Dimensionality reduction and trajectory grouping

Complex Systems research project

Nardi Lam, 6313159

Utrecht University

August 23, 2020

In this study, dimensionality reduction (DR) is investigated in the context of the trajectory grouping problem (TGP), which is the problem of identifying groups among a set of moving entities from their trajectory data. A method of extracting group structure from the output of a DR algorithm is devised, and various DR methods (PCA, ICA, and a custom clustering procedure) are evaluated on a synthetic data set generated using the Kuramoto model. As a comparison baseline the method from Buchin et al. (2013) is used, which is based on spatially connected sets of entities. From the results, it seems that the notions of 'groups' and dimensionality agree, as PCA can be used to predict the number of groups accurately. In predicting the specific group structure, the grouping via DR does not perform well enough to act as a substitute for the method by Buchin et al. However, the application of DR methods to a problem such as this seems to be sensible, and the procedure outlined here may be useful for similar applications of dimensionality reduction.

Contents

1 Introduction

The large size of contemporary datasets makes **dimensionality reduction (DR)** a relevant problem in many different fields. Though state-of-the-art algorithms may be able to deduce complicated statistical relationships from large amounts of data, it is often useful to be able to generate a reduced representation of the data first. The reasons for this may be as follows:

1. When unimportant information is filtered out beforehand, other algorithms may decrease in running time, or algorithms which only function on low-dimensional data may be used.
2. Some operations might be hard to perform on large datasets because of memory restrictions.
3. For transparency: an algorithm converting a large input to a small (e.g. yes/no) output might be very opaque if it is not clear what the relevant information is. Reducing the data set beforehand allows for a clearer interpretation of the inferences on which eventual decisions are based.

However, DR is not a well-defined problem in general. Each DR method will have some type of compressive capability (i.e. the information contained in the data set is represented using less bits than before), but what is a good way to reduce a data set depends to a large degree on the application. The application often determines two things:

1. Which information is considered relevant (the content).
2. What a suitable representation of the information (the form) is.

For this reason, it is most interesting to study DR methods in the context of a specific application. This not only provides new ways to tackle a specific problem, but also produces further evidence as to which methods are useful in what kind of situations, and why. If a certain DR method is well-applicable to a certain problem, it provides practical intuition for the use-cases in which that method is applicable, and how to interpret the results.

The problem studied in this work is the **trajectory grouping problem (TGP)**: given a number of time series containing location data of entities (i.e. trajectories), we would like to identify 'groups', or sets of entities which move 'together' in some sense. From a DR perspective, these groups provide

an efficient representation of the data: by replacing the trajectory for a group of entities with a single representative trajectory, we can reduce the dimensionality of the data set while retaining much of the relevant information.

In this paper, the application of a number of dimensionality reduction methods to synthetic data sets in order to find groups of entities will be studied. This not only provides an alternative view of the trajectory grouping problem, but also answers questions about the concrete geometrical interpretation of these dimensionality reduction methods. First, the TGP will be discussed in detail. Then, the DR methods under consideration and their applicability to the TGP will be discussed on a theoretical basis, and then experimental results of their performance will be presented to support these theoretical hypotheses.

2 The trajectory grouping problem.

In the last decade, the rise of smartphones and other mobile computing devices had led to a large amount of sensor data that is available in many different scenarios. In some cases, this data consists of time-series data which records the location[1] of certain objects. We will refer to these moving objects as **entities**, and the sequences measuring their location as **trajectories**. Examples of this could be people walking through a city, cars driving in traffic, animals moving around a nature reserve or even gene expression data (Phang et al., 2002).

[1] The location is measured in terms of spatial coordinates, so these will usually be sequances of vectors in $\mathbb{R}^2$ or $\mathbb{R}^3$, but the idea works in any metric space.

The **trajectory grouping problem (TGP)** is the problem of finding 'groups' of entities in this type of data (Buchin et al., 2013). By a common sense definition, these are entities whose movement is somehow linked, though the formal definition may vary. The most obvious reason for wanting to find these groups is a practical one: if there is a lot of data, the individual trajectories are by themselves often not so important in the grand scheme of things, or there are simply too much of them to handle in some sensible way. Reducing the set of individuals to a (smaller) set of groups enables further analysis.

There are also many domain-specific uses for trajectory grouping. If the group structure is known, it may be used to efficiently utilize common resources, by for example discovering opportunities for carpooling (Lee and Liang, 2011). Information about the group structure may also be necessary for constructing useful models of behavior. Both humans (Pellegrini et al., 2010) and non-human animals (Couzin, 2009) show interesting behavioral patterns in how they form groups and how these relate to each other. Finally, a purpose for which grouping may be applied is to protect the privacy of individuals by only

retaining 'agglomerate' trajectories rather than people's personal location data (Nergiz et al., 2008).

2.1 The proximity-based definition of a group.

In order to find groups of entities, a formal definition of what constitutes a group needs to be established. For inspiration, the definition from Buchin et al. (2013) can be used. They formulate a group in terms of three parameters: a spatial parameter ϵ, a temporal parameter δ, and a size parameter m. Suppose we have a set of entities E which are located in a metric space M with associated metric d. A **group** G is then defined as a *maximal*[2] set of at least m entities that are all ϵ-*connected*[3] to one another for at least a time period of δ. This definition will henceforth be called the **proximity-based definition of a group**.

To apply this definition of a group to a sample of trajectory data, reasonable values for the parameters need to be chosen. We will be considering data which is assumed to lie on the attractor of some system, and as such the group structure can be assumed constant. Because we presuppose a constant group structure, this means a group only exists if it exists during the entire length of the sample. Therefore we can set $\delta = T$, where T is the length of the sample.

For simplicity we will divide all entities into groups, so that $m = 1$. This means the maximum number of groups is equal to the total number of entities. Then, finding the groups in a data sample amounts to choosing a distance ϵ, and finding ϵ-connected sets of entities that persist over during the entire sample. What is a suitable value for ϵ depends on the specific structure of the data, and will be treated later on.

The procedure to find the group structure in a data sample by proximity will then be as follows:

1. Let M be the metric space in which entities live, $\Theta(t) = (\theta_1(t), ..., \theta_N(t)) \in M^N$ the locations of the N observed entities at time t, and $d : M^2 \to \mathbb{R}^+$ a suitable metric[4] on M.
2. Choose a distance parameter $\epsilon \in \mathbb{R}^+$ which controls how close entities must be for them to be grouped together.
3. Calculate the maximum pairwise distances between all pairs of entities (θ_i, θ_j) and store them in a matrix R such that $R_{ij} = \max_t d\left(\theta_i(t), \theta_j(t)\right)$.
4. Create a graph $\mathcal{G}$ with N vertices, where each vertex corresponds to an entity, and an edge between two vertices exists iff $R_{ij} < \epsilon$.

[2] *Maximal* means that there is no other group H such that either $G \subset H$ or such that the time interval over which G exists is contained in the interval over which H exists.

[3] To be ϵ-*connected* means that between each pair of entities $e, f \in G$ there exists a path $p = (h_1, ..., h_n) \in G^n$ for $n \geq 2$ with $h_1 = e, h_n = f$, such that the distance between each adjacent pair of entities $d(h_i, h_{i+1})$ with $1 \leq i < n$ is at most ϵ.

[4] The usual requirements for a metric are assumed to hold: $d(x, y) = 0$ implies $x = y$, $d(x, y) \geq 0$, d is symmetric, and the triangle inequality.

5. The groups are then given by the set of maximal connected components of $\mathcal{G}$. As such, the graph $\mathcal{G}$ can be taken as a description of the group structure of the data sample.

We will take the **group structure graph** $\mathcal{G}$ to be the object of interest in determining the groups.

When calculating the group structure graph $\mathcal{G}$ using the proximity-based method, the existence of a group directly depends on the maximum inter-entity distance, which is used as a measure of 'relatedness' of two entities. However, we may try to find this group structure using dimensionality reduction instead, in which case the measure of relatedness changes, and as such the interpretation of what it means to be a group changes as well. In the next section, various DR methods will be introduced and it will be discussed how the kind of groups these methods would construct may differ from the proximity-based groups.

2.2 The relationship between grouping and dimensionality reduction.

The central thesis motivating this attempt to perform trajectory grouping via dimensionality reduction is:

A dimensionality-reducing transformation $f : M^N \to M^k$ with $N, k \in \mathbb{N}$ and $k < N$, can sometimes be viewed as combining the initial N variables together into a smaller number of group variables, and therefore performs a *grouping operation*.

Two questions immediately arise here:

1. *When exactly* is it reasonable to view a dimensionality-reducing transformation as a grouping operation?
2. *How similar* is the grouping structure it produces to that of the proximity-based grouping method?

This study aims to tackle these questions, where the first is mostly a theoretical question to do with the interpretation of DR techniques, and the second question is an empirical one which can be studied experimentally.

With regard to the first question, it is important how a given transformation may be interpreted. Generally, dimensionality reduction is achieved by eliminating *redundancies*: when two dimensions (partially) contain the same information, they may be (partially) combined into one. As such, which

dimensions are mapped onto which is determined by evaluating the similarity between pairs of variables. This means that for different measures of similarity, the interpretation of what constitutes a group (i.e. what property 'groups' two dimensions together) will differ, and some measures may lead to more intuitive interpretations than others.

In the proximity-based case, the measure of similarity between entities which is applied is a *maximum connected inter-entity distance*, leading to a binary decision on whether two entities should be grouped or not. As such, the resulting groups are sets of entities which are not too far away from each other, where 'not too far' is specified by the parameter ϵ. In the case of dimensionality reduction methods, this measure of similarity between entities or trajectories will usually be something more abstract, such as a statistical measure. In that case, it is not as evident how to determine whether two entities belong in a group together. Therefore, to compare the results of grouping via DR methods to the proximity-based method, a generalized version of the group extraction procedure is necessary. Here the comparison between entities is made in a generic way, independent of how the actual transformation f is defined. This generalized variant will be described in the following section.

2.3 Generalized grouping via dimensionality reduction.

Suppose we have a trajectory data set containing location data for $N \in \mathbb{N}$ entities, with $T \in \mathbb{N}$ samples each, stored in a data matrix $X \in M^{N\times T}$. If we apply our dimensionality reducing transformation $f : M^N \to M^k$ to each column of X, we will obtain a reduced data matrix $Y \in M^{k\times T}$, where $k < N$. Now, if we interpret these k new 'trajectories' as groups, we may take one of the original trajectories x_j and one of the group trajectories y_i, and compare these in some way to determine whether entity j should be a member of group i.

Here $M^{N\times T}$ is the set of N-by-T sized matrices, with elements from M.

Let $r : \left(M^T\right)^2 \to [0,1]$ be a function which measures the 'similarity' of two trajectories[5], where M^T is the space containing trajectories for a single entity (i.e. the rows of X). Then, we may construct a matrix $R \in [0,1]^{k\times N}$ such that $R_{ij} = r(y_i, x_j)$. We will call this the **relationship matrix** R, as it relates the original entities to the reduced set of variables.

[5] The only condition placed on r is that $r(x,x) = 1$ for any $x \in M^T$.

We have seen before that from a group structure graph $\mathcal{G}$, we may determine the groups by looking at the set of connected components. If the set of entities is denoted E, there is by definition[6] a one-to-one correspondence between E and the vertices of the graph $\mathcal{G}$. As such, the connected components of the graph

[6] See the definition of $\mathcal{G}$ on page 5.

$\mathcal{G}$ will give a partition $\mathcal{C} \subset \mathcal{P}(E)$ of the set of entities. Let this $\mathcal{C}$ be called a **grouping partition**. For such a partition, each entity is assigned to exactly one group. This assignment can be equivalently expressed by a binary relationship matrix $R \in \{0,1\}^{k \times N}$ where each row (corresponding to one entity) contains only one non-zero value (the group it is assigned to). As such, we can easily construct a group structure graph from a binary relationship matrix. We will call the group structure specified by a binary relationship matrix a **strict group assignment**.

However, if the relationship matrix is generated using an arbitrary similarity measure r, R will typically be a real-valued matrix with entries in $[0,1]$. Therefore we must assign an interpretation to the full range of values in order to obtain a group assignment from an arbitrary R.

One way to do so is to consider **soft group assignments**. Instead of each entity belonging to exactly one group, we will specify a probability distribution over the k different groups for each entity. To construct such a probability distribution we will use the matrix R. Let the probability P_{ij} that entity $j \in 1, ..., N$ is in group $i \in 1, ..., k$ be related to R as follows:

$$P_{ij} = \frac{\exp\left(\beta R_{ij}\right)}{\sum_{i=1}^{k} \exp\left(\beta R_{kj}\right)},$$

with $\beta \in \mathbb{R}$, such that $P \in [0,1]^{k \times N}$ is a matrix of probabilities[7]. The function applied here is called the **softmax function** (Bridle, 1990), which provides us with a parameter β that controls the 'certainty' with which we make a group assignment. As $\beta \to \infty$, the application of this function amounts to setting the largest value to 1 and the rest to 0, so in the limit it becomes a strict assignment. In this case each entity is definitely assigned to a group, even if the evidence supporting that assignment is weak. For smaller values of β, the probability distribution will be less concentrated on the largest value and so we allow uncertainty in which group an entity belongs to. This makes it possible that entities are not assigned to any group, but *if* an assignment is made it will be more reliable. In other words, β allows us to make a choice in how 'quick we are to judge' an entity as being part of a group.

