Technical Report: Building a Neural Ensemble Decoder by Extracting Features Shared Across Multiple Populations

Chia-Jung Chang
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
`chiajung@mit.edu`

ABSTRACT

To understand whether and how a certain population of neurons represent behavioral-relevant variables, building a neural ensemble decoder has been used to extract information from the recorded activity. Among different ways to decode neural ensemble activity, the parametric approach requires assumption of the spiking distribution and an underlying encoding model, which poses challenges for neurons with nonlinear, multi-modal, and complex receptive fields. Alternatively, non-parametric framework assumes no explicit probability distribution and discovers patterns from the data in an unbiased way, and thus training a machine learning model as a decoder has gained its popularity in the field. However, machine learning models require a big-enough dataset, yet the data size is often small due to limitations in recording techniques. Although increasing the number of subjects help increase the size of the overall training set, how to concatenate recorded ensemble activity across subjects while preserving their spatial-temporal structures is not trivial. In this technical report [1], a novel way to extract features shared across populations from multiple subjects to train a machine learning model is described. With this feature extraction framework, one can easily test upon different hypothesis of the underlying coding strategies. In addition, several common issues in applying a machine learning model to decode neural activity has been discussed. Overall, this report provides a rigorous protocol for applying machine learning models to decode a relatively small dataset - neural ensemble activity collected across multiple populations.

Keywords ensemble decoder · feature extraction · small datasets · model selection

1 Introduction

To understand whether and how a neural population contributes to a certain behavior, one way is to modify cell activity through electrical or optical manipulations to test causality, yet manipulation tools are not applicable to every experiment and have their limitations in revealing the underlying coding strategies. Therefore, a complementary method is to study neural correlates of behavior, in other words, to see how different aspects of behavioral variables such as sensory stimuli, motor actions, and internal states can be decoded from the neural activity. From the decoding performance (goodness-of-fit) or its extracted mutual information, we can quantify the degree of the neural population contributing to the behavior, investigate how neural coding evolves across time or varies between contexts, and compare functional roles between different brain regions in the same behavior.

To decode behavior from the neural activity, a popular method is to build a parametric model to capture each neuron's dynamics either through an integrate-and-fire biophysical model or Poisson point process, and use Bayesian approach to further represent population codes under uncertainty. However, neurons often fire in a non-Poisson fashion and their response also depends on their spiking history. Moreover, using Bayesian approach further assumes explicit probabilistic distribution. Therefore, a parametric decoder would not work well for neurons with nonlinear, multi-modal and complex receptive fields.

[1] This technical report only serves as archiving a specific proposed framework. Application examples on different datasets and their evaluation analysis will be described in the full paper

Alternatively, building a non-parametric decoder assumes no explicit probability distribution and let data speaks for itself. To achieve this, machine learning models are used as they can be trained to learn the population codes from heterogeneous responses across neurons in an ensemble. While recent interests grow in applying deep learning frameworks such as convolutional neural networks (CNN) and recurrent neural networks (RNN), such frameworks are only suitable in analyzing large-scale datasets. For smaller datasets, traditional machine learning models such as support vector machines (SVM) and random forest (RF) are still very useful in many scenarios. Nevertheless, a machine learning model still benefits from more training data in order to correctly sample the underlying distribution.

To satisfy the essential requirement of data size in training a machine learning decoder, recordings are performed across multiple subjects to construct a larger database. For example, one can use a silicon probe to record multiple single neurons from the same animal while performing a behavioral task. The number of recorded neurons are usually around 20 to 30 after sorting, which also varies across sessions. Although one can use a chronic array implant or even multiple probes to increase the number of recorded neurons within the same session, more damage is also caused. On the other hand, longitudinal imaging creates opportunity for collecting large-scale data from the same animal, yet imaging from the deep brain through a GRIN lens still faces the same challenge in terms of data size. Therefore, data will often times recorded from multiple animals performing a same behavioral task. Imagine 10 cells are recorded from one animal across 60 trials, and 40 cells are recorded from another animal across 15 trials, how to combine these data to train a decoder is thus non-trivial. A common practice is to obtain an average response (mean firing rate or z-score) of a population from each recording session, or to categorize individual neurons based on their tuning curves or dynamic profiles. These approaches either lose the temporal resolution or ignore the possibility that information is mixed and distributed across the entire ensemble. In this technical report, a method to extract ensemble features shared across multiple sessions and multiple subjects without losing their temporal structures is proposed.

