c_1 and c_2 common intervals, respectively. Let c be the number of common intervals shared the permutations of the union $T_1 \cup T_2$ of the two sets, the interval distance between T_1 and T_2 is defined as:

$$IN(T_1, T_2) = c_1 + c_2 - 2c.$$

Although this measure is relatively easy to compute, it has limitations; for instance, it does not take into account the length of common intervals, which appears to be an important factor in establishing dissimilarity of the sets.

4. Grouping problem: Difference of Positions (DoP)

Here we focus on the classification problem when noise is present in the training sets. In such cases the measure PAR is not applicable since a few transpositions of the symbols may completely alter the structure of the tree. To address this problem, we start pre-processing the training sets T_1 and T_2 and the query $\mathbf{q}$ to derive their representation in terms of the $n \times n$ matrices, M_{T_1}, M_{T_2}, and M_q, the difference of positions matrices. An entry in each matrix corresponds to a pair of symbols $[x, y]$ and its value depends on the relative positions of symbols x and y in the permutations of the set. We assign a weight to each entry of the two matrices M_{T_1} M_{T_2} that reflects the relevance of that pair in discriminating between T_1 and T_2. The weight of an entry is the same for $\mathrm{M}_{\mathrm{T}_1}$ and $\mathrm{M}_{\mathrm{T}_2}$. The classification of $\mathbf{q}$ is based the weighted sums of the absolute value of the elements in $\mathrm{M}_{T_1} - \mathrm{M}_q$ and of the elements in $\mathrm{M}_{T_2} - \mathrm{M}_q$.

4.1. Difference of Positions Matrix

The *difference of positions matrix* (DoP), M_T, associated with the set T of m permutations on n symbols is an $n \times n$ matrix whose elements $M_T[x, y]$ are the average over all permutations $\mathbf{p} \in \mathrm{T}$ of the absolute value of difference in position of x and y in T, that is:

$$\mathrm{M}_{\mathrm{T}}[x, y] = 1/m \sum_{\mathbf{p} \in \mathrm{T}} |\mathbf{p}^{-1}[x] - \mathbf{p}^{-1}[y]| \qquad (3)$$

Example: Let $\mathrm{T} = \{\mathrm{abcd}, \mathrm{acbd}, \mathrm{bcad}\}$ contain $m = 3$ permutations on $n = 4$ symbols. The DoP matrix of the set T is:

	a	b	c	d
a	0	5/3	4/3	7/3
b	5/3	0	1	2
c	4/3	1	0	5/3
d	7/3	2	5/3	0

where, for instance, $\mathrm{M}_{\mathrm{T}}[a, b] = (1 + 2 + 2)/3$ is the sum of the absolute distances of a from b in all three permutations divided by the number of permutations.

An interesting property of the matrix is that the sum of its elements depends only on the number n of symbols of the permutations, but is independent of the number of permutations of T, as well as of the specific permutations of the set. More precisely, in the Appendix A we prove that the average of the non-zero (off diagonal) elements of the difference of positions matrix M_T is equal to $(n+1)/3$, i.e.

$$1/n(n-1)\sum_{x,y} \mathrm{M}[x,y] = (n+1)/3 \tag{4}$$

4.2. Classification using the DoP *distance*

We now introduce the *Difference of Positions* (DoP) distance between the query permutation $\mathbf{q}$ and each of the two training sets based on the differences of positions matrices $\mathrm{M}_{\mathrm{T}_1}$ and $\mathrm{M}_{\mathrm{T}_2}$ and M_{q}. Crucial to the definition of this measure is the way we assign a weight to each entry-pair, $\mathrm{M}_{\mathrm{T}_1}[\mathrm{x},\mathrm{y}]$ and $\mathrm{M}_{\mathrm{T}_2}[x,y]$. This common weight, denoted by $\mathrm{W}[x,y]$, is computed from the two statistics on the values $|\mathbf{p}^{-1}[x]-\mathbf{p}^{-1}[y]|$ in permutations of T_1 and in permutations of T_2. The graph of the distribution of values $|\mathbf{p}^{-1}[x]-\mathbf{p}^{-1}[y]|$ determines a curve. Assuming normal distributions, we compute the areas under the curve of the two distributions of such values. They are denoted by a_1 and a_2, while the overlap area of the two distributions is denoted by o (see Fig. 2). The weight is then defined as:

$$\mathrm{W}[x,y] = 1 - o/(a1 + a2 - o) \tag{5}$$

This weight is a measure of overlap of two distributions and is similar to other statistical measures of similarity of discrete and continuous distributions, such as the Bhattacharyya distance, the Hillerson distance, and the overlapping coefficient (3; 5; 9).

