University of Tartu
Faculty of Mathematics and Computer Science
Institute of Computer Science
Computer Science

Kristina Martšenko

Using Machine Learning to Analyze Brain Activity During a Short-Term Memory Task

Bachelor's Thesis (6 ECTS)

Supervisors: Raul Vicente
Kristjan Korjus

Author: ..... "..." May 2014
Supervisor: ..... "..." May 2014
Supervisor: ..... "..." May 2014

Approved for defense
Professor: ..... "..." May 2014

Tartu 2014

Using Machine Learning to Analyze Brain Activity During a Short-Term Memory Task

Abstract
We analyze the electrical activity of neurons in the prefrontal cortex of a monkey while it performs a task requiring short-term memory. In the first part of the analysis we use supervised machine learning to see if we can predict the monkey's behavior from its brain activity. We find that, while unable to predict the behavior before it occurs, we are able to correctly determine it based on post-behavior brain activity 69% of the time. In the second part of the analysis we investigate how the activity of neurons changes during a day of repeating the task hundreds of times. We find that for many neurons it remains the same, but for some it increases or decreases. In addition, we find that how the activity of a neuron changes over the day is not related to how the neuron behaves during the task. These findings can lead to a better understanding of the properties of the prefrontal cortex and short-term memory.

Keywords: visual short-term memory, prefrontal cortex, delayed match-to-sample, machine learning, SVM, k-NN, hierarchical clustering

Masinõppe kasutamine ajuandmete analüüsiks lühimälu nõudva katse ajal

Lühikokkuvõte
Käesolev töö uurib ahvi prefrontaalse ajukoore neuronite elektrilist aktiivsust, kui ahv osaleb lühimälu nõudvas katses. Analüüsi esimeses osas kasutatakse juhendatud masinõpet, et näha, kas aju aktiivsuse põhjal on võimalik ennustada ahvi käitumist. Leitakse, et kuigi enne käitumist ei suudeta seda ennustada, on 69% tõenäosusega võimalik see õigesti välja lugeda käitumisejärgsest neuronite aktiivsusest. Analüüsi teises osas uuritakse, kuidas neuronite aktiivsus sadu katseid sisaldava päeva jooksul muutub. Leitakse, et enamasti püsib aktiivsus sama, kuid osadel neuronitel tõuseb või langeb. Samuti leitakse, et neuroni muutus päeva jooksul ei ole seotud sellega, kuidas see katse jooksul käitub. Need leiud võivad viia parema arusaamiseni prefrontaalse ajukoore ja lühimälu omadustest.

Võtmesõnad: visuaalne lühimälu, prefrontaalne ajukoor, masinõpe, SVM, k-NN, hierarhiline klasterdamine

Contents

Introduction

Understanding the brain is one of the biggest scientific challenges we face today. A deeper understanding could lead to breakthroughs in medicine, education, and other fields. Neuroscientists study the brain, collaborating with other scientific disciplines, including computer science.

The relationship between neuroscience and computer science goes both ways. On one hand, computational techniques and tools are invaluable in analyzing large quantities of data recorded from the brain. And conversely, the brain is known to be better than computers at many tasks (e.g. facial recognition) and learning about how the brain processes information can serve as an inspiration for the development of new algorithms (e.g. artificial neural networks). Clearly, there are many benefits to be had from studying the brain.

In this thesis we analyze the electrical brain activity of a monkey performing a task that requires short-term memory. The goal is to enhance our understanding of how short-term memory works. The analysis consists of two subprojects. In the first we try to predict the behavior of the monkey using just the brain recordings. In the second we try to relate the behavior of neurons during the day to their participation in the task.

In Section 1 we give an overview of some fundamental concepts in neuroscience which we will be using in our analysis. Additionally we describe the current state of research on each subject. In Section 2 we describe the dataset that our analysis is based on, and explain the machine learning techniques which we use for the analysis. In Section 3 we walk through the steps of our analysis, ending each subsection with the results from that step. In Section 4 we talk about the possible biological interpretation of our results, and discuss some potential limitations and future directions for our work. Finally, Appendix A points to the location of the scripts used in the analysis.

1 Background and related work

The brain is the central organ of the nervous system of most animals. It is responsible for processing sensory input and coordinating the actions of the animal, as well as for regulating sleep, hormones, learning, and memory. The human brain is estimated to contain around 200 billion nerve cells called *neurons*. One of the mechanisms through which the brain performs its duties is by passing around electrical impulses called *action potentials* or *spikes* [14]. Spikes are discrete signals generated by neurons. Neurons are said to *fire* when they initiate a spike. A neuron consists of a cell body, an axon which sends spikes to other neurons, and dendrites which accept spikes from other neurons [12]. When an action potential is emitted by a neuron, it moves along the axon, passes to another neuron's dendrite through a *synapse*, and gradually decays after reaching the other neuron's cell body. How frequently a neuron fires is called its *firing rate*. A sequence of spikes generated by a neuron is called a *spike train*.

In the brain, neurons are said to *code for* a specific activity or mental process, such as memorization, language, or the interpretation of visual images. Different regions of the brain contain neurons which code for different functions. The *prefrontal cortex* near the front of the head is thought to be related to cognitive control and the ability to coordinate thoughts and actions according to internal goals [9].

The prefrontal cortex has also been found to play a role in the function of short-term memory [4, 6]. *Short-term memory* is the ability to keep a small amount of information in mind for a short time [1]. The information can be recalled and manipulated quickly. In contrast, *long-term memory* can hold a very large amount of information for a long time, but the information also takes longer to recall. A subclass of short-term memory is *visual short-term memory*, in which an image is kept in mind for a short period of time.

Visual short-term memory is often tested in *delayed match-to-sample* tasks [5, 10]. In these tasks the subject is shown a *sample image*, followed by a *delay* of a few seconds, and then a *test image*. The subject must then make some decision based on the relationship of these two images, and upon making the correct decision is *rewarded*. This task is usually repeated hundreds of times, and each run is called a *trial*. During the trial the subject's brain activity is often recorded, for instance by using *microelectrodes*. The subject could be human or another animal. Monkeys are often used because they have a lot in common with humans and are generally allowed to have more severe procedures performed on them.

Machine learning comprises a set of techniques which are able to learn from data [2, 7, 16]. It has been used widely in neuroscience, for instance

to predict a person's behavior 10 seconds before the person even becomes aware of it [15], or to reconstruct the image a person is seeing from brain activity [11].

In our analysis we will use machine learning techniques with the goal of better understanding short-term memory. We use spiking data collected from prefrontal cortex neurons. We try to predict a monkey's actions while she performs a visual delayed match-to-sample task, and to analyze the change in brain activity over time.

2 Methods

2.1 Dataset

The data that we will analyze was experimentally collected by (and belongs to) professor Matthias Munk at the Max Planck Institute for Brain Research in Germany [13]. In the following we describe the experiment that was conducted and the resulting dataset. In addition, we describe two ways in which the data was preprocessed before analysis – smoothing and normalization.

2.1.1 Experimental setup

The experiment was performed on a female rhesus monkey (*Macaca mulatta*). Microelectrodes were placed in the monkey's prefrontal cortex. The electrodes measured the electrical activity of neurons while the monkey performed the following task, which is illustrated in Figure 1.