2 Method

A machine-learning algorithm is developed to extract features from multiple neural ensembles and combine them to form a dataset large enough to train the decoder. The core model is built on top of a framework described in [1], originally developed to categorize vocalizations by transforming acoustic signals into spectrograms, and extracting features from 2D images with varying sizes. This algorithm can be further generalized to different experiments or different recording techniques. In this section, the problem formation and its applicable dataset characteristics is defined first, and how to properly extract shared ensemble features across multiple populations is also discussed. Furthermore, a rigorous protocol for model section based on bias-variance trade-off is presented.

2.1 Dataset Characteristics

Due to technical limitations or intrinsic sparsity in the target brain region, it is sometimes difficult to obtain a large-enough neural dataset from the same animal and will require pooling data across multiple animals. Depending on the experimental design, neural signals are recorded from across several animals performing a behavioral task, where each animal undergoes either one or multiple sessions, with each session consisting of either a fixed number or random number of trials, with each trial duration being either fixed or varied.

Problem Formulation There are n animals in total. Each animal performs a k-way decision-making task. Total number of single neurons collected from animal i can be denoted as N_i, and each neuron can be denoted as U_{ij}, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., N_i$.

To simplify, we assume that the same set of neurons are collected across multiple sessions from the same animal utilizing a same behavioral strategy, and thus all the trials presented to one animal can be combined into one session in theory. Each animal i are presented with S_i trials in total, where a trial t_i has a duration of T_{t_i} timestamps, where $t_i = 1, 2, ..., S_i$ and T_t can either be fixed across trials or varied on a trial-by-trial basis.

For each trial, its associated behavioral outcome can be labeled as l, where $l = 0, 1, ..., k - 1$. For example, in a Go/NoGo task, labels for NoGo trials are 0, whereas the labels for Go trials are 1.

Response profile for neuron U_{ij} in trial t_i can be denoted as a row vector $R_{t_i j}$, and thus raw input data associated with each trial t_i can be represented as a 2D matrix of size $N_i \times T_{t_i}$, where $t_i = 1, 2, ..., S_i$. In a more general case, T_{t_i} can vary on a trial-by-trial basis. In cases where a trial structure is fixed, or timestamps can be padded for shorter trials to reach the same duration T, the input data for animal i can be represented as a 3D matrix of size $N_i \times T \times S_i$, where response profile of each neuron $R_{t_i j}$ can either be a vector of binary numbers to represent spiking, or a vector of non-negative numbers to represent firing rate or fluorescence intensity.

Example 1 Animals are trained to perform a Go/NoGo task depending on tone frequency, and each session ends when their performance surpasses a user-defined threshold. Single neurons are recorded with 1 msec resolution through a silicon probe with multiple channels. There are 3 animals in total. 10 single cells are recorded from one animal within a session of 60 trials, 15 cells are recorded from another animal within a session of 35 trials, and 40 cells are recorded from the other animal within a session of 15 trials.

Example 2 Animals are put in a T-maze and can freely choose between left and right arms, and each session ends when number of choices reaches 50. Single neurons are imaged at 10 Hz through a head-mounted miniscope. There are 3 animals in total. 50 cells are imaged from one animal running fast such that each trial lasts below 10 seconds, 70 cells are imaged from another animal running slow such that most trials last beyond 1 minute, and 30 cells are imaged from the other animal whose running speed varies across trials.

Example 3 Animals are put in a box and trained to run toward a port to obtain a drop of delivered fluid after hearing an auditory cue. After conditioning, the delivered fluid will be mixed with alcohol in behavioral sessions, where each session lasts an hour. There are 7 animals addicted to alcohol in a similar way. Single neurons are imaged at 10 Hz through a head-mounted miniscope. Calcium activity for each neuron on each trial is aligned to the first lick after fluid delivery, and trial structure is defined from 8 seconds before to 8 seconds after the lick, so the trial duration is fixed. However, the number of trials and the number of neurons vary across animals.

As revealed by above examples, how to properly combine data across animals and decode ensemble information underlying the decision-making task is non-trivial, and will be discussed in Section 2.2. After data across animals is combined into a feature set $\mathbf{D}$, it is split into two sets: prediction set and development set. Development set is further subdivided into a training set and a validation set for cross-validation in training. A supervised decoder algorithm will involve feature extraction and a machine learning classifier. An example framework for Example 3 is shown in Figure 1.

