Although the existence of correlated spiking between neurons in a population is well known, the role such correlations play in encoding stimuli is not. We address this question by constructing pattern-based encoding models that describe how time-varying stimulus drive modulates the
expression probabilities of population-wide spike patterns. The challenge is that large populations may express an astronomical number of unique patterns, and so fitting a unique encoding model for each individual pattern is not feasible. We avoid this combinatorial problem using a dimensionality-reduction approach based on regression trees. Using the insight that some patterns may, from the perspective of encoding, be statistically indistinguishable, the tree divisively clusters the observed patterns into groups whose member patterns possess similar encoding properties. These groups, corresponding to the leaves of the tree, are much smaller in number than the original patterns, and the tree itself constitutes a tractable encoding model for each pattern. Our formalism can detect an extremely weak stimulus-driven pattern structure and is based on maximizing the data likelihood, not making a priori assumptions as to how patterns should be grouped. Most important, by comparing pattern encodings with independent neuron encodings, one can determine if neurons in the population are driven independently or collectively. We demonstrate this method using multiple unit recordings from area 17 of anesthetized cat in response to a sinusoidal grating and show that pattern-based encodings are superior to those of independent neuron models. The agnostic nature of our clustering approach allows us to investigate encoding by the collective statistics that are actually present rather than those (such as pairwise) that might be presumed.

1 Introduction

A central question of neuroscience is whether neurons within a population encode stimuli independently or collectively, as complex patterns of action potential activity. One way to address this is to compare encoding models based on independent spike trains, with encoding models based on patterns of spiking across the entire population. Stimulus encoding by individual neurons is routinely quantified by fitting probability models to their spike trains (Brown, Nguyen, Frank, Wilson, & Solo, 2001; Truccolo, Eden, Fellows, Donoghue, & Brown, 2005). In contrast, quantifying how a neuronal population uses patterns of spikes to encode has been less feasible, if only because a population of \( N \) neurons may express any one of \( 2^N \) patterns at any given time. Although it is likely that the number of patterns \( M \) that are actually observed is far smaller (\( M \ll 2^N \)), an approach that directly models encoding by each individual pattern would still have far too many parameters to accurately fit. Thus, most techniques for modeling population spiking have considered neurons either as independent or as coupled (through second- or higher-order connections). Both Ising models (Martignon et al., 2000; Schneidman, Berry, Segev, & Bialek, 2006; Tang et al., 2008), and generalized linear models (GLMs) conditioned on cross history terms (Okatan, Wilson, & Brown, 2005; Pillow et al., 2008;
Truccolo, Hochberg, & Donoghue, 2010) are examples of the latter approach. However, neither of these model stimulus-driven patterns, where pattern is defined across neurons in the same time bin.

To clarify this, consider three neurons A, B, and C and a stimulus S. In principle, one would wish to model the joint distribution of these neurons, conditioned on their past spiking and on the stimulus:

$$P_{A_t, B_t, C_t | A_{t-1}, B_{t-1}, C_{t-1}, S_t}.$$ (1.1)

Ising models ignore the conditioning on stimuli and past spiking history and simply describe the stationary (stimulus and history averaged) joint probability.

$$p^{Ising} = P_{A_t, B_t, C_t}.$$ (1.2)

The reason for this is that because Ising models wish to capture the joint distribution of spiking within the same time bin, the Ising probability distribution has to be explicitly normalized by calculating a partition function. This requires summing terms over all $2^N$ possible patterns. Although stimuli or the previous spiking history of other neurons can be formally included in Ising models, the partition function then becomes stimulus or history dependent and computation time increases drastically. Thus, Ising models generally do not include stimulus drive or spike history.

GLMs do include both spike history and stimuli by assuming that neurons influence each other only with a time lag of at least one bin. The neurons are thus conditionally independent, and the joint distribution is approximated by a product of conditionally independent single-neuron models:

$$p^{GLM} = P_{A_t | A_{t-1}, B_{t-1}, C_{t-1}, S_t} \times P_{B_t | A_{t-1}, B_{t-1}, C_{t-1}, S_t} \times P_{C_t | A_{t-1}, B_{t-1}, C_{t-1}, S_t}.$$ (1.3)

Thus, although GLMs model the effect of stimulus drive to each neuron, they do not model the joint probability of spikes within the same time bin. That is, they do not model spatial patterns of spikes across the population.

The goal of this article is to define a probability model and fitting methodology that captures the relation between stimulus and spike patterns defined within the same time bin. That is, we wish to model the full joint distribution conditioned on the stimulus: $P_{A_t, B_t, C_t | S_t}$. In this article, we do not explicitly consider spike history or past stimuli (more on this
in section 4) but note that these could formally be included by defining $S_t \equiv \{S_{1,t}, A_{1,t-1}, B_{1,t-1}, C_{1,t-1}\}$. It should also be noted that both Ising models and GLMs are second-order models, and recent studies have indicated that second-order models are likely to be insufficient for capturing the pattern statistics of neural populations, particularly as the populations being recorded from grow in size (Roudi, Nirenberg, & Latham, 2009; Ganmor, Segev, & Schneidman, 2011). Hence, we would also like the formalism to allow statistics of arbitrary order (if they are present in the data) and be applicable to populations of arbitrary size.

To accomplish this, we proceed via the insight that some patterns may convey statistically identical information about a stimulus. The brain is a highly redundant system that processes information in a manner robust to noise (Shadlen & Newsome, 1994; Jacobs et al., 2009; Chelaru & Dragoi, 2008). Thus, if one spike is dropped from a $N \gg 1$ neuron pattern, it could matter for encoding, but very likely it does not, and it is even more likely that one does not possess sufficient experimental data to make such a distinction anyway. Guided by this intuition, we propose that patterns be grouped into as e to f clusters. For all patterns within a given cluster, the effect of any stimulus on the pattern probability is modeled identically. That is, instead of modeling each pattern parametrically, we make parametric models of the clusters.

