Data Analysis and Model Classification
Miniproject II

Takuya Ichihara, Freyr Sverrisson, Sharbatanu Chatterjee
Group 6

December 9, 2016

1 Introduction

In this miniproject, we intend to get familiarized with unsupervised learning techniques as Principal Component Analysis (PCA) and clustering. Then we will combine them with regression techniques (supervised learning) for the purpose of decoding continuous arm movements of a monkey from neural activities recorded from its brain. We propose to understand and use this to implement a control mechanism on the robotic arm based on the brain signals.

2 Methods

The first thing we did is to understand how spikes are extracted from the raw brain signals, thus understanding how the `Data.mat` dataset was created.

2.1 Spike extraction and sorting

In this section we worked with the data from `rawsignal.mat`, see figures 1 and 2.

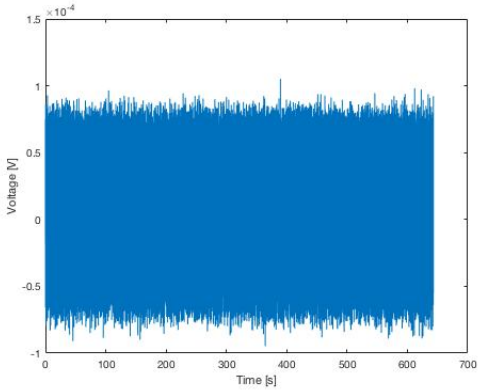

Figure 1: The data from `rawsignal.mat`.

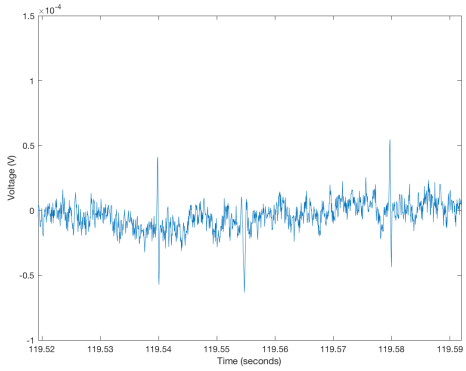

Figure 2: A zoomed in version of the `rawsignal`

We began with filtering the signal with a Butterworth filter of order 4 with cutoff frequencies low and high normalized by Nyquist frequency ($F_s/2$, where F_s is the sampling frequency of 20 kHz). Next we found the peaks of the signal (both positive and negative). We used $2ms$ for the minimum peak distance and experimented with different minimum peak heights (see figure 3). Based on figure 3 and our final clusters (figure 10) we iteratively (by eye) found 6.5 standard deviations of the signal to be a minimum peak height that found many peaks without adding to much noise (dense looking clusters with clean borders).

We then ran PCA on these extracted peaks and selected the minimum number of features that accounted for over 95% of the total variance of the data. We then used k-means clustering to assign peaks to different neurons. Finally, we calculated the firing rates of each neuron in 50 ms

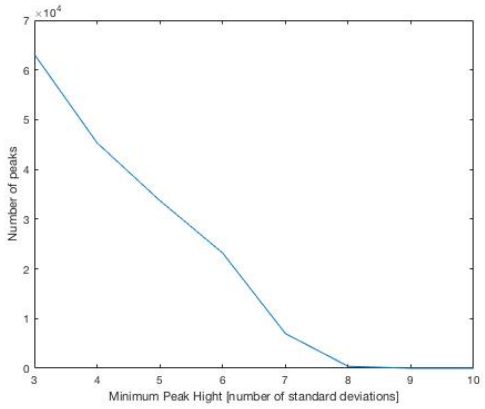

Figure 3: Number of peaks detected as a function of minimum peak height (number of standard deviations).

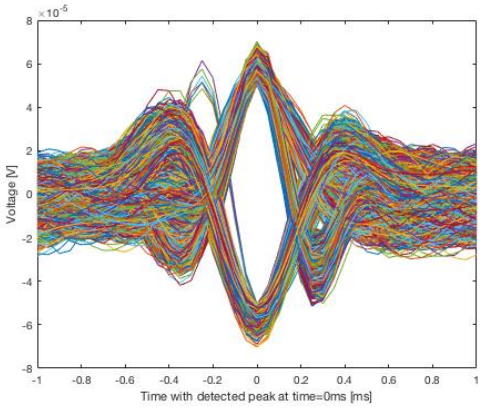

Figure 4: All extracted peaks in 2 ms windows (epochs).

windows, and by using maximum correlations with the 48 neuronal clusters of `Data.mat`, we found out the neuron numbers they correspond to.