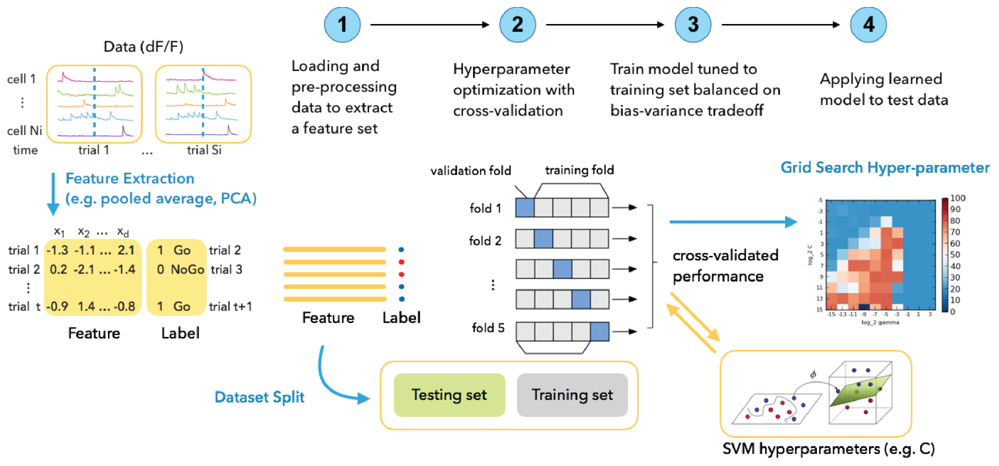
1
2
3
4
Data (dF/F)
cell 1
cell Ni
time
trial 1
trial Si
Loading and pre-processing data to extract a feature set
Hyperparameter optimization with cross-validation
Train model tuned to training set balanced on bias-variance tradeoff
Applying learned model to test data
Feature Extraction (e.g. pooled average, PCA)
x1 x2 ... xd
trial 1 -1.3 -1.1 ... 2.1
trial 2 0.2 -2.1 ... -1.4
trial t -0.9 1.4 ... -0.8
1 Go trial 2
0 NoGo trial 3
1 Go trial t+1
Feature
Label
validation fold
training fold
fold 1
fold 2
fold 5
cross-validated performance
Grid Search Hyper-parameter
log_2 C
log_2 gamma
Dataset Split
Testing set
Training set
SVM hyperparameters (e.g. C)

Figure 1: A decoder framework diagram.

2.2 Feature Extraction

While it is a relatively easy task to extract features from a same population, it is non-trivial to extract features shared across multiple populations for data concatenation. A few different ways based on different underlying coding models are discussed below.

Approach 1 Let's start from the simplest method. Obtain an average activity within a trial window across neurons r_{t_i} for each animal, such that data from all the trials across all the animals can be combined into a feature set $\mathbf{D}$ with size shown in Equation 2 (feature length = 1). This method is applicable when neurons have simple response profiles that are easily segregated by their tuning magnitude to the stimuli.

$$r_{t_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} \overline{R_{t_i j}} \tag{1}$$

$$\mathbf{D} := \sum_{i=1}^{n} S_i \times 1 \tag{2}$$

Approach 2 In Approach 1, too much information is lost. One can recover more information by adding other types of statistics in addition to average response. For example, obtain minimum, maximum, mean and covariance in the response profiles across neurons within a trial, such that data from all the trials across all the animals can be combined into a feature set $\mathbf{D}$ with size shown in Equation 3 (feature length = 4). By adding more statistical measurements such as second-order momentum, the length of feature set increases and the coding dynamics can be captured better.

$$\mathbf{D} := \sum_{i=1}^{n} S_i \times 4 \tag{3}$$

Approach 3 When response profiles are complex, above approaches are most likely to fail. One way is to obtain an average response profile across time across neurons $\hat{r}_{t_i}$ for each trial, which can be thought as neuronal population vector. When the trial duration is fixed ($T_{t_i} = T$ for $i = 1, 2, ..., n$), data from all the trials across all the animals can be combined into a feature set $\mathbf{D}$ with size shown in Equation 5 (feature length = T).

