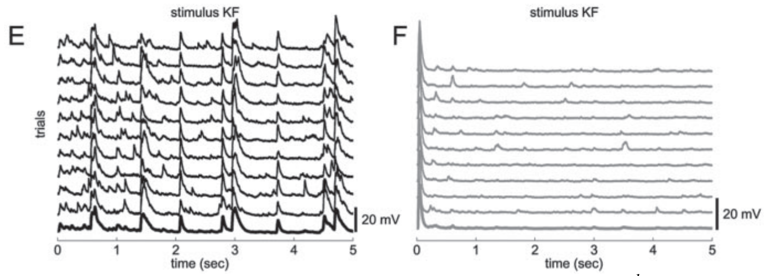
Kevin Nam Truong, MoorXu and Daniel Hawthorne

Neural Prediction Challenge: Primary Auditory Cortex

I. Introduction

The typical model of cortical processing assumes that it an inherently noisy, possibly degenerate, process. However, recent findings[1]have shown that individual cells in the primary auditory cortex of a rat could be described as a binary process, rather than a highly variable Poisson process. As Illustrated in figure 1, picturing neuron E and F's response in different trials to the same stimulus, a1 neurons are highly selective and consistent in their responses.

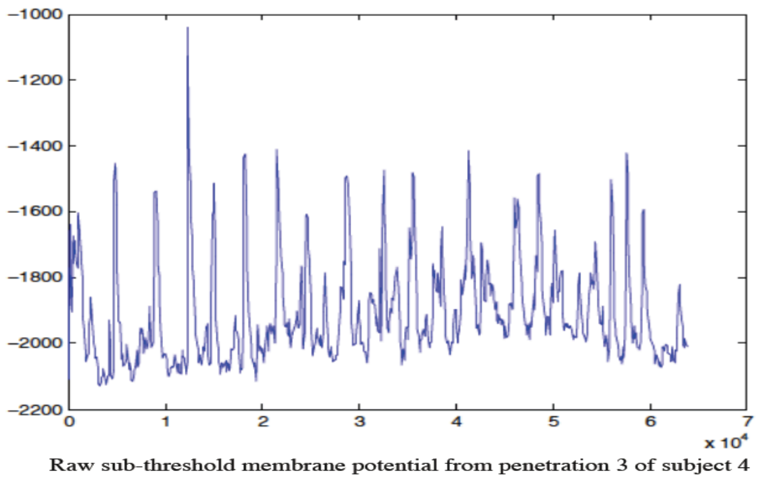


Neural responses of neuron E and F to stimulus KF[1]

Given the hypothesis that neural responses form consistent representations of the stimulus we endeavor to form a deterministic mapping between the neural responses to auditory stimuli.

A number of coding schemes such as rate coding, temporal coding, population coding, and sparse coding have been proposed to explain how continuous neural responses represent information. Recently in an extreme version of temporal coding Patrick Suppes et al.[2] found that they were able to distill the neural representation of spoken words through averaging the spike data across large swaths of auditory cortex, and then representing this signal with the superposition of a few sin waves. We endeavored to compare this method to a population coding scheme where the origin of the neural signal is leveraged as a feature.

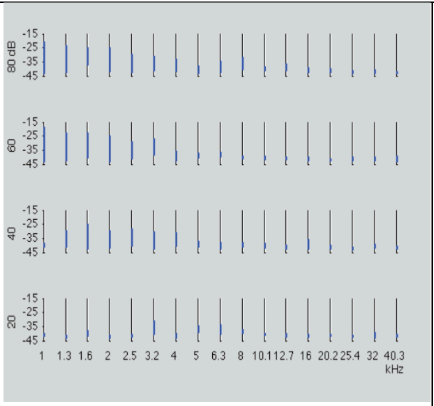
We used the attached single cell recordings of primary auditory cortex neurons of the *Sprague Dawley Rat* made available through the Collaborative Research in Computational Neuroscience (CRCNS)[3] organization's auditory cortex data set The first step was to process the data, pictured raw for one penetration below.



