Greg Kronberg
Machine Learning Final Project

Predicting behavioral state from the hippocampal extracellular field potential

Learning Problem

Predicting behavioral states from neural activity is a fundamental problem in neuroscience that has broad applications in medicine and technology. Here I have obtained multi-electrode extracellular voltage recordings from a rat hippocampus during periods of either rest or exploration of a maze. The goal of this report is to derive a set of features from this time series data and train a learning algorithm predict the target variable, whether the animal is at rest or exploring. The data were downloaded from the CRCNS data sharing website and provided by Loren Frank's lab at UCSF (https://crcns.org/data-sets/hc/hc-6).

The Data

The original data set contains time series extracellular voltage data recorded from 30 electrodes simultaneously in the rat hippocampus. These recorded data are separated into epochs and labeled based on the behavioral state of the animal during that epoch. For example, the first epoch contains a 60 second voltage recording from 30 electrodes during which the animal was asleep. The second epoch contains another 60 second recording from 30 electrodes in which the animal was exploring a maze. An example recording from both a rest and exploration epoch are displayed in figure 1. The data contain 36 such voltage recording epochs in total, each labeled as either "rest" or "run". The data could therefore be thought of as containing 36 potential training examples. As described below, I derived a set of features from the time series data recorded during each epoch/example.

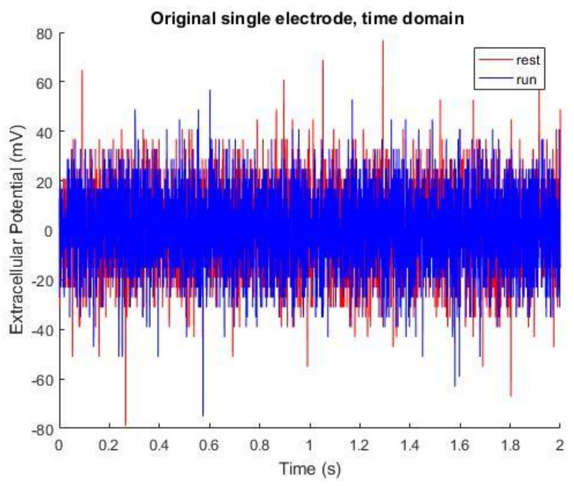

Figure 1. Original data from individual electrodes. Extracellular potential recordings from the same individual electrode during two example epochs, a rest epoch (red) and a run epoch (blue). Data were sampled at 1500 Hz

Approach

Dimensionality reduction

The data described above were sampled at a rate of 1500 Hz, therefore each epoch contained close to 100,000 samples in each of 30 electrodes. Therefore, to use each voltage sample as a feature would yield a feature space with ~3 million dimensions and only 36 examples. Feeding such a large number of features would make the learning algorithm highly susceptible to over fitting and would be computationally too expensive.

First, to reduce the number of spatial features I used principle components analysis (PCA). PCA was performed on the data during each epoch from all 30 electrodes. The data were then projected onto the first principal component, which is the linear combination of electrodes that explained the most variance in the data during that epoch. The time series of the first principle component during two example epochs are show in figure 2. All analyses were then done using only the first principle component from each epoch, reducing the number of spatial dimensions from 30 to 1.

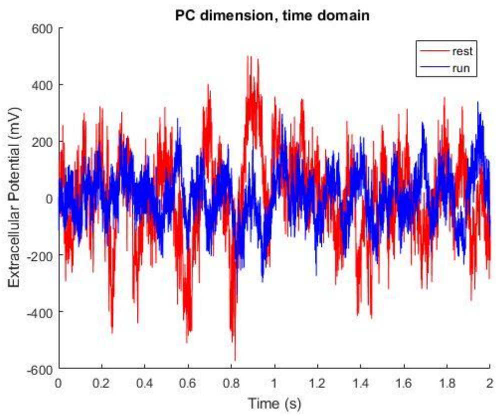

Figure 2. First principal component in the time domain. The projection of all recording electrodes onto the first principal component. The time domain signal is shown for two example epochs, during rest (red) and run (blue)

After reducing the number of spatial dimensions, each example epoch still had a large number of temporal features (~100,000). To reduce the number of features further, I used some prior knowledge of the neural correlates of exploration in rats. It is well established that hippocampal oscillations in the theta and gamma frequency range (~6-80 Hz) are associated with exploratory behavior in rats. I therefore wanted to use the magnitude of the signal (in the first PC for each epoch) at these frequencies as input features. To do this I used the fft function in matlab to convert each of the times series signals into the frequency domain. I kept frequency components in the theta and gamma range (up to 60 Hz) as potential features. These fft magnitudes during two example epochs are shown in figure 3. By performing PCA and taking advantage of prior knowledge about correlations, the potential feature space was narrowed down to 60 dimensions.

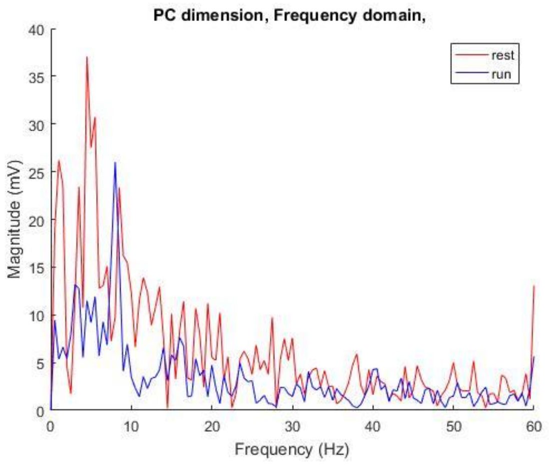

Figure 3. First principal component in frequency domain. The first principal component during each epoch was converted to the frequency domain with the fft function in MATLAB. Only frequencies below 60 Hz were included as potential features. Two example epochs are shown, during rest (red) and run (blue)

