Unsupervised alignment of conceptual systems

Exploitation of local structure information

Roseline Polle[1]

Msc Machine Learning

Supervised by: Bradley C. Love
Co-supervised by: Brett D. Roads

Submission date: 11th September 2020

[1]**Disclaimer:** This report is submitted as part requirement for the Msc Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

This works looks at how two conceptual systems can be aligned in an unsupervised way. Concepts are here represented by two systems of D-dimensional points that show structural similarities, and we aim to find an algorithm that finds the correspondences between the two systems without the use of labels or of any known correspondences. The project is motivated by the way humans learn concepts using observed statistics from the world, and overlaps with a number of fields such as Cognitive Science, Philosophy, Neuroscience and Computer Science. A baseline algorithm inspired from the image-to-image translation literature is first tested on the datasets, showing some but limited success.This approach is particularly sensitive to the initialization strategy, with naive initialization typically yielding solutions that become stuck in local minimums. We address this problem by adding a pre-training step that leverages information about the systems' internal structures. Local features are estimated for each point, allowing us to compute points similarities between systems. A subset of matching points are selected based on these similarities and used to pre-train the models as if they were true correspondences. This method shows significant improvements over the baseline method. It deals notably well with concept's dimensions of up to ten, where the previous method performed poorly. It also increases success for all levels of noises explored, despite being sensitive to added noise. These results are a promising step towards showing how the brain may use the statistics from the environment to build and align conceptual systems.

Acknowledgements

I would like to express my special thanks to Brett Roads, for the constant support and guidance throughout the project. It has been a great learning experience and I am grateful for all the time and feedback he has given me. I would also like to thank Professor Brad Love for giving me the opportunity to work within his lab, as well as for his very precious feedback. Finally, thanks to my family, friends and partner for bearing with me as the deadlines were approaching.

Contents

Chapter 1

Introduction

In this thesis, we are interested in finding algorithms that align *concepts* from different *conceptual systems* based on their internal structure only, without using supervised labels. We therefore focus on the class of unsupervised algorithms. The project is motivated by Cognitive Science, and more specifically the way human learn about the world. Although it is commonly accepted that some level of supervision is required for human to learn, there is some evidence that infants are able to leverage the statistics observed in the world to make associations and better generalize about the concepts they learn. Through this work we aim to explore the hypothesis that human may align the information from different modalities (e.g. vision, language) without the use of labels during concept acquisition. This is quite different from the way computers have been traditionally trained to learn, i.e. with tons of examples and associated labels, and there is active research in various computer science domains looking at replacing supervised algorithms with more unsupervised methods. One can refer to Figure 1.1 to visualise the difference between supervised and unsupervised learning within the context of our project. This Figure is further introduced in the next paragraph.

The scope of the problem is large, and there are several questions we need to consider to define it further. The first questions we need to ask are: what are concepts, what is a conceptual system and how will we choose to represent them? Chapter 2 (*Related work in Philosophy and Cognitive Science*) is mostly dedicated to answering these questions, but we will give a very basic first answer now to continue defining our problem in more details: concepts can roughly be thought as a mental representation that describes a coherent and relatively discrete aspect of reality. Similar to Plato's statement, concepts carve a contiguous reality at the joints. A conceptual system is a system composed of interrelated concepts, for instance within a cognitive agent. In this thesis, it is represented by a set of (potentially high-dimensional) embeddings, and the relationships between concepts are therefore reduced to the euclidean distance between them. We work with synthetic data exhibiting a relatively simple structure. A first random set of vector of a fixed chosen dimension is generated from a Gaussian or mixture of Gaussians, and constitute the first conceptual system. A new conceptual system is created by adding some noise and a random rotation to each vector. Eventually -but outside the scope of this thesis- we aim to work with more complex conceptual systems, represented by embeddings derived from text data (Common Crawl text corpus), image data (Open Images dataset) and audio data (Audioset dataset). Co-occurence statistics are collected from each domain (using a text corpora for the text dataset, and labels for the image and audio data), and the GloVe algorithm is then applied to derive the different sets of embeddings. These datasets are described in more details by Roads and Love [1], as well as in our literature review. Figure 1.1 is taken from Roads and Love's paper [1] and illustrates the supervised versus unsupervised learning mechanisms in the context of this dataset. Their work showed that relational structure can be used to determine mappings between embeddings. However, their work did not explore specific algorithms for performing alignment. This paper and its promising results are the main basis for this thesis. The idea here is to test promising algorithms on a simpler structure before generalising them to more complex ones.

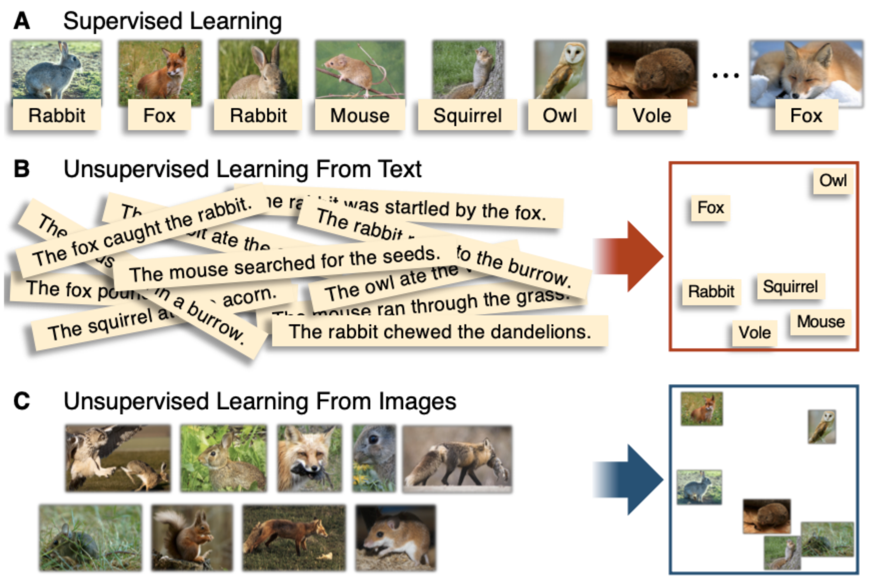

A Supervised Learning
Rabbit
Fox
Rabbit
Mouse
Squirrel
Owl
Vole
Fox
B Unsupervised Learning From Text
The fox caught the rabbit.
The mouse searched for the seeds.
The owl ate the
The rabbit chewed the dandelions.
The squirrel
The fox
Owl
Fox
Rabbit
Squirrel
Vole
Mouse
C Unsupervised Learning From Images

Figure 1.1: Figure and associated caption from Roads and Love's paper [1]. "Different modes of learning. (A) In supervised learning, feature representations are learned by co-presenting both an image and a label. Alternatively, unsupervised feature representations (i.e., embeddings) can be learned from (B) text and (C) images. The two unsupervised embeddings capture two viewpoints of the same underlying reality and therefore exhibit similar structure. Images shown here are from the public domain (Wikimedia Commons)."

The second type of questions we need to ask are: what is the scope of the problem in this thesis? The amount of knowledge an individual acquire in their life is huge, and typical embeddings in Computer Science have dimension in the hundreds. We eventually aim to be able to work with such data, but our goal is to first test promising ideas using simplified assumptions on our dataset. Working on a simpler problem allows us to explore different ideas, and successful algorithms can then be developed and adapted to fit the more complex setting. The first assumption that we made and that we already introduced is to work with synthetic data of a chosen dimension. This synthetic data has a simple "controllable" structure, allowing to experiment with different scenarios. We also introduce other simplifying assumptions. Indeed in the final problem that we are interested in, there might be concepts that belong to a system and not to another, or a many-to-one mapping. Additionally, each conceptual systems could have different dimensions, therefore requiring the discovery of a common latent space. In this thesis, we take the following decisions to reduce the scope of the problem:

1. Synthetic data of up to 10 dimension is used.
2. Conceptual systems have the same dimensions, hence latent space discovery and projection onto that space is not required.
3. We limit ourselves to systems where a one-to-one mapping exists, and the concepts comprised in the different sets of embeddings are exactly the same (no missing ones in each system).

When considering various algorithms in this scope, we need to keep in mind the final aim. Hence, computational efficiency and the possibility to scale the problem to many concepts and high-dimensional data will be important. Unsupervision is also a key aspect of any contender algorithm. We also need to be wary that in the final problem, not every concept will have exactly one correspondence: it might have none, or several. Finally, the dimensions of the embeddings might not be the same in each system. Note that we choose to tackle the problem starting from an easy structure, where we control the transformation between sets of embeddings, to a more complex structure which appear much more as a black box. Embeddings created using co-occurence statistics -or other sorts of static and dynamic embeddings we could have chosen to work with (e.g. Word2Vec, Bert) - might have their own specificities that are not observable using our synthetic examples. Notably, could each dimensions in the final embeddings be "interpretable" and have meaning? It is clear that this is not true in our synthetic data, but unclear for the full dataset. Another approach to the problem could be to already start from this more complex structure, and find algorithms with some level of supervision that we would look to make more and more unsupervised.

Now that the scope of the problem for this thesis is defined, we present our contribution. An algorithm largely inspired by CycleGAN [50] was initially developed within the Love lab at UCL to solve our alignment problem. We will refer to it as the baseline algorithm. CycleGAN is an algorithm that performs image-to-image translation by projecting an image from a source domain to a target domain (e.g. a landscape into a Monet painting). It is further described in Section 3.2.3 and the baseline algorithm in Section 4.2. Although this baseline algorithm sometimes manages to align the two systems of points, it often gets stuck in local minimums and success seems highly dependent on the random initialisation. To solve this issue, we propose to pretrain the model using local structure features largely inspired from the Manifold Alignment literature ([38],[40]). Local structure features capture the relationship between a point and its neighbours. Because our two domains share structural similarities, we hope that incorporating this additional information will improve the algorithm. A number of surrogate points are matched based on the similarity of their local structure features, and used as if they were true labels in the pretraining step. The baseline algorithm is then used on the pretrained model in the same way as before. A number of scenarios with varying parameters are compared for both the baseline and the three pretraining strategies that we will define in the Method section.

Note in the orignal CycleGAN paper, the local minimums issue was not raised and this seems to be specific to our problem. There are a number of potential reasons for this, which may stem from the differences between the image-to-image translation problem and ours. One notable difference is that in CycleGAN, each input is an image, i.e. a high-dimensional vector where each dimension (pixel) is correlated to other pixels around. In our problem, each input is a D-dimensional vector where dimensions are independent (as our data is synthetic and randomly generated). If we were to use embeddings derived from images or texts directly as our data, the correlation between dimensions would be less clear, but is also generally considered uninterpretable [2]. These differences are also the reason why the CycleGAN algorithm was not used as such but adapted. Notably we use dense networks instead of Convolutional Neural Networks (CNN), and the adversarial loss is replaced by a distribution loss. One may refer to the Related Work and Method sections for more details on the two algorithms (3.2.3 and 4.2)

Chapter 2

Related work in Philosophy and Cognitive Sciences

In this section, we look at concepts and learning from a philosophical and neurobiological point of view. We first explore the different theories of concepts in Philosophy to get a deeper understanding of what is thought by "concepts", and how they relate to each other. We then link some of these theories with some theories of meaning (still in Philosophy) with similar frameworks. Finally we give a quick sense of how concepts are thought to be formed and stored in the human brain according to Neuroscience. Through these overlapping fields, we hope to make clearer the cognitive motivations for exploring these algorithms.

2.1 Philosophical considerations

2.1.1 Theory of Concepts

The idea of concepts is extensively studied in the field of Philosophy and Cognitive Science. Oxford dictionary defines the word concept as "an idea or a principle that is connected with something abstract". This definition is vague, and rightly so as there are many theories of concepts, all with their own views about what a concept is. Most of this section's content is adapted from the presentation of theories of concepts by Eric Margolis and Stephen Laurence in [3],[4] and [5].

Most theories see concepts as mental representations, which occur in an internal system of representation and exist in the mind of each cognitive agent. This representational theory of the mind states that those mental representations act as building blocks for our "propositional attitudes", such as our beliefs and desires. It is the most widely adopted view ([8],[9]), with early adopter such as Locke, Hume as well as more contemporary philosophers (e.g. [10] [11] [12]). However, this view of concepts is not universal. Some prefer to see concepts as abilities rather than entities, for instance the ability to recognize a dog against a non-dog, and to draw inferences from this ([13] [14]). Other finally prefer to see concepts as abstract objects, which exist outside the minds and are objective entities ([15]). All the theories we will discuss here adopt the view of concepts as mental representations. They are also all related to lexical concepts, which are concepts which corresponds to single words in the natural language.

Classical theory

Up to the 1970s, the dominating theory of concepts was the Classical Theory. It defines concepts as mental representation with definitional structure, i.e. encoding necessary and sufficient conditions for their own applications. For instance, BACHELOR encodes the simpler concepts UNMARRIED and MAN: a bachelor is an unmarried man, and an unmarried man is a bachelor. Although dominant for many years, the theory has been under critics. One such critic is that it has failed to explain some experimental results from the field of Psychology, notably regarding the “typicality effect”. The typically effect means that for instance, apples are judged more representative of the concept FRUIT than plums.

Prototype theory

The prototype theory answer this concern by introducing probabilistic structure: a concept now encodes conditions that items in their extension tend to have; and an item is deemed from a concept if it has a sufficient number of them (rather than an exact match) (e.g. Hampton [18]). An APPLE share more of its constituents with FRUIT than PLUM, hence is more typical of, or similar to the concept fruit. However, prototypes are generally bad for characterizing a whole concept on their own. For instance, are all old women with grey hair grandmothers? The theory also does not abide to the principle of compositional semantic, which allows for the acquisition of an unbounded number of concepts. As illustrated by Fodor, the concept of PET FISH does not have the typical characteristics of PET and FISH: PET FISH’s prototype encodes “small fish with bright colours”, whereas the composition of PET and FISH’s prototypes does not yield any of these characteristics.

Dual theory

Most of these concerns disappear if we consider that a concept’s prototype is central to its structure, but does not exhaustively describe it. The dual theory considers that a concept has a “core” structure and an “identification procedure” structure. The “core” structure allows to characterize a concept fully and abides to the compositional semantic rule, whilst the “identification procedure” structure is where the concept’s prototype can be found. Remains the problem of understanding what is this core’s structure. It can be thought as falling under the Classical Theory, but hence inherits most of its limitations. Another more popular view in recent years is to explain its structure using the theory theory of concepts.

Theory theory

Within the theory theory of concepts, concepts are related to each other in a similar fashion as the components of a scientific theory, with inference relationships and contained within a mental theory (see [19][20][21]). During categorization, less emphasis is put on the observable properties of an item, and an internal “hidden” structure (or essence) is taken into account, which also plays in reference determination. There are however some limitations to the theory, one of which regards the possibilities for people to acquire the same concepts within this holistic theory (Fodor & Lepore [7]). Indeed, concepts are incorporated within a containing system (which includes personal beliefs) and determined by their relationships to other concepts, hence two concepts cannot be exactly similar within this framework.

All the above theories have as underlying assumptions that concepts are connected to each others, either in a Containment or Inferential model. A containment model states that a concept is related to other concepts if it contains them, whilst an inferential model relates a concept to other concepts using privileged (inferential) relationship, like in the theory theory of concepts. A last theory- conceptual atomism- differs radically from this view, stating that a concept does not have a semantic structure which depends on other concepts, but rather that its content is determined only by its relationship to the world.

2.1.2 Theory of Meaning

Another separate subject of study in Philosophy are theories of meaning. They do not look at what concepts are, but rather at what gives them their meanings. One such theory is the conceptual role semantics (CRS) theory, which states that the meaning of a concept is determined by its cognitive or inferential role in a person's cognition, and within its containing system. It is an extension of the "use theory of meaning", where an expression's meaning stems from its use in social interactions. Both the CRS and the Theory Theory - which are actually seen more as frameworks than actual theories- converge in content, with the ideas that concepts have inferential relationships to other concepts, and are contained within a cognitive system (see [6]). Moreover, the CRS also suffer from the issue of "translation holism" described by Fodor and Lepore [7] and mentioned before, which states that two people's conceptual systems must be highly similar for them to grasp similar concepts.

There are however a number of different theories of meaning, some of which considering that the role of a concept within a network is not enough to give it its meaning. Robert L. Goldstone and Brian J. Rogosky [29] refer to a "conceptual web" and an "external grounding" accounts of the theory of meaning. In the "conceptual web" accounts, the meaning of a concept is determined by its relationship with other concepts, as seen in most theories described above, whilst in the "external grounding" account, it must also depend on its connection to the external world.

In [29] and [30], Goldstone and Rogosky develop a simple neural network called ABSURDIST to find the correspondences between concepts in two systems (for instance cognitive systems of two different people). Even if these concepts are not exactly similar, being able to find correspondences could explain why different people can still communicate about them. Their model include both the internal "conceptual web" and "external grounding" components, and one or the other can be turned on and off, or be given more or less weight. They showed that using only intrinsic information gave better results than using only extrinsic information, and that within system information are sufficient to find correspondences between systems.

2.2 Cognitive Science and Semantic convergence zones

So far we have mostly talked of concepts as abstract entities and from a philosophical point of view but without mentioning the neurobiological mechanisms allowing human to acquire and store them. In Neuroscience, it is common to think of explicit memory (containing information/experience/concepts consciously recollected) as having two distinct constituents: episodic and semantic memory. This distinction was first proposed by Tulving in 1972 [22] and has persisted since then. Episodic memory relates to the memory of temporal episodes and their temporal relationships. We can think of it as a collection of past personal experiences. Semantic memory is where the more general knowledge of an individual lies, and where all the concepts, facts or symbols acquired are organized and related to each other. Linking it back with our philosophical exploration, it is where all the theories presented come into play.

Semantic memory has been a common research subject in Neurosciences. Two main theories exist about its neurobiological mechanisms, relating to how “embodied” or “disembodied” concept representation and retrieval is in the human brain. Embodied theories state that concept representation is anchored in sensorimotor system and is therefore “modality-specific”, whilst disembodied theories argue for a more abstract representation, detached from sensorimotor modules, i.e. amodal. There is evidence for the existence of both, and there has been many framework proposals around those theories. Neuroimaging experiments have indeed shown the activation of modality-specific areas of the brain (such as motion, visual , auditory, olfactory) during language processing tasks, supporting embodiement theories [24]. On the other hand, the existence of large areas of cortex in the brain that are situated in-between these sensorimotor zones has pushed for an amodal theories, where some proposes that a supramodal representation of concepts could exist.

Embodied models

Fully distributed (or embodied) models argue that the brain stores concepts as discrete features, within sensorimotor regions. Learning then occurs through repeated co-occurrence of a certain set of features which constitute a concept. This view has been criticized using much of the same arguments as the ones used against the Classical theory of concepts. Indeed, it means that a concept can only be a linear combination of some of these features, but this does not explain the non-linear “binding” of these features into a concept with its own emergent properties [25] (recall the typicality effect from the discussion on theories of concepts). Reilly et al. reports a metaphor from Lambon Ralph in [25] which illustrate this : “the mere presence of flour, butter, vanilla, and sugar do not ensure the presence of a cake”.

Disembodied models

Another category of models proposes that a disembodied amodal (or supramodal) representation also exist in the brain. It is formed from embodied representations through a transformation process occurring in the “convergence zones”, which are located in the cortex areas between the modality-specific sites. (see Binder and Desai , Damasio and Damasio 2009). However, although this theory fits in well in the cognitive and philosophical landscape, we know very little about the actual neurobiological processes involved in such a transformation and in the neural representation and manipulation of amodal symbols (Reilly et al. [25]). In this proposal, concepts are abstracted away from objects in the words, which additionally raises the “symbol grounding problem”. Detractors of this theory claim that there must be some external grounding, as an interrelated system of abstract concepts alone would not be sufficient to establish their meaning. The typical illustration would be to learn a foreign language using only words for that same language [26]. This is very close to what we discussed in the previous section about the CRS theory of meaning, and to the work from Gosltone and Rogosky ([29], [30]).

Hybrid models

Finally, there exist some hybrid approaches that lie somewhere in-between the two theories. One such example is the convergence zone framework. In 1989, Damasio [23] proposes that the "store" of any knowledge is directly in these multiple neural modality-specific regions, and that there is no single "amodal" store of the meaning of a concept that permanently exist. Instead, the meaning of an entity is recreated each time from these multiple modality-specific sites through these large aforementioned cortex areas or "convergence zones".

2.3 Representation of concepts in Computer Science

In Computer science, the field of Knowledge representation deals, as its name suggests, with the representation of knowledge. It organises information about the world in a way that allows a computer system to reason, and solve problems much like a human would do. Davis et al [28] see knowledge representation as a surrogate of an external entity, but which takes shape inside the reasoner. This -imperfect- surrogate is then intervening in the internal reasoning process in place of the outside world entity it semantically represents. This representation can be of tangible objects as well abstract notions, such as beliefs, actions or processes.

A popular such representation are semantic networks, which represents knowledge as interconnected nodes. These nodes can represent various entities (e.g. concepts) and the links between them represent their relationships. There are several types of such networks that vary in their way of representing relationships. In definitional networks for instance, links are labeled using expressions such as "is a" or "has" (e.g. DOG is an ANIMAL).

