CS886 Final Report: Machine Learning Methods for the Neural Decoding of Hippocampus Signals in Rats

David Xu D44XU@UWATERLOO.CA WATID 20552342
383 Albert St., Waterloo, Ontario N2L 6E3 Canada

Abstract

Decoding how the brain works is a difficult task. Machine learning methods can be leveraged to help understand its inner workings. Understanding how the brain works is key to further research in curing neurological disorders. A region of a rat's brain called the hippocampus is known to contain place cells. These place cells represent the spatial position of a rat in its surroundings. Using datasets from the Department of Biology at the University of Waterloo, this project implements and evaluates different machine learning methods used to decode a rat's position on a track, by analyzing signals from the rat's hippocampus. K-NN and SVM classifiers are used to predict location. Performance is heavily influenced by the hypothesis class and data preprocessing techniques used.

1. Introduction

The hippocampus is a region of the brain that is of particular interest due to its role in spatial navigation in mammals (O'Keefe and Recce, 1993). It is often associated with cognitive disorders, for example, a group of researchers found increased activity in the hippocampus on a group of individuals who had mild cognitive impairment (Dickerson et al., 2005). Interpreting signals from the brain is a difficult task without computers. This is where machine learning methods become applicable. Machine learning can do calculations and find patterns much faster than humans can. These methods can help reinforce existing neuroscience theories.

The field of computational neuroscience, has started to gain academic merit. For example, the 2014 Nobel Prize in Physiology or Medicine went to the discovery of brain cells in the hippocampus that act as a positioning system in a rat (Nobelprize.org, 2014). It was discovered that specific regions of the hippocampus in a rat would generate lots of cellular activity depending on the rat's position in a closed area. These are called place cells, and are theorized to create a place map of the spatial area that the mammal is in (O'Keefe, 1976). By observing a rat moving in a closed area, and measuring brain activity in the hippocampus, a set of data can be generated and analyzed. Reconstruction of a rat's location using its brain signals is referred to as neural decoding. It has been done using a machine learning method called Bayesian decoding; however, the reconstructions are not perfect, and accuracy is affected by assumptions the method makes (Zhang et al., 1998).

This project outlines machine learning techniques to reconstruct the position of a rat in a track by using signals from the rat's hippocampus. This project describes the feature extraction process, data preprocessing methods, and machine learning techniques to achieve neural decoding. We evaluate the affects of preprocessing neural data in time and space, and investigate performance of K-Nearest Neighbor (K-NN) and Support Vector Machine (SVM) algorithms. Figure 1 describes what the machine learning algorithm performs. The primary metric used to judge accuracy is the distance from the predicted position of the rat vs. the actual position.

2. Background Information

The process of extracting features from the hippocampus is a key step to neural decoding. Brain activity can be read from special brain cells called neurons. Neurons communicate with each other in the form of action potentials. They can be considered as binary actions of whether a neuron has fired or not. A mammalian brain is typically made up of a network of billions of neurons, and these interconnections enable the brain to provide

Final Report for CS886: Applied Machine Learning. University of Waterloo, Fall 2014.

Figure 1. Outline of neural decoding process for this project.

its functions (Dayan and Abbott, 2002). There has been many experiments that suggest the rate at which a neuron fires can encode a function. For example an experiment probed a neuron in the visual cortex (a brain region) of a cat and noticed the firing rates of a neuron would increase if the cat was shown a horizontal bar on a screen. The firing rate would subsequently decrease when the researchers gradually changed the orientation to a vertical bar (Hubel and Wiesel, 1959). This is similar to the approach taken to decode spatial positions of a rat. In other words, at a given time the specific neurons that fired in the rat's hippocampus can be decoded to determine the location of the rat.

It is very difficult to probe individual neurons because of their small size. Instead, the general approach is to attach several tetrodes to the area such as the hippocampus. A tetrode is an electronic device that can read brain signals when implanted into the brain region. Based on these signals, an unsupervised machine learning algorithm can be used to sort signals into cell groups (van der Meer, 2014). For example if the tetrodes read a signal with a similar frequency and amplitude as another reading, then it is likely that the signals is came from the same neuron. Then it is determined whether or not the neuron has fired a given time.