1. A sample image is displayed for half a second.
2. There is a 3 second delay.
3. A test image is displayed. The test image may be the same as the sample image or it may be different.
4. The monkey must press one of two buttons that are in front of it. One button means that the sample and test images are the same, and the other means that they are different.
5. If the correct button is pressed, the monkey is fed fruit juice as a reward.

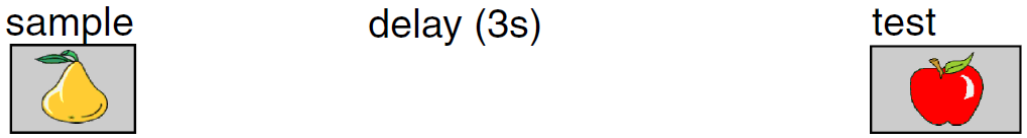

Figure 1: Steps of the experiment. Sample and test images from [13].

During the experiment the monkey had to use its short-term memory to remember the first image, compare it to the second, and press the correct button.

2.1.2 Dataset description

The electrodes were able to identify signals from 58 neurons. During each trial the electrodes recorded when each of these neurons fired. Since one trial lasted for about 6 seconds, the data collected from a trial consists of the firing times of 58 neurons during 6 seconds. An example of such a dataset can be seen in Figure 2.

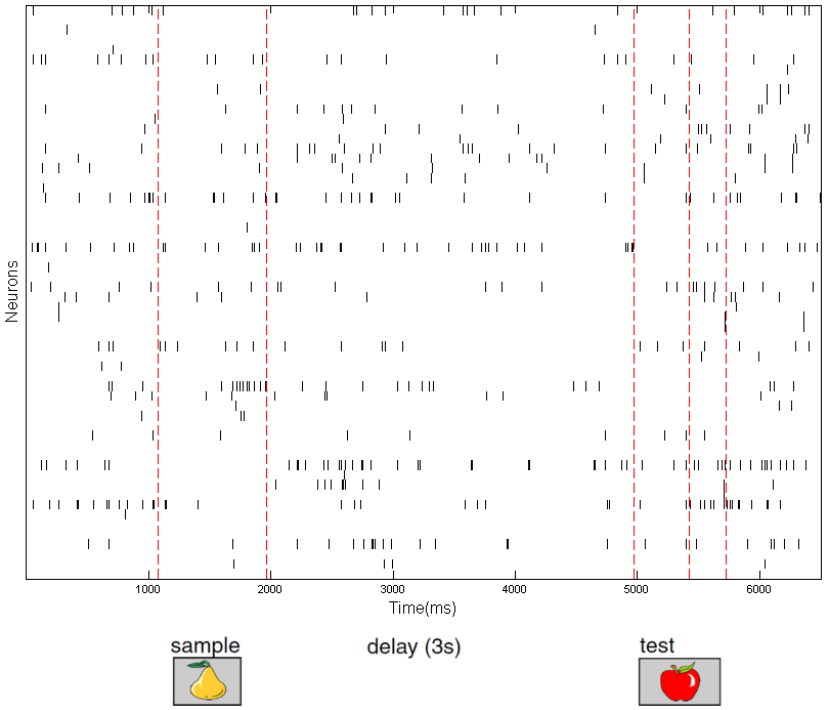

Figure 2: Data from one trial. On the x-axis is the length of the trial in milliseconds. On the y-axis are neurons; one neuron is one row. A vertical black line means that the neuron fired at that timepoint. The red lines show, from left to right, when: the sample image is shown; the sample image is removed; the test image is shown; the monkey presses a button; the monkey is fed.

The experiment was performed 871 times during one day. Out of those, the monkey answered correctly in 615 trials. We limit our entire analysis to only these "correct" trials, since the neurons are more likely to behave the same way in them, whereas the "incorrect" trials could have been caused by a number of factors (e.g. lack of attention or motivation, thinking about something else, looking somewhere else) and therefore the brain activity may be variable there.

To summarize, the dataset consists of 615 trials, each one with the firing times of 58 neurons during 6 seconds.

2.1.3 Smoothing

Before we can use the data, we need to apply some preprocessing steps to it. The first of these is smoothing. It simply spreads each spike out, as illustrated in Figure 3. It is almost always used with spike train data.

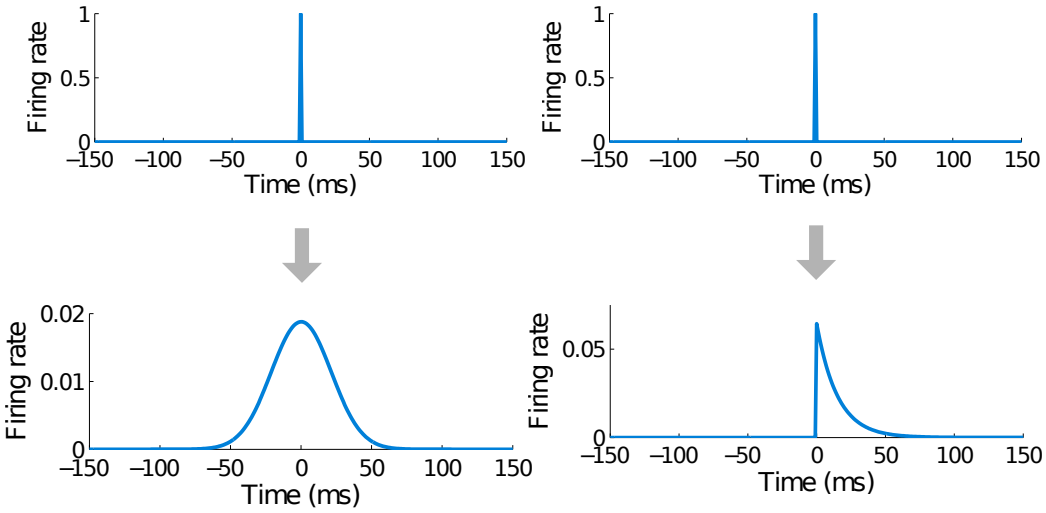

Figure 3: Smoothing functions. Smoothing with a Gaussian function on the left and with an exponential decay function on the right. Notice the change in scale.

Smoothing tries to account for noise in the data. For instance, if we have a neuron's spike trains from two trials, and the spikes of the second trial are exactly the same as those of the first, except shifted by one millisecond, then by default we would treat the two spike trains as different, because they don't have spikes at the same milliseconds. However, it's likely that they are actually similar, and the shift was caused by noise. One source of noise could be the stimulus, e.g. the picture in the experiment may appear a few milliseconds earlier or later in different trials. Another source of noise could be the intrinsic variability of the brain. Since there is constantly some spontaneous activity in the network of neurons, the state of the network in the beginning of a trial can vary across trials, and this can lead to a difference in spiking time. Smoothing lessens the effect of these kinds of noise by making the two spike trains much more similar.

Smoothing can be done with a Gaussian function or an exponential decay function. The exponential function is probably somewhat more realistic

because it's causal: after we record the spike on an axon, it reaches a cell and gradually decays there. This allows us to represent the electrical state of the neurons better.