More recently, techniques for deriving static or dynamic word embeddings from large corpus of texts have been widely used and are still actively researched in Natural Language Processing. As we mentioned before, basic concepts can be thought of as lexical concept (equating one word in the vocabulary), hence we can think of these embeddings as a representation of concepts, where relationships are only defined in terms of distance. If the different dimensions of these embeddings are difficult to interpret on their own, the euclidean distance between concepts captures the semantics of these concepts. One particular embedding algorithm that is of interest for this thesis is the Glove algorithm, which constructs these embeddings based on co-occurrence statistics when trained on a large corpus of text.

2.4 Links with the project

This thesis focuses on how human learn new information in an unsupervised way. It is indeed generally accepted that the mechanisms allowing a human to learn are a mixed of unsupervised learning, with some level of supervision. There also exists some evidence that infants can infer meaning from associations and co-occurrence statistics ([25],[27]), hence reducing the needs for labels. Finally, the most recent theories of concepts and of meaning also support the fact that learning occur partly within an inferential system of concepts, where all concepts are interrelated. This is the case in the theory theory of concepts and CRS theory of meaning, as well as in the disembodied view in Neuroscience.

We've however mentioned some of the challenges encountered by these theories. The first is the "symbol grounding" problem, which supports that there should be at least some "external grounding" or labelling for concepts within these conceptual systems to have meaning. The second is: how can one grasp the same concept, or compare concept within these holistic theories if every cognitive agent has slightly different conceptual systems. This is what Gosltone and Rogosky ([29],[30]) explored with ABSURDIST.

Our thesis looks at how one can infer correspondences between conceptual systems using the internal structure of each system only, with no labels, and is therefore very close to these questions, except its angle is slightly different. It deals with modality-specific conceptual systems within the same cognitive agent, and how this supports the unsupervised learning observed in infants. Indeed, neuroscientists have highlighted the role of modality-specific brain activity during concept acquisition. This thesis and the broader work associated with it feed the following thought experiment: if an infant were to receive separately an abundance of visual and textual information, could he use the internal structure and co-occurrence statistics within domains only to match information from one modality to another (i.e. without receiving any labels)?

In the final version of this experiment, concepts are represented within modalities (ex. visual, auditory, textual) by embeddings that are created using co-occurrence statistics (Glove algorithm). However, this thesis only focuses on synthetic data, where a domain is sampled from a (mixture of) Gaussian, and a second conceptual system is created by adding noise and a random rotation. The idea is to test promising algorithms first on simpler structure before attacking more complex settings.

Chapter 3

Related work in Computer Science

In this chapter, we review current related Computer Science research. After quickly summarizing our problem, we present research in the fields of Manifold Alignment, Point Set Registration and Image to Image translation, all showing some similarities with our problem. We conclude by introducing the paper by Roads and Love [1] which has motivated this thesis.

3.1 Our problem: a reminder

In its mathematical formulation, our problem is equivalent to matching a system of points in space (high-dimensional), to another system of points in space. As mentioned before, we will want to keep in mind the following required properties for a potential successful algorithm:

- Scalable to high-dimension, with potentially a different dimension for each system
- Scalable to a large number of concepts
- Can deal with the many-to-one and no correspondence problems (which implies the systems might have a different number of concepts)
- Does not require supervision during training

3.2 Computer science literature

In computer science, there are different fields of study that have shown similarity with our problem. We will first summarize what they are, and what they look at.

- **Manifold alignment**. Manifold alignment looks at finding correspondences between high-dimensional datasets. Formally, we are given two data sets X and Y of high dimensional vectors $X = \{\boldsymbol{x_1}, ..., \boldsymbol{x_m}\} \in \mathbb{R}^{D_X}$ and $Y = \{\boldsymbol{y_1}, ..., \boldsymbol{y_n}\} \in \mathbb{R}^{D_Y}$ with $D_X, D_Y >> 1$. How can we find the correspondences between points? These methods mostly rely on learning lower dimensional manifolds where the data can be projected and aligned. The mapping is learnt either using knowledge about a subset of correspondences between points (supervised) or using information about local structure (unsupervised). These techniques deal with data from different domain in high-dimensional spaces, which seems quite adapted to our framework. However, most algorithms are not able to scale to very high dimensions and numerous data points. In our work, we borrow some aspects of two papers from this literature ([38] and [40]). More specifically, we construct our local features the same way they do.

- **Point set registration in Computer vision**. Point set registration consists in finding correspondences between two (or more) set of points, and the associated transformation that maps one set to another (e.g. scaling, rotation, translation). It is a famous problem in Computer vision tasks, and has extensive applications, for instance in autonomous driving or medical imaging. Similar to the previous framework, we we are given two data sets X and Y of high dimensional vectors $X = \{\boldsymbol{x_1}, ..., \boldsymbol{x_m}\} \in \mathbb{R}^D$ and $Y = \{\boldsymbol{y_1}, ..., \boldsymbol{y_n}\} \in \mathbb{R}^D$. Although these methods are unsupervised, D is assumed to be the same for both datasets (which come from the same domain), and is usually equal to 2 or 3. Some methods however claim scalability to high-dimensional data.

- **Image to image translation**. Image-to-image translation deals with the translation of images from a domain to another domain (for instance real photos of landscape to Monet paintings and vice versa), usually using a training set of image pairs. Zhu et al [50] however looked at this problem in an unsupervised way. Given a source domain X and target domain Y, they learn a mapping $G : X \rightarrow Y$ and an inverse mapping $F : Y \rightarrow X$ such that the distributions of images given by $G(X)$ and Y is indistinguishable, and $F(G(X)) \approx X$ (and vice versa). This cycle consistency constraints allow the learnt mapping to be meaningful, as otherwise an infinity of them exists.

3.2.1 Manifold alignment

The literature around Manifold alignments seem particularly suited to our problem, and we will dedicate it a large subsection. We also construct our local features the same way as done in the papers from Pei et al [38] and Fan et al [40] presented later (in the *Unsupervised methods* section). As mentioned above, the problem boils down to: given two data sets X and Y of high dimensional vectors $X = \{\boldsymbol{x_1}, ..., \boldsymbol{x_m}\} \in \mathbb{R}^{D_X}$ and $Y = \{\boldsymbol{y_1}, ..., \boldsymbol{y_n}\} \in \mathbb{R}^{D_Y}$ with $D_X, D_Y >> 1$, how can we find the correspondences between points?

All the methods developed consist in learning embeddings in lower dimensional manifolds for the two domains and finding correspondences between the embeddings. Some methods do this in two separate steps (learning embeddings first, then matching them [35]) whilst other use a single step, where the optimisation problem of learning embeddings/matching them is solved in a single optimisation problem [36]. Two main strategies are used to match embeddings at the manifold level (see [33][34][35]), either using known correspondences (supervision is needed), or by matching local structures (see [36],[38]).

Methods with some level of supervision

Ham, Lee and Saul wrote two papers looking at the semi-supervised alignments of Manifold. The problem is similar in both: Say we have X and Y two data sets of high dimensional vectors:

$$X = X_\ell \cup X_u = \{\boldsymbol{x_1}, ..., \boldsymbol{x_m}\} \in \mathbb{R}^{D_X} \qquad \text{and} \qquad Y = Y_\ell \cup Y_u = \{\boldsymbol{y_1}, ..., \boldsymbol{y_n}\} \in \mathbb{R}^{D_Y}$$

with $D_X, D_Y >> 1$. We have the correspondences between data sets for a subset of these: X_ℓ and Y_ℓ (ℓ as labeled). How do we find the correspondences for the unlabelled examples X_u and Y_u? In [33], the authors have two different approaches. The first consists in learning a low-dimensional linear manifold common to the two datasets by concatenating the data as if it was one dataset with missing entries (using either PCA or FA). The E-M algorithm is then used for learning the optimal parameters and distribution over latent, which in turns allow to infer the distribution over the missing entries. Once learned, samples from either X or Y are projected to this manifold and then projected back to the other domain. The second approach is to learn low-dimensional non-linear manifolds separately using locally linear embedding (LLE), but constraining the known correspondence to be equivalent. Then, all points from X and Y are projected to their respective manifolds, and the points in correspondence are the nearest ones. All explored methods give better results than traditional supervised learning methods, and the non-linear LLE method gives the best results. In their second paper (Semisupervised alignment of manifolds [34]), Ham, Lee and Saul make use of Laplacian eigenmaps [32] to learn separate low-dimensional embeddings for the two domains, but again, with the additional constraint that points in known correspondence should be near enough in the manifold.

In [35], Wang and Mahadevan present their "Procruste alignment" method, where they first learn the respective manifolds of each domain using traditional dimensionality reduction techniques such as LPP or Laplacian eigenmaps (instead of learning it under constraint as in [33] and [34]). Then the learnt manifold from one set gets its translational, rotational and scaling components removed so that the samples in correspondence are aligned. This method learns a mapping function and therefore can be applied to new test points in a more convenient way than in [34]. Indeed, in Ham et al. work, the embeddings results are computed "at once" on the given data sets, which makes it more difficult to generalize to new data points.

Their technique is tested on a toy example (3D manifolds representing protein structure), and then tested on collections of documents in 2 languages: each collection is mapped to an embedding space of dimension 100, and the closest documents are matched together. The semi-supervised methods from Ham et al in [34] and a baseline method are also tested for comparison. The "Procruste" method show better results than the Semi-supervised one, with 60% probability of retrieving the right match in the 3 closest document. The semi-supervised method performs quite poorly for that specific application (less than 10% probability), which the authors claim could be because all corresponding points are constrained to be identical, which can lead to overfitting. In contrast in the Procruste alignment method, a single transformation is learned for all corresponding points, which reduces that overfitting effect. However, the Procruste alignment method is rigid, hence not adapted to data requiring a non-rigid transformation to be aligned.

Computationally, the Procruste analysis is faster, the most computational steps being solving eigenvalue problems for manifold learning over each specific domain (so over a $n_1 \times n_1$ and a $n_2 \times n_2$ matrix). This means that the computational complexity is in $O(N^3)$ where $N = \max(n_1, n_2)$ (number of instances). The Semi-Supervised method requires to solve an eigenvalue problem over a matrix of size $(n_1 + n_2 - m) \times (n_1 + n_2 - m)$ where m is the number of known correspondences, hence being in $O((n_1 + n_2 - m)^3)$.

Another algorithm worth mentioning is KEMA from Tuia et. al [41]. This algorithm uses the Semi-supervised method from Ham et al as a basis, and performs its kernelisation, hence making it linear in a higher dimensional Hilbert space. It claims being able to deal with stronger deformations and higher dimensions than the semi-supervised method, as this latter method does not scale way to high dimensional data (where $d >> n$). However, this method remain computationally intensive.

Note that so far, all of the proposed methods require some level of supervision, where a certain number of pairwise correspondence is needed. It is therefore not adapted to our problem but is the basis for further unsupervised work done by Wang and Mahadevan and others which we will use in this thesis.

Unsupervised methods

In [36], Wang and Mahadevan look at similar problems, but when no correspondence data are available. Given $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$, they want to learn the mapping functions α and β so that $\alpha^T x_i$ and $\beta^T y_j$ belong to the same space Z and can be directly compared. The mapping to Z needs to preserve the neighborhood relationships within the two domains, and bring close together two data points from each domain with similar local geometries. To understand the strategy adopted in this paper, we need to go back to the strategy adopted in the work of Ham et al. In [34], the constrained embedding problem minimizes the following cost function:

$$C(f, g) = \mu \sum_{i=1}^{\ell} (f_i - g_i)^2 + 0.5 \sum_{i,j} (f_i - f_j)^2 W_x^{i,j} + \sum_{i,j} (g_i - g_j)^2 W_y^{i,j}$$

where f_i and g_i are the low-dimentional embeddings of the i^{th} point from X and Y respectively, and $W_x^{i,j}, W_y^{i,j}$ are the similarity measures between point i and j within X and Y respectively. The first term forces corresponding embeddings to be close, whereas the second and third term learn embeddings for the data points in X and Y respectively. In [36], because there are no correspondence information, the first constraint is replaced by a soft constraint on the similarity of local geometries. Also, the mapping functions α and β are learned rather than the embeddings directly (so that it can be applied to new test points), hence the cost function to minimize becomes:

$$C(\alpha, \beta) = \mu \sum_{i,j} (\alpha^T x_i - \beta^T y_j)^2 W^{i,j} + 0.5 \sum_{i,j} (\alpha^T x_i - \alpha^T x_j)^2 W_x^{i,j} + \sum_{i,j} (\alpha^T y_i - \alpha^T y_j)^2 W_y^{i,j}$$

where $W^{i,j}$ is high for similar local geometries between x_i and y_j. They test this fully unsupervised method on the alignment of protein manifolds and document collection manifolds, as in their Procrustes method in [35],and were able to achieve a good alignment. For the document collection alignment, the true match has a 65% probability to be the top 1 candidate and 80% in the top 4. However in this method, local features are described using a distance matrix of the k nearest neighbours, and all the k! permutations need to be considered. One can refer to Section 4.3.2 for more details on the method used. This is computationally expensive even for small k.

In [38], Pei et al. adapted Wang et. al unsupervised algorithm from [36], but using a more efficient local matching technique, which would not involve k! permutation of the k nearest neighbours. Instead, they fit a B-spline curve to the discrete vector of distances of point i to the k nearest neighbour. Then the distance between points i and j from different manifold (in terms of local geometries) is defined by:

$$d_{ij}^c = \int_0^k (|g_i^s - g_j^t| + \beta |\nabla g_i^s - \nabla g_j^t|) dx \qquad (3.1)$$

In our work we borrow the local features and associated distance defined in this paper. Contrary to Wang and Madahevan's method, this method is non holistic (non-rigid) and computes one affine transformation per point instead of a holistic one. Affine transformations on neighbouring points is constrained so the Frobenius norm of the difference does not increase too much. Using the author's notations, we call the two domains to be matched S and T (Source and Target). A first set of sparse correspondence between points is determined using the distance in Equation 3.1, and their ensemble is called C. A non-linear optimisation problem is then solved to perform dense matching. The optimisation objective is:

$$E(\alpha) = \mu_d E_d + \mu_s E_s + \mu_c E_c$$

The first term is a measure of distance between the transformed points from S and the target domain T. It ensures that points are projected within the range spanned by the points in T. It somewhat plays the same role as the distribution loss in our work The second term: $E_s = \sum_{s_i \in S} \sum_{s_j \in knn(s_i)} ||\alpha_i - \alpha_j||_F^2$ ensures that the transformations associated with points in the same local structures are not too different (i.e. are relatively rigid). Finally the last term: $E_c = \sum_{(s_i,t_i) \in C} ||\alpha_i s_i - t_j||_2^2 w_{ij}$ ensure that points with similar local structure are kept close. The parameter $w_{ij} = e^{-d_{ij}^{c^2}/\gamma}$ (with γ set to 0.03) are the weights of each terms in the last sum, giving more importance to points with the most similar local structures.

Although lower than previous methods, the computational cost of the method remains high. There is one transformation $\alpha_i : \mathbb{R}^p \rightarrow \mathbb{R}^q$ to solve per point, hence $m \times (p+1) \times q$ in total (where p and q are the dimensions of the source and target manifold and m the number of points to be matched). The method is therefore not suitable for high dimension (e.g. 100).

In [39] and [40], Fan et al develop new local features that are not based on Euclidean distance only. In our work we also borrow their local features and associated distance, that we will compare with Pei et al's method. Fan et al claim that Euclidean distance are sensitive to data sampling, and that their solution is both robust to sampling and rotation invariant. They define angular features representing the relationship between a point and its neighbours. More details on this can be found in Section 4.3.2. These features are used to define an 16 or 8 dimensional histogram feature vector which characterize each point in each domain. Euclidean distance between those feature vectors is used to measure local structural similarities between the two manifolds. In [39], a first rough alignment is performed by matching areas based on these structural distances, where only the points with maximum confidence are aligned. In [40], all points are given a correspondence and not only a subset of them, with the added possibilty of defining outliers (i.e. having no correspondence). This gives a set of initial correspondences learnt in an unsupervised way, which is then used in the next step of the algorithm (i.e. the dense correspondence).This step is largely inspired from the work of Ham et al [34] and Wang et al [36] presented above, where a single loss function is minimized to learn a mapping from the two domains to a a joint lower-dimensional space, where the corresponding points are constrained to be near. There are again three terms in the objective function: the first minimizing the distance between the learnt manifolds, the second ensuring local relationships are preserved during the transformations, and the third minimizing the loss over the local structure correspondence information. Note that they use a holistic transformation unlike Pei et al [38]. To reduce computational complexity when assigning correspondences (which is NP-complete), the authors use a soft-assign technique instead of a binary correspondence matrix (using continuous values rather than 0 or 1), hence making the problem easier to solve.

All these manifold alignment methods are quite close to our problem, but due to generally high computational complexity are hard to solve for a high number of concepts or high dimensionality. In this report, our main focus is to integrate some local structure information borrowed from these papers (Pei et al [38] and Fan et al [40]) to the baseline algorithm. The construction of local features scale quadratically in the number of concepts for these two methods, which could be problematic when considering our full scale problem. However, observing how the baseline algorithm behaves when integrating local structure information for our simpler structures will still be informative, and help us decide whether this line of work can be optimised or pushed further.

3.2.2 Point set registration

Registration of point sets consists in finding correspondences between two set of points in computer vision tasks, and the associated transformation that maps from one set to another. It is a common problem in computer vision, for instance in stereo-matching. Although often applied to 2 or 3D algorithms, some of the algorithms can be adapted to higher dimensions. Transformations can be rigid (translation, rotation, scaling) or non-rigid, like in affine transformation, where there can be anisotropic scaling and skews. Non-rigid transformations are difficult to model: simplistic models tend to be inadequate, and other methods often have high computational complexities. Additionally the high number of parameters make them sensitive to noise and outliers, as well as likely to get stuck in local minima. Other challenges are the problem of missing data and outliers, which means not all points have a correspondence in the other set.

For reference, Zhu et al review the point set registrations methods in [42]. Methods can be split between pairwise (between two point sets) and groupwise (more than two), but we will focus here on pairwise methods, and present a few of the most popular methods.

Distance based methods

These methods work in two steps: (i) Assign matches based on closest distances between the two set of points, (ii) Minimize those distances. These include the widely use Iterative Closest Point (ICP) algorithm, from Besl and McKay [43] and Zhang [44], which uses a least-square distance minimization to estimate the rigid transformation that best align each corresponding points. It then re-assign correspondences based on closest distance and goes on iteratively until reaching a local minima. Its popularity is notably due to its low-computational complexity and simplicity. ICP has inspired many variants, affecting all stages of the algorithms (see [46], [45]). Note that a rough initial alignment is needed, and that this algorithm can get stuck in local minima.

Many other distance-based methods were developed. Robust Point Matching (RPM) [47] extends ICP (for rigid methods) using soft-assignment of points in order to convexify the optimization problem and make it less prone to ICP's local minima limitation. Chui and Rangarajan [48] also proposed TPS-RPM, a similar algorithm for non-rigid transformations but hardly applicable for high dimensional data. A notable other category of method is Graph Matching, which allows to take into account the relations of each points to their neighbours using information from vertices and edges. Many of the current graph-matching problems are NP-hard, and a lot of the research is concentrated in finding successful approximations [42].

Probability based methods

Another family of methods use maximum likelihood (ML) estimation, where a set of points is taken as Gaussian Mixture Model (GMM) centroids, whilst the other one are considered data points. The Expectation Maximum (EM) algorithm for GMM is used to fit the GMM centroids to the datapoints, where the E-step compute probabilities and the M-step update the transformations.

The most famous of these methods is Coherent Point Drift (CPD), proposed by Myronenko and Song in [49]. One point set is taken as a Gaussian Mixture Model (GMM) centroids, whilst the other one are considered data point. In this method, the GMM components are constrained to move coherently with respect to each other following the motion coherence theory. The authors proposed two algorithms, one for rigid problems and their most popular one for non-rigid problems. The rigid method parameterizes the transformation using a rotational, translational and scaling component, whilst in the non-rigid methods, the transformation is unknown but is forced to be smooth through Tikhonov regularization on the displacement function. The method can also be made linear in complexity, hence making it available to higher dimensional problems.

Link with Manifold Alignment literature

We note some strong similarities in the two research fields. The manifold alignment's problem formulation is a bit more general, as it deals with data of different dimensions that have to be projected in a shared manifold space. If manifold learning and alignment are done in two separate steps, then the second step becomes a point set registration problem, i.e. finding correspondence between points in the same domain.

Both find ways of including local structure data, with some of the Graph matching techniques in Point Set Registration using neighbouring information, in a similar fashion as the methods from Pei et al or Fan et al ([38],[40]) in the Manifold Alignment literature. The idea of coherent motion from the CPD method is another way of taking into account local structure, by preserving it through the transformation rather than by directly comparing local structures from domains X and Y.

3.2.3 Image to image translation

Image to image translation is the problem of mapping an image from one domain to another domain, a typical example being for instance to transform the photograph of a landscape into a painting of a specific style. This was traditionally done using paired images as training data, but a recent method from Zhu et al. [50] that proposed to do this in an unsupervised way gained a lot of attention. We borrowed an explanatory figure from the authors on Figure 3.1.