What remains is to use the probability matrix P to construct a group structure graph $\mathcal{G}$. The general condition for two entities being in the same group will be

[7] Note that a binary R containing one nonzero element in each column already specifies a distribution over groups for each entity, which means in this case we can simply take $P = R$.

defined as follows:

$$\text{Entity } i \text{ and } j \text{ are in the same group} \iff \sum_{k=1}^{N} P_{ki} \cdot P_{kj} > \frac{1}{2}.$$

This definition says that two entities are in the same group if the probability of them being in the same group is larger than the probability of them *not* being in the same group[8], according to P. Because the groups are taken to be the connected components of $\mathcal{G}$, we can use this as a condition for edges to exist:

$$\text{There is an edge between vertices } i, j \text{ in } \mathcal{G} \iff \sum_{k=1}^{N} P_{ki} \cdot P_{kj} > \frac{1}{2}.$$

[8] Because these are binary outcomes (either they are in the same group, or not), this is the same as the probability of the first outcome being greater than $1/2$.

Note that the sum on the right is exactly the dot product of columns i and j of P. Then, if $P^\top$ is the transpose of P, we may also define the graph as follows:

$$\text{There is an edge between vertices } i, j \text{ in } \mathcal{G} \iff \left(P^\top P\right)_{ij} > \frac{1}{2}.$$

Using this general method, we can construct an analog of the proximity-based group structure graph using the original data matrix X and a reduced data matrix Y, together with a similarity measure r and a fixed 'certainty parameter' β.

3 Dimensionality reduction methods

This section discusses the various **dimensionality reduction (DR)** methods applied in this study. First, a few important terms will be clearly explained. A DR method is generally speaking a procedure that given a set of points in some high-dimensional space, will generate a set of points in a lower-dimensional space that is in some way representative of the original data set. The metric by which 'representativeness' is measured differs per algorithm. This low-dimensional data set is called an **embedding** of the original data set. For conciseness, I will sometimes refer to the high-dimensional space in which the original data is contained as the **input space** or **large space** L, and to the space in which the embedding is contained as the **output space** or **small space** K. Then, given a set of points $X \subseteq L$, the objective of a DR method is to find a set $Y \subseteq K$, $|Y| = |X|$ which is representative of X.

A nice complementary article on dimensionality reduction is the review by Van Der Maaten et al. (2009), where various other dimensionality reduction methods which are not discussed here are compared (and vice versa most methods used here are not treated there).

In the case of the trajectory grouping problem, the original data set consists of $N \in \mathbb{N}$ trajectories followed by entities in some metric **entity space** M, such as the plane $\mathbb{R}^2$ or the circle S^1. Then, the large space L is formed as the metric

product space of N of these entity spaces:

$$L = \prod_{i=1}^{N} M = M^N.$$

If $d : M^2 \to \mathbb{R}$ is a metric on M, the associated metric on M^N can be defined as the l_p-norm of the component-wise distances in the entity space M:

$$d(v, w) = \sqrt[p]{\sum_{i=1}^{N} d(v_i, w_i)^p},$$

with $v, w \in M^N$, $v_i, w_i \in M$ and usually we will choose $p = 2$. Moreover, we will take the reduced space to be $K = M^k$, where $k \leq N$ is the reduced dimensionality.

An important difference between different methods of generating an embedding is their type of **parameterization**. Most methods construct dimensionality-reducing (DR) transformation $f : L \to K$ which maps points from the large space into the small space, and use this to generate the embedding Y from the input set X. These are called **parametric methods**. Some methods do not construct any explicit embedding function[9]: these are *non-parametric*. As the name suggests, parametric methods work by assuming a certain functional form for the embedding function, and then adjusting its parameters so that it generates a suitable embedding. Because parametric methods provide a function which has a larger domain than the points contained in the data set (i.e., $X \subset L$), they provide **generalization**: new points that were not available when the initial embedding was constructed can be embedded without changing existing points or having to rebuild the embedding function.

[9] If we take $L = X$ and $K = Y$, there always exists a trivial embedding function which simply maps each point in X onto one in Y. However, this function does not give us any additional information about *how* X is transformed into Y, so it is not interesting.

Whether an embedding is representative of the original is often determined by the value of some **error metric** between the two data sets. A common metric for parametric methods is the **reconstruction error**. This measures the mean distance between each point and its reconstructed counterpart, which is the point obtained after it has been embedded into the small space, and then transformed back into the large space by the approximate inverse of the embedding function. Suppose we have a input set X and a pair of functions $f : L \to K$ and $\widetilde{f} : K \to L$, where $\widetilde{f}$ is an (approximate) inverse[10] of f. The reconstruction error can then be written as

$$RE(X, f) = \left\| \widetilde{f}(f(X)) - X \right\|^2,$$

where $\| \ldots \|$ denotes a matrix norm. In addition, other constraints are often also

[10] For example, for a linear transformation $\widetilde{f}$ can be the Moore-Penrose psuedo-inverse. In most other cases a reasonable definition for $\widetilde{f}$ can also be given, for example by (linearly) approximating f around the inverses of points in $Y = f(X)$ and using the psuedo-inverse.

imposed either on the embedding function, or the embedding itself. These serve to **regularize** the embedding process and prevent unwanted characteristics of the output.

For non-parametric methods, the reconstruction error is not defined, as there is no embedding function. A quantity which measures the difference between distributions may be used instead, such as the Kullback-Leibler divergence or the Wasserstein metric (also known as earth movers' distance) (Amari et al., 2018).

Most of the different DR methods can be characterized by these properties: their parameterization, the error metric between original and embedding that is minimized, and the precise procedure by which minimization occurs.

3.1 PCA

One of the most frequently used DR methods is **principal component analysis (PCA)**. This method reduces the dimension by calculating the projection of each data point onto a basis of orthogonal axes in the original space. These *principal axes* are ordered, and chosen such that the projection onto each consecutive axis captures as much of the (remaining) variance in the original data set as possible in one (additional) dimension. The projection of the original data onto these axes are called the **principal components (PCs)**.

To implement PCA is quite simple, one only needs to calculate the covariance matrix for the set of trajectory time series and find the eigenvectors of this matrix, which is standard functionality in most math libraries. An introduction to PCA can be found in Shlens (2014).

PCA uses the measure of **captured variance (CV)** to determine the ideal set of components. This measure is directly related to the reconstruction error, in the sense that a k-dimensional (linear) projection which captures the maximal amount of variance is exactly the k-dimensional projection which minimizes the reconstruction error. Another important property is that the variance captured by a set of principal components is equal to the sum of the variance per component, because the components are uncorrelated.

Due to this last property, a k-dimensional basis may be constructed by ordering the components by their captured variance, and selecting the first k components. Then, any N-dimensional point may be embedded into a k-dimensional space by approximating it with the projection onto the basis spanned by these k components.

In the terminology which has been introduced earlier, PCA is a parametric method which constructs a linear embedding function which produces an embedding with minimal reconstruction error. PCA is very simple to apply and to interpret, but that also makes it less powerful than other techniques.

If the data set contains relationships between variables which cannot be well-approximated by a linear relationship, DR by PCA will not provide a very good reduced representation. However, it is the most widely used technique, and is very easily interpretable as each principal component is a linear combination of the original variables, and there is a clear motivation as to why, for example, the first principal component is more important than the rest. As such, it provides a good first attempt to test the application of a dimension reduction method to the TGP.

In terms of the grouping problem, PCA provides a new idea of what constitutes a group. Each principal component corresponds to a mix of the original variables which is highly correlated. As such, PCA may inform us as to whether the proximity-based definition (the idea of groups as sets of entities with limited distance between them) aligns with a notion of groups as sets of entities whose position is sufficiently correlated. If PCA is effective in determining group structure, that provides evidence that these two notions may amount to the same thing.

3.2 ICA

Independent component analysis (ICA) takes the idea of PCA one step further. Instead of determining a basis of uncorrelated components, it aims to find a basis of components which are maximally statistically independent (Hyvärinen and Oja, 2000). To be precise, it aims to find a transformation from N variables onto N components $c_1, c_2, \dots, c_N$, so that the joint probability distribution $P(c_1, \dots, c_N)$ is approximately factorizable as the product of one-component distributions:

A good overview of ICA, its properties, and various algorithms is Hyvärinen and Oja (2000). Common implementations are FastICA and InfoMax (Langlois et al., 2010), but in this project the implementation by Zarzoso and Comon (2010) was used because it showed the most stable results.

$$P(c_1, \dots, c_N) \approx \prod_{i=1}^{N} P(c_i).$$

Usually, ICA is framed in the context of a **demixing problem**: suppose we have N 'sources', where each one emits an independent signal. Then, assume that what we observe are not the N source signals, but mixtures: N signals where each consists of some combination of the source signals. ICA then aims to find an 'unmixing' transformation which reconstructs the source signals from the mixtures.

The kind of ICA applied here is linear ICA, where the sources are assumed to be linearly mixed, and so the unmixing transformation is also linear. Note that, because the ICA components are maximally independent, they should also

be uncorrelated. This makes ICA similar to PCA, but with the requirement of sequentially 'optimal' components in terms of CV replaced by the condition of stronger independence between components. However, the CV is still a useful measure to determine the 'importance' of the independent components. ICA in its standard form is meant to solve a system where the amount of variables is equal to the amount of unknowns, while the TGP is usually applied to an overdetermined system (more entities than groups). Therefore the CV can be used to perform dimensionality reduction using ICA by throwing away the components which contribute the least to the total variance.

One interesting aspect about ICA is that it gives the interpretation of groups as 'sources'. This means the group is considered to be a generator of the entities, and therefore conceptually more fundamental than the entities themselves. However, in practice this has no effect on the mathematical formulation of the algorithms.

3.3 SOMs

Self-organizing maps (SOMs) provide a way to construct embedding functions which project the original data onto a finite set of **anchor points**[11]. These are points taken from the small space K, arranged in a lattice structure (usually square or hexagonal-shaped). As such, the resulting embedding is discrete, the embedding function is discontinuous, and the embedding is (a priori) contained in a connected region of K with finite volume.

The idea is as follows: each anchor point $a_i \in K$ has an associated 'source point' $w_i \in L$ in the large space, usually called its **weight**[12]. When a data point p is embedded, each anchor's **activation** is determined, which is a real number given by a decreasing function of distance between the data point and the anchor's weight $d(p, w_i)$. The input point is then projected onto the *best matching unit (BMU)* a_{BMU}:

$$a_{BMU} = \underset{a_i}{\operatorname{argmax}}\, \mathcal{A}(d(p, w_i)) = \underset{a_i}{\operatorname{argmin}}\, d(p, w_i),$$

where $\mathcal{A} : \mathbb{R} \to \mathbb{R}$ is a non-negative, monotonically decreasing *activation function*. Usually, a Gaussian function is chosen as activation function:

$$\mathcal{A}(x) \propto \exp\left(\frac{-x^2}{2\sigma^2}\right),$$

with $\sigma \in \mathbb{R}$. The BMU is the anchor with the highest activation, which is also the

The original proposal of the SOM algorithm is by Kohonen (1990) and further details on its properties and implementation details can be found in Cottrell et al. (2016). The implementation used in this project was based on the optimization formulation by Heskes (1999).

[11] Because the SOM is inspired by the way neuronal pathways adapt in the brain, what is referred to here as anchor points are usually called *neurons*.

[12] These weights can be initialized either randomly or by picking equally spaced values along the axes of the first k principal components.

anchor with the weight closest to the input point, so that the precise activation function chosen does not matter for determining the BMU (however, it does influence the update procedure, as we will see).

An important property of SOMs is that they try to preserve topological characteristics of the input data. This is enforced while *learning*, the stage in which the map is modified to properly embed the input data, through a process inspired by Hebbian learning: "neurons that fire together, wire together". Each point in the input data is provided to the map, and the BMU is calculated. Then, the weights in the map are updated so that the BMU's weight moves toward the input point, so that in the future it will activate even more strongly. But: this update affects all weights in the map, where the amount the weight w_j is changed is proportional to the activation function applied to the inter-anchor distance:

$$\Delta w_j \propto \mathcal{A}\left(d(a_{BMU}, a_j)\right).$$

Because $\mathcal{A}$ is decreasing (e.g. Gaussian), the points close to the BMU will be changed more than those far away.

The effect this has is that anchor points that appear 'close' in the low-dimensional map will have their weights kept close as well, and so are modified over time to also respond strongly to similar input points. This means that generally points which are close together in the large space, will be projected onto anchors which are close together in the small space[13]. Additionally, as the learning process progresses, the 'radius' σ of the activation function will usually be decreased. This means that the map is initially 'flattened out' by giving each update a large influence over its neighbors, but eventually each weight is allowed to specialize almost independently.

[13] This 'neighbourhood effect' can be alternatively viewed as a form of regularization which limits the size of the discrete derivatives across the map.

The SOM generates an embedding function which is not limited to a linear parameterization such as in PCA or ICA. Arbitrary shapes in the large space can be sampled by the anchor points, which creates the possibility for nonlinear curves to be 'unrolled' onto straight lines, reducing the dimensionality while providing low reconstruction error. This is balanced by the trade-off that the embedded data set will be discretized, causing a loss of information when embedding. In addition, the learning process for the SOM is more complicated and so is less guaranteed to produce good results.

Unfortunately, the largest disadvantage of the SOM algorithm is its computational complexity. The grid of anchor points grows exponentially in the 'small' dimension k. If r is the number of points along each of the k axes, the total number of points grows as r^k. This means that for reducing into a data set with

$N = 1$ to 3 it performs reasonably well, but beyond that it quickly becomes very expensive in both memory and computation time. Additionally, from a theoretical point of view it is strange that the number of anchor points, and by extension the number of parameters, may quickly become much larger than the number of input data points[14]. This seems like it would lead to overfitting, but in practice this is not so much a problem as regularization makes the effective degrees of freedom of the map much lower.

However, it is still necessary to store all anchor points and compute all r^k pairwise distances, which quickly becomes computationally intractable. These limitations make the traditional SOM unsuitable for this application, as it is common to have more than a few groups. For this reason, a modified version of the SOM algorithm has been designed, and this is introduced in the following section.