Given the query $\mathbf{q}$, we define the difference of positions distances $\mathrm{DoP}(\mathbf{q},\mathrm{T}_1)$ and $\mathrm{DoP}(\mathbf{q},\mathrm{T}_2)$ of $\mathbf{q}$ from T_1 and T_2, respectively, as weighted differences of the matrices.

$$DoP(\mathbf{q},\mathrm{T}_1) = \sum_{x<y} W[x,y]\,|M_q[x,y] - M_{T_1}[x,y]| \tag{6}$$

$$DoP(\mathbf{q},\mathrm{T}_2) = \sum_{x<y} W[x,y]\,|M_q[x,y] - M_{T_2}[x,y]| \tag{7}$$

Finally, $\mathbf{q}$ will be classified as belonging to T_1 if

$$\mathrm{DoP}(\mathbf{q}, \mathrm{T}_1) < \mathrm{DoP}(\mathbf{q}, \mathrm{T}_2)$$

otherwise to T_2.

4.3. Results on synthetic test sets using the distance DoP

To evaluate the performance of the proposed classification method, we run experiments on synthetic data using the DoP distance. The goal was to assess the ability of the method to make correct decisions when the permutations of the training sets shared proximity of some symbols. We generated sets of permutations with well-defined groups and computed the accuracy of the classification on large sets of queries as the fraction of correct predictions over all queries.

Given two training sets T_1 and T_2, we generated two sets of queries: Q_1 and Q_2 such that $\mathrm{T}_1 + \mathrm{Q}_1$ and $\mathrm{T}_2 + \mathrm{Q}_2$ have the same PQ-trees as T_1 and T_2, respectively. Next, to assess the robustness of the model we introduced chaos in the permutations of Q_1 and Q_2 and determined the loss in accuracy as a function of the amount of chaos. Chaos was obtained by applying transposition operations, that is by changing the order of the elements in a query. In a transposition, two randomly selected symbols x and y were swapped. The symbols need not belong to a group. We remark here that for a query generated by injecting chaos in a permutation of Q_1 (Q_2), we consider correct the classification that places it in the set T_1 (T_2). Finally, we added noise by including, in the training sets, permutations with symbols in random orders.

We report here the results obtained with sets T_1 and T_2 of permutations on the 26 characters of the alphabet and of sizes 100, 1,000 and 10,000. The permutations of set T_1 have k groups, while those of T_2 have h groups, where k and h are random number in the range [1-5]. The size of the groups are randomly assigned in the range [2-6]. Each group has probability p of having a nested group inside it. Each set of queries Q_1 and Q_2 consists of 500 permutations. In figure 3 we report the accuracy of the classification results for values of the parameter chaos equal to 0, 2, 4, and 6; the parameter noise varies from 0% up to 70%. The results of the prediction on the sets Q_1 and Q_2 (the queries are permutations in the population of the PQ-tree of either T_1 and T_2 and no noise nor chaos is present) are very accurate with an accuracy very close to 100 % (first row of the table).

A good performance of the classifier is also observed when the noise is below 50%, that is at least half the permutations in each training set have the same PQ-tree. More precisely, the accuracy is above 95% when no chaos is present and above 78% otherwise. When the size of the training sets is large (10,000) even at high rate of noise (70%) above 97% of the queries are correctly classified.

The computation of the proposed distances is fast. The classification approach has been implemented in Python and run of a standard PC (Intel Core i7-7700HK @ 2.8GHz). The running times in seconds for the classification of 10,000 queries permutations of size 26 are reported in Table 1 that separates the pre-processing time to build the DoP matrix from the classification time.