Two functions F and G are mapping images from domain X and Y respectively to the other domain. Two discriminators D_X and D_Y try to distinguish the real images from a domain from the images that were mapped into it. Functions F, G and D_X, D_Y end up competing against each other, until the mapped image can not easily be recognized by the discriminators. A cycle loss is also taken into account, as mapping to a domain, and then back into the source domain should give the original image. The two losses used are therefore:

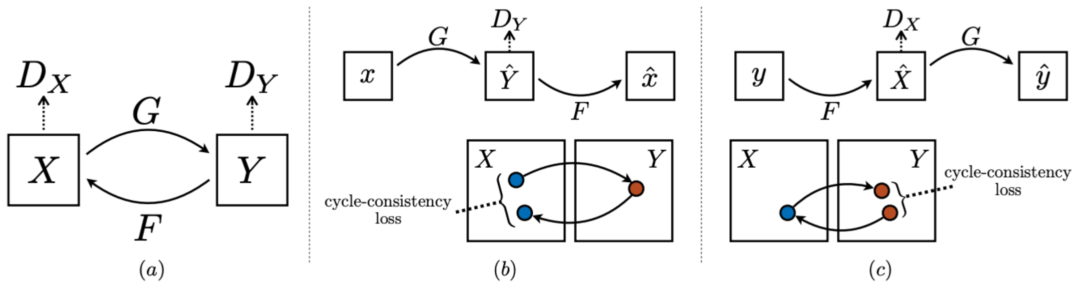


Figure 3.1: Figure from CycleGAN paper ([50]). Two networks G and F respectively map images from domains X to Y and Y to X. Two discriminators are trained to discriminate the mapped images from the original ones in each domain. A cycle consistency loss is computed as illustrated in (b) and (c).

- An adversarial loss:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = E_{y\sim p_{data}(y)}[\log D_Y(y)] + E_{x\sim p_{data}(x)}[\log(1 - D_Y(G(x)))]$$

- A cycle loss:

$$\mathcal{L}_{cyc}(G, F) = E_{x\sim p_{data}(x)}[||F(G(x)) - x||_1] + E_{y\sim p_{data}(y)}[||G(F(y)) - x||_1]$$

The final loss is a weighting of these two losses.

Many papers since then were inspired by this one. For instance CrossNet [51] learns the mappings from one domain to a shared latent space, much like in the Manifold alignment literature. Another example is the work from Almahairi et al [52], where the method is extended to a many-to-many mapping.

This paper is the main inspiration for the baseline algorithm presented in the Method Section.

3.3 Basis for the thesis : Roads and Love paper [1]

Roads and Love's paper [1] is the main work on which this thesis is based. The authors created modality-specific conceptual systems using the Glove algorithm trained on co-occurence statistics from the following domain: (i) a Common Crawl text corpus (directly using the pretrained Glove embeddings [54]), the Open Images dataset ([55]), and the AudioSet dataset ([56]). The original Glove algorithm works by sliding a window of a certain size across the text corpus, and by counting when words co-occur in a window. This creates a co-occurrence symmetric matrix. A specific loss function is then used on this matrix to create the embeddings. Co-occurrence statistics were created for the different domains by adapting this method to the data. For the images dataset, all labels occurring in a specific image were counted as co-occurring. The audio dataset is comprised of 10 seconds labelled audio clips, and similarly, all labels corresponding to one audio clip were counted as co-occurring.

An alignment correlation metric can be computed for a mapping between two systems. This is done by first computing a similarity matrix for each system -where the order of the concepts in the matrices should be kept according to the mapping- and then by computing the Spearman correlation between the upper diagonal portion of the two similarity matrices. The similarity matrix indeed captures aspects of the internal structure for each domain, and a a good mapping should have a high alignment correlation.

Using the embeddings aforementioned, the authors showed a strong correlation between the mapping accuracy and the alignment correlation, but revealed the existence of "misleading mappings". Those are "wrong" mappings having a higher alignment correlation than the true mapping.

However, the work shows that despite these misleading mapping, the data created from co-occurrence statistics maximise the alignment correlation metric most of the time for the correct mapping. This suggests that concepts have a unique signature within different conceptual systems, and can therefore be matched or compared. A naive alignment algorithm would be to test alignment correlation for all permutations and pick the permutation which maximises the metric. This is too computationally expensive, which leads us to look for more efficient algorithms to solve the problem.

3.4 Summary and link with our method

The cycleGAN unsupervised approach (Zhu et al. [50]) to the image to image translation offers a scalable way of mapping images from one domain to another. Applying it to our system of points -although replacing the discriminator networks by distribution losses- is the main inspiration for the baseline algorithm.

The Manifold alignment literature corresponds closely to our problem (i.e. aligning two set of points of potentially high and different dimensions by first projecting them onto a shared low-dimensional latent space), but most approaches are hardly scalable to high dimensional data or to a high number of concepts. We decide to borrow some of the local features from this literature to determine a subset of potentially corresponding points, that we then use to pretrain the models (i.e. Pei et al. [38], Fan et al [40]). This will allow us to see how the baseline model reacts to additional structural information. If successful on our synthetic data, it gives a potential lead to explore similar methods on the more complex data (i.e. created using the GloVe algorithm as described in the Introduction and in [1]).

Finally, the Point Set Registration literature is also quite similar to our problem, but is applicable only to set of embeddings of the same dimension. It however offers good insights into another set of techniques which look for rigid or non-rigid transformations between set of points. Notably, the Coherent Point Drift method (Myronenko et al [49]) and variants claim scalability, and make use of structure preservation during transformation through regularization techniques. This could also be an inspiration for future methods.

Chapter 4

Method

Recall that this thesis focuses on finding efficient algorithms aligning two conceptual systems. Those systems will eventually consist in GloVe embeddings derived from text, image and audio data. However for the purpose of this thesis, we work with simplified synthetic data, where one system consists in a randomly generated set of points, and the second system is created by adding noise and a rotation. This allow to test algorithms on various controlled structures.

In this chapter, we first present how the synthetic data is created and what parameters are attached to it. We then introduce the baseline algorithm (inspired from CycleGAN and developed within the Love lab at UCL), before detailing our proposed improvements which make use of local features as in the Manifold Alignment literature. Finally, we describe the experiments we carry out to compare both, as well as the measures we use for success.

The associated code can be found on the following link: https://github.com/rosie31/unsupervised-alignment-team.git

4.1 Synthetic data

In the base configuration represented on Figure 4.1, the domain X is created by generating a number of random samples from a mixture of Gaussian in $\mathbb{R}^D$. The domain Y is created by adding a random noise and a random rotation.

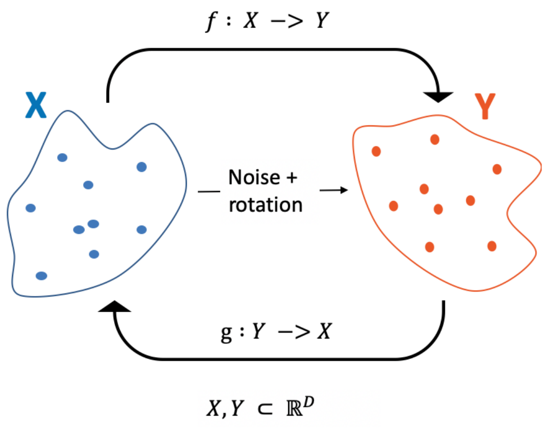


Figure 4.1: Baseline algorithm. A first set of points is randomly generated from a (mixture of) Gaussians in a D-dimensional space. A second set of point is created by adding noise and a rotation. These constitute our two conceptual systems.

The main parameters involved in the generation of the two synthetic conceptual systems are as follow:

- D or *embdim*. The dimension of the samples.
- *n_epicentres*. The number of generating gaussians (uniformly sampled in $[-1, 1]^2$).
- *linear_sep*. A measure of "linear separability" between these gaussians epicentres. We define the variance of each gaussian as $\sigma^2 = \frac{1}{linear\_sep^3}$. For higher values of this parameter, the samples generated by each gaussians are more and more distincts (i.e. linearly separable).
- *n_concepts*. The number of concepts.
- *noise*. The standard deviation of the random Gaussian noise added to each point between the first and second system.

Note that before creating the second system by adding a random noise and rotation, the first system is pre-processed so that every points from it all fit in the interval $[-1, 1]^D$. We will also be using the *tanh* activation function which maps points to this interval. During pre-processing, the mean of all the points is substracted so they are centered around zero. Then the samples are divided by the maximum absolute value of all the points coordinates (from any dimension) so they belong to $[-1, 1]^D$. Because noise is added when creating the second system, each sample is further divided by 3. All the points are therefore centered around zero and around the range $[-0.3, 0.3]^D$.

Clumpiness scenarios

We define here seven level of clumpiness, each with different values of *n_epicentres* and *linear_sep*. We will use these when comparing the performances of the models under different scenarios.

Clumpiness	1	2-2	2-4	2-8	3-2	3-4	3-8
n_epicentres	1	2	2	2	3	3	3
linearsep	na	2	4	8	2	4	8

Table 4.1: Clumpiness scenarios - Parameters. Number of epicentres and linear separability for each clumpiness scenario. For the scenario with one generating gausian only (Clump1), the linearsep parameter is non applicable as it represents the level of linear separability of two or more gaussians.

Figure 4.2 shows an example of a system generated using each scenarios from Table 4.1.

Note that because all these scenarios are in the same range of values in a D dimensional space, points from very clumpy scenarios such as 'Clump2-8' and 'Clump3-8' are "clumped" closer together. It is likely that a fixed level of noise will therefore affect these scenarios to a greater extent than others.

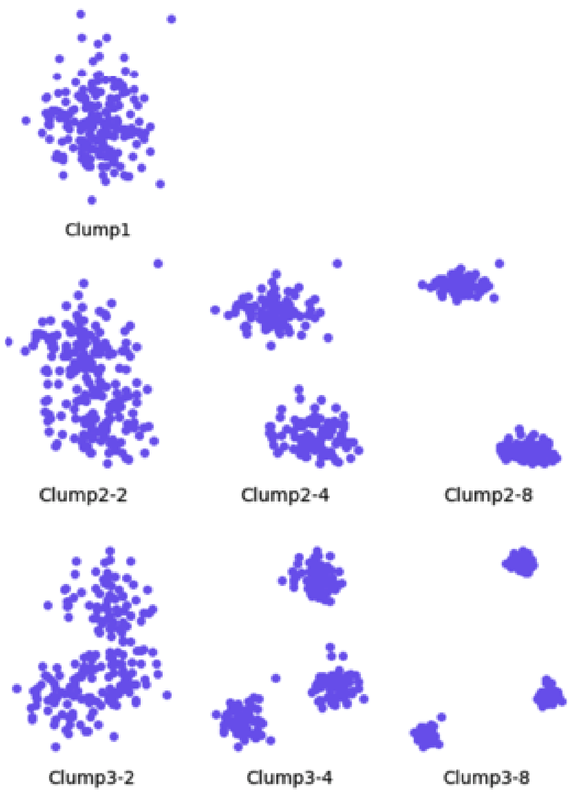


Figure 4.2: Clumpiness scenarios - Examples. We define levels of "clumpiness" of the data to test our algorithms on. Each row correspond to respectively one, two and three generating Gaussians. For the first row (scenario 'Clump1' with only one generating Gaussian), the linear separability parameter is non-applicable. The remaining scenarios are organized so each column correspond to a linear separability level of respectively 2, 4 and 8 (e.g. Clump2-4 and Clump3-4 both have a linearsep of 4).

4.2 Baseline Algorithm

4.2.1 Architecture

The baseline algorithm is largely inspired from CycleGAN [50]. We learn two functions $f : X \to Y$ and $g : Y \to X$ mapping points from one domain to another as presented on Figure 4.1.

Learning these functions directly is possible as we only consider data in the same dimensions ($X, Y \subset \mathbb{R}^D$), hence not projecting the data on a shared latent manifold. Both f and g are dense neural networks with the same architecture, as shown in Figure 4.3. They are composed of three dense layers of *n_hidden* nodes, and one output layer of the size of the dimension of the domain to be mapped to. In this work, we take *n_hidden* $= 100$. The activation function at each layers are *tanh*. No dropout, nor normalisation layers are applied. The optimiser used is Adam, with a learning rate of 0.001.

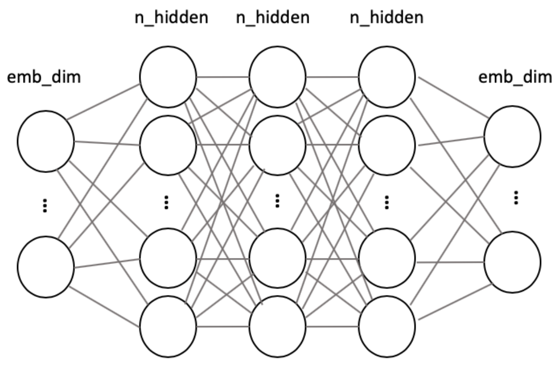


Figure 4.3: Neural network used for models f and g. It is a dense network composed of 3 hidden layers of *n_hidden* (fixed to 100) nodes each.

4.2.2 Base losses

We use a similar cycle loss as in CycleGAN (although using l2 norm rather than l1). The discriminator was deemed more suited to image data and a distribution loss is used instead, ensuring that the projected points from X to Y match the distribution of points in Y, and vice versa. Inspired from the Point Set registration literature, the distribution loss uses a GMM likelihood estimation method, where one set of point is considered as the centers of gaussians, and the other set of points sample points. A low distribution loss therefore correlates with a high likelihood of the data points to have been samples from the gaussians. Note that there are many other methods to match distribution that could be used. This includes KL divergence or MMD, as well as Optimal transport distance such as Wesserstein or Sinkhorn ([53]). This latter distance has computational speed advantages that could benefit our algorithm, but this is not explored further in the scope of this thesis.

Given a batch of data X (or Y) of shape ($bsz \times emb\_dim$), we call $f(X) \in Y$ (or $g(Y) \in X$) the resulting data points from applying the function f (or g) to each data point. The cycle loss is defined as:

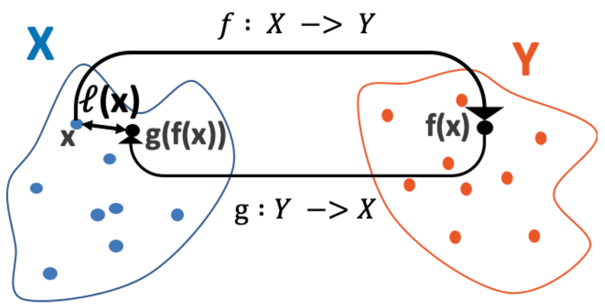
$$\mathcal{L}_c(X, Y) = \frac{1}{2}\left[\frac{1}{N_X}\sum_{x \in X}[x - g(f(x))]^2 + \frac{1}{N_Y}\sum_{y \in Y}[y - f(g(y))]^2\right]$$

For the distribution loss, we consider that one dataset is the centroids of a Gausian Mixture Model and the other one are datasample. We then calculate the log likelihood of the samples to have been generated by the GMM. Let's note $P(f(x)|gmm_Y)$ the probability that point $f(x)$ was generated by the gmm formed by the points of Y, we have:

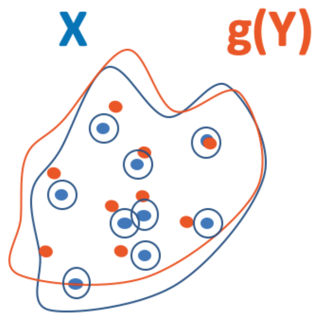
$$\mathcal{L}_d^f(X) = \frac{1}{N_X}\sum_{x \in X}\log P(f(x)|gmm_Y) \qquad \text{and} \qquad \mathcal{L}_d^g(Y) = \frac{1}{N_Y}\sum_{y \in Y}\log P(g(y)|gmm_X)$$

Another parameter resulting from this approach is the standard deviation of each gaussians considered (*gmm_scale*). It is set to 0.01.

We give an intuition of how these two losses work through the illustration on Figure 4.4



(a) Cycle loss



(b) Distribution loss

Figure 4.4: Illustration for the two losses. On Subfigure (a), point $x \in X$ is projected to Y and then projected back to X. If we call $\ell(x)$ the distance between between x and the resulting $g(f(x))$, the cycle loss for X constitute is $\mathcal{L}_c(X) = \frac{1}{N_X} \sum_{x \in X} \ell(x)^2$ or in other words, the average of the square of this distance for all points in X. On subfigure (b), points in system X are considered as the centroid of Gaussians (constituting a Gaussian Mixture Model or GMM). $g(Y)$ are considered sample points and the parameters of the GMM are fitted to maximise the probability of them having been generated by the GMM.

Finally, each loss is assigned a weight (or scale) for its contribution in the total loss. The scale for the distribution loss (loss_distr_scale) is set to 1, and the cycle loss scale (loss_cycle_scale) is 1e4. This was set empirically and the impact of varying this latter scale is explored for the baseline algorithm (See Baseline results). A value of 1e4 showed reasonable results compared to the other values tested (1e3 and 1e5).

4.2.3 Training procedure

Because the models (f and g) often get stuck in local minimum, we train them for only 30 epochs, and then perform "restarts", where the models are randomly re-initialised. Early experiments showed that 30 epochs are enough for the models to be trained, if they do not get stuck in local minimums. The pseudo code is shown below:

Algorithm 1: Training procedure

```
for i_run in range(n_run) do
    generate X,Y data;
    for i_restart in range(max_restart) do
        initialise models (model f & model g);
        for i_epoch in range(max_epoch) do
            for x_batch,y_batch in batches do
                train_batch_step(x_batch,y_batch) ;
            train_full_step(X,Y) ;
        get final f & g models of restart ;
        check if best so far and keep in memory if so ;
```

We choose to perform 10 runs of 100 restarts (of each 30 epochs):

$$\begin{cases} n\_run = 10 \\ max\_restart = 100 \\ max\_epoch = 30 \end{cases}$$

Let's now zoom on the two functions $train\_batch\_step$ and $train\_full\_step$.

train_batch_step
Let's call $\boldsymbol{\theta_f}$ the parameters of f and $\boldsymbol{\theta_g}$ of g. We calculate the gradients:

$$grad_c^f = \nabla_{\boldsymbol{\theta_f}} \mathcal{L}_c(x\_batch, y\_batch) \qquad \text{and} \qquad grad_c^g = \nabla_{\boldsymbol{\theta_g}} \mathcal{L}_c(x\_batch, y\_batch)$$

We then use the Adam algorithm to update parameters from $\boldsymbol{\theta_f}$ and $\boldsymbol{\theta_g}$ at each batch training step. The batch size is set at 100. Our base scenario has 200 concepts, which means there are only two batches.

train_full_step
We calculate the gradients:

$$grad_d^f = \nabla_{\boldsymbol{\theta_f}} \mathcal{L}_d^f(X) \qquad \text{and} \qquad grad_d^g = \nabla_{\boldsymbol{\theta_g}} \mathcal{L}_d^g(Y)$$

We then use the Adam algorithm to update parameters from $\boldsymbol{\theta_f}$ and $\boldsymbol{\theta_g}$ at each full training step.

4.3 Proposed improvement: Include internal structure

4.3.1 Proposal

Inspired from the Manifold alignment literature, we explore the possibility of including information about the local structure in each domain. The principle is that each point gets assigned a local structure feature based on their k nearest neighbours. Then a term is added to the optimisation objective which minimizes the distance between local structures. Different papers use different local features. Some of the papers have a two-step approaches, where initial surrogates (correspondences) are determined based on the distance between the local features, and the points are first loosely aligned using this information (sparse matching), and then a dense algorithm does the rest of the work.

In this thesis we explore this two-step approach according to the following method:

1. Calculate some local features for each points in X and Y (of size N each) based on their nearest neighbours.

2. Construct the "similarity" matrix D_{XY} of size (N, N) which calculates the distances between

each pair of points (x, y) with $x \in X$, $y \in Y$.

3. The indices (i, j) of the lowest values in D_{XY} give us the initial matching. We pick the $N_{surrogate}$ smallest values, corresponding to the number of surrogates we use.
4. At each restart, the randomly initialised models are pre-trained for 30 epochs using the surrogates data as if they were true matches (using a mean squared error loss).
5. The pretrained models are trained as before on all the data.

4.3.2 Local features and similarity matrix

Different local features have been proposed in the manifold alignment literature. An influential paper in this field was published by Wang and Mahadevan in 2009 [36]. Several papers then built upon their method since then. Two of them are from Pei et. al [38] and from Fan et al [40]. The local features and similarity matrices used in these two latter papers are the ones used in this thesis. We extend the presentation of these papers done in the Literature Review here.