[14] As an example, consider a sample with $N = 8$ of 200 points. This may be projected into $k = 4$-dimensional space with a resolution of $r = 10$ anchor points per dimension. Then, the number of anchor points is $r^k = 10000$, which means there are 50 times as many parameters as there are data points!

3.4 A SOM variant for trajectory grouping.

There are variants of the SOM algorithm which aim to reduce the computational complexity. One example is **curvilinear component analysis (CCA)** (Demartines and Herault, 1997), which is a non-parametric variant. It does not construct a full mapping from the input space L to the output space K, but only a suitable embedding of the data points in which the 'neighborhood' regularization is respected. As such, it only takes the pairwise distances of the in/output points into consideration. However, in the current application the constraint that the output must lay on a fixed grid in the output space is useful, as it seems to encourage separation of independent components onto orthogonal axes[15]. Furthermore, if we base our mapping only on the pairwise distances between data points, we cannot give a clear interpretation to the meaning of the axes in the reduced data set.

[15] This may be understood as a ℓ_1-type regularization performed by encouraging alignment to a rectangular grid. It is also corroborated by Pajunen (1996) from a statistical point of view.

However, there is a way to reduce the complexity by using some additional structure found in the problem. We will assume that the underlying groups are independent, and we would like to project each of the groups onto an orthogonal subspace for the purposes of identification. As such, the projection mechanisms onto these subspaces do not need to interact: in fact, it is better if they are as mutually independent as possible. This motivates a new SOM variant, which aims to restrict the amount of parameters by only defining a set of k non-interacting 1-dimensional projections, instead of a single projection onto a full k-dimensional grid. We will refer to this procedure as **dimension grouping (DG)** in reference to the problem for which it has been designed.

There are two key observations on which this method is based:

1. Because we assume it is possible to partition our set of N trajectories into k independent groups, the dimensionality-reducing mapping

$$f : M^N \to M^k$$

should be **separable** into a concatenation of per-group transformations:

$$f(x_1, \dots, x_N) = \left(f_1(x_i, \dots), \dots, f_k(x_j, \dots)\right)^\top,$$

Here $x^\top$ is the transpose of x.

where $f_i : M^{q_i} \to M$ is a function projecting the members of group i of size q_i onto a single instance of the entity space.

2. Because we have trajectories which are continuous in time, any two points on a single trajectory which are close in time will also be close in space. Therefore we can reasonably substitute the spatial distance between points by the temporal distance when performing regularization.

First these two observations will be further illustrated. Then, the algorithm will be described, after which the advantages of this approach will be discussed.

Consider a point $y = \left(y_1, y_2\right)^\top$ in a two-dimensional output space $K = M^2$, which is the projection of some higher-dimensional point $x \in M^N$. If we would like these two components y_1, y_2 of y to represent one group each, we should be able to determine their value independently of one another, as *changing the position of one group should not change the position of the other*. This means that we should be able to find two independent projections $\{x_i, x_j, \dots\} \mapsto y_1$ for the first group and $\{x_k, x_l, \dots\} \mapsto y_2$ for the second group (where all indices i in x_i are distinct), and build up our full projection $x \mapsto y$ by combining these projections. We can project the subset of the entries of x which form the first group onto y_1, and the other entries (which form the second group) onto y_2 separately. This division of x into disjoint subsets corresponds to the grouping partition discussed earlier.

Because of this independence property of our groups, this means instead of finding a single projection $f : M^N \to M^k$, we can find a number of projections $f_i : M^{q_i} \to M$ where $1 \leq i \leq k$ and q_i is the size of group i, and then combine the resulting k trajectories. In other words, instead of projecting our N trajectories onto a grid in k-dimensional space as in the SOM, we project them onto a set of k trajectories, which may then be reassembled into a k-dimensional product space. The important part here is that the optimization

process does not explicitly construct a k-dimensional space, which for the SOM caused intractability from $k \geq 3$ onwards. To be specific the space complexity reduces from $O(r^k)$ to only $O(k \cdot r)$, and the time complexity reduces similarly as well, as it is no longer necessary to compute r^k possibilities to find where a trajectory should be projected or to regularize the output.

This leaves the question of how to regularize these k separate projections efficiently, as the grid structure is no longer available. For the regular SOM, close points in the output space should be close in the input space, and vice versa. Calculating this regularization term leads to an amount of computation proportional to the r^k grid size, as these constitute all possible outputs. However, because in this case the input trajectories are continuously parametrized by the time parameter, two points along a trajectory which are close in time will also be close in the input space. Therefore, as a similar but weaker condition, we may require that points which are close *in time* in the input data are also close in space in the output data. This reduces the regularization in k spatial directions into a regularization performed in one temporal direction.

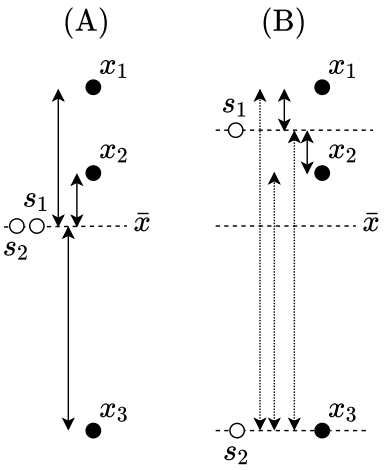

Figure 1: An illustration of how the DG algorithm divides entities into groups. The representatives s_1 and s_2 are chosen such that the distances to the entities $x_{1,2,3}$ are minimized. Minimizing the total distance would place both representatives on the average value $\bar{x}$. However, because only the distance from each entity to the closest representative is taken into account (the solid arrows), the representatives are free to ignore the distance to members of other groups (the dotted arrows). In this example, configuration B would be preferred over configuration A.

The algorithm works as follows. The input data is a set of N trajectories $\{x_1, \ldots, x_N\}$ with each $x \in M^T$, and so can be described by a $N \times T$ data matrix X where each row is one trajectory. These trajectories can also be viewed as discrete functions $x[t]$ of a time parameter $1 \leq t \leq T$. If there is group structure in this trajectory data such that the trajectories can be divided over k groups, that means we should be able to replace each group of trajectories $\{x_i,\ x_j,\ \ldots\}$ by a single representative s. Because there are k groups, after replacing each group we have a set of k representative trajectories $\{s_1, \ldots, s_k\}$, each also T points long, which we will store as a kxT matrix S. Because we are mapping from X (NxT) to S (kxT), our dimensionality reduction takes place from M^N to M^k as expected.

The method of forming the representatives s_i is as follows:

1. Start with a set of k trajectories of length T with random values. These are the representatives $\{s_1, \ldots, s_k\}$, which are stored in a matrix S.

2. For each input trajectory x_j (given by the rows of a matrix X) and time t, define the **grouping error** as

$$\varepsilon_{\sigma,S,X,j}[t] = \min_{1 \leq i \leq k} G_\sigma \star d\left(s_i[t], x_j[t]\right)^2.$$

   Here $s_i[t]$ represents the value of representative s_i at time t (and similarly for x), G_σ is a 1-dimensional Gaussian kernel with standard deviation σ,

$\star$ represents convolution, and d is the distance metric associated with the space M. The Gaussian kernel plays a similar regularizing role as the Gaussian activation function $\mathcal{A}$ in the SOM algorithm.

The convolution here happens between two discrete functions $G_\sigma[t]$ with domain $\{-L, ..., L\}$ and $D[t] = d\left(s_i[t], x_j[t]\right)^2$ with domain $\{0, ..., T\}$ and is defined as:

$$(G_\sigma \star D)[t] = \sum_{l=-L}^{L} G[l]D[t-l],$$

which also has domain $\{0, ..., T\}$. In case $t - l$ is negative or greater than T, the value of D is taken to be zero. The size of the kernel $2L + 1$ is based on σ (a rule of thumb is $L = 3\sigma$ which covers nearly all of the Gaussian curve).

The reasoning behind this error function is as follows:

(a) Each trajectory will be projected onto the closest representative at each time step. Therefore, the local 'projection error' will be a function of the smallest distance between the entity x_j and any of the representatives s.

(b) This projection error is also 'smoothed' over time by the Gaussian convolution. The local error then becomes a weighted combination of the current error and the error at nearby times. As a result, if $s_i[t]$ is close to $x_j[t]$, it is preferred (i.e. gives a lower error) if at nearby time values $t + \Delta t$, $s_i[t + \Delta t]$ is close to $x_j[t + \Delta t]$ as well.

Together, these two principles support two desirable grouping properties of separability and consistency, as pictured in Figures 1 and 2.

3. Define an energy function[16] as follows:

$$E_\sigma(S, X) = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{N} \varepsilon_{\sigma,S,X,j}[t]$$

This simply takes the total energy to be the total grouping error for all trajectories, averaged over time.

4. This energy function is then minimized using a gradient descent algorithm which adjusts the values of the $\{s_i\}$. As the algorithm progresses, the value of σ is decreased, so that first the general structure of the output is determined, and then the fine details of the representatives are tuned.

This algorithm can be viewed in two ways. The first is in its similarity to the SOM algorithm. It is similar in the sense that the same operations are

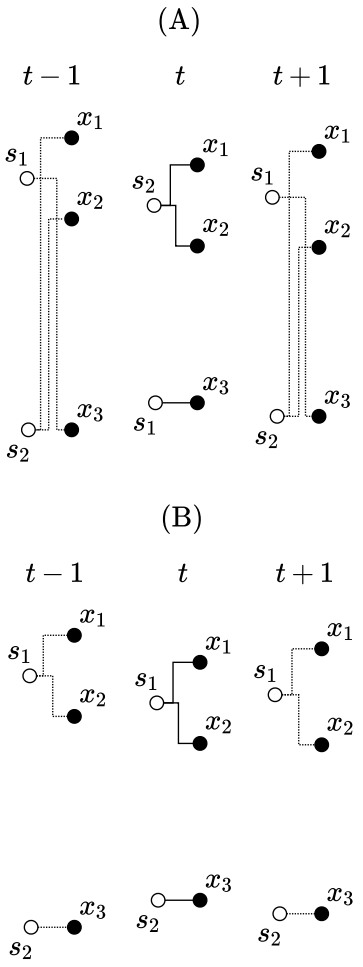

Figure 2: An illustration of how the DG algorithm promotes consistent groups. The relevant distances between entities and representatives are marked by lines, where only the vertical part corresponds to actual distance. When looking only at one time t, the configurations A and B are equally valid. However, not only the current distances between entities and their representatives (solid lines) are taken into account, but also *these same distances at nearby times* (dotted lines). Therefore, configuration B would be preferred over configuration A.

[16] The term *energy function* is used here simply to mean 'a function of the state of the system which is to be minimized'.

performed: a finite set of 'anchor points' is defined (by the representatives s_i), and iteratively adjusted while locally preserving the distances between input points and their projections (through regularization). The decreasing value of σ is also taken from the SOM algorithm. Performing the learning process by function optimization is not part of the original SOM algorithm by Kohonen (1990), but is taken from the energy function approach to SOMs by Heskes (1999).

It differs from the SOM in that the anchor points are *implicitly* determined. At each time step, there are k possible representatives for each trajectory. This means that the number of N-dimensional weights available is determined by the number of combinations one can choose one of the k representatives for each of the entities. Because multiple entities may end up in the same group, these are chosen with replacement, so there are $\binom{k+N-1}{N}$ weights in the N-dimensional space at each time step, constructed from combinations of the k different representatives. However, unlike the SOM more than one weight will map onto the same output or 'anchor point' in k-dimensional space.

The second way to view this algorithm is as a kind of *time-smoothed clustering procedure*. In the limit $\sigma \to 0$, the Gaussian kernel becomes a Dirac delta distribution (discretely, a Kronecker delta function $\delta[t] = \delta_{t,0}$), so that convolving with it becomes an identity operation: $\delta[t] \star x[t] = x[t]$ for any function x. Then, the energy function reduces to:

$$E_\sigma(X, S) = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{N} \min_{1 \le i \le k} d\left(s_i[t], x_j[t]\right)^2.$$

This is exactly the cost function for the k-means algorithm[17], performed for every time step and then averaged over time. This means that effectively this algorithm performs a k-means clustering at each time step, with the added constraint that these clusters should be consistent from one instant to the next.

[17] See for example Bottou and Bengio (1994) for a formulation of k-means as a gradient descent procedure.

Compared to a k-dimensional SOM, this algorithm should be much less time-consuming. The k representatives can be efficiently trained simultaneously while avoiding the exponential complexity of the SOM, because they do not interact directly, but only through the mechanism of competitive learning. As each trajectory point can only map onto one representative, the representatives 'feel' each other's presence in the sense that, when a trajectory is assigned to one representative, the other representatives are free to specialize in other ways without negatively affecting the overall error.

The regularization by convolution ensures that the resulting representatives indeed take on the shape of *trajectories* instead of arbitrary time series. Just

as in the regular SOM, where Gaussian convolution assures points close in the small space are also close in the large space, here the convolution ensures points that are close in time in the input will map to points which are close in the output space, preserving the continuity of the trajectories. This makes the representatives *consistent* in the sense that they prefer to stay with the same set of trajectories $\{x_i\}$ over time.

The interpretation of the new k-dimensional coordinates is clear: they are trajectories in their own right, which serve to approximate the larger number of N trajectories we had initially. This makes them explicitly well-suited to the trajectory grouping problem, as they provide a projection of the trajectory data into a space where the basis axes directly correspond to groups.

4 Experimental setup

4.1 Trajectory data from the Kuramoto model.

The trajectory data which will be studied here comes from the Kuramoto model. This is a dynamical system consisting of N entities which each have a position on the unit circle and so are represented by N angles $\{\theta_1, \dots, \theta_N\}$. These entities move at their respective **natural velocities** $\{\omega_1, \dots, \omega_N\}$. Additionally, an interaction term of the form $\kappa \cdot \sin(\theta_j - \theta_i)$, which aims to minimize the difference of velocities between all entities and bring them together. This makes the parameter κ an **interaction parameter**, which influences the group structure of the resulting trajectory data.