	N. of permutations in T_1 and T_2		
	100	1,000	10,000
Preprocessing	0.17	0.4	2.5
Classification (10,000 queries)	2.2	2.2	2.2

Table 1: Running times for the classification of 10,000 queries

As a last note, the DoP distance is able to identify patterns of not necessarily consecutive elements that occur at approximately the same distance from each other, as such it may be viewed as an extension of these group-based recipes and hence captures information that is not captured in the PQ-tree.

5. Conclusions and summary

We proposed measures of distance of permutations for the binary classification of query permutations. We considered different criteria for the design of such measures: one that emphasizes the importance of the precedence and the other of the contiguity of symbols. Another criterion is whether the distances are exact or approximate and therefore whether they are relatively immune to noise in the data.

The foundation of our exact measure PAR is the data structure PQ-tree. Our DoP distance is also based on groups and is relatively insensitive to the presence of noise in the data. Orthogonal to these three measures is the SAP distance that builds upon the notion of precedence and Kendall tau distance. On synthetic data generated with well-defined patterns of groups

SAP provides a classification that does not reflect in any way the presence of groups (results not shown in the paper).

An issue that arises in applications involving rankings is that of incomplete permutations, i.e. permutations with missing symbols. For instance, in video application viewers may rank different sets of movies. In biological applications, such are classification of genomes based on the order of genes in different organisms, not all genes may be present in all organisms. We address the issue of incompleteness in a companion paper (17) where we extend our proposed distance DoP to deal with such cases.

6. Acknowledgements

C. Guerra and X. Zhao were in part supported by the United States - Israel Binational Science Foundation (BSF) grant 2014028.

7. References

[1] J. Barthold III, C.A. Tovey, M.A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare, 6, 2 (1989) 157165.

[2] A. Bergeron, J. Stoye, On the similarity of sets of permutations and its applications to genome comparison. J Comput Biol. 3, 7 (2006)1340-5.

[3] A. Bhattacharyya, On a measure of divergence between two multinomial populations. The Indian, Journal of Statistics (1933-1960), 7, 4 (1946) 401406.

[4] K.S. Booth G.S. Lueker, Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. 13 (1976) 335379

[5] E. Choi, C. Lee, Feature extraction based on the Bhattacharyya distance. Pattern Recognition, 36, 8 (2003) 17031709.

[6] D. Doerr, J. Stoye, S. Bfocker, K. Jahn, Identifying gene clusters by discovering common intervals in indeterminate strings. BMC Bioinformatics, 15 (Suppl. 6):S2 (2014).

[7] C. Dwork, et al. Rank aggregation methods for the web. Proceedings of the 10th international conference on World Wide Web. ACM (2001).

[8] S. Heber, R. Mayr, J. Stoye, Common Intervals of Multiple Permutations, Algorithmica, 60, 2 (2011) 175206, DOI 10.1007/s00453-009-9332-1.

[9] H.F. Inman, E.L. Bradley, Jr, The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities. Communications in Statistics - Theory and Methods, 18 (1989) 10.

[10] J. Kemeny, Mathematics without numbers, (1959), Daedalus 88, 577591.

[11] G.M. Landau, L. Parida, O. Weimann Gene Proximity Analysis across Whole Genomes via PQ Trees. J. Computational Biology, 12, 10 (2005) 12891306.

[12] L. Rossignac-Milon, LeoRoss/PQ Tree. GitHub, (2018). https://github.com/leoRoss/PQTree

[13] C.R. Reeves, Landscapes, operators and heuristic search. Annals of Operations Research, 86 (1999) 473-490.

[14] T. Schiavinotto, T. Sttzle, A review of metrics on permutations for search landscape analysis. Computers operations research 34, 10 (2007) 3143-3153.

[15] W.-K. Shih, W.-L. Hsu, A new planarity test. Theoretical Computer Science. 223 (1-2): 179191 (1999). doi:10.1016/S0304-3975(98)00120-0.

[16] T. Uno, M. Yagiura, Fast algorithms to enumerate all common intervals of two permutations. Algorithmica, 26 (2000) 290309.

[17] X. Zhou, A. Amir, C. Guerra, G. Landau, J. Rossignac, EDoP distance between sets of incomplete permutations: Application to bacteria classification based on gene order (2018) (submitted).