Wang and Mahadevan [36]

In Wang and Mahadevan [36], the similarity between point x_i and y_j is defined as :

$$W^{ij} = e^{-dist(R_{x_i}, R_{y_i})/\delta^2}$$

R_{x_i} and R_{y_j} are $(k+1) \times (k+1)$ matrices such that $R_{x_i}(a, b) = dist(z_a, z_b)$ for $(a, b) \in [1 : k+1]^2$, where $z_1 = x_i$ and $\{z_2, ..., z_{k+1}\}$ are the k nearest neighbours of x_i. The Euclidean distance is used to calculate the entries of R_{x_i} and R_{y_j}. Because there are $k!$ possible permutations of the k neighbours of a point, there are as many possible matrices R_{x_i}. We note $\{R_{x_i}\}_h$ the h^{th} variant, following the notations in [36]. The same can be done for R_{y_j}. Then the distance between R_{x_i} and R_{x_i} is defined as:

$$dist(R_{x_i}, R_{y_i}) = \min_{1<h<k!} \left[\overbrace{||\{R_{y_j}\}_h - k_1 R_{x_i}||_F}^{dist_1(h)}, \overbrace{||R_{x_i} - k_2\{R_{y_j}\}_h||_F}^{dist_2(h)} \right]$$

where

$$\begin{cases} k_1 = trace(R_{x_i}^T \{R_{y_j}\}_h)/trace(R_{x_i}^T R_{x_i}) \\ k_2 = trace(\{R_{y_j}\}_h^T R_{x_i})/trace(\{R_{y_j}\}_h^T \{R_{y_j}\}_h) \end{cases}$$

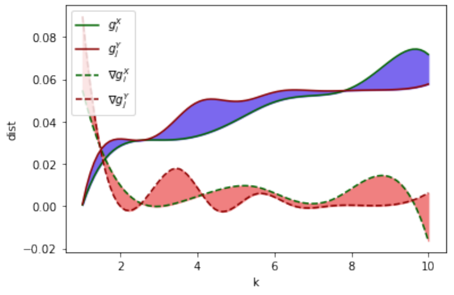
Indeed, it can be shown that this choice of k_1 and k_2 respectfully minimizes $dist_1(h)$ and $dist_2(h)$ for a fixed value of h (see proof in [36]). Solving this problem requires to calculate $\{R_{y_j}\}_h$ for each value of h, i.e. $k!$ times and is therefore computationally intensive.

Pei et al [38]

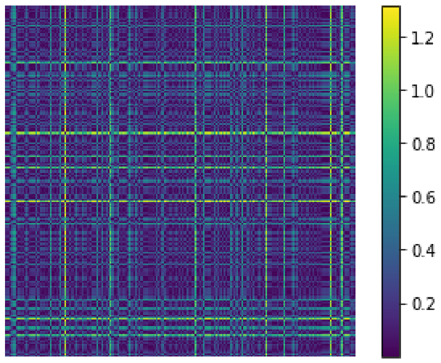
In Pei et al [38], the heavy computational cost that comes with the former technique is lifted by fitting a B-spline curves to the distances of a point to its k nearest neighbours. We will test this technique in this paper. Let d_{ij}^X be the euclidean distance between a point $x_i \in X$ and its j^{th} nearest neighbour in X. For each point in $x_i \in X$, we define the vector $\boldsymbol{d_i^X} = (d_{i1}^X, ..., d_{ik}^X)$ where k is a fixed parameter equal to the number of nearest neighbour considered. B-spline curves $\{g_i^X\}_i$ are then fitted to the vectors $\{\boldsymbol{d_i^X}\}_i$. This process can be repeated for domain Y, and the similarity between points $x_i \in X$ and $y_j \in Y$ is given by:

$$d_{ij}^c = \int_0^k \left(|g_i^X - g_j^Y| + \beta|\nabla g_i^X - \nabla g_j^Y|\right) \tag{4.1}$$

The two terms in the integral measure the difference between the areas under the two curves, as well as the difference between the areas under the gradients. This is illustrated on Figure 4.5.



(a) B-spline curves, gradient curves, and respective enveloped areas for two points x_i and y_j(k=10)



(b) similarity matrix between X and Y

Figure 4.5: Example of B-spline curves, distances between local features and similarity matrix. The distances between a point and its 10 nearest neighbours is calculated and a curve is fitted. The solid red and green lines in (a) represent two such curves for two separate data points. The blue area is the distance between curves. The gradients of these and distance between them is also computed (dotted lines and red area). The final distance between two points based on their local structure is a linear combination of the two areas. On (b), we represent the similarity matrix between points in X and points in Y based on this distance.

Note that the distances are computed after centering and normalisation of the two data systems X and Y. This is important as distances depend on the scales of the domains, hence comparing them require normalisation. In this work, as described in 4.1, the first system is centered around zero and scaled down to the range $\sim [-0.3, 0.3]^D$. The second system is created by adding random noise and performing a random rotation (with center zero). Hence these conditions are automatically fulfilled.

Fan et al [40]

Fan et al. developed a more complex feature based on the angles between the neighbours. As it is not based on Euclidean distances between points, it is less sensitive to data sampling than the two previous methods. The method is also rotation invariant, which is necessary for this problem. For each point $p_i \in X$, the matrix $R_i = [p_i, p_1^i, ..., p_k^i]$ of the point itself and its k nearest neighbours is defined (of size $D \times k$). The normal of the best fit plane is called $\boldsymbol{n_i}$ and can be obtained as the left eigenvector corresponding to the smallest value. More specifically, it is obtained by first centering the points in R_i around zero (by removing the mean vector). If we call Z_i the centered points, the singular value decomposition of the transpose of Z_i gives:

$$Z_i^T = U_i D_i V_i^T \tag{4.2}$$

The column in U_i corresponding to the smallest eigenvalue on the diagonal of D_i is the normal vector of R_i.

Let's call $I_i = \{i : p_i \in R_i\}$ the ensemble of the indices of the points belonging to R_i, the k nearest neighbours of point i. For each pair of points $(t, s) \in I_i^2$, a unit vector $\boldsymbol{v_{t,s}}$ is defined, as well as three scalar features f_1, f_2, f_3:

$$\boldsymbol{v^{t,s}} = \frac{p_t - p_s}{||p_t - p_s||}$$

$$\begin{cases} f_1^{t,s} = \max(n_t \cdot n_s, -n_t \cdot n_s) \\ f_2^{t,s} = |\arccos(\max(n_t \cdot v, -n_t \cdot v)) - \arccos(\max(n_s \cdot v, -n_s \cdot v))| \\ f_3^{t,s} = ||p_t - p_s|| \end{cases}$$

Recall that a dot product of two vectors measures their angle (as the cosine of the angle between the 2 vectors). Hence f_1 measures the angle between the normal vectors of R_t and R_s, in effect measuring the difference in orientation of the best fit-planes of the k neighbours of point t and s respectively. f_2 similarly measures the angles between the normals and the unit vector, and f_3 is simply the euclidean distance between the pair of points.

We then get the feature vector for a specific point i by following these steps:

1. Create a 3-bit code (common to all points $p_i \in X$) :

$$bit\_codes = [000, 001, 011, 010, 110, 111, 101, 100]$$

2. Get $f_1^{t,s}, f_2^{t,s}, f_3^{t,s}$ for all $(t, s) \in I_i^2$.

3. Define threshold values for each features : $thresh_1^i, thresh_2^i, thresh_3^i$. These are defined for a specific i as the averages of the respective features values for all the pair of indices in R_i:

$$thresh_k^i = \frac{1}{N_{pairs}} \sum_{(t,s) \in I_i^2} f_k^{t,s} \quad \text{, for } k = 1, 2, 3$$

4. For each pair $(t, s) \in I_i^2$:

   -Define the bit-code $b(t, s) = [c_1^{t,s}, c_2^{t,s}, c_3^{t,s}]$ such that:

$$c_k^{t,s} = \begin{cases} 1 & \text{, if } f_k^{t,s} > thresh_k^i \\ 0 & \text{, otherwise} \end{cases} \qquad (\text{ for } k = 1, 2, 3)$$

   -Find the corresponding index of $b(t, s)$ in the bit code list defined in the first point, then create a one-hot vector $h_i^{t,s}$ of size 8 with 0 everywhere but a 1 at that index. For example, if the bit code is '011', the corresponding index is 2 and the resulting one-hot vector is :

$$h_i^{t,s} = (0, 0, 1, 0, 0, 0, 0, 0)$$

5. Find the final feature vector for point i h_i by adding the one-hot vectors from all the pairs $(t, s) \in I_i^2$:

$$h_i = \sum_{(t,s)\in I_i^2} h_i^{t,s}$$

Matching accuracy

Matching accuracy (also referred as surrogate accuracy) correspond to the number of selected surrogates that are indeed true matches. Because the data is synthetically created and rotated, we have access to the true mapping. The surrogate selected in our proposed method are compared with the true mapping, and the proportion of true surrogates gives the matching (or surrogate) accuracy.

New parameters

These methods introduce a few new parameters: the number of neighbours to consider for each point when constructing the local feature (k), as well as the number of surrogate to select ($N_{surrogate}$). For Pei's method (or "B-spline" method), we choose a fixed value of k for each scenario. For a specific scenario we compute the surrogates for 10 different datasets for different values of k, and select the value of k that gives the best average matching accuracy. We do this before running the algorithms. In reality, it is not possible to do this as the true mapping is not available, but this allow for better and faster comparisons of our proposed methods for the purpose of this report.

We do the same for Fan's method (or "Angular feature method"), but we also explore another way of selecting the number of neighbours inspired by the authors: using a fixed radius. All neighbours of a point at a distance under this radius are selected to construct the local feature for that point. Similarly to k, we select the best radius for each experiment.

The selection process is further detailed in Section 6.1.

Summary of the methods tested

In total, we therefore test three methods:

- "B-spline" which uses Pei's local features with the best value of k for each scenario.
- "Angular Feature - k", which uses Fan's local features with the best value of k for each scenario.
- "Angular Feature - rad", which uses Fan's local features with the best radius for each scenario.

4.4 Experiments

4.4.1 Base scenario and tested scenarios

For each experiment and method, we test a number of scenarios, each with their own set of parameters. Table 4.2 present the base parameters and the values tested for each parameter (if any, as some parameters will stay fixed).

Parameter	Base value	Test values
Fixed parameters		
n_runs	10	∅
max_restart	110	∅
max_epoch	30	∅
gmm_scale	0.01	∅
loss_distr_scale	1.0	∅
Adam parameters (β_1, β_2, ϵ)	Defaults (0.9,0.999,1e-07)	∅
Data generation		
n_concepts	200	[50,100,300,500]
emb_dim	2	[3,4,5,7,10]
clumpiness	1	[2-2,2-4,2-8,3-2,3-4,3-8]
noise	1e-3	[5e-3,1e-2,2e-2]
Training		
loss_cycle_scale	1e4	[1e3,1e5]
batch_size	100	[50,200]
learning_rate	1e-3	[1e-4]

Table 4.2: Base scenario parameters and tested values.

Each tested parameter constitute what we call a “scenario”, that we will refer in the report using specific names. The scenarios and associated names are:

- clump1 (base) , clump2-2, clump2-4, clump2-8 , clump3-2, clump3-4 and clump3-8. These correspond to different level of ”clumpiness”, as presented in Figure 4.2.
- embdim2 (base) , embdim3, embdim4, embdim5, embdim7, embdim10 - for respective embedding dimensions.
- noise5e-4, noise1e-3 (base), noise5e-3, noise1e-2 and noise2e-2 - for respective levels of noise.
- ncon50 , ncon100, ncon200 (base) , ncon300, ncon500 - for respective number of concepts.
- cycle1e3, cycle1e4 (base) and cycle1e5 - for respective values of loss_cycle_scale.
- bz50 , bz100 (base) , bz200 - for respective batch size.
- lr1e-3 (base), lr1e-4 - for respective values of learning rates

The batch size, learning rate and cycle scale scenarios are only tested for the baseline algorithm, as well as ncon500 and noise5e-4 as their results in the baseline case behave similarly to other tested values. All the scenarios annotated as base correspond to the base scenario and give therefore the same results. They are presented in the context of each varying parameter to allow for better comparison.

We run Algorithm 1 on all the scenarios described above. This means that we perform 10 runs (where new data is created) of 100 restart each (where model f and g are re-initialised randomly). Each restart is trained for 30 epochs and yields a trained model f and g. To allow for unbiased comparison, the synthetic data created at each of the 10 runs use the same seeds, e.g. run 1 for scenario clump2-2 use the same dataset as run 1 for scenario clump3-8.

4.4.2 Measure of success

Top k accuracy

We note $f(X)$ the ensemble of the transformed points from X, and for each transformed point, we estimate its closest k neighbours from Y. If the true correspondence is in the k neighbours, the top k accuracy for f for that point is 1, otherwise it is 0.

For point $x \in X$, let's call:

- $\mathrm{knn}_Y(f(x))$: the k nearest neighbours of $f(x)$ in Y , $x \in X$.
- $\mathrm{acc}_f^k(x)$: the top k accuracy for model f and point $x \in X$.
- $\mathrm{acc}_f^k(X)$: the final top k accuracy for model f.

We have:

$$\mathrm{acc}_f^k(x) = \begin{cases} 1 & \text{, if } f(x) \in \mathrm{knn}_Y(f(x)) \\ 0 & \text{, otherwise} \end{cases}$$

The final model accuracy is determined as:

$$\mathrm{acc}_f^k(X) = \frac{1}{N_X} \sum_{x \in X} \mathrm{acc}_f^k(x)$$

The equivalent accuracy exists for model g and Y, and we finally define the accuracy of restart r as the average top 1 accuracy of model f and g after being trained for 30 epochs:

$$\mathrm{acc}(r) = \frac{1}{2}(\mathrm{acc}_f^1(X) + \mathrm{acc}_g^1(Y)) \tag{4.3}$$

The matching process and top 1 accuracy calculation is illustrated on Figure 4.6.

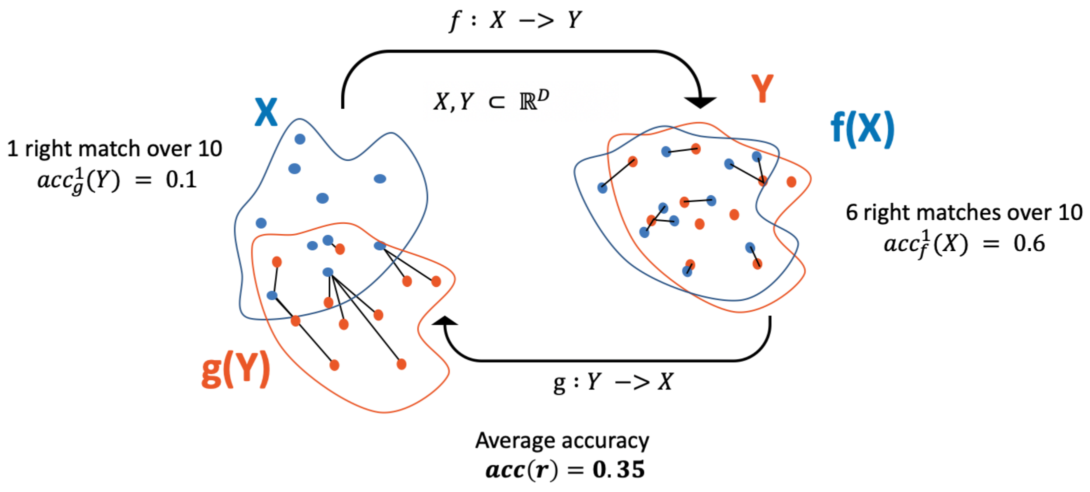


Figure 4.6: Matching process and accuracy calculation. Black lines identify matches. A top 1 accuracy metric is calculated and displayed for both f and g.

Number of successful restarts

From empirical observations, the algorithm seems to behave in a binary way, either it aligns the points well -in which case 30 epochs seem to be enough to train it- or it gets stuck in some local minimum. We measure the success of a restart by comparing its final accuracy (as calculated in eq. 4.3) against the ceiling accuracy of the synthetic data. The ceiling accuracy is the top k accuracy calculated between the ensemble X and the ensemble Y before it gets rotated (with noise only). Such examples are represented on Figure 4.7 for different level of noise, along their ceiling accuracy. With increasing level of noise, the internal structures of X and Y further diverge and the accuracy that the algorithm can reach decreases.

In total, there are 1,000 restarts (100 for each of the 10 datasets tested) for each scenario. The number of successful restarts reported for each scenario and methods is therefore always out of 1,000. The distribution of these successful restarts per run can vary widely, and results presented later display these distributions.

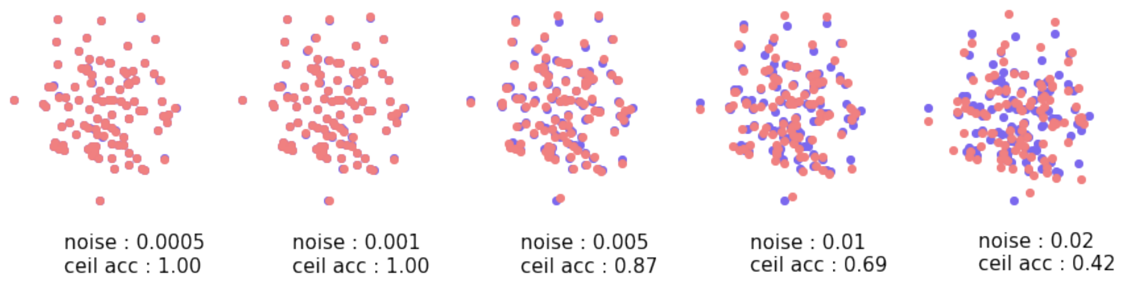


Figure 4.7: Illustration of ceiling accuracy. As noise increases, ceiling accuracy decreases.

4.5 Summary

In the next two chapters, we run the experiments presented in this chapter using both the baseline algorithm and our pre-training algorithms ("B-spline","Angular Features - k" and "Angular Features - rad"). These make use of local features to determine a subset of matching points which serve to pre-train the models. They differ by how the local features are estimated ("B-spline" vs "Angular Features" methods), as well as how the neighbours to build these local features are selected ("Angular Features-k" vs "Angular Features-rad"). We compare their performances using the number of successful restarts per run and per scenario, as described in Section 4.4.2.

Chapter 5

Baseline Results

5.1 Experiment reminder

We test the baseline algorithm on the scenarios described in Section 4.4.1. A scenario corresponds to the base scenario (base values for parameters), except that we change the value of one of these parameters. This is so we can test the algorithms for various configurations of the data. For each scenario, we perform 10 runs (where new synthetic conceptual systems are created each time) of 100 restart each (where the models are re-initialised randomly). Each restart is trained for 30 epochs, and the final top 1 accuracy (taken as the average top 1 accuracy of models f and g) is saved. This is the procedure presented previously in Algorithm 1.

5.2 Results

Figure 5.1 shows the number of successful restarts per scenario and is further broken down per run. Figure 5.2 and 5.3 further details the results per scenario by showing how the top-1 accuracies are distributed (using a violin plot). In total it is $10 \times 100 = 1000$ accuracies that are represented in each violin plot. We also calculate the number of total "successful" restarts : a restart is successful if its final average accuracy is above a threshold value, that we set to be 0.9 times the ceiling accuracy. The ceiling accuracy is determined by the level of noise in the original dataset and is the maximum accuracy we can expect to reach in a run.

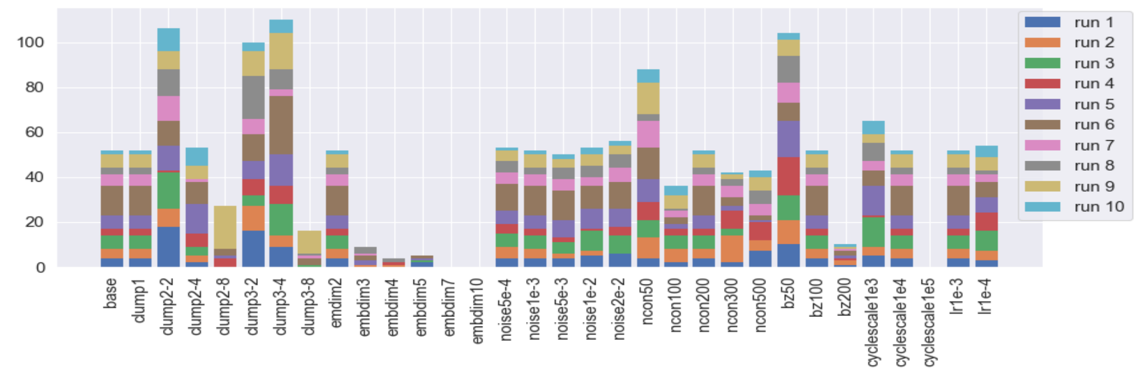


Figure 5.1: Successful restarts per scenario and per run.
The base scenario has around 50 restarts successful (over 1,000). A little linear separability in the clumpiness scenario seems to improve the number of successes whilst high linear separability has the opposite effect. Increasing the embedding dimensions results in a significant performance loss, whilst varying the noise levels has negligible impact. A number of concepts of 50 seems to give better results, but there is no clear relationship observable for other number of concepts. Finally, a smaller batch size and lower cycle loss scale seem to be beneficial, but lowering the learning rate has little impact.