2.2 Regression Analysis

First, to understand regularization, considering the big size of the data, we split it into 5% for the train data and 95% for the test data. However, for the final evaluations, we use 75% of the data for training and the rest for testing.

As we do not know if the neurons for the arm movement on the X axis are the same for that of on the Y axis, we do the regression analysis for X and Y independently.

For the analysis, we have four options to make a regression model. The first one is normal regression as the simplest one. However, in this project, we have too many features so that we can easily get overfitted models. In fact we tried out using linear and quadratic regression with the PCA-ed features, using all features at a time, and also while steadily going on reducing the features. We see that the test error does not improve. Trying to see how close the two curves are, we see by eye, a lot of overfitting. We present clear examples of overfitting in the following linear and quadratic regression examples. Just visually looking at the data shows the amount of overfitting. (5 and 6).

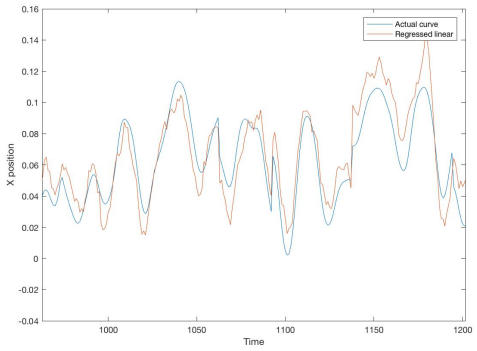

Figure 5: Linear regression of X position without regularization

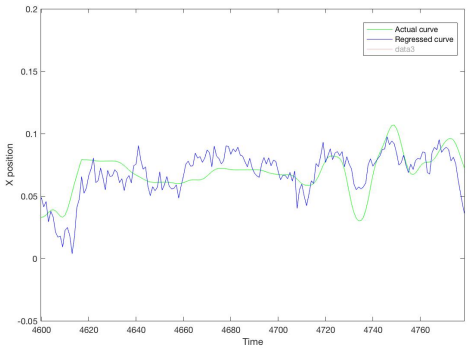

Figure 6: Quadratic regression of X position without regularization

To avoid this, we have three options to implement regularized regression; Lasso, Ridge, and Elastic nets. The equation for regularized regression has the penalty factor as below in addition to Mean Squared Error (MSE) used in the normal regression analysis.

$$\lambda \sum_{j=1}^{M} |w_j|^q \tag{1}$$

$$\lambda \sum_{j=1}^{M} (\alpha |w_j| + (1-\alpha) w_j^2) \tag{2}$$

If q=1, we have L1 regularization, Lasso, and if q=2, we have L2 regularization, Ridge. Lasso can reduce the number of features to 0. On the other hand, Ridge is a method with moderate feature selection and can pick up features which would be eliminated by Lasso. As an alternative, there is a combination of the two models, called Elastic nets. In this case, the weight for each regularization is given by α. There are two biggest drawbacks in Lasso, though it seems so effective with the feature reduction. The first one is: When $N < p$, it only sets a maximum of N out of p weights to a non-zero value (N: number of samples and p: number of features). This is not so much of a problem in our case, since p is not so much larger than N. The second one is: When groups of features are correlated, LASSO will only select one of these features since it optimises sparsity. This will cause a problem, as our 960 features are composed of 48 neurons $\times$ 20 50-ms bins, so that 20 features for each neurons should be correlated to some extent, though of course we might use PCA to remove them to some extent. But the problem of choosing between L1 and L2 norms is removed if we use Elastic nets. Then we might as well employ Elastic nets.

Therefore, there are two things we have to decide. One thing is whether we will make regression models with or without regularization and the other thing is how much weight we should put on Ridge and Lasso, if we use regularization. To do this, we do regression analysis in the following way.