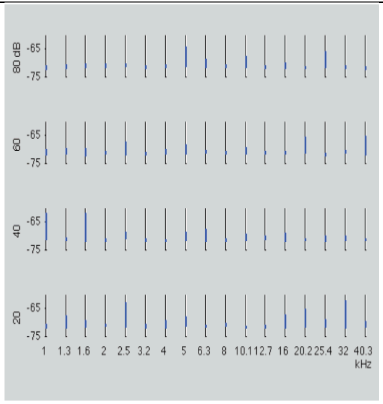
Raw sub-threshold membrane potential from penetration 3 of subject 4

To do this, we identified the location of each trigger in the neural response in order to divide the neural response into two portions: the spontaneous response (background neural signal) and the evoked response from the auditory trigger. We filtered out the neural signals that did not pertain to the trigger response, and isolated the portions of the signal that might have been induced by our trigger.

From our preprocessing we could determine whether the characteristic frequency for each penetration was unique, as we would expect given previous findings. We found that the characteristic frequencies were distinct as pictured below in the side-by-side tuning curves ($t=8.77$, $P < .001$).



Tuning Property of Penetration 1



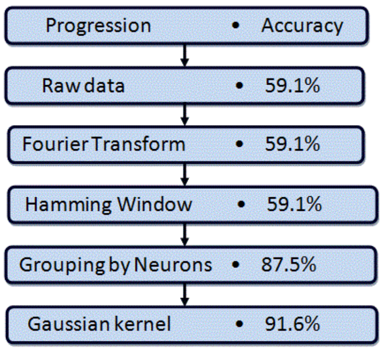
Tuning Property of Penetration 3

II. Support Vector Machines for Neural Prediction

In hopes of an accurate prediction, we initially ran a SVM using the evoked period neural traces as our training data. We aggregated the trigger frequencies into two classes -- low frequencies and high frequencies -- and we then did a linear two-class SVM using the Liblinear package in Matlab. This yielded an initial accuracy of 59.1%.

We then pursued a number of avenues to improve the accuracy of our SVM algorithm. We took the discrete Fourier transform of the neural trace to convert the data into the frequency domain. Testing Suppes' method in which he encapsulated the trace into a small subset of the highest amplitude frequencies, we took the five most prominent frequencies of the Fourier transformed neural trace. We then ran the two-class linear SVM using this as our training set. In order to correct for artifacts induced by the fast fourier transform, we applied a Hamming window function to the result. Training on the five highest spikes gave the same accuracy of 59.1% as training on the raw data.

To contrast with Suppes' averaging method we separated out the data by penetration (neuron) to allow the learning algorithms to do population-like coding. Grouping responses by specific neuron improved our our accuracy to 87.5% in the binary two-class case. We also tried using a Gaussian kernel for the SVM instead of the linear kernel. This was done using the LibSVM package for Matlab, and we achieved a slightly better accuracy of 91.6%.

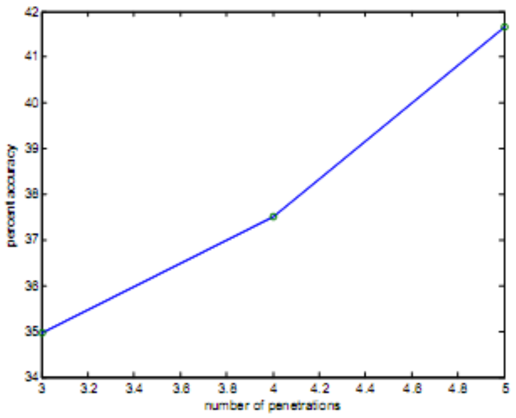


Summary of our binomial SVM implementations. Note that grouping by neurons induces the most significant rise in accuracy

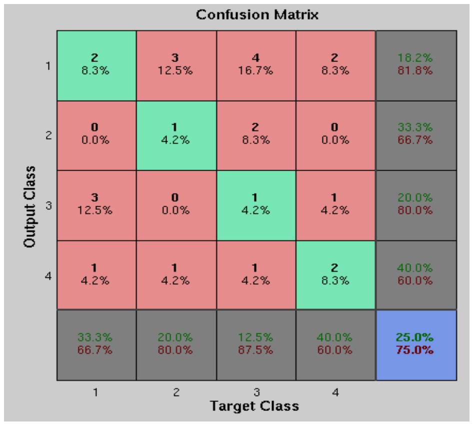
Up to now, we see that our algorithm can distinguish between low frequencies and high frequencies with fairly good accuracy. We wanted to expand our problem by separating the original 17 different pure tones into four frequency ranges. Using four classes and 4 classes and 5 penetrations we first attained an accuracy of 28.3%, not much larger than what would be expected by chance. With a larger data set we attained an accuracy of 37.5% using 4 classes and 5 penetrations, indicating that our hypothesis may have not yet converged, and that we are at least partly bottlenecked by the number of training samples.