The result of smoothing is a dataset which no longer consists of discrete spikes, but instead of continuous values. The value at each timepoint can be thought of as the firing rate at that timepoint. The effect of smoothing applied to one trial can be see in Figure 4.

The width of the function applied can also vary. Figure 5 depicts some of these.

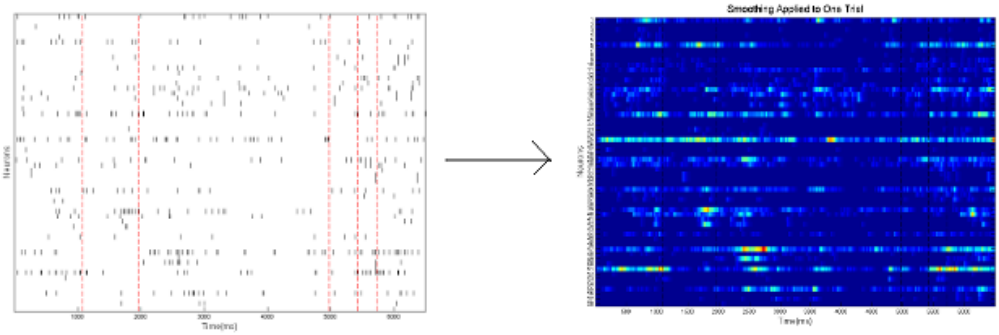

Figure 4: Smoothing on a trial. Dark red areas have many spikes, while dark blue areas have few spikes.

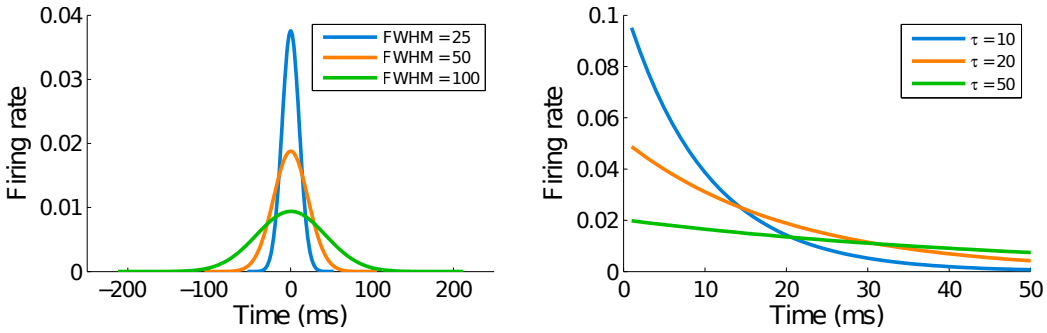

Figure 5: The width of smoothing functions. FWHM (full width at half maximum) and τ are parameters which determine the width of the functions. A larger value means the spike is spread out wider.

2.1.4 Normalization

The second preprocessing step is normalization. This puts the neurons on the same scale, which allows us to better compare their activity. To normalize,

we take each neuron's average firing rate before the experiment starts (the first 570 ms before the sample image is shown) as that neuron's *baseline*, or "usual" firing rate. Then we divide each subsequent timepoint (from 571 onwards) with the baseline. This makes each timepoint reflect how many times the neuron's firing rate went up or down at that timepoint, compared to before the experiment started. For example, a value of 2 means that the neuron's firing rate doubled compared to the baseline.

Additionally we may subtract 1 from the value at each timepoint, to make it reflect the size of the change itself. So a value of 2 would then mean that the firing rate went up by twice the baseline (and has therefore tripled).

Figure 6 depicts a normalized dataset. The firing rate is near 1 at the baseline, as expected. The rest of the plot shows how much the neurons' firing rates changed during the task. We see that even "quiet" (dark blue) neurons actually changed their activity quite a bit during the task.

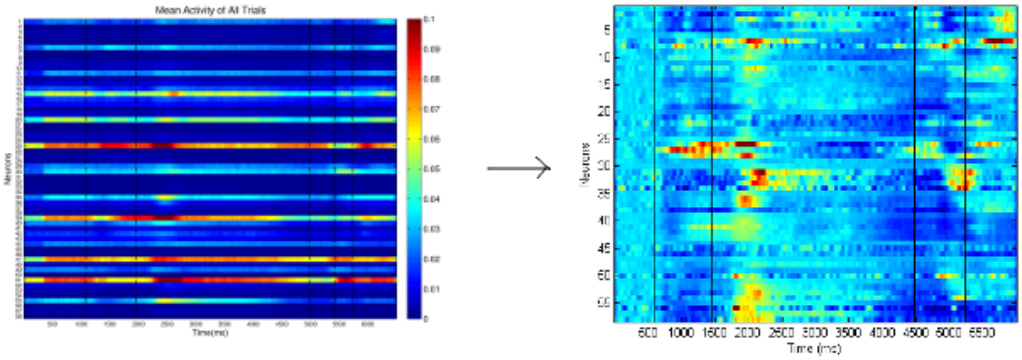

Figure 6: Normalization. The image on the left has been averaged over all trials (i.e. each point is the average firing rate at that point over *all* trials, not just the firing rate of one trial). The image on the right is derived by dividing each neuron's activity with its baseline.

2.2 Machine learning

Machine learning is the study of systems (models and algorithms) which learn from data. It is related to pattern recognition, in that we teach a program to recognize and act based on patterns. "Learning" can mean several things, such as making and improving predictions based on data or learning the structure of data. The important thing is that it's always based on some sort of data, such as observations, examples, or experience. The data could be as varied as spam and non-spam emails, handwriting, sensor data from robots, faces in images, and patients' medical histories.

Machine learning is a huge field with hundreds of different algorithms for solving a large number of specific problems. There are several types of algorithms. In our analysis we use two types: supervised learning and unsupervised learning.

2.2.1 Supervised learning

Supervised learning deals with algorithms which learn a model from a dataset and then use the model to predict properties of new data instances. There are two types of such algorithms: classification algorithms and regression algorithms. The former predicts which class an object belongs to, while the latter predicts a real-valued property (such as the price of a house). In our analysis we only use classification algorithms.

The learning process works as follows. First a *training* dataset is collected. The class membership of each object in the dataset (called a *training example*) is known, and each object is labeled with its class. Certain properties, or *features*, of the objects are chosen as the variables the algorithm will *classify* by. Next the algorithm is trained using the training examples. It learns to distinguish which features or feature combinations correspond to which class. The result is a generalized mathematical *model* (or *classifier*), which is a function of the features. Now, when we input the features of a new, previously unseen example to the model, it will *predict* the class the object belongs in.

The correctness of the prediction depends on how successful the training was. It could classify incorrectly for a number of reasons: too few training examples, too many features, an inappropriate choice of algorithm, or simply the lack of a relationship between the features and classes. One way to estimate the success of the training is to collect a *test set* of data (with known classes), input it to the classifier, and see what proportion it classifies correctly. This is called the *accuracy* of the classifier. A slightly more advanced version of this is *cross-validation*. The training and tests sets are merged, and the examples are randomly partitioned into a new training and test set. After training and validation, we get another estimate of the model's accuracy. The whole partition-train-validate routine can be repeated any number of times, such as 10 (called *10-fold cross-validation*). The 10 accuracy values are then averaged to obtain a "final" accuracy for the model.