The space in which the entities live is the circle S^1. Points in this space can be uniquely represented by their angle $\theta \in [0, 2\pi)$ relative to some arbitrary origin $\theta_0 = 0$. As such, the space containing a sample of N entities will be the product space $L = \left(S^1\right)^N$, and the associated distance metric will be the ℓ_2-norm combination of angular distances, where the angular distance is simply the shortest distance along the circle between two points on the circle. This may be written as:

$$d(\theta_i, \theta_j) = \left|\mathrm{Arg}\, e^{i(\theta_j - \theta_i)}\right| = \min(|\theta_j - \theta_i|,\ 2\pi - |\theta_j - \theta_i|).$$

A number of different data sets have been generated for this study, with varying numbers of entities N and interaction parameter κ. In all cases, the natural velocities ω_i have been sampled from a standard normal distribution, while the

For more details on the Kuramoto model and the kind of behavior it can generate, see for example the study by Maistrenko et al. (2005).

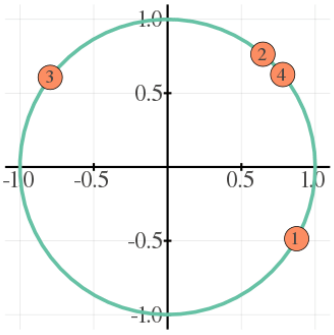

Figure 3: An example simulation of the Kuramoto system with $N = 4$ and $\kappa = \frac{7}{8}\kappa_c$. This is an animation which plays in supported PDF viewers.

The second form of d amounts to saying: either use the distance travelled when moving counterclockwise from θ_i to θ_j, or use the distance *not* travelled when moving counterclockwise from θ_i to θ_j (which is the clockwise distance), whichever is shorter.

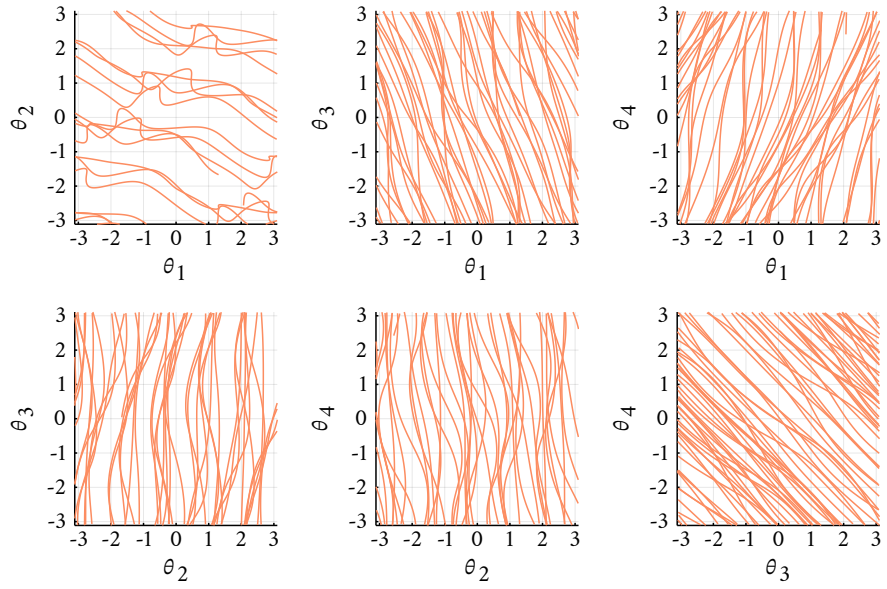

Figure 4: A phase plot of a simulation of the Kuramoto system with $N = 4$ and $\kappa = \frac{7}{8}\kappa_c$. The position for each pair of entities $(\theta_i, \theta_j) \in [-\pi, \pi)^2$ with $i \neq j$ is displayed to view the movement of entities relative to each other. In this case, it intuitively seems that the 'connection' between θ_1 and θ_2 is stronger than the other combinations: the distribution of the position pairs (θ_1, θ_2) seems less uniform than the other combinations. To perform dimensionality reduction, we might therefore choose to approximate the $\theta_1 - \theta_2$ plane using a single dimension (i.e. a line wrapped around the torus). Alternatively, if we were performing trajectory grouping, we might take there to be two groups: $\{\theta_1, \theta_2\}$ and $\{\theta_3, \theta_4\}$.

initial angles θ_i were chosen 'evenly separated' over the interval $[0,\ 2\pi)$:

$$\theta_i(t = 0) = \frac{i - 1}{N} \cdot 2\pi,\ 1 \leq i \leq N$$

This is to limit the appearance of 'coincidental groups', where both the initial velocity and angle are very close by chance, so that they will appear as a group even without the influence of the coupling term.

In each case, the system was simulated for a length of 300 time units, and with time steps of $\Delta t = 0.01$, leading to trajectories with a total length of $3 \cdot 10^4$ time steps. Before analysis, the first 10^4 time steps were excluded in order to study the behavior on the attractor only, leading to trajectories of length $T = 2 \cdot 10^4$.

Initially, a number of data samples with $N = 4$ were generated, as these could be investigated visually by viewing all 6 two-dimensional perspectives of the 4D phase space, as in Figure 4. For higher values of N, the analysis was only numerical. The values of N chosen for the simulations were all multiples of 4 up to 32:

$$N \in \{4, 8, 12, 16, 20, 24, 28, 32\}.$$

The values of κ were chosen in relation to the so-called **critical value** κ_c, which for a given distribution of ω_i's determines the point where the system switches between a globally synchronized and a desynchronized state. The value of κ_c is determined from the natural velocity probability density function $p(\omega)$ as

$$\kappa_c\,(p) = \frac{2}{\pi \cdot p(0)}$$

in the case of an unimodal distribution. The values used for the analysis below were

$$\kappa \in \left\{\frac{2}{3}\kappa_c, \frac{7}{8}\kappa_c\right\}.$$

For each combination of parameters (N, κ) a number of $C = 30$ simulations were run, with each using a new random sample of initial velocities.

Using the proximity-based method, the amount of groups in each run of the simulation was determined. These counts are shown in Figure 6 in the next section. As the values of κ chosen lie slightly below the critical value, the number of groups typically lies around halfway between the minimum number of groups (1) and the maximum number of groups (N).

4.2 Dimensionality reduction on Kuramoto data.

A number of different dimensionality reduction techniques have been applied to the simulations of the Kuramoto model: *PCA*, *ICA*, *SOM*s, and the newly introduced technique described earlier, referred to as *dimension grouping (DG)*.

These methods have been tested in two ways:

(1) How well does the ideal number of 'components' agree with the number of groups as determined by the proximity-based method?

(2) How well do the group structure graphs $\mathcal{G}$ obtained from the dimensionality reduction and the proximity-based method agree?

For (1), it is necessary that some of these methods provide a measure by which the 'importance' of each component can be determined. PCA uses the (fraction of) captured variance (CV) to determine the ideal set of components. As such, the CV can be used to order the components, and also to determine the (minimum) number of components needed to represent the data set. When the number of components k is determined, this can be compared to the number of groups given by the proximity-based method. For ICA, because the resulting components are (ideally) statistically independent, the CV for a given set of components is also equal to the sum of the CV per component. As such, the CV per component may also be calculated and used to determine how many components are relevant. For the SOM-based methods, the number of dimensions k must be chosen manually, and so only PCA and ICA may be used to inform the number of groups present.

For (2), the original simulation data in N dimensions was reduced to a set of

k components where k is the number of groups determined by the proximity-based method. Because the data from the Kuramoto model consists of angular variables, it is important that the application of statistical methods to them occurs in a valid way. For the SOM-based methods, all statistics are applied to the distances between points, so there is no problem. However, for PCA and ICA the trajectories are compared directly, and quantities such as the mean and covariance are calculated. These are not trivially defined in the case of angular data[18], so a transformation is made to complex numbers instead. Then, both PCA and ICA may be applied as normal, with some caveats. This is discussed in more detail in Appendix B.

[18] For data on the unit circle, the mean cannot easily be identified, because the origin $\theta = 0$ is arbitrary, and because nearby angles might have a large difference in numerical value (consider e.g. 5° and 355°).

A common method (Fisher, 1993; Jammalamadaka and Sengupta, 2001) to define the mean of a set of angles $\{\theta_k\}$ is through the sum of their vectorial counterparts. This corresponds to taking the sum of complex numbers $z_k = \exp(i\theta_k)$. This motivates the transformation to complex numbers. For another instance of angular data being treated in this way see Altis et al. (2007).

4.3 Extracting the group structure.

The next step in the experiment is the construction of a group structure graph $\mathcal{G}_{DR}$ using the reduced data set. An overview of this process is given in Figure 5. To construct this graph, the original data matrix X and the reduced data matrix Y are compared. Every trajectory x in the original (i.e. every row in X) is compared to every dimension y in the reduced space (i.e. every row in Y) by a similarity measure $r(y, x)$. Two measures were applied:

(i) Correlation.

(ii) Normalized mutual information (NMI).

Both of these measures produce a real number between 0 and 1. The correlation measures how close the relationship is to a pure linear relationship $y = ax + b$ for some a, $b \in \mathbb{R}$. The NMI is a more general measure of shared information, which is independent of the functional relationship. Rather, it measures how far away the joint probability distribution $P(y, x)$ is from the product distribution $P(y)P(x)$. If these two are equal, y and x are independently distributed and so the NMI will be zero.

Given distributions x and y, the NMI is defined as

$$NMI(x, y) = \frac{2MI(x, y)}{H(x) + H(y)},$$

where $MI(x, y)$ is the mutual information of x and y and $H(x)$ is the entropy of x.

The results of these comparisons were stored in a $k \times N$ relationship matrix R, where each entry

$$R_{ji} = r(y_j, x_i),$$

with y_j being the j-th row of Y and x_i the i-th row of X. As discussed in Section 2.2, the matrix R is then normalized via softmax to produce a number of probability distributions, and from these a partition assigning the original trajectories (entities) to the reduced dimensions ('groups') can be constructed. This grouping partition is then used to construct the graph $\mathcal{G}_{DR}$.

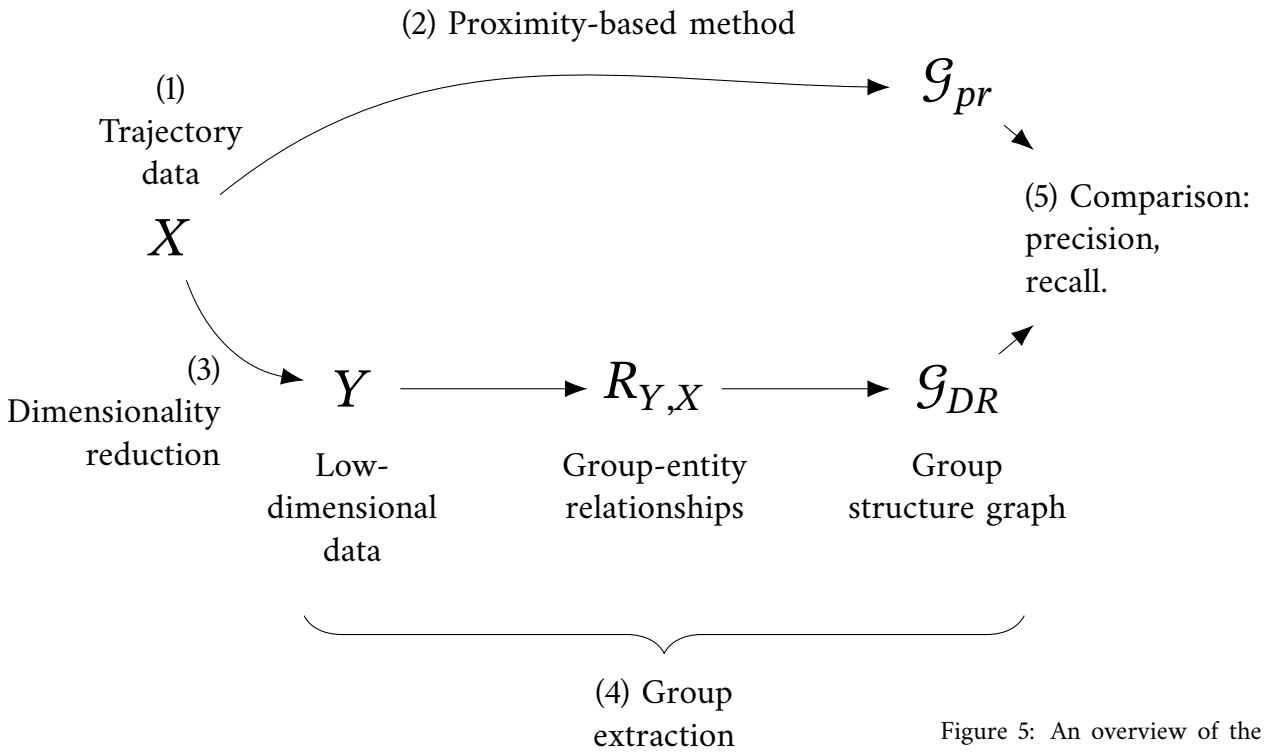

Figure 5: An overview of the experimental setup. Starting from the simulated trajectory data X (1), a reference set of groups is generated via the proximity based method described in Section 2 (2).
Additionally, a dimensionality reduction algorithm is applied to X generate a representation Y which has a dimensionality equal to the number of groups (3). By comparing these two representations of the trajectory data, a set of group assignments is determined (4).
This set of groups may be compared to the groups from (2) to test the effectiveness of this method of trajectory grouping via dimensionality reduction (5).