In addition to using the raw neuronal response data or the most prominent frequencies, we also tried to predict trigger frequency using other features. For example, we computed the running average of the response trace and used this running average as our training examples. Heuristically, this running average captures the trend of the neuronal response trace, and indeed, we attained an accuracy of 40.8% using 4 classes and 5 penetrations

Using only three penetrations we achieved an accuracy of only 31.7% compared to the 40.8 we achieved with five. Including more penetrations increases the accuracy as indicated in the graph below. Even with the increased sample size (by a factor of 1.6) that was available when reducing the number of penetrations, our algorithm's accuracy was better using all five penetrations. This indicating that having multiple different neurons was more important than having more many training examples (even controlling for the difference in information contained in each training point in the two conditions.



Accuracy versus number of penetrations, which varies inversely with number of training samples



Confusion Matrix

Output Class \ Target Class	1	2	3	4	
1	**2** 8.3%	**3** 12.5%	**4** 16.7%	**2** 8.3%	18.2% 81.8%
2	**0** 0.0%	**1** 4.2%	**2** 8.3%	**0** 0.0%	33.3% 66.7%
3	**3** 12.5%	**0** 0.0%	**1** 4.2%	**1** 4.2%	20.0% 80.0%
4	**1** 4.2%	**1** 4.2%	**1** 4.2%	**2** 8.3%	40.0% 60.0%
	33.3% 66.7%	20.0% 80.0%	12.5% 87.5%	40.0% 60.0%	**25.0%** **75.0%**

Confusion matrix using four class labels and five penetrations

III. Experimental SVM

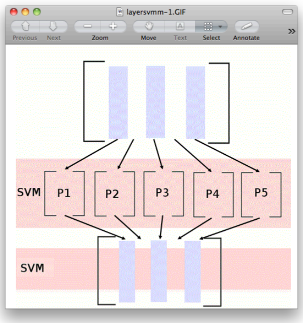
Even through all of our processing of the data, liblinear's SVM was unable to properly predict the tonal frequency from the neural signals. Across the 17 multiclass labeling of the tonal frequencies, liblinear was able to predict on average 4% of the test data using hold-out cross validation. Our test error is therefore 96%, and at the same time, our training error is surprisingly at 0%. This suggests that we were overfitting our data.

It is our goal to be able to better classify this multiclass labeling of tone frequencies, and we believe that there exist a better algorithm to process the data and extract features from it would significantly increase the accuracy and reduce our problem of overfitting. Our main motivation in further processing the data comes from the fact that each neuron has a different tuning curve. Instead of clumping all five neuron data into one matrix, it would be more natural to mimic the neurons and train on each separate neuron individually, and only after do we have five separate trained models do we aggregate them into a single model.

As a result, we attempt to add a new "layer" of SVM training to our data. In our outer most layer, we have our original data matrix (that contains the top 5 spikes in each of the 5 penetrations in each row) split into five separate matrices, one for each penetration area. We then train five SVM models, one for each penetration. After training the model, we extract the confidence vector of each penetration, which has size n, where n is the number of total labels we have. Each entry corresponds to a specific label (indicated by the label vector that we extracted), and contains a real number. The index to highest entry in the confidence vector is read into the same index in the model label vector, and it is this entry in the label vector that indicates what label the SVM predicts.