A common example of a classification problem is spam classification. A training set of spam and non-spam emails is collected. Features are chosen, e.g. length of the email, sender, most used words. The algorithm is then trained on the emails and learns which feature combinations correspond to spam. The resulting model can classify new emails into spam and non-spam

folders. Other examples of classification problems include the classification of handwritten characters, the detection of faces in images, weather prediction, and medical diagnosis.

In our analysis we will use supervised learning to predict how quickly the monkey presses the button during the task. The trials will be the training examples, and we will classify between fast and slow response trials, using neurons' firing rates (at just one timepoint) as the features. Section 3.1 describes the process in more detail.

The performance of machine learning algorithms can vary depending on the dataset [3]. An algorithm may predict accurately on some datasets, but inaccurately on others. It can therefore be useful to try several algorithms on a given dataset. Some of the more popular classification algorithms are support vector machines (SVM), neural networks, Naive Bayes, k-nearest neighbor (k-NN), and decision trees. In our analysis we use SVM and k-NN, which we describe next.

SVM

A support vector machine (SVM) is a model which classifies between exactly two classes. It treats the examples as points in space, and trains on them to find as wide a gap as possible between the examples of the two classes. New examples are then placed in the same space, and their class membership is predicted by which side of the gap they lie on. The *support vectors* are the examples closest to the gap. An illustration of the idea behind an SVM can be seen in Figure 7.

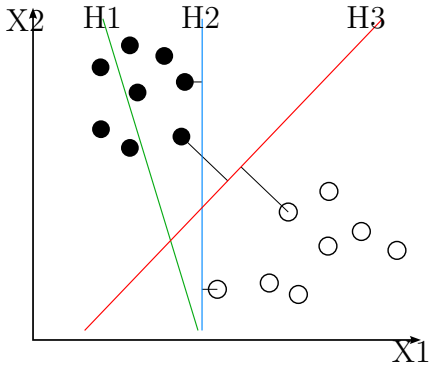

Figure 7: Illustration of an SVM model. H1 does not separate the two classes, while H2 barely does. H3 creates the widest gap between the classes, and is therefore chosen by SVM. The points with lines extending out of them are the support vectors.

Sometimes the classes aren't linearly separable, that is, they aren't well separated by, for example, a straight line in two dimensional space, or a flat plane in three dimensional space. An SVM can then create a more complex non-linear gap (e.g. a wavy line) by using a *kernel*, which maps the examples into a higher-dimensional space using a kernel function. An example of such a function is the Gaussian radial basis function (RBF) kernel.

k-NN

k-nearest neighbors (k-NN) is one of the simplest machine learning algorithms. It is trained by placing the labeled training examples in a feature space (i.e. a space where each feature is a dimension). A new object is then assigned to the class which is most common among the k training examples closest to it (its *neighbors*).

The value of k is a positive integer, usually small. The class of a new object depends on which k is chosen, as illustrated in Figure 8. The best choice of k depends on the dataset; we explore different values for it in our analysis.

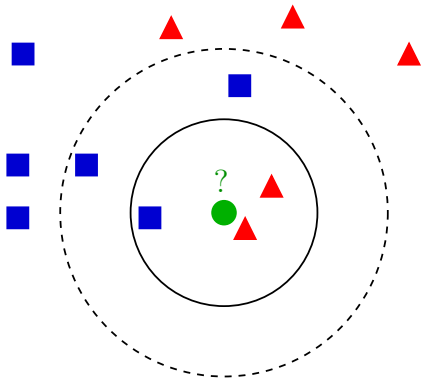

Figure 8: Illustration of k-nn classification. If $k = 3$, then the new (green) example is classified as red, but if $k = 5$, then it's classified as blue.

A number of distance measures can be used to calculate the distance between objects. In our analysis we use the most commonly used one, Euclidean distance, which usually works well for normalized data.

2.2.2 Unsupervised learning

The second type of machine learning technique we use is unsupervised learning. Unsupervised learning tries to find hidden structure in data. Unlike in supervised learning, there is no training set where the labels of objects are

already known; the algorithms need to find patterns in the data without first knowing what the "correct" patterns for other data instances are. Unsupervised learning includes techniques such as clustering, hidden Markov models, and blind signal separation. We focus on **clustering**.

Clustering involves taking a set of objects and finding groups, or *clusters*, in them, so that the objects in a cluster are more similar to each other than they are to objects in other clusters. An illustration of this is in Figure 9.

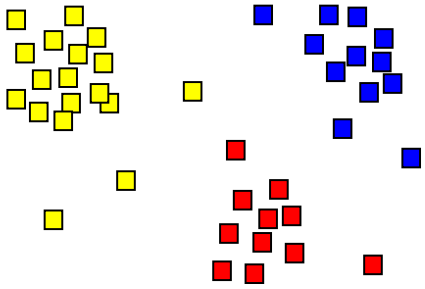

Figure 9: Illustration of clustering. Three clusters (red, blue, and yellow) have been found among the objects. Objects in one cluster are closer to each other than to objects in other clusters.

Clustering is used in a wide range of fields, such as to cluster DNA sequences in biology, or similar groups of people in social networks, or opinions in polls. In our analysis we cluster neurons based on their activity during the task. More on this in Section 3.3.

A number of different clustering algorithms exist; some examples are k-means, hierarchical clustering, and expectation-maximization. We use hierarchical clustering.

Hierarchical clustering

Hierarchical clustering builds a hierarchy of clusters out of a set of objects. An example of a hierarchy is shown in Figure 10. There are two ways to build the hierarchy: agglomerative and divisive. Agglomerative starts with each object in its own cluster, and then combines clusters until there's only one. Divisive starts with one cluster that has all the objects, and divides them until each object is in its own cluster. We will use **agglomerative** hierarchical clustering. The clusters are joined in a greedy way – first the two most similar clusters, then the next two most similar ones, and so on.

It's important to note that hierarchical clustering does not result in a clear partition of the data, such as the one in Figure 9, but instead builds a hierarchy, and it's up to the user to select clusters from it. A *dendrogram* is

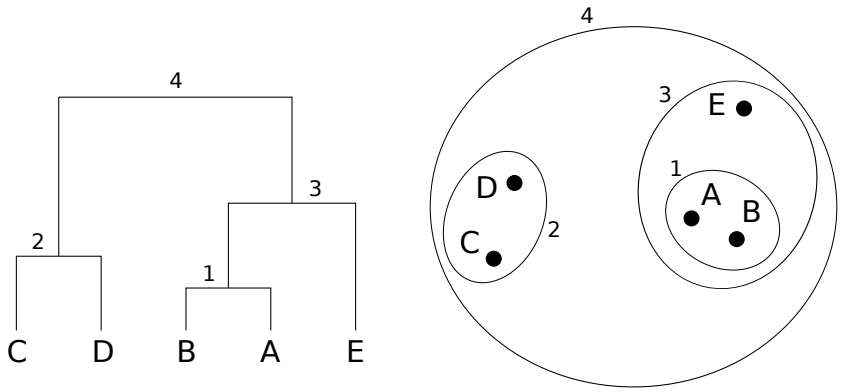

Figure 10: Hierarchical clustering. The image on the right shows the objects in feature space, and the circles represent the order in which they are joined into clusters and eventually a hierarchy. The image on the left is a dendrogram depicting the same clustering.

usually the best way to visualize the hierarchy and pick clusters from it (see Figure 10). In a dendrogram the objects are lined up at the bottom and the links show what order the objects and clusters are joined in. In general, the longer you have to trace a link to get from one object to another, the less similar those two objects are.