Figure 2, shows an outline of the feature extraction approach taken by Einevoll, Gaute T., et al. A tetrode is inserted into the brain region, in this case a rat's hippocampus, and analog signals are read. Filtering techniques are applied to remove noise and extract action potentials. Given the different waveforms of the spikes, clustering techniques are applied to detect where the location each spike was fired from. Spikes coming from similar locations would represent the same neuron. With this technique, the amount of neurons can be detected, and the firing times of each neuron can be calculated. This data will be the features used in order to decode rat position.

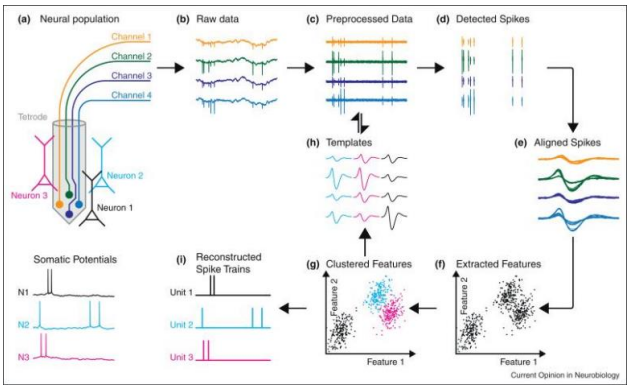

Figure 2. Sample process for spike time feature extraction from the brain. From the paper "Towards reliable spike-train recordings from thousands of neurons with multielectrodes. (Einevoll et al., 2012)"

3. Data Available

The van der Meer lab at the University of Waterloo is the main source of rat hippocampus data. Matt van der Meer is a professor under the Department of Biology, who has made his experimental data available for us to use. This paper expands on a project course under the Biology department, which will be referred to as NSB2014 (van der Meer, 2014). NSB2014 a tutorial on working with neurophysiological data from the lab, and uses MATLAB to process and evaluate the data.

The datasets consist of numerous experimental recordings of a rat inside a closed track. Each recording is approximately 40 minutes, and consists of the X, Y coordinates of the rat (2D top down view). Tetrode readings from the hippocampus are included, and consist of an array of specific times a neuron has fired. The data is sampled at a rate of 30Hz (every 33.3ms). Figure 3 shows a sketch of one of the track set ups, as well as X,Y locations of a 40 minute run plotted in MATLAB.

The tracks are set up in a linear fashion and experiments are monitored by humans. Water or food is given at each side of the track. After the rat has reaches one of those areas, they are given some time to move around. Then the rat is picked up and moved to the middle position which is the start of the track. The positions are still recorded, however the neural record-

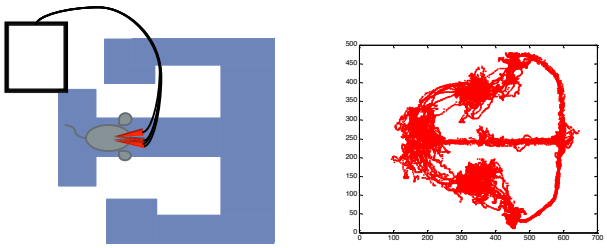

Figure 3. Left: A sketch of the track set up. Right: Plotted positions of the rat given a 40 minute recording.

ing is cut. Neurons were recorded with null data when the rat was picked up. This will not impact our results because training and prediction of location is not performed on empty neural data.

Tetrode readings have been clustered into neurons already. NSB2014 details the unsupervised machine learning methods to extract neuron classifications and firing rates. They use the software packages for neuron clustering called MCLust (Redish, 2014) and KlustaKwik (KlustaKwik, 2014). We did not have to perform the unsupervised methods described in Section 2. Instead we used the results of this process, which are neuron firing times as features.

4. Related Work

NSB2014 implements a simple version of position decoding. They create tuning curves for each neuron. Which are basically maps of the firing rate at each position for a given cell. Using these tuning curves the position can be decoded, given a sequence of time and the firing rates of each cell. They do this by using Bayes rule and assuming spikes follow a the Poisson distribution. This means, for each position, and given a time window, they estimate the expected spike rates for each neuron. A comparison of the incoming spike rates to their model generates an estimation for the rat's position.