8. Appendix A

Properties of the DoP *Matrix*

The *difference of positions* matrix is clearly symmetric and its diagonal elements are zero. We prove the following:

Given a set T of m permutations on n symbols, the average of the non-zero (off diagonal) elements of the difference of positions matrix M_T is equal to $(n+1)/3$, i.e.

$$1/n(n-1)\sum_{x,y} \mathrm{M}[x,y] = (n+1)3 \tag{8}$$

Proof. We first show that the sum in (1) is independent of the permutations in T and of m. We show that by showing that every permutation contributes the same amount to the $\sum\limits_{x,y} M[x,y]$, therefore the average does not change when permutations are added/removed from T. More precisely, we show that if $\mathbf{p}$ is any permutation on n symbols (thus $m=1$) then

$$\sum_{x,y} \mathrm{M}[x,y] = 2\sum_{j=0}^{n-1}\sum_{j=0}^{j} i \tag{9}$$

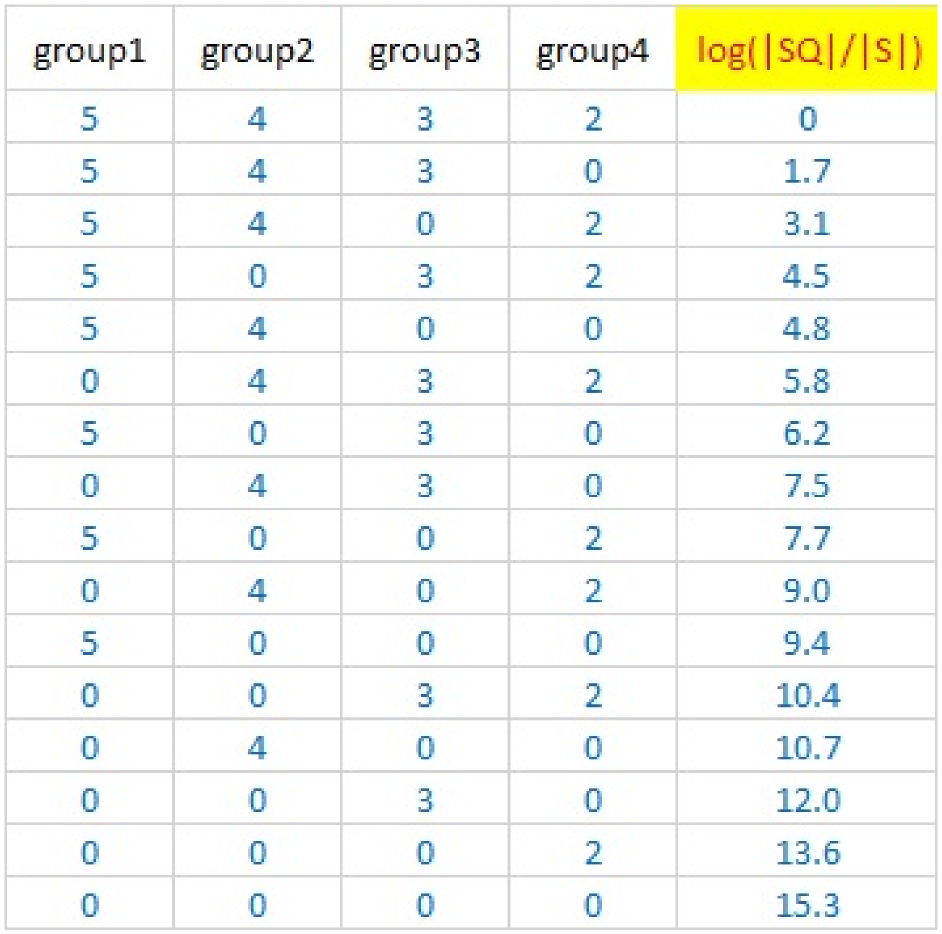
In fact, let p $= x_1x_2\cdots x_n$, then the sum of the absolute values of the distance of symbol x_i to all other symbols is given in the table below:

