Automated classification of behavioural and electrophysiological data in Neuroscience

Tiago Victor Gehring

Department of Computer Science
Faculty of Engineering

This dissertation is submitted for the degree of
Doctor of Philosophy

Supervisor: Prof. Eleni Vasilaki

May 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and are the result of my own work and research during the three years of my PhD. Most of the material presented has already been peerreviewed and published, or is about to be, in the form of journal papers with myself as first author. None of the contents has been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Tiago Victor Gehring

May 2018

Acknowledgements

First and foremost a very special thanks to my supervisor, Eleni Vasilaki, who was always suportive and patient, finding always time to give me some advice or feedback. Without her this work would not have been possible.

Special thanks also to Michele Giugliano and Daniel Wójcik for welcoming me for extended visits to their labs in Antwerp and Warwsaw and giving me the opportunity to have direct contact with great neuroscientists and other researchers. It was really a fantastic experience.

I would also like to thank my examiners, Richard Clayton and Thomas Novotny, for all their work and thorough feedback when reviewing this thesis. Thomas reviewed my work in great detail on two separate occasions and was kind enough to send me his handwritten notes and comments. This helped to improve the quality of the final version substantially.

Finally I would also like to thank my family and friends in Germany, Brazil, and England, and especially my girlfriend, Nicole Trethewey, for their patience and support while I was working on my dissertation.

Abstract

Due to technological advances the amount of data that can be collected in modern science is increasing every day and neuroscience is no exception. Integrating large amounts of data at different spatial and temporal scales is essential for uncovering the underlying mechanisms of the brain but poses also new challenges since drawing conclusions from vast amounts of data is increasingly difficult. New automated and fast analysis methods that can make sense of large and complex data sets are therefore in need and will become increasingly important in the years and decades ahead. This work proposes new tools for the analysis of two important types of data commonly found in neuroscience. The first is behavioural data from rodent navigation tasks in the form of animal movement paths. Two novel classification methods based on machine learning algorithms are proposed here. The methods are able to automatically or semi-automatically reduce the complex movement paths of the animals to a series of stereotypical types of behaviour, leading to both more detailed and consistent results. The second type of data considered here is electrophysiological data, in the form of extracellular multielectrode array (MEA) recordings which can record the electrical activity of thousands of neurons in parallel over long periods of time. Here a new highly-parallel data processing tool which can reduce the MEA data to a series of spike trains is presented. The tool can serve as basis for more sophisticated analyses like the reconstruction of the individual cell spike trains, for which machine learning methods are again essential. The results presented here show that machine learning algorithms and parallel processing architectures are both fundamental tools for coping with large and complex data sets, like the ones found in modern neuroscience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and motivation

Modern science has to deal with an ever increasing amount of data. Technological advances in materials, electronics and computers mean that we are increasingly able to collect data at multiple scales at higher resolutions and for longer periods of time. The "big data" revolution has arrived and, indeed, has been a reality for many fields of science for some time already. For example, in the field of particle physics, the Large Hadron Collider collected large amounts of data from particle collisions leading to the confirmation of the existence of the Higgs boson. Other examples include genetics, where full DNA sequencing of an organism is becoming commonplace; or astronomy, where new telescopes such as The Atacama Large Millimetre/sub-millimetre Array (ALMA) will generate hundreds of terabytes of data every year when in full operation.

The big data revolution is also slowly reaching the field of neuroscience (Sejnowski et al. 2014). Understanding how fundamental mechanisms in the brain, such as memory formation and learning, work remains one of the major challenges facing modern science. In order to support the development of new theories and models of the brain large amounts of data spanning multiple temporal and spatial scales have to be collected and integrated.

New instruments and experimental techniques are making it possible to collect vast amounts of data, but this brings also new challenges since drawing conclusions from the large data sets becomes increasingly difficult. For example, it is relatively straightforward to interpret patch clamp recordings (Sakmann and Neher 1984) that record the electrical activity of a single neuron over a period of minutes to a few hours; however, analysing and interpreting the results of the recorded activity of many thousands of neurons from multi-electrode extracellular recordings, that can collect data over a period of days or weeks, can be a significantly more daunting task.

In order to analyse and interpret the new large data sets that are becoming so prevalent nowadays new automated analysis methods have to be developed. Machine learning is a field of research concerned with the creation of algorithms that can extract information automatically from data sets. Machine learning algorithms are especially well suited for analysing large data sets and have been the focus of much attention in the past few decades. They are pervasive nowadays, finding applications not only in scientific research dealing with complex data sets, such as drug discovery and speech recognition, but also in diverse fields such as credit card fraud detection and behavioural advertisement.

Applications of machine learning algorithms in neuroscience include the automatic classification of neuron cell types (Armañanzas and Ascoli 2015), interpretation of fMRI (functional Magnetic Resonance Imaging) images with the objective of inferring the feelings or thoughts of a person (Pereira et al. 2009), and the reconstruction of neuronal firing patterns of individual neurons from recordings of an electrode surrounded by many neurons (a process known as spike-sorting, considered in Section 5.4.1). It is expected that as the complexity and amount of data that becomes available increases, the use of machine learning algorithms to cope with the data will become increasingly common (Akil et al. 2011).

The main objective of this thesis is to investigate how machine learning methods can be applied to other types of data sets commonly found in neuroscience. This thesis is divided into two parts; the first part will be devoted to developing new analysis methods for behavioural data. The second part of this thesis will focus on another fundamental type of experiments found in neuroscience: electrophysiological recordings.

Behavioural research, the topic of the first part of this thesis, is arguably one of the most fundamental areas in neuroscience but analysing and comparing the behaviour of animals can be a very laborious task. It is therefore fundamental to develop automated behavioural analysis methods which can consistently evaluate the behaviour of animals. Machine learning algorithms can be especially useful in achieving such goals (Sejnowski et al. 2014). For example, a study by Dankert et al. (2009) presented a method for automatic behavioural monitoring of Drosophila.

In this thesis, new classification methods for two rodent navigation tasks commonly used in neuroscience, the Morris Water Maze and the Place Avoidance Task, will be presented. The methods are based on the analysis of the movement paths of the animals within the arenas, which poses interesting challenges because of the complex nature and high variability of the data. Here, however, the problem is made more manageable by first splitting the movement paths into shorter segments. The segments are then classified into stereotypical classes of behaviour using cluster analysis, which makes it possible not only to classify the data but also to identify the common types of behaviour in a set of experiments.

In the field of electrophysiological recordings benefits can be gained from recent technological advances such as the ability to record extracellular activity from hundreds or thousands of locations and from many thousand cells at the same time. Multielectrode array recordings are crucial because they make it possible to study how neuronal ensembles work. Complex activity results from a collection of many cells and not a single neuron. Although machine learning methods can be used for tasks such as spike sorting, the raw electrical activity of the cells have to be first processed and transformed into action potential (spike) trains emitted by the cells. The amount of data generated from modern multielectrode arrays is substantial but also inherently parallel since the data recordings from different electrodes are independent from one another.

The second objective of this thesis will be to investigate how modern parallel hardware architectures can be utilised to create high performance analysis tools for electrophysiological data. The second part of the thesis will present a new highly parallel processing tool for handling multichannel recordings that exploits the parallel nature of both the data as well as of modern hardware architecture. The new tool can be used for processing large recordings even on modest computers and can serve as a basis for more sophisticated automated analyses, such as spike sorting, which can again be reduced to a classification problem, for which machine learning algorithms are especially well suited.

In order to be able to successfully make sense of large data sets both more sophisticated and also faster and highly scalable algorithms have to be developed. Both of these goals are addressed in this thesis, the first by showing that machine learning methods can be used to develop algorithms that are able to find patterns in data and to create automatic classifiers. The second goal is addressed by applying modern programming paradigms and techniques that can extract as much performance as possible from current hardware, which is becoming increasingly parallel in nature.

This thesis is built around two journal and one conference paper that I produced with the role of first author during my PhD:

I **Gehring T V, Luksys G, Sandi C, Vasilaki E (2015)** *Detailed classification of swimming paths in the Morris Water Maze: multiple strategies within one trial.* Scientific Reports, 5, 14562

II **Gehring T V, Vasilaki E, Giugliano M**. *Highly scalable parallel processing of extracellular recordings of Multielectrode Arrays.* 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2015.

III **Gehring T V, Węsierska M, Wójcik M, Vasilaki M** *Classification of behavioural patterns in the Active Place Avoidance Task* (in preparation)

The thesis is structured as follows. Chapter 2 presents a short literature review that introduce some neuroscience and machine learning concepts relevant to this work. Chapter 3 presents a new classification method for the swimming paths in the Morris Water Maze, a rodent navigational task commonly used in behavioural research. More specifically, in this chapter a semi-automated classification method for the trajectories of the animals based on a constrained (semi-supervised) clustering algorithm is introduced. This chapter is based on work that was published in a journal article (Gehring, Luksys, et al. 2015). Chapter 4 builds on the previous chapter by generalising the classification method to another important behavioural task in neuroscience, the Place Avoidance Task. The chapter also shows how the method developed in Chapter 3 can be transformed into a fully automated (unsupervised) classification and data extraction method. The work in this chapter is at the time of writing being prepared for submission to a journal. Chapter 5 changes the focus from behavioural to electrophysiological data and presents a new high performance data processing tool for Multielectrode Array recordings. This chapter is based on material that was presented at the IEEE EMBC Conference in 2015 (Gehring, Vasilaki, et al. 2015). General conclusions, limitations of the methods and results presented in this thesis, and suggestions for future improvements are presented in the last chapter.

Chapter 2

Literature review

This chapter introduces some concepts that will be relevant for this work. It is divided in two major sections: the first consists of a brief introduction to the field of Behavioural Neuroscience and discusses how animal experiments can be useful for understanding the brain. The second section will provide an overview of the field of Machine Learning and introduces some important concepts and algorithms that will be used in this work.

2.1 Behavioural Neuroscience

Behavioural Neuroscience is a field of research that studies how neural and physiological processes in the brain give rise to different types of behaviour. It tries to understand how functions in the brain are related to an animal's behaviour and the environment.

Important areas of study within Behavioural Neuroscience include:

- Learning and memory: how are memories in the brain formed and retrieved and do how diseases, such as dementia, affect these?
- Sensorimotor processing: studies how the brain processes sensory inputs from, for example, the visual, tacticle, and the auditory systems, allowing us to experience the world. Many people that have autism have difficulties integrating touch, smell, and sound sensory inputs. How are these inputs processed in the brain and which neural or physiological disorders lead to autism? This is just one of the questions that the study of the sensorimotor processing in the brain tries to answer.

2.1.1 Animal studies

In order to study how the brain processes lead to behaviour and how brain disorders can be prevented animal models are frequently used. Animals are used instead of humans for obvious ethical considerations, although the moral aspects of using animals is also highly debated (Bovenkerk and Kaldewaij 2014). Animal studies are nevertheless fundamental to advance our understanding of the brain. They have already led to major breakthroughs in our understanding of some of the cerebral mechanisms and to the development of drugs and treatments that substantially improves our health and quality of life (Miller 1985).

The human brain is much more evolved than the brain of every other animal, so it might at first not seem logical to use animals to understand the brain. Nevertheless, many fundamental brain processes are very similar in animals and humans and so the understanding that we gain from animal experiments can frequently be applied to humans.

Rats and mice are the predominant model organisms used in neuroscience, although other animals such as fruit flies and zebrafish are also used.

Different experimental procedures for animals designed to test different aspects of information processing in the brain have been proposed over time. This includes, for example, the development of various different tasks where animals have to escape from water (Blumberg et al. 2010). The water used in these experiments is usually uncomfortably hot and so animals want to leave it as quickly as possible; this motivates learning and memory processes (Wever 1932). Water maze tasks have been shown to be useful for testing spatial learning and recall. They can be used to, for example, test the effect of lesions to the brain, drugs or aging. One of the most widely used water tasks is the Morris Water Maze, which will be described in detail in Chapter 3.

Other behavioural tests of spatial memory not involving water have also been proposed in the literature. One of these is the Active Place Avoidance Task, where animals are allowed to move freely within an arena but have to learn to avoid a shock sector and to ignore local cues for navigation. Place Avoidance Tasks will be discussed in more detail in Chapter 4.

2.2 Machine Learning

Machine Learning is a branch of the artificial intelligence research field and one of the fastest growing areas of research in Computer Science (Shalev-Shwartz and

Ben-David 2014). Machine Learning has made major progress in the last couple of decades (Jordan and Mitchell 2015) and machine learning algorithms are now at the core of many modern technological developments such as autonomous cars and speech recognition systems, among many others.

In general terms, Machine Learning is concerned with the development of algorithms and systems that learn from experience (Flach 2012). How experience is defined and how it's acquired or delivered to the system will depend on the type of problem at hand. The field of Machine Learning is vast and consists of a multitude of methods and algorithms that can be applied to a wide range of problems.

One typical class of problems that machine learning methods are well suited to solve are classification problems. In a classification problem the objective is to assign objects into one of a group of categories based on a training set of data, or objects for which the category is known. The training set is also known as the *labelled data*. This is the data that is fed to the algorithms and which is then used to create a classifier, which can then be used to predict the category of future, unseen data.

A classic classification problem is for example the one of automatically identifying handwritten digits. This can be useful to automatically extract the postal code from letters in a mailing system, for example. In this case the categories (or classes) are the 10 possible digits (0-9) and the labelled data consists of pairs of sample images of handwritten digits and the value that they represent. The task at hand is then to use this data to create a classifier that can take a handwritten digit as input and output the digit that it represents making as little mistakes as possible. The ratio of incorrect to correct predictions is a measure of performance that can be used to compare different classifiers. Classification problems usually fall in a category of problems known as *supervised learning*, although reinforcement-learning and semi-supervised algorithms (considered below) also exist.

Another class of problems in Machine Learning make no use of labelled data but are rather concerned with the development of automated methods for finding patterns in data. The patterns that are found can then be used to predict probable future data outcomes (Murphy 2012). These are known as *unsupervised learning* problems.

Some algorithms do not fall neither in the unsupervised class nor in the supervised class. This is the case for example for *reinforcement learning* problems, which do not make use of labelled data, but are also not completely unsupervised since they receive feedback from signal that evaluates the actions taken by the algorithm or an agent. Algorithms for which both labelled and unlabelled data is used also exist. These are known as *semi-supervised learning* problems and will be especially relevant in this work.

The major types of machine learning algorithms are listed in Table 2.1 and described in more detail in the next few sessions.

Table 2.1: Classes of machine learning algorithms. For each class of algorithms the type of input data, its main applications and a some commonly used algorithms are listed. ANN: Artificial Neural Network based algorithm.

Type	**Input**	**Uses**	**Algorithms**
Supervised learning	Labelled data	Classification, regression	Decision tress, backpropagation algorithm (ANN), support vector machines
Unsupervised learning	Data	Model structure of data, finding patterns and correlations	Clustering, self-organized maps (ANN), adaptive resonance (ANN)
Reinforcement learning	Data + reward signal	Agents, learning from experience	REINFORCE
Semi-supervised learning	Data + some labelled data	Classification, finding patterns	Constrained clustering

2.2.1 Supervised learning

Supervised learning can be described as learning by example. In supervised tasks a learning algorithm is used to train a model by feeding it with pairs of values consisting of a data point, or an *instance*, and an associated *label*, or the category to which the data point belongs to. This collection of points is called the *training set* or *labelled data*. After training the learned model can then be used to predict the category of future data (the category is also known as the *class* of the data point). Models constructed in this way are known as *predictive models*.

Supervised learning can be subdivided into *classification* and *regression* problems. In classification problems the labels used for training are discrete and the output function is known as a *classifier*. If only two different classes exist the problem is known as binary classification, otherwise as a multi-class classification. In regression problems the labels are continuous and the trained function is called the *regression function*.

Classifiers can be constructed, for example, by a method known as decision tree learning (Breiman 1984; Rokach and Maimon 2008). Decision tree learning constructs classifiers by means of a *classification tree*. It can also be used to solve regression problems, by building a *regression tree*. Artificial Neural Networks (ANN) are another class of supervised algorithms that are commonly used in classification problems (C. M. Bishop 1995). They too can be used to construct (linear or non-linear) regression functions (Specht 1991).

2.2.1.1 Test data and cross-validation

The performance of supervised algorithms can be tested with a test data set distinct from the data that was used for training the model. Both sets have to be distinct in order to avoid *overfitting* the data. Overfitting occurs when the constructed model is too complex and tries to reproduce the training data to a high level of detail, without taking statistical fluctuations and noise into account. This leads to models that do not generalise to new data besides the one that was used for training the model.

In general both the training and test data sets are taken from the same set of labelled data. It is important, however, that the set of test data (usually much smaller) is not always the same but rather randomly selected for each model that is constructed. An even better approach is to construct not one but many models using different sets of test data. In a *10-fold* cross-validation, for example, the labelled data is randomly partitioned into 10 sets which are used in turn as the test data set; the data remaining (90%) is used for training the model. In the end the performance of the algorithm is evaluated by averaging the results for the 10 models that were constructed.

2.2.2 Unsupervised learning

In unsupervised learning no labelled data is available and the learning algorithm is simply fed with data. The aim of unsupervised algorithms is to build representations of the input data or find patterns and statistical correlations in the inputs that would otherwise be considered pure noise.

Self-Organized Maps (SOM) (Kohonen 1982) and Adaptive Resonance Theory (ART) (Grossberg 1987) are both examples of unsupervised methods based on artificial neural networks. Most clustering algorithms (described below) are also unsupervised and are frequently employed to group objects.

Since unsupervised algorithms make no use of test data the performance of the algorithms has to be defined differently than for supervised ones and will depend on the problem at hand and on the algorithm used.

2.2.3 Reinforcement learning

Closely related to unsupervised algorithms are *reinforcement learning* algorithms (Sutton and Barto 1998). Reinforcement learning deals with a different type of problems, however; it is concerned with learning what to do under different situations, or how to map situations to actions. Reinforcement algorithms have a reward signal that acts as an evaluative feedback: actions that lead to rewards are reinforced and other actions are weakened. The objective of reinforcement learning algorithms is to find the actions that maximize the total reward.

Reinforcement learning is believed to be the base mechanism by which biological organisms learn which actions to take in different situations (Holroyd and Coles 2002). One example of a reinforcement algorithm is the REINFORCE[*] algorithm (R. Williams 1992).

2.2.4 Semi-supervised learning

Semi-supervised algorithms (Chapelle et al. 2006; Zhu 2006; Zhu and Goldberg 2009) fall somewhere between unsupervised and supervised algorithms. Whereas in the supervised case only labelled data is used for training, the data is fully unlabelled in the unsupervised case. Semi-supervised algorithms make use of both labelled and unlabelled data to extend both unsupervised and supervised learning paradigms.

Unsupervised algorithms, such as data clustering (described in the next section), can be improved when a set of labelled data is also available. This data can then be used to constrain the algorithm. In the case of clustering algorithms this can be done by, for example, defining *must-link* and *cannot-link* constraints between two instances, requiring them to be in the same cluster if they have the same label or different clusters if the labels are different (Bilenko et al. 2004).

Supervised algorithms, such as classifiers, can also be improved when both labelled and unlabelled data are available. For classification problems the classifier can be trained with both labelled and unlabelled data; the objective is to construct a better classifier than one trained with the labelled data alone. This might sound counter-intuitive at first, but under the right conditions using both sets of data can in fact lead to better results. This is true, for example, when the instances of each class are concentrated in a limited region of space and not widely dispersed. In this case the set of unlabelled data, usually much larger than the labelled one,

[*]**RE**ward **I**ncrement = **N**on-negative **F**actor times **O**ffset **R**einfor**c**ement times **E**ligibility

can help to define the boundaries of each class and therefore improve the results (Liang et al. 2007).

An overview of the differences between supervised, unsupervised, and semi-supervised algorithms is shown graphically in Figure 2.1.

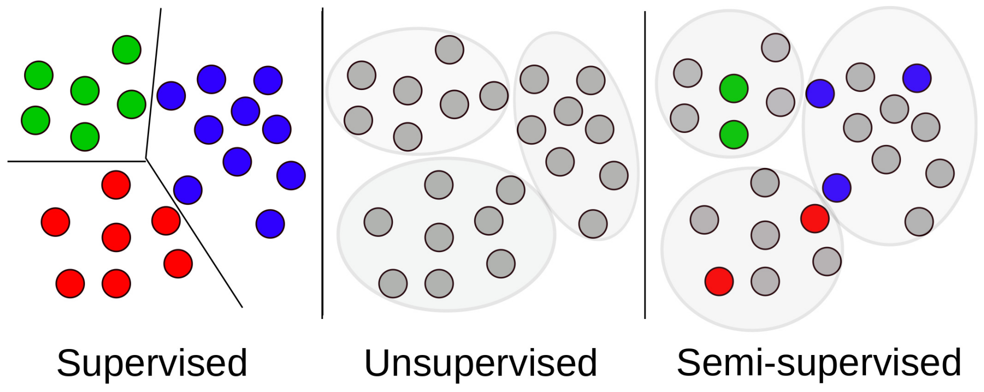


Figure 2.1: The three major classes of algorithms in machine learning. In supervised algorithms (left) the data is fully labelled and the objective is to find a partitioning of the space respecting the labels. Unsupervised algorithms (middle) on the other hand make no use of labels at all and try to find a partitioning solely on similarity or distance between elements. Finally, semi-supervised algorithms have only a partial set of labelled data which can be used to improve the results.

Semi-supervised algorithms can be further divided into *inductive* and *transductive* settings. In the former the objective is to predict labels of future data; in the latter the goal is to assign labels only to the unlabelled training set. In Chapter 3 a transductive semi-supervised algorithm based on a constrained clustering algorithm is used to classify a data set. The objective of the algorithm in this case is to reduce the classification effort and label only a subset of the data, and not to train a generic classifier that can be applied to new data sets.

2.3 Data clustering

Clustering algorithms (Aggarwal and Reddy 2013; Jain 2010; I. H. Witten et al. 2011) have as objective to partition a set of data points into groups, or *clusters*, so that *intraclass* elements, or elements within the same cluster, are as similar as possible; conversely, *interclass* elements of distinct clusters should be as dissimilar as possible (Bijuraj 2013). Although formulations of clustering algorithms which make use of labelled data exist (see below), clustering algorithms are generally unsupervised. They can be considered as a form of data summarisation and are commonly used in applications such as data mining (I. Witten and Frank 2005) and pattern recognition (C. Bishop 2006).

In order to successfully partition the data into meaningful clusters the notion of similarity between elements has to be defined and provided to the clustering algorithm. In many cases similarity functions based on a distance measure between points are used.

2.3.1 Clustering strategies

Different types of clustering algorithms exist, but the algorithms can be roughly divided into three broad classes: hierarchical and partitioning clustering, which cover the vast majority of available clustering algorithms, and density based algorithms. Many other more specialised clustering methods exist but will not be discussed here.

2.3.1.1 Density based clustering

In density based clustering methods the clusters are defined as contiguous areas of high density. Sparse regions between clusters containing few elements are usually considered noise and discarded. One example of a density based clustering algorithms is DBSCAN (Daszykowski and Walczak 2010). Closely related to density clustering are grid clustering algorithms, which divide the space into a hierarchy of discrete cells. One example is the STING algorithm (Wang and Muntz 1997), which approaches the results of DBSCAN in the limiting case of infinitely many cells.

2.3.1.2 Hierarchical clustering

Hierarchical clustering algorithms produce a hierarchy of clusters by following either a bottom-up (agglomerative) approach, where individual clusters which are found to be compatible are continuously merged at each level, or a top-down (divisive) one, where clusters are split at each step if appropriate. Hierarchical clustering makes usually use of a *similarity matrix* of $N\,x\,N$ elements, which can trivially be constructed from the similarity function, to decide if elements should be part of the same cluster or not. The resulting cluster hierarchy, which has the form of a dendrogram, can be cut at any level to give the resulting set of clusters. Example of hierarchical clustering algorithms are the classic SLINK (Sibson 1973) and Chameleon (Karypis et al. 1999) algorithms.

2.3.1.3 Partitioning methods

Contrary to hierarchical and density based methods, for which the number of final clusters is typically not set beforehand, partitioning methods take as input the target number of clusters and attempt to find the best partitioning that matches the given number of clusters. This is usually done by some sort of iteractive optimisation that relocates elements between clusters attempting to improve the solution. This can be done, for example, by taking a probabilistic view that assumes that the data comes from a mixture of probability distributions whose parameters have to be determined. The distribution parameters can be found, for example, by computing the Log-likelihood function and maximising it using the *expectation maximisation* method (Mitchell 1997).

Another approach to find a suitable partitioning of the data is to use a non-parametric algorithm and not assume any specific underlying data probability distributions but to instead define an objective function based solely on inter/intra-cluster relations, such as the distance or similarity/dissimilarity between elements. Since comparing all pairs of elements becomes quickly prohibitively expensive, typically only one cluster representative for each cluster is compared to the others. The cluster representative can be defined as one element in the cluster chosen to be the most appropriate one, which leads to the *k-medias* method (Kaufman and Rousseeuw 1987), or it can be defined as the centroid of the cluster; the latter case leads to the *k-means* algorithm (Section 2.3.3). The objective function of both k-medoids and k-means is constructed so as to sum all the distances between cluster representatives and other cluster members. This objective function is then iteractively minimised, leading to a solution.

2.3.2 Choosing a clustering method

Compared to density and partitioning methods hierarchical clustering algorithms have some advantages, such as being completely deterministic and returning a structure that is more informative than a flat set of clusters. However, they are also more complex having typically at least quadratic complexity on the number of elements (Manning et al. 2009). Moreover, hierarchical clustering algorithms usually don't revisit clusters after defining them, and so don't attempt to improve the results. Elements that are found to be linked (agglomerative algorithms) or which should be split (divisive approach) in one clustering step stay so until the end (Berkhin 2006; Kaufman and Rousseeuw 1990). Hierarchical clustering algorithms also make it difficult to incorporate constraints (Wagstaff 2000) and cannot therefore easily be extended to a semi-supervised paradigm (Section 2.3.4).

Density based clustering algorithms such as DBSCAN offer some advantages compared to the other methods, such as being able to find clusters of arbitrary shapes and being robust against outliers. However, they also have some major limitations, such as not being able to cluster data sets with large differences in densities and the difficulty of specifying a proper density threshold ϵ in many cases.

Partitioning based clustering methods are probably the most popular type of clustering algorithms (Ayram and Kainen 2006) and are preferred in applications such as pattern recognition (C. Bishop 2006) or data mining (I. H. Witten et al. 2011). They can also be easily extended to include constraints (Section 2.3.4) and were therefore the type of clustering algorithm that was used in this work. They also have some drawbacks, however, such as the problem of determining the 'right' number of target clusters (Section 2.3.6).

There is a myriad of cluster algorithms that were proposed in the literature (Estivill-Castro 2002). Here only two will be considered indetail: first the classic K-Means partitioning algorithm will be introduced, followed by a description of MPCK-means, an algorithm based on K-Means that also supports constrained clustering and which was used extensively in this work.

2.3.3 K-Means clustering

Despite having been proposed over 50 years ago[†] the K-Means clustering algorithm (Lloyd 1982; MacQueen 1967) is still widely used today. K-Means is a partitioning algorithm that uses the cluster centroids as cluster representatives. At first centroids are chosen at random; elements are then assigned to the nearest centroid based on a given distance measure. After this the new centroids are computed and the process is repeated until a convergence criterion is met (typically when less than 1% of the elements change cluster ownership). The process is described in Algorithm 1.

The K-Means objective function that is to be minimised is just a double sum, first over all the clusters then over the distance of the elements within the cluster to the centroid (Equation 2.1).

$$f_{K-Means} = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \mu_k\|^2 \tag{2.1}$$

[†]The original algorithm formulation by Lloyd from Bell Labs dates back to 1957 but was published only in 1982

```
Data: points $\mathbf{x}_i$, target number of clusters $\mathbf{K}$
Result: clusters $\{C_1, C_2, \dots, C_l\}$, $C_i = \{c_{i1}, c_{i2}, \dots, c_{i\,n_i}\}$
$1 \le c_{ij} \le N$; $c_{ij}$ = index of $jth$ element of $ith$ cluster
begin
    initialize centroids $\mu_i$;
    while convergence criterion not met do
        form clusters $C_i$ by assigning each point to the closest centroid;
        $\mu_i \leftarrow$ new centroids;
    end
end
```

Algorithm 1: K-Means clustering algorithm

In Equation 2.1 C_k, $k = 1..K$, are the clusters and μ_k the cluster centroids. Different distance functions can be used in K-Means with the Euclidean distance being the most common one. For the case of the Euclidean distance it is easy to show (Aggarwal and Reddy 2013) that the objective function in Equation 2.1 leads to the choice of using the centroid as cluster representative. This can be seen by taking the derivative of the objective function with respect to the cluster representative, μ_i:

$$\begin{aligned}\frac{\partial f_{K-Means}}{\partial \mu_j} &= \frac{\partial}{\partial \mu_j} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \mu_k)^2 \\ &= \sum_{\mathbf{x}_i \in C_j} 2(\mathbf{x}_i - \mu_j)\end{aligned}$$

Setting then the last expression to zero in order to find the minimum of the objective function:

$$\begin{aligned}\sum_{\mathbf{x}_i \in C_j} (\mathbf{x}_i - \mu_j) = 0 \quad &\Longrightarrow \quad N_j.\mu_j = \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \\ &\Longrightarrow \quad \mu_j = \frac{\sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i}{N_j}\end{aligned}$$

where N_j is the number of elements of cluster j. The last expression shows that the cluster representative is just the mean value of the coordinates of the elements in the cluster, or its centroid.

Not only the target number of clusters but also the choice for the initial centroids will impact the results of K-Means. This is because K-Means is a greedy algorithm, i.e., it chooses at each step the best local solution in the hope that this will lead to the global optimum. Since the position of the initial centroids is usually randomised the clustering results are usually non-deterministic. However, many other, non-randomised, initialization methods for K-Means have been proposed in the literature (Arthur and Vassilvitskii 2007; Bradley and Bradley 1998; Hartigan and Wong 1979).

2.3.4 Semi-supervised clustering

In its most basic form data clustering is completely unsupervised and, in many cases, like in the case of the K-Means algorithm, non-deterministic. Without any additional knowledge the clustering results will depend largely on how the similarity between elements is defined, and possibly some other parameters such as the target number of clusters. However, sometimes additional information about the data relationships or constraints is available and it is desirable to incorporate this knowledge into the clustering process. This additional information can be, for example, a partial set of labelled data, which tells which elements should belong to the same cluster and which not. Clustering algorithms which are able to make use of this knowledge are known as semi-supervised clustering (SSC) algorithms.

Although semi-supervised clustering algorithms have been far less studied than standard unsupervised ones, there has been a high interest in developing new semi-supervised algorithms recently (e.g. Anand et al. 2014; Xiong et al. 2014). This is because in fields such as data mining or pattern recognition it is often the case that a large data set has to be analysed but only a relatively small amount of labelled data is available or labelling is too time consuming. As was discussed in Section 2.2.4 some labelled data can be beneficial and improve the results, specially if the objective is to classify a large data set. A few different semi-supervised clustering algorithms have been proposed over the years, many based on the basic K-Means algorithm attempting to use the labelled data to overcome some of its limitations.

Two algorithms based on the standard K-Means algorithm but which are able to use a partial set of labelled data to improve the results were proposed by Basu et al., (2002). Their first algorithm uses the available partial set of labels to define the initial cluster centroids, which are taken to be the labelled points. This effectively turns the algorithm into a deterministic one. However, it also imposes some constraints on the labelled data itself because each cluster has to contain at least one labelled data point. The second algorithm proposed by Basu et al. uses the labels to not only fix the initial conditions, but also to constrain the clustering by moving labelled elements to other clusters in case that the labels of the clusters and data points don't match. As their experiments show, both algorithms improve the clustering performance significantly over standard K-Means, with the second proposed algorithm giving slightly better results than the first. However, the constraints used in this form are “hard” constraints that have to be fulfilled, which can be too restrictive at times. It is often desirable to have only a set of “soft” constraints that help to guide the clustering process but don't have to necessarily be all enforced at the same time.

A better and most common choice for incorporating constraints into clustering algorithms is to define pairwise ”must-link” (positive) and ”cannot-link” (negative) constraints (Basu, Bilenko, et al. 2004; Law et al. 2005; Wagstaff 2000), also known as *equivalence constraints* (Shental et al. 2004). These are shown graphically in Figure 2.2. A must-link constraint between two elements states that they are of the same class and should belong to the same cluster, or to clusters that are mapped to objects of the same class in the case that multiple clusters per class are allowed. A cannot-link constraint states that two elements are of different classes and should not belong to the same cluster or to clusters of the same class. The constraints defined in this form can be hard (Wagstaff 2000; Wagstaff et al. 2001) or soft, allowing them to be violated; this can be accomplished by, for example, using a modified objective function that includes the constraints as additional costs (Basu, Bilenko, et al. 2004; Law et al. 2005).

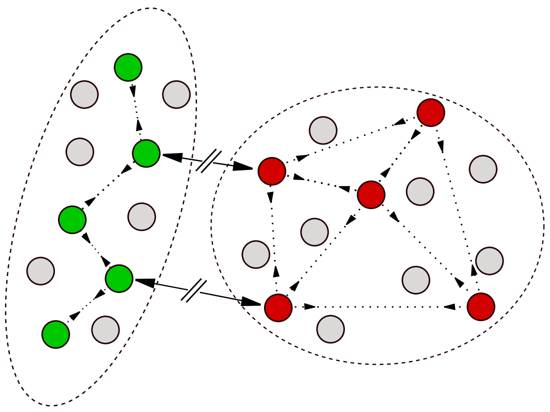


Figure 2.2: Equivalence constraint in semi-supervised clustering. *Must-link* constraints are shown as dotted lines and make elements of the same class end up in the same cluster. *Cannot-link* constraints (solid broken lines) push elements apart forcing them to end up in different clusters.

2.3.5 The MPCK-means algorithm

MPCK-means (Metric Pairwise Constrained K-Means) is another clustering algorithm based on K-Means that was proposed by Bilenko et al. in 2004 and which was used extensively in this work. MPCK-means supports constrained clustering in the form of soft must-link / cannot-link constraints that are incorporated into a modified objective function:

$$f_{MPCK-means} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|\mathbf{x}_i - \mu_k\|^2 + \sum_{\substack{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M} \\ c_i \neq c_j}} w_{ij} + \sum_{\substack{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C} \\ c_i = c_j}} \overline{w}_{ij} \tag{2.2}$$

where, as before, $x_{1..N}$ are the elements to be clustered, $C_{1..K}$ the clusters and $\mu_{1..K}$ the centroids of the clusters; $\mathcal{M}$ and $\mathcal{C}$ are the sets of must-link and cannot-link pairs of elements and c_i is the cluster number of the ith element.

Equation 2.2 is just the basic K-Means objective function (Equation 2.1) with the addition of two cost terms for violating the pairwise constraints: one for the penalty for violating must-link constraints (all pairs $(\mathbf{x}_i,\, \mathbf{x}_j) \in \mathcal{M}$ where $c_i \neq c_j$) and one term for the penalty for violating cannot-link ones ($(\mathbf{x}_i,\, \mathbf{x}_j) \in C$ where $c_i = c_j$); the penalty costs for violating the constraints are w_{ij} and $\overline{w}_{ij}$ respectively. The objective function is then minimised iteractively using the same basic algorithm as the one used by standard K-Means (Algorithm 1).

The MPCK-means algorithm uses the constraints not only for determining cluster ownership but also to adapt the distance metric; this is known as *metric learning* (Kulis 2013; Xiang et al. 2008; Yin et al. 2010). The original feature space might not offer a good separation between different clusters of elements, but metric learning can use the constraints to effectively warp the space so that elements from the same cluster seem to be closer and elements from different clusters seem farther apart. This is done by using a modified distance function so that the distances in each dimension can be rescaled (Xing et al. 2003):

$$\|\mathbf{x}_i - \mathbf{x}_j\|_A \coloneqq \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \,\mathbf{A}\, (\mathbf{x}_i - \mathbf{x}_j)} \tag{2.3}$$

where A is a symmetric positive-definite matrix that defines the scale factors in each direction. In MPCK-means this matrix is defined independently for each cluster, which gives rise to solutions involving clusters of multiple different shapes.