In NSB2014 they group together firing rates and position on a neuron by neuron basis. This project will take a different approach. Each time window with the average position, and histogram of firing neurons will be used as the feature set. Another thing is that NSB2014 uses a full 40 minute rat run as the data. It does not split the data into any test set, and does prediction on data the algorithm has already learned from. They also do not provide any error metric.

5. Experimental Setup

5.1. Data Preprocessing

This paper considers a single rat dataset in order to stay consistent with the data. A dataset is an approximate 40 minute continuous recording of a rat in a track. There are many variables that change between datasets, such as the rat being used, the track set up, and position of the tetrode implants. These are all characteristics that limit the ability to train machine learning methods between datasets. The hippocampus structure is unique between rats which means the neurons extracted are not the same. The position of tetrodes may also change between datasets, which will extract different neurons.

How the neural data is preprocessed has an impact on the performance and accuracy of the algorithm. We used an approach which separates the data into samples of time. For example, we took a look at how many times each neuron fired in a given time frame. Figure 4 shows a histogram of neuron firings in a period of 500 milliseconds. The ensemble of firing neurons is theorized to represent the location of the rat. This approach is known as population coding (Abbott and Dayan, 1999).

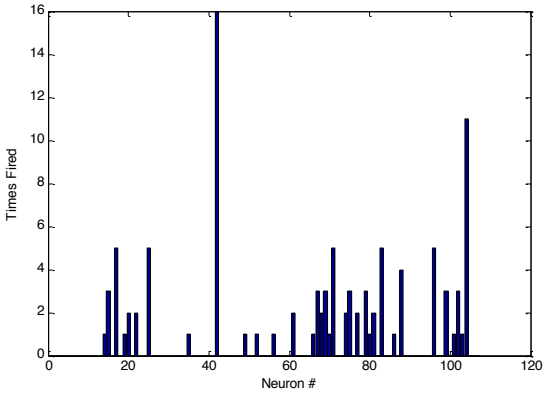

Figure 4. A neural population for a 500ms sample. The X axis represents the 107 neurons read from the tetrode. This is basically a histogram how many times each neuron fired in the given time window.

The time interval of these neuron populations affects accuracy and this is evident in the results section. The reason we did not consider a single time interval is because we don't know the time period that these neurons require to encode a position. An interval too small, such as 33ms may be too fast, and not capture enough data to encode anything. In addition, with a

Table 1. Example representation of data in MATLAB

Time Sample(sec)	Avg X	Avg Y	Neuron 1 Fire ...
1-1.99	232.1	241.1	2
2-2.99	232.0	246.1	2
3-3.99	234.6	250.6	5
4-4.99	234.5	252.2	6

40 minute run, this would generate 72000 data points which increases the time to train the algorithm. Conversely an interval too large, such as 1 second, may capture encodings for more than one position. Keep in mind, we are using the average position in each time interval. A tabular representation of time intervals can be seen in Table 1.

We considered the impact keeping the time series data sequential. With 10-Fold cross validation, the data is split into training, validation and test sets. Given a 40 minute recording, the data could be split so the first 32 minutes of the run is for training, the next 4 minutes for validation, and the last 4 minutes for testing. This may lead to skewed results. For instance the last 4 minutes of the rat could be tested in a locations that the machine learning algorithm was never trained on in the first 32 minutes. That situation would lead to poor results. This can be avoided by randomizing samples into a non sequential order. Conversely if place cell activity is influenced by the prior positions, then splitting the data non sequentially would impact the accuracy of the prediction as well. Both of these situations are considered in the results.

It was ideal to keep location data exact. And keep each individual X,Y location in raw data form such as 234.313, 261.111 when possible. Keeping each individual point with this precision took too long for some classification algorithms as this lead to thousands of possible classifications. It was necessary for some algorithms to make use of grouping locations in order to reduce the number of classifications. This process is shown in Figure 5, and is referred to as binning.