Since it is possible that patterns that appear very different may encode similar information, we do not presume that spiking patterns that appear similar must be clustered together. Instead we use a regression tree to group patterns so that the data likelihood is maximized (Breiman, Friedman, Olshen, & Stone, 1984; Quinlan, 1993). The regression tree recursively splits the observed patterns into subgroups such that the expression probability of each pattern within a subgroup covaries with the stimulus in an identical manner. Thus, the grouping does not depend on the patterns themselves being similar, although this can be reintroduced by adding a regularization term dependent on the Hamming distance. When clusters are allowed to be defined by the data rather than by one’s preconceptions, pattern encoding models can be generated irrespective of the order of correlative (between neurons) statistics present in the data.

Regression tree–generated pattern clusters constitute a true population encoding model. Using the data likelihood as a measure of encoding (goodness of fit), they can be used to determine if stimuli are encoded by independent neurons or collectively driven patterns. We demonstrate this using both simulated populations and real multiple unit data from area 17 of an anesthetized cat. In the case of the simulated data, pattern probabilities are accurately recovered for large (60 neuron) populations expressing many (thousands) of unique patterns, even when the collective structure is weak compared to neurons’ independent stimulus drive. In the case of the cat data, we find that a grating stimulus is better encoded by collectively driven patterns than by independent neuron encodings.
2 Methods

In this article, we consider discrete time, spatial patterns of spikes across a neuronal population. That is, each pattern is an $N$-element vector of ones and zeros corresponding to neurons that are, or are not, firing within a particular time bin (see Figure 1A). For $N$ neurons, $2^N$ patterns are possible; however, in general only a subset $M \ll 2^N$ will be observed. Each observed pattern is assigned an ordinal numeral $m \in 1, \ldots, M$ and this designation is used to define a multinomial process that describes the temporal evolution of the population spiking.

We wish to construct an encoding model of how each pattern’s expression probability depends on an arbitrary, multidimensional stimulus $s_t$. We write this probability using a multiplicative form:

$$P_{m|s_t} = P_m \rho_m(s_t). \quad (2.1)$$

$P_m$ is the mean (null) probability that pattern $m$ is expressed and is independent of the stimulus. This is what a second-order coupled model, such as an Ising model, would estimate, and much effort has gone into accurately estimating such null models. Our goal is to determine how the pattern probabilities given by the null model are modulated due to stimulus drive. Toward this end, we will initially evaluate $P_m$ by simply counting the number of times each pattern occurs ($N_m$), in a manner analogous to estimating a neuron’s mean firing rate by counting spikes:

$$P_m = \frac{N_m}{\sum_{m} N_m}. \quad (2.2)$$

Later we generalize the null distribution so that all patterns are allowed some nonzero probability.

$\rho_m(s_t)$ is a function that describes how the stimulus modulates the expression probability of pattern $m$ about its mean. In general, there will be too many patterns to accurately model this for each individual pattern $m$. Instead, we partition the patterns into a set of $C \ll M$ clusters whose member patterns have probabilities that covary identically with the stimulus (see Figure 1B). That is, for each pattern $m$ in a cluster $c$, $\rho_m(s_t) = \rho_c(s_t)$. Since it is not a priori clear how many clusters should be used or from which patterns they should be constructed, we use a logistic regression tree to perform a recursive binary partitioning of the patterns.

An example of such a tree is shown in Figure 1C. The root node comprises all the patterns. The patterns are then split into two groups defining child nodes. Patterns within each of the child nodes have the same stimulus-dependent probability term $P_{\text{child}}(s_t)$. This is estimated using a logistic regression model that depends on the stimulus and also the
Figure 1: Schematic of the tree-based pattern encoding model. (A) Spikes are binned into discrete time (here 10 ms) patterns across neurons. (B) The encoding model partitions the $M$ unique observed patterns into $C$ clusters. The member patterns of these clusters have similar stimulus encoding properties. (C) Partitioning is accomplished using a regression tree that constitutes a probability model for each pattern. The root node contains all patterns (colors denote patterns with similar encoding properties) and is split into two child nodes. The probability of each child node is described by a stimulus-dependent logistic regression model. The child nodes are themselves split, until the BIC is minimized. The leaves on the tree are the final clusters, comprising patterns with the same probabilistic dependence on the stimuli. (D) Each split in the tree defines a soft partitioning (white line) of the stimulus space into subspaces. The leaf has high probability in the final subspace (denoted by red).

partitioning of patterns between child nodes. Optimum partitioning is achieved using an expectation-maximization-type algorithm (described below) that maximizes the data likelihood. Crucially, this algorithm does not depend on the patterns in each child node “looking similar,” although this
can be reintroduced by adding a prior (see below). The child nodes are then themselves split into additional child nodes and the process repeated until further splitting is no longer justified by the data. In this article, the Bayesian information criterion (BIC) is required to be further minimized by each branching, although other criteria could be envisioned (see Hastie, Tibshirani, & Friedman, 2009).

The “leaves” of the tree (i.e., the nodes at the edges) are the clusters $c \in 1, \ldots, C$ and the tree itself constitutes a stimulus-dependent probability model for each leaf. The logistic regression fitted probabilities of the child nodes of each branching are conditioned on the probability of their parent node. Thus, the probability $P_{c|s_t}$ that a member pattern from leaf $c$ is observed is the product of the conditional probabilities of each node along the path connecting leaf $c$ to the root of the tree,

$$P_{c|s_t} = \prod_{i \in \text{path}} P_{i|s_t, \text{parents}}$$

where the product is over the nodes $i$ along the path. The modulatory term $\rho_c(s_t)$ is then simply obtained by dividing by the mean leaf probability:

$$\rho_c(s_t) = P_{c|s_t} \frac{T}{\sum_{m \in c} N_m}.$$  (2.4)

The regression tree can be thought of as a type of latent class model for which both the correct number of classes and the partitioning of patterns between classes (clusters) must be determined.