Including metric learning the objective function of MPCK-means becomes

$$
\begin{aligned}
f^*_{MPCK-means} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \left(\|\mathbf{x}_i - \mu_k\|^2_{\mathbf{A}_k} - \log(\det(\mathbf{A}_k)) \right) \\
+ \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ c_i \neq c_j}} w_{ij} + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ c_i = c_j}} \overline{w}_{ij}
\end{aligned}
\tag{2.4}
$$

with separate metric scaling matrices $\mathbf{A}_k$ for each cluster. The log-term comes from the normalisation of the k*th* cluster[‡].

One additional advantage of MPCK-means over standard K-means is that it can use the constraints not only to guide the clustering process and update the metric matrices, but also to initialize the centroids of the clusters. This turns MPCK-means into a deterministic algorithm, provided that enough constraints are available. In order to initialize the centroids, first the set of must-link constraints, $\mathcal{M}$, is expanded by taking the transitive closure of the set. The number of connected components of this set, λ, is then used to define λ neighborhood sets. The centre of these neighborhoods are then used to initialize the K centroids. If $\lambda > K$ then a weighted farthest-first traversal algorithm is used to select the first K neighborhoods (starting from the largest one). If $\lambda < K$ then the remaining clusters are initialized at random.

Algorithm 2 shows the general steps of the MPCK-means algorithm. Some details, such as how the metric matrices, $\mathbf{A}_k$, are updated are omitted for brevity. Please refer to Bilenko et al. (2004) for more details.

2.3.6 Choosing the target number of clusters

One of the difficulties with using clustering algorithms such as K-Means is determining the ideal number of target clusters (Everitt 1979; Milligan and Cooper 1985). Many methods for estimating the correct number of clusters, K, have been proposed over the years. These include, for example, using the a metric called the *gap statistic* (R. Tibshirani et al. 2001), which compares within-clusters dispersions with a null distribution, or the *Bayesan Information Criterion* (BIC) (Chen and Gopalakrishnan 1998). In many cases, however, custom measures for the problem at hand have to be developed and the results for different target number of clus-

[‡]more precisely, K-means and MPCK-means can be viewed as Expectation-maximisation algorithms on a mixture of Gaussian (Basu, Banerjee, et al. 2002) with identity and $\mathbf{A}_k^{-1}$ covariance matrices, respectively. The term in the equation arrives due to normalisation constant of the k*th* Gaussian (Bilmes 1997)

```
Data: points $\mathbf{x}_i$, target number of clusters $\mathbf{K}$,
      must-link constraints $\mathcal{M} = (\mathbf{x}_i, \mathbf{x}_j)$,
      cannot-link constraints $\mathcal{C} = (\mathbf{x}_i, \mathbf{x}_j)$
Result: clusters $\{C_1, C_2, \dots, C_l\}$, $C_i = \{c_{i1}, c_{i2}, \dots, c_{i\,n_i}\}$
$1 \leq c_{ij} \leq N$; $c_{ij}$ = index of $jth$ element of $ith$ cluster
begin
    create $\lambda$ neighborhoods from $\mathcal{M}$ and $\mathcal{C}$;
    if $\lambda \geq M$ then
        initialize centroids $\mu_i$ using weighted farthest-first traversal
    else
        initialize centroids $\mu_i$, $i \leq \lambda$, from constraints;
        initialize remaining centroids randomly;
    end
    while convergence criterion not met do
        Form clusters $C_i$ by assigning each point to the closest centroid:
        $C_i =$
        $\arg\min_{\mathrm{k}}(\|\mathbf{x}_i - \mu_k\|^2_{\mathbf{A}_k} - \log(\det(\mathbf{A}_k)) + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ c_i \neq c_j}} w_{ij} + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ c_i = c_j}} \overline{w}_{ij}$;
        $\mu_i \leftarrow$ new centroids;
        $\mathbf{A}_k \leftarrow$ new metric matrices;
    end
end
```

Algorithm 2: MPCK-means clustering algorithm

ters compared. This was the approach taken here for the classification methods presented in chapters 3 and 4 and which use the MPCK-means algorithm.

2.3.7 Dimensionality reduction

Technological advances in the available instruments and computers make it possible to collect an ever increasing amount of data. However, the large dimensional data sets commonly generated in modern experiments usually contain a significant amount of noise or information that is not relevant for the analysis objectives at hand. These extra data dimensions can significantly impact the results and performance of clustering and other machine learning algorithms. This is one of the forms of the problem known as "curse of dimensionality" (Aggarwal and Reddy 2013). In the case of clustering algorithms a high dimensional feature space can be problematic because common distance measures between elements tend then to be very similar in high dimensional spaces (Aggarwal, Hinneburg, et al. 2001), making it difficult to find meaningful clusters. For these reasons it is often necessary to reduce the dimensionality of a data set before performing cluster analysis.

Dimensionality reduction in cluster analysis can be done by selecting the most relevant data dimensions (or features), a process known as *feature selection*, or by projecting the existing dimensions onto a smaller feature space, generating a new set of features based on the old ones. The latter process is known as *feature extraction*.

2.3.7.1 Feature selection

Feature selection methods have as objective to remove redundant features and select only the ones that match some given relevant criteria (Guyon and Elisseeff 2003). This is often done by some sort of *variable ranking* (Stoppiglia et al. 2003), which assigns scores to variables and selects only those that rank better than a completely random variable.

2.3.7.2 Feature extraction

Another popular method for reducing the dimensionality of a data set is to define a new smaller subset of features based on the original ones. This can be done, for example, by Independent Component Analysis (ICA), which computes a new set of maximally independent variables based on the input data (Kwak et al. 2001). However, perhaps the most common and simplest feature extraction methods is

Principal Component Analysis (PCA) (Wold et al. 1987). PCA transforms N possibly correlated sets of features into N linearly uncorrelated variables, or principal components. The principal components are defined so that each successive component points to the direction that maximises the variance of the data. That is, the first principal component accounts for most of the variability of the data, followed by the second, and so on. Therefore, by selecting only the first few principal components and projecting the old feature values onto them, a new reduced set of features that accounts for most of the variability of the data can be defined.

Summary This chapter introduced briefly some machine learning concepts and relevant terminology and algorithms that are relevant to this work. In particular, an overview of cluster analysis and some of the available clustering algorithms, as well as some problems commonly encountered when clustering data, was presented. Clustering algorithms will be of major relevance to the next two chapters, where a novel classification method for two navigation tasks commonly used in neuroscience, the Morris Water Maze and the the Place Avoidance Task, will be presented.

Chapter 3

Detailed classification of trajectories in the Morris Water Maze

The Morris Water Maze (MWM) (Morris 1984; Morris 1981) navigation task is widely used in spatial learning studies (Bruin et al. 1994; D'Hooge and Deyn 2001; Varvel and Lichtman 2002). In this task, rodents are placed in a circular water pool with the goal of finding a submerged escape platform, which is made invisible by using, for instance, milky water or completely black walls in the experimental setup (Figure 3.1).

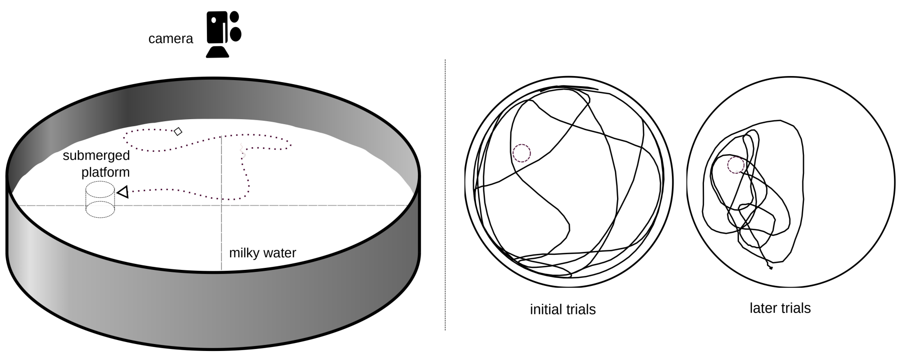


Figure 3.1: *Left:* schematic drawing of the Morris Water Maze. *Right:* example recorded trajectories for the initial and later trials. In the former case animals spend much more time close to the walls; in the latter they actively search for the platform.

In the MWM the animals have no internal reference points and have to therefore rely exclusively on external contextual visual cues for orientation and navigation (Sharma et al. 2010). It has been shown (Redish and Touretzky 1998) that the hippocampus plays a crucial role to navigate the MWM and that it is necessary for self-localisation and route replay.

MWM pool sizes vary greatly between experiments and depend on the types of animals used. For experiments with rats typical maze dimensions are a tank with a diameter of 1 to 2 meters, in which a $10-15$ cm platform is placed (Brandeis et al. 1989; Vorhees and M. Williams 2006). A typical experimental protocol consists of a set of trials, spread over a few days, during which the starting position of the animals is changed but the platform is kept at the same location. In a subsequent probe trial, the platform is removed and the animals' memory is evaluated by their swimming persistence in the surroundings of the previous platform location (D. P. Wolfer, Stagljar-Bozicevic, et al. 1998). This experimental protocol is known as *spatial acquisition* (D'Hooge and Deyn 2001; Vorhees and M. Williams 2006).

Typically, the initial trials in the MWM are characterised by the animals' tendency to spend most of the time next to the walls (a behaviour known as *thigmotaxis* D. P. Wolfer and H. P. Lipp 1992), or to perform random searches in the arena. In later training trials, however, animals show a gradual change in behaviour, characterised by progressive active searches for the platform. As they get familiar with the environment and testing rules, their ability of finding the platform from different starting positions is improved, as indicated by the reduced times to reach the platform.

Despite its widespread use in rodent behavioural studies, many studies quantify and compare the behaviour of animals in the MWM and base their results on just a handful of simple direct measures, without taking the different behavioural patterns into account. One of the most commonly used performance measures is the *escape latency*, defined as the time for the animal to find the platform and escape the maze. Other measures proposed in the literature include the swimming path length, which was suggested to be a better measure than escape latency (Lindner 1997; Lindner and Gribkoff 1991; Morris 1984).

It has long been recognised, however, that simple performance measures alone are not sufficient to quantify the wide range of different behaviours observed in MWM experiments (Dalm et al. 2000; Gallagher et al. 1993). Swimming paths with similar escape latencies, for example, can show very different types of behaviour (Figure 3.2). In order to be able to better characterise the behaviour in the MWM, more sophisticated quantification methods were therefore proposed over the years. Petrosini et al. (1998), for example, developed a scoring system of swimming paths based on measures such as the time that the animal spends in the "right" or "wrong" quadrants of the maze. Dalm et al. (2000), showed that the cumulative distance to the platform is correlated with the time spent next to the platform, but not with the escape latency. By combining different measures, they were able to create a classification method for the swimming paths, assigning each one of them to one of three different classes of behaviour. Their method, however, was applied only to a small group of animals (24) and attempted to classify only swimming paths from the first two trials, which are usually long and are typically characterised by common traits, such as a tendency for animals to move along the walls. In their work they did not consider the later stages of learning which show much more variability in the behaviour of animals and a wide distribution of swimming path lengths as animals learn how to find the platform increasingly faster.

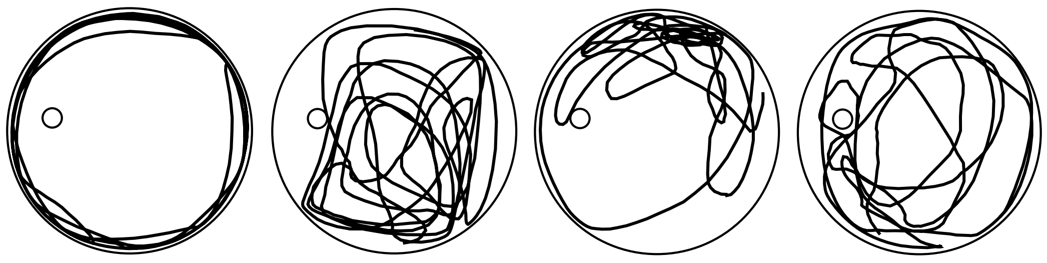


Figure 3.2: Some examples of long swimming paths in the Morris Water Maze. All have similar escape latencies but display completely different types of behaviour.

Categorisation methods for swimming paths in the MWM were also proposed in other studies. Wolfer and Lipp (2000), for example, associated various types of behaviour of the animals in the maze with different stages of learning. They computed more than a dozen measures for each swimming path and showed that these can be used to classify the paths into different behavioural classes (Figure 3.3), which they then associated with the different learning stages. However, they noted that their categorisation is valid only for large populations of animals and that single individuals might skip learning stages, or display more than one type of behaviour within a single trial.

Graziano et al. (2003) developed a more complete classification of swimming paths by dividing them into seven different classes, or *exploration strategies* in their terminology. The classes of behaviour ranged from thigmotaxis, where the animal almost never finds the platform (a type of behaviour mostly seen in early trials), to straight paths to the platform (*direct finding*). These classes showed a high correlation with the escape latency value but offered a more detailed quantification of the different stages of learning. By using the escape latency in addition to more than two dozen other path measures, they were able to create an automatic classifier that could assign swimming paths to one exploration strategy. However, similar to the previous categorisation attempts discussed above, their method can only map each swimming path to a single exploration strategy and is therefore not able to detect behavioural changes within individual trials.

A further categorisation approach for trajectories in the MWM was proposed by Stover et al. (2012). They extended the behavioural classes of Graziano and collaborators to 9, which they then divided into 4 strategy types. However, contrary to the work of Graziano et al., they didn't attempt to develop an automated classifier for the swimming paths. Illouz and collaborators proposed another classifier, which they named MUST-C, that is based on a hierarchy of Support Vector Machines (SVMs)*. SVMs are particularly well suited for dividing complex data sets into two regions and, by using a hierarchy of 8 SVMs, they were able to classify swimming paths into 9 different classes.

All the classification methods just mentioned attempted to classify only full swimming paths. They are therefore not well suited to quantify mixed types of behaviour within a single trial, which frequently characterise animals' strategies in the maze. Swimming paths, in particular longer ones, frequently show traits of more than one exploration strategy, making an unambiguous classification very difficult.

In this chapter a new, more granular classification method for swimming paths in the MWM is presented. In order to be able to quantify changes in behaviour within a trial, the classification is targeted not at complete swimming paths, but rather at shorter path segments. The method presented here consists of classifying multiple segments of a single swimming path into stereotypical classes of behaviour. As a result, swimming paths are not mapped to a single behavioural class, but to several classes. This makes it possible to detect subtle changes in behaviour between trials and among different groups of animals.

The development of this new method was motivated by the necessity of a more sophisticated quantification framework for analysing MWM data. The method was applied to a set of behavioural data where a strong manipulation (peripubertal stress) did not yield obvious learning performance differences compared to a control group when using more traditional quantification methods. A manual classification

*See for example Suykens and Vandewalle (Suykens and Vandewalle 1999) or Murphy (Murphy 2012) for an introduction to Support Vector Machines

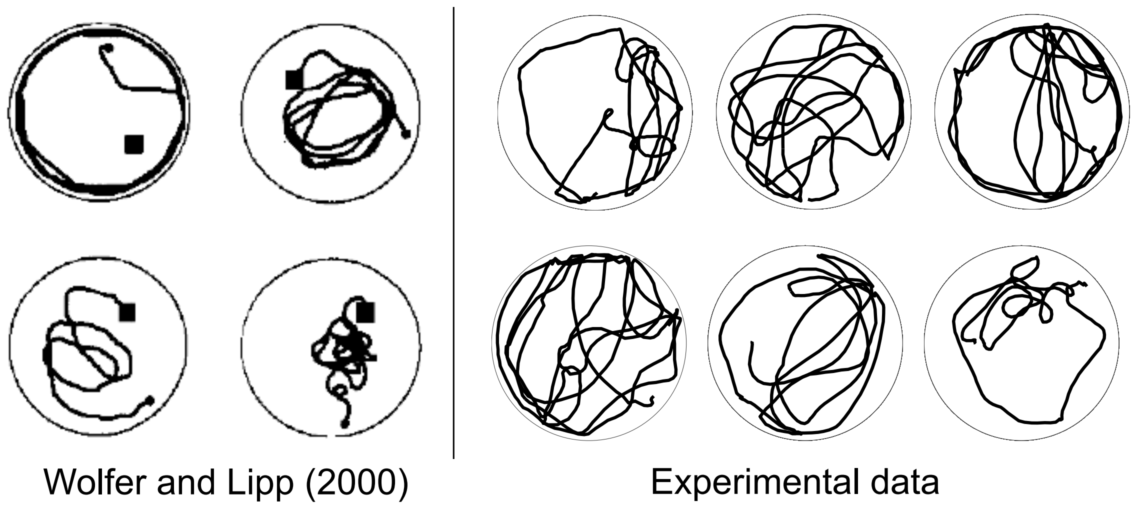


Figure 3.3: Examples of recorded trajectories in the Morris Water Maze. *Left:* Trajectories from Wolfer and Lipp (2000) showing stereotypical classes of behaviour which they associated with different stages of learning. *Right:* Sample trajectories from the Morris Water Maze recordings considered here (Section 3.1). As can be seen the recorded trajectories show multiple types of behaviours making it very difficult to unambiguously classify them.

of the full swimming paths also didn't produce satisfactory results because many of them showed mixed types of behaviour and the data set was not sufficiently large (around 300 paths per group). This led to large uncertainties in the classification making clear that a new approach was needed.

The segmentation of swimming paths shown here is performed automatically by custom analysis tools, described in Appendix A. The segmentation is done so that each segment has approximately the same length and overlaps substantially with the previous one. This overlap is important to make sure that the classification is not affected by an unfavourable segmentation, which could happen if a charateristical type of behaviour is 'cut' in the middle into two segments, for example. The large overlap between segments tries to overcome this by moving at small steps along the swimming path and looking at many possible segments, with only small variations between them, at the same time. However, this also means that a large number of segments (from a couple of dozens to a few hundred per swimming path) are generated. The large number of path segments (up to 30,000 for the data set considered here) makes a complete manual classification intractable for all practical purposes. In order to overcome this problem, a semi-automatic classification method is adopted. The classification method, based on a semi-supervised class of machine learning algorithms (Section 2.2.4), is able to automatically classify segments into behavioural classes based on a small percentage (between 5% and 12% of the segments for the data analysed here) of manually classified data. Because the classifier is based on a data clustering algorithm, it is also ideally suited for finding patterns in data, so that behaviour classes don't need to be known *a priori*.

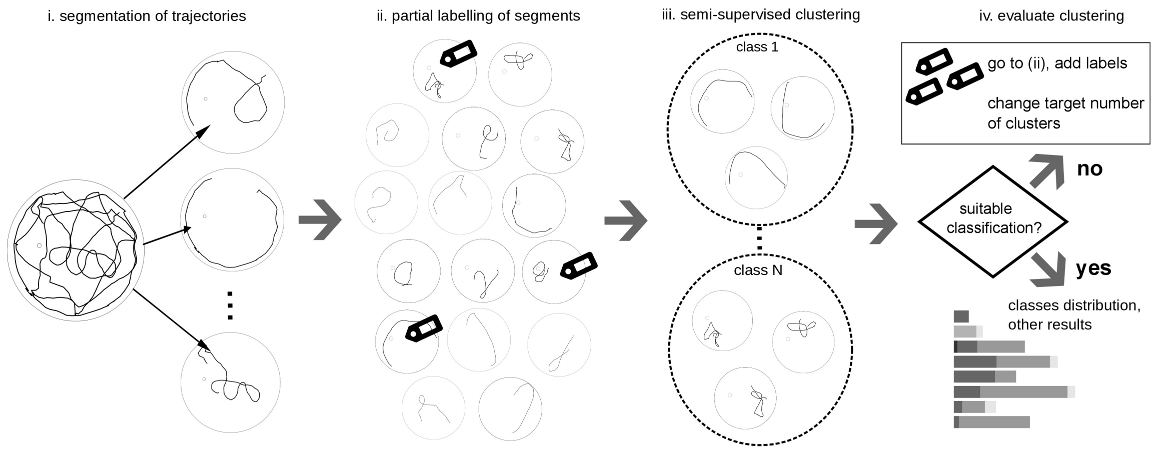

i. segmentation of trajectories
ii. partial labelling of segments
iii. semi-supervised clustering
iv. evaluate clustering
class 1
class N
go to (ii), add labels
change target number of clusters
no
suitable classification?
yes
classes distribution, other results

Figure 3.4: Diagram illustrating the swimming path classification method. Swimming paths are first segmented and then classified by means of a semi-supervised clustering algorithm.

The analysis method proposed here is shown schematically in Figure 3.4. In the next section first the experimental setup and then the data analysis method proposed here will be detailed. This is followed by a presentation of the results and finally a discussion about how the method compares to previous methods and how it can in principle be further improved.

3.1 Morris Water Maze Experiments

The test data set consisted of MWM trajectories of a group of 57 rats, 30 of which were subjected to peripubertal stress (Márquez et al. 2013; Veenit et al. 2013); the other 27 animals were the control group.

The objective of the experiments was to assess how stress and adverse experiences during youth can have an effect on behaviour in adulthood. Previous studies suggest that maltreatment during childhood is associated with a higher risk of developing violent behaviours later in life (Jonson-Reid et al. 2010; Su et al. 2010). In order to study how stress early in life impacts behaviour a group of rats was exposed to stress invoking situations at a peripubertal age (postnatal days P28 to P42) using the protocol described in Toledo et al. (2011). The protocol subjects animals regularly to fear-inducing situations, such as placing them on top of an elevated platform under bright light or submitting them to synthetic odours resembling the ones produced by predators found in nature.

Experiments were performed at the Laboratory of Behavioural Genetics, EPFL. All procedures were conducted in conformity with the Swiss National Institutional Guidelines on Animal Experimentation and approved by a license from the Swiss Cantonal Veterinary Office Committee for Animal Experimentation.

The water maze had a diameter of 2 m with a submerged platform 12 cm in diameter. Recordings were performed with the help of object tracking software, EthoVision (Noldus et al. 2001) version 3.1, and were done for a total of 12 trials for each animal. The trials were spread into 3 consecutive days with 4 trials each. Trials for each animal were applied consecutively so that the inter-trial interval for a given day was on the order of only a few minutes. The starting position of the animals was alternated between a few pre-defined locations over trials. Animals were allowed to swim for 90 seconds and were guided to the platform if they failed to find it during this time interval.

3.2 Data analysis

The main characteristic of the classification method of MWM trajectories described here is that trajectories are first divided into shorter segments; these segments are then classified into a discrete set of classes of behaviour. This is in stark contrast with previous classification methods that attempted to classify full swimming paths directly.

Because the segmentation of swimming paths leads to a large number of segments to be classified, labelling all of them manually becomes intractable. To deal with this problem, a semi-supervised learning algorithm (Section 2.2.4) was used here for the classification of the segments. Such an algorithm requires only a small set of manually labelled data in order to constrain and validate the classification results. The classification method consists of the following steps (Figure 3.5):

1. Segmentation of trajectories to overlapping segments of fixed length;
2. Computation of feature values for each segment;
3. Labelling of stereotypical segments for each segment class of interest;
4. Clustering and mapping of clusters to classes using the labelled data;

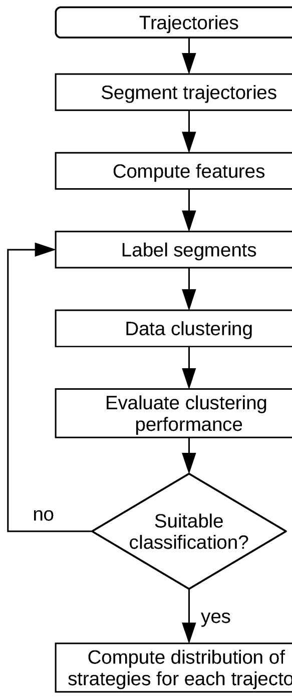


Figure 3.5: Steps of the classification procedure of swimming paths

5. Evaluation of clustering performance;
6. Steps 3-5 are repeated until an acceptable clustering quality is found;
7. Computation of the distribution of behavioural classes for each trajectory;

These steps will be detailed below, after introducing the behaviour classes that were adopted here.

3.3 Behavioural classes

A set of eight different classes of behaviour was chosen in the data analysis introduced here (Figure 3.6). The choice of behavioral classes was motivated by stereotypical behaviours that were observed during the classification process and focused on behavioural traits that may be crucial for the learning outcome. For this reason, a distinction was made between paths that pass nearby the platform but are not centred on it (scanning-surroundings) and closed paths that circle the platform and where the animal actively searches for it (target-scanning). A distinction was also made between paths concentrated almost exclusively on the periphery of the arena (thigmotaxis) and ones in which the animal starts moving inwards (incursion).

Self-orienting, a behaviour identified by Graziano et. al (2003), and chaining response (D. Wolfer and H. Lipp 2000), where the animal memorises the distance from the walls to the platform, are intermediate phases of spatial learning frequently observed in the experiments and were, therefore, assigned to individual classes. Finally, scanning and focused-search are both types of behaviour associated with random searches of other regions of the arena, mainly the centre in the former case. The difference between these two cases is that in focused-search the search is limited to a very small region, whereas in scanning larger areas are swept. Due to this distinction, they were split into two separate classes but are otherwise similar, in the sense that the search does not target the platform but rather other parts of the arena.

3.3.1 Segmentation of trajectories

The segmentation of trajectories is a method proposed to overcome the inherent difficulty of classifying very long swimming paths exhibiting many different types of behaviour. Classifying segments of trajectories is simpler because shorter paths usually don't show multiple types of behaviour. Also, due to the segments having approximately the same length, the variance of feature values is smaller, which also improves the classification results. The use of shorter segments and not the full trajectories for the classification therefore makes it possible to obtain a precise and detailed quantification of swimming paths. The disadvantages of segmenting the trajectories is an increased complexity of the classification method and the need for fully automated validation methods since the number of segments generated can be very large, making full manual proofing of the results intractable.

The segmentation adopted here consists of dividing a trajectory into N segments of length d (with small variations due to the discrete nature and spacing of available data points) which overlap significantly with previous segments. The overlap is necessary to reduce the classification variance due to unfavourable segmentations.

More formally, trajectories were split into segments of length d, where segment i is defined as the set of points of the recorded trajectory lying in the interval

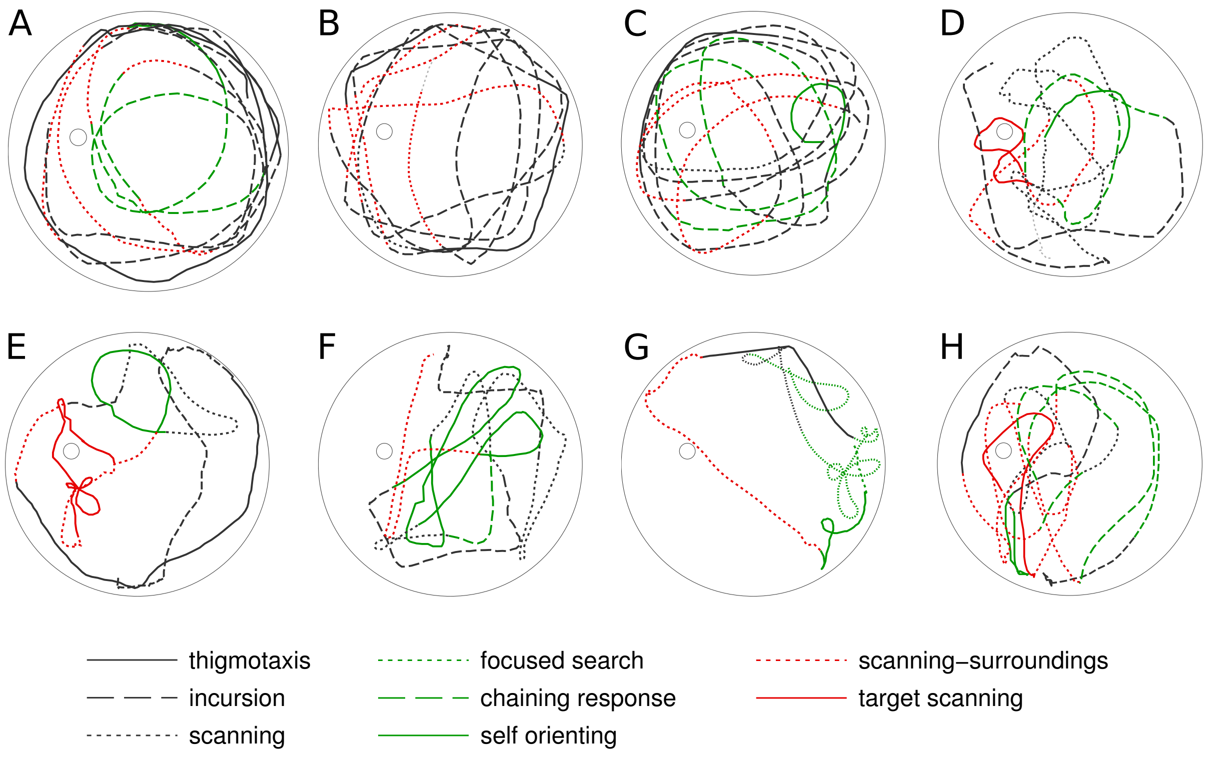


Figure 3.6: Examples of swimming paths showing different types of behaviour. Data: Laboratory of Behavioural Genetics, EPFL. Swimming paths were segmented and the generated segments were classified into a total of eight different types of behaviour, distinguishable by different line types/colours. Behavioural classes: **i)** *Thigmotaxis* (solid-black lines): Time is spent almost exclusively next to the walls; **ii)** *Incursion* (dashed-black lines): paths where animal still touches the walls but starts making incursions inwards; **iii)** *Scanning* (dotted-black lines): characterised by tighter paths that sweep a specific region of the arena; **iv)** *Focused search* (dotted-green lines): animal randomly searches a very small area of the arena; **v)** *Chaining response* (dashed-green lines): concentric paths where the animal memorises the distance from the walls to the platform (D. Wolfer and H. Lipp 2000); **vi)** *Self-orienting* (solid-green lines): Paths where the animal makes one full turn to orient himself (Graziano et al. 2003); **vii)** *Scanning-surroundings* (dotted-red lines): Open paths passing through a critical region around the platform; **viii)** *Target scanning* (solid-red lines): search is focused on regions next to or surrounding the platform;

$[l_i, l_i + d]$. For the analysis performed here overlaps of 70% and 90% between segments were adopted. The number of segments for a trajectory of length L, segment length d, and overlap α is $N = \lceil (L/d - 1)(1 - \alpha)^{-1} \rceil$, where $\lceil .. \rceil$ is the ceiling function. Trajectories shorter than the segment length are mapped to a single segment. The starting point of segment i, $1 \leq i \leq N$, is $l_i = d \cdot (1 - \alpha)(i - 1)$.

The choice of the appropriate segment length depends on the size of the arena and on the stereotypical behaviour types of interest. The classification performance is not affected by small variations in the segment length, provided that the features are only weakly correlated with the length. This allows for mixing of segments of slightly different lengths in one classification. It also follows from this that the number of segments, which is related to the segment length and overlap used in the classification, does not have a large influence on the final classification results.

For the data analysed here, recorded on an arena with a diameter of 2 meters, segment lengths of 250 cm and 300 cm and overlaps of 70% and 90% were adopted.

3.3.2 Computation of features

A total of eight segment features were selected for the classification procedure (Table 3.1). A schematic overview of the required dimensions for computing the features, as well as other definitions, is shown in Figure 3.7. The features adopted here quantify different geometrical properties of the segments and other characteristics, such as their relative proximity to the platform. Where possible, features were made independent from the choice of segment length. In this way small segment length variations don't have an appreciable impact on the classification results.

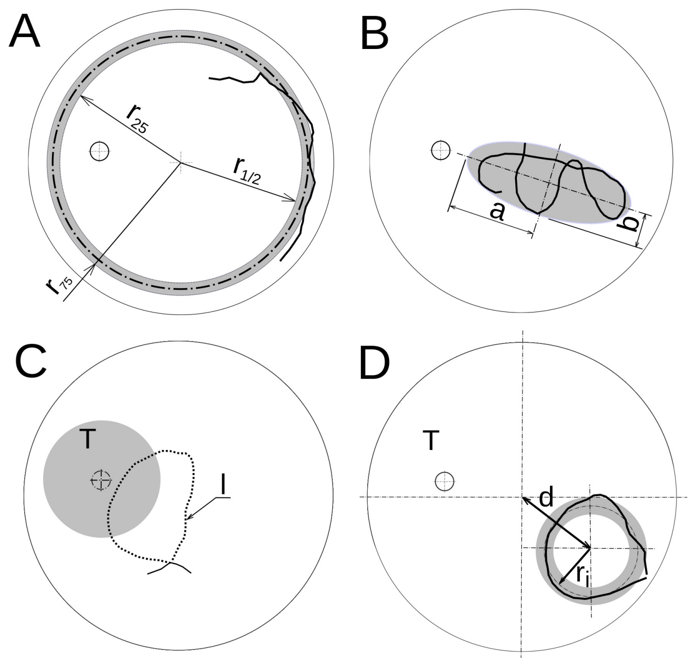


Figure 3.7: Definition of variables used for computing the measures, or feature values, for each swimming path segment. The features are an essential part of the clustering process since the feature values are used to estimate how similar the different segments are.

Table 3.1: Features used for the classification of swimming path segments. All features are dimensionless and only weakly correlated with the choice of segment length as to improve the classification robustness against segment length variations.

Feature	Definition
Median distance to centre	$r_{1/2} = median(\|\mathbf{r}\|)/R_{arena}$, where $\mathbf{r}$ is the set of vectors to the centre of the arena from each point of the segment
Interquartile range of the distance to centre	$(r_{75} - r_{25})/R_{arena}$ where r_{25} (r_{75}) is the first (third) quartile of the distance to the centre (Figure 3.7A)
Focus	$1 - 4\pi ab/l^2$, $l =$ segment length; a and b are the major and minor axes of the surrounding ellipse (Figure 3.7B)
Target proximity	Percentage of path lying within an area centred on the platform and with radius equal to 6 times the platform radius (Figure 3.7C area T_1)
Eccentricity	$\sqrt{1 - b^2/a^2}$ (Figure 3.7B)
Maximum loop length	Length of the longest self-intersecting loop in the trajectory divided by the segment length (see Figure 3.7C, loop "l")
Inner radius variation	$(r_{i,75} - r_{i,25})/r_i$, where r_i is the inner radius, or median distance to the centre of the surrounding ellipse (Figure 3.7D)
Central displacement	Distance from the centre of the surrounding ellipse to the centre of the platform divided by the arena radius (3.7D, "d")

Median distance to centre Calculated from the trajectory by computing the distance to the centre of the arena for each data sample and taking the median over the data. The median is used here instead of the (more commonly used) mean because it is a more robust statistic, in the sense that a few points that are very distant from the others, such as ones at the edges of the segments, will have a smaller effect on the final value.

The distance to the centre is a very useful measure to indicate if an animal spends most of its time on the periphery or moves also to more central regions of the arena. It can be seen as a generalisation of other measures adopted in previous studies, such as the time spent next to the walls or at the centre of the arena (Graziano et al. 2003).

Interquartile range of the distance to centre Besides the median of the radial distance, its spread can provide information about the type of behaviour involved. As a measure of spread, the interquartile range, or the distance between the first and last quartile of the data set (Figure 3.7A), was computed. This value is again more robust against outliers than the more commonly used standard deviation.

Focus The focus measures if and by how much the animal targets its search to specific parts of the arena. It is defined as $f \equiv 1 - 4\pi ab/l^2$, where a and b are the axis of the minimum enclosing ellipse around the trajectory (Figure 3.7B) and l is the segment length. For computing the enclosing ellipse the algorithm described in Moshtagh (2005) was used. With this definition increasing focus values are associated with increasingly closed paths.