$$\begin{gathered}
x_1 : 1+2+\cdots+n-1 \\
\mathrm{x}_2 : 1+1+2+\cdots+n-2 \\
\mathrm{x}_3 : 1+2+1+2+\cdots+n-3 \\
\mathrm{x}_4 : 1+2+3+1+2+\cdots+n-4 \\
\cdots \\
\mathrm{x}_{n-1} : 1+2+\cdots+n-2+1 \\
\mathrm{x}_n : 1+2+\cdots+n-1
\end{gathered}$$

where the values in red in each row represent the distances of x_i to the symbols to its right in $\mathbf{p}$; black values represent the distances to the symbols to the left of x_i. It is easy to see (because of symmetry) that the sum over all x_i of black values is the same as that of red values; furthermore, each sum is equal to $\sum\limits_{j=0}^{n-1}\sum\limits_{i=0}^{j} i$. This proves equality (9). The black sum (and the red sum) is a well-known sum referred to as the $(n-1)^{th}$ *tetrahedral number*, Tet_{n-1}, because of the way the integers can be geometrically arranged in a triangular tetrahedron. It is given by:

$$Tet_{n-1} = 1/6(n-1)n(n+1) \tag{10}$$

From (9) and (10) the equality 8 is immediately derived. The difference of position matrix associated to the query **q**, is well characterized. It contains the same set of values which are independent of **q**, although assigned to different entries of the matrix for different queries. More precisely, the matrix has $n-1$ entries with value 1, $n-2$ entries with value 2 and so on.



group1	group2	group3	group4	log(|SQ|/|S|)
5	4	3	2	0
5	4	3	0	1.7
5	4	0	2	3.1
5	0	3	2	4.5
5	4	0	0	4.8
0	4	3	2	5.8
5	0	3	0	6.2
0	4	3	0	7.5
5	0	0	2	7.7
0	4	0	2	9.0
5	0	0	0	9.4
0	0	3	2	10.4
0	4	0	0	10.7
0	0	3	0	12.0
0	0	0	2	13.6
0	0	0	0	15.3

Figure 1: The PAR distance values for all combinations of queries from a set T of permutations on 26 symbols