4.4 Comparative analysis.

To evaluate the quality of the groupings produced by application of DR methods, the graph produced by the proximity-based method $\mathcal{G}_{pr}$ is used as a benchmark or 'ground truth', and the graph $\mathcal{G}_{DR}$ produced by dimensionality reduction is evaluated by seeing how well it agrees with $\mathcal{G}_{pr}$.

In these graphs, the vertices represent the entities, while the presence of an edge represents two entities being in the same group. As such, an accurate 'reproduction' of $\mathcal{G}_{pr}$ by $\mathcal{G}_{DR}$ consists of two parts: it should contain edges when two entities are in the same group according to $\mathcal{G}_{pr}$, and it should not put edges between entities which are not in the same group according to $\mathcal{G}_{pr}$. If a correctly placed edge is a *true positive (TP)*, a failure of the former kind (a missing edge) would be a *false negative (FN)* and of the latter kind (an extraneous edge) a *false positive (FP)*.

From these three quantities, the accuracy of a graph $\mathcal{G}_{DR}$ may be evaluated using the metrics of **precision and recall**:

$$\text{precision} = \frac{TP}{TP + FP},$$
$$\text{recall} = \frac{TP}{TP + FN}.$$

Precision represents the proportion of group assignments made which are correct, while recall represents the proportion of possible correct group assignments which are made. As such, there is a tradeoff between precision and recall which is related to the certainty parameter β seen before. A high value of β will make many assignments and thus lead to high recall but low precision, while a low value of β will be conservative in it's assignments and thus have high precision but low recall.

These two measures can also be combined into a single measure called the F_1-**score**:

$$F_1 = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

This is the harmonic mean of precision and recall, and places equal emphasis on the two. Other variants with a parameter also exist, called F_β, where β controls the relative weight of recall to precision. As such, this parameter is very similar to our parameter β, and therefore it should be sufficient to use the F_1-score only, given that we have already chosen fixed values for our softmax-β.

Note that it is only conjectured here that the two β-parameters (in the F_β-score and in the softmax function) are roughly interchangeable, it has not been rigorously tested nor proven mathematically.

Because the evaluation of these measures is comparative between the different methods, it is not very important which value of β is chosen, as it simply changes the tradeoff made between precision and recall. It is only important that a value is chosen which leads to useful measurements, i.e. one which does not lead to one measure being heavily favored over the other, so that there is little nuance in the results. Suitable values for β also depend on the particular similarity measure used (correlation or NMI) as these produce a different 'spread' of values in the range $[0, 1]$ and so are naturally predisposed to valuing precision over recall in a certain way[19]. For this reason, fixed parameter values of β were chosen for each of the two measures by hand (as to show a reasonable variety in both precision and recall scores), but not varied any further. The exact values for all parameters are discussed in the next section.

[19] For example, if the numbers produced by the similarity measure are likely to be clustered around a certain value, a higher emphasis on precision might be necessary to correctly distinguish 'strong' assignments from weak ones, compared to a case where the values are evenly spread between 0 and 1.

4.5 Implementation.

The simulations and further data processing steps were performed by code written in Julia, using libraries such as:

- `DrWatson`[20], for project management.
- `DynamicalSystems`[21], for simulating the Kuramoto system.
- `Zygote`[22], for function optimization.

[20] `https://github.com/JuliaDynamics/DrWatson.jl`

[21] `https://github.com/JuliaDynamics/DynamicalSystems.jl`

[22] `https://github.com/FluxML/Zygote.jl`

(Complex) PCA was performed by using the basic statistics functionality in Julia. Given data matrix X, the PCA algorithm simply calculates an eigendecomposition of the covariance matrix of X, which gives the PCs (eigenvectors) and CV per component (eigenvalues). ICA was performed using the algorithm presented in Zarzoso and Comon (2010), by calling out to the provided MATLAB code[23] using a combination of GNU Octave, Python, and the `oct2py`[24] library. This algorithm was used because it is explicitly designed to handle complex inputs as well as real ones. The SOM algorithm and the DG variant were implemented by hand, by using Julia's multidimensional arrays to store the anchor points, and constructing an energy function. This energy function could then be optimized by using `Zygote` to set the parameters (the anchor points and weights) for the mapping. Other implementations of the SOM exist and were tested, but none could handle output dimensions $k > 2$ (which is sensible in hindsight). The code for the DG algorithm is available online as a Julia package[25].

[23] `http://www.i3s.unice.fr/~zarzoso/robustica.html`

[24] `https://github.com/blink1073/oct2py`

[25] `https://github.com/nardi/DimensionGrouping.jl`

When performing PCA, the data was not mean-centered because of its angular characteristics (see Appendix B for more details). When performing ICA, the algorithm parameters used were the `regression` deflation type, `dimred` enabled, and no prewhitening (for the same reasons as in PCA). In both cases, the angular data was provided in complex form, as this enables an interpretation of the linear transform as translation along the circle (see Appendix B).

For the DG algorithm, the number of optimization steps had to be chosen. The learning process was divided into four parts, with each lasting a maximum number of 'rounds' (gradient descent steps) R. Using absolute and relative tolerance parameters, each of the four parts could terminate early if the remaining absolute/relative error was within tolerance. The 'width' of the convolution σ would decline with each round r as

$$\sigma_r = \frac{\sigma_1}{1 + \frac{2r}{R}},$$

where σ_1 is the starting width. In this experiment the parameter values used when training were $(R, \sigma_1) = (60, T/2)$, $(30, T/6)$, $(60, T/20)$, $(50, T/100)$, where T is the number of samples in the entire trajectory.

The correlation function is available in base Julia, while the NMI was programmed using the mutual information function from the package `InformationMeasures`[26]. The softmax parameters used were $\beta_{cor} = 5$ for the correlation measure and $\beta_{NMI} = 20$ for the NMI.

[26] `https://github.com/Tchanders/InformationMeasures.jl`

5 Results

In this section, the results of the experiments described in Section 4 are reported. All of the analyses were performed on a dataset of 480 simulations of the Kuramoto model. These contained all combinations the selected parameter values, and each combination was simulated $C = 30$ times with different sets of randomly sampled natural velocities. The parameters are described in detail in Section 4.1.

During the development of the algorithms and methods of analysis, a 'training set' was used to establish working algorithms and to set a number of constant parameters which could not be determined *a priori*, such as the β-parameters for correlation and NMI, and suitable tolerances and amount of steps for optimization in the DG algorithm. To produce the results shown here, this training set was swapped out for a 'test set', where the same parameters were used with the exception of C newly randomly generated sets of natural velocities per configuration of parameters. As such, in both the training and test sets the same parameters and ω-distributions were represented, so that on they should give the same qualitative grouping behavior, but the actual numerical trajectories used were entirely different as to show robustness within this range of parameters. The number of groups present in this data set is shown in Figure 6. Here it can be seen that as N increases, the distribution of the number of groups k tends to the center of the total range $1 \leq k \leq N$, and for every N at least half of the total range is covered by the data.

Many of the figures here represent distributions using box plots. If there is any uncertainty on how to read these, please refer to Figure 12 in Appendix A for a general explanation, and to the captions for each figure for details.

5.1 Estimating the number of groups.

When estimating k (the number of groups), the 'true' value of k was taken to be the number of groups predicted using the proximity-based method. To compare the number of groups obtained by performing PCA or ICA, first the captured variance per dimension was determined. Then, the dimensions were ordered by their CV in descending order, and a value for k was chosen by putting a condition on the sum of the first k values. If the ordered principal/independent

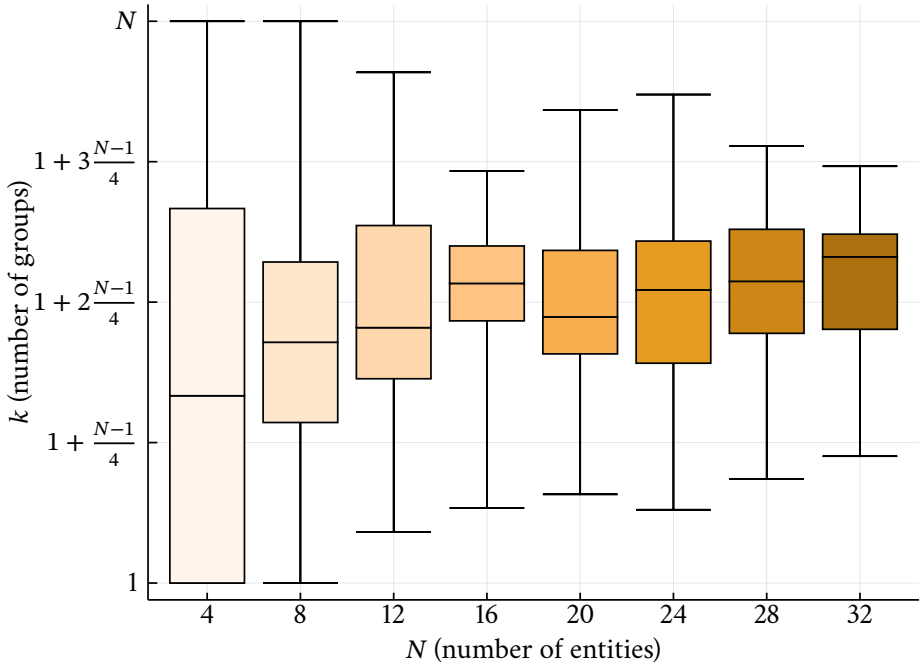

Figure 6: The distribution of the number of groups k in the data set used for the analysis, scaled relative to the number of entities N. For each N, the distribution is described by the median (line inside the box), the first and third quartile (bottom/top of the box), and the minimum and maximum values (outer edges).

components are given by y_i for $1 \leq i \leq N$, then k was chosen such that

$$\frac{\mathcal{P}_k}{1 - \mathcal{P}_k} \geq \mathcal{O} \tag{1}$$

where $\mathcal{P}_k$ is the proportion of variance contained in the first k components,

$$\mathcal{P}_k = \frac{\sum_{i=1}^{k} \text{var}(y_i)}{\sum_{i=1}^{N} \text{var}(y_i)} \tag{2}$$

and $\mathcal{O}$ is a constant representing the required 'odds' for a sufficient representation of the full N-dimensional data set. The interpretation of this condition is that k is chosen such that *there is at least $\mathcal{O}$ times as much variance contained in the first k components as is contained in the other $N - k$ components.*

When comparing the estimated number of groups N_{est} to the actual number of groups N, we may consider the difference $\Delta N = N_{est} - N$. The distribution of these differences is displayed in Figure 8. The accuracy can be calculated as the number of correct estimates ($\Delta N = 0$) divided by the total number of estimates. For PCA, the accuracy is $\frac{369}{480} \approx 77\%$ and for ICA the accuracy is $\frac{161}{480} \approx 34\%$. If we allow off-by-one errors ($-1 \leq \Delta N \leq 1$), then the accuracy becomes $\frac{471}{480} \approx 98\%$ for PCA and $\frac{370}{480} \approx 77\%$ for ICA.

The values used for the odds-constant $\mathcal{O}$ were $\mathcal{O}_{PCA} = 150$ and $\mathcal{O}_{ICA} = 12$. These were chosen so that on average the error ΔN was approximately 0 on the 'training set'. These values show that the distinction between the relevant components and the discarded components is much stronger for PCA than for

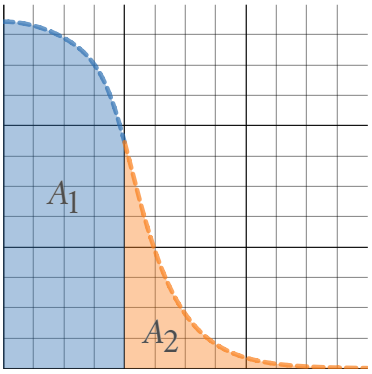

Figure 7: Choosing the number of dimensions to keep when performing DR via PCA amounts to choosing a 'cutoff' point for which components with captured variance (CV) below the cutoff are dropped. For example, if this graph shows the CV (y-axis) per component (x-axis, in decreasing order), one might split it into two regions A_1 and A_2, where the components in the left region are kept and the rest are dropped. Using the rule in Equation 1, the cutoff is chosen so that the ratio of these areas A_1/A_2 is at least $\mathcal{O}$.

ICA: the fraction of variance included in the first k components is about an order of magnitude larger for PCA than for ICA.

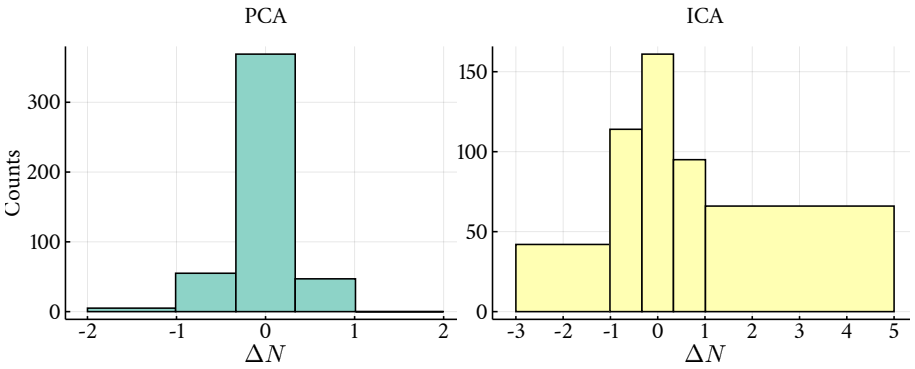

Figure 8: The performance of using PCA or ICA to estimate the number of groups as the number of relevant components, determined using the criterium in Equation 2. The x-axes represent the difference between the estimate and the 'correct' number of groups (as determined by the proximity-based method).