Target proximity The proximity value measures the percentage of the path lying within a circle centred at the platform (Figure 3.7C) and with a radius of 6 times the platform radius.

Eccentricity The eccentricity measures how elongated the paths are; it also makes use of the values computed from the minimum enclosing ellipsoid. It is defined as $\epsilon \equiv \sqrt{1 - b^2/a^2}$, where a and b are semi-major and semi-minor axis of the enclosing ellipse (Figure 3.7B).

Maximum loop length This value measures the length of the longest loop, or self-intersecting sub-segment of the path. To compute this value all pairs of lines defined by two consecutive trajectory points were tested for intersection. If no intersection was present a value of zero was assigned to the feature. This choice leads to a discontinuity in the feature values but does not affect the clustering performance (in fact, it was shown empirically to improve the clustering performance since it divides the space into two regions, one of segments with loops and another one without).

Inner radius variation The inner radius (Figure 3.7D), r_i, is here defined as the median distance of every point in the path to the centre of the minimum enclosing ellipsoid. The coefficient of variation of the inner radius measures the relative dispersion of points relative to a circle. A perfect circle has a coefficient of variation equal to zero. It is here defined as $iqr\{r_i\}/r_i$, where $iqr\{r_i\}$ is the inter-quartile range of the inner radius.

Note: strictly speaking the coefficient of variation is defined in the literature as the standard deviation divided by the mean. Here, however, these values are replaced by the inter-quartile range and median to increase the robustness and stability against outliers.

Central displacement The central displacement is the Euclidean distance of the centre of the minimum enclosing ellipsoid to the centre of the arena (Figure 3.7D, "d"). It is an important measure to identify concentric paths with the arena.

3.3.3 Labelling of data and definition of constraints

The labelling of swimming path segments was done interactively, using a custom Graphical User Interface (GUI) written in Matlab (Appendix A). The graphical interface allowed to interactively browse through the segments, label them, and check the data clustering results. Multiple labels could be assigned to a single segment for cases in which characteristics of more than one behavioural class were found. Other labels, not used in the clustering process, could also be defined and assigned to segments or complete swimming paths. This was used to, for example, tag segments or trajectories of interest that would later be exported or analysed in more detail.

From the set of segments generated in the segmentation process (up to 30,000) between 5% and 12% were labelled (Table 3.4). The selection of segments to be labelled was done interactively from the complete set of segments. The custom GUI made it possible to sort segments according to different criteria, such as feature values or distance to the centre of the corresponding clusters. Various filters made it also possible to select only a subset of the segments, making it easier to identify segments that had to be labelled. One of those filters selected only isolated segments, or segments that were still not classified and did not overlap with any other successfully classified segments (see also discussion of the coverage value below). Segments that were isolated or lying on the boundaries of clusters were given priority in the labelling process.

Each pair of labels generated either a "cannot-link" (in case they differed) or "must-link" constraint (in case that they were the same). The number of constraints was therefore proportional to N^2, where N is the number of labelled segments. Because of the large number of constraints generated in this way (more than 2 million for the full set of labels), and the resulting computational performance impact on the clustering process, constraints were defined only between relatively close points. The Euclidean distance, d, between two labelled data points (the same distance function employed by the clustering algorithm) was calculated and a constraint was defined only if $d < 0.25$ (feature values were all re-scaled to lie in the $[0, 1]$ range). With this, the total number of constraints was reduced to less 10,000 and the time to run the clustering algorithm on a modern computer (AMD Ryzen 7 CPU running at 3.8GHz) was reduced from over 32 minutes to about 30 seconds without affecting the results.

3.3.4 Semi-supervised clustering

Semi-supervised learning methods (Section 2.2.4) are applicable to cases where large pools of data are available and labelling them all may not be possible. Whereas unsupervised algorithms search for structure in data without labels, supervised algorithms are provided with data and labels and try to infer a mapping between the two. Semi-supervised learning (SSL) can be considered an intermediate case between these two extremes. In one of the SSL formulations (Abu-Mostafa 1995), a pool of unlabelled data is provided together with an incomplete set of labels; the objective is again to find a suitable mapping between data and labels. A semi-supervised method was chosen here since the objective was precisely to be able to classify a large set of trajectory segments (between 8,000 and 30,000 for the data set analysed here) without having to label all of them manually, which would be very time consuming.

Among standard machine learning methods, a clustering algorithm was chosen (Section 2.3). The main reason for this choice was that clustering algorithms make it easy to detect new classes of behaviour because, in its broadest sense, they have as objective to group data points which are as similar as possible (Aggarwal and Reddy 2013). Therefore clustering algorithms can be used not only to classify data by finding clusters of similar segments, but also to identify new clusters with common types of behaviour. Hence, classes of behaviour don't have to be defined *a priori* but are rather discovered by the algorithm by splitting the space into groups of segments with common traits.

The semi-supervised clustering algorithm that was adopted here is known as MPCK-means (Metric Pairwise Constrained K-Means), which was described in Section 2.3.5. This is an algorithm inspired by the classical k-means algorithm, but which is able to incorporate previous knowledge to guide the clustering process.

Here the standard MPCK-means implementation[†] was used. The code is written in Java and is provided as part of the WekaUT library, a modified version of the popular Weka (Waikato Environment for Knowledge Analysis) machine learning library (Holmes et al. 1994). The Java code was integrated directly into Matlab and all algorithm options (such as the metric function and constraint weights) were left at their defaults.

As with most clustering algorithms, the MPCK-means algorithm also requires that the target number of target clusters is specified as an input. Determining the ideal number of clusters is a common problem in data clustering (this is discussed in more detail in Section 3.3.5). How this value was chosen here will be discussed below.

3.3.4.1 Two-stage clustering

The MPCK-means algorithm uses "must-link" and "cannot-link" pairwise constraints between elements to guide the clustering process (Section 2.3.5). Here, however, only cannot-link constraints were used in a first clustering stage. The reason for this is that the feature space is not easily separable and therefore multiple clusters can belong to one class. This means that a large proportion of the must-link constraints in Equation 2.2 cannot possibly be satisfied since two elements of the same class may well be ending on different clusters. It was found that adding must-link constraints at this stage had a negative impact on the clustering results. The same problem does not happen with cannot-link constraints since these must be satisfied even if there is not a one-to-one mapping between clusters and classes. However, "must-link" type of constraints were added in the second clustering stage; this second clustering step attempted to further divide larger clusters which either contained labels of more than one class or clusters which did not contain a sufficient number of labels. Intuitively the idea of the algorithm adopted here is to first find a rough partitioning of the data at large scales and then do further local partitionings as necessary. For the local splitting of clusters (second stage of the clustering) must-link constraints are relevant again since it is assumed that, at least locally, only one cluster will be mapped to a class of behaviour.

Experiments showed that the 2-stage clustering algorithm adopted here improves the clustering performance considerably (Figure 3.8). The algorithm is detailed in Algorithm 3. In the first step, the algorithm clusters the data into the target number of clusters using only "cannot-link" types of constraints. Clusters

[†]available at http://www.cs.utexas.edu/users/ml/risc/code/

are then mapped to label classes (how this mapping is done is discussed in the next section) and clusters which could not be unambiguously mapped to a class are sub-divided further by another clustering step. It this second clustering step each cluster that could not be mapped to a single class in the first step is split once more, this time, however, using both must-link and cannot-link constraints; multiple target number of clusters (from 2 up to two times the number of different classes in the original cluster) are also tried in succession. The first successful sub-partitioning, or the one with the smallest target number of clusters, is then chosen (a partitioning is considered successful if at least one of the sub-clusters could be mapped to a single class).

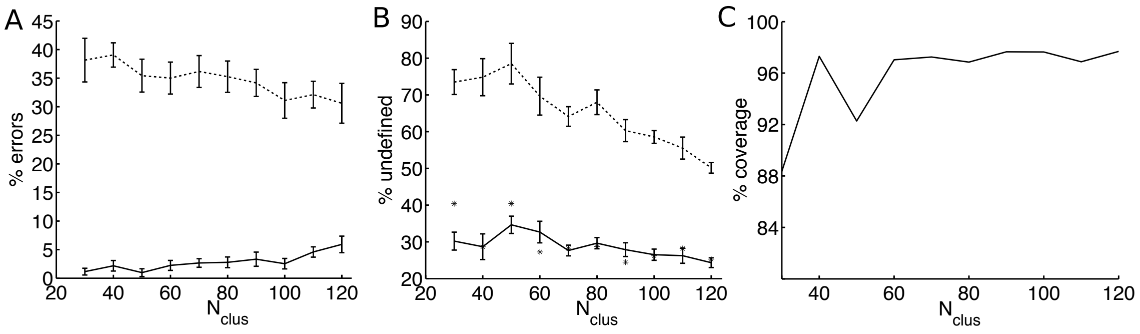


Figure 3.8: **A-B:** Impact of the number of clusters on the clustering performance for a set of 29,476 segments (Classification 3 in Table 3.4). Only the number of clusters in the first stage, N_{clus}, is shown. **A** Percentage of classification errors (percentage of labelled segments mapped to the wrong class); **B** Percentage of segments belonging to clusters that could not be mapped unambiguously to a single class; Error bars represent the 95% CI of a ten-fold cross validation over a set of 1,605 labels. *Continuous lines*: two-stage clustering. *Dashed lines*: single stage clustering. The results show that the second clustering stage leads both to significantly less classification errors and a smaller percentage of unclassified segments. Asterisks in the middle plot mark the results when using the full set of labels. **C** Percentage of the full swimming paths that are covered by at least one segment of a known class. The target number of clusters, N_{clus}, was chosen so that the coverage value is as high as possible while still having a low number of classification errors.

As Figure 3.8 shows, the two-stage clustering shows significantly better results than a single clustering stage. The two-stage algorithm also leads to less variance of the final results over different target number of clusters.

3.3.4.2 Mapping clusters to classes

Labelled data was used not only to guide the clustering but also to map clusters to classes and to estimate the quality of clustering (discussed in Section 3.3.5).

```
Data: feature vectors $\mathbf{x}_i$, $1 \le i \le N$
labels $\{L_1, L_2, ..., L_N\}$
target number of clusters $k$
maximum constraint distance $d_{max}$
Result: clusters $\{C_1, C_2, ... , C_l\}$, $C_i = \{c_{i1}, c_{i2}, ... , c_{i\,n_i}\}$
$1 \le c_{ij} \le N$; $c_{ij}$ = index of $jth$ element of $ith$ cluster
cluster labels $\{\mathcal{L}_1, \mathcal{L}_2, ... , \mathcal{L}_l\}$
/* initialization                                                    */
$\mathcal{M} \leftarrow \emptyset$, $\mathcal{C} \leftarrow \emptyset$;                     /* must/cannot-link constraints */
/* creation of constraints                                           */
foreach $\{i, j\} \,|\, L_i \ne \emptyset \wedge L_j \ne \emptyset \wedge |\mathbf{x}_i - \mathbf{x}_j| < d_{max}$ do
    if $L_i \cap L_j \ne \emptyset$ then
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{i, j\}$;                /* create must-link constraint */
    else
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{i, j\}$ ;               /* create cannot-link constraint */
    end
end
/* main algorithm                                                    */
cluster data into $k$ clusters $C_1, C_2, ..., C_k$ using constraints $\mathcal{C}$
foreach cluster $C_l$ do
    if $C_l = \emptyset$ then
        discard $C_l$ and move to next cluster
    end
    map cluster $C_l$ to label $\mathcal{L}_l$
    $m_l = |M_l|$, $M_l = (L_{l_1} \cap L_{l_2} \cap ... L_{l_n} | \{l_1, ..., l_n\} \in C_l)$;
    /* number of distinct labels in $C_l$ */
    if $m_l \ge 1 \wedge \mathcal{L}_l = \emptyset$ then
        $k'_l \leftarrow \max(m_l, 2)$ ;                         /* number of sub-clusters */
        while $k'_l < 2\,m_l$ do
            cluster $C_l$ into $k'_l$ sub-clusters $C'_{l1}, ... , C'_{l|k'_l}$ using constraints $\mathcal{C} \cup \mathcal{M}$
            map clusters $C'_{l1}, ...., C'_{l|k'_l}$ to classes $\mathcal{L}'_{l1}, ... , \mathcal{L}'_{l|k'_l}$
            if $\mathcal{L}'_{l|i} \ne \emptyset$ for any $1 \le i \le k'_l$ then
                $C_l \leftarrow \{C'_{l1}, ... , C'_{l|k'_l}\}$;            /* accept sub-clustering */
                $\mathcal{L}_l \leftarrow \{\mathcal{L}'_{l1}, ... , \mathcal{L}'_{l|k'_l}\}$
                exit while
            else
                $k'_l \leftarrow k'_l + 1$;                 /* increase number of sub-clusters */
            end
        end
    end
end
```

Algorithm 3: Two-stage clustering. Steps 1 and 2 define the set of constraints. The main clustering, using only "cannot-link" constraints, is done at step 3. Steps 4 describes how clusters that were not mapped to a class, because they contained multiple label classes or an insufficient number of labels of one class, are sub-divided. The sub-division process attempts to break down the cluster into an increasing number of smaller ones until the smaller clusters can be mapped to classes (in which case the larger cluster is replaced by the smaller ones - step 4h) or an upper limit of sub-clusters is reached.

Clusters were mapped to the class of the labelled segments within the cluster. Clusters containing labels of multiple classes, or clusters with less than the minimum amount of labels, m_i, were marked as *undefined;* segments belonging to these clusters were discarded from further analyses. Since the clustering algorithm generates clusters with a wide range of different sizes the minimum number of required labels of cluster i, m_i, was made dependent on the cluster size:

$$m_i \equiv \lceil n_i \, p_{min,\,i} \rceil \tag{3.1}$$

where n_i is the cluster size, and $p_{min,\,i}$ is defined as

$$p_{min,\,i} \equiv \max(n_i^{-\gamma},\, p_{min}) \tag{3.2}$$

Here $\gamma = 0.7$ and $p_{min} = 0.01$ were adopted.

With the definition above a larger proportion of labelled data is required in the case of smaller clusters; for larger clusters this proportion gets smaller but is always at least 1%.

3.3.5 Clustering validation, target number of clusters

The quality of the clustering was estimated by evaluating three values: i) the number of classification errors; ii) the percentage of unclassified segments (i.e., proportion of segments belonging to one of the undefined clusters, as described in the previous section) and iii) by the *coverage* value. The coverage value is defined as the percentage of the trajectory that is covered by segments of known classes. Because of the overlap between the segments of a trajectory this value is not the same as the percentage of segments that were successfully classified. The coverage value was chosen as the main criterion used to evaluate the classification quality and in each classification the target was to achieve a coverage of at least 90%. It was chosen as the main quality criterion because the final objective of the classification was to map as much of the trajectories to segments of a known class. Only secondary attention was given to the absolute percentage of classified segments which, contrary to the coverage value, does not take the overlap and distribution of classified segments into account. It can therefore have a large value even if the total coverage is small (this can happen, for example, if mostly redundant and overlapping segments are successfully classified).

In order to estimate the classification error, another important measure of the quality of the clustering, a 10-fold cross-validation was used (Section 2.2.1.1).

This consisted in leaving 10% of the labelled segments for testing and not using them to define the clustering constraints. The error was calculated by dividing the number of incorrectly classified segments by the number of correct ones and taking the mean over all 10 runs. The same two-way data split was also used to tune clustering parameters, such as the target number of clusters. Since the classification performed here was targeting only one data set, and the objective was never to apply it to new data or to create a generic classifier, a two-way split was favoured over a three-way one; in doing so more labelled data was left for the clustering itself.

The percentages of unknown segments and of classification errors are not fully independent. This is because rejecting more clusters will lead to less classification errors, but will also leave a larger proportion of segments unclassified. The trade-off between the two can be controlled in part by changing the minimum number of labels required to assign a cluster to a class. This is controlled by parameters γ and p_{min} (defined in *Mapping clusters to classes* above). Requiring more labels per cluster leads to fewer classification errors but requires more labelled data, because otherwise most clusters will not have enough labelled points and will therefore be left as undefined.

Figure 3.8 shows how the classification errors, percentage of unknown segments and coverage values vary as the target number of clusters, N_{clus}, is increased for one particular classification (continuous lines show the final results and dashed ones the results after the first clustering stage). The results shown are for Classification 3 in Table 3.4 and as can be seen in the figure the coverage value (right plot) remains practically constant and above 96% for $N_{clus} \geq 60$. The middle plot shows that the number of undefined segments does continue to slowly decrease with growing N_{clus}, but the new classified segments have little or no impact on the overall final results. However, the leftmost plot in Figure 3.8 shows that the classification error increases the more clusters are used but that it remains practically constant between $N_{clus} = 60$ and $N_{clus} = 80$. Running the classification for different target number of clusters within this range showed that values around $N_{clus} = 75$ provided the best overall results with a smaller number of undefined segments (even if this value is only of secondary importance), so this was the value chosen for this particular classification.

A further important indicator of the quality of a classification, which can also be used to detect issues such as mixing of classes, is the confusion matrix. It counts the number of classification errors between individual classes, so that the diagonal shows the number of correct classifications and values outside of the diagonal the classification errors. Table 3.2 shows the confusion matrix for the data analysed in Section 3.4.2 for a 10-fold cross-validation. As can be seen the total number of classification errors is relatively small in all cases, which shows that the classifier is performing adequately. However, the confusion matrix shows that most of the classification errors involve the Incursion (IC) class, specially the Incursion-Thigmotaxis and Incursion-Scanning Surroundings cases. It shows that separating the Incursion and the two other classes is more difficult because they show many similar traits, such as very open paths. This shows that the confusion matrix can be a valuable tool for identifying problematic combination of classes that might need additional features in order to be successfully separated from one another.

Table 3.2: Confusion matrix for the classification of the segments. Values are the total number of miss-classifications for a 10-fold cross validation of the clustering algorithm (i.e. 10 runs using 10% of the values each time for validation). Values in the diagonal show the number of correct classification for this class. The confusion matrix can be used to identify classes that are not well separated in the classification process (non-diagonal values different than zero)

	TT	IC	SC	FS	CR	SO	SS	TS
Thigmotaxis (TT)	267	14	0	0	0	0	2	0
Incursion (IC)	8	331	3	0	4	0	4	2
Scanning (SC)	0	3	144	0	0	0	2	0
Focused search (FS)	0	0	1	72	0	0	0	1
Chaining response (CR)	0	8	0	0	90	0	1	0
Self orienting (SO)	0	0	0	0	0	61	0	0
Scanning surroundings (SS)	2	11	0	0	3	0	252	0
Target scanning (TS)	0	5	0	0	0	0	2	73

3.3.6 Mapping segment classes to swimming paths

After classifying the swimming path segments, the evolution of the strategies along the swimming paths was computed. The mapping was done for discrete path intervals that depended on the segment overlap (Table 3.4) and took into consideration all the segments that overlap with the intervals. The corresponding segment class for an interval I_i was computed from the following expression:

$$\mathcal{C}_{I_i} \equiv \arg\max_{\mathrm{c_k}} \sum_{\substack{\mathrm{S_j} \in \mathrm{c_k} \\ I_i \cap S_j \neq \emptyset}} w_k \exp\left(-d_{ij}^2/2\sigma^2\right) \tag{3.3}$$

where S_j is a segment and d_{ij} is the distance from the centre of the jth path segment to centre of the interval, in minimum path interval units (Figure 3.9). The summation is done for all segments that overlap with the interval I_i and which belong to the class c_k; it is computed for all classes and the class that maximises the expression is then selected. The parameter σ, which controls how much segments influence the choice of segment classes over the swimming paths, was set to $\sigma = 4$. There is also a class weight factor, w_k, which is introduced to account for the fact that certain behavioural classes are transient and have only a few representative segments at a time whereas others are longer lasting and have many segments in sequence. It was defined as

$$w_k \equiv \frac{L_{max}}{L_{max,k}} \tag{3.4}$$

where $L_{max,k}$ is the longest trajectory segment of consecutive intervals of class k and L_{max} the longest homogeneous trajectory segment (that is, $L_{max} = \max\{L_{max,k}\}, k = 1...N_c$, where N_c is the number of classes).

The class weight in Equation 3.3 is inversely proportional to the maximum length of segments of the class so that transient classes, or classes which tend to be shorter, do not get overshadowed by more common ones. Figure 3.10 shows one example using unitary class weights ($w_k = 1$) and using the above definition. As can be seen in the former case the self-orienting and focused search segments do not get detected because of the larger influence of surrounding classes (incursion/thigmotaxis).

Table 3.3 shows the average and maximum lengths of consecutive intervals of each class using first $w_k = 1$ and then the definition above. An example of a trajectory classified with constant and variable weights is shown in Figure 3.10. Note that the values

3.3.7 Final classification results

The classification of the swimming paths was done for multiple times, for different segment lengths and overlaps. To produce the final results, the multiple classifications were then combined. In order to combine two classifications, the classes

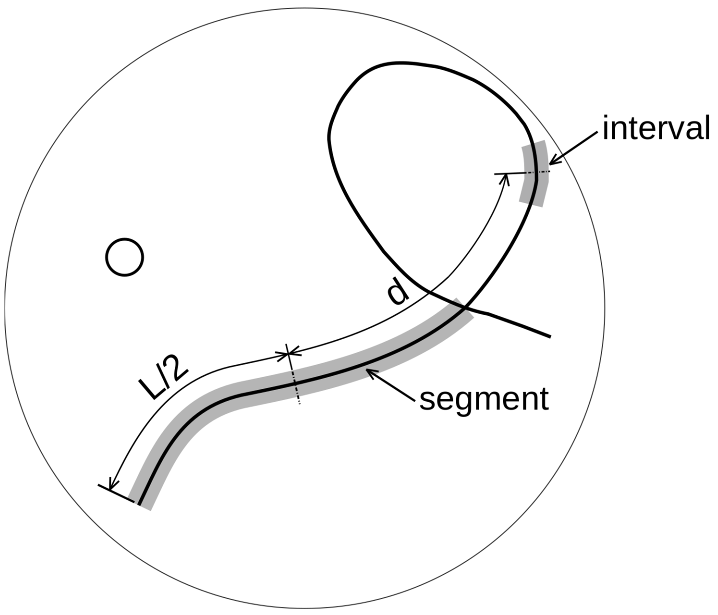


Figure 3.9: The distance, d, between a segment and an interval is computed by measuring the distance between their midpoints. Shown here schematically is one trajectory segments of length L and a small interval.

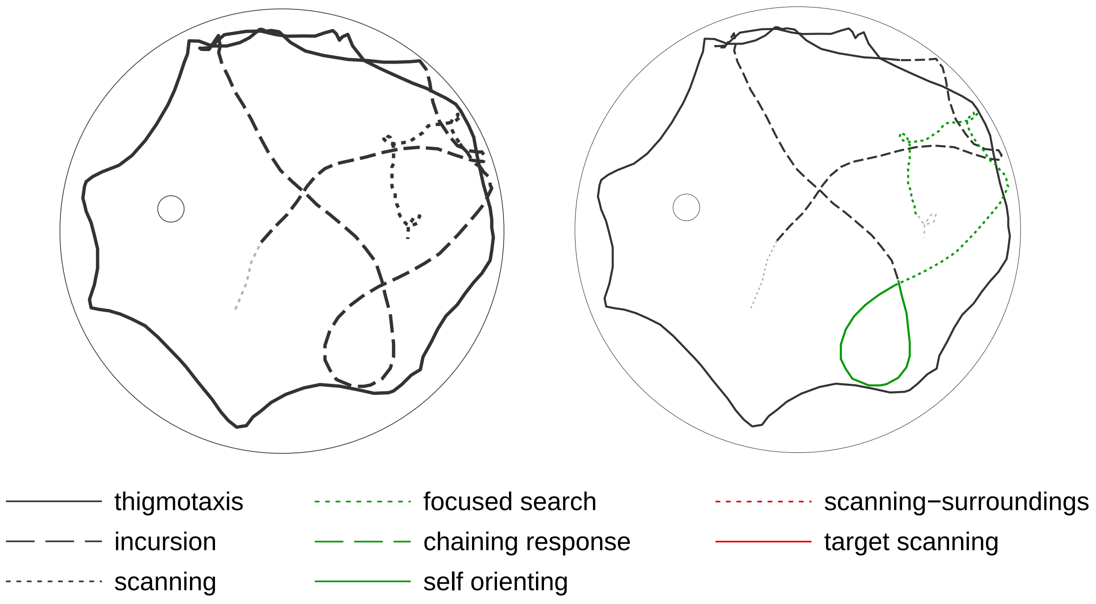


Figure 3.10: Detailed classification of one trajectory. *Left*: classification results using constant class weights. *Right*: results using weights as defined in Equation 3.4. As can be seen in the former case the self-orienting loop (continuous green line) and focused search segments (dotted green line) did not get detected and were replaced by neighbouring classes (incursion, dashed black lines, and scanning, dotted black lines, respectively).

of matching segments in both classifications were compared. Mismatches were discarded and not used in the mapping of strategies to paths.

Table 3.3: Mean and maximum length (in cm) of consecutive intervals of each class for the 250 cm / 90% overlap classification (Table 3.4) with constant weights ($w_k = 1$ in Equation 3.3) and after adopting differentiated weights for minor and major classes (Equation 3.4). See Figure 3.6 for a description of the behavioural classes.

k	**Class**	**Mean seg. length [cm]**		**Max seg. length [cm]**		$w'_k = w_k(Eq.\,3.4)$
		$w_k = 1$	$w_k = w'_k$	$w_k = 1$	$w_k = w'_k$	
1	Thigmotaxis	359.3	328.6	3,000	3,000	1.0
2	Incursion	194.5	191.2	1,225	1,200	2.4
3	Scanning	118.5	105.7	850	725	3.5
4	Scanning surroundings	145.2	155.5	675	650	4.4
5	Focused search	107.4	125.5	550	575	5.4
6	Self orienting	105.1	139.1	375	400	8.0
7	Target scanning	123.2	129.6	350	375	8.6
8	Chaining response	97.3	139.1	300	400	10.0

3.3.8 Statistical significance tests

Multi-factor testing of variance was done using a Friedman test (Siegel 1956), a non-parametric test that is well suited for the type of data being analysed here, which is not normally distributed. Besides this important characteristic of the data being analysed here, the Friedman test is also a *matched* test, and can control for experimental variability among subjects. For the case considered here, the same animals were analysed over a set of 12 trials; the variability between different trials, that affects all animals, is not taken into account in the Friedman test.

The p-values shown in the analyses answer the question: if the different treatments (control vs. stress) are identical, what is the chance that a random sampling would result in the distribution of values (or *ranks*, as used by the Friedman test) as far apart as observed? Small p-values (< 0.05 in our analyses) lead us to discard the null hypotheses that the results are identical and differences are only due to random sampling.

3.4 Results

This section presents some results obtained with the analysis method proposed here. First, results of a validation of the method will be presented; this was done in order to gain confidence in the method and ensure that the results are consistent.

Table 3.4: Parameters and results for three different classifications with variable segment lengths and overlaps. The coverage value indicates the percentage of the swimming paths that are covered by segments of known classes. This was the main criterion for determining the quality of the clustering results. Different values for the target number of clusters in the first clustering stage were used and the ones giving the best results (highest coverage values with a low classification error) were selected (see Section 3.3.5 and Figure 3.8). The clustering algorithm uses a second stage where larger clusters with mixed labels are further sub-divided. The agreement between two classifications was computed by sub-dividing each trajectory into a series of small segments with the same time interval, computing the resulting major class for each interval (similar to what was done in Figure 3.13) and comparing them for both trajectories. As the table shows, the agreement for the two 250 *cm* classifications is very good; for classifications with different segment lengths the agreement is slightly lower.

	Classification 1	**Classification 2**	**Classification 3**
Segment length	300 cm	250 cm	250 cm
Segment overlap	70%	70%	90%
Minimum path interval	90 cm	75 cm	25 cm
Segments	8,847	10,388	29,476
Labels	989	1,301	1,605
Clustering constraints	6,464	7,390	6,599
Clusters (first stage)	37	35	75
Clusters (final)	125	139	200
Unclassified segments	18.8%	20.0%	28.2%
Classification errors	0.51%	0.62%	0.19%
Coverage	94.1%	94.9%	97.3%
Agreement with 1	-	84.2%	71.1%
Agreement with 2	84.2%	-	97.5%
Agreement with 3	71.1%	97.5%	-

After this, the results from the analysis of the data set introduced in Section 3.1 will be presented.

3.4.1 Method validation

In order to validate the classification method, the initial classification results for the data set analysed in the next section, introduced in Section 3.1), were compared to a manual tagging of the swimming paths. The manual tagging consisted in visually looking for the presence of four behavioural traits which are easily identifiable in the trajectory and to make sure that the proposed classification method was able to identify them (i.e. that it was able to find segments belonging to the identified classes). The four behavioural classes used for this were: i) Thigmotaxis; ii) scanning of the region around the platform (called *Target Scanning* here); iii) *Incursion*, or paths where the animal still touches the walls but start moving inwards; and iv) *Scanning*, paths where the more central regions of the arena are searched (these classes are described in more detail in Section 3.3).

The comparison of the manually tagged trajectories and automatic classification results (presented in the next section) are shown in Table 3.5. The data set considered here is the same as the one introduced in Section 3.1. The table shows how many manually identified behavioral classes were not identified by the automatic classifier. The values show that the automatic method is in good agreement with the manually tagged trajectories, showing a maximum disagreement of about 7% for the Scanning class, which is a problematic case because the automatic classification also looks for more specific types of behaviour, such as Circular Response and Self-orienting, which were not taken into account in the manual classification and were usually identified as Scanning. The results for Scanning in Table 3.5 therefore include also these other sub-classes of behaviour (the same can be said for Target Scanning, where the results for Scanning Surroundings – a class not included in the manual labelling – were included as well).

Table 3.5: Manual classification results comparison. Table shows how many types of behaviour from a manual labelling were not identified by the automatic classifier. The first column shows the total number of manually tagged trajectories that displayed a type of behaviour and the second one the number from them that were not identified by the classifier. Values show a good agreement between the manual labelling and the automatic classifier. Results for the Scanning class include also the sub-classes of behaviour *Circular Response*, *Self-orienting*, and *Focused search* (Section 3.3), which were not included in the manual classification. In the same way, results for Target Scanning also include values for the *Scanning Surroundings* class.

	Tagged	**Not identified**	**Error**
Thigmotaxis	240	6	2.5%
Incursion	300	5	1.7%
Scanning	152	10	6.6%
Target scanning	67	0	-

It has to be noted that the manual tagging of the trajectories did not quantify to which extent each one of the four behavioral classes was present in each trajectory. It could only identify the presence or not of a behavioural class within a trajectory and is therefore not as reliable as the classification method proposed here. Nevertheless, the manual classification results, which show the proportion of trajectories that displayed each type of behaviour (Figure 3.11), suggest that stressed animals tend to spend at least some time next to the walls (thigmotaxis) in significantly more trials than non stressed ones. This is also confirmed by the more detailed classification results, presented in the next section, that is able to quantify to what extent each class of behaviour is adopted in each trial. The results of the next section also show that other types of behaviour, not easily identifiable by manual inspection, show clear differences between both groups of animals.

3.4.2 Application: comparison of stressed and non-stressed groups of animals

The analysis method proposed here was applied to a set of 57 rats, 30 of which were submitted to peripubertal stress (Márquez et al. 2013); the other 27 animals were the control group. The data consisted of recordings of swimming paths over a set of 12 trials, divided into three daily sessions with 4 trials each.

Commonly used behavioural measures of learning, such as escape latency (Figure 3.12A), fail to show significant differences between the two groups. Although differences in movement speed are pronounced (Figure 3.12B), they cannot be unambiguously interpreted as differences in learning since it does not take into account factors such as the path to the platform that was taken.

3.4.2.1 Behavioural classes

The method presented here produces results that offer a detailed overview of the swimming paths and are able to identify four additional types of behaviour that were not included in the manual labelling of swimming paths. Those were (Figure 3.6): *chaining-response* (D. Wolfer and H. Lipp 2000), characterised by concentric paths where the animal sweeps all the points at the platform distance from the wall; *focused-search*, where the animal limits its search to very small areas randomly and repeatedly sweeping them; *self-orienting* (Graziano et al. 2003), characterised by a loop where the animal orients itself in the arena; and *scanning-surroundings*, or paths that cross a critical region around the arena but are not limited to this region. Figure 3.6 shows examples of swimming paths including all eight behavioural classes, as identified by the new classification method.

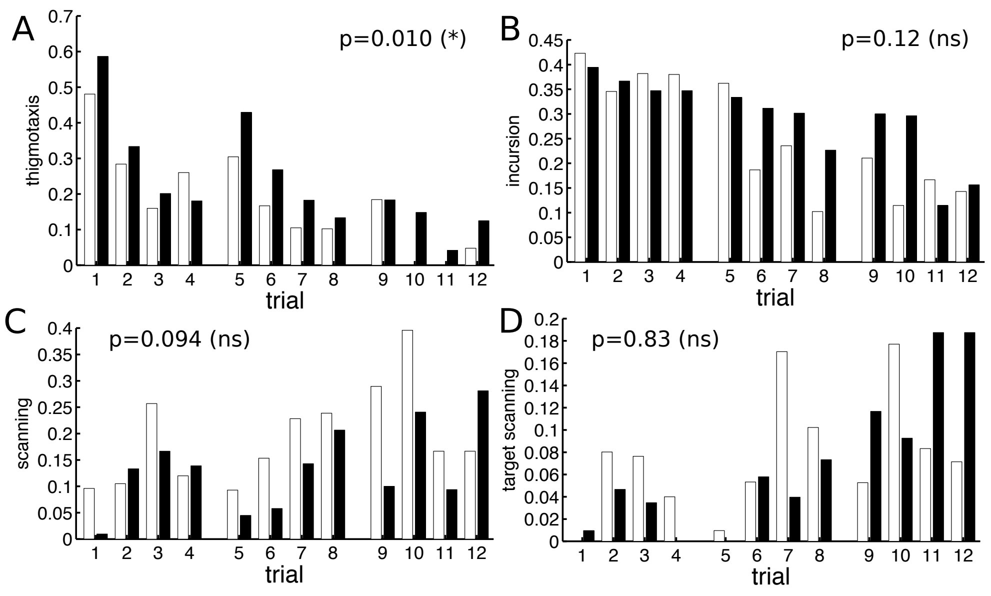


Figure 3.11: Results of a manual classification of complete swimming paths. Labels corresponding to four behavioural classes (thigmotaxis, incursion, scanning, and target scanning / plots A-D respectively; see Figure 3.6 for a description of the classes) were assigned to swimming paths depending on the types of behaviour that were identified by a manual inspection. Multiple behavioural classes could be assigned to each swimming path; values show the percentage of swimming paths for a trial that show traits of each class. The data set is introduced in Section 3.1 and consists of a total of 57 rats, 30 of which were submitted to peripubertal stress. *White bars*: control group; *Black bars*: stress group. Both groups were compared over the complete set of trials using a Friedman test. Panel A shows that stressed animals tend to spend at least some time next to the walls (thigmotaxis) in a trial more often than non-stressed ones. The more detailed classification results, presented in Section 3.4.2, take also into account to which extent each strategy is adopted within a trial and confirm this. They also show other differences in behaviour between the two groups which cannot be easily identified by a simple manual inspection of the trajectories.

3.4.2.2 Classification of swimming paths

The swimming path classification, as described in detail in Section 3.2, was done three times: once for segment lengths of 300 cm with 70 % overlap between segments, then for segments of 250 cm with 70 % overlap, and finally again 250 cm with 90 % overlap. In each case the clustering algorithm was run multiple times for different target number of clusters in the first clustering stage. The results giving the highest coverage values (i.e. fewer unclassified trajectory segments), while still showing small number of classification errors, were then chosen (Section 3.3.5). The different classifications were then compared and combined for producing the final results.