$$\hat{r}_{t_i} = \sum_{j=1}^{N_i} R_{t_i j} \tag{4}$$

$$\mathbf{D} := \sum_{i=1}^{n} S_i \times T \tag{5}$$

Approach 4 To further extend features extracted from Approach 3, one can add standard deviation of response profile across time into features, the final feature set $\mathbf{D}$ will double its feature length, as shown in Equation 6 (feature length = $2T$).

$$\mathbf{D} := \sum_{i=1}^{n} S_i \times 2T \tag{6}$$

Approach 5 Above approaches ignore information distributed across neurons. While Approach 1 and 2 ignore temporal dynamics across time, Approach 3 and 4 are prone to stochastic noise in dynamic profiles. One way to solve these is to extract features by only preserving the most relevant components. By assuming that relevant features are preserved in a similar way across subjects, we can first reduce the temporal dimension of neuronal activity linearly using either PCA or non-linearly using Isomap down to D_t. We can then reduce the dimension of ensemble activity down to D_u to extract codes shared across neurons in a population. By selecting a fixed dimension number explaining the major variance in the data (D_t and D_u), feature set for each animal is ensured and can be concatenated easily into a feature set $\mathbf{D}$ with size shown in Equation 7. This 2D feature set $\mathbf{D}$ can be further flattened into a 1D feature vector.

Although many studies would normalize data before applying PCA, it is not necessarily reasonable for neural data analysis. Considering that the amplitude difference among neurons may also carry information, yet one does not want to capture components referring to variance in mean response, a soft normalization as in Equation 8 is recommended. Overall, this approach is used to extract shared features across subjects in the example algorithm in Figure 1.

$$\mathbf{D} := D_u \times D_t \tag{7}$$

$$\mathbf{D} = \frac{\mathbf{D}}{\sqrt{\| \mathbf{D} \|}} \tag{8}$$

Approach 6 There is a basic assumption in Approach 5 that components extracted for each subject are of similar scales and coordinates. This however is not always true, and might make two identical dynamics diverge depending on the projection plane. To extend Approach 5, one can first concatenate all data across all the trials across all the animals first, as long as all the trials are of the same length. New shared coordinates are established through reducing dimensions on this concatenated dataset, and then individual dataset for each trial is projected onto these coordinates to extract neural features for each trial.

2.3 Model Construction

Take Example 3 described in Section 2.1 as instance. A decoder to separation between two output classes is needed, and can be achieved by finding a hyperplane that had the largest distance to the nearest training data point (margin) through support vector machine (SVM) algorithm. For a k-way decision making task, we can also apply C-supported vector classification method [2] with pairwise one-against-one strategy for multi-class classification.

The final feature set with outcome labels is first split into two sets: development set and test set. Development set is further split into two sets: training set and validation set. Given a training feature set $\mathbf{D}$, where each row corresponds to ensemble feature with length D_t for a trial, and a label vector $\mathbf{L}$ with length D_u. If the training set is highly unbalanced for its outcome labels, a subset of training set is sampled where the number of trials is matched between different outcome label classes. Take Example 3 as instance, collection trials are usually more common than no-collection trials, and thus it is suggested to select a subset of collection trials to match the number of no-collection trials to avoid bias in training, so as to yield an empirical chance level. C-SVC algorithm [2, 3] finds the hyperplane to solve the optimization problem below.

$$\min_{\mathbf{a}} \frac{1}{2}\mathbf{a}^T Q\mathbf{a} - \mathbf{v}^T\mathbf{a} \tag{9}$$

$$Q_{pq} = L_p L_q \cdot K(D_p, D_q) = L_p L_q \cdot \phi(D_p)^T \phi(D_p)$$

$$\text{subject to } \mathbf{L}^T\mathbf{a} = 0, \quad 0 \le \mathbf{a} \le C, \quad p = 1, 2, ..., D_t$$

where $\mathbf{v} = [1, 1, ...1]^T$ is the vector of all ones, Q is a positive semi-definite matrix, K is the radial basis kernel function, $\phi(x)$ maps x into a high-dimensional space, and C is the hyper-parameter.

To select proper parameters γ for kernel function and the hyper-parameter C, a grid search is applied. Five-fold cross validation can be used. The summit of the contour suggested the parameter pair for maximal cross-validated accuracy. After optimization, the trained decoder can be applied to test dataset.

References

[1] Chia-Jung Chang. Automated classification of marmoset vocalizations and their representations in the auditory cortex. Master's thesis, Johns Hopkins University, 2014.

[2] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[3] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 6(Dec):1889–1918, 2005.