In order to calculate the similarity between two objects and decide which objects should be joined, a *distance measure* is used. The greater the distance, the less similar they are. More on this in the next section. In addition, once objects are joined, they form clusters, and now we also need to be able to calculate the distance between clusters, to further join them up in a hierarchy. This is done using a *linkage criterion*. More on this below.

Cosine distance

A number of distance measures exist to calculate the distance between objects, and the best one to use depends on the properties of the data. Some measures often used with hierarchical clustering include Euclidean distance, squared Euclidean distance, Manhattan distance, and cosine distance.

We cluster neurons based on their spike trains, treating them as 6000-dimensional vectors (one dimension for each timepoint), and try to find the distances between the vectors. We want neurons to be considered similar if they respond similarly to stimuli. For example, two neurons would be similar when both of their firing rates go up when the sample image appears. A neuron whose firing rate does not change or goes down would not be considered similar to them.

We use cosine distance because it meets these requirements. We also considered Euclidean distance, but found that it is affected too strongly by the magnitude of the firing rate. As an example, the Euclidean distance between the points (1, 0) and (0, 1) is $\sqrt{2}$, and the Euclidean distance between the points (0, 50) and (0, 100) is 50, making the first pair more similar than the second pair. However, if we treat the two dimensions as timepoints and the values as firing rates, we notice that the first two are firing at different timepoints, while the second two are both firing at the second timepoint. We would like to treat the second pair as more similar than the first, and as such Euclidean distance is not a very good measure. Cosine distance, on the other hand, uses the angle between the points (treated as vectors), and therefore considers the first two different and the second two similar, which is what we want.

We also tried correlation distance (i.e. one minus the sample correlation between points), and got similar results as with cosine distance.

Average linkage

Once some neurons have been joined into clusters, we need to also find distances between clusters, i.e. the distance between one set of objects and another set of objects. There are several possible linkage criteria for this. *Single-linkage* calculates the distances between all pairs of objects in the two clusters and uses the smallest of them as the distance between the clusters. In other words, it uses the distance between the two *closest* elements of the two clusters. *Complete linkage* instead uses the largest distance, i.e. the distance between the two *farthest* objects in the clusters. *Average linkage* calculates the average of all the pairwise distances and uses that as the distance between the clusters. We used average linkage because we did not expect there to be very clear clusters among the neurons, and average linkage may be a bit more stable than the others, lessening the effect of outliers.

3 Analysis and results

We now present the analysis we performed on the dataset and our results. The analysis is exploratory, but consists of two larger subprojects: 1) predicting the time it takes the monkey to press the button (Section 3.1), and 2) investigating how the firing rate of different types of neurons changes during the day (Sections 3.2–3.4). We used MATLAB to perform the analysis and plot the results.

3.1 Reaction time prediction

First we look at how quickly the monkey presses the button after the test image appears. We call this the monkey's *reaction time*. We want to use supervised machine learning to predict the reaction time in a trial based on the firing rate of neurons at some timepoint in that trial. (Note that *prediction* in this section always refers to its meaning in machine learning, as described in Section 2.2.1, and does not mean "foretelling the future".)

Predicting the reaction time might suggest which parts of the memorization process are important, or which activity or mental process the given 58 neurons code for. For example, if we could predict it in the first half of the experiment, it could mean that the neurons are modulated by motivation, attention, or some other global brain state, so that, for instance, more attention when looking at the sample image leads to faster reacting when the test image comes up. As another example, high prediction accuracy when looking at the test image might signal that the neurons code for motor movement, the actual physical action required to press the button. Finally, it would be interesting if we would be able to tell the reaction time several seconds ahead of time, before the button press itself.

In order to train the predictive model, trials were treated as training examples. Trials were divided into two groups, with reaction times shorter and longer than the median. The model was meant to classify trials into "fast" and "slow" classes. The features of one trial were the firing rates of its neurons at one fixed timepoint. Therefore each trial had 58 features, one for each neuron. An algorithm, either k-NN or SVM, was run on this setup to produce an estimated accuracy of how well the given timepoint can predict the reaction time (fast or slow). This was repeated for every 25th timepoint (1, 26, 51 ms, ...), which should provide a good enough approximation of the accuracy at every timepoint.

Both SVM and k-NN were used. Additionally, they were run with different parameters, in order to find what works better on our dataset. The k in k-NN was set to 3 or 10, and Euclidean distance was used as the distance

measure between trials. SVM was run with both a linear kernel and an RBF kernel. When using RBF, its scaling factor σ was set to 0.1, 1, or 10. The cost factor C was chosen to be 0.01, 0.1, 1, 10, or 100.

To assess the effect of smoothing on prediction accuracy, the learning was done on both smoothed and non-smoothed data. All data was normalized. When smoothing was done with the exponential function, the width parameter τ was set to 5, 25, or 50. In case of the Gaussian function, the parameter FWHM was set to 10, 50, or 100.

Additionally, we performed the analysis on each timepoint by including the two timepoints preceding it (e.g. for the 500th ms we included the 475th and 450th ms), for a total of $3 \times 58 = 174$ features. This may improve classification accuracy by providing information about how a neuron's activity changes over time. We did not use more than 3 timepoints because, in order for machine learning to work, the number of trials needs to be much greater than the number of features, and we only had 615 trials.

Classification accuracy at each timepoint was assessed using 10-fold cross-validation.

The results of the classification are shown in Figure 11. Prediction accuracy remains near 50% for most of the task, equivalent to classifying randomly. Accuracy rises somewhat after the button press, near the reward time. The most predictive timepoint is the 5651th millisecond, with an accuracy of 68.94%, meaning that on average 68.94% of the trials were classified correctly based on the firing rates at that timepoint.

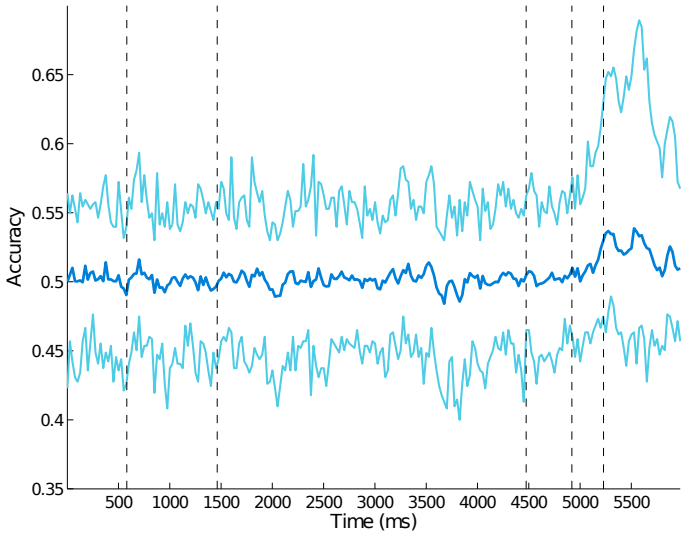

Figure 11: Best, average, and worst prediction accuracy. The average accuracy is calculated over all described parameter combinations.