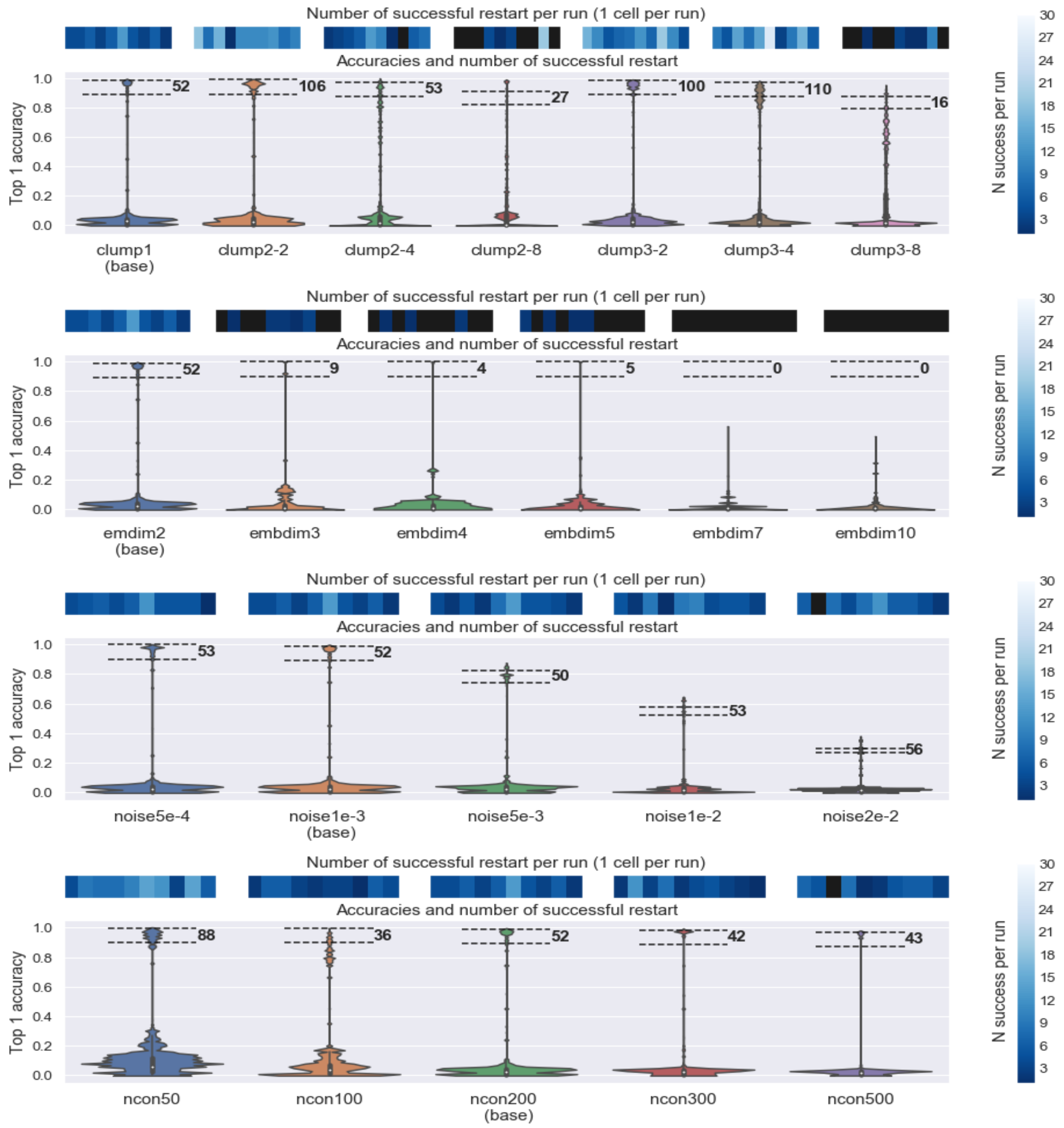


Figure 5.2: Baseline algorithm - Distribution of restart accuracies and number of successes.
Reading instructions : For each scenario (i.e. violin plot), we create 10 different datasets (10 runs) and perform 100 restarts on each (see Algorithm 1). The two dotted lines represent the average ceiling and threshold accuracy of the 10 runs. The threshold value is defined as 0.9×ceiling accuracy. For a specific run, if the restart accuracy is higher than the run's threshold, it is considered a success. For each scenario, the total number of successful restarts is displayed, and is further broken down per run using a colormap. The black color indicates no successful restart for a run. **Notable results :** The base scenario has 52 successful restarts over 1,000 in total. A little clumpiness (i.e. several gaussians with little linear separability) increase the number of successes, but too much does the opposite (clump2-8 and clump3-8). The number of successful restarts sharply decreases when increasing dimension, with 0 successes for dimension 7 and 10. Interestingly,the algorithm is not sensitive to noise with around 50 successes for all level explored. Using a number of concepts of 50 seems advantageous, but there is no clear difference between using 100,200,300 or 500 concepts.

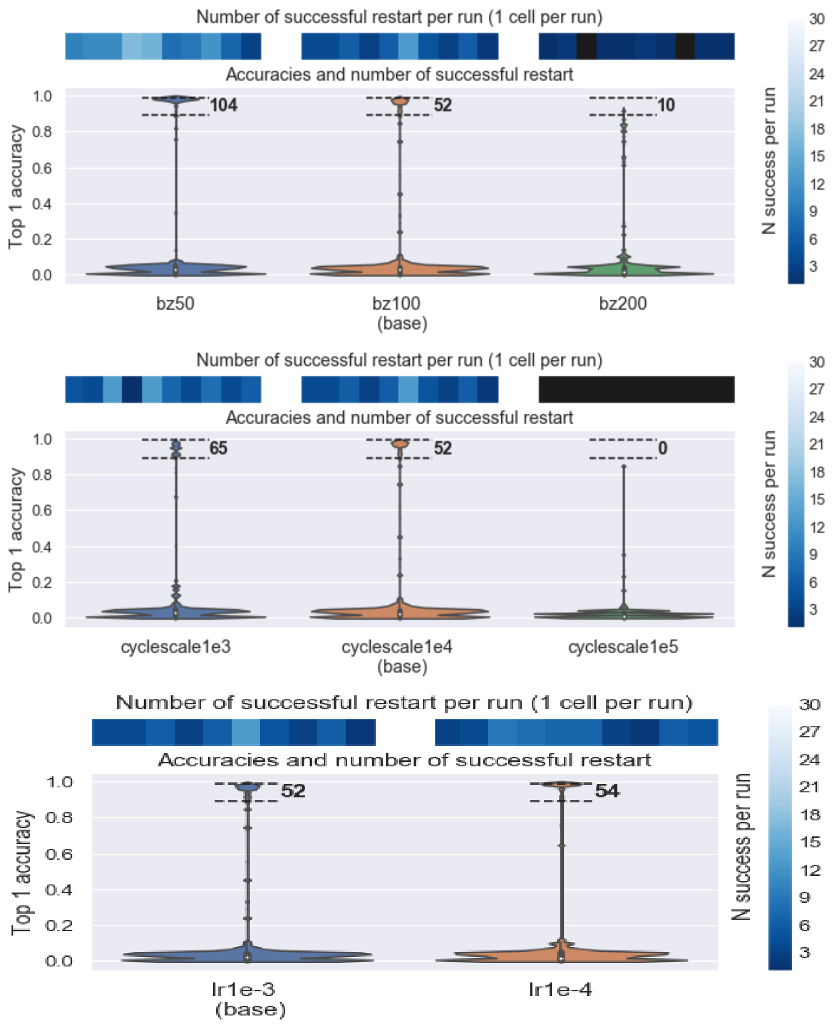


Figure 5.3: Baseline algorithm - Distribution of restart accuracies and number of successes. **Reading instructions :** See Figure 5.2. **Notable results :** A lower batch size increases the success rate by two-fold, likely helping the models to get out of local minimums. A cycle scale of 1e3 instead of 1e4 gives slightly better but similar results, whilst a value of 1e5 gives no success. Finally, adopting a learning rate of 1e-4 rather than 1-3 does not impact successes much.

Impact of clumpiness

Recall that the clumpiness scenarios are presented on Figure 4.2 for reference. Clump2-8 and clump3-8 perform particularly badly, with a high proportions of runs with about half the runs having no successful restarts (represented by black squares on Figure 5.2. These scenarios corresponds to 2 and 3 generating gaussians which are far apart. We can also note that the ceiling accuracy is lower for these two scenarios than for the others. This is because to create these very distinct generating gaussians, we have "clumped" the points together, which mean that the average distance between neighbours in one of these clumps is lower than for other clumpiness scenarios. This means that the noise (which is fixed), will have a greater impact on the ceiling accuracy. Clump1 and clump2-4 perform equally whilst clump 2-2,3-2 and 3-4 give the best results. This seems to suggest that introducing a little more structure whilst still having a certain level of overlap between the gaussians give the best results. When the distance between the gaussians is too high, the model performs worst. The impact of noise could play in this result, but we will see that noise only does not impact the success rate, hence suggesting that the main factor playing is indeed clumpiness.

Impact of embedding dimension

It is clear from the results that the performance of the algorithm decreases as the dimension goes up, with no successful restarts at all for dimensions 7 and 10. There is no significative difference between the numbers of successful restarts for dimensions 3,4 and 5 (respectively 9,4 and 5 successes), but they all perform much worst than the base algorithm (52 successes).

Impact of noise

Although the effect of noise on the ceiling accuracy is very visible from Figure 5.2, noise does not seem to affect the success rate of the algorithm at all.

Impact of number of concepts

The algorithm seems best suited for 50 concepts, but performs worst for 100 concepts than for respectively 200, 300 and 500 concepts where the performance is similar. Note that for 100 concepts, there are some clusters of restart accuracies just below the threshold. These are therefore not counted as successes, but the difference in distribution of the restart accuracies between the ncon50/100 and ncon200/300/500 is notable. Whilst the latter is very binary (either close to 0 or 1), the former span a wider range of accuracies.

Impact of batch size, cycle scale and learning rate

A batch size of 50 rather than 100 multiplies the number of successes by 2 whilst a batch size of 200 (which means there is no batch as the number of concepts is 200) show very little successes. It is likely that a lower batch size helps the models to get out of local optimum by adjusting the cycle loss per batch.

Each loss is scaled empirically to maximise efficiency. The base values are a scale of 1 for the distribution loss and a scale of 1e4 for the cycle loss. We tried changing that latter scale to 1e3 and 1e5. A scale of 1e5 gives a zero success rate, whilst a scale of 1e3 gives a slightly higher one, although not significantly. As with the number of concepts, a lower cycle scale seems to impact the distribution of the restart accuracies, which span a slightly wider range.

Finally, having a lower learning rate seems to have little impact.

5.3 Summary

To conclude, we see that the baseline algorithm shows some (but quite limited) level of success (around 50 successful restarts out of 1,000). Surprisingly, it seems not impacted by the level of noise, and little impacted by the number of concepts. A little clumpiness improve performance but the opposite effect is observed for too clumpy systems. The algorithm is also very sensitive to embedding dimension.

Chapter 6

Results with Pre-training

In this section, we test the proposal from Section 4.3.1 using both the method from Pei et al [38] (B-spline curves) and from Fan et al. [40] (angular features). We test these on most of the same scenarios as described in Section 4.4.1. We however drop the scenarios involving learning rates, batch size and cycle scale as we are more interested in how the algorithms react to data structures, leaving us testing the clumpiness, embedding dimension, level of noise and number of concepts. We also don't test for a level of noise of $5e-4$ or a number of concepts of 500 as these two scenarios seem to behave quite similarly to their closest neighbour (respectively a level of noise of $1e-3$ and a number of concepts of 300).

6.1 Choice of new parameters

As a reminder, the algorithms consist in matching the points from each system which are the closest in feature distance, and use this information to pretrain the models as if they were true correspondences. The new parameters introduced are:

- $N_{surrogate}$: The number of surrogates used. We choose the value 10.
- k : The number of neighbours to consider to build the local features. As mentioned before, we find k that gives the best number of true correspondences for the scenario-specific set of parameters. We do this both for the 'B-spline' and 'angular feature' methods.
- *radius* : Instead of choosing a fixed number of neighbours, it is possible to choose a fixed radius: all the neighbours at a distance less than the radius are selected. As for k, we choose the best radius for each scenario.

When a best k or best *radius* is chosen for a specific scenario, we create 10 different random datasets based on the scenario-specific parameters, and for each dataset we test different values of k or *radius*. We record the number of true correspondences amongst the surrogates to estimate the matching accuracy, which we average for each parameter tested over the 10 runs. The parameter with the highest average is selected. The values tested are:

$$\begin{cases} k\_tested = \{5 : 20, step = 1\} \\ radius\_tested = \{0.01 : 0.05, step = 0.01\} \cup \{0.05 : 0.5, step = 0.5\} \end{cases}$$

Finally for the 'B-spline' method, there is an additional parameter β, which plays in Equation 4.1. It weights the importance of the difference in the gradient curves areas with respect to the difference in the B-spline curves areas. We use a fixed value of 1. This was selected empirically by comparing the number of true correspondences within the surrogates for selected scenarios.

In summary, we run all the scenarios for:

1. 'B-spline' method. Best k.
2. 'Angular features' method. Best k.
3. 'Angular features' method. Best radius.

6.2 Comparing results for the base scenario

The entirety of the results are presented in Appendix A,B and C. We present on Figure 6.1 the detailed results for the base scenario, comparing each method.

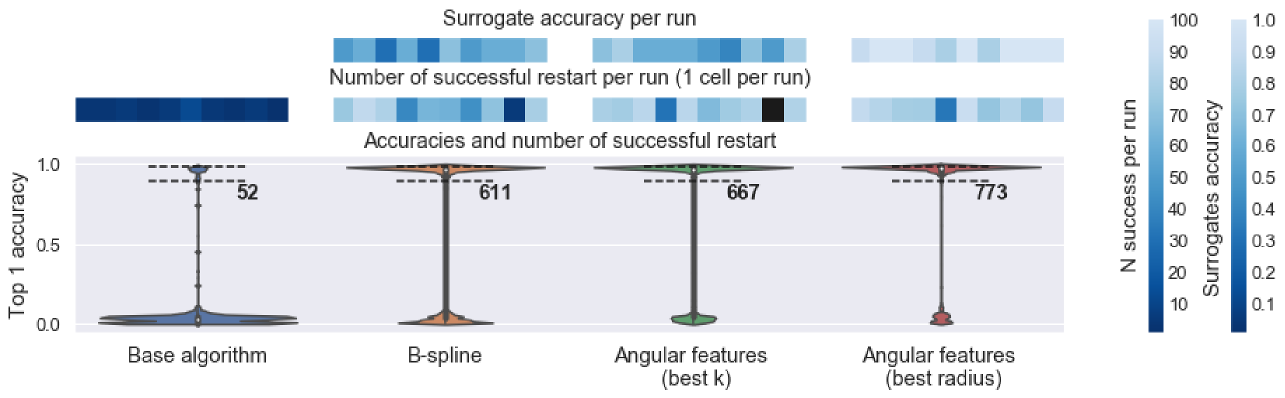


Figure 6.1: Number of successes for the base scenario for various algorithms.
Reading instructions : See Figure 5.2. In addition to the colormap representing the number of successes per run (10 runs for each scenarios), we add here a colormap corresponding to the matching accuracy of the surrogates (which is define per run, as one run corresponds to one dataset and surrogates are defined per dataset). This allows to qualitatively check the relationship between matching accuracy and number of successful restarts for a given run. **Notable results :** All methods explored considerably increase the number of successes, from 52 to more than 600 (over 1,000 restarts in total). The angular feature methods perform better, with the one selecting neighbours based on best radius winning the contest.

We note considerable better results using any of the three method. Additionally, the "angular features" method seem to perform better with 667 and 773 restarts against 611, and out of the two, the one using the best radiant gives the best results. Note that the datasets were created using the same random seeds for each, hence each run (represented by squares in the colormaps) are directly comparable. This will remain true throughout the report. Compared with the previous way of presenting results, we have also added a colormap at the top showing the matching accuracy of the surrogates per run

6.3 Comparing total successes for all scenarios

Figure 6.2 shows the number of successful restarts over the 10 runs for each scenario tested, for the base algorithm, and the three pre-training methods tested.

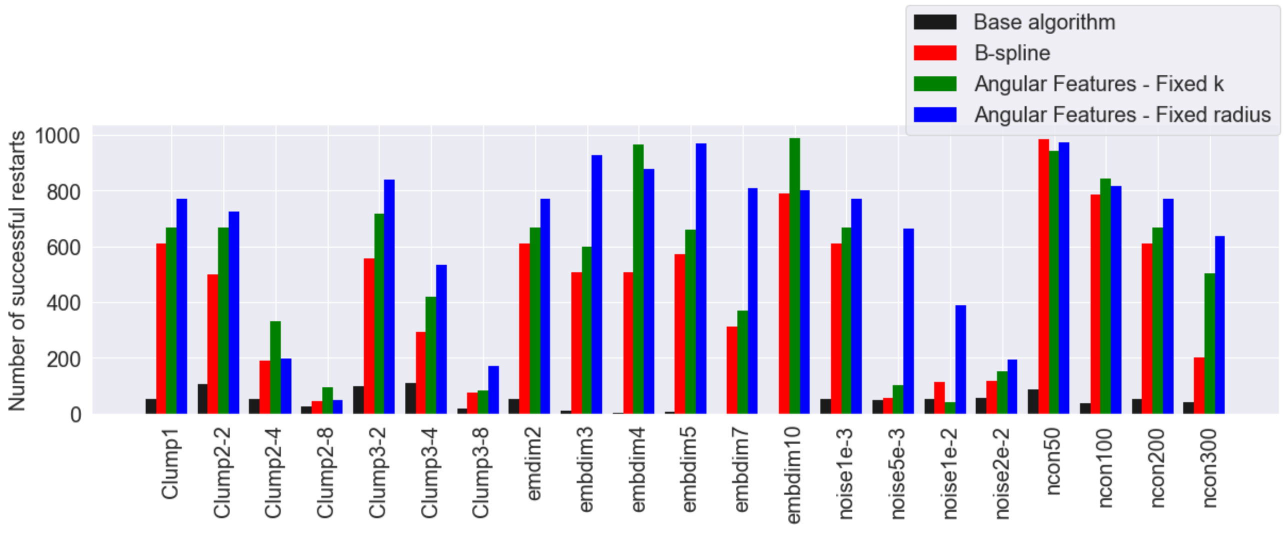


Figure 6.2: Successful restarts per scenario (1,000 restarts in total for each scenario) - Comparison of the algorithms. All three methods perform better on almost all scenarios. This is particularly true for embedding dimensions of up to 10 where the baseline algorithm was performing particularly badly. The "B-spline" and "Angular Features - Fixed k" methods don't perform well for increased noise, but the "Angular Features - Fixed radius" one performs much better than the baseline algorithm. However, its number of successes decrease when increasing noise, whilst it is constant for the baseline method. Results are further detailed per parameter in the main text.

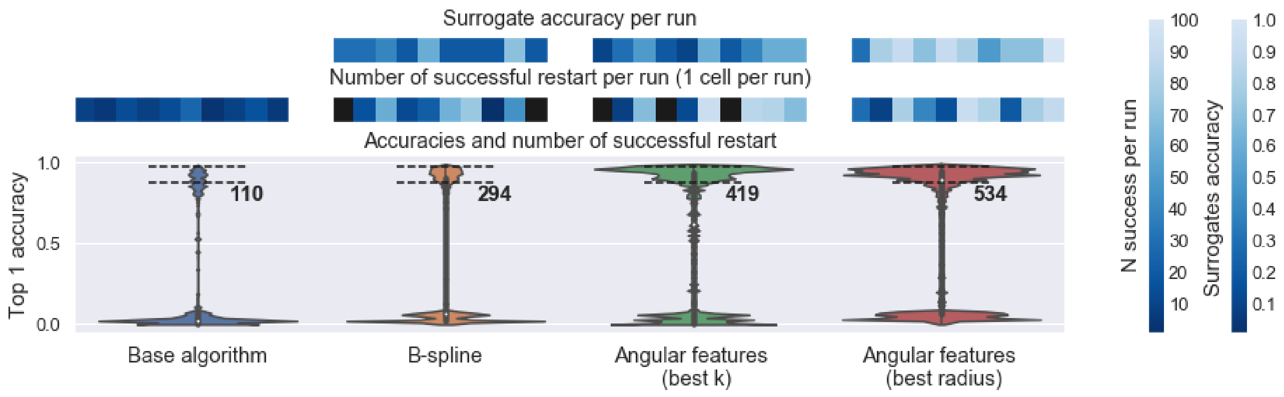
Recall that the scenarios called "clump1" , "embdim2" , "noise1e-3" and "ncon200" all corresponds to the base scenario, but are presented in the context of the varying parameter for more clarity.

The proposed methods show a very significant improvement of the performances in almost all the scenarios, except for some increased level of noises and for very clumpy systems. A deeper analysis is conducted in the next section.

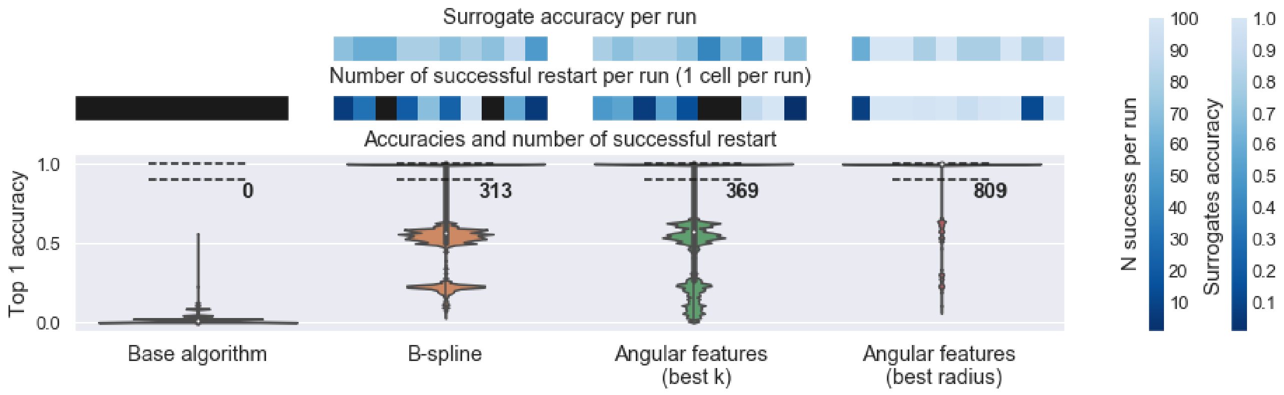
6.4 General results and zoom on selected scenarios

In this section, we analyse the difference of the methods for each vatying parameter based on the bar chart on Figure 6.2. We also select four scenarios (one for each parameter) and display their results on Figures 6.3 and 6.4. Those scenarios are clump3-4','embdim7','noise1e-2' and 'ncon300' and were selected as they show clear improvements with respect to the baseline method.