5.2. Testing Framework

A testing framework was built in MATLAB to achieve neural decoding, and evaluate the performance of data preprocessing techniques and machine learning methods.

This framework uses a simple method to measure error, which is the distance between the predicted position and the rat's actual position. The error will be calculated in terms of units, as the physical dimensions of the track are not provided. The equation is given in Figure 6. Figure 7 shows the possible locations the rat could be in which spans a 700 by 500 unit space. It also shows an example of error distance of approximately 40 units.

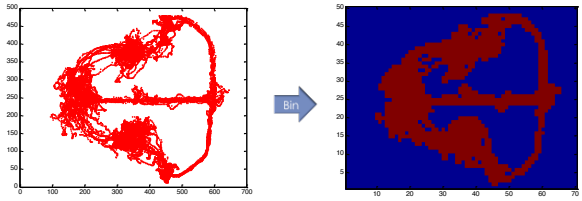

Figure 5. Left: Given X,Y readings of the data, there is a very large number of possible locations the rat can be in. Right: After applying a binning function, to translate into 50 possible X locations, and 70 possible Y locations, there are now only 1024 possible locations to classify to, in a 40 minute rat run

$$\hat{J}_H = \frac{1}{n}\sum_{i=1}^{n} DistanceBetween(h(x,y),(x,y)))$$

Figure 6. Error distance metric. Where N represents the amount of time samples to predict location, h(x,y) represents the predicted location, and (x,y) is the actual location.

The framework begins by loading the 40 minutes of raw location and neural data into MATLAB. Data is partitioned into 10 K-Folds. We chose 10 because it is a typical choice (Hastie et al., 2009). That results in 8 partitions of training data, 1 partition of validation, and 1 partition of test data saved until end of this process. The different hypothesis class variables are tested for a given algorithm and the error distance is calculated. This is performed for each permutation of 9 partitions (8 training, 1 validation). The best hypothesis class is chosen and then used to evaluate on test data to provide true generalization error. We also consider different windows of time to split neural populations. This process is outlined in pseudocode in Figure 8.

Note that time windows stay consistent in the process. If a time window of 250ms is chosen, then the population histograms for training, validation and test data are all sampled at 250ms.

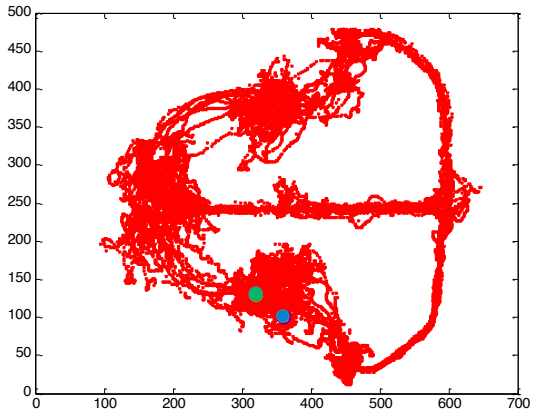

Figure 7. An example of approximately 40 unit difference between actual and predicted location. Green represents the actual position, while blue represents the predicted location. The red represents all the locations the rat was in.

```
For each timeWindow:
    Split data into 10 partitions
    For each permutation of training and validation sets:
        For each hypothesis class:
            Train algorithm with hypothesis class on training set
            Use trained algorithm on validation set
            Calculate error distance
        End
        Set best hypothesis class
    End
    Use best hypothesis class on test data
    Calculate true generalization error distance
End
```

Figure 8. Pseudocode for testing framework

6. Machine Learning Methods

6.1. K-NN Classifier

A K-NN classifier was implemented manually in MATLAB. This method stores all the training data of X,Y locations and population histograms for the training period. The X,Y location is predicted by looking at the input population histogram, and calculating the difference between histograms of the training data. The predicted X,Y location is a average of the X,Y locations in the K nearest distances. The K-Fold cross validation is meant to find the best K value in K-NN to use. The K values tested are 1,2,3,4 and 5.

6.2. SVM

The software library called LIBSVM (Chang and Lin, 2011) was used for an SVM algorithm in MATLAB. LIBSVM was chosen because of its easy to use interface, and built-in support for multi-class classification. LIBSVM builds a series of binary classifiers to perform multi-class classification. Two approaches are taken with the SVM.