Normally, the SVM would only output the label with the highest confidence value; however, we believe that valuable information is lost this way. We instead normalized the magnitude of each entry in the confidence vector to range from 0 to 1, and raised every term to the fifth power (as a result from trial and error) to greatly reduce the weight of less confidence labels while keeping the high confidence ones at approximately the same value. We then normalized the labels vector such that the lowest frequency is replaced with the entry 1, the second lowest with entry 2, and so on. We now treat the confidence vector as a vector of weights with each weight corresponding to the same entry in the labels vector. We then do a weighted average of the normalized labels vector, which gives us a single number that indicates the best weighted guess influences not just by the most confident label, but also the second, third, and other less confident labels. We believe this number tells us more about each neuron's prediction of the tone than just finding the most confident label.

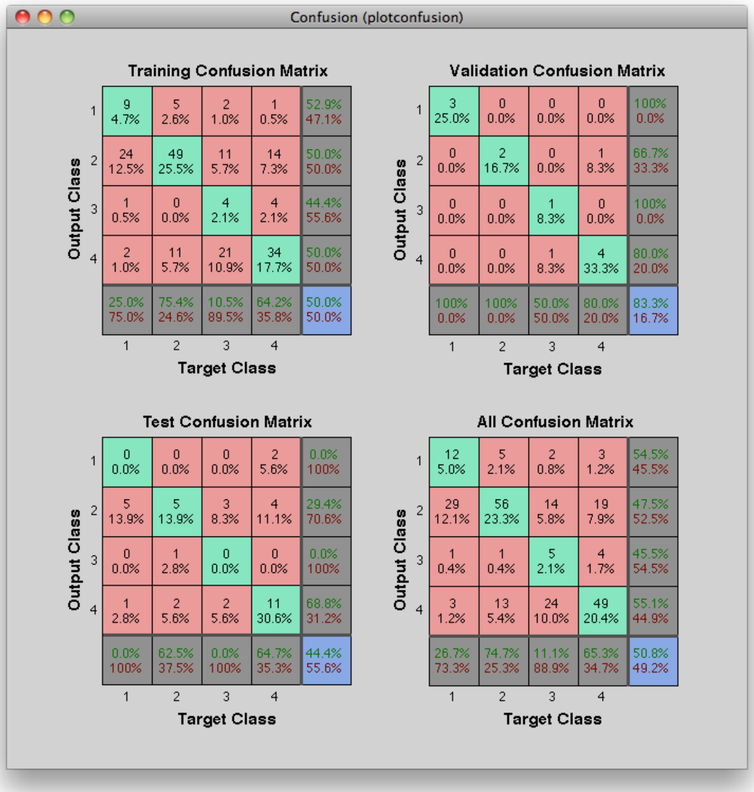
We repeat this for every m training samples, and eventually receive five m by 1 vectors, one for each penetration, with each entry corresponding to the best weighted guess of that particular training sample. We merge these five vectors into a single aggregate matrix. This aggregate matrix has five entries per row, each corresponding to a specific penetration area's best weighted guess. This matrix makes more intuitive sense to predict that the original data matrix because we are treating each neuron as a specific feature instead of treating each neuron's top 5 spike as a specific feature. In the latter case, we would have a matrix 25 entries (5 neurons, 5 spikes each), but this is actually five groups of five unordered features. libSVM cannot distinguish between ordered features and these unordered features within each penetration area, and thus we believe is a flaw in predicting the tonal frequencies.



As a result, our accuracy significantly improved from 4% to a peak of 30%. We believe this result to be considered impressive, as we are trying to classify 17 different labels from a sample of 80 training sets, where 56 of them are training samples and 24 are tests. That means on average we have three training samples per label, and this layered SVM was able to predict the correct label one fourth to one third of the time. Our testing error increased from 0% to an average of 35%, which indicates that we are no longer overfitting our data.

Although the numbers are still low in an absolute scale, we believe that it is due to the bottleneck of the number of our training samples. We attempted to further overfit and underfit the data with other various features, including the Gaussian kernel, and all returned with either the same or less predictive power. This idea is further supported by the fact that if we reduce the number of distinct labels (as in demarcating them into intervals, e.g. high frequencies and low frequencies), our accuracy significantly improves. Without this layered SVM, our accuracy of a 17 multiclass label is 4%; with a multiclass of 3 labels, our accuracy averages 40%, and with a binomial label, our accuracy rises to above 90%.