Results for scenarios "clump3-4", "embdim7", "noise1e-2" and "ncon300" are presented on Figures 6.3, we can see that the 'Angular Features - best radius' method has 534 successful restarts, with no unsuccessful runs (all runs at least have one successful restart at least) and 6.4. We detail here results for each varying parameter based on Figure 6.2 (total number of successes), as well as zooming on the selected scenarios.



(a) Scenario “clump3-4”



(b) Scenario “embdim7”

Figure 6.3: Distribution of number of successes for the baseline algorithm and the three proposed methods - Scenarios “Clump3-4” and “embdim7”. **Subfigure (a).** Scenario “Clump3-4” corresponds to 3 gaussians with medium linear separability. The “B-spline” method performs better than the baseline algorithm with almost 300 good restarts against 110 (over 1,000 in total). “Angular Features” methods outperform both with 419 and 534 restarts. Note that for the “B-spline” and “Angular Features - best k” methods, some of the runs were completely unsuccsessful (shown by black squares) whereas this is not the case for baseline and “Angular Features - best radius”. **Subfigure (b).** For an embedding dimension of 7, we go from 0 successes with the baseline algorithm to more than 300 with the pre-training method. Again, the “Angular Features-Best radius” performs better than the others with about 800 successes. Many of the runs for the “B-spline” and “Angular Features - best k” methods seem to have gotten stuck at a top-1 accuracy of around 0.5.

Clumpiness

The algorithm still deals relatively badly with clumpiness, with the worst results being obtained for ’clump2-8’ and ’clump3-8’ (respectively 2 and 3 highly linearly separable clumps). It seems however that all of the new methods tested perform better than the baseline algorithm. The angular feature method using a fixed radius give better results for all except ’clump2-4’ and ’clump2-8’, where the ”Angular Features- fixed k’ method perform better. In the ’Clump3-4’ scenario presented on Figure 6.3, we can see that the 'Angular Features - best radius' method has 534 successful restarts, with no unsuccessful runs (all runs have one successful restart at least). Surprisingly, the baseline method also has no unsuccessful runs, but none of the run is highly successful and there are only 110 successful restarts. The 'B-spline' method has 294 successful restarts with 2 unsuccessful runs (black squares) and three highly successful runs (green squares). Interestingly, the successful runs do not seem to be the one with the highest matching accuracy. For the "Angular-best k" method, there are 419 successful restarts but 3 unsuccessful runs. Hence the strategy adopted seems to be "hit-or-miss" for this specific scenario for the "B-spline" and "Angular-best k" methods. Additionally, the average matching accuracy for the methods selecting a best value of k ("B-spline" and "Angular-best k") are much smaller on average than for the "Angular - best radius" method. The "Angular - best radius" method only has about a 100 more successful restarts then the "Angular-best k" method, but these successes are better distributed between runs.

Embedding dimension

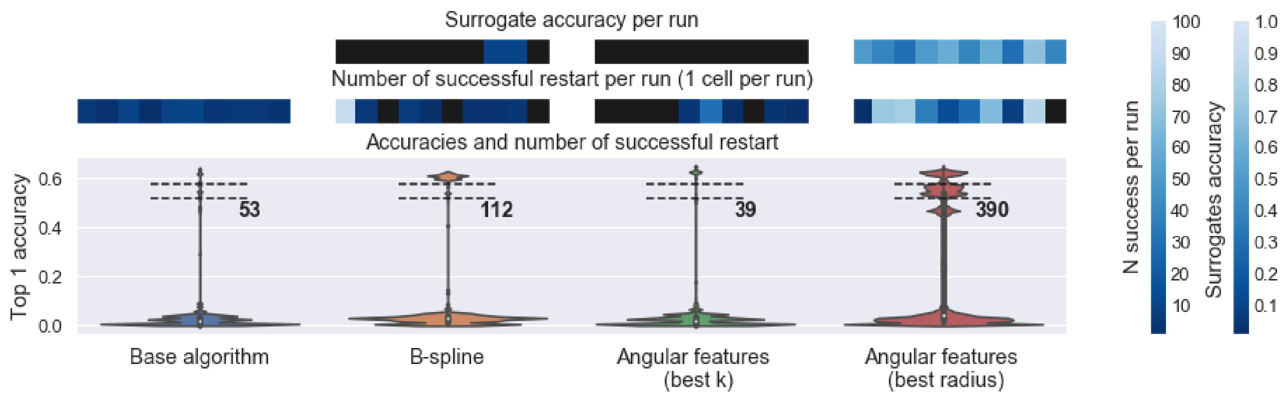
Pre-training the models using local structure data was particularly beneficial to the scenarios with embeddings of dimensions 7 and 10, where the baseline algorithm showed no success at all over the 10 runs. The examples shown on Figure 6.3 for 'embdim7' illustrates this result. There is a strange phenomenon for dimension 7 which is not observed for other dimensions where many of the restarts gets stuck at around 50% top 1 accuracy. Both the "B-spline" and "Angular-best k" methods show a similar number of successes (313 and 369) and a similar pattern. Each has 2 unsuccessful runs, and most of the successful restarts only come from 2 runs (i.e. 2 different datasets). The matching accuracy for both these methods is however relatively good compared to the 'clump3-4' scenario. In the "Angular - best radius" method, the matching accuracy is even better, and 8 of the runs are very successful, with no runs being completely unsuccessful.

Noise

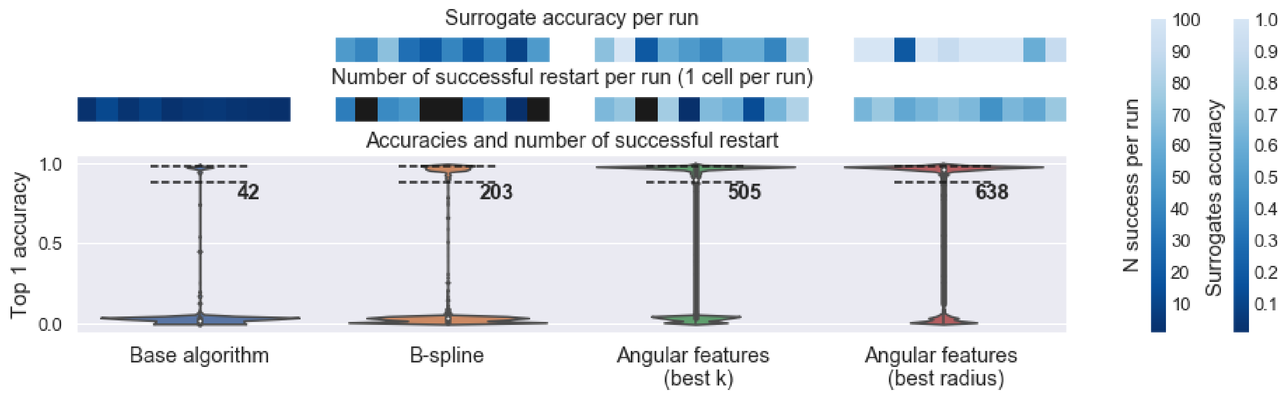
The “Angular Features - radius” method shows a significant improvement on all the noise scenarios. The two methods using a fixed k (“B-spline” and “Angular - Fixed k”) perform much better than the baseline algorithm for low level of noises but their performances on higher level of noises is unclear, sometimes at the same level as the baseline method. This is illustrated on Figure 6.4 for the scenario “noise1e-2”. The matching accuracy for both the “B-spline” and “Angular Features - best k” methods is almost zero everywhere. This leads to many unsuccessful or little successful runs, especially so for the “Angular Features - best k” method.

Interestingly, the "Angular Features - best radius" method displays medium matching accuracy values, but it is apparently enough to guide the models in the right direction as there are 390 successful restarts (coming mainly from 4 runs) and only one run is completely unsuccessful.

Note that many of the successful restarts reach a top 1 accuracy that is above the average ceiling accuracy. As a reminder, the average ceiling accuracy is just an average, and that number varies for each run. This is also true for the threshold line (bottom line of the two for a violin plot). Furthermore, for a specific run, all restarts above the run's threshold are counted as successful, even if they are above the ceiling accuracy by chance. Hence the number of successful restarts is not limited to only the restarts between the tow dotted line represented, but also include the restarts with higher accuracies.



(a) Scenario “noise1e-2”



(b) Scenario “ncon300”

Figure 6.4: Distribution of number of successes for the baseline algorithm and the three proposed methods - Scenarios “noise1e-2” and “ncon300” **Subfigure (a)**. For a noise level of 1e-2, only the “Angular Features - best radius” method seems to show a much increased performance with 390 successes. “Angular Features - best k” performs worst and “B-spline” better but most of the successful restarts seem to come from 1 run (light blue colored squared). **Subfigure (b).** For 300 concepts, we have again the “Angular Features - best radius” performing the best with 638 successes, followed by “Angular Features - best k” with 505 and a bit further behind “B-spline” with 203. In comparison, the base algorithm only has 42 successes.

Number of concepts

All proposed methods show a significant improvement compared to the baseline one, but performance seem to steadily decrease when the number of concepts increases, particularly for the 'B-spline method'. The scenario 'ncon300' on Figure 6.4 illustrates this. The "B-spline" method has 203 successful restarts, with 4 totally unsuccessful runs. The successes seem however evenly distributed amongst the other runs. Matching accuracy is on average low. The "Angular Features - best k" method shows a better matching accuracy (although it is still much lower on average than for the last method) with only one unsuccessful run and 505 successful restarts. The last method - "Angular Features - best radius" - shows a very high matching accuracy, with no unsuccessful runs and 638 successful restarts.

Conclusion

Overall, the angular features method using a fixed radius outperforms the other two methods. Additionally, any of the proposed method strongly outperforms the baseline method. In the next section, we will conduct some further analysis to understand and interpret better these results. Notably, we have for now little information about how many neighbours are selected in the radius method. We will see in the next section that the average number of neighbours is much higher for the radius method than for the k method, hence although boasting higher performances, it is more computationally complex.

6.5 Further analysis

In this section, we dive deeper into some aspects of the proposed pte-training methods to understand better their dynamic. We notably explore the correlation between the matches accuracy and the number of successful restarts and the impact of lowering the number of surrogates. We also analyse how the best k or best radius is chosen for each scenario. As a reminder, this parameter is used to select a number of neighbours for each point within a system in order to construct its local feature. We finally explore the number of neighbours selected when using the radius selection methods (where all the points within the radius are selected as neighbours), and draw a comparison with the "Fixed k" method (where all the points have a fixed number k of neighbours).

6.5.1 Correlation between number of successful restarts and matches accuracy

From the detailed results in Appendix A,B and C, we see that the number of successful restarts is somewhat correlated with the matches accuracy amongst the surrogates. This seems intuitive as this matching process is also the only difference between the three methods, and also because this is the main motivation for using them. However, it is not always true and there are cases were the two seems uncorrelated. We experimented running a few of the least successful scenarios ("clump3-8" and "noise 2e-2") using 10 exact matches (unavailable in a real situation) to see how the models would react. Results are presented on Figure 6.5.

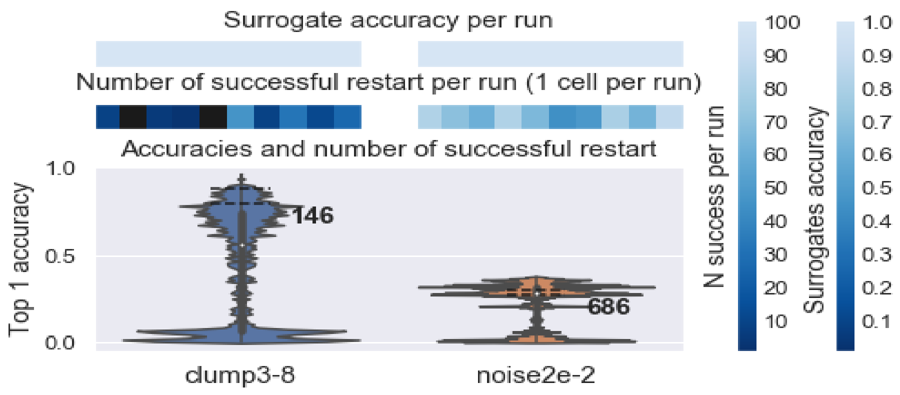


Figure 6.5: Results when using 10 true matches for two example scenarios. There seem to be no correlation between the success rate of a run and its matching accuracy in scenario "clump3-8".

As can be seen, the runs for scenario "clump3-8" are still quite unsuccessful, with 2 runs showing no successes at all and only 146 good restarts, against 171 for the "Angular features - best radius" method as shown in Appendix C.1. The scenario "noise2e-2" however reaches 686 vs 195 good restarts for the most successful of the three other methods. This shows that one could just focus on finding local features yielding a high matches accuracy in a number of scenarios, and this would get better results. However, there are limitations to this thinking, as some scenarios don't respond well to this pretraining method, even when using 100% matches accuracy.

An angle to improve this work could therefore be to focus on local features that improve matching accuracy for specific scenarios. This approach has limits as even using 10 true matches can lead to relatively bad results for some scenarios. It can however improve results significantly for non-clumpy noisy scenarios.

Finding right matches for noisy scenarios remains however a challenge. An idea to improve accuracy could be to match points with the most singular local structure. For instance, we can see on Figures 6.11.b and .c that some of the points are more isolated, and have much less neighbours than others. Let's consider a pair of corresponding points in two separate domains that verify this. Although their local structure similarity might not be the highest in absolute terms, the similarity with the next closest points might be significantly lower, strongly indicating they could be a match. In other words, there might be less points competing for the title of "match" to a singular point than for a more "regular" point. There is no such mechanisms integrated in the current methods, and this could be an area to explore.

6.5.2 Impact of number of surrogates

We also tested to lower the number of surrogates to 3, to see how results would be impacted. Results for the "B-spline" scenario are presented in Figure 6.6.

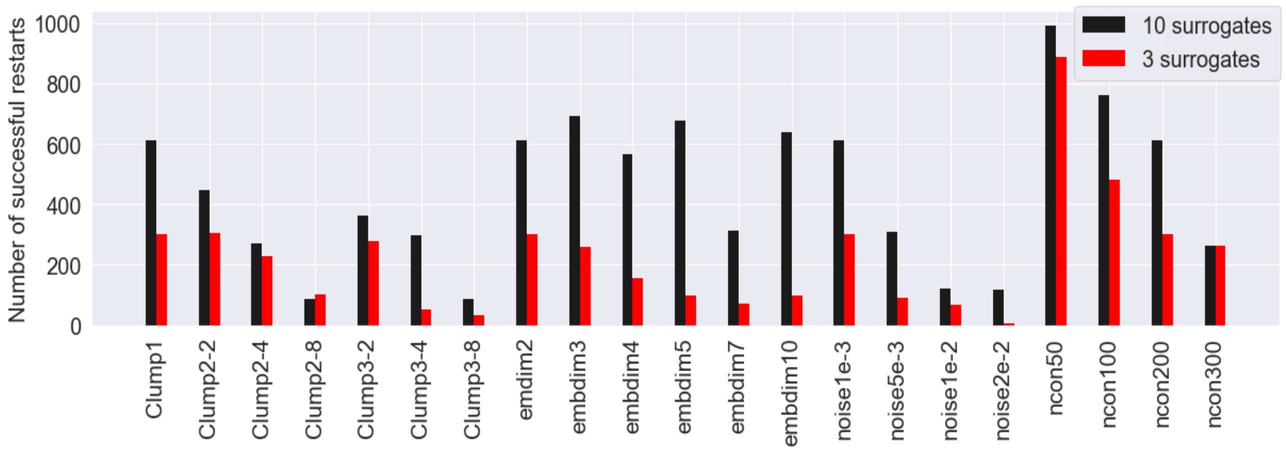


Figure 6.6: Number of successful restarts per scenario for the B-spline method- Varying number of surrogates. Using 3 surrogates instead of 10 decreases the success rate in almost all scenarios tested.

Results for 10 surrogates are better for almost all scenarios. Further analysis of the impact of this parameter on accuracy could be done, but we work with a fixed value of 10 for the purpose of this work. Our intuition is that using only 3 surrogates does not hedge the risks of getting a wrong match enough. One wrong match already limits the total matches accuracy up to 66%. It also provides less data for the models to be pretrained on.

6.5.3 Analysis on best parameter chosen

On Figure 6.7, 6.8 and 6.9, we explore the Surrogate matches accuracy for different values of k or for different radius. As explained in Section 4.3.2 and 6.1, we create 10 datasets and estimate their 10 best surrogates for each parameter value. The number of true correspondences out of these 10 gives the matches accuracy. We report in Figures 6.7, 6.8 and 6.9 the average matches accuracy for each scenario over the 10 runs.

We note that for the B-spline method, there is no clear k that gives good accuracy. Indeed, the accuracy seems to be quite noisy. Both the angular methods show clearer trends and reach higher accuracies, with the fixed radius method clearly winning. This is further illustrated in Figure 6.10, where a box plot is plotted for each parameter tested, representing the distribution of the matches accuracy for the 10 runs for the base scenario only.

B-Spline

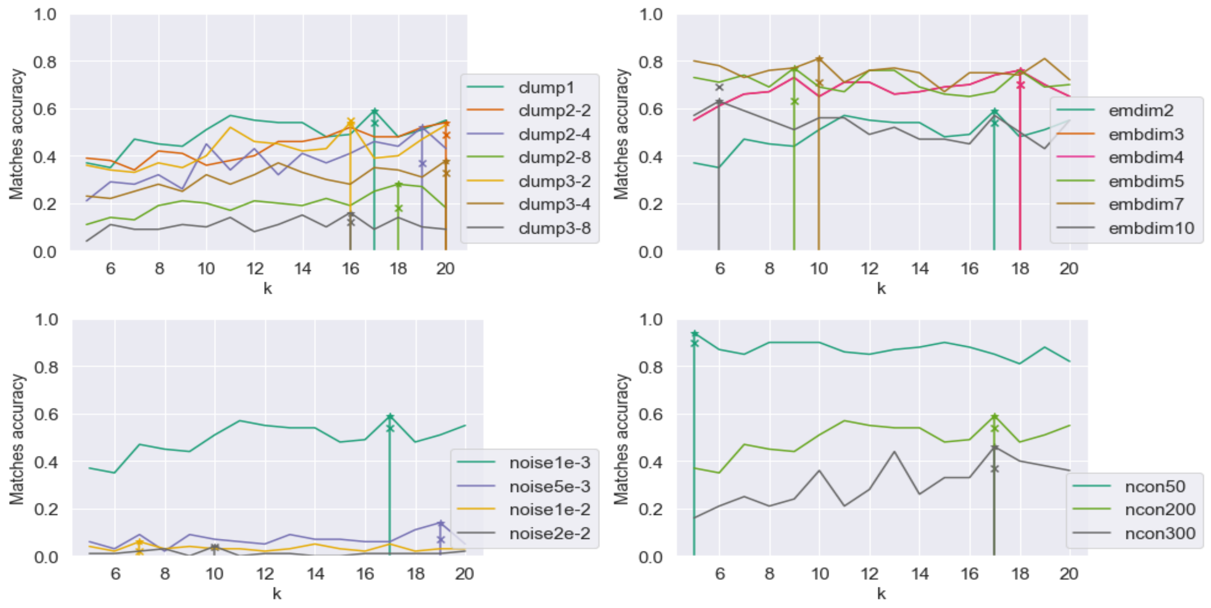


Figure 6.7: B-spline algorithm : Matches accuracy for all scenarios for different values of k. There seem to be a lot of variance with an "unclear winner" for the best value k.

Angular Feature -k

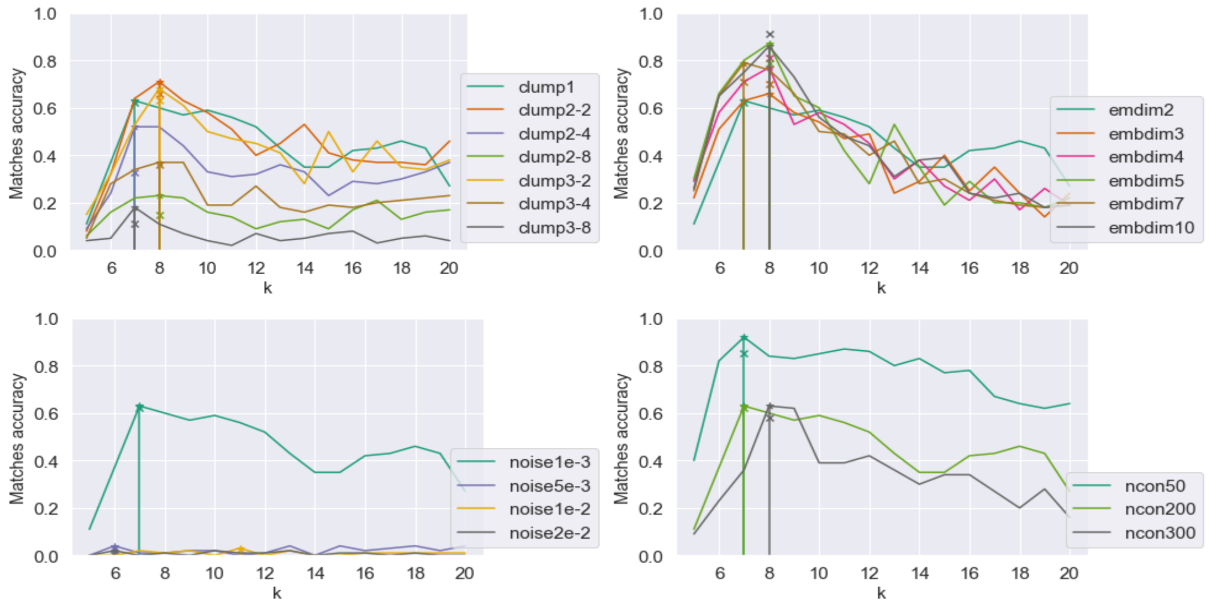


Figure 6.8: Angular Features algorithm : Matches accuracy for all scenarios for different values of k. The best values are all consistently around 7 or 8.