6.3. Time samples as labels

The first approach uses time samples as the training labels and the population histograms as the training data. The SVM predicts the training label that the test data is most likely classified to. The X,Y location associated to that training label will be used to calculate distance between the X,Y location associated to the test data. This means each training point is a class that only occurs once in the training data. This means the C value does not affect the SVM. The reason we took this approach is to maintain precision of the location data, such as 234.313, 261.111. If we grouped data into bins, as described in the preprocessing section, we would end up with a location such as 23, 26 (binned by factor 10), and precision would have to be accounted for in the error. One downside of this approach is that we now thousands of possible classifications for location, which takes a lot of computing power. The hypothesis class tests the best type of kernel to use. Those are Linear, Polynomial, Radial and Sigmoid, implementation details can be found in the LIBSVM manual.

6.4. Binned locations as labels

The second approach is to group locations into bins in order to reduce classes, e.g. 234.313, 261.111 is grouped into 23, 26 by binning by a factor of 10. We still use time samples to define the data points, but instead of using timestamps as the training labels, the binned locations are the training labels. With this model multiple training points can be classified into the same class using the SVM. The hypothesis class tested the C value from a logarithmic scale from 1 to 10000 at first, and then scaled down to more precise C values. We also test the four types of SVM kernels provided in LIBSVM. One downside of this approach is that we lose precision, and have to account for the worst case error when trying to compare with other methods. For example, we binned location data on both axes by a factor of 10, the worst case is that a number such as 23.99 is binned into 23. A distance of 10 on each instance axis is the worst case scenario. This translates into a precision loss of 14.142 units

Table 2. Position accuracy for sequential vs random data with K-NN

Time Bin (ms)	Sequential	K Value	Avg Error
1000	Yes	2	57.294
1000	No	1	35.095
500	Yes	2	59.568
500	No	1	19.906
250	Yes	1	60.967
250	No	1	7.219

Table 3. K-NN Results with K-Value chosen from 10-Fold Cross Validation

Time Bin (ms)	K-Value	Avg Error	Std Dev.
1000	2	38.124	58.534
500	1	16.575	38.137
250	1	6.1791	15.994
100	2	1.959	8.811
67	2	1.006	6.383

(calculated by $\sqrt{10^2 + 10^2}$).

7. Results

7.1. Sequential Data vs Random

Table 2 shows the results for keeping data sequential vs non sequential. Because performance is improved, the rest of the results used non sequential data.

7.2. K-NN

Using 10-Fold cross validation, each time binning technique chose either 1-NN or 2-NN as the best hypothesis class. Table 3 shows that as time binning became smaller, the average error decreased.

7.3. SVM

7.3.1. Time samples as labels

Table 4 shows results for using SVMs with time samples as the labels. Using 10-Fold cross validation, each time binning technique chose a Linear kernel as the best hypothesis class. The C value is not shown because it had no affects on the results. Each class is trained with one unique data point. With SVMs as the time binning became smaller, the average error decreased.

Table 4. SVM Results with Kernel value chosen from K-Fold cross validation. The C value was tested as well but had no affect on results (every class was basically trained with 1 data point) so it is omitted from the table.

Time Bin (ms)	Kernel	Avg Error	Std Dev.
1000	Linear	39.894	67.418
500	Linear	27.251	58.525
250	Linear	18.177	47.786
100	Linear	8.106	22.141
67	Linear	3.468	3.675

Table 5. SVM Results with Kernel value chosen from K-Fold cross validation.*This is in terms of binned position units. In order to compare with original location, a precision error of 14.142 units needs to be accounted for.

Time Bin (ms)	C	Kernel	Avg Error*	Std Dev.*
1000	1	Linear	3.884	7.337
500	5	Linear	2.069	4.802
250	1	Linear	1.440	4.185

7.3.2. Binned locations as labels

Table 5 shows results of using SVMs with binned locations as the labels. Using 10-Fold cross validation, each location binning technique chose a Linear kernel as the best hypothesis class. The best C value varied. Again we observe that as time bins decrease, the average error deceases as well.