### 2.1 Splitting Algorithm.

At each branching we partition the patterns into child nodes using an expectation-maximization-type procedure that increases the log likelihood of the observed patterns. The algorithm is shown schematically in Figure 2. First, the patterns within a node are assigned randomly to two child nodes designated here as $+$ and $-$. Second, we fit a logistic regression model to the clustered patterns,

$$P_{\pm|x_t} = \frac{e^{\pm[\beta_0 + X_t, \beta]}}{1 + e^{\pm[\beta_0 + X_t, \beta]}}.$$  (2.5)

where $P_+$ and $P_-$ are the node probabilities and $X_t$ is the $t$th row of the covariate matrix $X$, which parameterizes the stimulus. $\beta_0$ and $\beta$ are the fitted parameters of the logistic regression model.

Third, we fix the parameters of the logistic regression model and reassign the patterns between the child nodes so as to maximize the likelihood. We show in appendix A that the total log likelihood is maximized when each
pattern $m$ is assigned to the cluster that individually maximizes

$$\log \mathcal{L}_m = \sum_{t \in T_m} \log P_{m|s_t},$$

where the sum is over the time bins $T_m$ in which pattern $m$ appears. (The full log likelihood is the sum of equation 2.6 over all patterns.) This sum can be calculated explicitly, although we also show in appendix A that if the covariate matrix $X$ is standardized so that each column has zero mean, then the log likelihood is equivalently maximized when each pattern $m$ is assigned to child node $+$ if

$$\sum_{t \in T_m} X_t \beta > 0$$

and to child node $-$ if the sum is less than zero.\(^\text{1}\) Thus, the assignment of each pattern depends not on its mean probability of expression (bias) but on how that probability varies with changing stimuli.

The logistic regression model is then refit and the patterns reassigned; this process is then iterated until patterns no longer shift between nodes

---

\(^{1}\)The pattern is assigned randomly if the sum equals zero exactly.
or an upper iteration bound is reached. If the final result minimizes the BIC, the split into child nodes is retained, and they become new leaves on the tree. Otherwise the child nodes are removed from the tree. The whole procedure is repeated at all leaves on the tree until the BIC is no longer minimized and the splitting terminates. In essence, the tree recursively partitions the stimulus space into smaller subspaces, which are predictive of the patterns (see Figure 1D). (See appendix B for algorithmic benchmarking using various simulated data sets.)

Generating Logistic Regression Trees

1. Assign each unique observed pattern to an ordinal numeral $1, \ldots, M$ and generate a multinomial time series. Designate the group of all patterns as the root node of the tree.
2. Using the occupation probability of each pattern, calculate the log likelihood (see equation 2.17) of the multinomial time series and the BIC.
3. Split the set of patterns $1, \ldots, M$ into two groups (child nodes) by random assignment. Generate a binomial time series where 0 corresponds to the first group and 1 to the second.
4. Fit a logistic regression model to this binomial time series using the stimulus covariate matrix $X$.
5. Using equation 2.6, reassign each pattern to the group (node) that maximizes the log likelihood.
6. Iterate steps 4 and 5 until either no patterns are reassigned or an upper iteration bound is reached.
7. Recalculate the log likelihood and BIC according to equations 2.17 and 2.3. If the BIC post splitting is less than that presplitting, the child nodes become new leaves on the tree. Otherwise, remove them.
8. Iterate steps 2 to 7 at all leaves until no additional leaves are generated.

Matlab code for this algorithm may be found online at http://www.neurostat.mit.edu/software and also http://robhaslinger.org.

2.2 Hamming Regularization. Due to insufficient observations, patterns that occur infrequently may be difficult to assign to a cluster based solely on the data likelihood. To take advantage of any prior information one might have about how patterns should be grouped, a regularization term can be added to the log likelihood. That is, instead of maximizing the log likelihood, equation 2.6, it can be required that

$$\log L_m + \eta R = \sum_{t \in T_m} \log P_{m|s_i} + \eta R$$

(2.8)
be maximized for each pattern $m$ within step 5 of the expectation-maximization algorithm. $\eta$ is a tunable regularization parameter, and $R$ depends on the prior chosen.

In this article, we use a regularization that depends on the Hamming distance between patterns. The Hamming distance is the total number of neurons whose spikes differ between two patterns (L0 norm). This regularization biases the regression so that patterns that look similar tend to be grouped together,

$$R = -\frac{T}{NM_c} \sum_{n \in c} D_H(n, m),$$

where $D_H(n, m)$ denotes the Hamming distance between patterns $n$ and $m$, and the sum is over patterns currently assigned to the child node in question. The denominator $NM_c$ is for convenience and normalizes out the experiment-specific factors of total neuron number $N$ and number of patterns $M_c$ in each child node. Use of $T$ in the numerator puts the regularization on the same scale as the log likelihood $L_m$ and weighs the regularization more heavily (compared to $L_m$, which scales as $T_m$) for patterns that occur infrequently. Thus, patterns that occur infrequently tend to be grouped with others that are similar, while frequently occurring patterns are assigned based on likelihood maximization.

2.3 Generalization. As discussed above, the data set used to fit the regression tree most likely contains only $M \ll 2^N$ unique patterns. For fitting purposes, the probability of the other $2^N - M$ patterns is set to zero. However most likely, these patterns do not have zero probability, and it was simply due to chance (and finite data lengths) that they were not observed. Recall that the tree describes the training data pattern probabilities using a multiplicative form,

$$P_m|s_t = P_m \rho_m(s_t)$$

Generalizing the tree so that it describes the stimulus driven probabilities of all $2^N$ patterns requires three steps; (1) assign a null probability to every possible pattern, (2) assign each unique pattern to a leaf on the tree (so that its covariation with the stimulus is determined, and (3) normalize the full probability distribution so that $\sum_m P_{m|s_t} = 1$ for all stimuli $s$.

2.3.1 Generalize the Null Probabilities. In this first step, we replace the null distribution used for fitting the tree ($P_m = N_m/T$) with one that allows all the patterns nonzero probability, but, we hope, is not too different from that used for fitting. One possibility is to simply use an Ising model (or other parametric model) for the generalized null distribution. A second is to use
the Good-Turing estimator of the missing probability mass (Good, 1953; Orlitsky, Santhanam, & Zhang, 2003). Originally developed by Alan Turing during his efforts to crack the Enigma code (Good, 2000), this estimates the total summed probability of all patterns not observed in the training data by counting the unique patterns observed only once in the training data and dividing this value by the total number of observations in the training data:

$$\mu^{GT} = \# \text{ patterns occurring once} / T.$$ (2.11)

This estimator is common in many fields, for example, in biological censuses, where it is used to estimate the probability of all animal species not observed (Good, 1953; Dornelas et al., 2013). It is unbiased (Good, 1953), and it can be shown that tight confidence bounds on the estimated missing mass exist (McAllester & Schapire, 2000; Berend & Kontorovich, 2012).