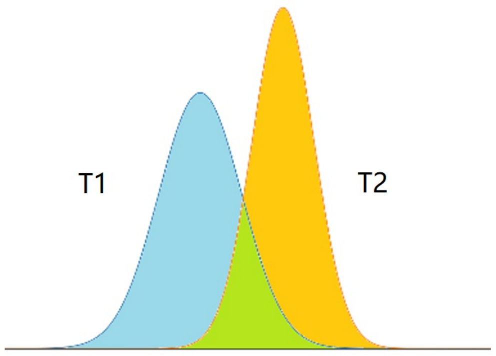
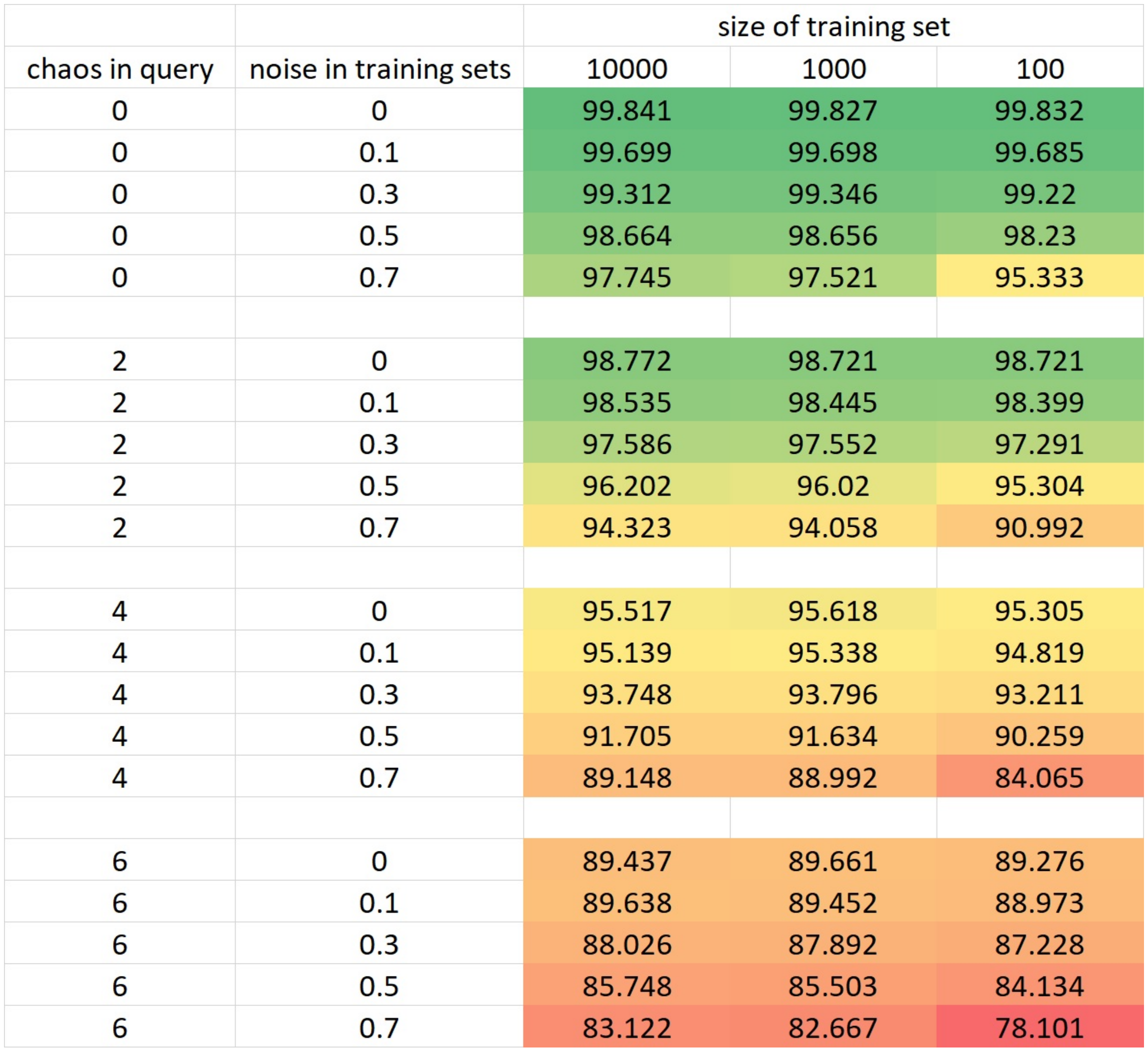


Figure 2: Distribution of $|\mathbf{p}^{-1}[x] - \mathbf{p}^{-1}[y]|$ in two training sets. The two normal curves representing the distribution of the values $|\mathbf{p}^{-1}[x] - \mathbf{p}^{-1}[y]|$ in $\mathbf{T_1}$ and T_2 are plotted in a reference system where the values $|\mathbf{p}^{-1}[x] - \mathbf{p}^{-1}[y]|$ are in increasing order on the x axis while their frequencies are on the y axis. The area a_1 is that of the blue and green regions, while a_2 is the area of the yellow and green regions. The overlap area, denoted by o, is that of the green region.



		size of training set		
chaos in query	noise in training sets	10000	1000	100
0	0	99.841	99.827	99.832
0	0.1	99.699	99.698	99.685
0	0.3	99.312	99.346	99.22
0	0.5	98.664	98.656	98.23
0	0.7	97.745	97.521	95.333
2	0	98.772	98.721	98.721
2	0.1	98.535	98.445	98.399
2	0.3	97.586	97.552	97.291
2	0.5	96.202	96.02	95.304
2	0.7	94.323	94.058	90.992
4	0	95.517	95.618	95.305
4	0.1	95.139	95.338	94.819
4	0.3	93.748	93.796	93.211
4	0.5	91.705	91.634	90.259
4	0.7	89.148	88.992	84.065
6	0	89.437	89.661	89.276
6	0.1	89.638	89.452	88.973
6	0.3	88.026	87.892	87.228
6	0.5	85.748	85.503	84.134
6	0.7	83.122	82.667	78.101

Figure 3: Results on synthetic data for different values of the parameters chaos and noise