First we prepare a set of $\alpha : \{0, 0.1, \cdots, 1.0\}$. For each α, we do regression analysis on the train data with λ with the range from 0 to 1 with 15 points logarithmically (when $\lambda = 0$, regression analysis will be without regularization) and do it with cross validation method.

Second for each α, we select a regression model with the least MSE and use it for the test data. Then we get the testing errors for each α.

Finally, compared the test errors we select the best model with λ and α. In the end, by applying this regression model to training data set, we obtain projected trajectories of X and Y.

3 Results

3.1 Spike extraction and sorting

Only two features after PCA were needed to account for over 95% of the total variance of the data (see figure 7).

When these features were plotted against each other (figure 8) it gave the first hint that the number of neurons could be 3.

For the clustering we experimented with different number of clusters (figure9).

As the reduction in total point-to-centroid distance is minimal with increasing the number of clusters over 3, we came to the conclusion that the number of clusters (neurons) was 3 (see figure 10).

Finally, figure 11 shows the firing rates for each neuron in 50 ms windows. By comparing these firing rates with the first 48 columns of the `Data.mat`, we find out which neurons they correspond to, by calculating which columns are most correlated with them. We find them to be no. 6, 15 and 40 respectively.

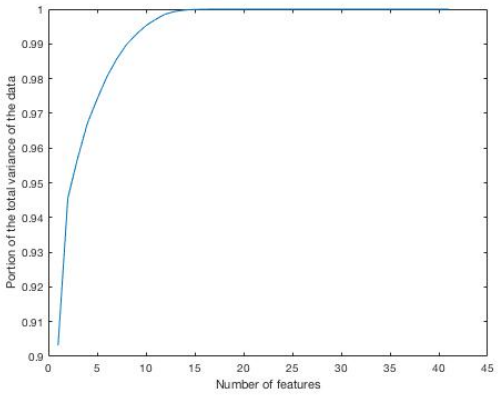

Figure 7: Cummulative portion of the total variance of the data as a function of number of features after PCA.

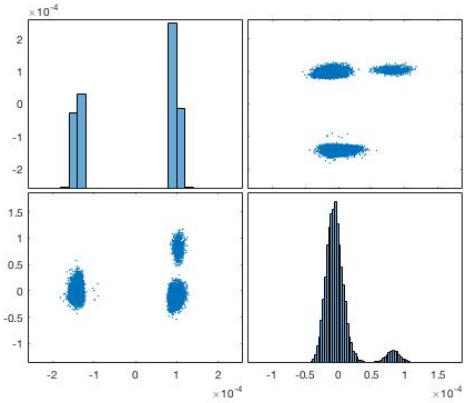

Figure 8: Values of features plotted against each other after PCA and the distributions of these values.

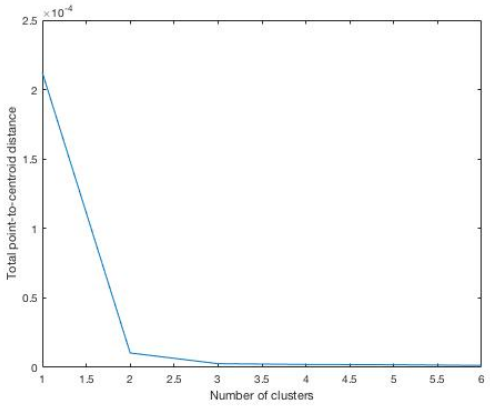

Figure 9: Total point-to-centroid distance as a function of number of clusters.

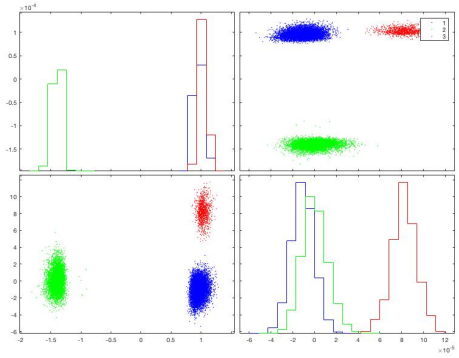

Figure 10: Values of features plotted against each other after PCA and the distributions of these values (same as figure 8) with color codes for different clusters added.