Via the Good Turing estimator, the null probability of patterns observed in the training data can be updated as

$$P_m \rightarrow P^{GT}_m = (1 - \mu^{GT})P_m.$$ (2.12)

Novel patterns are assigned a null probability that is a fraction of the missing mass, with that fraction based on an Ising (or other parametric) model assumption:

$$P^{GT}_m = \mu^{GT} \frac{P^{ising}_m}{\sum_{m \in \text{train}} P^{ising}_m}.$$ (2.13)

This null probability distribution is normalized over all possible patterns: $$\sum_m P^{GT}_m = 1.$$

2.3.2 Assign Each Unique Pattern to a Tree. To generalize $$\rho_m(s_t)$$ to all patterns, each novel pattern is assigned to a leaf on the tree. Specifically, motivated by the Hamming regularization used above, it is assigned to the leaf composed of patterns with which it is closest via minimum mean Hamming distance,

$$c = \arg\min_c \frac{1}{M_c} \sum_{m \in c} D_H(m, m'),$$ (2.14)

and we set $$\rho_m(s_t) = \rho_c(s_t)$$ as above. Intuitively, in the absence of any information about how the novel pattern covaries with the stimulus, we revert to our “prior” that patterns that look similar are grouped together.
2.3.3 Normalize the Full Probability Distribution. Finally, the full stimulus-dependent probabilities are normalized by simple division:

\[
P_{m|i} \rightarrow P_{m|s_i}^{\text{norm}} = \frac{P_{m|i}}{\sum_m P_{m|i}}.
\]

(2.15)

This last step ensures normalization for all possible stimuli \(s\); however, it is important to note that since \(\rho_m(s)\) is a modulatory term, \(\int P_{m|i} P(s) ds = 1\), where \(P(s)\) is the distribution of stimuli in the training set, for all patterns \(m\) even prior to this step regardless of which leaf the pattern is assigned to.

Further, if the missing mass is low, the normalization of step 3 will be minimal. In this case, one can skip not only this final step but also the majority of step 2, assigning only to leaves novel patterns that are actually observed in the test set, and greatly diminish the computation time. For example, the V1 cat data presented in section 3 has a missing mass of 0.006, and the final normalization step does not change the pattern probabilities or test data log likelihood in any appreciable fashion.

2.4 Bagging Trees. The regression tree algorithm uses random initial assignments of patterns to subnodes. Thus, each implementation will generate a different tree. This variability is an advantage because the stimulus-dependent probabilities \(\rho_m(s)\) can be averaged across trees, or bagged (Breiman, 1996). Specifically, if \(N\) different trees are generated from the data and the stimulus modulation of pattern \(m\) provided by tree \(n\) is \(\rho_{m,n}(s)\), then the bagged estimate is

\[
\rho_{m}^{\text{bag}}(s) = \frac{1}{N} \sum_{n=1}^{N} \rho_{m,n}(s).
\]

(2.16)

The full pattern probability is \(P_{m|i}^{\text{bag}} = P_{m}^{\text{bag}} \rho_{m}(s)\). Bagging improves generalizability to a validation data set by emphasizing structure conserved throughout the data set and reducing the risk of overfitting.

2.5 Model Evaluation. To evaluate the trees’ goodness of fit, we use the log likelihood of a test data set not used for model fitting. An increase in log likelihood is analogous to a reduction in mean squared error (for gaussian variables, it is identical). It is important to distinguish between patterns being present in the data and patterns actually covarying with the stimulus. Our formalism uses a multiplicative form for each pattern probability \(P_{m|i} = P_{m} \rho_{m}(s)\) (see equation 2.1). This allows the test data log likelihood to be split into two terms with these different physical interpretations.
Denoting \( m_t \) as the pattern \( m \) that is observed at time \( t \), the log likelihood may be written as

\[
\log \mathcal{L} = \sum_{t=1}^{T} \log [P_{m_t | s_t}] = \sum_{t=1}^{T} \log [P_{m_t} \rho_{m_t}(s_t)]
\]

\[
= \sum_{t=1}^{T} \log [P_{m_t}] + \sum_{t=1}^{T} \log [\rho_{m_t}(s_t)]
\]

\[
= \log \mathcal{L}_{\text{null}} + \log \mathcal{L}_{\text{stim}}. \tag{2.17}
\]

The first term depends on only the mean expression probability of the pattern. The second depends on the modulation. In this article, we compare both of these to what would be expected under an independent neuron assumption. If the first is larger than for independent neurons, then structured patterns are present in the data. If the second is larger, then these patterns are also modulated in a fashion that cannot be described by independent stimulus drive. The reason we do not compare to second-order models is that neither GLMs nor Ising models capture both stimuli and dependencies between neurons in the same time bin (see sections 1 and 5).

In this article, we compare both \( \mathcal{L}_{\text{null}} \) and \( \mathcal{L}_{\text{stim}} \) to what would be expected under an independent neuron assumption (see below). For the tree to be deemed “better,” we require that both \( \mathcal{L}_{\text{null}} \) and \( \mathcal{L}_{\text{stim}} \) of the test data be larger for the trees and that the increase, over independent neuron models, be statistically significant. Statistical significance can be determined by noting that \( \mathcal{L}_{\text{null}} \) and \( \mathcal{L}_{\text{stim}} \) are both sums over contributions from different time bins and that the contributions of each individual bin (e.g., \( \log [P_{m_t}] \) and \( \log [\rho_{m_t}(s_t)] \)) are random variables. Since the difference between two random variables is normally distributed by the central limit theorem, a \( t \)-test can be used to determine if the means (over bins) of \( \mathcal{L}_{\text{null}} - \mathcal{L}_{\text{null}}^{\text{indep}} \) and/or \( \mathcal{L}_{\text{stim}} - \mathcal{L}_{\text{stim}}^{\text{indep}} \) are statistically different from zero. Further discussions of this approach can be found in appendix C of Clauset, Shalizi, and Newman (2009) and also Vuong (1989). Again we emphasize that statistical significance of increased test data log likelihood, not training data, is being tested.