Angular Feature -radius

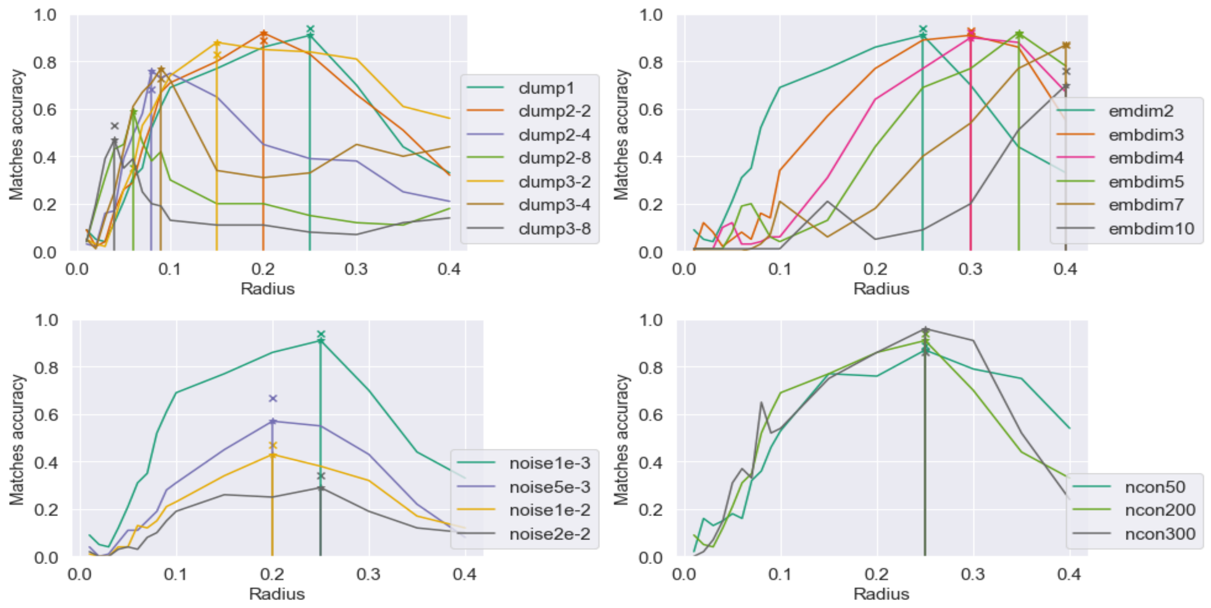


Figure 6.9: Angular Features algorithm : Matches accuracy for all scenarios for different radius. There is a very clear peak indicating what the best radius is for each scenario. Unlike the ”fixed k” method, the best radius varies with some the parameters values.

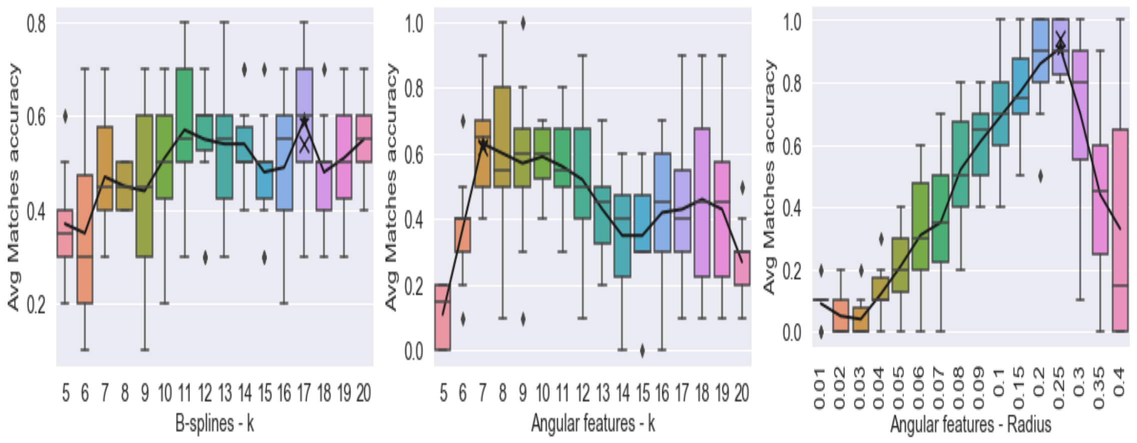


Figure 6.10: Matches accuracy per parameter - Zoom on base scenario. This comparison makes clear the difference of the best k or radius selection, with a much clearer ”winner” for the angular features methods, especially for the method using a radius rather than a fixed k to select the neighbours.

We note some other interesting results. Notably, best k values for the B-spline method span a wider range than for the angular method with a fixed k, where almost all the best k values are around 7 or 8. Our assumption is that this is due to the high variance of the former.

For the angular method with a fixed radius, the best radius varies following distinct patterns. For instance, the best radius is unchanged (or almost) for various level of noise or number of concepts. For clumpier scenarios (“clump2-8” and “clump3-8” being the clumpiest, and “clump1”,“clump2-2” and “clump3-2” the least clumpy, see Figure 4.2), the radius is lower. In effect and as mentioned previously in Section 5, clumpier scenarios mean that the points are closer together. Hence this seems to suggest that the best radius is “adapting” so the average number of neighbours is optimal. Finally, the best radius increases steadily as the dimension increase, which seems logical as distances between points increase as the dimension grows. This also seems to suggest that the best radius still selects a number of optimal neighbours. However, a notable difference with the methods using a fixed k is that this number of neighbours vary from point to point, hence capturing precious local information about the neighbours density around a point.

In the next subsection, we explore the average number of neighbours resulting from using the best radius for each scenario.

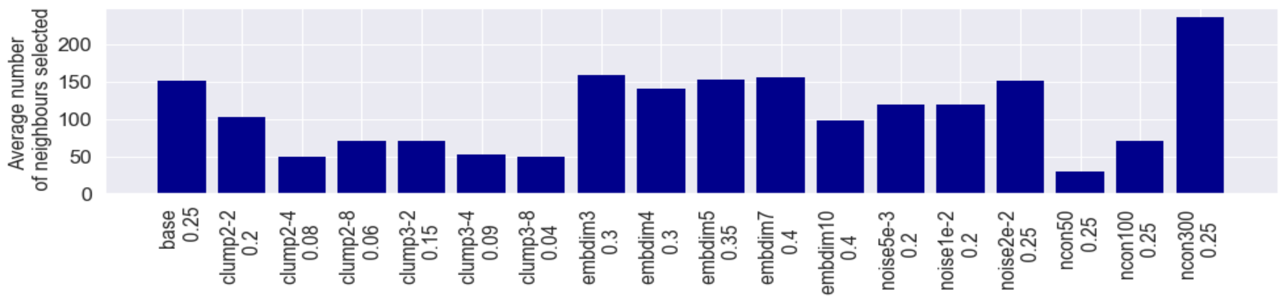
6.5.4 Best radius : average number of neighbours

Figure 6.11 shows per scenario:

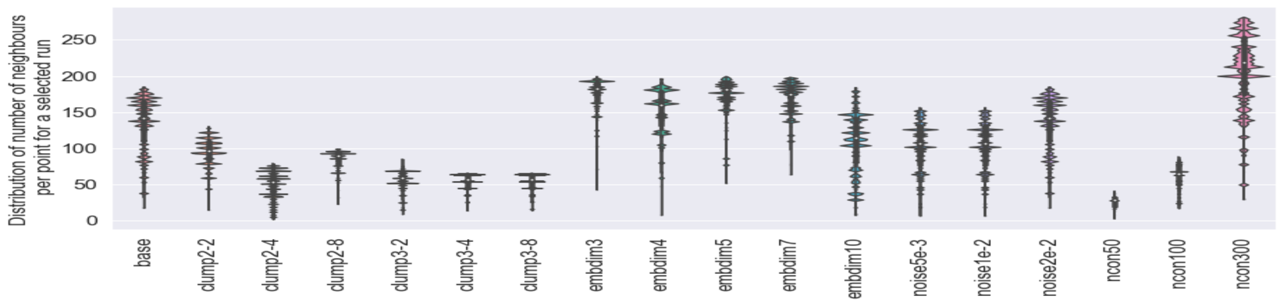
- Subfigure (a) : The average number of neighbours per scenario for the selected best radius (which is reported next to the scenario’s name).
- The distribution of the number of neighbours per point for an example dataset for:
  - Subfigure (b). the best radius.
  - Subfigure (c). a fixed radius of 0.25 (i.e. the best radius for the base scenario)

Note that all of these scenarios have 200 concepts (except the ’ncon’ scenarios). For some of these scenarios, the average number of neighbours reaches 150, with some points having up to 200 neighbours and some other less than 50. This means that the selected radius spans a wide proportion of the total range. When choosing a fixed k, we only tried values up to 20. We further test the matches accuracy (average over 10 datasets) for a fixed k for values from 5 to 200 for the base scenario to draw a better comparison. The results is presented on Figure 6.12. We can see that there is no clear peaks, and that testing values from 5 to 20 in the previous experiment was a reasonable choice. As mentioned before, considering higher values of k also results in a higher computational complexity, hence choosing lower values of k can be an advantage, even though the radius method shows better performances.

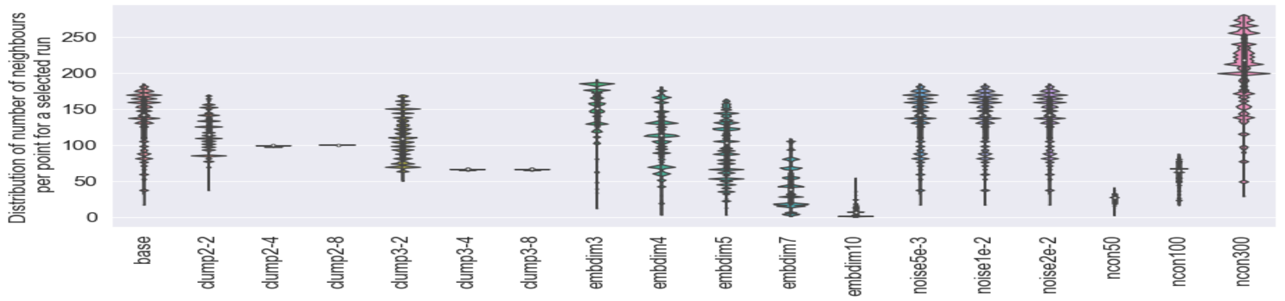
From Figure 6.11.b and 6.11.c, we see that for the scenarios testing for different embedding dimensions, the radius chosen is increased as the dimension increases, so that the distribution of the number of neighbours per point become roughly the same. Note that for dimension 10, the distribution is not quite similar but observing Figure 6.9 shows that the true best radius would be outside of the range of the radius tested, hence it is ”on the way” to get to the same distribution. Our intuition from the previous section that is adapts to select the same number of neighbours on average is therefore verified.



(a) Average neighbours - best radius



(b) Distribution of number of neighbours (example dataset) - best radius



(c) Distribution of number of neighbours (example dataset) - Fixed radius of 0.25

Figure 6.11: Number of neighbours selected by best/fixed radius per scenario. **Reading instructions :** Subfigure (a) shows the average number of selected neighbours (over all points) for each scenario when using this said scenario's best radius. Subfigures (b) and (c) show the distribution of the number of selected neighbours for all the points of an example conceptual system (created using the scenario-specific parameters).Subfigure (b) uses the best radius for each scenario whilst (c) uses a fixed radius of 0.25 for all scenarios.**Notable results :**We note the high value of this average compared to the total number of concepts of 200 (except for the ncon scenarios where the number of concepts varies). We also see how, when the best radius is used in subfigure (b), the distribution looks similar for each scenarios despite a difference in the total range. This is not the case if we use a fixed value of 0.25 (for instance) as in subfigure (c)

For the number of concepts, the distribution is also similar but scaled up or down with respects to the number of concepts, and there is no adjustment of the best radius for these scenarios. Similarly the noise scenarios already all have a similar distributions with a fixed radius, and are not (or barely) adjusted.

The situation for the clumpiness scenarios is a bit more complex. "clump2-4" and "clump2-8"'s dataset are generated using a mixture of two "linearly separated" gaussians, resulting in two "clumps". A fixed radius of 0.25 means that each point from a clump has all the other points from that clump as neighbours, hence all points have 100 neighbours. For “clump3-4” and “clump3-8” it is the same but with three ”clumps”, and they all have 66 neighbours. This is a coincidence on this example as there exists some cases were the clumps are closer, resulting in a varying number of neighbours if the radius from a point in a clump overlaps with another clump, but this example was chosen to demonstrate how the selected best radius change that distribution. The radius selected is much smaller than the other ones for these four examples, and also results in a similarly shaped distribution, although scaled down as acting “within clumps”. It is like having several systems with half or a third of the number of concepts.

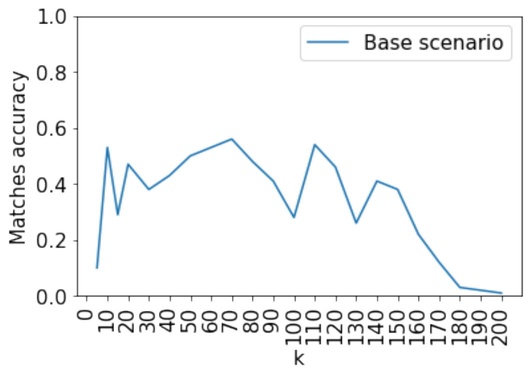


Figure 6.12: Average matches accuracy for the ”Angular Features - Fixed k” method when choosing different values of k (base scenario). This graph shows that, when using a fixed k for selecting neighbours, choosing values that are closed to the average number of neighbours in the radius selection method (150+) does not yield any clear benefits. The benefit of the radius selection method is therefore not solely due to the fact that it selects a high number of neighbours.

6.5.5 Conclusion

These further analysis show that the pre-training strategy does not work for all scenarios, as using 10 true matches to pre-train yields limited success for the very clumpy scenarios (e.g. “clump3-8”). However, an increased matching accuracy does improve the performance for some other scenarios (e.g. “noise2e-2”) and one could focus on improving that metric to improve the algorithm’s performance.

It also showed that lowering the number of surrogates to 3 instead of 10 decreases the success rate, sometimes quite significantly.

We also learnt that the two local features methods proposed (“B-spline” and “Angular Features”) behave quite differently when it comes to selecting the best k or radius parameter. This selection is done by choosing the parameter yielding the higher average matching accuracy over 10 randomly generated dataset. The “Angular Features” method shows a much clearer peak in matches accuracy for selected values of k (or radius) compared to the “B-spline” method.

Finally, we explored how many neighbours on average were selected when using the radius method, and the distribution of these numbers of neighbours for each points within a system. The best radius selected seems to “look” for a particular distribution which seems to maximise the success rate of the algorithm. This “winning” distribution notably has a high average number of neighbours compared to the total number of concepts.

Chapter 7

General Discussion

7.1 Summary of the approach

Human acquire knowledge using limited label information, making use of the natural statistics present in the world to better generalise, or acquire new concepts. The role of modality-specific information in the brain during concept acquisition and retrieval has also been recognized within neuroscientists. Inspired by these observations, this work looks at how one could align concepts from two similarly structured conceptual systems in an efficient way without the use of any labels, therefore using only the relationships of concepts within each system. Conceptual systems are represented here by points in a high-dimensional space, and their relationships by the euclidean distance separating them.

An algorithm inspired by CycleGAN was first used to try and align the two conceptual systems, optimized through a cycle and a distribution loss. This approach proved sensitive to the initialization strategy, with naive initialization yielding results that get stuck in local minimums. We proposed to solve this issue by first assigning a local feature to each point, which is determined based on its relationships to its k nearest neighbours. Using these, it is then possible to estimate a distance measure between points from different conceptual systems. A subset of surrogate corresponding points is estimated based on these distances, then serving as a labelled dataset to pre-train the model. This approach is borrowed from the Manifold Alignment literature. Two different local features were used, the first using a B-spline curve fitting over the distances between a point and its neighbours, the second building an 8-dimensional histogram feature vector from the angular relationship of a point and its neighbours. For the second method, two different ways of selecting the nearest neighbours were explored: using a fixed number for each point or using a radius.

These pre-training methods showed a significant improvement over the baseline algorithm, decreasing dramatically the number of times the algorithm gets stuck in local optimums. It proved particularly effective for dimensions above two and up to ten, where the baseline algorithm was performing poorly. Out of the two local features explored, using the angular method yielded better results, and the one using a fixed radius even more so. These methods are however quite sensitive to noise, even if overall, they still increase success for all level of noises.

In the remaining of this section, we discuss some future works, first exploring how we could improve the methods explored on a similar synthetic dataset, and then discussing how we can adapt and scale this work to the "full" problem that we aim to work with, i.e. when using the GloVe embeddings data as described in Roads and Love [1] and in Section 3.3.

7.2 Improvement of the explored methods

As seen in Section 6.5.1, one could focus on improving matches accuracy within the surrogates to increase the performances of the methods. This has limitations as we have seen that even using 10 true matches do not lead to optimal results in some specific scenarios (such as very clumpy ones).

Another approach would be to build a loss term based on the local structure features constructed, and integrate it to the main algorithm to be optimised at the same time as the cycle loss and distribution loss. This would allow to have a one-step algorithm (i.e. without pre-training) that could potentially yields better performance than the baseline, but starting from any randomly initialised models.

7.3 Back to the full problem

Although the current method can be improved or extended as per the two latter points, it is important to put this thesis's work back in context and understand how this work can be adapted to the full problem. Recall that the full problem uses data with a more complex and unknown structure, with thousands and thousands of concepts and embedding dimensions in the hundreds.

Computational considerations

The methods used in this work are significantly more efficient than the naive alignment algorithm, which considers all permutations and is in $O(N!)$. Indeed finding the k nearest neighbours for a single point in a D-dimensional space containing N points can be done with a computational cost of $O(ND \log k)$. If this needs to be done for each of the N points, this cost becomes $O(N^2 D \log k)$. This scales linearly in D, but is quadratic in N. Any further steps processing the neighbours into local features does not add any more complexity in D or N (but makes it linear in K), hence the pretraining step is linear in D and quadratic in N.

The fact that it is quadratic in N could make the methods presented here not efficient enough when considering a very high number of concepts. It is possible of course to try and reduce the dimensions of the embeddings by projecting them on a shared latent space of lower dimension, as is done in the manifold alignment literature. This also allows to deal with systems of different dimensions. However reducing dimensions to "acceptable" levels could be challenging, plus this does not address the scaling up in terms of numbers of concepts, which is the most problematic.

The baseline algorithm (i.e. without the pre-training step) is however quite computationally efficient, and this works gives a precious indication on how it can be used with local structure information to maximise success.

Occlusion and outliers

In the full problem, there could also be "missing points" or "outliers". Both terms are borrowed from the point set registration literature. Missing points occur when there is occlusion on part of an image and feature extraction does not capture the points. Outliers designate points that have no correspondence in the other domain (for instance because of different viewpoints). This has a double effect in the context of our work: first it is not always possible to match a point with another from the other domain, second missing points causes “deformation” in the local structure. However, Fan et al showed ways of integrating outliers in their work for a similar problem [40].

Other ways of integrating local structure

The methods show that including some additional structure information helps. Our experiments' results suggested that the baseline algorithm already captures some of the structure information “naturally” as slightly clumpy systems show better performances (Figure 5.2). It could have been the case that adding some local structure information brought no or little benefits. It instead brings significant improvements and methods focusing on local structures could be further explored.

There are other ways of including structure information in the algorithm which could be investigate. In particular the Point Set Registration literature could be more closely looked at, and especially the popular Coherent Point Drift (CPD) method [49]. In effect, instead of comparing local structures from domains X and Y, this method learns a smooth transformation which retains local structure. This is often done through l2-regularization penalizing “sinusoidal” transformations. Another idea, also borrowed from the Computer Vision field is to try and use a ResNet architecture in conjunction with regularization of the transformations. Because ResNet learns a displacement from identity, using for instance l2 regularization on the networks weights with such an architecture penalizes transformations that diverge too much from identity, hence forcing the structure to remain coherent. Note that the biases should not be regularized as we still want the systems to be rotated or translated enough to be correctly mapped to other systems. Some early experiment were conducted in that direction, but were not conclusive so far. However, pursuing this idea further could be interesting.