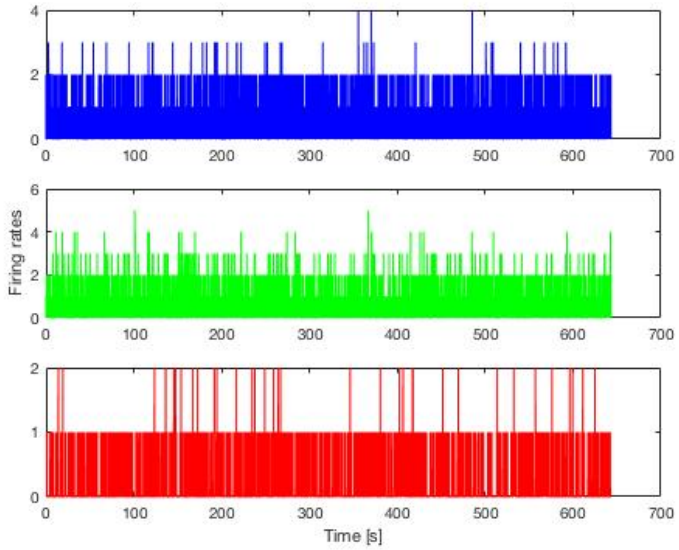

Figure 11: Firing rates for each neuron in 50 ms windows (same color codes as in figure 10).

3.2 Result from Regression analysis

3.2.1 Best regression model

As we described in the method section, we try to get the best MSE for each α comparing the results.

Alpha	train error X	test error X	λ X	train error Y	test error Y	λ Y
0	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.1	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.2	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.3	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.4	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.5	2.9644e-4	6.8881e-4	1.62231e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.6	2.9278e-4	6.8982e-4	1.34687e-4	1.7223e-4	3.3636e-4	1.1615e-4
0.7	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
0.8	2.9448e-4	6.8920e-4	1.47819e-4	1.7223e-4	3.3636e-4	1.1615e-4
0.9	2.9448e-4	6.8920e-4	1.47819e-4	1.7135e-4	3.3676e-4	1.0583e-4
1	2.9278e-4	6.8982e-4	1.34687e-4	1.7135e-4	3.3676e-4	1.0583e-4

Table 1: Errors and λ for each α

Table 1 shows the training errors, test error, and lambda for each α. From this, for α, we will use 0.6 in X and 0.5 in Y, and for λ, we will use 1.34E-04 in X and 1.06E-04 in Y.

When we use α mentioned above for each train data X and Y, figure 12) shows the relation between the number of features and λ. As you can see, the larger λ is, the smaller the number is.

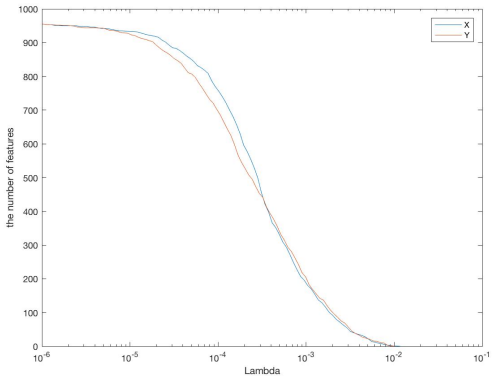

Figure 12: number of features and λ for X and Y

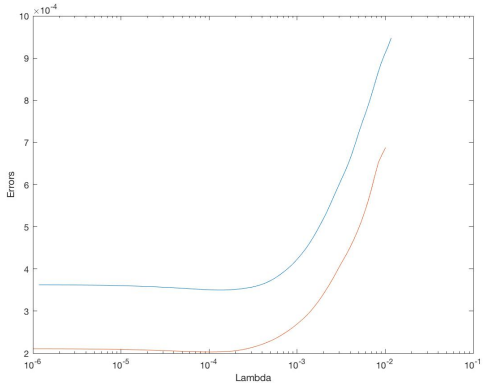

Figure 13: train errors and λ for X and Y

We also look at the relation between test errors and the number of features. figure 13) shows it. From this graph, if λ is big enough to eliminate so many features, the number of features would become to small to do regression analysis. As we see, the optimum point of the graph has come before the end of the curve, we can think the range for λ is enough in this case.

As the final output, figure (14) shows the projected trajectories of X and Y when we apply the parameters to **TEST** data.