\( \log \mathcal{L}_{\text{null}}^{\text{indep}} \) and \( \log \mathcal{L}_{\text{stim}}^{\text{indep}} \) can be calculated for a set of \( N \) independent neuron models as follows. If each neuron is modeled with its own discrete-time, stimulus-dependent probability of spiking (conditional intensity function) \( \lambda_{n|s_t} \), the probability of pattern \( m \) is given by a product

\[
P_{m|s_t}^{\text{indep}} = \prod_{n=1}^{N} \lambda_{n|s_t}^{\theta_{n,u}(t)} (1 - \lambda_{n|s_t})^{1-\theta_{n,u}(t)}, \tag{2.18}
\]
where $\theta_{m,n}(t) = 1$ if pattern $m_t$ has a spike for neuron $n$ and is 0 otherwise. In analogy with equation 2.1, $p_{m}^{\text{indep}}(s_i) = P_{m|s_i}^{\text{indep}}/P_m$ where

$$p_{m}^{\text{indep}} = \frac{1}{T} \sum_{t=1}^{T} p_{m|s_i}^{\text{indep}} \tag{2.19}$$

is the mean probability of the pattern (similar to the mean firing rate of a neuron). These probabilities can be used to calculate $\log L_{\text{null}}^{\text{indep}}$ and $\log L_{\text{stim}}^{\text{indep}}$ for comparison with the regression tree pattern model.

It should be noted that $p_{m}^{\text{indep}}$ is not the same as the pattern probability that would be calculated for a mean firing rate model. This is given by

$$p_{m}^{\text{rate}} = \prod_{n=1}^{N} \lambda_{n}^{\theta_{m,n}(t)} (1 - \lambda_{n})^{1-\theta_{m,n}(t)} \tag{2.20}$$

where $\lambda_{n} = (1/T) \sum_{t} \lambda_{n|s_i}$. Essentially the difference is that the product of a sum is not the same as the sum of a product.

3 Results

We first present results from simulated data to illustrate four points. First, the algorithm can recover the correct pattern groupings. Second, bagging can substantially improve goodness of fit. Third, Hamming regularization can improve generalizability to an independent test set. Fourth, our method can distinguish between neurons driven independently by stimuli and neurons driven collectively. We then apply the regression tree algorithm to population activity collected in V1 of an anesthetized cat stimulated by a moving grating and show that the encoding of the grating stimulus is better described by patterns than individual neurons.

3.1 Algorithm Recovers Correct Pattern Groupings. To demonstrate that the algorithm accurately groups patterns according to their encoding properties, we generated ordinal patterns from a logistic regression tree (shown in Figure 3A). In the absence of Hamming regularization, nothing in the formalism or regression tree algorithm requires that the patterns be generated by spiking neurons. Instead they can simply be designated by an ordinal list $m \in 1, \ldots, M$ of distinct observation types, which is what we do in this section.

The tree used to generate the data has 20 leaves, each containing 20 patterns such that the total number was $M = 400$. Given a particular leaf, each pattern within it was equally probable. The leaf probabilities were generated using a covariate matrix $X$ with 1 constant bias column, and 24
Figure 3: Algorithm recovers correct pattern groupings and probabilities. (A) Logistic regression tree used to generate ordinal patterns with time-varying probabilities. The 20 leaves are in gray, each comprising 20 patterns. (B) Fitted logistic regression tree. Although the branchings of the tree are different than the model, each leaf is isotropic to (comprising the same patterns as) a leaf of the model tree. (C) Time-varying probabilities of four leaves. Gray is the model. Black is the fitted regression tree. The plots are nearly identical.

Time-varying columns. The time-varying columns were chosen to mimic processes occurring at different timescales and had the form

\[ x_t = \cos(2\pi ft + \phi_0), \]

where \( f \) was chosen randomly between 0.1 and 10 Hz and \( \phi_0 \) was chosen randomly between 0 and \( 2\pi \). The parameters \( \beta \) defining the logistic regression model of each branching were chosen randomly within \([-0.5, 0.5]\). We simulated 200 seconds of data at 1 msec resolution from this model.

Since the initial assignment of patterns at each branching is random, the regression tree algorithm generates a different tree each time it is applied. We show one tree fit to the data in Figure 3B. Although the fitted tree looks different from the model, it has 20 leaves, and each leaf in the fitted tree is
isotropic (comprising the same patterns) to a leaf on the model tree. It is well known that different regression trees can have similar or even identical input output mappings (Ripley, 1996). For example if there are three true leaves, splitting the root node \(\{A,B,C\}\) into children \(\{A,B\}\) and \(\{C\}\) and then \(\{A,B\}\) into \(\{A\}\) and \(\{B\}\) produces the same partition as splitting \(\{A,B,C\}\) first into \(\{A\}\) and \(\{B,C\}\) and then \(\{B,C\}\) into \(\{B\}\) and \(\{C\}\). What matters is not the exact structure of the tree but how well it fits the data. When compared to the tree originally used to simulate the data, the fitted tree accounts for 96% of the stimulus-driven log likelihood. The null log likelihoods are statistically identical \(p < 10^{-3}\) via \(t\)-test. Moreover, the mean correlation coefficient between the pattern probabilities as defined by the model and as deduced by the regression tree is \(r = 0.94\) \((p < 10^{-3})\). Finally in Figure 3C, we show the probability of four of the leaves (over a 2 second example epoch) as defined by the model tree and as predicted by the fitted tree, and we demonstrate extremely accurate agreement.