5.2 Estimating the group structure.

In estimating the group structure, 4 ways of performing dimensionality reduction were applied: PCA, ICA, dimension grouping (DG) and additionally a combination of PCA and ICA, hereafter referred to as **PICA**. Because the results of Section 5.1 indicate PCA is better at predicting the number of groups compared to ICA, it was decided to first reduce the dimensionality using PCA from N to k, and then perform a decomposition into independent groups using ICA (from k to k). This way, both methods can be used where they perform best. In short,

$$\text{PICA is: } X\ (N \times T) \overset{PCA}{\Longrightarrow} X'\ (k \times T) \overset{ICA}{\Longrightarrow} Y\ (k \times T).$$

Here X is the input data and Y the reduced representation as normal. Because both of these methods perform linear transformations, the transformation resulting from their combination will also be linear.

All four of these methods were applied to the 'test' data set, in order to reconstruct the group structure graph $\mathcal{G}$ given by the proximity-based method. Their performance was measured in terms of precision and recall, and additionally an F_1 score was calculated. Both similarity measures (correlation and NMI) were tested separately.

Detailed results are available in Appendix A where both precision and recall are displayed as functions of N for each method/similarity measure combination. There it can be seen that both correlation and NMI produce comparable results, except in the case of dimension grouping. For that reason, from here on only

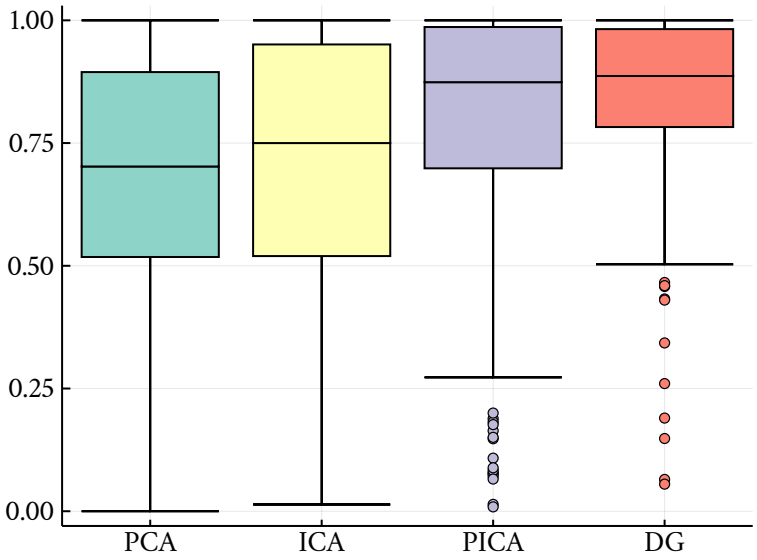

Figure 9: The distribution of F_1-scores in the grouping task for each dimensionality reduction method. Here the whiskers extend to 1.5 times the IQR beyond the first/third quartile, and the rest of the data are plotted as outliers.

the results using the NMI to compare trajectories are considered.

A summary of the results is given in Table 1, and the distribution of F_1-scores for each method is displayed in Figure 9. The performance of each of the methods as a function of N is displayed in Figure 11.

The SOM was found to only be computationally feasible if the output dimension (the dimension of the map) was at most 3. To test whether it would make any sense to use a SOM for data sets with a small number of groups, an additional data set was produced by collecting simulations where the number of groups was found to be $k = 2$. For this data set, the SOM was compared to PCA, ICA and their combination. These results are visible in Figure 10 and Table 2.

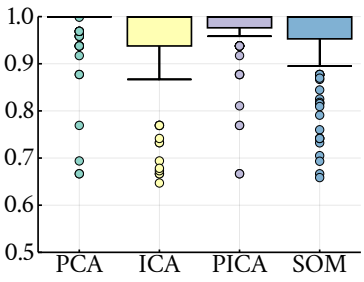

Figure 10: The distribution of F_1-scores in the grouping task for a number of dimensionality reduction methods, performed on the data set where the number of groups is constrained to $k = 2$. Here the whiskers extend to 1.5 times the IQR beyond the first/third quartile, and the rest of the data are plotted as outliers.

6 Discussion & further work

6.1 Counting groups.

From the results of the experiments on the Kuramoto data set, we can draw some conclusions. First of all, there is an agreement between the dimensionality of the data set as estimated by PCA and the number of groups according to the proximity-based definition. PCA has some off-by-one errors, but these may be the result of very slight differences, due to the discretization applied when deciding whether to place an edge between two vertices in the graph based on the probability matrix P. Because PCA finds principal components which are mutually uncorrelated, this indicates that the notion of "groups" based on

Method	Precision	Recall	F_1
PCA	0.8413 ± 0.0091	0.639 ± 0.015	0.644 ± 0.014
ICA	0.8687 ± 0.0082	0.672 ± 0.015	0.702 ± 0.013
PICA	0.8772 ± 0.0077	0.801 ± 0.011	0.8089 ± 0.0097
DG	0.9573 ± 0.003	0.8103 ± 0.0089	0.8613 ± 0.0065

Table 1: Performance results of the various dimensionality reduction methods in the task of grouping. For each method the mean values over the entire data set of the precision, recall and the combined F_1 score are displayed. The uncertainty measured is the *standard error of the mean*.

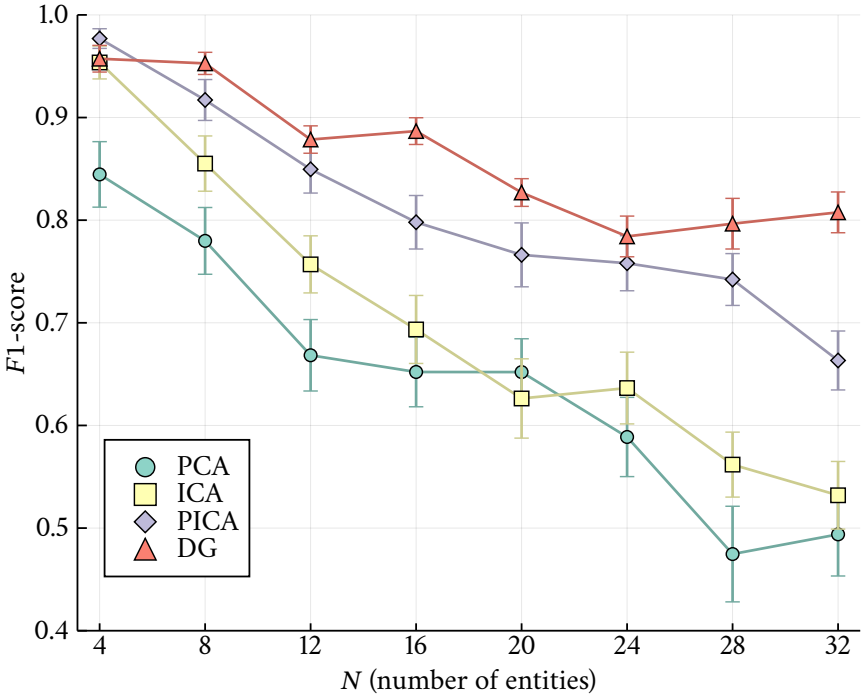

Figure 11: The mean F_1-score in the grouping task for each dimensionality reduction method as a function of the number of entities N. The error bars indicate the uncertainty, quantified as the *standard error of the mean*.

Method	Precision	Recall	F1
PCA	0.9749 ± 0.0067	1.0 ± 0.0	0.9851 ± 0.0043
ICA	0.9418 ± 0.0086	0.9816 ± 0.0068	0.9562 ± 0.0068
PICA	0.9631 ± 0.007	0.99922 ± 0.00078	0.9786 ± 0.0044
SOM	0.9686 ± 0.0056	0.9513 ± 0.0096	0.9564 ± 0.0071

Table 2: Performance results of various dimensionality reduction methods in the task of grouping, performed on the data set where the number of groups is constrained to $k = 2$. For each method the mean values over the entire data set of the precision, recall and the combined F_1 score are displayed. The uncertainty measured is the *standard error of the mean*.

distance agrees with the notion of "groups" of entities as sets of entities with correlated trajectories. However, this is not an universal observation: it has only been shown for this particular system. One possibility is to derive an implication from this observation: if these two notions of groups seem to align for some data set, it indicates that grouping via dimensionality reduction might be feasible. Testing this for several other data sets, especially non-synthetic data sets and data sets with a two- or three dimensional entity space, would be a good intial experiment to establish the general feasibility for this dimensionality reduction-based approach to the trajectory grouping problem.

Also, it is clear that the ICA algorithm[27] does not perform well when asked to 'rank' components. This is to be expected, as the algorithm is designed to handle systems with N signals and N unknown sources. If there are more observed signals than independent components, the algorithm will try to produce additional independent sources and will end up creating superfluous 'sparse' components (Hyvärinen et al., 1999). However, because both PCA and ICA are linear transformations and easily interpretable, they can be combined well. The result is a linear transformation which produces statistically independent components like ICA, but with the addition that any 'sparse' components are first filtered out by PCA, leading to a more robust procedure which allows for ICA to be applied to overdetermined systems as well. This combination of PCA and ICA, here called PICA, has been known to be useful for some time (Hyvärinen et al., 1999), but in practice has led to mixed results. See Artoni et al. (2018) for an example where the application of dimensionality reduction via PCA was found to have adverse affects on the ICA decomposition.

[27] At least, the ICA algorithm as applied here.

6.2 Determining group structure.

In the task of reproducing the group structure graph $\mathcal{G}$ none of the methods seem to be very reliable: in each case there were some runs of the simulation for which the F_1 score was quite low (say, below 0.5). Therefore no firm conclusions about the performance of these algorithms can be established. The more detailed figures in Appendix A show that low performance can mostly be attributed to a low recall. This means that given these parameters (β_{cor} and β_{NMI}), the generalized grouping algorithm tends to be 'conservative' in grouping entities together. However, it is not simply the case that the precision has been overemphasized to the detriment of recall, as the precision also fell below 50% for each of the methods except for dimension grouping. Therefore the reliability cannot be improved much by simply increasing the 'certainty'

parameter β.

Because in the case of $k = 2$ groups all algorithms performed very well (see Table 2), the SOM seems to not be a useful algorithm in this context. It is encouraging that in the case of various k (Table 1), going from the most general (PCA) to the most specialized algorithm (DG) seems to increase the performance in terms of all measures applied. This shows that at least each specialization or addition applied is worthwhile.

The algorithms each have certain properties which make them useful in the context of this task, and possibly in other contexts as well. The combination of PCA and ICA is a nice alternative to regular PCA when its dimensionality reduction properties are needed, but when the uncorrelatedness of components is not meaningful or not a strong enough condition to make the resulting components independent in a meaningful way. The statistical independence aimed for by ICA may be useful in a larger variety of situations when trying to distinguish between important components of a data set. The dimension grouping algorithm also has some advantages of its own. Because of its roots in the SOM, the algorithm is quite flexible: it can function given an arbitrary distance metric between entities and it is not limited to only linear relations between entities like PCA or ICA.

6.3 Related problems and further research.

Larger number of entities. In the present study, only simulations with a number of entities N up to 32 were tested. From the results, it can be seen that in every case the performance on the grouping task decreases as N increases. Again this seems to be because of declining recall, as the precision does not show clearly show such a trend. This means that it might be improved if the certainty parameter β is increased as a function of N. In any case, it would be worthwhile to test these algorithms for data sets with larger N as well to quantify this relationship properly.

Central trajectories. Related to the trajectory grouping problem is the task of finding *central trajectories*: trajectories which are representative of a group as a whole (Kreveld et al., 2017). When performing dimensionality reduction, this problem is implicitly also treated, as the output space of the DR algorithm can be considered a space in which these central trajectories live. However, the various methods fill this space in a different way. The axes determined by PCA follow a sequence of optimal 'fits' to the data, which makes

the axes not interpretable as central trajectories[28]. From the first experiment, we expect that central trajectories are contained in the space of the first k PCs, but we cannot locate them easily.

The addition of ICA helps with this: because the resulting components are maximally independent, it would be disadvantageous if one group would be shared by two components, as there would then be some dependence. The axes of the PCA+ICA reduction might therefore be more suitable as central trajectories. However, each central trajectory will be a linear combination of entities, which is discouraged by Kreveld et al. as this might lead to 'impossible' trajectories.

The dimension grouping algorithm has a more direct approach to this problem, as the output trajectories are explicitly modelled as group representatives. Moreover, it provides the flexibility to control how a good representation is determined. Currently a good representative is one that minimizes the sum of squared distances to the group members, which means it produces the *centroid* of the group through iteration. This could for example be changed to the *spatial median* by removing the square in the energy function.

A better choice might be the *medoid*, where the central point is constrained to be equal to one of the group members. To incorporate this constraint is more difficult, as it does not suggest a clear continuous objective function, but there may be ways to adjust the energy function such that representatives very close to the original trajectories are strongly preferred. One example of this might be to parametrize the representatives in terms of the original trajectories, and apply appropriate regularization to the parameters. For example, one might choose a (constant) linear parametrization

$$s_i[t] = \sum_{j=1}^{N} B_{ij}\, x_j[t],$$

where we have introduced the parameters B_{ij} with $1 \leq i \leq k, 1 \leq j \leq N$. These parameters may then be adjusted in order to minimize the regular DG objective combined with an additional regularization term which promotes configurations where only *one* B-value is nonzero for any given i, i.e. where each representative is (approximately) given by a single trajectory. This allows one to obtain a set of representatives which are (approximately) the medoids of their respective groups.

To allow *which* trajectory is chosen as group representative to vary over time (which may be useful even if the group structure does not change, but another

[28] For example, the first component provides an overall 'average' of all entities, and therefore incorporates all groups, which means it cannot be considered a central trajectory of any one group. This applies to the second component as well, and so on.