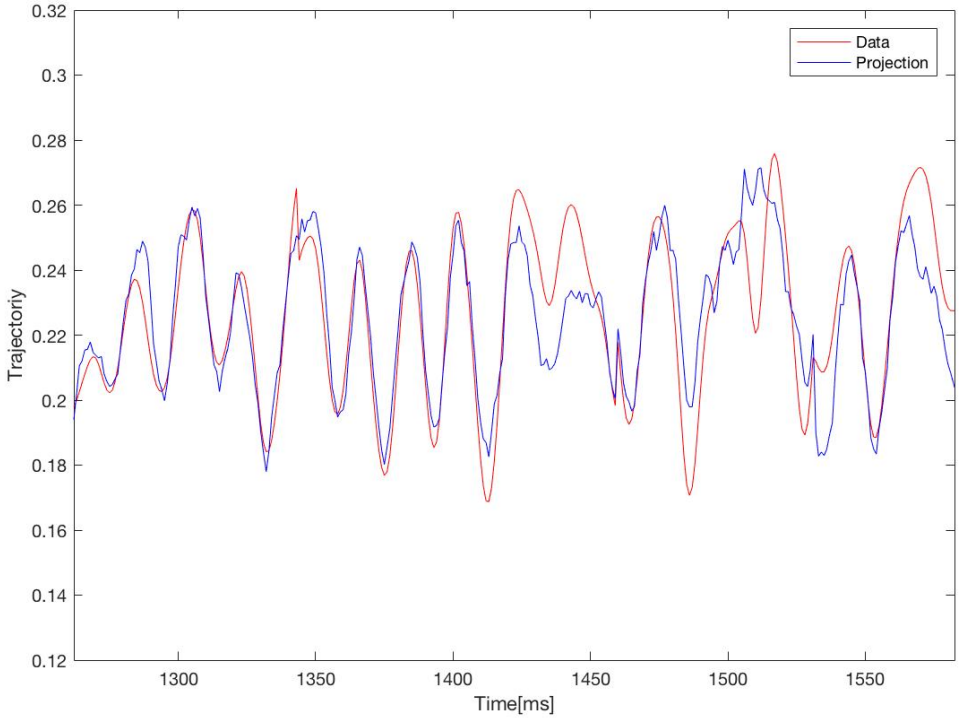

Figure 14: Projected and real trajectories of X

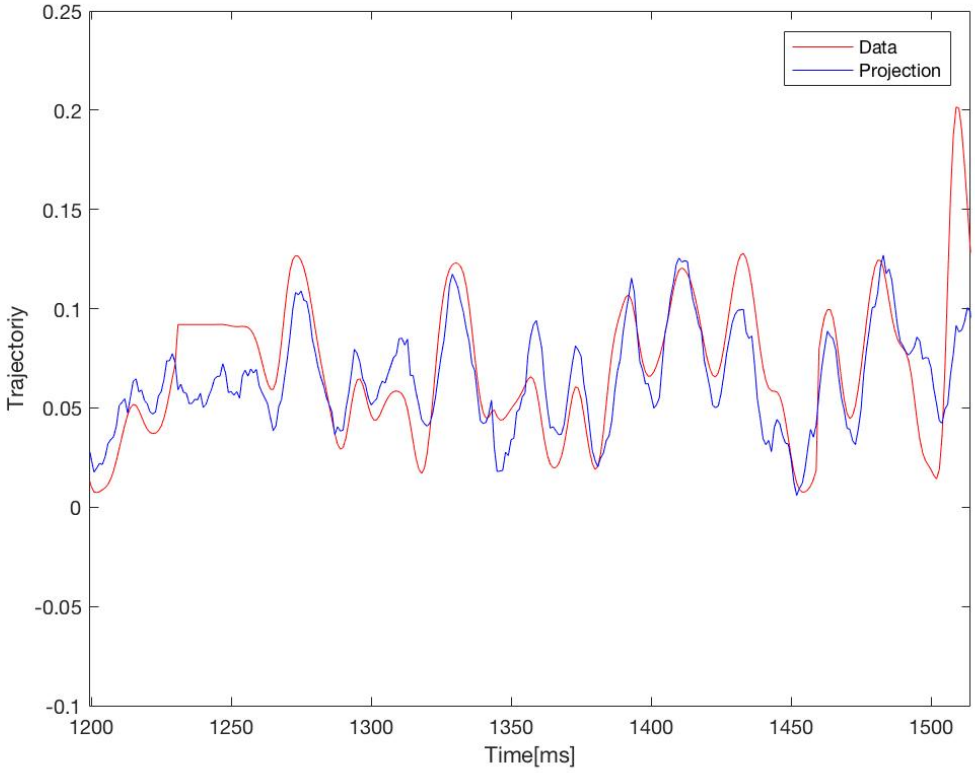

Figure 15: Projected and real trajectories of Y

4 Discussion

4.1 Data set

In the project, we have large data both in terms of time and features. Therefore, one of our objectives here will be to be able to reduce the computational time and eliminate the meaningless data using threshold or dimension reduction methods. As the signals cannot have the labels, we also analyze the data with unsupervised method. We did however use regression to learn the robotic arm position.

4.2 Recommended method

There are three things to discuss; dimension reduction, overfitting, and regularization weight.

About dimension reduction, as we have quite a large number of features, we should use a method which can reduce the number. For this, we can have two options, PCA and Lasso. As PCA converts features into different forms and our purpose in the project is detect neurons which have influence on the arm movement, we would perhaps like to keep the characteristics of features. Besides, just using the variance cut off might not be good enough, where do we know what amount of variance is sufficient? In that case, Lasso is the better as it does not change characteristics of the features.

About overfitting, to avoid this, we use regularization and cross-validation. We also think about penalty factors when calculating MSE.

About regularization, there are three options: Lasso, Ridge, and Elastic nets. In the data, features are based on 48 neurons and 20 features in each neurons are closely related. As Lasso can not consider the relation well, we should use Elastic nets. Also, ridge regression uses L2, but we are not sure whether L1 or L2 is more useful in our case! According to the result, Elastic nets was certainly chosen in our case as well.