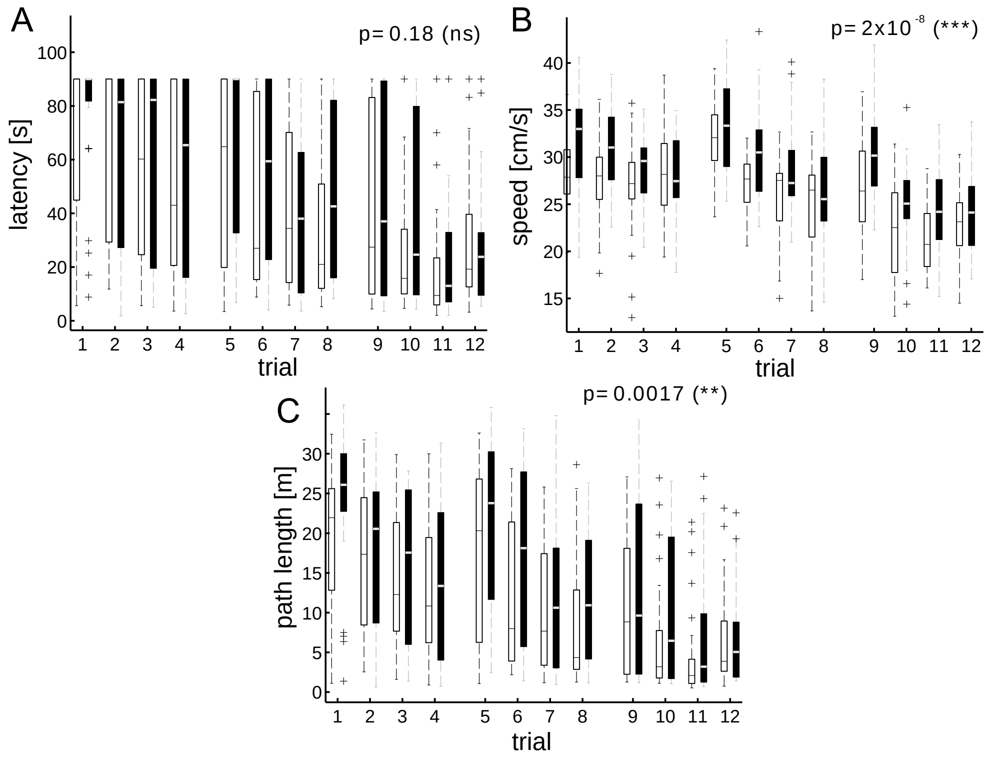


Figure 3.12: Comparison of full trajectory metrics for two groups of animals over a set of 12 trials. *White (black) boxes*: control (stress) group. Boxes represent the first, second (median, shown as a band) and third quartiles; whiskers are the minimum and maximum values. Outliers are marked as a cross. The p-values of a Friedman non-parametric test comparing both groups of animals over the full set of trials is shown in each plot (see Section 3.3.8 for a discussion and interpretation of the Friedman test and the p-values). **A** Escape latency shows a wide dispersion of values until later trials. Trials were limited to 90 seconds, after which the animals were moved to the platform and the trial was ended. **B** Average movement speed shows that stressed animals move significantly faster than non-stressed ones. **C** Average path length. Stressed animals tend to sweep longer paths than control animals; the difference between the two groups is however less distinctive than when comparing the movement speeds.

In order to combine two classifications, the classes of matching segments in both classifications were compared. Mismatches were discarded and not used in the mapping of strategies to paths. The final results were based on a combined classification of both the results for segment lengths of 250 cm.

Parameters and results for the three classifications performed here are listed in Table 3.4. The percentage of corresponding segments that were assigned to the same class, or the consistency rate between both classifications, is shown in the last row of the table. The results show that the agreement between the two classifications with 250 cm segment length is very good ($> 97\%$). Between classifications with different segment lengths (250 and 300 cm) the agreement is slightly

$(15-25\%)$ lower. This is mostly due to the fact that the same segment mapping parameters (Section 3.3.6) were used for both segment lengths. However, this had no impact in the final classification since they were computed from a combination of multiple classifications (Section 3.3.7), which increased the confidence in the results.

Note that although the agreement factor seems to indicate an error of 10 to almost 30%, the differences are mostly due to segments that show traits of two different classes of behaviour and were mapped to opposite classes in both cases. These differences do not impact the final classification results significantly, because when computing the resulting class for a given point of a swimming path, all segments that overlap the point are taken into consideration. This means that small discrepancies of the actual segment classes in the case of transition segments average out, or that when there is a difference in the results, it is limited to only short path segments, of the order of one or two minimum discrete path length intervals (between 25 and 75 cm depending on the segmentation parameters).

3.4.2.3 Evolution of strategies

After classifying the path segments, the distribution and evolution of strategies for each swimming path was computed. This was done by computing the resulting behavioural class for each minimum discrete path interval (25-90 cm depending on the segmentation, Table 3.4). The class for each interval was selected based on the classes of the overlapping segments.

Figure 3.13 shows the classification results for the first 6 trials for the stress and control groups of animals. The detailed overview of each swimming path cannot be achieved with previous classification methods. The results show gradual changes in behaviour over the trials, and indicate at which points in time during a trial a given strategy was adopted. Multiple strategies are usually present within a single trial.

Initially, behaviours such as thigmotaxis and incursion dominate the trials, as the animals look for wall contact where they presumably feel safer. However, it can be seen that already at the second or third trials an increased presence of the other more sophisticated strategies is observed. In these strategies animals explore more central regions of the arena (scanning), orient themselves (self-orienting) or actively look for the platform (chaining response, surroundings/target scanning).

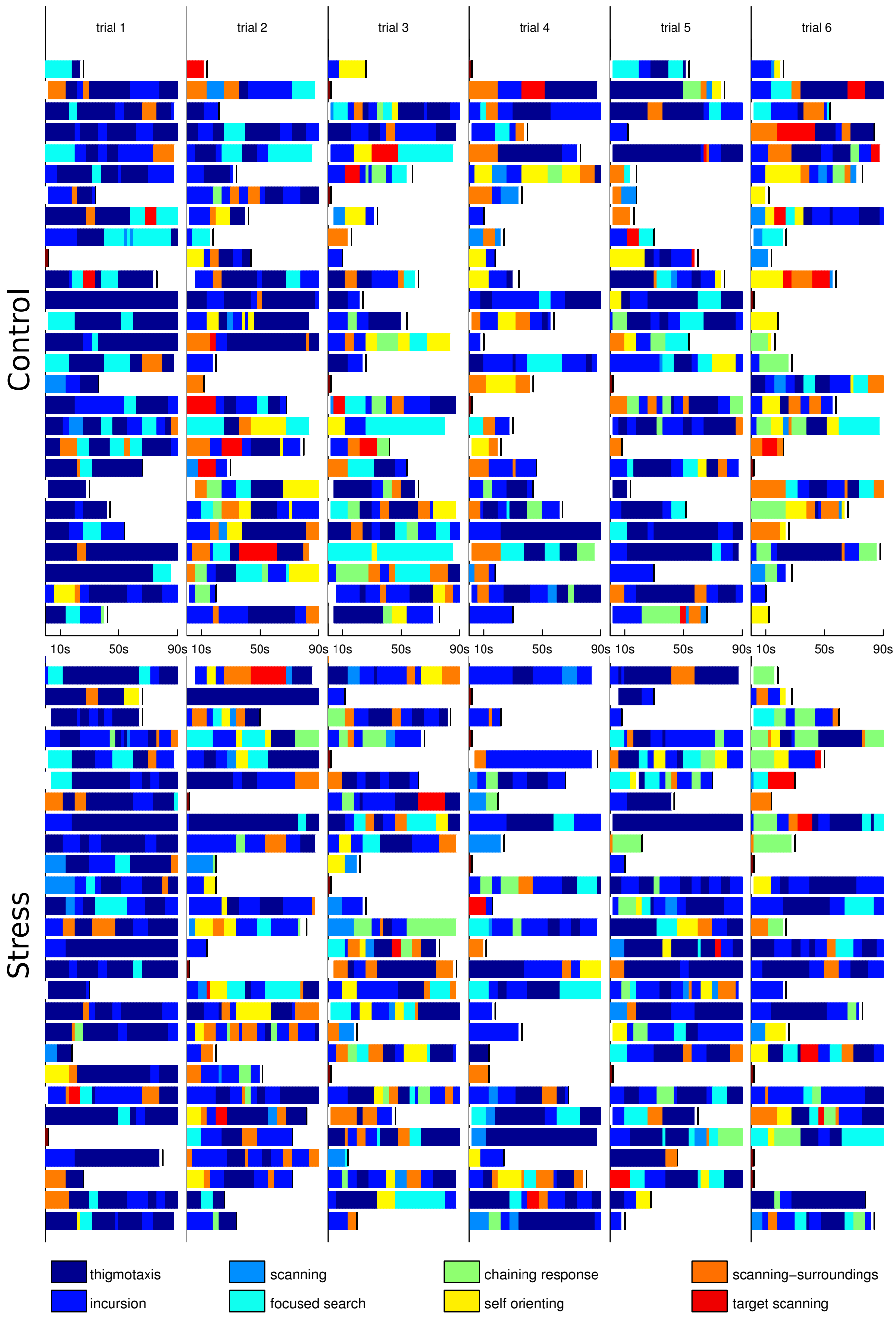


Figure 3.13: Classification results for the 6 first trials for the control (top) and stress (bottom) group. Each bar represents a full trial (up to 90 seconds) and shows changes in exploration strategies over the trial. Short paths, where the animal found the platform directly, and which were not segmented, are marked in dark red. White boxes indicate segments with behaviour not falling into any of the classes and which could not be categorised. The results show that paths almost always correspond to multiple types of behaviour. Also, it can be seen that on later trials animals are not only able to find the platform faster, but they also change their strategies.

3.4.2.4 Distributions of strategies

Basic analysis methods based on simple measures such as the escape latency (Figure 3.12A) fail to show significant differences between the two groups of animals. One noticeable difference, however, is that stressed animals tend to move faster than non-stressed ones; their paths to the platform are also longer (Figure 3.12B-C).

The higher movement speeds of the stressed animals would suggest lower escape latencies since, other behaviour characteristics being equal, they would find the platform by chance more often. This, however, is not observed (Figure 3.12A). In order to better understand and characterise the differences between the two groups of animals, the technique presented here was used to compare how the preference for the different strategies between the two groups of animals differs.

Figure 3.14 shows the distribution of strategies, using the same eight classes of behaviour defined above, for both groups of animals over the complete set of 12 trials. The plots show the average path lengths spent on one strategy over a trial. As expected from the differences in speed between both groups (Figure 3.12B), stressed animals generally show longer paths than the control group. However, the difference in path lengths between the two groups is not homogeneous among all strategies. The results show that for the stressed animals there is a significant increase in thigmotaxis, incursion, and scanning classes, all associated with low chances of finding the platform (because most of the time is spent next to the walls or centre of the arena in these behaviour patterns). Other, more cognitively sophisticated strategies, such as self-orienting and target scanning, don't show significant differences among the two groups. The exception is the chaining-response class, which shows a slight increase in the stress group. This difference, however, is small compared to the differences between stress and control animals in the first group of classes (i.e. thigmotaxis, incursion, and scanning).

The analysis presented here suggests that, although stressed animals sweep significantly longer paths, most of that difference can be attributed to staying near the walls and using simple exploration strategies such as scanning. More sophisticated strategies, where animals actively look for the platform, are used similarly by both experimental groups, explaining why stressed animals move faster but take about the same time (or even longer) to escape the maze.

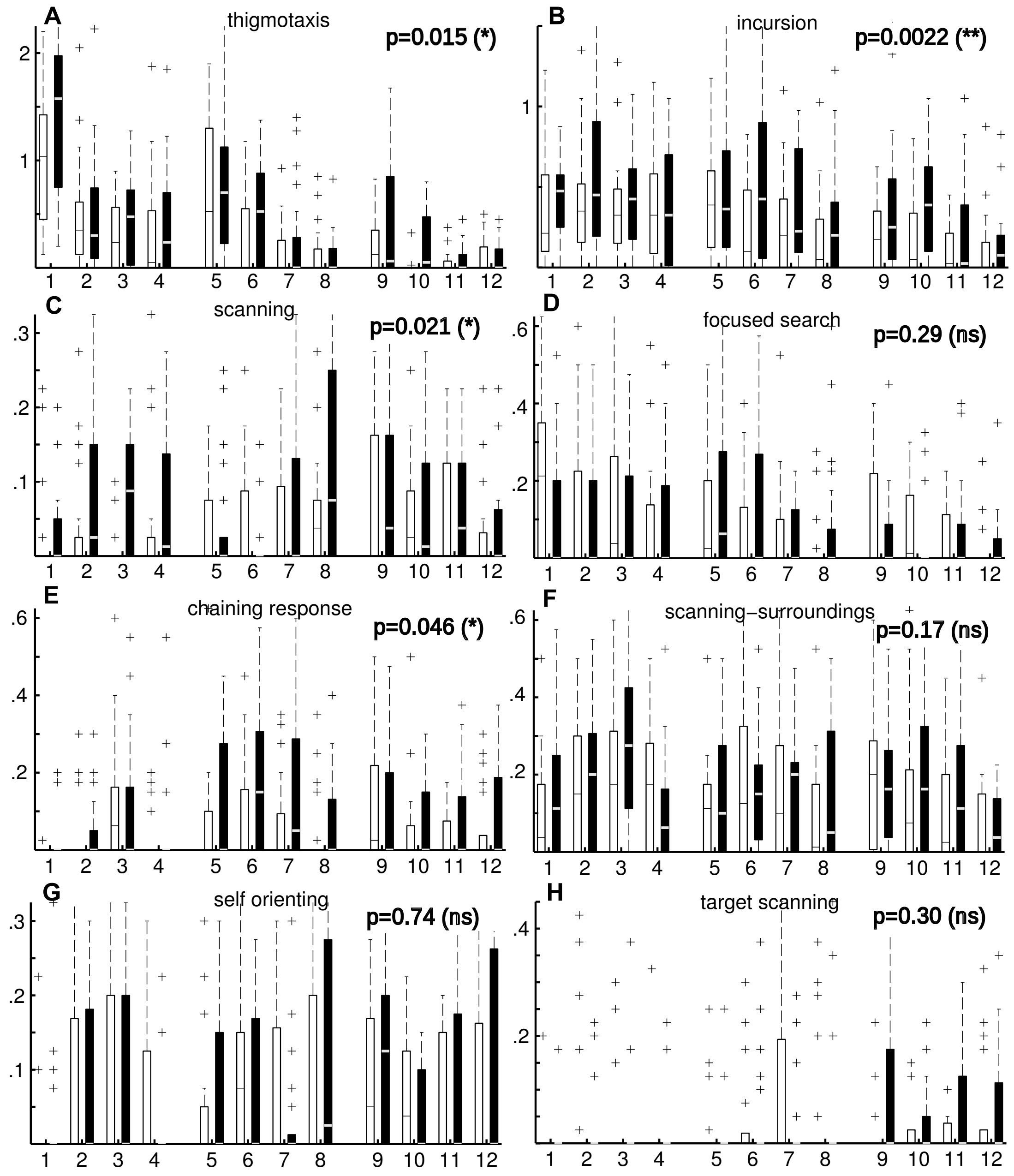


Figure 3.14: Average segment lengths for each strategy adopted by stress (black) and control group (white) of animals for a set of 12 trials divided in 3 sessions (days). Plots show the average length in meters that animals spent in one strategy during each trial. Bars represent the first and third quartiles of the data; line shows the median and crosses the outliers. Whiskers (when shown) indicate minimum and maximum values. A Friedman test (Section 3.3.8) was used to compare both groups of animals over the complete set of trials; p-values are shown on the top right. The results show that, as expected, stressed animals display longer average paths much more often but the increase is non-uniform among the different strategies. According to the plots there is a clear difference in the path lengths for the thigmotaxis, incursion, and scanning strategies, all of which are characterised by low chance of finding the platform. For the scanning-surroundings, self-orienting, and target scanning strategies, all which are associated with an increased chance of finding the platform, no statistically significant differences were found. Chaining response shows a slight difference in favour of the stress group; focused-search shows no significant differences. These results may explain why stressed animals sweep longer paths but on average they don't find the platform faster that non-stressed animals (Figure 3.12).

3.4.2.5 Strategy transitions

Table 3.6 presents the results from another perspective: it compares the transition probabilities within the same trial between strategies for both the control (left table) and stress (right table) groups of animals. Values that deviate appreciably between both groups are shown in bold. It can be seen that stressed animals show a higher tendency for changing to thigmotaxis and incursion strategies, whereas the control group shows higher values for a transition to self-orienting and scanning surroundings strategies, both of which lead to increased chances of finding the platform. These results are in agreement with the conclusions above. Furthermore, they provide an alternative angle to look at the behaviour of both groups of animals, which would have been impossible using previous classification methods.

Table 3.6: Transition probabilities of strategies within trials for the control (left) and stress (right) group of animals. Rows and columns indicate the starting and ending strategies respectively. Row values (for the same starting strategy) are normalised. Bold values indicate the most significant differences between the two groups. Results show a higher tendency for stressed vs non-stressed animals to move to less efficient strategies (thigmotaxis, incursion, and scanning - the ones where chances of finding the platform are reduced); this is in agreement with results from Figure 3.14. *TT* = thigmotaxis, *IC* = incursion, *SC* = scanning, *FS* = focused search, *CR* = chaining reaction, *SO* = self orienting, *SS* = scanning surrounding, *ST* = scanning target.

	TT	**IC**	**SC**	**FS**	**CR**	**SO**	**SS**	**ST**
TT	-	0.51	0.035	**0.23**	0.022	0.036	0.15	0.015
IC	0.35	-	0.055	0.12	0.13	0.13	0.17	0.036
SC	0.22	0.31	-	0.098	0.033	0.081	0.17	0.094
FS	**0.45**	0.25	0.015	-	0.051	0.045	0.15	0.033
CR	0	0.30	0	0	-	**0.37**	**0.33**	0
SO	0.28	0.25	0.11	0.12	0.082	-	0.089	0.078
SS	0.32	0.27	0.060	0.040	0.14	0.057	-	0.12
ST	0.17	0.35	0.083	0.083	0	**0.13**	0.20	-

	TT	**IC**	**SC**	**FS**	**CR**	**SO**	**SS**	**ST**
TT	-	0.54	0.017	0.13	0.054	0.075	0.17	0.020
IC	**0.51**	-	0.054	0.11	0.073	0.071	0.16	0.023
SC	0.25	0.30	-	0.065	0.095	0.13	0.11	0.053
FS	0.27	**0.42**	0.025	-	0.10	0.046	0.12	0.021
CR	**0.32**	0.23	0.059	0.024	-	0.094	0.15	**0.13**
SO	0.31	**0.36**	0.057	0.042	0.061	-	0.15	0.024
SS	0.33	**0.36**	0.043	0.055	0.12	0.040	-	0.046
ST	**0.28**	0.30	0.024	0.13	0.059	0.021	0.18	-

Finally, Figure 3.15 shows that stressed animals also change strategies significantly more often than non-stressed ones within a single trial. It has been suggested (Aston-Jones et al. 2000; Luksys, Gerstner, et al. 2009; Luksys and Sandi 2011) that high levels of arousal or stress lead to labile attention and frequent strategy switches that may prevent efficient learning. In the reinforcement learning framework (Section 2.2.3), which is relevant for computational modelling of the Morris Water Maze (Richmond et al. 2011; Vasilaki et al. 2009), it has been shown (Luksys, Gerstner, et al. 2009; Luksys and Sandi 2011) that stress and anxiety can lead to steeper reward discounting, which may be the computational reason behind excessive strategy switches and impaired ability to learn tasks with significantly delayed rewards. The results based on the classification of strategies provide empirical evidence to support this view.

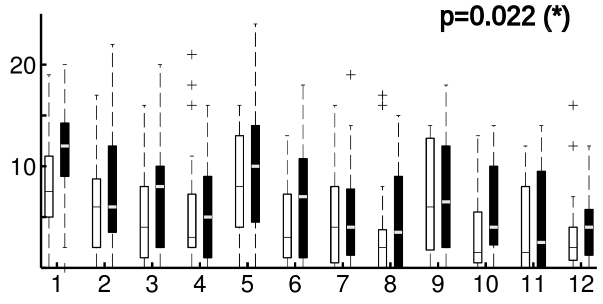


Figure 3.15: Number of transitions between strategies for both groups showing that stressed animals change their behaviour more often within single trials

3.5 Discussion

The classification of swimming paths in the Morris Water Maze into behavioural classes is a useful method to study spatial learning in rodents, since the different classes of behaviour can be mapped to different stages of learning (D. P. Wolfer, Stagljar-Bozicevic, et al. 1998). However, for some experiments, especially the ones with a limited number of animals, longer trials, or larger arena sizes, the discretisation of all swimming paths into only a few behavioural classes might not be adequate. This is due to swimming paths that display characteristics of more than one class of behaviour and that therefore cannot be reliably assigned to a single class of behaviour (D. Wolfer and H. Lipp 2000). This in effect means that results are valid only in a statistical sense, or for larger number of swimming paths. In order to address these limitations, a new, more granular classification method that allows a detailed description of all strategies employed by the animals in a single trial was proposed here.

Contrary to previous approaches, the main target of the classification method introduced here is not the full swimming paths, which can vary greatly in length and consist of multiple types of behaviour, but rather shorter path segments of constant length. By classifying multiple segments of swimming paths, changes in behaviour within a single trial can be detected and quantified. As a result, instead of one single behavioural class, a distribution of classes is assigned to each swimming path. It was shown that such detailed quantification can reveal subtle and novel behavioural differences between two groups of animals (Figure 3.14); approaches comparing only individual path measures failed to provide clear insight. In addition, the method presented here not only provides information about which types of behaviour were exhibited, but also at which point during the trial they were adopted (Figure 3.13).

This chapter presented a semi-automated classification method that requires only a reduced set of labelled data to map path segments to classes of behaviour. This is a useful feature, since it is impractical due to the large number of segments usually generated. Earlier work by Graziano et al. (2003) has also proposed an automated classification method of swimming paths. Their method was based on linear discriminant analysis (LDA), and made use of a high-dimensional feature space (more than 20 features), which made their classifier very robust. However, in contrast to the work presented here, their method assigned complete trials to a single class of behaviour. Here, instead, this problem was approached by applying a semi-supervised clustering algorithm over a smaller feature space. The choice of the clustering algorithm was motivated by the fact that clustering is ideally suited for finding structure in data, without a priori knowledge of the classes. Therefore, behavioural classes do not have to be predefined, but can rather be identified by looking at common characteristics of elements of individual clusters. Also, clusters containing ambiguous segments (for example transitional segments between two classes) can be easily discarded.

In the implementation here, a set of features that captured both geometrical (focus, eccentricity, distance to centre spread, maximum loop length, inner radius variation) and positional (median distance to centre, target proximity, central displacement) aspects of the segments was chosen. A smaller set of features was preferred because more features span a space of higher dimensionality, which tend to require a larger number of clusters to separate the data into their classes (clusters that contain elements of a single class; in general multiple clusters can be mapped to a single class). This in turn means that more labelled data has to be provided to map these clusters to their respective classes.

Standard clustering algorithms usually fall in the unsupervised class of algorithms and find patterns in data without any previous knowledge. However, here a semi-supervised approach was adopted. In this approach a partial set of labelled data was used to guide the clustering process and improve the results. The labelled data is also essential to map clusters to classes and to compute an error estimate.

The number of required labelled segments for classifying a set of swimming paths will depend on the specific data set at hand, on the number of classes and on parameters such as the minimum number of labelled segments. However, the labelling and clustering process can be performed incrementally, and so the number of labels can be increased until a suitable classification is found. To help with this process a custom Graphical User Interface (GUI) was developed here (Appendix A). This GUI makes it easy to label swimming paths or segments, to cluster the data, and to visualise the results of the clustering. All the code developed here is freely available in GitHub[‡].

The method presented here was successfully applied to a set of experiments with two groups of animals for which standard analysis methods based on single performance measures, such as comparing the escape latency values, failed to find any significant differences in the performance between the groups, even though one of the groups had been subjected to a strong manipulation. This manipulation led to one group of animals moving significantly faster than the other one, but no other differences in performance could otherwise be identified. Previous categorisation methods for swimming paths also failed because of the inherent ambiguity of the classification and the relatively small data set that had to be analysed. The method presented here was developed with the objective of overcoming these limitations and showed that the two groups of animals in the experiments considered here display indeed subtle differences in their behaviour.

One valid criticism of the classification methods of swimming paths is that the choice of measures can be very subjective. In his work, Korz (2006) makes this point and provides an alternative, bottom-up, approach for the analysis of swimming paths. His method is based on normalising the trajectories by reducing them to 50 equidistant points and using the actual coordinates of the points to compare trajectories. This is done by using principal component analysis (PCA), and has the advantage that no arbitrary measures have to be introduced. He also shows that the first few principal components are sufficient to account for most of the variability in the swimming paths, and can therefore be used to simplify trajectories. Although PCA-based methods have disadvantages such as interpretability of the principal components, in principle a similar approach could be used here for swimming path segments, if the intention were to avoid a subjective choice of features. In this case, the feature set could be defined by the first principal components of the segments rather than the geometrical and positional measures used here.

[‡]https://github.com/RodentDataAnalytics/mwm-ml

The work presented in this chapter can serve as a basis for defining more sophisticated scoring mechanisms for swimming paths. Scoring methods of swimming paths in the MWM were introduced in previous studies (e.g. Petrosini et al. 1998) and are important because they make it easy to compare swimming paths according to certain criteria between different groups of animals. The basis for such scoring system strategies could, for example, be a weighted sum of the relative distribution of strategies where their correlation with efficient swimming paths is used as a weighting factor. This approach, however, was not pursued here and was left for further investigations.

While the methods presented here were applied to trajectories of rats in the MWM, the method is likely to work similarly well for other species of rodents, such as mice. As mice and rats exhibit comparable trajectories in MWM, even if they learn at different speeds and possibly use slightly different strategies, the method itself should remain applicable, even if the exact number of behavioural classes and their specific description may differ. The method presented here is also general enough and can be extended to other behavioural tasks that employ different types of mazes and trajectories. The next chapter will show how it can be extended to a completely unsupervised paradigm method and be applied to another experimental setup, the Active Allothetic Place Avoidance (AAPA) task.

Summary This chapter introduced a semi-supervised classification method for Morris Water Maze swimming paths. The method offers a more detailed overview of the behaviour of animals and is able to identify multiple exploration strategies within single trials, something that was not possible in previous approaches. The method was here successfully applied to a data set where it found subtle differences in behaviour between two experimental groups of animals where other methods failed. The next chapter will build on this and will show how the method presented here can be extended to another experimental setup, the Active Allothetic Place Avoidance task. It will also consider how the method can be turned into a fully unsupervised method.

Chapter 4

Classification of trajectories in the Place Avoidance Task

The Active Allothetic Place Avoidance (AAPA) task (Cimadevilla, M. Wesierska, et al. 2001; Dockery and M. J. Wesierska 2010; Stuchlík et al. 2013) is an experimental setup in which animals are placed in a dry circular arena where they are free to move but have to learn to avoid a shock sector. If they enter this sector they are punished by periodic mild electric shocks until they leave it again (Figure 4.1). In this task the arena is slowly rotating over time and the shock sector is fixed in space, so that animals have to constantly move around to avoid receiving a shock. The AAPA task is a variation of the passive Place Avoidance Task (Bammer 1982; Haroutunian et al. 1985), in which the arena does not rotate.

Navigation in a stable environment requires two types of memory mechanisms (Bures et al. 1997; Fenton et al. 1998) which are brought into conflict by the rotation of the arena in the AAPA task: the distal (room) cues and the local (idiothetic) orientation within the arena, defined by the self-motion and local references such as urine, faeces, and other marks left by the animals. In the AAPA task animals have to learn to ignore the irrelevant local cues and to use only the room reference frame in order to avoid the shock sector.

The AAPA task has been shown to be hippocampus dependent (Cimadevilla, Kaminsky, et al. 2000; M. Wesierska 2005) and more sensitive to damages in this brain region than the Morris Water Maze (MWM), described in Chapter 3. In the MWM, animals have to also make use of only distal cues to orient themselves (Morris 1981) and find the target platform. It has been shown (Cimadevilla, M. Wesierska, et al. 2001; Stuchlik, Rezacova, et al. 2004), however, that lesions to the hippocampus don't affect the ability to navigate to fixed goals in space, such as

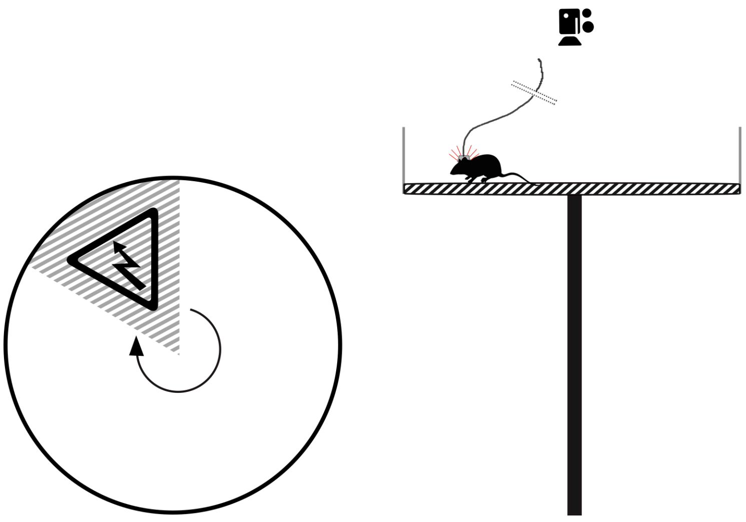


Figure 4.1: The Active Place Avoidance task. Animals are placed on top of an elevated arena which is slowly rotating (1 RPM). They can move freely around the arena but need to learn how to avoid the shock sector, which is fixed in space. If they enter the shock sector a small current pulse is delivered to their heads through an implanted electrode. The position of the animals is tracked with a LED and a top-mounted camera.

the escape platform in the MWM. In the AAPA task there is no fixed target for the animals to find; instead, they have to constantly move around ignoring local cues in order to avoid the shock region. This makes the task more suitable for studying the effect of drugs or lesions to the hippocampus, known to be fundamental for memory formation and spatial navigation. Another advantage of the AAPA task compared to the MWM is that animals don't have to swim in a tank of water but can instead move normally around the arena, a more natural setting for them.

Common performance measures for place avoidance experiments (Stuchlik and Vales 2008; Vales et al. 2006; M. J. Wesierska et al. 2013) typically compare values such as the total number of shocks received, the maximum shock avoidance time, or the time for receiving the first shock in a session, among others. Although very useful, these abstract performance measures don't give any direct indication of how the animals behave and how their behaviour changes over time.

In the previous chapter the problem of using only single performance measures for comparing the behaviour of animals for the case of the Morris Water Maze was discussed. In order to better quantify and compare the performance of animals a categorisation method for the trajectories was presented there. The classification was done in a semi-automated fashion: a partial set of the data was manually classified and used to constrain a clustering algorithm and to map the clusters to behavioural classes. Because a partial set of the trajectories were labelled there, the classification method falls into the semi-supervised class of algorithms (Section 2.2.4).

In this chapter the analysis method for the MWM introduced in Chapter 3 is extended to AAPA experiments. The method presented here is also based on a classification of the trajectories of the animals inside the arena into stereotypical types of behaviour. Similar to what was done for the case of the MWM, trajectories are first split into segments; these segments are then grouped into different classes of behaviour by means of a clustering algorithm.

This chapter will show not only how the method developed for the MWM can be generalised to other experimental setups, but also how it can be turned into a completely unsupervised one. That is, the classification done here does not make use of any labelled data. Behavioural classes of interest also don't have to be selected in advance but are instead identified by the clustering algorithm itself. These differences are significant because they allow to analyse the data both faster and without any previous knowledge about the types of behaviour seen in the experiments.

The objective of this chapter is to develop an analysis method for AAPA experiments that is complementary to standard performance measures and which can give further insight into how the behaviour of animals changes over time and differs between groups of animals. As a case study, and in order to validate the method, a set of AAPA experiments investigating how silver nanoparticles affect the spatial memory of rats will be analysed.

In what follows the proposed analysis method is first described in detail. The method is then applied to the test data set in order to demonstrate that it can successfully identify stereotypical types of behaviour in the data. The differences in behaviour between treated and untreated animals is then compared with standard analysis results to make sure that the results are consistent. This is followed by a discussion about the results and future work perspectives.

4.1 Experimental Setup

The Place Avoidance Test experiments were conducted at the Nencki Institute of Experimental Biology, Warsaw, Poland. The same basic experimental setup as described in Wesierska et al. (2009) was used. The setup consisted of an aluminium circular arena, 80 cm in diameter, which rotated at a frequency of 1 Hz. The arena was positioned at a height of 80 cm and placed in the centre of a 3x4 meter lightly lit room which contained many stable external visual cues.

Infrared light-emitting diodes (LED) for tracking the position of the animals and a 25G (0.50 mm) hypodermic needle electrode, used for delivering shocks, were attached to the backs of the rats. The experimental setup is shown schematically in Figure 4.1.

Five recording sessions of 20 min each with a fixed shock sector (in the room coordinates) were performed over a set of five consecutive days. This was followed by a test trial five days later where the shock sector was not active. For the trials with an active shock sector animals received a short (0.5 s) constant current pulse whenever they entered a predefined 60° shock sector, which remained fixed across the trials. The amplitude of the shock pulses varied between 0.2 and 0.5 mA and was determined individually for each animal so that shocks did not make the animal freeze or induced attempts to escape the arena. Shocks were repeated every 1.5 seconds until the animal left the shock sector. The position of the animal and the current state of the electrode (shock active or not) was recorded at 25 Hz using a commercial software package (Bio-Signal Group, New York).

4.2 Animals and treatment

Twenty naïve adult (2.5 month-old) male Wistar rats, weighing 270–310 grams, were obtained from the breeding colony of the Center of Experimental Medicine of the Medical University of Bialystok, Poland. They were accommodated in transparent plastic home cages, four per cage, under standard conditions (a constant temperature of 22°C, 12:12 light/dark cycle, humidity at 50%). Water and food were available in the cages ad-libitum. Ten of the animals were treated orally with silver nanoparticles (experimental group) and ten with water (control group).

All manipulations were done according to the European Community directive for the ethical use of experimental animals and the Polish Communities Council for the care and use of laboratory animals.

4.3 Data analysis

The recorded trajectories of the animals were exported from the data acquisition system as text files and further processed by custom data analysis software, written in Matlab.

The data analysis method employed here is an extension and generalisation of the method described in Chapter 3. Like done there for the case of the Morris Water Maze, animal movement paths were first split into shorter segments and then classified into stereotypical behavioural classes by means of a clustering algorithm. The method shown here, however, does not make use of labelled data and is therefore a completely unsupervised algorithm. Contrary also to the method developed for the MWM, here the behavioural classes were not pre-defined but are rather identified by the clustering algorithm itself. This is done by mapping each one of the resulting clusters to one class of behaviour and by making sure that the partitioning of the data does not lead to clusters that are too correlated or empty (to avoid redundant or non-representative classes of behaviour, respectively).