### 3.2 Bagging Trees

The fact that the algorithm produces different trees for different implementations is an advantage because the pattern probabilities predicted by different trees can be averaged, or bagged, to improve goodness of fit. In this example, we used the same (as above) grouping of 400 ordinal patterns into 20 groups, but used a multinomial logit model to generate the group probabilities (see Figure 4A):

\[
P_c(X_t) = \frac{e^{X_t \beta_c}}{\sum_{c' = 1}^{C} e^{X_t \beta_{c'}}}.
\]

The covariate matrix \(X\) was identical to that of the above section, and the parameters \(\beta\) were again chosen randomly within \([-0.5, 0.5]\).

Figure 4B shows two different 20-leaf regression trees, generated by different applications of the algorithm. Each has an isotropic mapping of leaves to multinomial logit groups. Both trees account for 83% of the stimulus-driven log likelihood of the original multinomial logit model, and the mean correlation coefficients (over all 400 patterns) between the fitted and model probabilities are \(r = 0.81\) and \(r = 0.82\) \((p < 0.001\) for both\) respectively. The null log likelihoods are statistically identical for the fitted trees and the original multinomial logit model \((p < 10^{-3} t\)-test\). The time-varying probabilities (over a representative 1 second epoch) for six patterns (each from a different group) as defined by the model and the first fitted tree are shown in Figure 4C. This level-of-goodness of fit is repeatable, as shown by the log likelihoods and correlation coefficients of 50 different trees in Figure 4D.

The individual trees have poorer fit than in the previous example because they were not generated from a model with treelike structure. However, the fit can be substantially improved by bagging (averaging pattern probabilities) across all 50 trees (black lines in 4C and vertical dashed lines in 4D). The
Figure 4: Bagging can substantially improve goodness of fit. (A) A multinomial logit model with 20 pattern groupings was used to generate data. (B) Two different regression trees, each with 20 leaves isotropic to the original model. (C) Time-varying probabilities of six patterns (from different groups) according to the original model (gray), the first tree shown in panel B (dashed black), and 50 bagged trees (black). (D) Left: Histogram of the log-likelihood ratio (compared to original model) accounted for by all 50 fitted trees. (Right): The mean correlation coefficient between the true and fitted pattern probabilities. Bagging substantially improved these goodness-of-fit measures (vertical dashed lines). (E) Log-likelihood ratio and mean correlation coefficients as a function of the number of trees used for bagging.

Bagged trees account for 90% of the stimulus-driven log likelihood of the original multinomial model, and the mean correlation coefficient between the bagged and model pattern probabilities increases to 0.93. Figure 4E shows how these goodness-of-fit measures increase with the number of
trees used for bagging. Thus, even when data are not generated by a tree, multiple bagged trees can still provide a good probability model.

3.3 Simulated Cell Assemblies: Hamming Regularization. For large neuronal populations, many of the patterns that are observed may occur only rarely, making it difficult to accurately group them. Here, regularization can be used to stabilize the regression trees and improve generalizability to an independent test data set. We simulated neural spiking patterns from 60 cells that normally fired independently at 40 Hz. However, for certain values of a simulated input, stimulus groups of 6 cells had a high probability of firing together, expressing four or more spikes (out of six). Patterns of three spikes were not allowed. Given a group, or cell assembly, each of its member patterns (22 total per group) was equally probable. (See Figure 5A for a raster plot in which patterns corresponding to cell assemblies are in gray.)

Formally, the data were generated using a multinomial logit model (see equation 3.2) with 11 groups. Ten of these were cell assembly groups, and the 11th was the independent neuron group. The same 25-column stimulus covariate matrix $X$ as above was used. The parameters of the multinomial logit were chosen so that the independent neuron group was most probable (independent firing 92% of the time), and the cell assembly groups were expressed more rarely (each roughly 0.8% of the time). Specifically, $\beta_c$ was chosen randomly within $[-1, 1]$ except for the bias parameter, which was chosen to be 4 for the independent neuron group and $-4$ for the other 10 groups. This model has 2051 unique patterns, and from it, we simulated 100 seconds (at 1 msec resolution) of training data and 50 seconds of test data. The total number of patterns expressed by the training data, and the percentages of time for which they were expressed, are both shown in Figure 5B as a function of the number of spikes in the pattern.

In Figure 5C, we show an 11-leaf tree, which recovers the correct pattern groupings, and in Figure 5D, we compare four pattern probabilities from the model with those deduced by bagging over 50 trees. The tree and probabilities shown were generated by regularizing the node-splitting algorithm. As discussed in section 2, a penalization was introduced so that patterns that look similar via Hamming distance tended to be grouped together. The strength of this regularization was controlled using a regularization parameter $\eta$. For small values of $\eta$, the algorithm produced trees with too many leaves and did not group the patterns properly (see Figure 5E, top). This is likely due to both a large number of patterns and the fact that many of them were observed infrequently. However, for a regularization of approximately $\eta = 10^{-2}$, the correct number of leaves was recovered and the patterns were grouped correctly. This was the case across the majority of fitted trees as evidenced by the mean (within group) Hamming distance between patterns approaching that of the original model for all fitted trees (see Figure 5E, bottom).
Figure 5: Regularization improves generalizability. (A) Sample raster plot of 60 simulated neurons firing independently, except when groups of four or more cells fire collectively as a cell assembly (large gray dots). (B) Number of unique patterns as a function of number of spikes contained and percentage of time expressed. (C) Regularized regression tree recovering correct pattern groupings. Panels show patterns contained in each leaf. White dots denote spiking neurons. Top panel: Independent neuron group. Bottom 10 panels: cell assembly groups. (D) Bagged pattern probabilities (black) for four patterns (from different leaves). Gray: original model. (E) Mean number of leaves and mean within leaf Hamming distance versus regularization parameter. Dashed lines are original model. Small dots are unregularized result. (F) Stimulus-driven log likelihoods for training and test data. Black: average over 50 trees. Gray: bagged trees. Black and gray dots are maxima of test data log likelihood. Dashed line is original model.
The optimal value for $\eta$ is that which maximizes the stimulus-driven test data log likelihood (see Figure 5F, bottom). Specifically, we used 50 logarithmically spaced values of $\eta \in [10^{-4}, 10^{-1}]$. Increased regularization degrades the fit of the training data; however, this is due to over-fitting when no regularization is used (see Figure 5F, top). In contrast, the stimulus-driven log likelihood of the test data is maximized near $\eta = 10^{-2}$ at the same regularization that recovers the correct pattern groupings. In addition to showing the mean stimulus-driven log likelihoods across 50 individual trees (black) we also show the stimulus-driven log likelihoods when trees are bagged (gray). Validation data maxima are denoted by gray and black dots. Note that the null (stimulus-independent) log likelihood is not affected by the regularization since it is set before the trees are fit. In summary, regularization of the regression tree algorithm, motivated by the idea that patterns that look similar should be clustered together, produces a more parsimonious and, in this case at least, more accurate representation of the data.