Near the end of the task, SVM was found to predict somewhat better than k-NN. For SVM, the RBF kernel with $\sigma = 10$ performed best; the cost factor C did not seem to affect the accuracy too much. For k-NN, $k = 10$ predicted better than $k = 3$.

The algorithms predicted better from a smoothed trial than a non-smoothed one. The wider the smoothing function, the better the accuracy. The Gaussian function performed slightly better than the exponential function.

3.2 Change across the day

Next we would like to see how neurons' firing rates change during the day. A change during the course of the day could mean several things. It could be related to the task, possibly reflecting the process of learning. The connections between neurons could change in order to adapt to the task. Alternatively, the change in firing rate could be related to a global change, e.g. to a change in attention, motivation, or tiredness.

We found that the average firing rate of all neurons increases during the day. However, not all neurons' firing rates go up; some decrease as well and some fluctuate without a clear direction. Figure 12 summarizes the changes. Each neuron's change across the day was calculated by plotting its firing rates in all trials, fitting a line using linear regression (with least squares), and taking the slope of the line as the neuron's change.

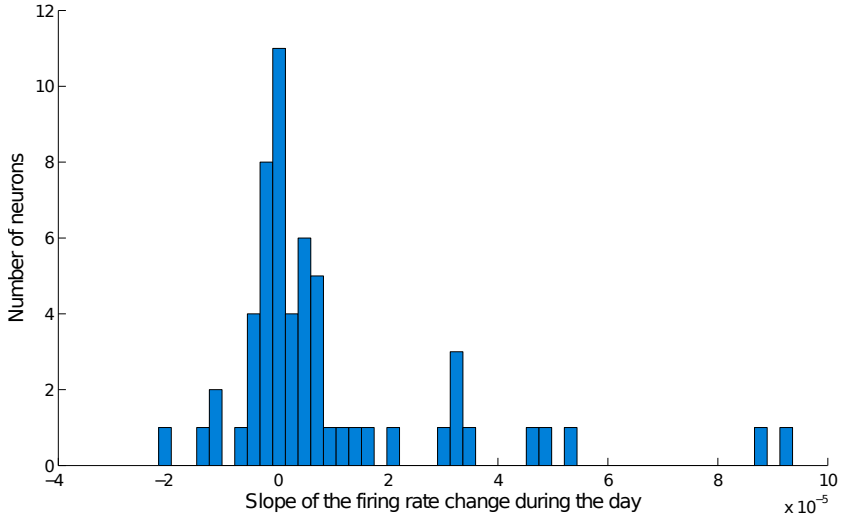

Figure 12: Distribution of neurons' firing rate changes during the day. A lot of neurons stay the same, some start firing less, and more start firing more.

3.3 Similarities in task activity

Next we would like to see how similarly neurons behave during the task. If they form clusters where neurons in one cluster respond the same way to the same stimuli, then this might indicate that they have certain roles in the task.

We clustered the neurons using hierarchical clustering. All neurons were first smoothed with a Gaussian function (FWHM = 50 ms), averaged across all trials, and normalized. Distances between neurons were calculated using cosine distance, treating each neuron as a 6000-dimensional vector (one dimension for each millisecond). Distances between clusters were found using average linkage.

The resulting dendrogram can be seen in Figure 13. We selected six clusters from it based on visual inspection, and left out the one neuron that did not belong to any larger cluster (neuron number 50 on Figure 13). The six clusters are shown in Figure 14.

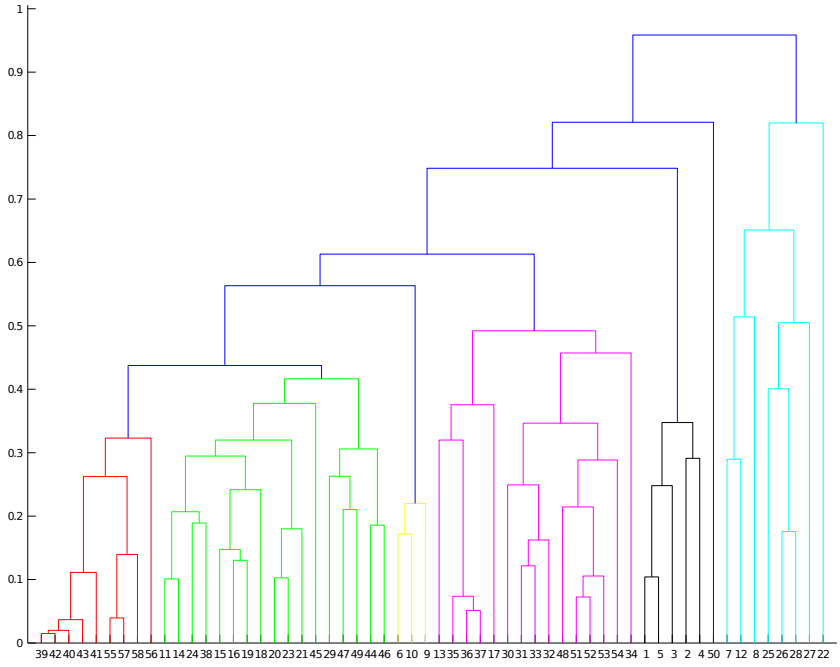

Figure 13: Dendrogram representing the hierarchical clustering of neurons. The six colored clusters were picked based on visual inspection.

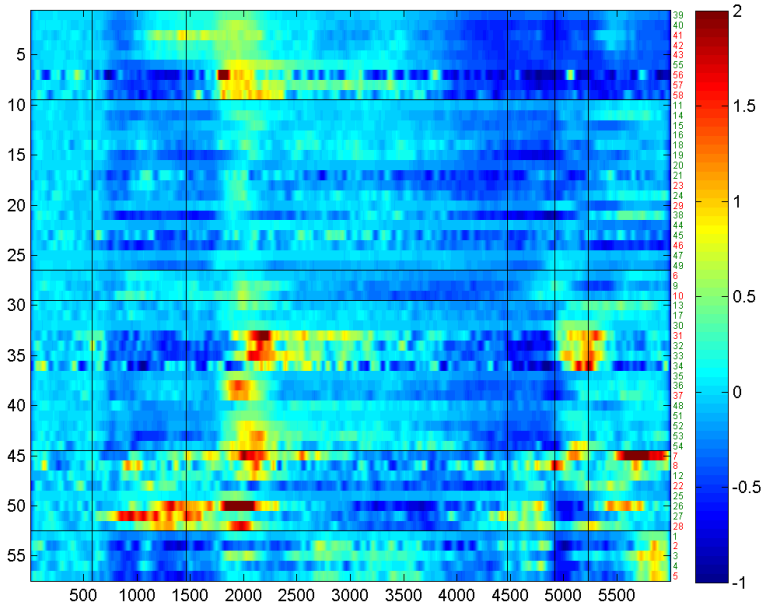

Figure 14: Neurons clustered based on their activity in the task. Black horizontal lines separate the six clusters. (The depicted activity has been produced by smoothing, averaging over all trials, and normalizing.)