Feature Selection

After limiting the potential feature space to 60 dimensions, I needed to choose an appropriate combination of these features. To do this, I used a forward search algorithm, which iteratively added chunks of 5 consecutive frequencies (e.g. 0-5, 5-10, 10-15 Hz) and selected the best combination of features at each stage. After each iteration the combination of features was evaluated by leave-one-out cross validation, with the best feature set having the fewest number of misclassifications. If there was a tie between multiple feature sets, the set containing the lowest frequency components was chosen. While the forward search algorithm is not exhaustive and does not necessarily find the optimal combination of features, it a computationally tractable way to improve the feature vector. This feature selection process was repeated during each iteration of the model selection process described below in order to find the best combination of features and model parameters.

Model Selection

The learning problem, as I have posed it, is a binary classification with the classes being whether the animal is at rest or exploring. I chose to use a support vector machine (SVM), as SVM's perform well at binary classification and can be easily adapted using kernels to handle higher order feature spaces without much added computational cost. Here I took advantage of this by testing SVM models with polynomial kernels of varying degree up to fifth order. Another parameter choice of the SVM algorithm is the box constraint, which determines a penalization parameter for misclassified examples during training. Relaxation of the box constraint acts a means of regularization to prevent overfitting. Here I tested box constraints of 1, 10, and 100. To run the SVM algorithm I used the fitcsvm function in MATLAB, and all other parameters were left as the MATLAB default. To select the best model I iteratively tested all combinations of these parameters with each of the feature combinations described above. In each iteration the model was evaluated via leave-one-out cross validation, and the best model was chosen to have the fewest misclassifications. Figure 4 shows the algorithm's performance as a function of model parameters and feature vectors.

Findings

Performance Evaluation

In all cases, the algorithm was evaluated via leave-one-out cross validation. For each combination of features (frequency components up to 60 Hz) and model parameters (kernel polynomial order and box constraints of SVM algorithm), each of the 36 examples was iteratively used as a test of the model (trained on the other 35 parameters). Of these 36 tests, prediction error was quantified as the percentage of misclassifications. Features and model parameters were chosen to minimize this prediction error. The minimum error that was achieved by the best combination of features and parameters was ~38.9%. The optimal feature vector had 15 features, containing the magnitudes of the first PC during each epoch at integer frequencies from 1 to 15. The optimal SVM model was used a first order kernel and a box constraint of 1 (Figure 4)

To evaluate the performance of the best model against "chance," the model was fed randomly labeled features and the prediction error was determined as described above. This was iterated 5 times to account for variability in the random labels. The average classification error for chance labeling was 50.6% The model therefore performed better when fed properly labeled features (Figure 4), indicating that the features and the algorithm are informative when trying to predict the behavioral state of the animal.

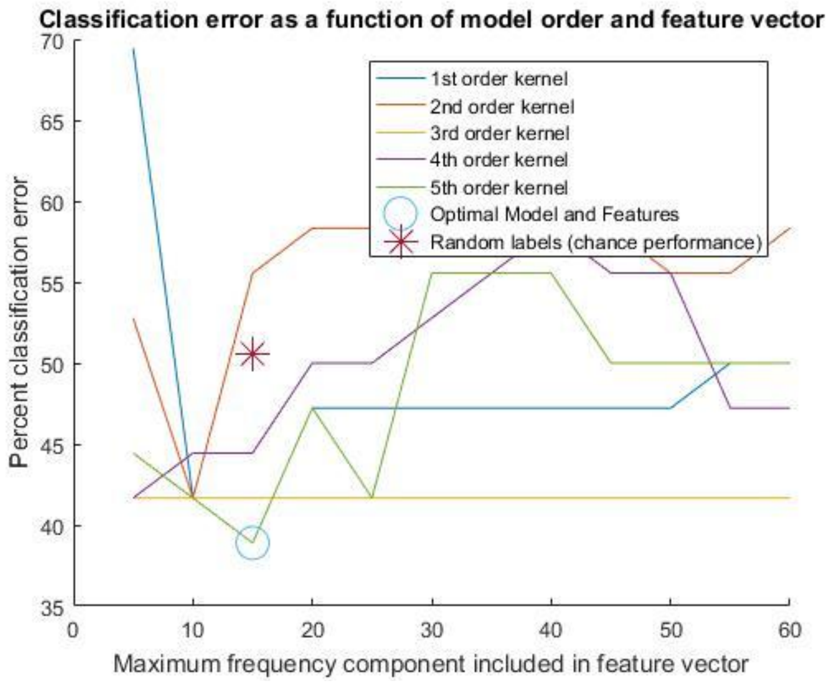

Figure 4. Model and feature selection. Classification error is plotted for various combinations of feature vectors and polynomial kernel orders for the SVM algorithm. All integer frequency components below the indicated frequency were included in the model. The optimal model order and feature vector is compared to chance model performance, where the model is randomly fed mislabeled data

Future Improvements

A simple improvement on this algorithm would be to increase the number of training examples available. For example, this data set was only recorded from a single animal over multiple days. Recording s from multiple animals may make the model more robust and prevent overfitting based on the specific neural dynamics of a single animal.

Another improvement may be to better choose the input features. The forward search used here does not exhaustively search all feature combinations, or find the optimal combination of features, as this is very computationally taxing. Similarly, I did not search the entire parameter space of box constraints for the SVM algorithm, as this would be impractical. However, given more time and computational power, performing a larger search of this parameter space may improve the algorithm's predictions.