3.4 Distinguishing Patterns from Independent Neurons. Our algorithm can distinguish between neurons being driven independently and neurons being driven collectively. To demonstrate this, we used the same 60 neuron cell assembly model as above, but varied the percentage of time during which cell assemblies were observed. Specifically, we varied the multinomial logit model bias parameters, but kept the form of the multinomial logit model, the patterns forming cell assemblies, the covariate matrix $X$ (which parameterizes $s_t$), and the parameters corresponding to stimulus modulation (nonbias parameters) fixed.

What was crucially different was that when neurons fired independently, they did so with rates that were stimulus modulated (the previous section used a mean rate). To simulate independent firing for each neuron $n$, we modeled its independent stimulus-driven spike probability (per time bin) with a logistic regression model,

$$\lambda_{n|s_t} = \frac{e^{X_t\alpha_n}}{1 + e^{X_t\alpha_n}},$$

and randomly drew spikes according to these stimulus-dependent probabilities. The parameter vectors $\alpha_n$ were chosen so that the 60 neurons had mean firing rates ranging between 3 and 10 Hz (population mean 5.5 Hz). These rates were strongly modulated by the stimulus, with standard deviations ranging between 10 and 30 Hz (population mean standard deviation was 17.7 Hz). In summary, for each time $t$, we used the multinomial logit model to determine if the population was firing collectively (in which case, we drew a pattern as in the previous section) or if it was firing independently (in which case, we drew independent spikes according to equation 3.3. We simulated 100 s of training data and 50 s of test data at 1 ms resolution.
Comparing Tree Based and Independent Models

![Graph showing the comparison between cell assemblies and randomized cell assemblies](image)

**Figure 6**: Algorithm detects the presence of weak patterns. (Top): Ratio of null test data log likelihoods for bagged tree and independent neuron models. (Bottom): Ratio of stimulus-driven log likelihoods. Black squares: cell assembly model of Figure 5. Gray circles, same model but with patterns randomized across cell assemblies (see text). The horizontal lines denote equality for the bagged tree and independent neuron model log likelihoods. The vertical dashed lines in the bottom panel denote the minimum percentages at which the collective stimulus drive can be detected, that is, the test data stimulus log likelihood is significantly higher for the trees than the independent neurons. The null log likelihood (top panel) is significantly larger for the trees at all data points.

Both bagged (unregularized) regression trees and independent neuron models (see equation 3.3) were fit to the simulated population spikes.

We then compared (in Figure 6) both the null test data log likelihoods and the stimulus-driven log likelihoods of the tree-based and independent neuron-based models (see equation 2.17). The null log likelihoods were always higher for the trees, indicating that collective patterns were present. However, the trees could also detect that these patterns were collectively driven by the stimulus when the cell assemblies appeared more than 2.7% of the time. That is, the stimulus-driven log likelihood was greater for the
trees than the independent neuron models and the difference was statistically significant. To ensure that this was not a function of the exact manner in which the patterns were assigned to cell assemblies, we randomized the patterns with four or more spikes across cell assemblies (while retaining 10 assemblies total and 22 patterns per assembly) and repeated the calculations. The percentage at which pattern drive could be detected changed only marginally to 3.9%. These results demonstrate that even when the collective patterns being driven by stimuli appear only rarely (and the independent drive to neurons is strong), the collective structure can still be detected.

4 V1 Cat Population Data

To demonstrate the algorithm on real neurobiological data, we used 20 neurons recorded in area 17 of an anesthetized cat in response to a drifting sinusoidal grating. The neurons had firing rates that ranged between 2 and 21 Hz, with a median rate of 7.4 Hz. A raster plot is shown in Figure 7A. The grating stimulus had a temporal frequency of 0.83 Hz and was presented in repeated trials of 3.5 seconds each; 20 trials for each of 12 different directions (30 degree spacing) were recorded. We partitioned these into 15 training data trials and 5 test data trials for each direction. We concatenated the last 3 seconds of each trial (to eliminate transients from the stimulus onset) and discretized the spikes using 3 ms bins. This resulted in 2600 unique patterns in the training data and 1213 patterns in the test data, 860 of which were also in the training data (see Figure 7B). Further experimental details are in appendix C (also see Yu, Huang, Singer, & Nikolić, 2008).

We first modeled the response of each neuron independently using a standard logistic regression model fit by maximum likelihood:

\[
\lambda_n(\phi, t) = \frac{e^{X(\phi, t)\alpha_n}}{1 + e^{X(\phi, t)\alpha_n}}.
\]  

(4.1)

\(\lambda_n(\phi, t)\) is the discrete time probability that neuron \(n\) has a spike in time bin \(t\). \(X(\phi, t)\) parameterized the grating stimulus as a function of both grating direction \(\phi\) and the time since stimulus onset \(t\). \(\alpha_n\) is a parameter vector fit individually to each neuron’s spikes.