3.4 Task activity compared to change across the day

Lastly, we would like to compare the activity of neurons during the task to their activity during the day. More specifically, we would like to see if neurons that behave similarly during the task also alter their firing rate similarly during the day. If this were true then we might be able to see whether these neurons are specific to a particular stage of the task, or respond to a particular stimulus, which might suggest its significance in the memorization process.

To do this we combined our results from Sections 3.2 and 3.3. We created two partitions of neurons. For the first, we split the neurons in two based on if their firing rate increases or decreases during the day. In the following we refer to them as *up* neurons and *down* neurons. For the second partition, we used the six clusters from Section 3.3 that describe task activity.

To formalize our goal, our null hypothesis is now that both *up* neurons and *down* neurons are distributed uniformly across the six task clusters. Our alternative hypothesis is that they are not distributed uniformly across task

clusters. We assign our significance level to be 0.05.

We compared the two partitions to see how much they overlap. The result is shown in Figure 14. A red number to the right of a neuron indicates a *down* neuron and a green number an *up* neuron.

It is difficult to judge by eye how well the partitions overlap, or correlate. One possible way to do it mathematically is to take the *up* neurons (or *down* neurons), see what percentage they make up of each task cluster, and calculate the standard deviation between the six percentages. A standard deviation of zero would mean that *up* neurons are uniformly distributed among the task clusters (because all the percentages are equal), which would mean that the two partitions are completely unrelated. A higher standard deviation would indicate a stronger correalation, i.e. that *up* neurons are more present in some task clusters, and *down* neurons are more present in other task clusters.

We got a standard deviation of 0.2323. (Note that the standard deviation for both *up* and *down* neurons is the same, because their percentages are complementary.) However, this does not tell us much because we do not know how likely we are to get this result by chance under the null hypothesis.

Therefore, to determine the statistical significance of our result, we perform a permutation test. A permutation test is convenient because we can use it with any statistic, including standard deviation. To perform the test, we randomly redistribute the *up* and *down* neurons among the task clusters, keeping both task and *up*/*down* cluster sizes fixed. We repeat this 10,000 times. Each time we calculate the standard deviation of the new partition. This gives us 10,000 standard deviations which are possible with our partition sizes. The frequencies of the standard deviations are plotted in Figure 15.

We can calculate the p-value of our result as the fraction of standard deviations to the right of our result on the histogram. We got 0.1344 as the p-value. As this is not under 0.05, we cannot reject the null hypothesis. We cannot say that *up* neurons and *down* neurons are not distributed uniformly across the task clusters. It seems that the *up*/*down* clusters and task clusters are independent, meaning that neurons that alter their firing rate similarly during the day do not necessarily behave similarly during the task.

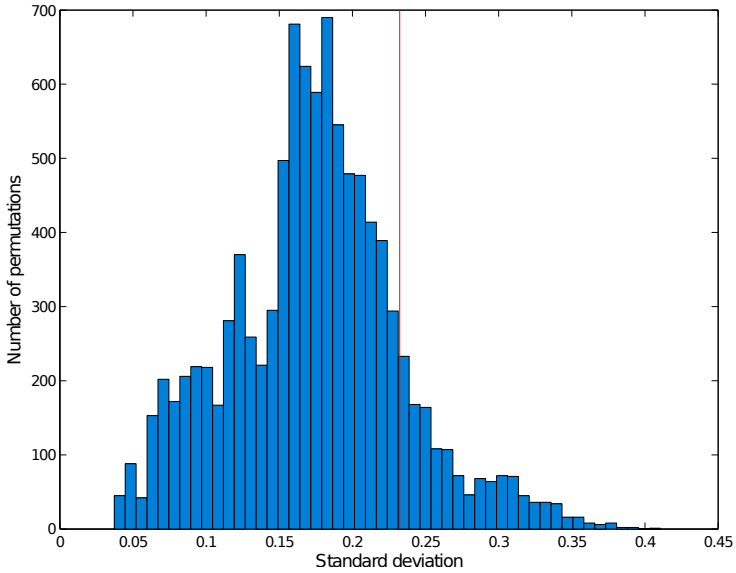

Figure 15: Distribution of possible standard deviations. The vertical line shows the standard deviation of our partition.

4 Discussion

4.1 Interpretation of results

In the first subproject, we were not able to predict the reaction time of the monkey before the button is pressed. It appears that that information is not part of the neurons' activity before the button press. We are only looking at a very small subset of neurons in the brain, so it is entirely likely that these neurons simply do not participate in the process of short-term memory.

On the other hand, we were able to predict the reaction time with almost 69% accuracy *after* the button press. While this may seem unremarkable because at that point we, as observers, already know how fast the button was pressed, it is important to remember that we are predicting purely from the activity of neurons, not from any external observations, and as such it is well worth considering.

It is difficult to say why prediction accuracy increases after the button press, near the reward time. One possible interpretation might be that the neurons are reacting to the reward. Since the reward comes sooner when the button is pressed faster, and later when the button is pressed slower, it might be that our prediction for a trial depends on if the reward in that trial has already arrived at the specified timepoint. Biologically it is plausible, since higher predictiveness is expected in response to a particular stimulus, e.g. the reward or a picture appearing. However, it is still only a very theoretical possibility, and the real reason could be something else entirely.

In addition, we found that SVM predicted better than k-NN, and which of their parameters worked better. We also found smoothing to be very important. We believe that this knowledge can benefit future analyses of spiking data, perhaps in other regions of the brain, where the number of neurons and their average firing rates are similar.

In the second subproject, we found similar groups of neurons in Section 3.3. The clusters turned out clearer than we expected. For instance, the first cluster can be easily identified as having low activity near the end of the task, the last cluster can be identified as having high activity at the very end, and so on. Finding such well defined clusters could indicate that the neurons may indeed have specific roles in the task. However, this is probably not the case here, as the clusters are more likely caused by the way the neurons were measured by electrodes (see Section 4.2).

In the last step of the analysis, we compared neurons' activity during a trial to how their firing rate changes during the day. We did not find a relationship between the two. It appears that neurons which behave similarly during the task can change differently over the day. And conversely, neurons

which change similarly during the day can play a part in any stage of the task.

4.2 Limitations

There are a number of issues which should be taken into account when interpreting the results.

For one, our dataset was relatively small, in both the number of neurons and trials. Reaction time prediction might have achieved better accuracy with more trials, as it would have meant more training examples and possibly a better model. In the second subproject, having more neurons per cluster could have provided more confidence in the relationship between the partitions.