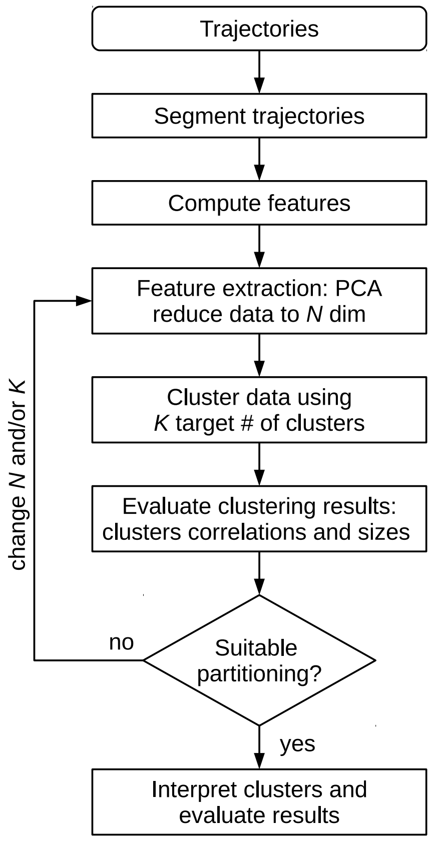

Trajectories
Segment trajectories
Compute features
Feature extraction: PCA reduce data to *N* dim
Cluster data using *K* target # of clusters
Evaluate clustering results: clusters correlations and sizes
Suitable partitioning?
no
yes
change *N* and/or *K*
Interpret clusters and evaluate results

Figure 4.2: Steps of the classification procedure for the AAPA task trajectories

The steps of the analysis done here consisted in (Figure 4.2):

1. splitting the trajectories of the animals into shorter segments;
2. computing a set of features for each segment and reducing the dimensionality of the data;
3. clustering the data;

4. analysing the distribution of the resulting clusters for both groups of animals;

Each one of these steps will be detailed in the next few sections.

4.3.1 Segmentation of trajectories

The main focus of the analysis presented here was to understand how the strategies of animals for avoiding the shock sector evolve over time and differ between treated and untreated animals. Therefore, in a first segmentation step the recorded trajectories were split into segments delimited by entrances/exits from the shock sector. That is, only the parts of the trajectories not falling in the shock sector were considered. However, because the length of the trajectories between shocks varied widely, from the order of a few seconds up to the duration of a trial (20 min), these segments were split further.

Long trajectories usually display multiple types of behaviour; the second segmentation step had per objective to isolate the different behaviours found in a trajectory and to generate a more uniform distribution of segment lengths, making it easier to classify them. The second segmentation used changes in the angular speed as criteria for splitting the trajectory segments. This is because in the Active Place Avoidance Test animals have to move in the angular direction in order to evade the shock sector. Therefore, changes in the sign and magnitude of the angular speed were taken as the delimiting points of the segments. More formally, trajectory points were processed sequentially and added to a sub-segment if the difference between the local and median angular speed of the sub-segment (recomputed for each new added point) was less than 0.6 rad/s; if the difference was larger a new sub-segment was created. Segments shorter than 5 seconds were discarded.

The two segmentation steps are shown schematically in Figure 4.3. From the original 120 trajectories 1,737 segments were generated after the first segmentation step and 5,787 after the second. Other statistics of the two segmentation steps can be seen in Table 4.1.

4.3.2 Computation of features

This section describes the 11 features, that measure different geometrical and positional aspects of the segments, that were used in the classification (Table 4.2). Some features are computed using the room (world) reference frame, that is, the coordinates including the rotation of the arena; other features are computed in

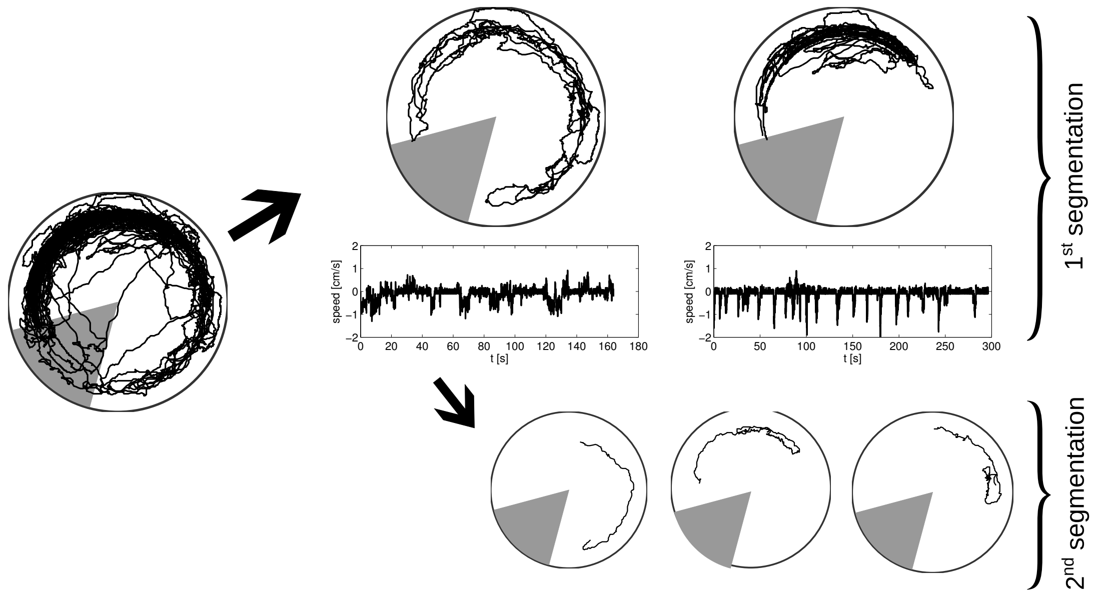


Figure 4.3: Two-step segmentation of the trajectories. In the first step (top) trajectories are split into segments containing only the parts of the paths not falling inside the shock sector. Shown here are two example segments of the trajectory shown on the left. In the second segmentation step (bottom), sudden changes in the angular speed (middle plots) are taken as the delimiting points of the segments. Shown here are three sub-segments of the first example segment from the first step (usually dozens of sub-segments are generated in the second step for each segment).

Table 4.1: Segmentation of trajectories statistics. All lengths are measured in the arena (rotating) reference frame. The last column shows the total length of the resulting segments compared to the input, i.e., without the short segments that were discarded.

Segmentation	Segments	Avg. length	Min. length	Max. length	Rel. length
(Full paths)	120	23,633 cm	3,565 cm	34,138 cm	100%
1st	1,737	1,442 cm	66 cm	24,370 cm	88,2%
2nd	5,787	384 cm	26 cm	3,887 cm	78,3%

the rotating reference frame, i.e., using the real paths swept by the animals. A detailed description of each feature is given in what follows.

Angular distance to shock sector This value measures the angular distance from the centre of the shock sector in the room coordinate frame to the angular centre of the segment. The latter is computed by adding the normalised* position vectors of each sample in the trajectory (i.e. the vector to the centre of the arena),

*vectors are first normalised so that points farther away from the centre of the arena don't overweight more central ones

Table 4.2: Features for the data clustering of trajectory segments

Feature	Unit	Reference Frame
Angular distance to shock sector	rad	Global
Angular dispersion	rad	Global
Angular dispersion (Arena)	rad	Arena
Median log radius	-	Global/Arena
IQR log radius	-	Global/Arena
Trajectory centrality	%	Global/Arena
Median speed	cm/s	Arena
IQR speed	cm/s	Arena
Median angular speed	rad/s	Arena
IQR angular speed	rad/s	Arena
Speed change frequency	s^{-1}	Arena

adding them and then taking the angle of the resulting vector relative to the middle shock sector angle. If the resulting angle is negative, 2π is added to it so that the resulting values are in the $[0, 2\pi)$ range.

Angular dispersion The angular dispersion measures the angular spread of the trajectories in the room coordinate frame. It is here defined as the difference between the maximum and minimum angles of the position vectors (relative to the centre of the arena) of the data samples in the trajectory.

Median/IQR of the log-radius These values are calculated from the trajectory by computing the distance to the centre of the arena for each data sample, taking the logarithm and then computing the median (interquartile range) of the values[†]. The reason for taking the logarithm of the value is to give more weight to the more central areas of the arena, which are more rarely explored by the animals. The median and IQR were also chosen over mean and standard deviation because they are less susceptible to outliers.

Trajectory centrality Measures the relative amount of time that the animal spends at the more central regions of the arena. The value is computed by computing the length of the trajectory falling within a concentric circle with a radius

[†]since the logarithm of zero is not defined, values equal to zero are changed to a small positive number

of 75% of the radius of the arena and dividing this value by the total length of the trajectory.

Median/IQR speed The moving average of the speed between pairs of successive trajectory samples is calculated and the the median/IQR of the resulting values is then computed. More formally, the speed s_i of sample i is computed as:

$$s_i = \alpha \frac{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|}{t_i - t_{i-1}} + (1 - \alpha)s_{i-1} \tag{4.1}$$

where $\mathbf{x}_i$ and t_i are the coordinates and sample times of the ith sample and $\alpha = 0.5$ was adopted here. Once again median and IQR were used instead of the mean and standard deviation because of their improved stability.

Median/IQR angular speed The angular speed between each trajectory sample and the previous one is computed and the median/interquartile-range of the resulting values is then taken. The angular speed between two samples is defined as the difference in the angular position of two samples (relative to the centre of the arena) divided by the time elapsed between them.

Speed change frequency Measures the number of times that the speed changes abruptly within the segment. This happens, for example, when the animal changes its movement direction. Calculated by counting the number of times that the speed (calculated for each successive pairs of samples) crosses 25% of the median speed of the segment, which is taken as the baseline.

The exact threshold value is not so important as long as it is not too high so that almost no segments have samples that cross it and not too low so that even small speed variations within a segment are counted. Through experimentation it was found that a 25% threshold offers a good compromise between these two extremes.

The 11 features computed for each trajectory segment were not used directly for clustering the data. This reason for this is that clustering can be problematic and inefficient in high dimensional vector spaces (Section 2.3.7). Also, since one criterion for selecting the target number of clusters was based on the maximum correlation between clusters (discussed below), it was important to discard redundant features. For this reason a new smaller set of features orthogonal to each other in the spanned space was generated using Principal Component Analysis

(PCA) [‡], one of the most common algorithms used for feature extraction (Section 2.3.7.2).

Here the data was clustered multiple times using different numbers of principal components. The criteria used to select the appropriate number of components, or the dimension of the resulting feature space, are described below.

4.3.3 Clustering

The clustering algorithm that was adopted here was MPCK-means (Metric Pairwise Constrained K-Means, Section 2.3.5), the same as used in Chapter 3. The MPCK-means algorithm supports constraints but this feature was not used here since no labelled data was used. The method described here falls therefore into the unsupervised category of algorithms.

Although it would have been possible to use a standard K-Means algorithm (Section 2.3.3) for the analysis here, the MPCK-means was chosen instead in order to maintain consistency with Chapter 3 and because it supports additional features, such as metric learning, which leads to clusters with different sizes and shapes. This is in contrast to the standard K-means algorithm, which uses a single metric function which usually leads to a more homogeneous distribution of cluster sizes and shapes[§]

4.3.4 Number of principal components and target number of clusters

One of the difficulties in using a clustering algorithm is choosing the appropriate target number of clusters since there is no standard measure to quantify the relative clustering quality. In order to find a suitable value for the target number of clusters and number of principal components (or features) to be used for clustering the data the following criteria were used:

1. The maximum correlation between any two clusters should not be too large ($\leq$ 90% as a thumb rule). A large correlation between clusters means that two or more clusters are too similar and therefore redundant. The correlation between two clusters was computed by averaging the correlation between the

[‡]Wold et al. 1987

[§]Using its default settings, the standard MPCK-means implementation produces also deterministic results, even when not using constraints. However, this is merely due to the fact that the random seed used to initialize the centroids in the absence of constraints is fixed. Here the seed was left at its default value and therefore the clustering results were always deterministic.

first N elements closest to the centroids of each cluster, where N is the size of the smallest cluster;

2. The minimum number of elements inside a cluster is not too small ($\geq 5-10\%$ of the total elements approximately for the range of number of clusters being considered here). This is to avoid having clusters that are empty or close to empty;

The data was clustered using different number of principal components and number of clusters (Figure 4.4). The results that fulfilled both of the above conditions with the minimum number of principal components (or dimensions) and maximum number of clusters (or types of behaviour) were then adopted.

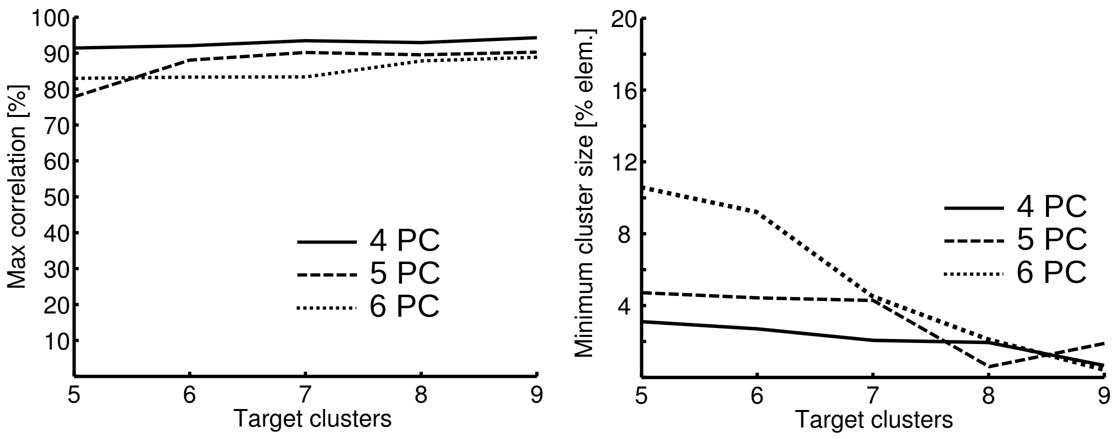


Figure 4.4: Clustering results for different number of principal components and target number of clusters. *Left panel*: maximum correlation between clusters; *Right panel*: minimum cluster size (in % of the total number of trajectory segments). Results are shown for 4 to 6 principal components. As can be seen, results for 4 and 5 PC components (solid and dashed lines) always lead to very small clusters being generated. Results for 6 PC (dotted lines) on the other hand lead to more homogeneous cluster sizes and smaller correlation values between clusters on average.

4.4 Results

As a case study the method is applied to a set of experimental data acquired at the Nencki Institute of Experimental Biology, Warsaw. In the experiments 20 rats were submitted to the AAPA task and their movement trajectories were recorded. Of those animals 10 were treated orally with silver nanoparticles; the other 10, which received water, were the untreated control group. Each animal was submitted to

5 sessions of 20 minutes each during which they could move freely in a circular rotating arena and had to learn to avoid a shock sector which was fixed in the room coordinate frame. This was followed by a test trial in which the shock sector was deactivated.

Previous performance analysis methods of AAPA experiments typically rely on individual performance measure comparisons, which can offer a good indication about which group of animals is performing better, but offers little insight about the differences in behaviour between them. The method shown here, on the other hand, is able to identify and highlight the differences in behavioural patterns in the data. This makes it possible to better understand how animals learn to navigate the arena.

4.4.1 Single performance measures

Standard performance measures for the AAPA usually compare the number of entrances to the shock sector, number of shocks received by an animal during a session, and the maximum period of time between two shocks (see M. J. Wesierska et al. 2013 for a detailed description of these and other statistics). Figure 4.5 shows a series of single performance measures for animals treated orally with silver nanoparticles and the untreated control group. As the results show, there is a clear difference between both groups and the animals in the treated group perform poorly compared to the untreated ones.

The performance measures shown in Figure 4.5 indicate that there is a clear difference between the two groups of animals, but only the results for the number of entrances to the shock area (Figure 4.5a) show statistically significant differences (according to a Friedman test) between both groups.

Besides showing statistical significant differences only in one case, the results shown in Figure 4.5 provide no indication about the types of behaviour that lead to such differences in the first place. The method described here, on the other hand, is able to identify changes in the behaviour of the animals offering detailed information on how the treatment that animals were subjected to affects their comportment.

4.4.2 Trajectories classification

The recorded trajectories (120 in total) for each animal and session were segmented as described in the Section 4.3.1, resulting in $5,787$ trajectory segments. A set of 11 features (Table 3) was computed for each segment. The dimensionality of the data

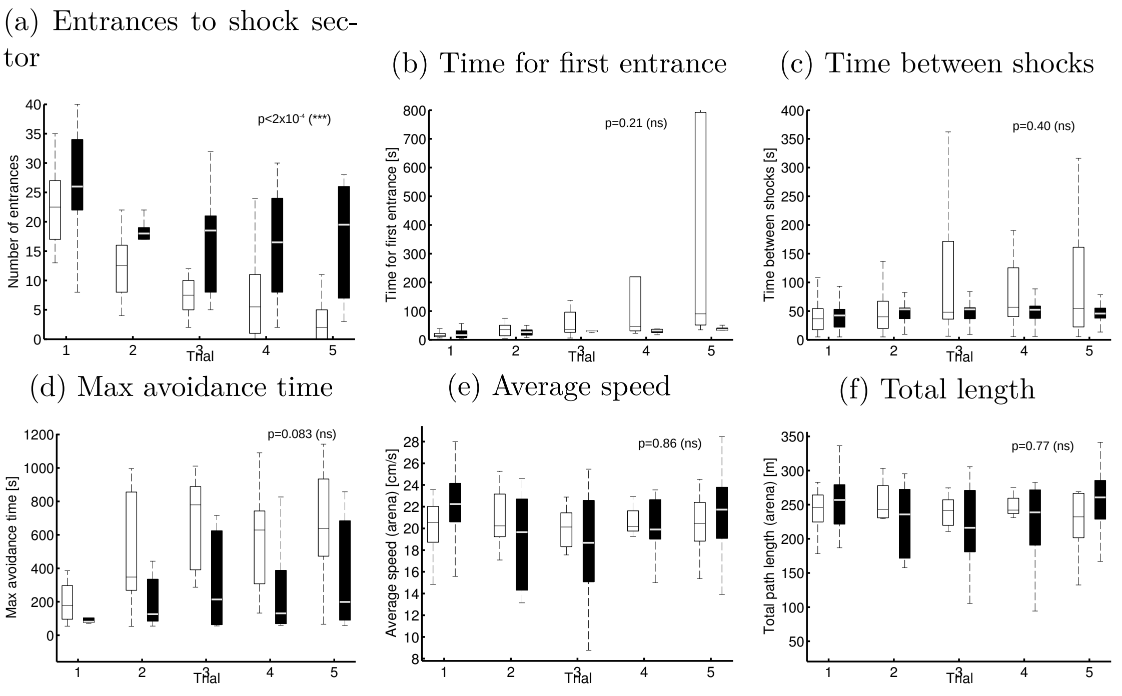


Figure 4.5: Comparison of performance between untreated control (white) and treated (black) animals over a set of 5 sessions. The p-values from a Friedman multi-factor test (Section 3.3.8) are shown on the top right corner. (a): Animals in the control group are able to quickly learn how to avoid the shock sector and are able to avoid the shock area perform on average much better than treated animals, although only the results for the number of entrances to the shock area are statistically significant. (e-f): Average speed and the total length of the trajectories between both groups of animals don't show any significant differences.

was then reduced from 11 to N_{pc} components using principal component analysis (Section 2.3.7.2). The data was then clustered using the MPCK-means algorithm for different numbers of target clusters, N_c . In order to select appropriate values for N_{pc} and N_c the clustering algorithm was run multiple times for different N_{pc} and N_c values and the results were then compared. The criteria for choosing both values is based on running the clustering algorithm multiple times and comparing the maximum correlation between clusters and size of the smallest cluster (Section 4.3.4). They effectively discard classifications which contain redundant or empty (or close to empty) clusters.

Figure 4.4 shows the values for the maximum correlation between clusters (in %) and minimum cluster size (in % of the total number of elements) for N_{pc} between 4 and 6 and an increasing number of clusters, starting at $N_c = 5$. As the right panel shows the results for $N_{pc} = 4$ and $N_{pc} = 5$ (continuous and dashed line) produce always close to empty clusters. Results for both number of principal

Table 4.3: Clustering statistics showing relative size and percentage of elements beginning or ending at the shock sector. A segment beginning at the shock sector are associated with types of behaviour after the animal received one or more shocks. Segments ending in the shock sector are related to behaviour just before the animal (most likely) receives a shock.

Cluster	Elements	Leaving shock	Entering shock
1	16.8%	3.5%	0.2%
2	16.5%	37.6%	85.1%
3	10.9%	36.6%	3.8%
4	29.2%	0.1%	0.1%
5	12.2%	31.9%	64.0 %
6	14.4%	5.8%	1.7%

components have also highly correlated ($\geq$ 90% correlation) clusters (left panel); they were therefore discarded. The results for $N_{pc} = 6$ (dotted lines), on the other hand, show both a more homogeneous distribution of cluster sizes with no empty clusters and a smaller maximum correlation between clusters; $N_{pc} = 6$ was therefore adopted here.

For the number of clusters both $N_c = 5$ and $N_c = 6$ produce similar results but $N_c = 6$ was chosen here. The reason for this choice is that the maximum correlation value doesn't increase between $N_c = 5$ and $N_c = 6$ is an indication that a division into 6 and not 5 clusters is more natural. Starting at $N_c = 7$ a sharp decline in the minimum cluster size is observed, making $N_c = 6$ a suitable choice.

4.4.2.1 Classes of behaviour

Figure 4.6 shows two example segments of trajectories for each one of the resulting 6 clusters. Segments are shown both in the room coordinates (as registered by the camera) and the rotating arena reference frame (real path swept by the animals); the movement speed in the arena reference frame is also shown for each one of the segments. A description of the observed behavioural traits of each cluster is given below. Table 4.3 gives also some statistics for each cluster, such as the relative size and percentage of segments that start or end up within the shock sector. The descriptions for each class are based solely on the observed traits of each one of the computed clusters.

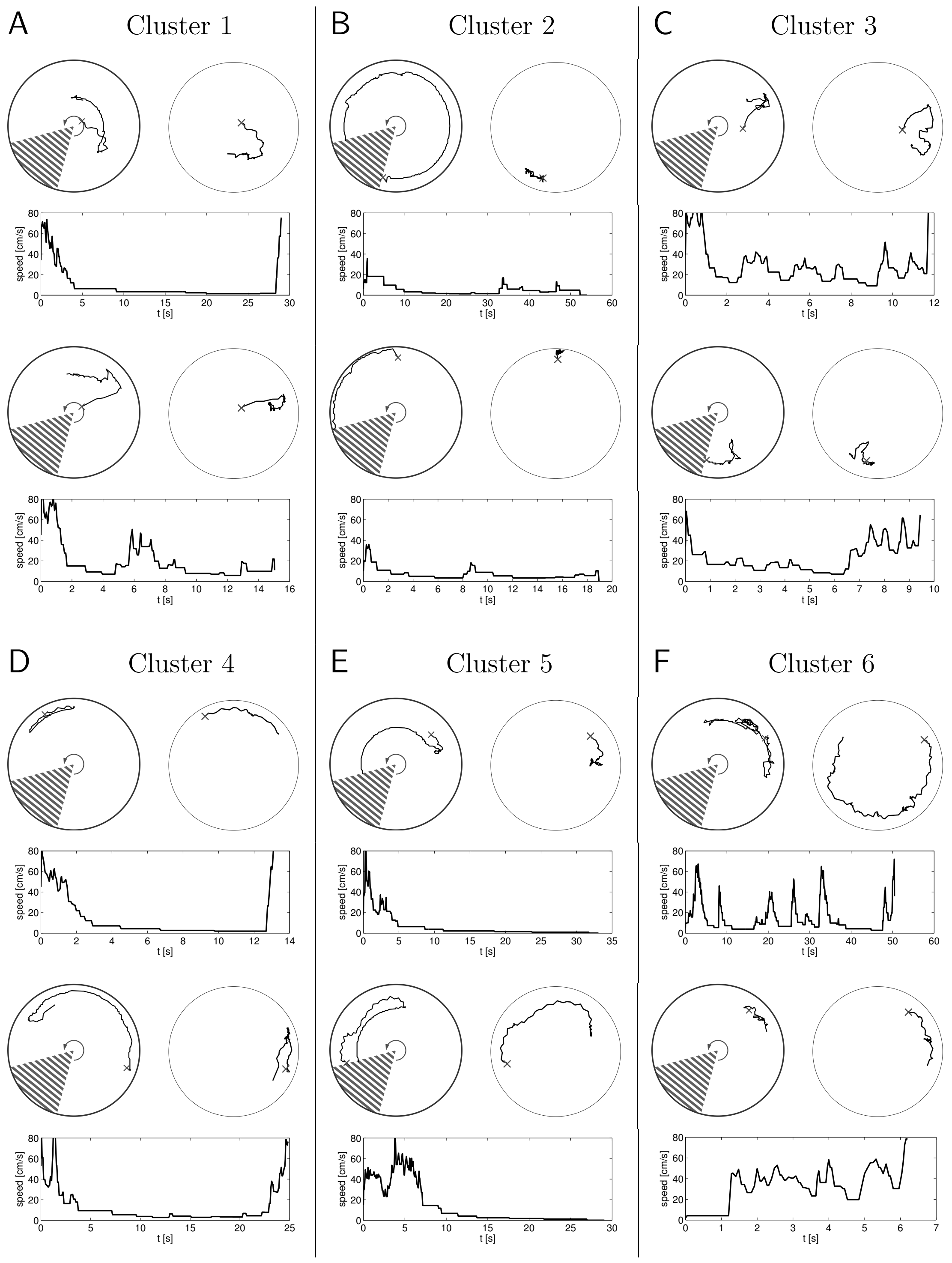


Figure 4.6: Example trajectory segments for the six resulting clusters. Two examples are shown for each cluster. Top left figures show the trajectory in the room (global) reference frames. Top right plots show the trajectory compensating for the rotation of the arena (arena reference frame). Lower plots show the movement speed of the rats. Crosses mark the start of the segments.

Class 1 Animals move inwards and explore the more central parts of the arena.

Class 2 Animals move very little around the arena and sit mostly in one position (bottom speed plots in Figure 4.6). Paths frequently end in the shock sector (4.3).

Class 3 Relatively chaotic paths concentrated on the right/lower half of the arena, on the right side and immediate vicinity of the shock sector.

Class 4 Paths focused above the shock sector. Animals frequently stand still for a while and then move contrary to the disk rotation direction as they get close to the shock sector.

Class 5 Animals move to a more central point in the arena then sit still ending frequently in the shock sector (Table 4.3).

Class 6 Paths focused on the extreme opposite of the shock sector.

4.4.2.2 Treated and control groups compared

Figure 4.7 shows the percentage of path segments per day from each cluster for both groups of animals. The differences between the groups were then checked for significance using a Friedman multi-factor test (Section 3.3.8). Resulting p-values for the Friedman test are shown in the plots for each cluster.

For 4 of the 6 clusters significant differences ($p \leq 0.05$) were found between the control and silver nanoparticles group. Animals in the treated group have a much higher tendency (Class 2) for standing still in the arena, ending in many cases in the shock sector. The same behavioural pattern is observed in Class 5 for the more central regions of the arena. The third pattern seen more often in the treated group are chaotic paths focused in the lower right section of the arena (Class 3). This type of behaviour is frequently associated with paths that originate in the shock sector.

Animals in the untreated control group, on the other hand, demonstrate a behavioural pattern in which animals sit still for a while but then move away from the shock sector as they approach it (Class 4). This shows that animals are aware of the location of the shock sector and use a strategy where they move as little as possible.

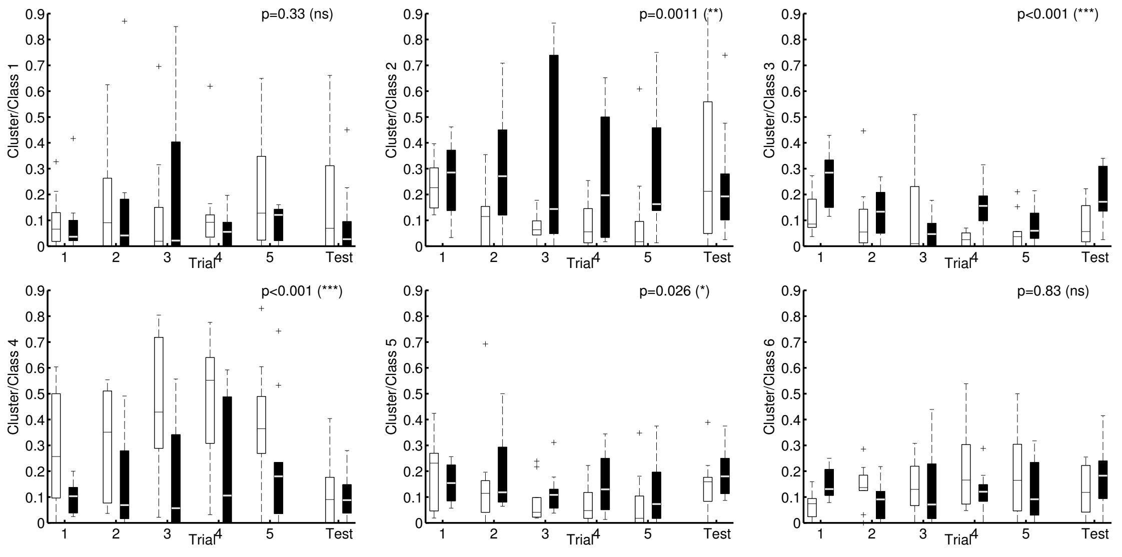


Figure 4.7: Distribution of segments of each cluster for control (white boxes) and treated (black boxes) groups of animals for a set of 5 training (with active shocks) and 1 test session (without shocks). Boxes represent the first and third quartiles of the data, lines the median, crosses outliers, and whiskers minimum and maximum values. A Friedman test (Section 3.3.8) was used to compare both groups of animals over all sessions; p-values are shown on the top right.

4.5 Discussion

The method presented in this chapter for the Allothetic Active Place Avoidance (AAPA) task relies on splitting the recorded trajectories of the animals in the arena, computing a set of features for each resulting trajectory segment, and then using a clustering algorithm to identify similar behavioural patterns in the data. This is a generalisation to the approach of Chapter 3, which analysed the behaviour of animals in the Morris Water Maze. There, however, the classes of behaviour were predefined and a partial set of pre-labelled data was used to classify the trajectories. This effectively led to a semi-automated or semi-supervised classification method. The method presented here, on the other hand, offers a completely unsupervised approach to the classification; behavioural classes of interest don't have to be defined beforehand and no manual labelling of data is necessary. Classes of behaviour are instead identified by the algorithm itself as a result of the output clusters. The fact that no manual labelling is necessary not only saves time but can also lead to fully automated classifiers.

As a case study the new method presented here was applied to a data set consisting of 20 animals. Half of the animals were treated with silver nanoparticles, the other half was the untreated control group. Although standard performance measures, such as the number of entrances to the shock sector within a session, already show a clear difference in performance between the two groups of animals, these differences become much more evident when looking at the different behavioural patterns of the two groups. The analysis shown here is able to highlight the differences in behaviour between both groups of animals. This is especially true at later trials, which shows that the spatial navigation and learning performance of the animals is heavily affected by the presence of silver nanoparticles in the organism. The method was also able to identify six types of distinct behaviours which, to the best of the author's knowledge, were never described in the literature before.

This chapter shows once again how machine learning algorithms can be applied to behavioural neuroscience and help to find patterns in data and interpret results. In this chapter the semi-supervised method developed for the MWM in the previous chapter was reduced to a fully unsupervised context and generalised to work with another completely different experimental setup. This shows that the method is general enough and can be extended further to other experimental setups. It is to be noted that the software tools developed here, which are also an extension of the tools used in Chapter 3, are freely available and can be used as basis for creating more sophisticated tools, or to generalise the method to other experimental setups. A brief description about the tools is given in Appendix A.

One possible point of criticism of the method that remains, and which was also already highlighted in Chapter 3, is that the classification method depends on relatively fine tuned features. In order to apply it to another experimental setup an appropriate set of features that measure geometrical aspects of the trajectories or their relative position relative to an objective (e.g. the escape platform in the MWM) or special sector (e.g. the shock sector in the place avoidance task,) has to be defined. In this chapter the problem was minimised to some degree by using a larger pool of features and by using feature extraction in the form of PCA to reduce the dimensionality and find a smaller set of features that can account for most of the data variability. Nevertheless, the problem of first having to design appropriate features for a given experiment remains. In order to fully overcome this limitation the possibility of defining abstract measures that can be more universally applied remains to be investigated. As mentioned previously, Korz (2006), for example, proposes a method in which the coordinates of the paths of animals in the MWM is used to define a new set of features. In his work PCA is also used to reduce the resulting high dimensional feature space and he shows that the first few principal components are enough to account for most of the variability in the data. A similar approach was considered but not adopted here since the trajectories were classified not only by their geometrical aspect but also by other factors, such as their relative position in the maze and the movement speed of the animals. However, a more general approach for defining and identifying features of interest could be investigated in a future work.

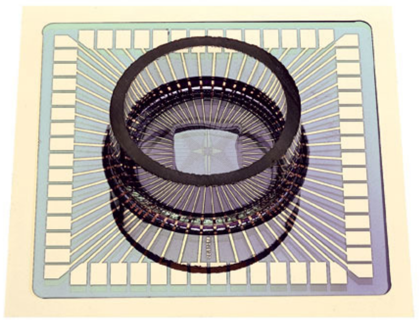
Summary This chapter focused once more on behavioural experiments in neuroscience and on how machine learning algorithms can be used to find behavioural patterns and to create automated analysis methods for behavioural experiments. The semi-supervised classification method that was developed in Chapter 3 for analysing swimming paths in the Morris Water Maze was generalised here to another important experimental paradigm in behavioural neuroscience, the Place Avoidance Task. The semi-automated method presented in Chapter 3 was reduced here to a fully unsupervised method, which does not require any labelled data and pre-defined classes of behaviour. Since producing labelled data is expensive, this is a major step forward from the point of view of the classification effort. The fact that the behavioural patterns of interest are extracted automatically from the data is also a major advantage. The next chapter will shift the attention from behavioural neuroscience to another important class of data in neuroscience: multi-electrode array data recordings. Analysing this type of data poses other types of challenges that will be considered there.

Chapter 5

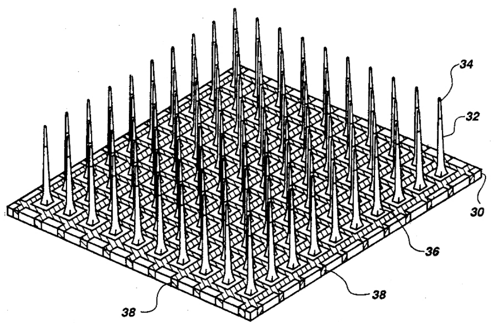
MEAs: Highly Scalable Parallel Processing of Extracellular Recordings

Multielectrode Arrays (MEAs) are devices that contain multiple sites through which neural signals can be recorded or neurons can be stimulated (Spira and Hai 2013) (Figure 5.1). They are a valuable tool for investigating the activity and information processing in neural ensembles since they allow multiple locations to be recorded simultaneously (Obien et al. 2015). They can be used for *in-vitro* or *in-vivo* recordings. In-vitro recordings can be used to record the activity of acute brain slices (Heuschkel et al. 2002; Oka et al. 1999), with recording times on the scale of hours, or cell cultures (Marom and Shahaf 2002; Pine 1980; Potter and DeMarse 2001), which can be recorded non-invasively for long periods of time (over a month). In-vivo recordings, on the other hand, use implanted electrodes in the brain to record the activity of a living organism (BeMent et al. 1986; Kovacs et al. 1992).

Technological advances in the construction of MEAs lead to collections of an increasing amount of data. For example, a simple calculation shows that recordings employing new large-scale MEAs with 4096 electrodes (Berdondini et al. 2009; Ferrea et al. 2012) with a sampling frequency of 7.7 kHz generate about 63 MiB/sec, (using a 16 bit analogue-digital converter), or almost 230 GiB of raw (uncompressed) data per recorded hour. It is expected that future technology advances will lead to an even larger amount of data being generated which has then to be processed and interpreted in order to draw conclusions (Mahmud and Vassanelli 2016).



(a) Multielectrode Array for in-vitro recordings commercialized by Multi-channel Systems MCS GmbH



(b) Utah in vivo multielectrode array. *Source*: Richard A. Normann - US Patent 5,215,088. Public domain.