8. Discussion

Randomizing the data into a non sequential dataset decreased the average error distance. The result makes it hard to claim that prior position of a rat has a strong effect on the neural populations in the rat hippocampus. A possible reason for the decrease in error is that when the datasets are randomized, it is more likely that the training sets capture a broader spectrum of data to learn from. Likewise the test sets will have a closer representation to the training data. A trivial case of this with sequential data, is if the first 36 minutes comprise of a rat on one side of the track, and the last 4 minutes (test set) it is on the other side of the track. However we are surprised to see that it has such a large effect. Looking at the 250ms bin brackets in Table 2, there is a decrease in error distance by almost tenfold. It would be interesting to see how the algorithm would perform if the test set is in sequential form, but the training and validation set used is in non sequential form. Unfortunately due to time

constraints we could not run that test.

With non sequential data, in both K-NN and SVM methods we see a decrease in error distance as time bins became smaller. We did not reach a point where a smaller time bin increased the error distance. This may provide evidence that rat locations in its hippocampus are encoded in neuron populations with a time frame smaller than 67ms. Some restrictions keep us from going to smaller time bins. The SVM cross validation time was reaching the scale of days in our framework for a 40 minute run is split into approximately 36000 samples (40min/67ms). In addition, the sampling rate for positions is 30Hz, which prevents us from looking at positions in a time window less than 33.3 ms.

In regards to the SVM testing in both cases, a linear kernel performed the best in K-Fold cross validation. Table 6 shows an example of the kernel performance inside of a fold. It appears that the linear kernel performs significantly better in all cases. Initially we tested C values 1, 10, 100, 1000 and 10000, and observed 1 was always chosen. Then we retested C values 1, 3, 5 and 10. Unfortunately due to time constraints, we could not test SVMs trained with binned position labels for the 100ms and 67s time bins. In both cases, we see that the SVM trained with time sample labels, or binned position labels, perform better when the time bins are smaller.

Upon further inspection of the SVM results for binned positing labels, we were able to generate a histogram of the error distances seen in Figure 9. An interesting observation is that we saw a high frequency of errors in the 250 unit range. Upon inspection of the map, we noticed that in those instances, the prediction is in a location that is spatially symmetric to the rat's actual location, as pictured in Figure 10. This may be evidence that rat hippocampus neural populations appear similar when the rat is in a spatial location that looks similar as well. Although to make this claim would require more in depth testing.

Comparing Table 3 and Table 4, K-NN performed better than an SVM used in the same way (time samples as labels). The performance is much better in smaller time bins. The reason could be that at smaller time bins, K-NN has a lot more data stored in its algorithm. At 67ms, the algorithm stores about 35000 data points it could refer to.

It is very hard to make a fair comparison for Table 3 and 5, or Table 4 and 5, as the SVM using position bins as labels needs to account for precision loss when position binning.

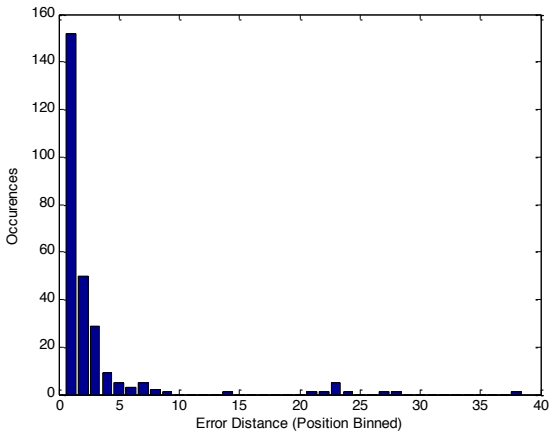

Figure 9. Histogram of error distance for SVM binned by position labels. Using a Linear kernel, a C value of 1, and time bins of 500ms.

Table 6. Results inside a K-Fold for position binned SVM. Time bin of 500ms. *This is in terms of binned position units. In order to compare with original location, a precision error of 14.142 units needs to be accounted for.