Bibliography

[1] Roads BD, Love BC. Learning as the unsupervised alignment of conceptual systems. Nature Machine Intelligence. 2020 Jan 17:1-7.

[2] Senel LK, Utlu I, Yücesoy V, Koc A, Cukur T. Semantic structure and interpretability of word embeddings. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2018 May 24;26(10):1769-79.

[3] Margolis E, Laurence S, 'Chapter 8 - Concepts' (2003), in Stich SP, Warfield TA. The Blackwell guide to philosophy of mind. John Wiley & Sons; 2008 Apr 15.

[4] Laurence S, Margolis E. Concepts and cognitive science. Concepts: core readings. 1999 Jul;3:81.

[5] Margolis, Eric and Laurence, Stephen, "Concepts", The Stanford Encyclopedia of Philosophy (Summer 2019 Edition), Edward N. Zalta (ed.), URL = ¡https://plato.stanford.edu/archives/sum2019/entries/concepts/¿.

[6] Brigandt I. Conceptual role semantics, the theory theory, and conceptual change.2004.

[7] Fodor J, Lepore E. Holism. 1992.

[8] Carey S. The origin of concepts. Journal of Cognition and Development. 2000 Feb 1;1(1):37-41.

[9] Pinker, S. (2007). The stuff of thought: Language as a window into human nature. Viking.

[10] Fodor, J. A. (2003). Hume variations.

[11] Carruthers P. Precis of the architecture of the mind: massive modularity and the flexibility of thought. Mind Language. 2008 Jun;23(3):257-62.

[12] Millikan RG. On clear and confused ideas: An essay about substance concepts. Cambridge University Press; 2000 Jul 31.

[13] Dummett, M. (1993). Seas of Language, Oxford: Oxford University Press.

[14] Bennett, M. & Hacker, P. (2008). History of Cognitive Neuroscience, Oxford: Wiley-Blackwell.

[15] Peacocke, C. (1992). A Study of Concepts, Cambridge, MA: MIT Press.

[16] Gettier, E. (1963). "Is Justified True Belief Knowledge?" Analysis, 23: 121–3.

[17] Dancy, J. (1985). Introduction to Contemporary Epistemology. Cambridge, MA: Blackwell.

[18] Hampton, J. (2006). Concepts as Prototypes, in B.H. Ross (ed.), Psychology of Learning and Motivation (Volume 46), New York: Academic Press, pp. 79–113

[19] Carey, S. (1985). Conceptual Change in Childhood, Cambridge, MA: MIT Press.

[20] The Origin of Concepts, Oxford: Oxford University Press.

[21] Gopnik, A., Meltzoff, A. (1997). Words, Thoughts, and Theories, Cambridge, MA: MIT Press.

[22] Tulving E. Episodic and semantic memory. Organization of memory. 1972;1:381-403.

[23] Damasio AR. Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. Cognition. 1989 Nov 1;33(1-2):25-62.

[24] Binder JR, Desai RH. The neurobiology of semantic memory. Trends in cognitive sciences. 2011 Nov 1;15(11):527-36.

[25] Reilly J, Peelle JE, Garcia A, Crutch SJ. Linking somatic and symbolic representation in semantic memory: the dynamic multilevel reactivation framework. Psychonomic bulletin & review. 2016 Aug 1;23(4):1002-14.

[26] Searle, J.R., 1980. Minds, brains, and programs. & &, pp.417-457.

[27] Zwaan, R. A. (2008). Experiential traces and mental simulations in language comprehension.

[28] Davis R, Shrobe H, Szolovits P. What is a knowledge representation?. AI magazine. 1993 Mar 15;14(1):17-.

[29] Goldstone RL, Rogosky BJ. Using relations within conceptual systems to translate across conceptual systems. Cognition. 2002 Jul 1;84(3):295-320.

[30] Feng Y, Goldstone RL, Menkov V. ABSURDIST II: A Graph Matching Algorithm and its Application to Conceptual System Translation. InFLAIRS Conference 2004 (pp. 640-645).

[31] Engle J, Feng Y, Goldstone R. Mining Relatednsess Graphs for Data Integration. InProceedings of the Annual Meeting of the Cognitive Science Society 2012 (Vol. 34, No. 34).

[32] Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation. 2003 Jun 1;15(6):1373-96.

[33] Ham JH, Lee DD, Saul LK. Learning high dimensional correspondences from low dimensional manifolds.

[34] Ham J, Lee DD, Saul LK. Semisupervised alignment of manifolds. InAISTATS 2005 Jan 6 (Vol. 120, p. 27).

[35] Wang C, Mahadevan S. Manifold alignment using procrustes analysis. InProceedings of the 25th international conference on Machine learning 2008 Jul 5 (pp. 1120-1127).

[36] Wang C, Mahadevan S. Manifold alignment without correspondence. InTwenty-First International Joint Conference on Artificial Intelligence 2009 Jun 26.

[37] Wang C, Mahadevan S. Heterogeneous domain adaptation using manifold alignment. InTwenty-second international joint conference on artificial intelligence 2011 Jun 28.

[38] Pei Y, Huang F, Shi F, Zha H. Unsupervised image matching based on manifold alignment. IEEE transactions on pattern analysis and machine intelligence. 2011 Dec 6;34(8):1658-64.

[39] Fan K, Mian A, Liu W, Li L. Unsupervised iterative manifold alignment via local feature histograms. InIEEE Winter Conference on Applications of Computer Vision 2014 Mar 24 (pp. 572-579). IEEE.

[40] Fan K, Mian A, Liu W, Li L. Unsupervised manifold alignment using soft-assign technique. Machine Vision and Applications. 2016 Aug 1;27(6):929-42.

[41] Tuia D, Camps-Valls G. Kernel manifold alignment for domain adaptation. PloS one. 2016;11(2).

[42] Zhu H, Guo B, Zou K, Li Y, Yuen KV, Mihaylova L, Leung H. A review of point set registration: From pairwise registration to groupwise registration. Sensors. 2019 Jan;19(5):1191.

[43] Besl PJ, McKay ND. Method for registration of 3-D shapes. InSensor fusion IV: control paradigms and data structures 1992 Apr 30 (Vol. 1611, pp. 586-606). International Society for Optics and Photonics.

[44] Zhang Z. Iterative point matching for registration of free-form curves and surfaces. International journal of computer vision. 1994 Oct 1;13(2):119-52.

[45] Fitzgibbon AW. Robust registration of 2D and 3D point sets. Image and vision computing. 2003 Dec 1;21(13-14):1145-53.

[46] Rusinkiewicz S, Levoy M. Efficient variants of the ICP algorithm. InProceedings third international conference on 3-D digital imaging and modeling 2001 May 28 (pp. 145-152). IEEE.

[47] Gold S, Rangarajan A, Lu CP, Pappu S, Mjolsness E. New algorithms for 2D and 3D point matching: pose estimation and correspondence. Pattern recognition. 1998 Aug 1;31(8):1019-31.

[48] Chui H, Rangarajan A. A new algorithm for non-rigid point matching. InProceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) 2000 Jun 15 (Vol. 2, pp. 44-51). IEEE.

[49] Myronenko A, Song X. Point set registration: Coherent point drift. IEEE transactions on pattern analysis and machine intelligence. 2010 Mar 18;32(12):2262-75.

[50] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. InProceedings of the IEEE international conference on computer vision 2017 (pp. 2223-2232).

[51] Sendik O, Cohen-Or D, Lischinski D. CrossNet: Latent Cross-Consistency for Unpaired Image Translation. InThe IEEE Winter Conference on Applications of Computer Vision 2020 (pp. 3043-3051).

[52] Almahairi A, Rajeswar S, Sordoni A, Bachman P, Courville A. Augmented cyclegan: Learning many-to-many mappings from unpaired data. arXiv preprint arXiv:1802.10151. 2018 Feb 27.

[53] Cuturi M. Sinkhorn distances: Lightspeed computation of optimal transport. InAdvances in neural information processing systems 2013 (pp. 2292-2300).

[54] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[55] Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Duerig T, Ferrari V. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. arXiv preprint arXiv:1811.00982. 2018 Nov 2.

[56] Gemmeke JF, Ellis DP, Freedman D, Jansen A, Lawrence W, Moore RC, Plakal M, Ritter M. Audio set: An ontology and human-labeled dataset for audio events. In2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2017 Mar 5 (pp. 776-780). IEEE.

Appendix A: ”B-spline” method results

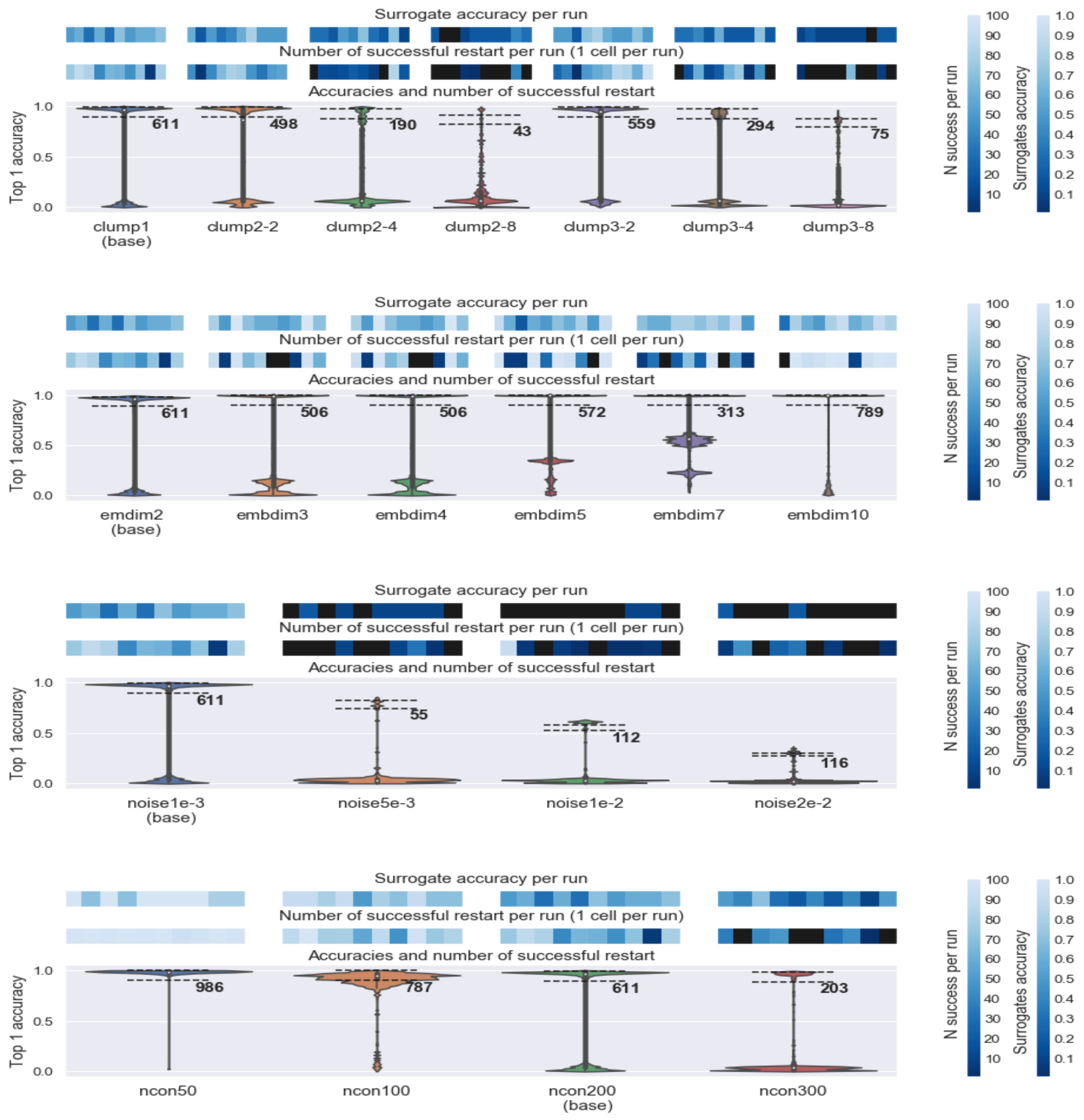


Figure A.1: B-spline - Distribution of restart accuracies and number of successes. **Reading instructions :** See Figure 5.2. **Notable results :** The method prefers no clumpiness and performances decrease drastically for increased levels. It performs well with dimension of up to 10, although many restarts seem to have an accuracy stuck “in-the-middle”. This is not the case for dimensions of 2 and 10. The method is not very resilient to noise, with successes dropping considerably. It also shows a trend of decreasing the number of successful restarts when increasing the number of concepts, with notably almost all restarts successful for 50 concepts (986 over 1,000).

Appendix B: "Angular Features - best k" method results

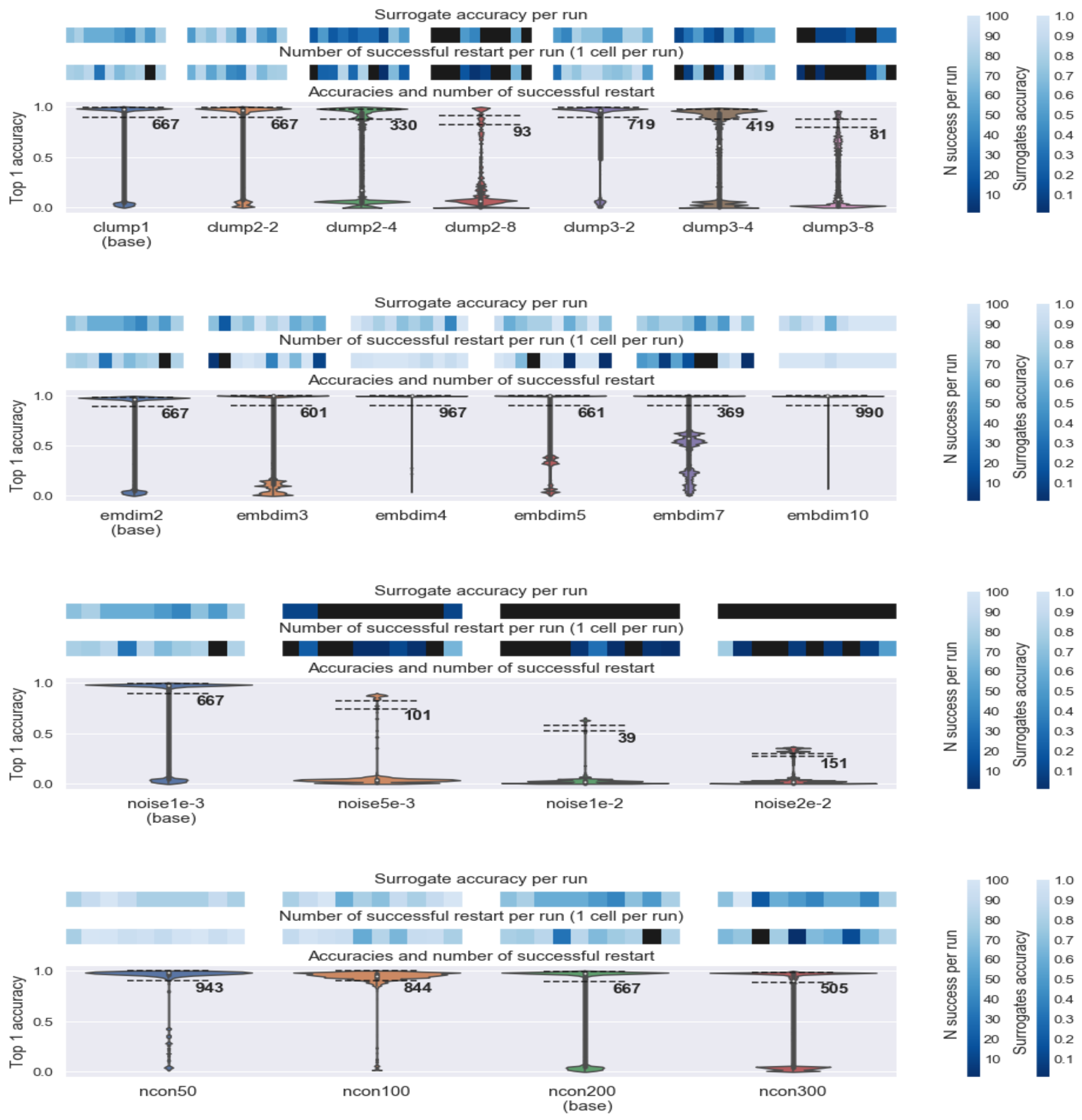


Figure B.1: "Angular Features - Best k" - Distribution of restart accuracies and number of successes. **Reading instructions :** See Figure 5.2. **Notable results :** Contrary to B-spline, this method seems to perform similarly or slightly better with a little clumpiness ("clump2-2" and "clump3-2"), but increasing clupminess further also yields bad results. Again, it performs well with dimension of up to 10 and again some scenarios show that some restarts seem to have an accuracy stuck "in-the-middle". A dimension of 10 has 990 succssful restarts over 1,000. Incresing noise once again decreases success. Finally, the effect of varying the number of concepts is less obvious, but still seems to suggest that less concepts is better.

Appendix C: "Angular Features - best radius" method results

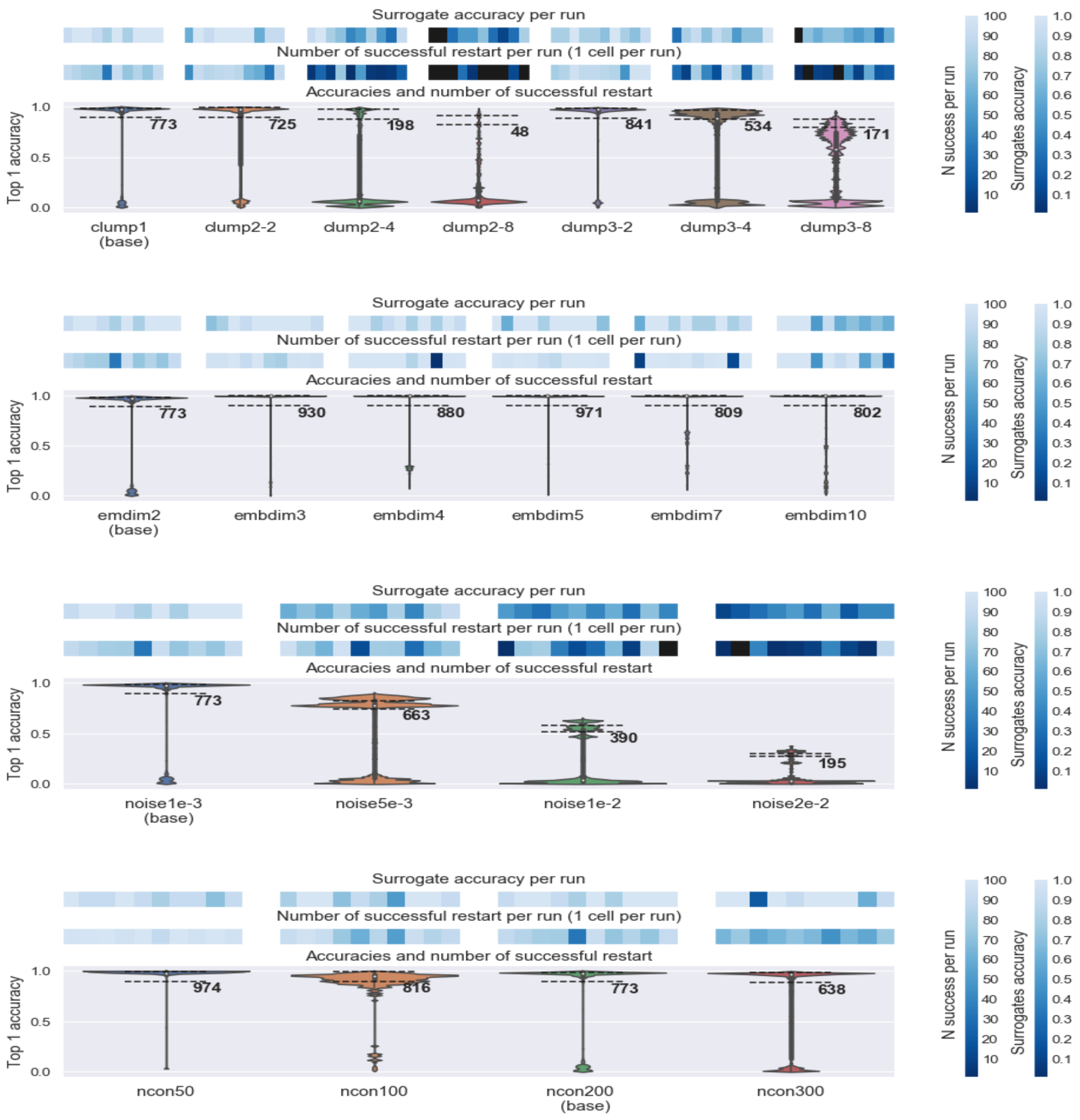


Figure C.1: "Angular Features - Best radius" - Distribution of restart accuracies and number of successes. **Reading instructions :** See Figure 5.2. **Notable results :** Again, increasing clumpiness too much decreases success. This method performs really well with all dimensions of up to 10. Adding noise decreases its performances, but it seems more resilient than the two other pre-training methods as it still performs well with little levels of noise. Again, a lower number of concepts helps performance.