Figure 5.1: Multielectrode arrays

In order to be able to make sense of the data, raw MEA recordings have to be first processed. Typical processing pipelines usually involve a filtering step followed by a spike detection and artifact removal algorithm (Mahmud, Pulizzi, et al. 2014). These pre-processing steps are then typically followed by a spike sorting algorithm (discussed in Section 5.4.1) and/or other post-processing steps, such as burst-detection in the case of cultured cells (Pelt, Wolters, et al. 2004).

Different software packages aiming to simplify and automate the processing and analysis of MEA recordings were proposed over the years (Egert et al. 2002; Mahmud, Pulizzi, et al. 2014; Wagenaar et al. 2005). These pioneering tools, however, were designed primarily with ease of use and not performance in mind, or were designed during a time were parallel hardware was not so prevalent as it is today. They therefore do not scale well on modern hardware architectures (e.g. multi-core CPUs with vector instruction sets or GPUs).

In this chapter a new MEA data processing tool, SpikeCL, is presented. It is designed with scalability and parallel hardware in mind from the beginning and offers substantial performance gains to previous tools on modern CPUs. The new tool is based on OpenCL (Open Computing Language), an industry standard for programming multi-core CPUs, GPUs, and dedicated accelerators.

The focus of the new tool is to optimise the performance-critical, pre-processing steps that deal with large amounts of data. The objective is not to create a new analysis framework, but rather to complement and speed-up existing ones. In order to maximise the reuse of existing software the processing steps performed here match closely the ones from one popular MEA data processing framework, QSpike tools (Mahmud, Pulizzi, et al. 2014), a previously released and freely available software package. The results from the pre-processing stages of the QSpike tools software package and SpikeCL are interchangeable so that, for example, the available report generating facilities provided by QSpike tools can be used without major modifications using the new tool for pre-processing.

The processing pipeline in the tool was built so that different data sources can be supported. Initially only offline data recordings are supported but the code can be easily adapted to process online MEA recordings (live data streams). Another possible extension to the tool is to add support for online spike sorting, or grouping spikes based on their shapes. Spike sorting is discussed in Section 5.4.1.

In what follows first the architecture of the SpikeCL tool will be discussed. This will then be followed by code benchmarks and an overview of possible extensions of the tool. The chapter closes with a discussion about the achieved results and a future outlook of MEA data processing tools.

5.1 Code

The software tool presented here, SpikeCL, was developed in C++ and is platform agnostic; the code was tested on Linux, Windows, and OSX systems. It is provided as a command line tool for manual processing of offline recordings (Multichannel Systems' MCD files are currently supported).

5.1.1 OpenCL

Modern CPUs consist of multiple cores and rely increasingly on SIMD (Single Instruction Multiple Data) extensions, such as SSE (Streaming SIMD Extensions) (Raman et al. 2000) and, more recently, AVX (Advanced Vector Extensions) (Firasta et al. 2010), to increase throughput. The basic idea behind SIMD is to apply each CPU instruction to multiple data streams at once (Esterie et al. 2012). This is in contrast to SISD (Single Instruction Single Data) architectures which apply each instruction to a single data stream (Figure 5.2). Besides multi-core CPUs other modern highly parallel hardware components, such as GPUs* (Graphical Processing Units) or Intel's MIC (Many Integrate Core) boards, available in the form of the Xeon Phi line of dedicated accelerators, are becoming increasingly widespread.

OpenCL is a widely used and supported framework for writing programs that target modern parallel hardware architectures (see Tompson and Schlachter 2012

*Modern GPUs can be used also for general, highly parallel, computing tasks

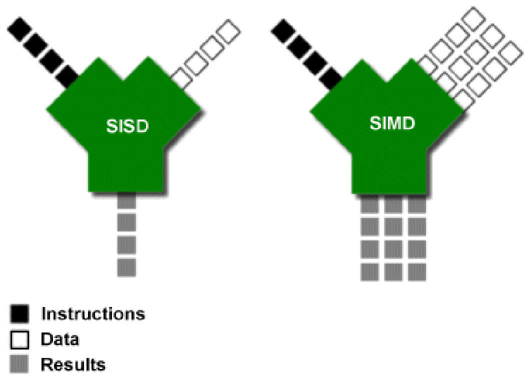


Figure 5.2: Single Instruction Single Data (SISD) and Single Instruction Multiple Data (SIMD) CPU architectures. SISD uses a single data stream per instruction. SIMD can apply one instruction to multiple data streams, leading to massive performance gains. *Source:* Ars Technica (www.arstechnica.com). Licensed under the Creative Commons License.

for a short introduction, Munshi 2012 for a detailed treatment). The main advantages of using OpenCL, besides being an industry standard, is that it vastly simplifies and abstracts the programming of parallel code that has to run in multiple hardware configurations. This is achieved by means of OpenCL kernels, which are just special functions, written in a subset of the C programming language, which are compiled at runtime. Because kernels are compiled at runtime, the code can be highly optimised for the specific hardware that will execute it. This can be, for example, a multi-core CPU with a dozen or or so cores, or a GPU with thousands of simplified computing cores. Both require very different types of optimization, that are handled by OpenCL at runtime. Figure 5.3 shows the general architecture of an OpenCL application.

The SpikeCL tool makes use of the dynamic nature of OpenCL code and generates custom kernels for performance critical functions. The custom kernels can be build for a specific set of parameters that are known only at runtime (e.g. the coefficients of the filtering function) and targeting the hardware architecture at hand (e.g. by using the maximum vector width supported by the CPU, allowing it to process multiple channels at a time). All data intensive processing functions in the SpikeCL tool, such as the data filtering and spike detection, were written in the form of custom OpenCL kernels for maximum performance. Listing 5.1 shows one OpenCL kernel for a data filtering step that was generated at runtime. Embedded in the code is information such as the filter coefficients and the supported hardware vector width (4 x 32 bit floats per CPU core in this case).

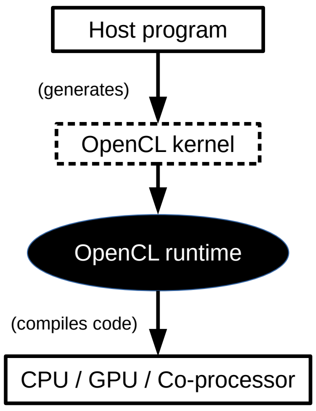


Figure 5.3: Overview of the OpenCL programming and runtime architecture. The host application generates the OpenCL kernels, or special functions, which are then compiled at runtime targeting the specific hardware on which the code is to be run (e.g. a multi-core CPU or a GPU). Because of the runtime compilation highly optimised code can be generated.

Listing 5.1: Example of OpenCL Kernel that is generated by the SpikeCL tool. Kernels are generated and compiled at runtime for an specific architecture and set of arguments. This particular code is one pass of a 2nd order bidirectional filter that was generated for a CPU that can process 4 channels per core at a time (therefore 4-vector floating points, `float4`, are used in the code). Filter coefficients are also embedded in the kernel, which makes the code highly efficient.

```
__kernel void filter_16_4_rev_f_f_16934953305227821146(
    __global float4* x, ulong n,
    __global float4* y, __global float4* z)
{
    long i = (long)get_global_id(0);
    long co = (long)get_global_offset(0);
    z[4*4 + i] = 0.;
    ulong j;
    ulong k;
    for(k = 0; k < n; ++k)
    {
        j = n - k - 1;
        y[j*4 + i - co] = (float)0.512646935729248*x[j*4 + (i - co)] + z[i];
        z[0*4 + i] = (float)-0.0038683462786876*x[j*4 + (i - co)] + z[1*4 + i]
                   - (float)-0.872973291418191*y[j*4 + i - co];
        z[1*4 + i] = (float)-1.01707478174515*x[j*4 + (i - co)] + z[2*4 + i]
                   - (float)-0.487313079047431*y[j*4 + i - co];
        z[2*4 + i] = (float)-0.0038683462786884*x[j*4 + (i - co)] + z[3*4 + i]
                   - (float)0.0993040356576583*y[j*4 + i - co];
        z[3*4 + i] = (float)0.51264693572925*x[j*4 + (i - co)] + z[4*4 + i]
                   - (float)0.309222050406897*y[j*4 + i - co];
    }
}
```

5.1.2 Block processing, MCD files

The tool has built in support for processing Multichannel Systems' MCD files. However, other file formats or data sources such as online recorded data, can be easily supported by adding a new data source type to the code (any source which provides a continuous data stream can be used).

The main processing steps of the code work on data blocks, which contain a chunk of multichannel recordings. These data blocks can be read and processed in parallel, which can speed up the processing considerably. The reason for the increased processing speed is large MCD files can be broken down into smaller blocks and the reading and processing of data can progress concurrently. It has to be noted, however, that using different block sizes can have an impact on the results as the results of the backward filtering step (see below) and the estimate of the spike detection threshold will vary slightly. Block processing is therefore optional and, as long as there is enough system memory available, MCD files can be processed as one block. The block size can also be set to a specific value via a command line parameter.

5.2 Processing workflow

Figure 5.4 shows the different data processing steps performed by the tool. Except for the generalised concept of input and output streams and block processing (described above), these steps are exactly the same as the ones performed by the original QSpike tools implementation in Matlab (Mahmud, Pulizzi, et al. 2014). When a single data block is used the results of both tools is exactly the same.

5.2.1 Filtering

The first step of the processing performs an (optional but active by default) band pass filter of the data using an IIR (Infinite Impulse Response) filter. By default a bi-directional (non-causal) 2nd order elliptic filter with a $400-3000\,\text{Hz}$ pass band is used. However, this can easily be changed by providing custom filter coefficients or a custom filter coefficient file with multiple sets of pre-computed filter coefficients. A non-causal (or zero-phase) filter, which requires a forward and a backward pass over the data, is chosen by default. A bidirectional filter is chosen in order to minimise spike shape distortion and time lags (Quian Quiroga 2009). Single pass filters can also be used with a one line code change.

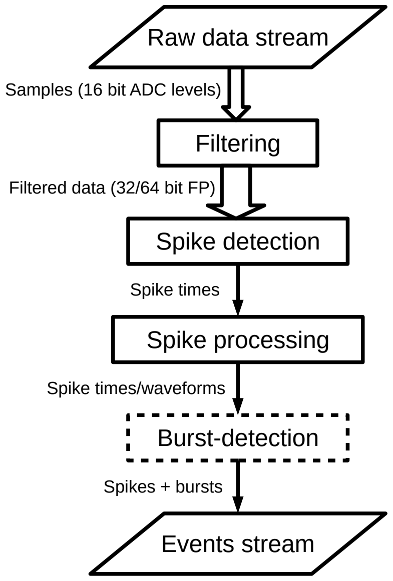


Figure 5.4: Overview of the data processing steps. Wider arrows symbolise steps where larger amounts of data have to be processed.

It should be noted that the filtering step also expands the input data from discrete ADC (Analogue Digital Converter) values (usually 16 bit integers) to 32 or 64 bit floating point values (depending on the compile time flags) in the first pass. The data conversion and filtering operations are combined to avoid an extra time consuming data copy operation.

5.2.2 Spike detection

The spike detection algorithm is based on a fixed threshold, t_{spk}, which is estimated from the background noise of the data (Quiroga et al. 2004), σ_n:

$$t_{spk} = K\sigma_n \tag{5.1}$$

where K is a constant and σ_n is computed from the following expression: :

$$\sigma_n = k\,\mathrm{MAD}(x) \tag{5.2}$$

where x is the (band pass) filtered signal and MAD is the *median absolute deviation*:

$$\mathrm{MAD} = \mathrm{median}\left\{|x - \mathrm{median}(x)|\right\} \tag{5.3}$$

In Equation 5.2 k is a scale factor, which can be computed from (Rousseeuw and Croux 1993):

$$k = 1/(\Phi^{-1}(3/4)) \approx 1.4826 \tag{5.4}$$

where Φ^{-1} is the quantile function for a normal distribution and the value 3/4 is chosen so that +/- 50% of the cumulative distribution function is covered.

If the standard deviation were used instead of the above expression (Equation 5.2), the spikes would have a large influence on the estimate of the noise. It was shown by Quiroga et al. (2004) that, under the assumption that the number of spikes is not too large, using the above estimate for the background noise using the median leads to better results. For the computation of the median a recursive binning algorithm is used (R. J. Tibshirani 2008).

5.2.3 Burst detection

The burst detection uses the method described in Van Pelt et al. (2005), which first bins the spikes in discrete time windows and then calculates the number of active electrodes (ones in which a spike was detected) at each time step. A burst is detected if the number of active electrodes crosses a fixed threshold.

5.2.4 Post processing

The code does not provide any new post-processing code, such as report generation, but rather makes use of the already available QSpike tools software stack. Only minor modifications had to be made for the QSpike post-processing code to be compatible with the output generated by the SpikeCL tool. As an example of post-processing output, Figure 5.5 shows a raster plot produced by QSpike tools.

5.3 Results

This section presents some benchmarks that were performed with the new tool.

5.3.1 System setup

Benchmarks were conducted on two different systems:

- AMD Ryzen 1700 @ 3.85 GHz desktop PC with 8/16 cores/hardware threads, 32 GB of RAM, and a Samsung 960 EVO NVMe SSD;

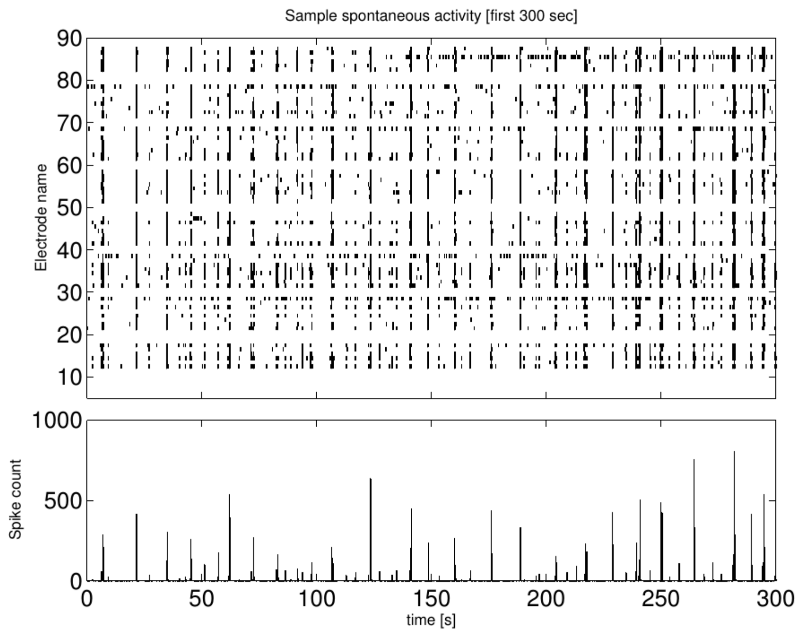


Figure 5.5: Example raster plot generated by the QSpike tools framework. *Top-plot*: Spike times for a 60-channel array. Each dot marks one spike. *Bottom plot*: Total number of spikes over all channels for a given time window.

- Intel i7-4600U (Haswell) @ 2.1 GHz laptop with 2/4 cores/hardware threads, 8 GB RAM, and a Samsung 830 SSD;

The operating system in both cases was Gentoo Linux 64 bit with Kernel version 4.13. For the AMD system the OpenCL driver version 14.41 was used. For the laptop the Intel OpenCL runtime version 4.4.0.117 was installed.

5.3.2 Performance scaling

Figure 5.6 shows how the main pre-processing operations, data filtering and spike detection, scale with different number of channels being processed in parallel. The number of parallel channels was always doubled, starting from 2, to allow for optimal code vectorization (SIMD instructions found in modern x86 CPUs work on floating point vectors of size 2, 4, 8 or 16[†]). As the graph shows, the filtering operation scales well and even beyond the number of available hardware threads. This can be explained by the fact that the filter code is highly vectorized since it consists only of arithmetic operations and no branching is involved. These operations can therefore be replaced by their SIMD counterpart (SSE or AVX

[†]Using the new AVX-512 instructions that are slowly being introduced by Intel and which can process up to 16 32-bit values at a time.

instructions) and with this process multiple channels at a time on a single hardware thread. Peak performance of the bi-directional filter used in the tests was on the order of 300 – 400 GiB/s, even when running on the relatively modest laptop used in the tests, showing that it does not represent a performance bottleneck.

The results in Figure 5.6 show that the spike detector does not scale as well as the data filter. This is due to the fact that it is much harder to vectorize the code of the spike detector because of the code branching involved. The branching arises from the conditional check for a threshold crossing, which signals that a spike was found. For this reason, only parts of the spike detection code, such as the background noise estimation, could be satisfactorily vectorized. However, as the results show, the spike detector does scale well up to the number of available hardware threads (16 or 4 for the two hardware configurations tested here). The achieved throughput of the spike detector is still multiple times more than the amount that even the most advanced MEAs currently available (Ferrea et al. 2012) can generate. The code can therefore be used as basis for an online processing tool than can run even on readily available and inexpensive hardware, such as a standard modern laptop.

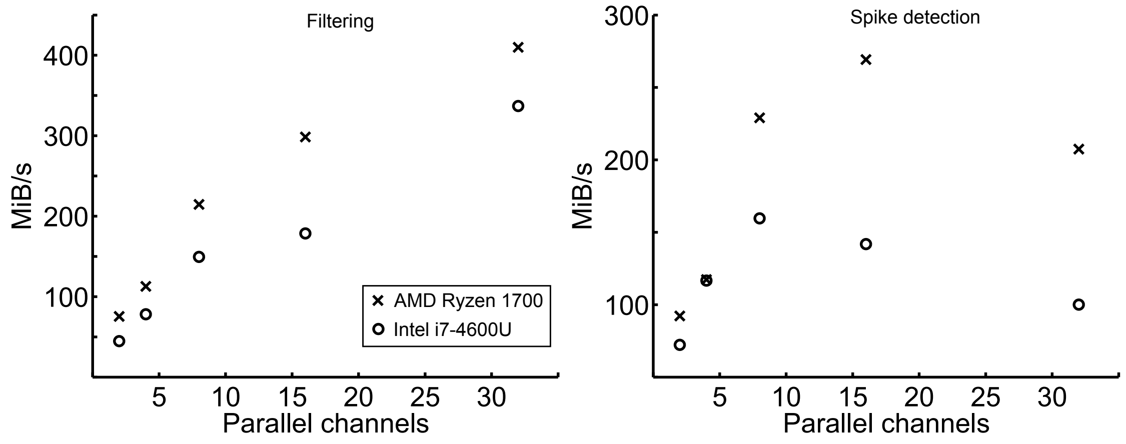


Figure 5.6: Performance scaling of filtering and spike-detection operations for different number of channels processed in parallel. Two different systems were tested: an AMD Ryzen 1700 desktop with 8/16 cores/threads (crosses) and an Intel Core i7 laptop with 2/4 cores/threads (circles). Details about the system configurations are given in Section 5.3.1. The results show that filtering shows close to linear scaling with the number of parallel channels (left panel), even beyond the number of physical cores/lanes due to the use of SIMD instructions. In the case of spike detection (right panel) the code cannot make full use of vector instructions due to branching in the code; the performance peaks therefore close to the point when all available hardware processing threads are exhausted.

5.3.3 Processing of offline data

Figure 5.7 compares the overall performance scaling, combining both filtering of the data and spike detection, when using different numbers of parallel channels in the processing. The results show good performance scaling up to a number of channels that is slightly higher than the number of available hardware threads. This is mostly due to the spike detector not scaling as well as the data filter (5.6).

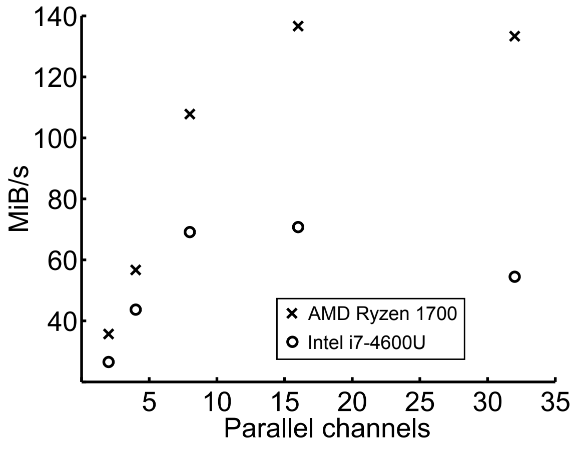


Figure 5.7: Combined performance scaling for filtering and spike detection operations for different number of data channels processed in parallel. Systems used: AMD Ryzen 1700 desktop with 8/16 cores/threads (crosses) and an Intel Core i7 laptop with 2/4 cores/threads (circles). Details about the system configurations are given in Section 5.3.1. Performance is again seen to scale well until the maximum number of hardware lanes is reached, when it flattens. This is due mostly to the spike detector code, which cannot be easily vectorized.

Figure 5.8 shows the time to process a 30 min recording with 60 channels for different block sizes. The results show that the performance improves slightly with decreasing block size. This is due to the fact that smaller blocks lead to increased levels of reading and processing parallelism. The rightmost point, with a block size as large as the file, means that reading and processing operations are sequential, as is the case in the original Matlab implementation. As described in the Section 5.1.2, the block size does have an impact on the results (Table 5.1) as the spike threshold with varying block sizes will vary slightly. Included in Figure 5.8 is also the time for processing the same file using the original implementation on the same AMD Ryzen 8-core machine. It shows that for smaller block sizes (300 seconds) the new code is approximately 8 times faster than the original one, a substantial improvement. Perhaps even more importantly, the memory requirements of the new code are much lower and the analysis can be run even on a laptop with 8 GB of RAM, whereas the original implementation requires multiple times more memory than that to process more than a couple of channels in parallel[‡]

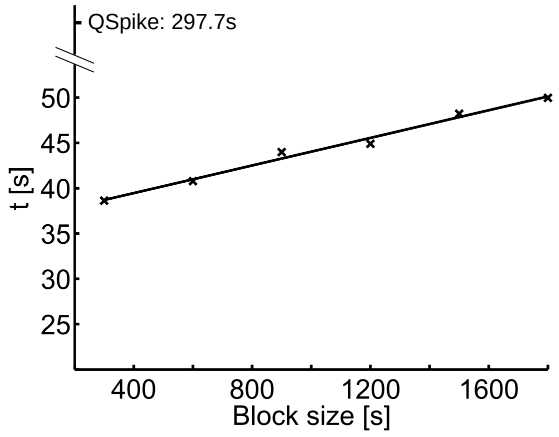


Figure 5.8: Time to process a 5.4 GiB MCD file with 60 data channels for different block sizes. Sampling frequency: 25 kHz (30 s recording). The AMD Ryzen system (Section 5.3.1) was used as a benchmark. The time to process the file on the same system using the original QSpike tools pre-processing stage (written in Matlab) is also indicated. The new pre-processing code is able to outperform the original QSpike code by a large margin; the results show also that the processing performance improves when using smaller block sizes because this allows for data extraction and processing to run in parallel (the rightmost point is the point where both operations are sequential - first all data is read and then processed, as is done in the original Matlab implementation). The line shows a linear regression of the data.

5.4 Future developments

This section describes briefly some possible improvements to the existing SpikeCL tool, which were planned but not finished at the time of this writing.

5.4.1 Spike sorting

Multielectrode Arrays capture the activity of multiple neurons at the same time, each electrode capturing the activity of possibly multiple neurons. In order to

[‡]Processing of the same file on a 12 core, 24 thread Intel Xeon system with 72 GB or RAM takes about 180 seconds. In order for the code to run on the AMD Ryzen machine with 32 GB of RAM used here, no more than 10 channels could be run at a time before running out of system memory.

Table 5.1: Number of detected spikes for different block sizes. File was the same one as in Figure 5.8. As can be seen the number of spikes can vary as much as 10% depending on the block size. This happens because the amount of available data used to compute the background noise is different in each case.

Block size [s]	Spikes detected
300	144,918
600	138,252
900	146,064
1200	151,362
1500	153,216
1800	154,908

assign spikes to individual neurons a process known as *spike sorting* has to be performed (Rey et al. 2015). It is a crucial step in studies of ensembles of neurons.

Spike sorting is based on the fact that spikes generated from two different neurons, even if from the same type of cell, show some differences in their shapes (Gold et al. 2006; Quian Quiroga 2009). The individual spike shape features characteristic of each cell can be used to assign spikes to individual cells in MEA recordings.

Spike sorting algorithms work by extracting a set of features from the spike shapes and separating them into groups according to their similarity. Clustering algorithms (Section 2.3) are well suited for this task and are usually chosen for this (Quiroga et al. 2004; Rey et al. 2015). In order to define the number of clusters, and therefore cells, manual (Gray et al. 1995), fully automated (Lewicki 1998), and semi-automated (Hazan et al. 2006) methods have been proposed over the years. A more complete overview of available spike sorting algorithms is given in Bestel et al. 2012.

From the coding perspective, extending the SpikeCL tool to also support spike sorting would be relatively straightforward, and would not require major modifications to the current code. The spike sorting would be just one more step to the processing pipeline (Figure 5.4) that takes as input the output from the spike detection phase (Figure 5.9). However, spike sorting in itself is a complex subject which would require some significant additional time. Scaling of the spike sorting algorithm would be highly dependent on the scaling of the clustering algorithm itself which is typically used for this task. Parallel clustering algorithms that scale well with the number of processors (e.g. Qing et al. 2009) have been proposed in the literature, however; by making use of such algorithms it should be possible to develop a spike sorting framework that scales well with the number of local processors/hardware threads.

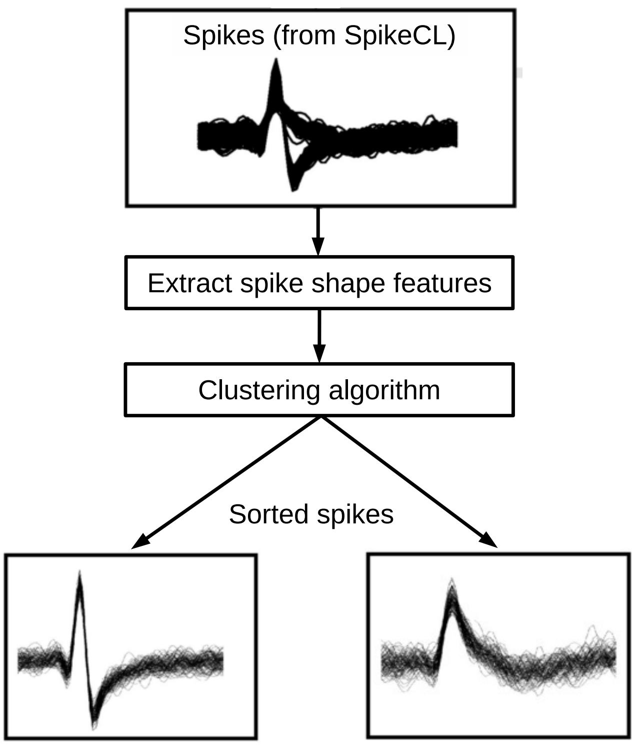


Figure 5.9: Spike sorting steps. Starting with the detected spikes the spike shape features are first extracted and then fed to a clustering algorithm, which assigns one cluster to each cell/source. Adapted from Quiroga et al. 2004.

5.4.2 Online processing

The SpikeCL tool currently processes only offline data but the code was written in a form that different sources can be easily integrated. It would be relatively straightforward to add support for processing a live data-stream, so that the intermediate raw data files don't need to be stored. A couple of challenges remain though:

i Filtering: the current filtering step uses a bi-directional (zero-phase) filter in order to minimise the distortion of the spike shapes. This would not be possible with a live recording but single-pass zero-phase filtering methods have been proposed in the literature (Powell and Chau 1991);

ii Spike sorting: the spike sorting algorithm (currently not implemented in SpikeCL) would have to make use of a fast, ideally parallel, clustering algorithm;

5.5 Discussion

The results presented in this chapter show that by employing modern programming technologies, such as OpenCL, highly efficient software that can target modern parallel hardware architectures can be developed. This is especially well suited for processing the large data sets generated by multi-electrode array recordings. The new tool introduced here demonstrates substantial performance improvements of data analysis compared to the original pre-processing stages of the QSpike tools (Figure 5.8). The main reason for the differences is that, while the latter focuses on process-level parallelism and processes data channels individually (one per CPU core), the new tool uses thread-level parallelism and code vectorization to process multiple channels per CPU core. Parallelisation does not pose major difficulties since data generated from multi-electrode recordings is inherently parallel; this is because individual data channels are independent from one another.

As technology advances there is a tendency for multi-site recordings with an increasingly large number of recording channels to become commonplace. At the same time modern hardware architectures are also becoming increasingly parallel. This means that software tools that can extract the maximum performance from state-of-the-art hardware will become increasingly important as well.

Although the focus here was to parallelize the code for modern CPUs, the code presented here would in theory work also on GPU (Graphical Processing Units) with minimal modification, since they are also supported by OpenCL. Tests showed, however, that at least for offline data where many gigabytes of recordings have to be processed at once, the data copying operation to the GPU is prohibitively expensive and has a drastic performance impact on the code; using system memory for this is also not ideal and impacts also performance. In order to overcome this problem would be a major task which would probably involve substantial changes to the processing pipeline. Furthermore, in order to extract maximum performance from the GPUs, which consist to up to a few thousand of micro-processing cores in modern hardware, most of the processing algorithms would have to be rewritten. For these reasons and because the code in its current form is able already to process recordings of MEAs with up to thousands of channels even on modest hardware that is readily available, the optimizations performed here were focused on CPUs only.

The focus here was on improving the performance of the pre-processing stages of MEA data analysis frameworks. This is a first step in being able to handle large amounts of MEA recordings, which are becoming both more common and also larger with new sophisticated MEAs with thousands of recording sites. This leaves space for future improvements, though, such as parallelisation of the spike sorting algorithm (Section 5.4.1), ideally using a highly parallel clustering algorithm (e.g. Qing et al. 2009). Another possible improvement is the ability to process the data in real-time (Section 5.4.2). This makes it not only unnecessary to allocate storage space for the raw data, which often has to be copied around for processing, but would also make it possible to create closed loop systems consisting of one or more MEAs that communicate by means of an exchange of spikes.

Summary This chapter shifted focus from behavioural data to another very important type of data found in neuroscience: Multielectrode Array (MEA) Extracellular recordings. One of the main challenges of analysing MEA recordings is processing and reducing the large amount of raw data collected by the arrays into data that can be more easily handled (spikes). Here a new tool for processing MEA recordings was introduced. By making use of a highly parallel programming model it offers substantial performance improvements compared to previous processing frameworks. The tool can serve as basis for creating more sophisticated MEA data processing systems that can, for example, interact with other systems in real-time. Future prospects and a conclusions about the work presented in the last three chapter will be considered in the next chapter.

Chapter 6

Conclusion

Modern technological advances allow us to now acquire vast amounts of data at multiple spatial and temporal scales. This is already true for many areas of science, for example in the analysis of sub-atomic particle collisions, or analysis of DNA sequencing data, and the same is also becoming a reality in neuroscience.

Collecting vast amounts of data at multiple scales is fundamental for understanding complex systems, such as the brain. However, it brings also new challenges because processing, extracting the relevant information, and drawing conclusions from massive and complex data sets can be very difficult.

Machine learning methods are especially well suited to find patterns in data and have been used with increasing frequency by scientists to analyse complex data sets. They have been successfully used to, for example, automatically detect features in images or to develop speech recognition applications.

The first and main objective of this thesis was to investigate how machine learning methods can be used to analyse behavioural neuroscience experiments. Behavioural data is fundamental in neuroscience because it can provide crucial information about how processes in the brain relate to behaviour and sensory input. For example, Chapter 3 of this thesis focused on one of the most commonly used rodent experiments in behavioral neuroscience, the Morris Water Maze (MWM). The Morris Water Maze is a navigation task in which animals are free to move within a circular arena and have to find a hidden platform to escape the maze. Because of the complex and highly variable movement paths of the animals, analysis of MWM experiments poses some challenges. Typical analysis methods of MWM focus on the evaluation of single performance measures, such as the time that the animals take to find the escape the platform (known as the escape latency) or the total distance moved until the platform is found. These simple measures are useful for initial performance evaluations but are also quite limiting because they do not provide any information about the behaviour of the animals.

Simple measures for behavioural experiments, like the MWM, cannot capture the multitude of behaviour patterns displayed by the animals. For this reason in many cases they are also not able to detect differences in behaviour between animals. This was the case, for example, for a set of experiments that investigated the effect of peripubertal stress on the performance of rodents performing the MWM task. Although the animals were submitted to a strong manipulation and displayed some evident differences in behaviour, such as a much higher movement speed, standard MWM performance measures, such as the escape latency, didn't show any differences between these animals and the control group. This was paradoxical to some extent because the higher movement speed would suggest that animals would be able to find the escape platform faster, but this was not the case. The failure to find quantifiable differences in this experiment was motivation for investigating other analysis approaches.

Categorisation methods of animal movement tracks have long been suggested as a better approach for analysing behavioural data. By assigning each track to a pre-defined class of behaviour many different path measures are effectively combined and taken into account. Categorisation approaches have been previously successfully applied to MWM experiments (D. Wolfer and H. Lipp 2000) but they usually also have some drawbacks, such as an increased effort to classify the data. In order to overcome this problem, automatic classification approaches have been proposed in the literature. One example is the work of Graziano and collaborators (Graziano et al. 2003), which combined over two dozen different path measures to create an automatic classifier for MWM movement tracks.

Although a clear improvement over simple performance measures, categorisation methods proposed previously in the literature had one major limitation: they are usually applicable only to large data sets. This is because each movement track is assigned to a single class of behaviour but animals usually frequently change their behaviour within a trial. This makes it difficult to map each track to a single class of behaviour and statistical significant results can be obtained only for sufficiently large data sets. This was not the case for the set of experiments considered here, however, which consisted of slightly more than 300 movement paths.

Chapter 3 of this thesis presented an improved classification method for MWM experiments. Compared to previous classification methods animal movement tracks were not assigned to a single class, but rather to multiple classes of behaviour. This was made possible by classifying not the complete trajectories of the animals, but multiple segments of the same path instead. This made it possible to detect changes in the behaviour of animals within a single trial, something that could not be done with previous approaches. The classification of the path segments itself was done with the help of a semi-supervised clustering algorithm, which can make use of a partial set of labelled segments to guide the clustering process. A semi-supervised method that requires only a partial set of labels to classify the data was chosen because labelling all segments (up to 30 thousand for the data set considered here) would be very time consuming. Moreover, paths were segmented so that segments had a significant overlap between them, so that there was no need to classify all of the segments.

The novel classification method for the MWM developed in Chapter 3 was successfully applied to the data set considered here, where previous methods failed. By focusing on path segments instead of complete movement paths and by mapping the segments to eight different behavioural classes, it was shown that the animals induced to stress tended to adopt less efficient exploration strategies than the control group. This explains why they perform the same as the control group even though they move faster.

The classification for the MWM data was done with the help of custom software tools that allow to classify the segments interactively, that is, by gradually labelling segments and evaluating the results. These tools have been made freely available.

The analysis method developed for the MWM showed how machine learning methods, in this case in the form of a semi-supervised clustering algorithm, can be used to create more sophisticated data analysis methods that are able to deal with complex data sets. Chapter 4 of this thesis focused on another type of behavioural experiments, the Active Allothetic Place Avoidance task. The objective was to investigate if the method developed for the MWM could be extended to other experimental setups.

The Active Allothetic Place Avoidance (AAPA) task is an experimental setup in which rodents are placed in a circular arena and have to learn to avoid a sector where they otherwise receive an electrical shock. Because the arena is rotating but the shock sector is kept fixed in space, animals have to constantly move around the arena to avoid the shocks. As was the case for the MWM, previous analysis methods for the AAPA task were based on single performance measures only. These include, for example, measures such as the total number of shocks received in a trial, or the maximum time interval for which animals are able to avoid the shock sector. As in the case of the MWM these simple measures offer a useful quantification of the performance of animals but give little information about their behaviour.