Finally, we have to evaluate the regression model we have chosen. As we do cross validation in train data, choosing the best model with the least train error and applying it to test data will give us the best regression model. To understand to what extent the model is fitted, we can look at MSE, trajectories between original and projections.

4.3 Shortcomings

From figures (14) and (15), our projections are close to the real trajectories. However, if we project X and Y in the same graph, we can roughly see the accuracy of our model and see that the arm sometimes moves so further than the average. As one reasonable explanation could be that the spikes have some limitation and not enough to describe extreme movements. To compensate this, we possibly need another threshold to describe this. In addition, when we look at the number of features in the regularized regression model, even in the best case, we still have 409 features for X and 337 features for Y. As we set the threshold in finding spikes, we might as well set threshold for features of neurons as well. This would explain another problem that the number of features used for trajectories is more than three, which we have detected in the first process.

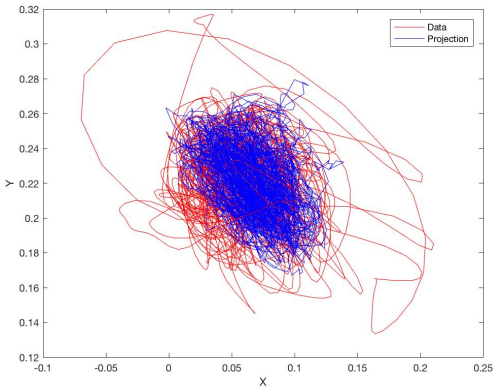

Figure 16: Plot for data and projections

4.4 Final pipeline and implementation of the control of robotic arm

Table (2) shows our hyper parameters for the final pipeline as we described in the recommendation. If we implement the control of robotic arm based on the electrical brain signals, we must see the relation between spikes and data, and those and the arm movements. As our model still have so many features and cannot explain all the trajectories, we need assumptions for implementation.

First, in some ways, we should reduce the number of features gained from the regression. If we obtain three possible neurons from this process. After that, Spikes and data set from neurons would have correlation. As we already have the links between neurons and movement, this would lead spikes describe the movement in the end.

X/Y	Method	Lambda	Alpha
X	Elastic Net	1.34E-04	0.6
Y	Elastic Net	1.06E-04	0.5

Table 2: Parameters for the final pipeline