Another important consideration is that spikes from neurons were recorded by a small number of electrodes. Each electrode recorded several neurons close to it. As a result, there is the possibility that readings from neurons which are from the same electrode may be similar, because by being very close to each other in the brain, the neurons influence each other. This might affect the clusters that we found when comparing the task activity of the neurons or their change across the day. Indeed, we did notice that, for several task clusters, a disproportionately large number of neurons in them were also next to each other in our original dataset. It is therefore possible that some of the clusters were not formed by the intrinsic similarity of the neurons, but instead their adjacency in the brain.

This also poses a problem for the permutation test, which assumes that all neurons are independent. Since in our case they do not seem to be, the chance of finding a correlation between the two partitions is higher, because the "artificial" similarity (caused by electrodes measuring nearby neurons) causes the affected neurons to always be in the same clusters (in both partitions). In other words, we are too likely to find a correlation and should increase our p-value to adjust for it. Since the p-value was higher than 0.05 anyway, our result of "no correlation" stays the same.

Another issue concerns the use of classification algorithms to predict reaction times. As reaction time is a continuous variable, it would make more sense to use regression algorithms. We decided to use classification algorithms because they are simpler (e.g. they don't have as many parameters to tweak), while still providing nonlinearity. The problem with classification is that if the variable has a normal distribution, then the objects in the middle will be separated into two classes, despite actually being similar. Since most objects are near the middle, this can result in poor classification accuracy. One possible way to alleviate the problem is to only use objects with more

extreme values, e.g. the top and bottom 25%, and leave out the middle part. Another option is to use regression algorithms.

A similar problem arises with the correlation of the two partitions in the second subproject. Since the changes in activity during the day are close to normally distributed (see Figure 12), the neurons with near zero slopes end up in different parts (increasing and decreasing), despite the slopes actually being close to each other. This could have made the correlation seem worse than it actually is. A better way would have been to calculate the standard deviation of a task cluster from the slopes of the neurons in that cluster, and then average over the task clusters to get the final standard deviation. However, as we had already visualized the correlation between the partitions using an increasing/decreasing day partition (see Figure 14), this partition naturally carried over into the calculation of the standard deviation as well.

Finally, it should be noted that when we calculated the slope of each neuron's change during the day (to produce Figure 12), not all neurons increased or decreased linearly. For such neurons the slope is not a very precise measure of how they changed during the day.

4.3 Future work

Based on our results, there are a number of additional ideas which would be promising to try.

As mentioned in the previous section, a larger dataset would be helpful. More trials could help predict the reaction time, and more neurons could help validate the relationship between the task and day partitions.

Several approaches could be used to try to improve the accuracy of reaction time prediction. The data could be locked to a different event, such as the button press time or reward time, to see how that affects the results. A trial's number during the day (i.e. whether it was the 1st trial of the day, the 5th, the 50th, etc.) could be added as an extra feature, to see if the change in neurons' activities during the day was affecting the accuracy. Feature selection methods, such as principal component analysis, could be used to either use fewer features, or possibly more timepoints at once. Our results suggest that changing the parameters of SVM/k-NN and smoothing might be useful. In addition, other algorithms, especially regression algorithms, could be tried instead of SVM and k-NN, as also discussed in the previous section. Finally, it would make sense to predict other aspects of the task, such as whether the monkey answered correctly or incorrectly, and to compare that to the current results.

In the second subproject, we considered how neurons' firing rates change during the day. It might be worthwile to not just look at how the total

firing rate (averaged over the whole task) changes, but also how the firing rate changes in individual stages of the task. In addition, when clustering neurons based on their task activity, it might be good to try a spike train specific distance measure [8, 17] instead of cosine distance.

Conclusion

We analyzed the activity of neurons in the prefrontal cortex of a monkey while the monkey performed a visual short-term memory task. In the first part, we tried to predict the monkey's behavior based on the activity of the recorded neurons. We were not able to do so before the behavior occurred, but had some success afterwards. Additionally we found which algorithms, parameters, and preprocessing steps work well for our kind of dataset. In the second part, we found that some neurons become more active as the day progresses, while others become less active or stay the same. We also found that some neurons respond similarly to stimuli presented during the task. These neurons did not, in general, also change similarly throughout the day. We believe that these findings provide important insights into the nature of the neurons under study, and perhaps even the prefrontal cortex itself.

The author of this thesis was responsible for performing the whole analysis, including writing all the necessary code. The research questions were provided by the supervisors. The author had very little prior knowledge of both machine learning and neuroscience, and learned a great deal in the process.

Bibliography

[1] A. D. Baddeley. *Human memory: Theory and practice*. Psychology Press, 1997.

[2] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[3] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[4] J. M. Fuster. *Prefrontal cortex*. Springer, 1988.

[5] J. M. Fuster and J. P. Jervey. Neuronal firing in the inferotemporal cortex of the monkey in a visual memory task. *The Journal of Neuroscience*, 2(3):361–375, 1982.

[6] P. Goldman-Rakic. Cellular basis of working memory. *Neuron*, 14(3):477–485, 1995.

[7] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[8] T. Kreuz, J. S. Haas, A. Morelli, H. D. Abarbanel, and A. Politi. Measuring spike train synchrony. *Journal of neuroscience methods*, 165(1):151–161, 2007.

[9] E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

[10] E. K. Miller, C. A. Erickson, and R. Desimone. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *The Journal of Neuroscience*, 16(16):5154–5167, 1996.

[11] Y. Miyawaki, H. Uchida, O. Yamashita, M.-a. Sato, Y. Morito, H. C. Tanabe, N. Sadato, and Y. Kamitani. Visual image reconstruction from human brain activity using a combination of multiscale local image decoders. *Neuron*, 60(5):915–929, 2008.

[12] J. Nolte. *The Human Brain: An Introduction to its Functional Anatomy*. Mosby, 2008.

[13] G. Pipa, E. S. Städtler, E. F. Rodriguez, J. A. Waltz, L. F. Muckli, W. Singer, R. Goebel, and M. H. Munk. Performance- and stimulus-dependent oscillations in monkey prefrontal cortex during short-term memory. *Frontiers in integrative neuroscience*, 3, 2009.

[14] F. Rieke. *Spikes: exploring the neural code*. MIT press, 1999.

[15] C. S. Soon, M. Brass, H.-J. Heinze, and J.-D. Haynes. Unconscious determinants of free decisions in the human brain. *Nature neuroscience*, 11(5):543–545, 2008.

[16] S. Theodoridis, A. Pikrakis, K. Koutroumbas, and D. Cavouras. *Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*. Academic Press, 2010.

[17] M. C. van Rossum. A novel spike distance. *Neural Computation*, 13(4):751–763, 2001.

Appendices

A MATLAB scripts

A number of MATLAB scripts were written to analyze the data. The scripts are available as a supplement to this thesis in the Graduation Thesis Registry of the Institute of Computer Science at the University of Tartu. The Registry can be publicly accessed at `http://comserv.cs.ut.ee/forms/ati_report/index.php?language=en`.

Non-exclusive licence to reproduce thesis and make thesis public

I, Kristina Martšenko (date of birth: 06.04.1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "Using Machine Learning to Analyze Brain Activity During a Short-Term Memory Task", supervised by Raul Vicente and Kristjan Korjus.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2014