In Chapter 4 of this thesis a new analysis method for AAPA experiments was introduced. The basic methodology was the same as the one applied to the MWM, and was based on segmenting the animal tracks and then classifying the segments. However, compared to the MWM the method was extended in two major ways. First, an unsupervised clustering algorithm, instead of a semi-supervised one, was used. That is, no manual labelling of data was necessary, which greatly reduced the classification effort. The second difference compared to the method developed for the MWM is that behavioural classes were not pre-defined but rather identified by the clustering algorithm itself. Clustering algorithms are ideally suited for finding patterns in data and were here therefore used to find the stereotypical classes of behaviour.

The method developed for the AAPA task in Chapter 4 was applied to a set of experiments that analysed the impact of silver nano-particles, commonly found in cosmetics, on the behaviour of animals or, more specially, on their spatial memory and navigation skills. The analysis results were able to detect six distinguishing classes of behaviour that, to the best of this author's knowledge, were never previously introduced in the literature. The results showed that animals treated with silver nano-particles displayed remarkably different patterns of behaviour compared to the control group. Standard performance measures were in this case already able to show clear differences in the performance of both groups, but the new classification approach proposed here was able to make this more explicit and relate it directly to differences in behaviour.

Chapter 4 showed once more how classification methods based on machine learning algorithms can be useful for analysing behavioural neuroscience experiments. Machine learning methods can be used to develop analysis methods that are able to find patterns of behaviour in complex data sets. The successful application of the methods developed in chapters 3 and 4 suggest that the same principles can be extended to other behavioral experiments.

In order be able to cope with large and complex data sets, now so ubiquitous in science, not only more sophisticated but also highly scalable analysis methods have to be developed. Computer hardware in the last few decades showed a shift from serial to increasingly parallel processing architectures. Multi-core CPUs supporting advanced vector instruction sets and GPUs with thousands of microprocessing cores are now found everywhere. In order to develop highly scalable algorithms that are able to deal with large amounts of data, analysis tools that are able to make full use of modern parallel hardware architectures have therefore to be developed.

The second objective of this thesis was to investigate how modern programming paradigms can be used to develop highly scalable analysis tools. The focus here was not on behavioural but on electrophysiological data, another fundamental type of data commonly found in neuroscience. Chapter 5 looked at analysis methods of data from multielectrode arrays (MEAs), which can record the extracellular activity of many thousands of neurons over extended periods of time. The amount of data that is recorded by MEAs can be substantial and this data has to be first processed in order to transform it from raw electrical signals into more useful information, such as the spike trains at each electrode. This gives rise to some formidable data processing challenges.

Chapter 5 presented a new highly parallel tool for processing MEA data recordings. It showed that, by making use of modern computing environments and tools, such as OpenCL, which can be used to compile code optimised for the hardware at hand at runtime, highly scalable software can be developed. The new data analysis tool developed here, SpikeCL, includes processing steps such as data filtering and spike detection and is able to reduce large MEA recordings to a set of spikes, greatly reducing the storage space requirements. Benchmarks showed that the tool scales well for multi-core CPUs with vectorization units and that it is able to outperform previous analysis tools by almost one order of magnitude with much lower memory requirements at the same time. The throughput achieved by the tool is sufficient for online data processing (where the data from a recording session is processed in real time) from a modern MEA with thousands of electrodes even on a modest laptop.

The MEA data analysis tool developed in Chapter 5 can serve as basis for more sophisticated analysis frameworks. For example, a processing step for reconstructing the spike trains from each cell, a process known as spike sorting (Section 5.4.1), can be added. Spike sorting methods are once more typically based on machine learning methods, such as data clustering. This shows again the importance of machine learning methods in modern data analysis.

In the coming years, as more sophisticated and sensitive instruments and electronics become available, the amount of data that is collected will increase exponentially. This is true for practically every field of science, from physics and astronomy to genetics and neuroscience. This means that new analysis methods for large data sets will become increasingly relevant and will become a fundamental part of everyday scientific research.

This work focused on two very different, but equally important, types of data in neuroscience: behavioral and electrophysiological. Analysis of each one of these presents its own set of challenges. For the case of behavioural data the results presented here showed that machine learning methods can be used to develop more sophisticated methods for analysing these complex data sets. For the case of electrophysiological data it was shown how recent advances in hardware architecture can be exploited to develop highly scalable data processing tools. Moving forward and in order to analyse the complex and large data sets of the future both of these objectives, of highly scalable and sophisticated tools that make use of the lastest developments in machine learning, will have to be combined.

Appendices

Appendix A

Software tools: behavioural data analysis with Machine Learning

This Section gives a brief overview of the software tools that were developed to classify the Morris Water Maze swimming paths (Chapter 3) and tracks in the Place Avoidance Task (Chapter 4).

The code was written in Matlab (Mathworks, Inc.) and consisted of a Graphical User Interface (GUI) and a series of reusable functions and classes for aiding in the classification process [*].

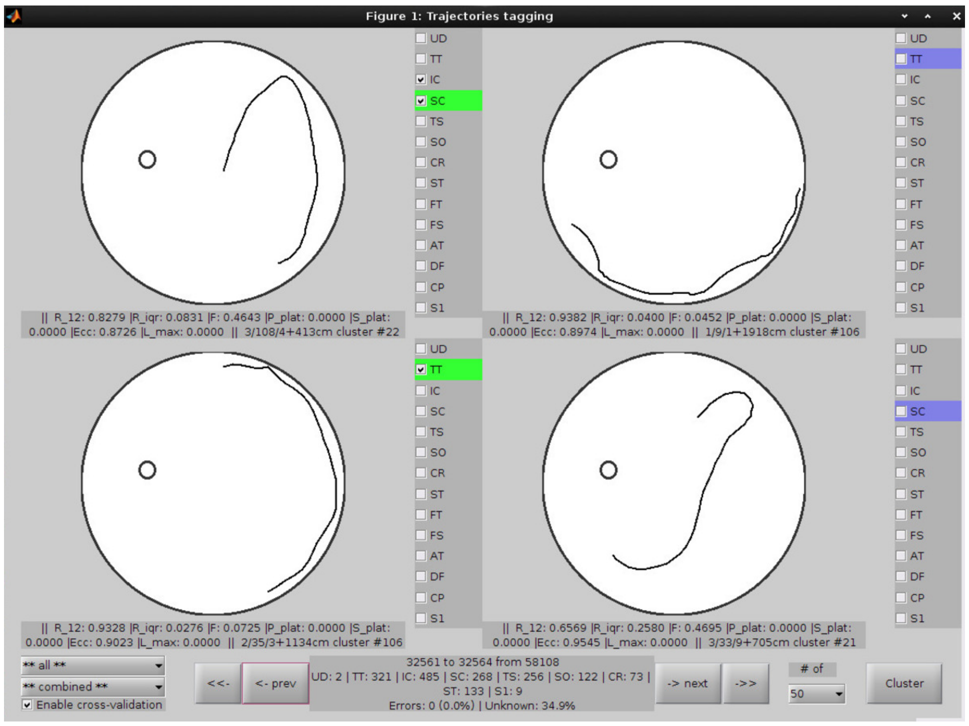
A.1 Graphical User Interface

The main Graphical User Interface (GUI) used for classifying trajectoy segments is shown in Figure A.1) for the case of the Morris Water Maze data set.

The main windows of the GUI makes it possible to interactively browse through the available segments and to assign labels to it (multiple labels can be assigned to each segment). The labels are saved to a file and can be reloaded on a future session, so that work can be resumed later. The labels are used to define the must-link / cannot-link constraints between the segments. The GUI shows also the computed feature values for each segment and allows to sort the segments according to individual feature values or to a combination of all features (using a distance function).

The objective of the main GUI window was to make it possible to not only label the segments but also to visualize the classification results. The data clustering

[*]The code is freely available and can be downloaded from GitHub: https://github.com/RodentDataAnalytics/roda



|| R_12: 0.8279 |R_iqr: 0.0831 |F: 0.4643 |P_plat: 0.0000 |S_plat: 0.0000 |Ecc: 0.8726 |L_max: 0.0000 || 3/108/4+413cm cluster #22

|| R_12: 0.9382 |R_iqr: 0.0400 |F: 0.0452 |P_plat: 0.0000 |S_plat: 0.0000 |Ecc: 0.8974 |L_max: 0.0000 || 1/9/1+1918cm cluster #106

|| R_12: 0.9328 |R_iqr: 0.0276 |F: 0.0725 |P_plat: 0.0000 |S_plat: 0.0000 |Ecc: 0.9023 |L_max: 0.0000 || 2/35/3+1134cm cluster #106

|| R_12: 0.6569 |R_iqr: 0.2580 |F: 0.4695 |P_plat: 0.0000 |S_plat: 0.0000 |Ecc: 0.9545 |L_max: 0.0000 || 3/33/9+705cm cluster #21

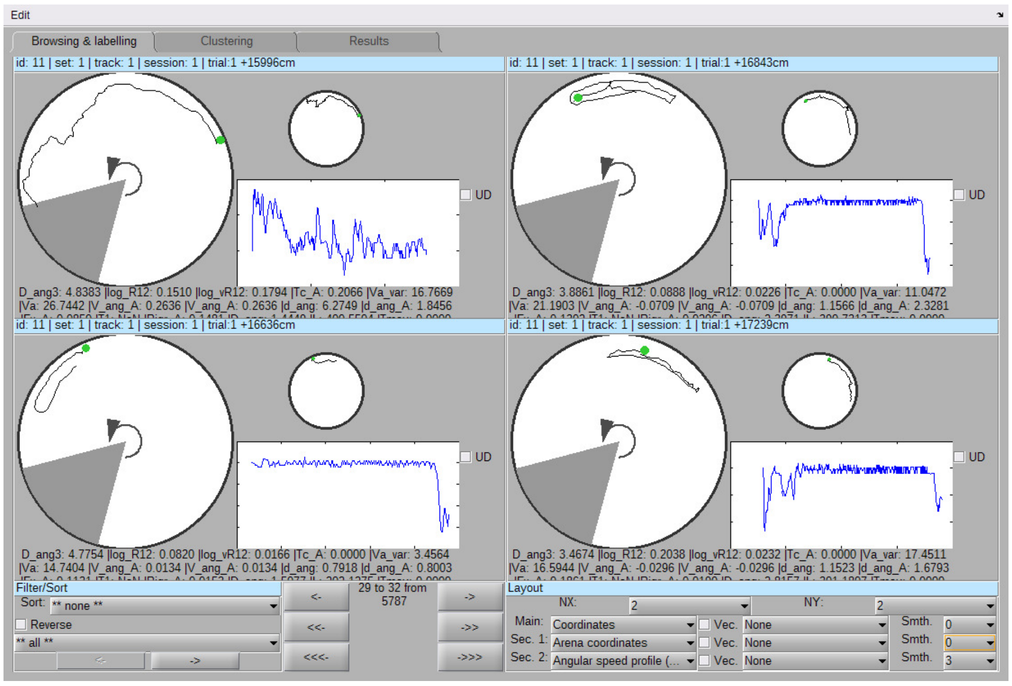
32561 to 32564 from 58108
UD: 2 | TT: 321 | IC: 485 | SC: 268 | TS: 256 | SO: 122 | CR: 73 | ST: 133 | S1: 9
Errors: 0 (0.0%) | Unknown: 34.9%

Figure A.1: Graphical User Interface used to assign labels to segments and visualize the classification results

algorithm for a given number of clusters can be started from the same window. The generated clusters can then be individually looked at by using the provided data filters. Pre-defined filters for identifying segments lying on cluster boundaries or isolated segments (i.e. ones for with a label was not assigned and which do not overlap with any other labelled segments); this makes it possible to also identify segments for which labels should be assigned in order to improve the classification. Other statistics of the current classification, such as the percentage of classification errors (if a cross-validation is enabled) and number of unclassified segments are also immediately visible in the GUI. This all makes it possible to dynamically add labels and almost immediately see the classification results.

For the Place Avoidance Task work (Chapter 4) the GUI was further extended as to also include user definable secondary visualizations / plots for each segment (Figure A.2). This was done in order to be able to show the segment both in the world and arena (rotating) reference frames and other information such as the movement speed. For the Place Avoidance Task work no labelling was done, the classification of the segments was completely unsupervised.

Additional windows make it possible to also graphically compare the distributions of the features between different groups of segments (Figure A.3) and visu-



id: 11 | set: 1 | track: 1 | session: 1 | trial:1 +15996cm

D_ang3: 4.8383 |log_R12: 0.1510 |log_vR12: 0.1794 |Tc_A: 0.2066 |Va_var: 16.7669
|Va: 26.7442 |V_ang_A: 0.2636 |V_ang_A: 0.2636 |d_ang: 6.2749 |d_ang_A: 1.8456

id: 11 | set: 1 | track: 1 | session: 1 | trial:1 +16843cm

D_ang3: 3.8861 |log_R12: 0.0888 |log_vR12: 0.0226 |Tc_A: 0.0000 |Va_var: 11.0472
|Va: 21.1903 |V_ang_A: -0.0709 |V_ang_A: -0.0709 |d_ang: 1.1566 |d_ang_A: 2.3281

id: 11 | set: 1 | track: 1 | session: 1 | trial:1 +16636cm

D_ang3: 4.7754 |log_R12: 0.0820 |log_vR12: 0.0166 |Tc_A: 0.0000 |Va_var: 3.4564
|Va: 14.7404 |V_ang_A: 0.0134 |V_ang_A: 0.0134 |d_ang: 0.7918 |d_ang_A: 0.8003

id: 11 | set: 1 | track: 1 | session: 1 | trial:1 +17239cm

D_ang3: 3.4674 |log_R12: 0.2038 |log_vR12: 0.0232 |Tc_A: 0.0000 |Va_var: 17.4511
|Va: 16.5944 |V_ang_A: -0.0296 |V_ang_A: -0.0296 |d_ang: 1.1523 |d_ang_A: 1.6793

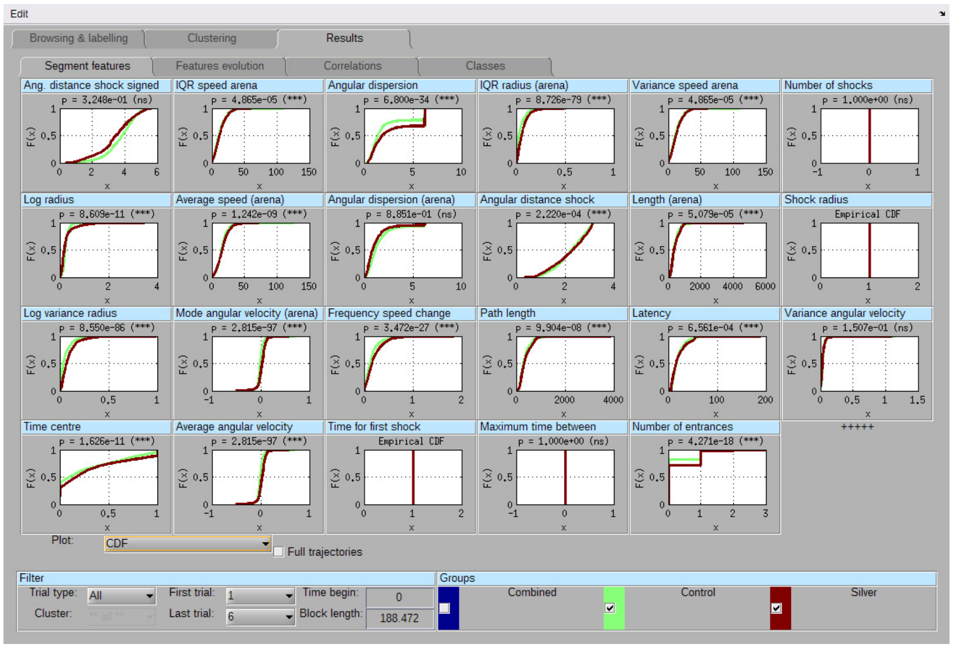
Filter/Sort
Sort: ** none **
Reverse
** all **

29 to 32 from 5787

Layout
NX: 2 NY: 2
Main: Coordinates Vec. None Smth. 0
Sec. 1: Arena coordinates Vec. None Smth. 0
Sec. 2: Angular speed profile (... Vec. None Smth. 3

Figure A.2: Graphical User Interface showing the Place Avoidance Task data with multiple visualizations per segment (Arena and Room coordintate frames; movement speed plot).

alize other results, such as the correlations between different experimental groups with the data clusters (Figure A.4).



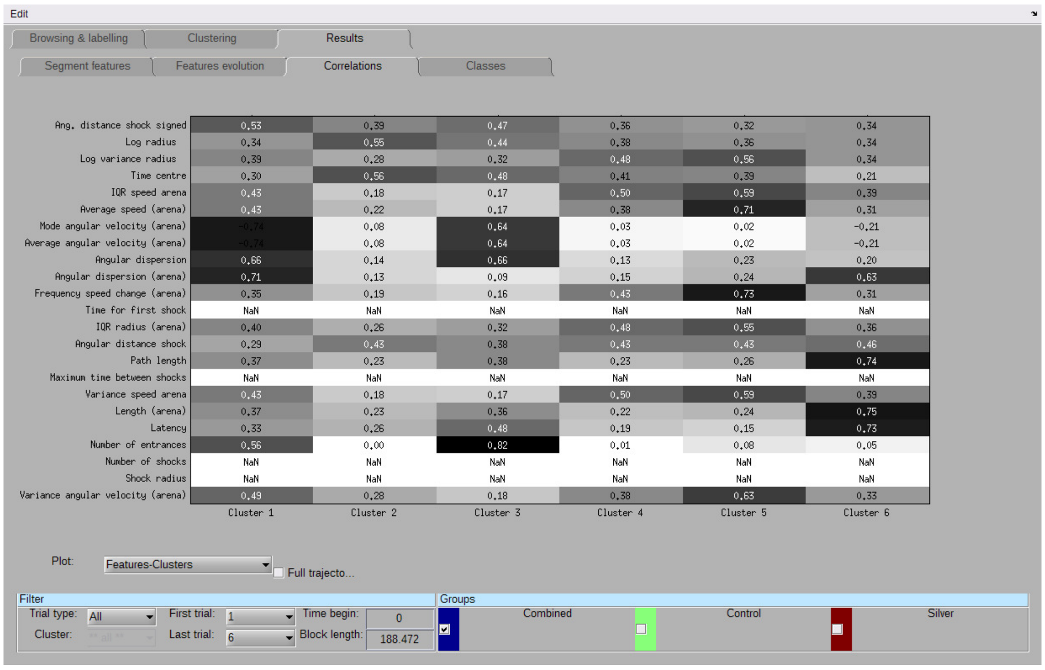
Ang. distance shock signed — p = 3.248e-01 (ns)
IQR speed arena — p = 4.865e-05 (***)
Angular dispersion — p = 6.800e-34 (***)
IQR radius (arena) — p = 8.726e-79 (***)
Variance speed arena — p = 4.865e-05 (***)
Number of shocks — p = 1.000e+00 (ns)
Log radius — p = 8.609e-11 (***)
Average speed (arena) — p = 1.242e-09 (***)
Angular dispersion (arena) — p = 8.851e-01 (ns)
Angular distance shock — p = 2.220e-04 (***)
Length (arena) — p = 5.079e-05 (***)
Shock radius — Empirical CDF
Log variance radius — p = 8.550e-86 (***)
Mode angular velocity (arena) — p = 2.815e-97 (***)
Frequency speed change — p = 3.472e-27 (***)
Path length — p = 9.904e-08 (***)
Latency — p = 6.561e-04 (***)
Variance angular velocity — p = 1.507e-01 (ns)
Time centre — p = 1.626e-11 (***)
Average angular velocity — p = 2.815e-97 (***)
Time for first shock — Empirical CDF
Maximum time between — p = 1.000e+00 (ns)
Number of entrances — p = 4.271e-18 (***)

Plot: CDF
Full trajectories

Filter
Trial type: All
Cluster:
First trial: 1
Last trial: 6
Time begin: 0
Block length: 188.472

Groups
Combined
Control ✓
Silver ✓

Figure A.3: Comparing feature values distributions among different experimental groups



	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Ang. distance shock signed	0.53	0.39	0.47	0.36	0.32	0.34
Log radius	0.34	0.55	0.44	0.38	0.36	0.34
Log variance radius	0.39	0.28	0.32	0.48	0.56	0.34
Time centre	0.30	0.56	0.48	0.41	0.39	0.21
IQR speed arena	0.43	0.18	0.17	0.50	0.59	0.39
Average speed (arena)	0.43	0.22	0.17	0.38	0.71	0.31
Mode angular velocity (arena)	-0.74	0.08	0.64	0.03	0.02	-0.21
Average angular velocity (arena)	-0.74	0.08	0.64	0.03	0.02	-0.21
Angular dispersion	0.66	0.14	0.66	0.13	0.23	0.20
Angular dispersion (arena)	0.71	0.13	0.09	0.15	0.24	0.63
Frequency speed change (arena)	0.35	0.19	0.16	0.43	0.73	0.31
Time for first shock	NaN	NaN	NaN	NaN	NaN	NaN
IQR radius (arena)	0.40	0.26	0.32	0.48	0.55	0.36
Angular distance shock	0.29	0.43	0.38	0.43	0.43	0.46
Path length	0.37	0.23	0.38	0.23	0.26	0.74
Maximum time between shocks	NaN	NaN	NaN	NaN	NaN	NaN
Variance speed arena	0.43	0.18	0.17	0.50	0.59	0.39
Length (arena)	0.37	0.23	0.36	0.22	0.24	0.75
Latency	0.33	0.26	0.48	0.19	0.15	0.73
Number of entrances	0.56	0.00	0.82	0.01	0.08	0.05
Number of shocks	NaN	NaN	NaN	NaN	NaN	NaN
Shock radius	NaN	NaN	NaN	NaN	NaN	NaN
Variance angular velocity (arena)	0.49	0.28	0.18	0.38	0.63	0.33

Plot: Features-Clusters ☐ Full trajecto...

Filter
Trial type: All First trial: 1 Time begin: 0
Cluster: Last trial: 6 Block length: 188.472

Groups
☑ Combined ☐ Control ☐ Silver

Figure A.4: Measuring clusters and features correlations

A.2 Code features and configuration

The software tools developed here include not only the series of graphical user interfaces just described but also a series of functions that can be used to perform various tasks such as importing data sets, segment trajectories, compute a series of features, define constraints from a set of labels, run a clustering algorithm, and evaluate the quality of the results, among others.

The complete set of functions was written using the Object Oriented Programming (OOP) model and with the intent to make most of the code reusable. Almost all functionality can be customized through a central configuration object, which is shared among all other task specific objects. In order to define a new configuration for a different experiment or data set most of the time only a new configuration object has to be defined. This object should inherit from an existing base one so that only the new properties and functions specific to the experiment at hand have to be defined. More information about the customizing the code base can be found in the code base's main page on GitHub (https://github.com/RodentDataAnalytics/roda).

Appendix B

Data calibration: Morris Water Maze recordings

Swimming paths of rats were recorded using an object tracking software (EthoVision Noldus et al. 2001, version 3.1). Due to the relatively close range of the camera and the lens system used the recorded trajectories have to be first calibrated.

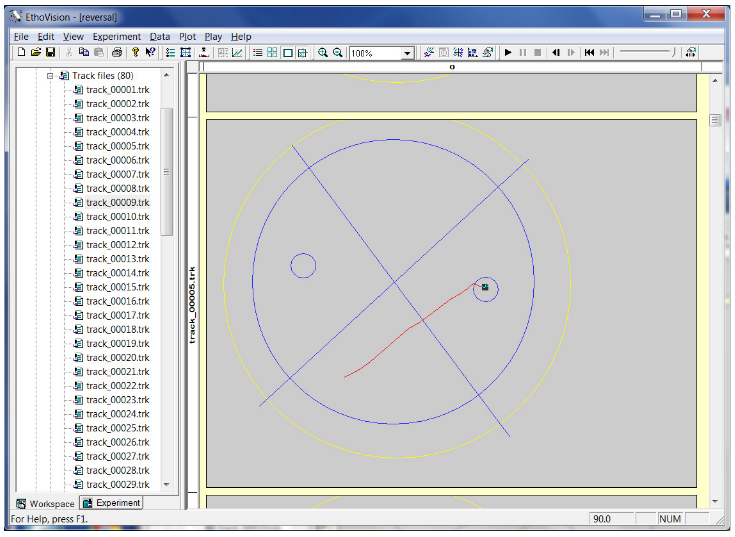


Figure B.1: Snapshot from the object tracking software (EthoVision) showing one trajectory segment. Current position of the animal is shown by the black square. The position of the black square relative to the yellow circle was used to automatically estimate the current position of the animal (using image processing algorithms).

The recorded trajectories were calibrated by using the trajectories as shown on screen in the tracking software, which has its own built-in calibration method, as reference. To extract the reference points from the software its playback capability, which replay a swimming path step by step, was used; screenshots saved to files included the recording time at which they were taken in their name. One example screenshot is shown in Figure B.1. The current position of the animal is shown as a black (dark-grey) square; this feature was used to automatically detect the position from the screenshots using imaging processing routines. As reference the outer (yellow) circle was used. The so extracted coordinates were then compared to the ones exported from the software, using the sample time information as shown in the screenshots (seen on the bottom right).

The pairs of real and exported coordinates were then used to compute a pair of error functions for x and y directions. These functions used a linear interpolation of the differences for estimating the correction for the given coordinate along the trajectory.

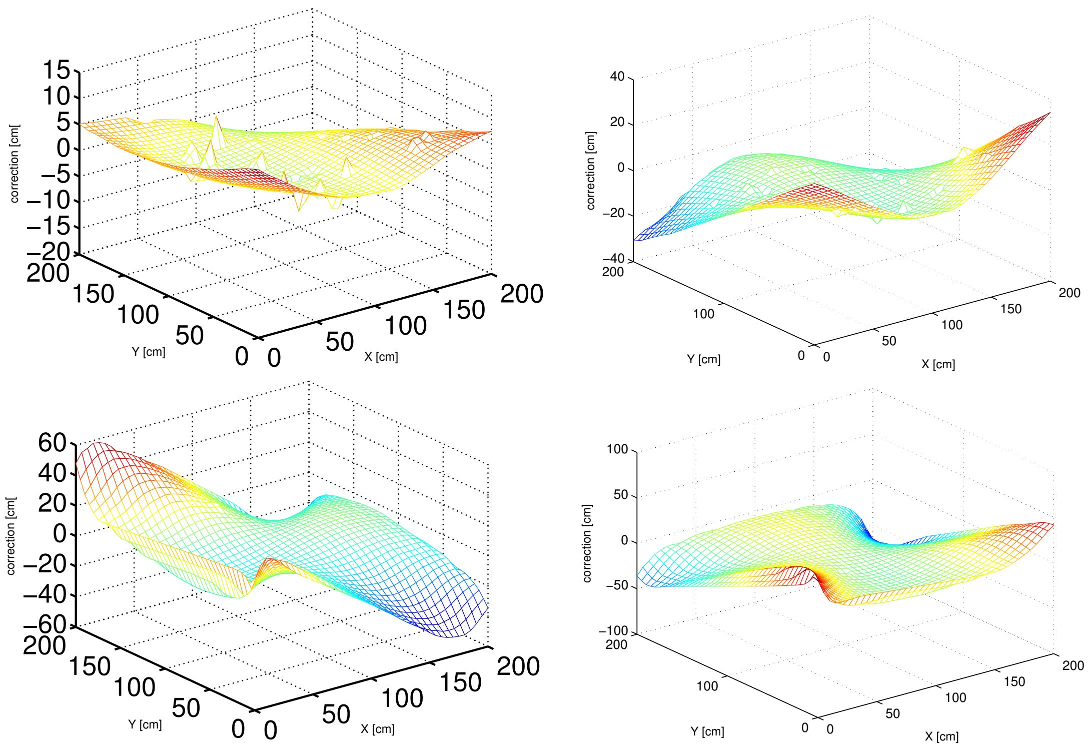


Figure B.2: Calibration functions. *Left:* x axis correction. *Right:* y axis correction.

$$d_x(x) = \delta x_i + (\frac{x - x_i}{x_{i+1} - x_i})(\delta x_{i+1} - \delta x_i), \quad x_i \leq x < x_{i+1} \tag{B.1}$$

$$d_y(x) = \delta y_i + (\frac{y - y_i}{y_{i+1} - y_i})(\delta y_{i+1} - \delta y_i), \quad y_i \leq y < y_{i+1} \tag{B.2}$$

where δx_i and δy_i, $i = 1, 2, ..., N$, are the sorted differences found between the N extracted points from the trajectories and the respective exported coordinates. Figure B.2 shows the two sets of error functions used to calibrate the data at hand.

The error functions were then used to correct the complete set of trajectories. Figure B.3 shows an example of a distorted trajectory as exported by the tracking software and the same trajectory after applying the correction.

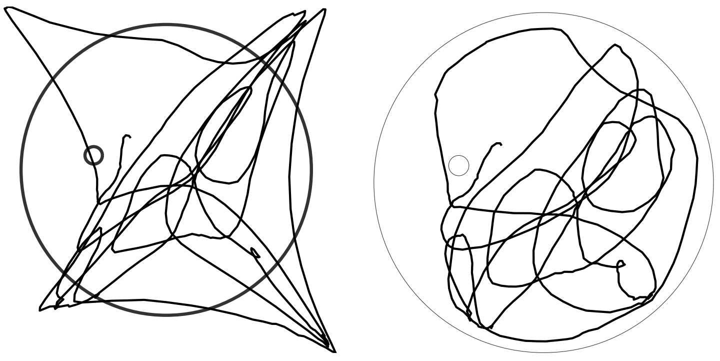


Figure B.3: Example of data calibration applied to one trajectory. *Left*: original trajectory as exported by the tracking software. *Right:* corrected trajectory that closely matches the trajectory shown by the tracking software on the screen.

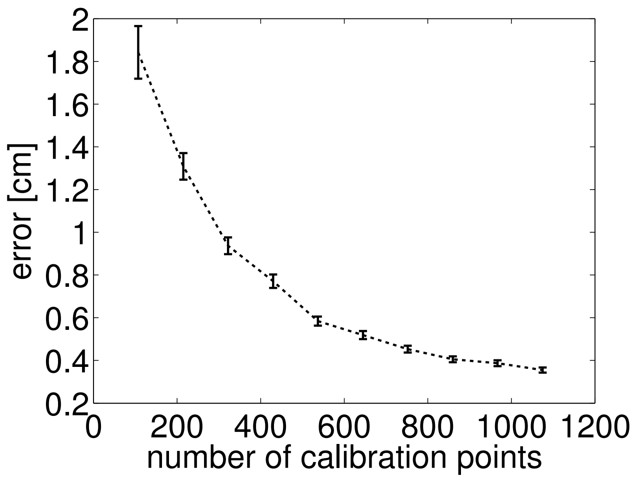


Figure B.4: Calibration error as a function of the number of calibration points. The error represents the distances (in the x or y axes) between corrected coordinates and points as extracted from snapshots taking from the recording software. Only minor improvements are observed after around 500 calibration points. Error bars represent the 95% CI.

The calibration was validated by using a (ten-fold stratified) cross-validation measuring the error of the interpolated function with calibration points not used in the interpolation (Figure B.4).

The results show the average error (x and y axis) for different number of calibration points; as can be seen the improvements are only minimal when using more than 500 calibration points (representing 7-8 trajectories in the data used here).

For the final data calibration the full set of data points was used, 1654 for each coordinate for the first set and 1195 for the second.

References

Abu-Mostafa, Y. S. (Apr. 1995). “Machines that Learn from Hints”. In: *Scientific American* 272.4, pp. 64–69. ISSN: 0036-8733. DOI: 10.1038/scientificamerican0495-64.

Aggarwal, C. C., A. Hinneburg, and D. A. Keim (2001). “On the surprising behavior of distance metrics in high dimensional space”. In: *Database Theory – ICDT 2001*, pp. 420–434. ISSN: 0956-7925. DOI: 10.1007/3-540-44503-X_27. arXiv: 0812.0624.

Aggarwal, C. C. and C. K. Reddy (2013). *DATA Custering Algorithms and Applications.* Ed. by C. Aggarwal, Charu; Reddy. Boca Raton, Florida: CRS Press, p. 652. ISBN: 1466558210 9781466558212.

Akil, H., M. E. Martone, and D. C. Van Essen (2011). “Challenges and Opportunities in Mining Neuroscience Data”. In: *Science* 331.6018, pp. 708–712. ISSN: 0036-8075. DOI: 10.1126/science.1199305. arXiv: NIHMS150003.

Anand, S., S. Mittal, O. Tuzel, and P. Meer (2014). “Semi-supervised kernel mean shift clustering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.6, pp. 1201–1215. ISSN: 01628828. DOI: 10.1109/TPAMI.2013.190.

Armañanzas, R. and G. A. Ascoli (2015). “Towards the automatic classification of neurons”. In: *Trends in Neurosciences* 38.5, pp. 307–318. ISSN: 1878108X. DOI: 10.1016/j.tins.2015.02.004. arXiv: 15334406.

Arthur, D. and S. Vassilvitskii (2007). “K-Means++: the Advantages of Careful Seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 8, pp. 1027–1025. ISSN: 0898716241. DOI: 10.1145/1283383.1283494.

Aston-Jones, G., J. Rajkowski, and J. Cohen (2000). “Locus coeruleus and regulation of behavioral flexibility and attention.” In: *Progress in brain research* 126, pp. 165–82. ISSN: 0079-6123. DOI: 10.1016/S0079-6123(00)26013-5.

Ayram, S. and T. Kainen (2006). *Introduction to partitioning-based clustering methods with a robust example.* Vol. 35. C, pp. 1–34. ISBN: 951392467X. DOI: C.1/2006.

Bammer, G. (1982). “Pharmacological investigations of neurotransmitter involvement in passive avoidance responding: A review and some new results”. In: *Neuroscience and Biobehavioral Reviews* 6.3, pp. 247–296. ISSN: 01497634. DOI: 10.1016/0149-7634(82)90041-0.

Basu, S., A. Banerjee, and R. Mooney (2002). “Semi-supervised Clustering by Seeding”. In: *Proceedings of the 19th International COnference on Machine Learning (ICML-2002)* July, pp. 19–26.

Basu, S., M. Bilenko, and R. Mooney (2004). “A probabilistic framework for semi-supervised clustering”. In: *Proceedings of the tenth ACM SIGKDD ...* August, pp. 59–68.

BeMent, S. L., K. D. Wise, D. J. Anderson, K. Najafi, and K. L. Drake (1986). “Solid-State Electrodes for Multichannel Multiplexed Intracortical Neuronal Recording”. In: *IEEE Transactions on Biomedical Engineering* BME-33.2, pp. 230–241. ISSN: 15582531. DOI: 10.1109/TBME.1986.325895.

Berdondini, L., K. Imfeld, A. Maccione, M. Tedesco, S. Neukom, M. Koudelka-Hep, and S. Martinoia (2009). “Active pixel sensor array for high spatio-temporal resolution electrophysiological recordings from single cell to large scale neuronal networks.” In: *Lab on a chip* 9.18, pp. 2644–51. ISSN: 1473-0197. DOI: 10.1039/b907394a.

Berkhin, P. (2006). “Survey of Clustering Data Mining Techniques”. In: *Grouping Multidimensional Data: Recent Advances in Clustering*, pp. 25–71. ISSN: 1557900X. DOI: 10.1007/3-540-28349-8_2.

Bestel, R., A. W. Daus, and C. Thielemann (2012). “A novel automated spike sorting algorithm with adaptable feature extraction”. In: *Journal of Neuroscience Methods* 211.1, pp. 168–178. ISSN: 01650270. DOI: 10.1016/j.jneumeth.2012.08.015.

Bijuraj, L. V. (2013). “Clustering and its Applications”. In: *Proceedings of National Conference on New Horizons in IT - NCNHIT 2013*, pp. 169–172.