The centroid of the set $X = \{x_i\}$, $1 \leq i \leq N$, $x_i \in M$ is the point $y \in M$ which minimizes

$$\sum_{i=1}^{N} d(x_i, y)^2.$$

The spatial median of X is the point $y \in M$ which minimizes

$$\sum_{i=1}^{N} d(x_i, y).$$

The medoid of X is the point $x \in X$ which minimizes

$$\sum_{i=1}^{N} d(x_i, x).$$

group member travels closer to the center), one can make the B_{ij} coefficients time-dependent as well. However this introduces many more degrees of freedom[29] and so extra regularization needs to be applied in order to retain a meaningful interpretation of the coefficients and the trajectories which are chosen as representatives.

Time-varying group structure. One assumption which is made here is that the group structure is constant over the time interval on which analysis is performed. This is generally not the case or at least not known in advance, and in fact the moments when groups change are often especially interesting. The most straightforward way to keep track of a changing group structure when using dimensionality reduction would be to divide the trajectory into 'frames' of some number of time steps δ, and extract the group structure from each frame separately. When applying dimensionality reduction, more than one instance in time needs to be considered to find a proper embedding, so there will necessarily be a limited temporal precision with which the group structure may be determined. In Buchin et al. (2013), a temporal 'stability' parameter δ is used to ensure that groups do not respond to very quick changes, which shows that this finite resolution is actually desirable. However, if the resolution is too high (i.e. δ is small) the quality of the embedding might drop, or it might differ wildly over consecutive frames which would make it hard to track groups over time. For this problem, the dimension grouping algorithm could possibly be applied in a 'layered' fashion, to ensure consistency of groups between frames.

[29] For example, if the coefficients are time-dependent and the trajectories live in a one-dimensional space, one can store an entire representative trajectory in the coefficients of one variable. As such, any trajectory s_i can be obtained by simply multiplying any other (nonzero) trajectory x_j with precisely chosen coefficients

$$B_i j[t] = \frac{s_i[t]}{x_j[t]}.$$

Applying a 'sparsity' constraint such as ℓ_1-regularization will then simply always choose the trajectory x_j with the biggest value t as the representative at t, so that the coefficient needed to obtain s_i will be as small as possible.

Bibliography

Adali, T., Schreier, P. J., and Scharf, L. L. (2011). Complex-Valued Signal Processing: The Proper Way to Deal With Impropriety. *IEEE Transactions on Signal Processing*, 59(11):5101–5125.

Altis, A., Nguyen, P. H., Hegger, R., and Stock, G. (2007). Dihedral angle principal component analysis of molecular dynamics simulations. *The Journal of Chemical Physics*, 126(24):244111.

Amari, S., Karakida, R., and Oizumi, M. (2018). Information geometry connecting Wasserstein distance and Kullback–Leibler divergence via the entropy-relaxed transportation problem. *Information Geometry*, 1(1):13–37.

Artoni, F., Delorme, A., and Makeig, S. (2018). Applying dimension reduction to EEG data by principal component analysis reduces the quality of its

subsequent independent component decomposition. *NeuroImage*, 175:176–187.

Bottou, L. and Bengio, Y. (1994). Convergence Properties of the K-Means Algorithms. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press.

Bridle, J. S. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In *Neurocomputing*, pages 227–236. Springer Berlin Heidelberg.

Buchin, K., Buchin, M., van Kreveld, M., Speckmann, B., and Staals, F. (2013). Trajectory grouping structure. In Dehne, F., Solis-Oba, R., and Sack, J., editors, *Algorithms and Data Structures, WADS 2013*, volume 8037 of *Lecture Notes in Computer Science*, pages 219–230. Springer Berlin Heidelberg.

Cottrell, M., Olteanu, M., Rossi, F., and Villa-Vialaneix, N. (2016). Theoretical and Applied Aspects of the Self-Organizing Maps. In *Advances in Self-Organizing Maps and Learning Vector Quantization*, pages 3–26. Springer International Publishing.

Couzin, I. D. (2009). Collective cognition in animal groups. *Trends in Cognitive Sciences*, 13(1):36–43.

Demartines, P. and Herault, J. (1997). Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154.

Fisher, N. I. (1993). Descriptive methods. In *Statistical Analysis of Circular Data*, pages 15–38. Cambridge University Press.

Heskes, T. (1999). Energy functions for self-organizing maps. In *Kohonen Maps*, pages 303–315. Elsevier.

Hyvärinen, A., Särelä, J., and Vigário, R. (1999). Spikes and bumps: artefacts generated by independent component analysis with insufficient sample size. In *Proc. Int. Workshop on Independent Component Analysis and Blind Separation of Signals (ICA'99)*, Aussois, France.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430.

Jammalamadaka, S. R. and Sengupta, A. (2001). *Topics in Circular Statistics*. World Scientific.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Kreveld, M. V., Löffler, M., and Staals, F. (2017). Central trajectories. *Journal of Computational Geometry*, Vol 8:No 1 (2017)–.

Langlois, D., Chartier, S., and Gosselin, D. (2010). An introduction to independent component analysis: Infomax and fastica algorithms. *Tutorials in Quantitative Methods for Psychology*, 6(1):31–38.

Lee, D. and Liang, S. H. L. (2011). Crowd-sourced carpool recommendation based on simple and efficient trajectory grouping. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science - CTS '11*. ACM Press.

Maistrenko, Y. L., Popovych, O. V., and Tass, P. A. (2005). Desynchronization and Chaos in the Kuramoto Model. In *Lecture Notes in Physics*, pages 285–306. Springer-Verlag.

Nergiz, M. E., Atzori, M., and Saygin, Y. (2008). Towards trajectory anonymization. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS - SPRINGL '08*. ACM Press.

Pajunen, P. (1996). Nonlinear Independent Component Analysis by self-organizing maps. In *Artificial Neural Networks — ICANN 96*, Lecture Notes in Computer Science, pages 815–820, Berlin, Heidelberg. Springer.

Pellegrini, S., Ess, A., and Gool, L. V. (2010). Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings. In *Computer Vision – ECCV 2010*, pages 452–465. Springer Berlin Heidelberg.

Phang, T., Neville, M., Rudolph, M., and Hunter, L. (2002). Trajectory clustering: a non-parametric method for grouping gene expression time courses, with applications to mammary development. In *Pac Symp Biocomput. 2003*, pages 351–362. World Scientific.

Shlens, J. (2014). A Tutorial on Principal Component Analysis. http://arxiv.org/abs/1404.1100v1.

Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: A comparative review. *J Mach Learn Res*, 10(66-71):13.

Zarzoso, V. and Comon, P. (2010). Robust Independent Component Analysis by Iterative Maximization of the Kurtosis Contrast With Algebraic Optimal Step Size. *IEEE Transactions on Neural Networks*, 21(2):248–261.

List of concepts

Appendices

A Additional figures of results

In this appendix additional results on the performance of each dimensionality reduction (DR) method in the task of grouping are displayed. For each method, the precision and recall compared to the proximity-based method have been measured. These are displayed as a function of increasing number of entities N in Figures 13-16. Results are reported using both similarity metrics, correlation and normalized mutual information (NMI). Because these figures employ box plots, an overview of how to read these plots is given in Figure 12.

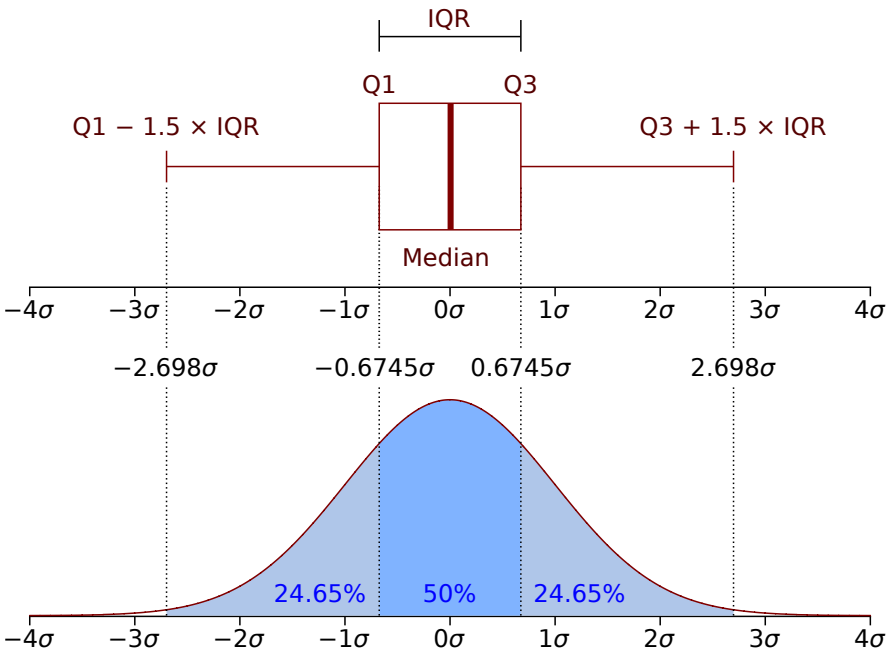

Figure 12: An overview of how to read a distribution visualized by a box plot. The box extends from the first quartile (Q1, 25%/75% split left to right) to the third quartile (Q3, 75%/25% split), with the median marked by a thick line. Then, the 'whiskers' of the plot are lines which extend beyond the box to mark either all of the other data (from minimum to maximum), or up to a point determined by the interquartile range IQR = Q3 - Q1. In the second case, any data outside this range may be considered outliers and are plotted using regular markers. For a normal distribution, the corresponding probability mass percentages are displayed below and in terms of standard deviations σ from the mean. This image has been adapted from Wikipedia under CC BY-SA 2.5 and was created by user Jhguch (`https://commons.wikimedia.org/w/index.php?curid=14524285`).

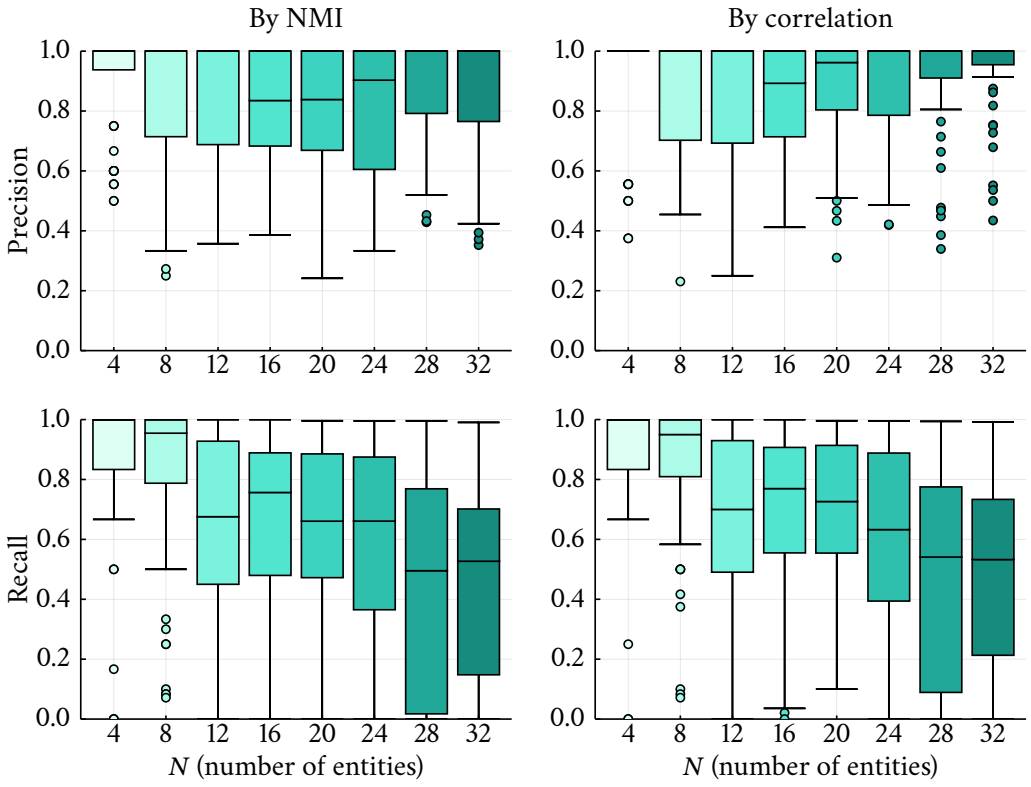

Figure 13: The performance of grouping using PCA in terms of precision and recall as a function of increasing N.

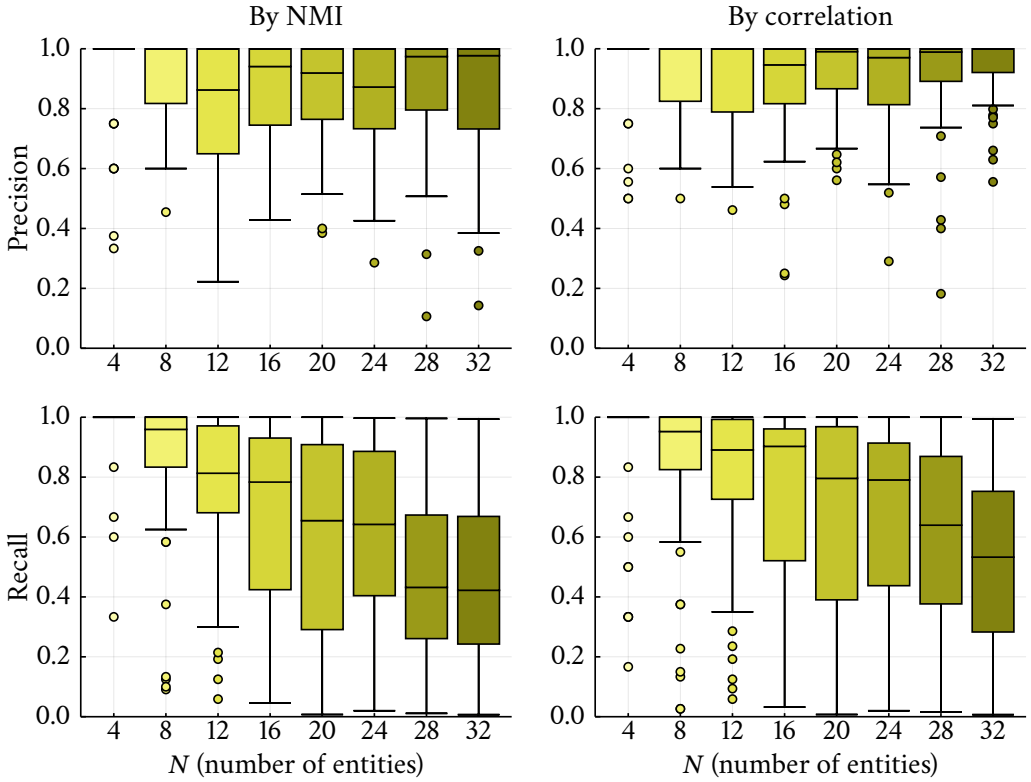

Figure 14: The performance of grouping using ICA in terms of precision and recall as a function of increasing N.