IV. Neural Networks



Confusion (plotconfusion)

Training Confusion Matrix

Output Class \ Target Class	1	2	3	4	
1	9 4.7%	5 2.6%	2 1.0%	1 0.5%	52.9% 47.1%
2	24 12.5%	49 25.5%	11 5.7%	14 7.3%	50.0% 50.0%
3	1 0.5%	0 0.0%	4 2.1%	4 2.1%	44.4% 55.6%
4	2 1.0%	11 5.7%	21 10.9%	34 17.7%	50.0% 50.0%
	25.0% 75.0%	75.4% 24.6%	10.5% 89.5%	64.2% 35.8%	50.0% 50.0%

Validation Confusion Matrix

Output Class \ Target Class	1	2	3	4	
1	3 25.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	2 16.7%	0 0.0%	1 8.3%	66.7% 33.3%
3	0 0.0%	0 0.0%	1 8.3%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	1 8.3%	4 33.3%	80.0% 20.0%
	100% 0.0%	100% 0.0%	50.0% 50.0%	80.0% 20.0%	83.3% 16.7%

Test Confusion Matrix

Output Class \ Target Class	1	2	3	4	
1	0 0.0%	0 0.0%	0 0.0%	2 5.6%	0.0% 100%
2	5 13.9%	5 13.9%	3 8.3%	4 11.1%	29.4% 70.6%
3	0 0.0%	1 2.8%	0 0.0%	0 0.0%	0.0% 100%
4	1 2.8%	2 5.6%	2 5.6%	11 30.6%	68.8% 31.2%
	0.0% 100%	62.5% 37.5%	0.0% 100%	64.7% 35.3%	44.4% 55.6%

All Confusion Matrix

Output Class \ Target Class	1	2	3	4	
1	12 5.0%	5 2.1%	2 0.8%	3 1.2%	54.5% 45.5%
2	29 12.1%	56 23.3%	14 5.8%	19 7.9%	47.5% 52.5%
3	1 0.4%	1 0.4%	5 2.1%	4 1.7%	45.5% 54.5%
4	3 1.2%	13 5.4%	24 10.0%	49 20.4%	55.1% 44.9%
	26.7% 73.3%	74.7% 25.3%	11.1% 88.9%	65.3% 34.7%	50.8% 49.2%

Confusion Matrix for Neural Network

A standard three-level, feed-forward, back-prop neural network was also used to classify the neural signals. A number of different feature schemes were used (each representing a different way of coding the neural data). The best network used a variant of a standard running-average measure of the membrane potential, taking into account the origin of the signal (which neuron the signal was from). This neural nework had an average training error of 40. The confusion matrix for this network is shown below. The comparable training and testing errors indicates that the algorithm is able to generalize well, which was not the case with alternative coding schemes.

V. Conclusion

Consistent with previous findings we determined that different penetrations in A1 had unique characteristic frequencies, and their tuning curves were determined. First in a test of Suppes' theory we averaged across neural signals and represented the responses as the primary frequencies of the spectral decomposition and used these frequencies to predict the stimuli that generated the neural responses. This method was unsuccessful at predicting the stimuli. Utilizing the unique characteristic frequencies of each penetration is a population code yielded significantly better results, and characterizing the neural trace's with a modified running average was more useful than the five best frequencies of the spectral decomposition. All neural signal coding schemes were used to characterize the neural response in order to predict the stimuli that elicited them with both support vector machines and neural networks. As more data and penetrations were added the results of both algorithms increased indicating that with more penetrations and data a better estimate of the initial stimulus is possible.

References

[1]DeWeese, M., Wehr, M., Zador, A. "Binary Spiking in Auditory Cortex." *Journal of Neuroscience Research.* (2003)
[2]Suppes and Han.Brain-wave representation of words by superposition of a few sine waves. Proceedings of the National Academy of Sciences of the United States of America (2000) vol. 97 (15) pp. 8738
[3]M.,Zador, A. "Linearity of cortical receptive fields measured with natural sounds." *Journal of Neuroscience Research.* (2004)
[4]R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and **C.-J. Lin**. LIBLINEAR: A library for large linear classification *Journal of Machine Learning Research* 9(2008), 1871-1874.