C Value	Kernel	Avg Error*	Std Dev.*
1	Linear	2.161	4.641
10	Linear	2.206	4.628
1	Polynomial	5.002	8.515
10	Polynomial	3.837	7.223
1	Radial	8.727	11.054
10	Radial	7.262	10.491
1	Sigmoid	5.855	8.575
10	Sigmoid	7.465	9.954

9. Conclusion

We used machine learning to test the theory of place cells in the rat hippocampus. We were able to reconstruct the rat's position with fairly high accuracy. This was done by looking at neural population firing inside of a rat's hippocampus within a time window, and associating that with a location inside of a track.

The performance of these algorithms are heavily influenced by the time windows of neural populations to look at. We saw increased performance when looking at smaller time windows. In addition, we can reconstruct the position with a high accuracy even if the data is non sequential.

This project found that machine learning methods can uncover new theories from the data as well. We saw

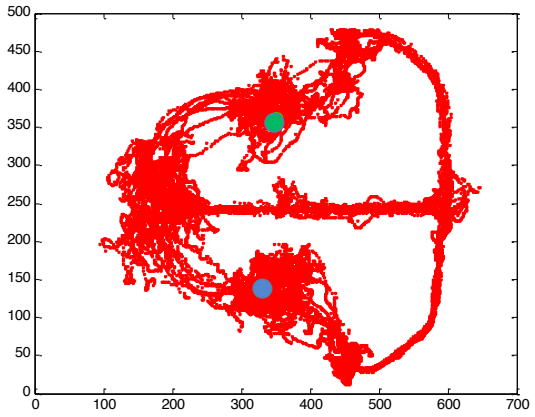

Figure 10. Sample representation of rat prediction and actual location when error distance is around 250. Green is the rat's actual location and blue is the predicted location.

that the results can show behaviour that similar neural populations may imply the rat is in a location that looks spatially similar, but much further than where it actually is. However; we cannot make that claim as that requires much more robust testing process, and more data.

A natural progression to this project would be to investigate more machine learning methods for neural decoding, and determine if these results are consistent between multiple rat datasets.

References

LF Abbott and Peter Dayan. The effect of correlated variability on the accuracy of a population code. *Neural computation*, 11(1):91–101, 1999.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Peter Dayan and L.F. Abbott. *Theoretical Neuroscience*. 2002.

BC Dickerson, DH Salat, DN Greve, EF Chua, E Rand-Giovannetti, DM Rentz, L Bertram, K Mullin, RE Tanzi, D Blacker, et al. Increased hippocampal activation in mild cognitive impairment compared to normal aging and ad. *Neurology*, 65 (3):404–411, 2005.

Gaute T Einevoll, Felix Franke, Espen Hagen, Christophe Pouzat, and Kenneth D Harris. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology*, 22(1):11–17, 2012.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574, 1959.

KlustaKwik. Klustakwik. http://klusta-team.github.io/klustakwik/, 2014. Accessed: 2014-10-16.

Nobelprize.org. The 2014 nobel prize in physiology or medicine - press release. http://www.nobelprize.org/nobel_prizes/medicine/laureates/2014/press.html, 2014. Accessed: 2014-10-16.

J O'Keefe and ML Recce. Phase relationship between hippocampal place units and the eeg theta rhythm. *Hippocampus*, 3(3):317–330, 1993.

John O'Keefe. Place units in the hippocampus of the freely moving rat. *Experimental neurology*, 51(1): 78–109, 1976.

David Redish. Mclust. http://redishlab.neuroscience.umn.edu/MClust/MClust.html, 2014. Accessed: 2014-10-16.

Matt van der Meer. Module 2: Introduction to neuralynx data formats and preprocessing. http://ctnsrv.uwaterloo.ca/vandermeerlab/doku.php?id=analysis:nsb2014:week2, 2014. Accessed: 2014-10-16 This is a graduate project course. Please contact Matt van der Meer at mvdm@uwaterloo.ca to request access.

Kechen Zhang, Iris Ginzburg, Bruce L McNaughton, and Terrence J Sejnowski. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *Journal of neurophysiology*, 79(2):1017–1044, 1998.