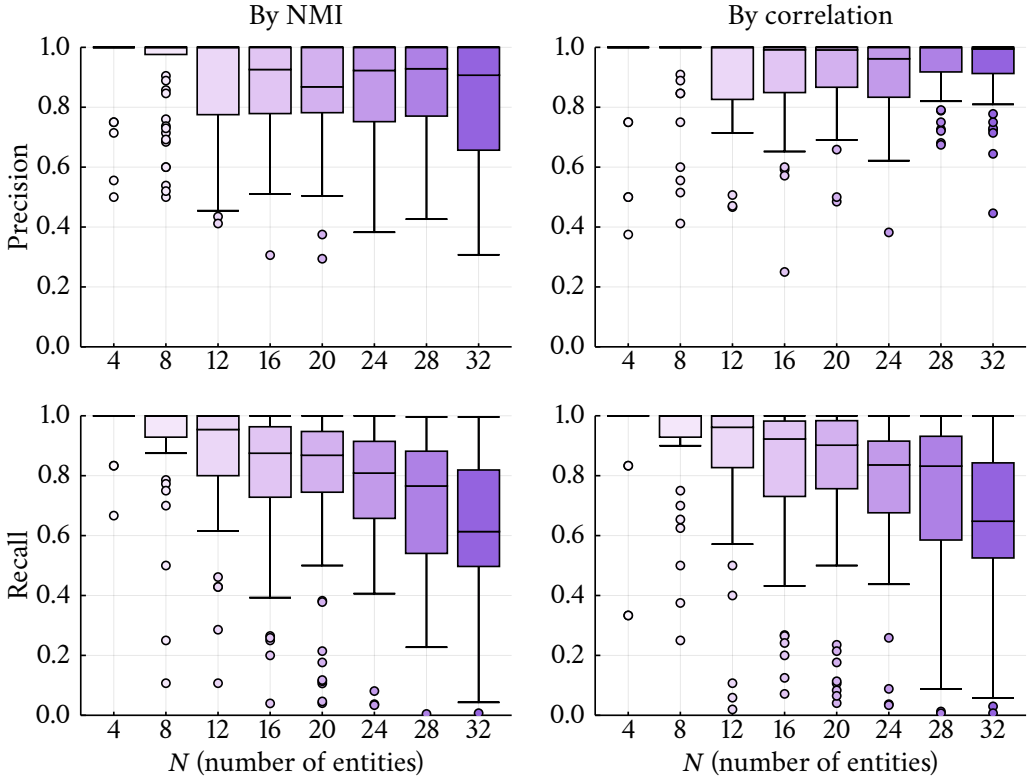

Figure 15: The performance of grouping using PICA in terms of precision and recall as a function of increasing N.

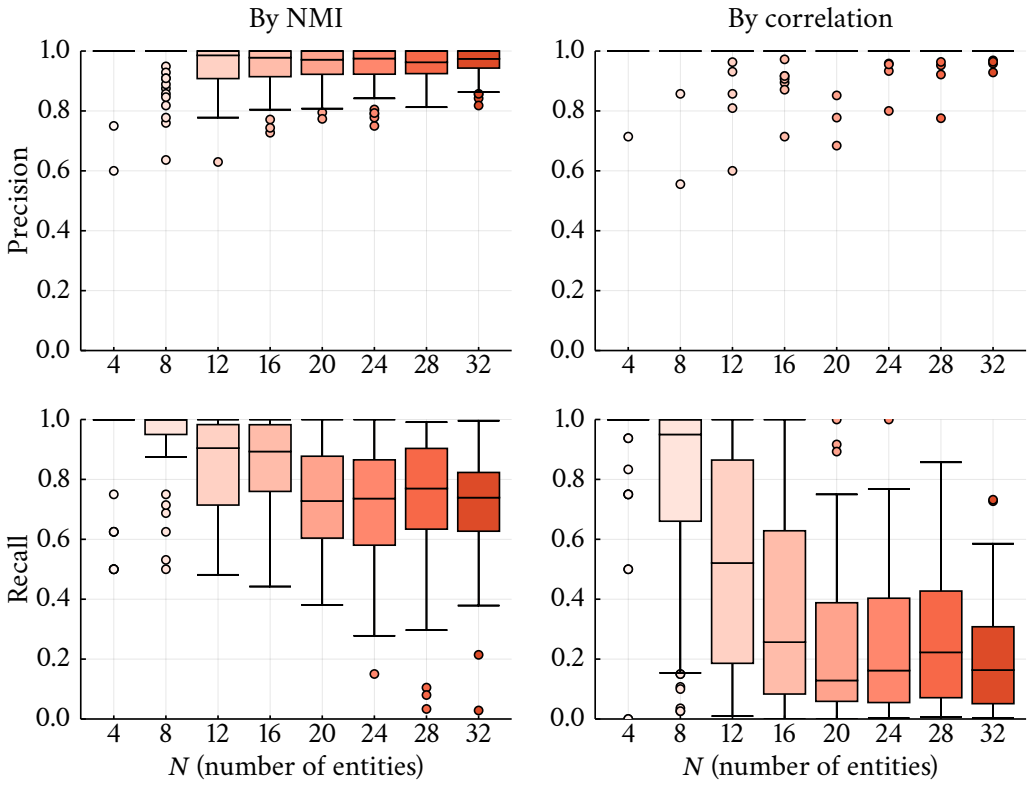

Figure 16: The performance of grouping using DG in terms of precision and recall as a function of increasing N.

B Angular and complex PCA

In order to apply PCA to a data set of angles, we have performed a transformation to complex numbers. This transformation is motivated by the fact that statistical notions such as (co)variance are not well-defined for angles. The main reason for this is the possibility of "wrap-around": data points can be very far apart in numerical value, while being very close in geometrical distance. Take for example two angles of 5° and 355°. The numerical difference is 350°, while their actual distance is only 10°. This makes it difficult to define a good metric on the circle without (implicitly) transforming to a different metric space such as $\mathbb{R}^2$ or $\mathbb{C}$

The ambiguity arises because of the rotational (or translational, in terms of angles) symmetry of the circle, so that any choice of an origin (the point which is 0° or zero radians) is arbitrary. As such, there is no clear "correct" way to define a mean angle of points on the circle. When given an uniformly distributed set of angles, no choice is intuitively more right than another. Take the mean angle between 90° and 270°. Should it be 0° or 180°, and why?

A common method to define the mean of a set of angles $\{\theta_k\}$ is through the sum of their vectorial counterparts (Fisher, 1993; Jammalamadaka and Sengupta, 2001). This corresponds to taking the sum of complex numbers $z_k = \exp(i\theta_k)$. Then, the angle of their mean $\overline{z}$ gives a reasonable mean angle in most cases. For a uniform distribution of angles, the magnitude of $\overline{z}$ will go to 0, and so the mean angle can be said to be undefined.

So to perform PCA with our angular data, the angles $\{\theta_k\}$ are first converted to complex numbers with unit magnitude $\{z_k\}$. Then, the different parts of the PCA process each differ a bit compared to regular PCA, and will be considered here one-by-one.

Mean centering. PCA expects the data to be mean-centered before analysis, or alternatively to first center the data before transforming to the PC-space. This is because in calculating (co)variance, the distance between samples and the mean is considered instead of the sample values themselves. For this, the complex mean can be used, which corresponds to the angular mean as discussed before. However, most of our data series are approximately mean centered already, as they fully rotate around the unit circle an arbitrarily large number of times. This means they are approximately uniformly spread along the unit circle, and as such the mean tends to $z = 0$.

A reference for a similar application of complex PCA to angular data is Altis et al. (2007). Also interesting for those who would like to perform PCA with complex inputs is the article by Adali et al. (2011). Here the similarities and differences between linear transformations in $\mathbb{C}$ and $\mathbb{R}^2$ are discussed. It turns out that linear transformations in $\mathbb{C}$ are more restrictive than those in $\mathbb{R}^2$. Because we are trying to establish what complex linear transformations mean in S^1, which is even more restrictive than $\mathbb{R}$, the projection to $\mathbb{R}^2$ is not so useful here.

Therefore, it is assumed here that the complex version of our angular time series will have a mean $\overline{z} \approx 0$, and the data is not mean-centered when performing PCA. This has the added benefit of simplifying the interpretation: the additive term b in the linear transformation $y = ax + b$ is assumed to be zero, and so the data will not be translated relative to the origin, which means the principal components must also lie on a circle of fixed radius around the origin. Therefore we may in the end simply calculate the angles of the principal components with respect to the origin to retrieve a meaningful angular representation.

Covariance matrix. In the case of complex numbers, the covariance matrix is a bit harder to interpret. First, note that the variance of a set of unit complex numbers with mean $\overline{z} = 0$ is always equal to 1. This can be show as follows.

The covariance is defined as:

$$\operatorname{cov}(x,\ y) = E[(x - \overline{x})\,(y - \overline{y})^{*}]$$

where $\mathrm{E}[...]$ is the expected value and $\overline{x}$, $\overline{y}$ are the means of x and y. If $z_k = \exp(\mathrm{i}\theta_k)$ and $\overline{z_k} = 0$, then

$$\operatorname{cov}(z_i, z_j) = \mathrm{E}\left[z_i z_j^{*}\right] = \mathrm{E}\left[\exp(\mathrm{i}\theta_i) \cdot \exp(-\mathrm{i}\theta_j)\right] = \mathrm{E}\left[\exp(\mathrm{i}[\theta_i - \theta_j])\right]$$

and

$$\operatorname{var}(z_k) = \operatorname{cov}(z_k,\ z_k) = \mathrm{E}\left[\exp(i[\theta_k - \theta_k])\right] = \mathrm{E}[1] = 1.$$

Furthermore, in linear regression the slope α of the best-fit linear relationship $z_i = \alpha z_j + \beta$ can be expressed as:

$$\alpha = \frac{\operatorname{cov}(z_i, z_j)}{\operatorname{var}(z_j)}.$$

Because $\operatorname{var}(z_j) \approx 1$ in our case, that means $\operatorname{cov}(z_i, z_j)$ is equal to the slope (linear coefficient) α of the (best-fit) linear relationship between z_i and z_j. In the case of complex numbers, this will also be a complex number $\alpha = r\mathrm{e}^{\mathrm{i}\phi}$. This multiplication can then be interpreted as a transformation consisting of a scaling factor r and a rotation by an angle ϕ.

Correlation. Normally, the covariance can be used to calculate Pearson's correlation coefficient which shows whether a linear relationship is present:

$$\operatorname{corr}(x,\ y) = \frac{\operatorname{cov}(x,\ y)}{\sqrt{\operatorname{var}(x) \cdot \operatorname{var}(y)}} \in [-1,\ 1].$$

Here x, y are real-valued. If this same formula is applied to the unit-complex

numbers z, this becomes

$$\operatorname{corr}(z_i, z_j) = \frac{\operatorname{cov}(z_i, z_j)}{\sqrt{1 \cdot 1}} = \alpha,$$

but this value cannot straightforwardly be interpreted as an indicator of a linear relationship, as it is a complex number which cannot be ordered to contrast a strong relationship from a weak one. But because our data is constrained to lie on the unit circle, a clearer interpretation can be given.

Suppose we have an approximate linear relationship $z_i = \alpha z_j$ with a value of $\alpha = re^{i\phi}$ with $r < 1$ or $r > 1$. This means that transforming z_j to z_i or vice versa, which would require multiplying by α, would amount to scaling the data such that it moves it off the unit circle. Because z_i and z_j both lay on the unit circle, a scaling should not necessary, and since this is the best-fit linear relation between the two the only possible conclusion is that they are simply not very well modelled by a linear relationship. As such, we can take the magnitude $\|\alpha\| = r$ as a measure of whether two angular variables are linearly relatable: if it is close to 1, the two are likely well correlated and vice versa. Note that this aligns with the notion of $R^2 = \operatorname{corr}(z_i, z_j)^2$ as a measure of goodness-of-fit in linear regression.

Interpretation of linear transformations. Through the previous steps we have established a number of extra assumptions or constraints on the complex linear transformation relating the unit-complex versions of our angular variables. In the end, we aim to find a *unit-complex linear transformation* of the form

$$z_i = \alpha z_j = re^{i\phi} z_j$$

which maps some z_i and z_j onto each other. Moreover, if the two are linearly relatable we expect that $\|\alpha\| \approx 1$. As such, the transformation is essentially parametrized by only one number, the rotation $\phi \in \mathbb{R}$.

This means that a unit-complex linear transformation essentially amounts to a rotation in complex space, or a translation in angular space, and as such is more restrictive than a real-valued linear transformation. This makes sense in the context of the circle: translations by some angle are meaningful, but because there is no good choice for an origin angle, there is nothing relative to which scaling may be performed, and so there is no meaningful way to perform a 'scaling' transformation on angular data.