Bilenko, M., S. Basu, and R. J. Mooney (2004). “Integrating constraints and metric learning in semi-supervised clustering”. In: *Twenty-first international conference on Machine learning - ICML ’04*, p. 11. DOI: 10.1145/1015330.1015360.

Bilmes, J. (1997). “A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models”. In: *Tech. Report ICSI-TR-97-021.*

Bishop, C. (2006). “Pattern recognition and machine learning”. In: *Ulb.Tu-Darmstadt.De.* ISSN: 0162-1459. DOI: 10.1198/jasa.2008.s236.

Bishop, C. M. (1995). *Neural networks for pattern recognition.* Clarendon Press, p. 482. ISBN: 0-19-853864-2.

Blumberg, M. S., J. H. Freeman, and S. R. Robinson (2010). *Oxford handbook of developmental behavioral neuroscience*, p. 784. ISBN: 9780195314731. DOI: `10.1017/CBO9781107415324.004`. arXiv: `arXiv:1011.1669v3`.

Bovenkerk, B. and F. Kaldewaij (2014). "The Use of Animal Models in Behavioural Neuroscience Research". In: *Current topics in behavioral neurosciences.* Vol. 19, pp. 17–46. DOI: `10.1007/7854_2014_329`.

Bradley, P. S. and P. S. Bradley (1998). "Refining Initial Points for K-Means Clustering". In: *Microsoft Research*, pp. 91–99. DOI: `10.1.1.44.5872`.

Brandeis, R., Y. Brandys, and S. Yehuda (Sept. 1989). "The use of the Morris Water Maze in the study of memory and learning." In: *The International journal of neuroscience* 48.1-2, pp. 29–69. ISSN: 0020-7454.

Breiman, L. (1984). *Classification and regression trees.* Wadsworth International Group, p. 358. ISBN: 0534980546.

Bruin, J. P. C. de, F. Sànchez-Santed, R. P. W. Heinsbroek, A. Donker, and P. Postmes (1994). "A behavioural analysis of rats with damage to the medial prefrontal cortex using the morris water maze: evidence for behavioural flexibility, but not for impaired spatial navigation". In: *Brain Research* 652.2, pp. 323–333. ISSN: 00068993. DOI: `10.1016/0006-8993(94)90243-7`.

Bubenikova-Valesova, V., A. Stuchlik, J. Svoboda, J. Bures, and K. Vales (2008). "Risperidone and ritanserin but not haloperidol block effect of dizocilpine on the active allothetic place avoidance task." In: *Proceedings of the National Academy of Sciences of the United States of America* 105.3, pp. 1061–1066. ISSN: 0027-8424. DOI: `10.1073/pnas.0711273105`.

Bures, J., a. a. Fenton, Y. Kaminsky, J. Rossier, B. Sacchetti, and L. Zinyuk (1997). "Dissociation of exteroceptive and idiothetic orientation cues: effect on hippocampal place cells and place navigation." In: *Philosophical transactions of the Royal Society of London. Series B, Biological sciences* 352.1360, pp. 1515–1524. ISSN: 0962-8436. DOI: `10.1098/rstb.1997.0138`.

Chapelle, O., B. Schölkopf, and A. Zien (Sept. 2006). *Semi-supervised learning.* Ed. by O. Chapelle, B. Scholkopf, and A. Zien. The MIT Press. ISBN: 9780262033589. DOI: `10.7551/mitpress/9780262033589.001.0001`.

Chen, S. S. and P. S. Gopalakrishnan (1998). "Clustering via the Bayesian information criterion with applications in speech recognition". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2, pp. 645–648. ISSN: 15206149. DOI: `10.1109/ICASSP.1998.675347`.

Cimadevilla, J. M., Y. Kaminsky, A. Fenton, and J. Bures (2000). "Passive and active place avoidance as a tool of spatial memory research in rats". In: *Journal of Neuroscience Methods* 102.2, pp. 155–164. ISSN: 01650270. DOI: `10.1016/S0165-0270(00)00288-0`.

Cimadevilla, J. M., M. Wesierska, A. A. Fenton, and J. Bures (2001). "Inactivating one hippocampus impairs avoidance of a stable room-defined place during dissociation of arena cues from room cues by rotation of the arena." In: *Proceedings of the National Academy of Sciences of the United States of America* 98.6, pp. 3531–3536. ISSN: 00278424. DOI: `10.1073/pnas.051628398`.

D'Hooge, R. and P. D. Deyn (2001). "Applications of the Morris water maze in the study of learning and memory". In: *Brain Research Reviews* 36, pp. 60–90.

Dalm, S., J. Grootendorst, E. R. de Kloet, and M. S. Oitzl (Feb. 2000). "Quantification of swim patterns in the Morris water maze." In: *Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc* 32.1, pp. 134–9. ISSN: 0743-3808.

Dankert, H., L. Wang, E. D. Hoopfer, D. J. Anderson, and P. Perona (2009). "Automated monitoring and analysis of social behavior in Drosophila". In: *Nat Methods* 6.4, pp. 297–303. ISSN: 1548-7105. DOI: `nmeth.1310[pii]10.1038/nmeth.1310`.

Daszykowski, M. and B. Walczak (2010). "Density-Based Clustering Methods". In: *Comprehensive Chemometrics* 2, pp. 635–654. ISSN: 09758887. DOI: `10.1016/B978-044452701-1.00067-3`. arXiv: `10.1.1.71.1980`.

Dockery, C. A. and M. J. Wesierska (2010). "A spatial paradigm, the allothetic place avoidance alternation task, for testing visuospatial working memory and skill learning in rats". In: *Journal of Neuroscience Methods* 191.2, pp. 215–221. ISSN: 01650270. DOI: `10.1016/j.jneumeth.2010.06.029`.

Donoho, D. L. and I. M. Johnstone (1994). "Ideal spatial adapatation by wavelet shrinkage". In: *Biometrika* 81.3, pp. 425–455.

Egert, U., T. Knott, C. Schwarz, M. Nawrot, A. Brandt, S. Rotter, and M. Diesmann (2002). "MEA-Tools: An open source toolbox for the analysis of multi-electrode data with MATLAB". In: *Journal of Neuroscience Methods* 117.1, pp. 33–42. ISSN: 01650270. DOI: `10.1016/S0165-0270(02)00045-6`.

Esterie, P., M. Gaunard, J. Falcou, and J. T. Laprest (2012). "Exploiting multime dia extensions in C++: A portable approach". In: *Computing in Science and Engineering* 14.5, pp. 72–77. ISSN: 15219615. DOI: `10.1109/MCSE.2012.96`.

Estivill-Castro, V. (2002). "Why so many clustering algorithms". In: *ACM SIGKDD Explorations Newsletter* 8.

Everitt, B. S. (Mar. 1979). “Unresolved Problems in Cluster Analysis”. In: *Biometrics* 35.1, p. 169. ISSN: 0006341X. DOI: `10.2307/2529943`.

Fenton, a. a., M. Wesierska, Y. Kaminsky, and J. Bures (1998). “Both here and there: simultaneous expression of autonomous spatial memories in rats.” In: *Proceedings of the National Academy of Sciences of the United States of America* 95.19, pp. 11493–11498. ISSN: 0027-8424. DOI: `10.1073/pnas.95.19.11493`.

Ferrea, E., a. Maccione, L. Medrihan, T. Nieus, D. Ghezzi, P. Baldelli, F. Benfenati, and L. Berdondini (Jan. 2012). “Large-scale, high-resolution electrophysiological imaging of field potentials in brain slices with microelectronic multielectrode arrays.” In: *Frontiers in neural circuits* 6.November, p. 80. ISSN: 1662-5110. DOI: `10.3389/fncir.2012.00080`.

Firasta, N., M. Buxton, P. Jinbo, K. Nasri, and S. Kuo (2010). “Intel AVX: New Frontiers in Performance Improvements and Energy Efficiency”. In: May 2008, pp. 1–9.

Flach, P. (2012). *Data, Machine Learning: The Art and Science of Algorithms that Make Sense of*, p. 409. ISBN: 9781107096394.

Gallagher, M., R. Burwell, and M. Burchinal (1993). “Severity of spatial learning impairment in aging: development of a learning index for performance in the Morris water maze.” In: *Behavioral neuroscience.*

Gehring, T. V., G. Luksys, C. Sandi, and E. Vasilaki (2015). “Detailed classification of swimming paths in the Morris Water Maze: multiple strategies within one trial.” In: *Scientific reports* 5, p. 14562. ISSN: 2045-2322. DOI: `10.1038/srep14562`.

Gehring, T. V., E. Vasilaki, and M. Giugliano (2015). “Highly scalable parallel processing of extracellular recordings of Multielectrode Arrays.” In: *Annual International Conference of the IEEE Engineering in Medicine and Biology Society* 2015, pp. 4178–81. ISSN: 1557-170X. DOI: `10.1109/EMBC.2015.7319315`.

Gold, C., D. a. Henze, C. Koch, and G. Buzsáki (May 2006). “On the origin of the extracellular action potential waveform: A modeling study.” In: *Journal of neurophysiology* 95.5, pp. 3113–28. ISSN: 0022-3077. DOI: `10.1152/jn.00979.2005`.

Gray, C. M., P. E. Maldonado, M. Wilson, and B. McNaughton (1995). “Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex”. In: *Journal of Neuroscience Methods* 63.1-2, pp. 43–54. ISSN: 01650270. DOI: `10.1016/0165-0270(95)00085-2`.

Graziano, A., L. Petrosini, and A. Bartoletti (Nov. 2003). “Automatic recognition of explorative strategies in the Morris water maze”. In: *Journal of Neuroscience*

Methods 130.1, pp. 33–44. ISSN: 01650270. DOI: 10.1016/S0165-0270(03)00187-0.

Grossberg, S. (1987). “Competitive learning: From interactive activation to adaptive resonance”. In: *Cognitive Science* 11.1, pp. 23–63. ISSN: 03640213. DOI: 10.1016/S0364-0213(87)80025-3.

Guyon, I. and A. Elisseeff (2003). “An introduction to variable and feature selection”. In: *The Journal of Machine Learning Research* 3, pp. 1157–1182.

Haroutunian, V., E. Barnes, and K. L. Davis (1985). “Psychopharmacology Cholinergie modulation of memory in rats”. In: pp. 266–271.

Hartigan, J. a. and M. a. Wong (1979). “Algorithm AS 136: A K-Means Clustering Algorithm”. In: *Journal of the Royal Statistical Society C* 28.1, pp. 100–108. ISSN: 00359254. DOI: 10.2307/2346830.

Hazan, L., M. Zugaro, and G. Buzsáki (2006). “Klusters, NeuroScope, NDManager: A free software suite for neurophysiological data processing and visualization”. In: *Journal of Neuroscience Methods* 155.2, pp. 207–216. ISSN: 01650270. DOI: 10.1016/j.jneumeth.2006.01.017.

Heuschkel, M. O., M. Fejtl, M. Raggenbass, D. Bertrand, and P. Renaud (2002). “A three-dimensional multi-electrode array for multi-site stimulation and recording in acute brain slices”. In: *Journal of Neuroscience Methods* 114.2, pp. 135–148. ISSN: 01650270. DOI: 10.1016/S0165-0270(01)00514-3.

Holmes, G., A. Donkin, and I. Witten (1994). “Weka: A machine learning workbench”. In: *Intelligent Information Systems, . . .*

Holroyd, C. B. and M. G. Coles (2002). “The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity.” In: *Psychological Review* 109.4, pp. 679–709. ISSN: 1939-1471. DOI: 10.1037/0033-295X.109.4.679.

Illouz, T., R. Madar, Y. Louzon, K. J. Griffioen, and E. Okun (2016). “Unraveling cognitive traits using the Morris water maze unbiased strategy classification (MUST-C) algorithm”. In: *Brain, Behavior, and Immunity* 52, pp. 132–144. ISSN: 10902139. DOI: 10.1016/j.bbi.2015.10.013.

Jain, A. K. (2010). “Data clustering: 50 years beyond K-means”. In: *Pattern Recognition Letters* 31.8, pp. 651–666. ISSN: 01678655. DOI: 10.1016/j.patrec.2009.09.011. arXiv: 0402594v3 [arXiv:cond-mat].

Jonson-Reid, M., N. Presnall, B. Drake, L. Fox, L. Bierut, W. Reich, P. Kane, R. D. Todd, and J. N. Constantino (2010). “Effects of Child Maltreatment and Inherited Liability on Antisocial Development: An Official Records Study”. In: *Journal of the American Academy of Child & Adolescent Psychiatry* 49.4, pp. 321–332. ISSN: 08908567. DOI: 10.1016/j.jaac.2009.11.015. arXiv: NIHMS150003.

Jordan, M. I. and T. M. Mitchell (2015). “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245, pp. 255–260. ISSN: 0036-8075. DOI: 10.1126/science.aaa8415. arXiv: arXiv:1011.1669v3.

Karypis, G., E. Han, and V. Kumar (1999). “Chameleon: Hierarchical clustering using dynamic modeling”. In: *Computer* 32.8, pp. 68–75. ISSN: 00189162. DOI: 10.1109/2.781637.

Kaufman, L. and P. J. Rousseeuw (1987). *Clustering by means of medoids.*

Kaufman, L. and P. J. Rousseeuw (1990). “Finding groups in data. an introduction to cluster analysis”. In: *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York: Wiley, 1990.*

Kohonen, T. (1982). “Self-organized formation of topologically correct feature maps”. In: *Biological Cybernetics* 43.1, pp. 59–69. ISSN: 0340-1200. DOI: 10.1007/BF00337288.

Korz, V. (Aug. 2006). “Water maze swim path analysis based on tracking coordinates”. In: *Behavior Research Methods* 38.3, pp. 522–528. ISSN: 1554-351X. DOI: 10.3758/BF03192807.

Kovacs, G. T. A., C. W. Storment, and J. M. Rosen (1992). “Regeneration Microelectrode Array for Peripheral Nerve Recording and Stimulation”. In: *IEEE Transactions on Biomedical Engineering* 39.9, pp. 893–902. ISSN: 15582531. DOI: 10.1109/10.256422.

Kulis, B. (2013). “Metric Learning: A Survey”. In: *Foundations and Trends® in Machine Learning* 5.4, pp. 287–364. ISSN: 1935-8237. DOI: 10.1561/2200000019. arXiv: 2200000019 [10.1561].

Kwak, N., C.-H. Choi, and J. Y. Choi (2001). “Feature Extraction Using ICA”. In: *Artificial Neural Networks—ICANN 2001*, pp. 568–573.

Law, M., A. Topchy, and A. Jain (2005). “Model-based Clustering With Probabilistic Constraints.” In: *Sdm*, pp. 1–5. DOI: 10.1137/1.9781611972757.77.

Lewicki, M. S. (1998). “A review of methods for spike sorting: the detection and classification of neural action potentials.” In: *Network* 9.4, R53–R78. ISSN: 0954-898X. DOI: 10.1088/0954-898X/9/4/001.

Liang, F., S. Mukherjee, and M. West (May 2007). “The Use of Unlabeled Data in Predictive Modeling”. In: *Statistical Science* 22.2, pp. 189–205. ISSN: 0883-4237. DOI: 10.1214/088342307000000032.

Lindner, M. D. (Nov. 1997). "Reliability, distribution, and validity of age-related cognitive deficits in the Morris water maze." In: *Neurobiology of learning and memory* 68.3, pp. 203–20. ISSN: 1074-7427. DOI: 10.1006/nlme.1997.3782.

Lindner, M. D. and V. K. Gribkoff (Oct. 1991). "Relationship between performance in the Morris water task, visual acuity, and thermoregulatory function in aged F-344 rats." In: *Behavioural brain research* 45.1, pp. 45–55. ISSN: 0166-4328.

Lloyd, S. P. (1982). "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. ISSN: 15579654. DOI: 10.1109/TIT.1982.1056489.

Luksys, G., W. Gerstner, and C. Sandi (Sept. 2009). "Stress, genotype and norepinephrine in the prediction of mouse behavior using reinforcement learning." In: *Nature neuroscience* 12.9, pp. 1180–6. ISSN: 1546-1726. DOI: 10.1038/nn.2374.

Luksys, G. and C. Sandi (June 2011). "Neural mechanisms and computations underlying stress effects on learning and memory." In: *Current opinion in neurobiology* 21.3, pp. 502–8. ISSN: 1873-6882. DOI: 10.1016/j.conb.2011.03.003.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on Mathematical Statistics and Probability* 233.233, pp. 281–297.

Mahmud, M., R. Pulizzi, E. Vasilaki, and M. Giugliano (Mar. 2014). "QSpike tools: a generic framework for parallel batch preprocessing of extracellular neuronal signals recorded by substrate microelectrode arrays". In: *Frontiers in Neuroinformatics* 8.March, pp. 1–14. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00026.

Mahmud, M. and S. Vassanelli (2016). "Processing and analysis of multichannel extracellular neuronal signals: State-of-the-art and challenges". In: *Frontiers in Neuroscience* 10.JUN, pp. 1–12. ISSN: 1662453X. DOI: 10.3389/fnins.2016.00248.

Manning, C. D., P. Ragahvan, and H. Schutze (2009). "An Introduction to Information Retrieval". In: *Information Retrieval* c, pp. 1–18. ISSN: 13864564. DOI: 10.1109/LPT.2009.2020494. arXiv: 05218657199780521865715.

Markham, J., J. Pych, and J. Juraska (2002). "Ovarian Hormone Replacement to Aged Ovariectomized Female Rats Benefits Acquisition of the Morris Water Maze". In: *Hormones and Behavior* 42.3, pp. 284–293. ISSN: 0018506X. DOI: 10.1006/hbeh.2002.1819.

Marom, S. and G. Shahaf (Feb. 2002). "Development, learning and memory in large random networks of cortical neurons: lessons beyond anatomy." In: *Quarterly reviews of biophysics* 35.1, pp. 63–87. ISSN: 0033-5835.

Márquez, C., G. L. Poirier, M. I. Cordero, M. H. Larsen, A. Groner, J. Marquis, P. J. Magistretti, D. Trono, and C. Sandi (2013). "Peripuberty stress leads to abnormal aggression, altered amygdala and orbitofrontal reactivity and increased prefrontal MAOA gene expression." In: *Translational psychiatry* 3.October 2012, e216. ISSN: 2158-3188. DOI: 10.1038/tp.2012.144.

Miller, N. E. (Apr. 1985). "The value of behavioral research on animals." In: *The American psychologist* 40.4, pp. 423–40. ISSN: 0003-066X.

Milligan, G. W. and M. C. Cooper (June 1985). "An examination of procedures for determining the number of clusters in a data set". In: *Psychometrika* 50.2, pp. 159–179. ISSN: 0033-3123. DOI: 10.1007/BF02294245.

Mitchell, T. M. (1997). *Machine Learning*, p. 414. ISBN: 0071154671. DOI: 10.1145/242224.242229. arXiv: 0-387-31073-8.

Morris, R. (May 1984). "Developments of a water-maze procedure for studying spatial learning in the rat." In: *Journal of neuroscience methods* 11.1, pp. 47–60. ISSN: 0165-0270.

Morris, R. (1981). "Spatial localization does not require the presence of local cues". In: *Learning and motivation* 260, pp. 239–260.

Moshtagh, N. (2005). "Minimum volume enclosing ellipsoid". In: *Convex Optimization*, pp. 1–9.

Munshi, A. (2012). *OpenCL programming guide.* Addison-Wesley, p. 603. ISBN: 0321749642.

Murphy, K. (2012). *Machine learning: a probabilistic perspective.* ISBN: 9780262018029.

Noldus, L. P. J. J., A. J. Spink, and R. a. J. Tegelenbosch (Aug. 2001). "EthoVision: A versatile video tracking system for automation of behavioral experiments". In: *Behavior Research Methods, Instruments, & Computers* 33.3, pp. 398–414. ISSN: 0743-3808. DOI: 10.3758/BF03195394.

Obien, M. E. J., K. Deligkaris, T. Bullmann, D. J. Bakkum, and U. Frey (2015). "Revealing neuronal function through microelectrode array recordings". In: *Frontiers in Neuroscience* 9.JAN, p. 423. ISSN: 1662453X. DOI: 10.3389/fnins.2014.00423.

Oka, H., K. Shimono, R. Ogawa, H. Sugihara, and M. Taketani (1999). "A new planar multielectrode array for extracellular recording: Application to hippocampal acute slice". In: *Journal of Neuroscience Methods* 93.1, pp. 61–67. ISSN: 01650270. DOI: 10.1016/S0165-0270(99)00113-2.

Pelt, J. van, I. Vajda, P. S. Wolters, M. a. Corner, and G. J. a. Ramakers (Jan. 2005). “Dynamics and plasticity in developing neuronal networks in vitro.” In: *Progress in brain research* 147, pp. 173–88. ISSN: 0079-6123. DOI: 10.1016/S0079-6123(04)47013-7.

Pelt, J. van, P. S. Wolters, M. a. Corner, W. L. C. Rutten, and G. J. a. Ramakers (Nov. 2004). “Long-term characterization of firing dynamics of spontaneous bursts in cultured neural networks.” In: *IEEE transactions on bio-medical engineering* 51.11, pp. 2051–62. ISSN: 0018-9294. DOI: 10.1109/TBME.2004.827936.

Pereira, F., T. Mitchell, and M. Botvinick (2009). “Machine learning classifiers and fMRI: A tutorial overview”. In: *NeuroImage* 45.1, S199–S209. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2008.11.007.

Petrosini, L., M. G. Leggio, and M. Molinari (Oct. 1998). “The cerebellum in the spatial problem solving: a co-star or a guest star?” In: *Progress in neurobiology* 56.2, pp. 191–210. ISSN: 0301-0082.

Pine, J. (1980). “Recording action potentials from cultured neurons with extracellular microcircuit electrodes”. In: *Journal of Neuroscience Methods* 2.1, pp. 19–31. ISSN: 01650270. DOI: 10.1016/0165-0270(80)90042-4.

Potter, S. M. and T. B. DeMarse (2001). “A new approach to neural cell culture for long-term studies.” In: *Journal of neuroscience methods* 110.1-2, pp. 17–24. ISSN: 0165-0270. DOI: 10.1016/S0165-0270(01)00412-5.

Powell, S. and P. Chau (1991). “A technique for realizing linear phase IIR filters”. In: *IEEE Transactions on Signal Processing* 39.11, pp. 2425–2435. ISSN: 1053587X. DOI: 10.1109/78.97998.

Qing, Z. Weizhong, M. Huifang, and He (2009). “Parallel K -Means Clustering Based on MapReduce”. In: *Cloud Computing. Springer Berlin Heidelberg* 2009, pp. 674–679. DOI: 10.1007/978-3-642-10665-1_71.

Quian Quiroga, R. (Mar. 2009). “What is the real shape of extracellular spikes?” In: *Journal of neuroscience methods* 177.1, pp. 194–8. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2008.09.033.

Quiroga, R. Q., Z. Nadasdy, and Y. Ben-Shaul (Aug. 2004). “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering.” In: *Neural computation* 16.8, pp. 1661–87. ISSN: 0899-7667. DOI: 10.1162/089976604774201631.

Raman, S. K., V. Pentkovski, and J. Keshava (2000). “Implementing streaming SIMD extensions on the Pentium III processor”. In: *IEEE Micro* 20.4, pp. 47–57. ISSN: 02721732. DOI: 10.1109/40.865866.

Redish, A. D. and D. S. Touretzky (1998). “The role of the hippocampus in solving the Morris water maze”. In: *Neural Comput* 10.1, pp. 73–111. ISSN: 0899-7667. DOI: http://dx.doi.org/10.1162/089976698300017908.

Rey, H. G., C. Pedreira, and R. Quian Quiroga (2015). “Past, present and future of spike sorting techniques”. In: *Brain Research Bulletin* 119, pp. 106–117. ISSN: 18732747. DOI: 10.1016/j.brainresbull.2015.04.007.

Richmond, P., L. Buesing, M. Giugliano, and E. Vasilaki (Jan. 2011). “Democratic population decisions result in robust policy-gradient learning: a parametric study with GPU simulations.” In: *PloS one* 6.5, e18539. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0018539.

Rokach, L. and O. Maimon (2008). “Data mining with decision trees: theory and applications”. In: *World Scientific Pub Co Inc.*

Rousseeuw, P. J. and C. Croux (1993). “Alternatives to the median absolute deviation”. In: *Journal of the American Statistical Association* 88.424, pp. 1273–1283. ISSN: 1537274X. DOI: 10.1080/01621459.1993.10476408.

Sakmann, B. and E. Neher (1984). “Patch clamp techniques for studying ionic channels in excitable membranes.” In: *Annual review of physiology* 46, pp. 455–472. ISSN: 00664278. DOI: 10.1146/annurev.physiol.46.1.455.

Sejnowski, T. J., P. S. Churchland, and J. A. Movshon (Oct. 2014). “Putting big data to good use in neuroscience”. In: *Nature Neuroscience* 17.11, pp. 1440–1441. ISSN: 1097-6256. DOI: 10.1038/nn.3839.

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding Machine Learning*, p. 409. ISBN: 9781107298019. DOI: 10.1017/CBO9781107298019.

Sharma, S., S. Rakoczy, and H. Brown-Borg (2010). “Assessment of spatial memory in mice”. In: *Life Sciences* 87.17-18, pp. 521–536. ISSN: 00243205. DOI: 10.1016/j.lfs.2010.09.004.

Shental, N., A. Bar-Hillel, T. Hertz, and D. Weinshall (2004). “Computing Gaussian mixture models with EM using equivalence constraints”. In: *Advances in Neural Information Processing Systems* 16.8, pp. 465–472. ISSN: 10495258.

Sibson, R. (1973). *SLINK: an optimally efficient algorithm for the single-link cluster method.* DOI: 10.1093/comjnl/16.1.30.

Siegel, S. (1956). *Nonparametric statistics for the behavioral sciences.*

Specht, D. F. (1991). “A general regression neural network.” In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 2.6, pp. 568–76. ISSN: 1045-9227. DOI: 10.1109/72.97934.

Spira, M. E. and A. Hai (2013). “Multi-electrode array technologies for neuroscience and cardiology”. In: *Nature Nanotechnology* 8.2, pp. 83–94. ISSN: 1748-3387. DOI: `10.1038/nnano.2012.265`.

Stoppiglia, H., G. Dreyfus, R. Dubois, and Y. Oussar (2003). “Ranking a Random Feature for Variable and Feature Selection”. In: *Journal of Machine Learning Research* 3, pp. 1399–1414. ISSN: 15324435. DOI: `10.1162/153244303322753733`.

Stover, K., T. O’Leary, and K. Brown (2012). “A Computer-Based Application for Rapid Un BiasedClassification of Swim Paths in the Morris Water Maze”. In: *Proceedings ofMeasuring Behavior 2012, 8th International Conference on Methods andTechniques in Behavioral Research (Utrecht, The Netherlands).* Pp. 353–357.

Stuchlík, a., T. Petrásek, I. Prokopová, K. Holubová, H. Hatalová, K. Valeš, S. Kubík, C. Dockery, and M. Wesierska (2013). “Place avoidance tasks as tools in the behavioral neuroscience of learning and memory”. In: *Physiological Research* 62.SUPPL 1. ISSN: 08628408.

Stuchlik, A., L. Rehakova, L. Rambousek, J. Svoboda, and K. Vales (2007). “Manipulation of D2 receptors with quinpirole and sulpiride affects locomotor activity before spatial behavior of rats in an active place avoidance task”. In: *Neuroscience Research* 58.2, pp. 133–139. ISSN: 01680102. DOI: `10.1016/j.neures.2007.02.006`.

Stuchlik, A., L. Rezacova, K. Vales, V. Bubenikova, and S. Kubik (2004). “Application of a novel Active Allothetic Place Avoidance task (AAPA) in testing a pharmacological model of psychosis in rats: Comparison with the Morris Water Maze”. In: *Neuroscience Letters* 366.2, pp. 162–166. ISSN: 03043940. DOI: `10.1016/j.neulet.2004.05.037`.

Stuchlik, A. and K. Vales (2008). “Role of alpha1- and alpha2-adrenoceptors in the regulation of locomotion and spatial behavior in the active place avoidance task: A dose-response study”. In: *Neuroscience Letters* 433.3, pp. 235–240. ISSN: 03043940. DOI: `10.1016/j.neulet.2008.01.013`.

Su, W., S. Mrug, and M. Windle (2010). “Social cognitive and emotional mediators link violence exposure and parental nurturance to adolescent aggression.” In: *Journal of clinical child and adolescent psychology : the official journal for the Society of Clinical Child and Adolescent Psychology, American Psychological Association, Division 53* 39.6, pp. 814–24. ISSN: 1537-4424. DOI: `10.1080/15374416.2010.517163`.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction.* Cambridge: MIT Press.

Suykens, J. A. K. and J. Vandewalle (1999). “Least Squares Support Vector Machine Classifiers”. In: *Neural Processing Letters* 9.3, pp. 293–300. ISSN: 1573-773X. DOI: 10.1023/A:1018628609742. arXiv: 1018628609742.

Tibshirani, R., G. Walther, and T. Hastie (2001). “Estimating the number of clusters in a data set via the gap statistic”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, pp. 411–423. ISSN: 1369-7412. DOI: 10.1111/1467-9868.00293.

Tibshirani, R. J. (2008). “Fast computation of the median by successive binning”. In: *Unpublished manuscript, http://stat.stanford.edu/ryantibs/median.*

Toledo, M. and C. Sandi (2011). “Stress during adolescence increases novelty seeking and risk taking behavior in male and female rats”. In: *Frontiers in Behavioral Neuroscience* 5.17. ISSN: 1662-5153. DOI: 10.3389/fnbeh.2011.00017.

Tompson, J. and K. Schlachter (2012). “An Introduction to the OpenCL Programming Model”. In: *Digital version available here.*

Vales, K., V. Bubenikova-Valesova, D. Klement, and A. Stuchlik (2006). “Analysis of sensitivity to MK-801 treatment in a novel active allothetic place avoidance task and in the working memory version of the Morris water maze reveals differences between Long-Evans and Wistar rats”. In: *Neuroscience Research* 55.4, pp. 383–388. ISSN: 01680102. DOI: 10.1016/j.neures.2006.04.007.

Varvel, S. A. and A. H. Lichtman (2002). “Evaluation of CB1 receptor knockout mice in the Morris water maze”. In: *J.Pharmacol.Exp.Ther.* 301.0022-3565 (Print), pp. 915–924.

Vasilaki, E., N. Frémaux, R. Urbanczik, W. Senn, and W. Gerstner (Dec. 2009). “Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail.” In: *PLoS computational biology* 5.12, e1000586. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000586.

Veenit, V., M. I. Cordero, S. Tzanoulinou, and C. Sandi (2013). “Increased corticosterone in peripubertal rats leads to long-lasting alterations in social exploration and aggression.” In: *Frontiers in behavioral neuroscience* 7.April, p. 26. ISSN: 1662-5153. DOI: 10.3389/fnbeh.2013.00026.

Vorhees, C. and M. Williams (2006). “Morris water maze: procedures for assessing spatial and related forms of learning and memory”. In: *Nature protocols* 1.2, pp. 848–858. DOI: 10.1038/nprot.2006.116.Morris.

Wagenaar, D. a., R. Madhavan, J. Pine, and S. M. Potter (Jan. 2005). “Controlling bursting in cortical cultures with closed-loop multi-electrode stimulation.” In: *The Journal of neuroscience : the official journal of the Society for Neu-*

roscience 25.3, pp. 680–8. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.4209-04.2005.

Wagstaff, K. (2000). "Clustering with Instance-level Constraints". In: pp. 1103–1110.

Wagstaff, K., C. Cardie, S. Rogers, and S. Schroedl (2001). "Constrained K-means Clustering with Background Knowledge". In: *International Conference on Machine Learning*, pp. 577–584. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2002.1017616.

Wang, W. and R. Muntz (1997). "STING : A Statistical Information Grid Approach to Spatial Data Mining". In: *Proceedings of 23rd International Conference on Very Large Data Bases*, pp. 1–10.

Warburton, E. C. and J. P. Aggleton (1998). "Differential deficits in the Morris water maze following cytotoxic lesions of the anterior thalamus and fornix transection". In: *Behavioural Brain Research* 98.1, pp. 27–38. ISSN: 01664328. DOI: 10.1016/S0166-4328(98)00047-3.

Wesierska, M. (2005). "Beyond Memory, Navigation, and Inhibition: Behavioral Evidence for Hippocampus-Dependent Cognitive Coordination in the Rat". In: *Journal of Neuroscience* 25.9, pp. 2413–2419. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.3962-04.2005.

Wesierska, M. J., W. Duda, and C. a. Dockery (2013). "Low-dose memantine-induced working memory improvement in the allothetic place avoidance alternation task (APAAT) in young adult male rats". In: *Frontiers in Behavioral Neuroscience* 7.December, pp. 1–12. ISSN: 1662-5153. DOI: 10.3389/fnbeh.2013.00203.

Wesierska, M., I. Adamska, and M. Malinowska (2009). "Retrosplenial cortex lesion affected segregation of spatial information in place avoidance task in the rat". In: *Neurobiology of Learning and Memory* 91.1, pp. 41–49. ISSN: 10747427. DOI: 10.1016/j.nlm.2008.09.005.

Wever, E. G. (1932). "Water temperature as an incentive to swimming activity in the rat." In: *Journal of Comparative Psychology* 14.2, pp. 219–224. ISSN: 0093-4127. DOI: 10.1037/h0071825.

Williams, R. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning*, pp. 1–27.

Witten, I. H., E. Frank, and M. a. Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques (Google eBook)*, p. 664. ISBN: 0080890369. DOI: 0120884070,9780120884070. arXiv: arXiv:1011.1669v3.

Witten, I. and E. Frank (2005). *Data Mining: Practical machine learning tools and techniques.* ISBN: 9780123748560.

Wold, S., K. Esbensen, and P. Geladi (1987). "Principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 2.1-3, pp. 37–52. ISSN: 01697439. DOI: 10.1016/0169-7439(87)80084-9.

Wolfer, D. P. and H. P. Lipp (Jan. 1992). "A new computer program for detailed off-line analysis of swimming navigation in the Morris water maze." In: *Journal of neuroscience methods* 41.1, pp. 65–74. ISSN: 0165-0270.

Wolfer, D. P., M. Stagljar-Bozicevic, M. L. Errington, and H.-P. Lipp (June 1998). "Spatial Memory and Learning in Transgenic Mice: Fact or Artifact?" In: *News in physiological sciences : an international journal of physiology produced jointly by the International Union of Physiological Sciences and the American Physiological Society* 13, pp. 118–123. ISSN: 0886-1714.

Wolfer, D. and H. Lipp (2000). "Dissecting the behaviour of transgenic mice: is it the mutation, the genetic background, or the environment?" In: *Experimental Physiology* January 2001.

Xiang, S., F. Nie, and C. Zhang (2008). "Learning a Mahalanobis distance metric for data clustering and classification". In: *Pattern Recognition* 41.12, pp. 3600–3612. ISSN: 00313203. DOI: 10.1016/j.patcog.2008.05.018.

Xing, E. P., A. Y. Ng, M. I. Jordan, and S. Russell (2003). "Distance Metric Learning , with Application to Clustering with Side-Information". In: *Nips.*

Xiong, S., J. Azimi, and X. Z. Fern (Jan. 2014). "Active Learning of Constraints for Semi-Supervised Clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 26.1, pp. 43–54. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.22.

Yin, X., S. Chen, E. Hu, and D. Zhang (2010). "Semi-supervised clustering with metric learning: An adaptive kernel method". In: *Pattern Recognition* 43.4, pp. 1320–1333. ISSN: 00313203. DOI: 10.1016/j.patcog.2009.11.005.

Zhu, X. (2006). "Semi-supervised learning literature survey". In: *Computer Science, University of Wisconsin-Madison.*

Zhu, X. and A. B. Goldberg (Jan. 2009). "Introduction to Semi-Supervised Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3.1, pp. 1–130. ISSN: 1939-4608. DOI: 10.2200/S00196ED1V01Y200906AIM006.