To define \(X(\phi, t)\), we noted that most neurons exhibited periodic responses (reflecting the sinusoidal grating stimulus) but with different phase shifts (presumably reflecting their different retinotopies). Further, the magnitude of these responses varied considerably with grating direction. (See Figure 7A for an example.) In order to account for all these effects, we modeled the influence of the stimulus on each neuron’s spikes as

\[
X(\phi, t)\beta = \mu + A \cos(\phi - \phi_1) + B \cos(2\phi - 2\phi_2) + C(\phi) \cos(\omega t - \Psi(\phi)).
\]

(4.2)
Figure 7: Application of regression trees to a 20 V1 cat neuron population. (A) Example raster plot. Black dots correspond to patterns with one spike (green), two spikes (red), or three or more spikes. (B) Number of patterns and percentage of time expressed. (C) Neurons were stimulated by a sinusoidal grating moving in 12 different directions. Histogram is PSTH for each direction. The red line is an independent neuron model fit. (D) An optimally regularized regression tree. (E) Left: Stimulus-driven log likelihoods (blue) versus regularization for training and test data. The horizontal black line is an independent neuron model. Circles denote optimal regularization. The gray region denotes significant improvement over the independent model. Red lines are log likelihoods of shuffled and jittered data. (Right) Null log likelihood (blue) as a function of missing mass used for generalization. The circles are Good-Turing estimates. Red is shuffled and jittered data. The horizontal black line is an independent neuron model, and the horizontal dashed line is an Ising model.
We show in appendix D that the right-hand side of equation 4.2 can be rewritten as a linearly weighted sum where $X$ is a 13-column covariate matrix and $\beta$ is a 13-element parameter vector. Briefly, $\mu$ is a fitted parameter that quantifies the mean firing rate. The second and third terms model how this rate is modulated by grating direction $\phi$. $A$, $B$, $\phi_1$, and $\phi_2$ are parameters to be fit. The fourth term modulates the response as a sinusoidal function of time since stimulus onset. The phase shift of this last sinusoid depends on the grating direction, as does the magnitude of the sinusoidal response. To check that this stimulus parameterization was reasonable, we used it to fit independent logistic regression models to the spikes of each neuron in the population. We then compared these fits to the neurons’ empirical direction-dependent PSTHs (see Figure 7C for an example). The mean correlation coefficient between the fitted probabilities $\lambda_n(\phi, t)$ and the neurons’ PSTHs was 0.72. It should be noted that by comparing the logistic regression GLM with the PSTH, we are determining only if the trial-averaged stimulus drive to individual neurons is reasonably parameterized by $X(\phi, t)$.

We next used exactly the same covariate matrix $X(\phi, t)$ to model how the patterns of spikes across the population depended on the grating stimulus. We fit 50 different trees at each of 50 different logarithmically spaced values of the Hamming regularization parameter ($\eta \in [10^{-4}, 10^{1}]$). Figure 7D shows an optimally regularized tree, that is, a fit using the regularization corresponding to the largest test data bagged stimulus-driven log likelihood. In Figure 7E (left), we show these bagged stimulus-driven log likelihoods (in blue) of both the training and test data sets, normalized to a scale where the corresponding log likelihoods of the above independent neuron model are 1. The stimulus-dependent part of the test data log likelihood (that modeled by the regression trees) has a normalized maximum of 1.04, indicating that at least some of the patterns covary with the grating stimulus in a manner that cannot be fully explained by an independent neuron model. This difference is highly statistically significant at $p = 1.4 \times 10^{-4}$ by $t$-test on the set (over all time bins) of differences between the stimulus-dependent log likelihoods. A $t$-test is appropriate because the set of differences was normally distributed ($p < 10^{-3}$, Jarque-Bera test). To further ensure that this improvement in fit was robust, we split the test data into two halves and recalculated their stimulus-driven log likelihoods. We found that regardless of which half was used to choose the optimum regularization parameter (validation data), the other (test data) demonstrated significant ($p < 0.05$; $t$-test) improvement in fit over the independent neuron model.

Importantly, the stimulus-driven portion of the log likelihood is invariant to exactly which generalization of the null distribution is used. Neither the Good-Turing generalization nor replacing the null distribution with an Ising model alters the stimulus-driven log likelihoods (e.g., the curves of Figure 7E, left, are indistinguishable when the Good-Turing versus Ising null distributions are used, not shown). This is an advantage of the
pattern model’s multiplicative nature: \( P_{m|s} = P_m \rho_m(s_t) \). Generalizing the null distribution only influences \( P_{m|s} \) through a stimulus-dependent normalization factor common to all patterns (step 3 in section 1.3; see equation 2.15). This was minimal for the cat data, a consequence of the training data’s missing probability mass being very small, 0.006 by the Good-Turing estimate and 0.0075 and 0.0026 when estimated by Ising and independent neuron models, respectively. For all stimulus values, the renormalization of equation 2.15 was never more than a factor of 1 ± 0.01 and averaged 1 ± 0.0015 over the set of all stimuli. It should also be noted that as long as \( P_m \) is itself normalized, the renormalization of equation 2.15 cancels out when averaged over all stimuli (see the discussion of section 1.3). The crucial point is that the Good-Turing, the Ising, or even another null distribution (perhaps with higher-order correlations) can be used for generalization. Our approach is designed to determine how null pattern probabilities are modulated by stimuli, not necessarily determine those null probabilities with precision. For this particular data set, the Ising model is a slightly better generalization of the null distribution than the Good-Turing estimate (see Figure 7E, right), but the difference is not statistically significant (\( p = 0.82 \); \( t \)-test). However, both null distribution generalizations are significant improvements (\( p = 0.03 \) Good-Turing and \( p = 0.02 \) Ising; \( t \)-test) over the independent neuron model.

To ensure that we were actually modeling how the pattern structure depended on the grating stimulus, we randomly shuffled the individual test data trials and also temporally jittered the test data spikes (by ±15 ms, e.g., ±5 time bins). This destroyed the coordinated (across neurons) spike timing (pattern) structure but retained the relation between each individual neuron’s spikes with the stimulus. This was confirmed by fitting independent neuron models to each neuron’s original and shuffled spikes, visually comparing the firing probabilities fitted to the original and shuffled data, calculating the correlation coefficients between these probabilities (correlation coefficient \( r = 1; p < 0.001 \) for all neurons), and calculating the log likelihoods, which were unchanged for all neurons. However, when considering the bagged trees the log likelihoods of the shuffled spikes (Figure 7E, red) did change. The stimulus-driven portions of the normalized log likelihood dropped below 1. That is, once the pattern structure is destroyed, the fit of the bagged regression trees dropped to that of the independent neuron model. The null log likelihood dropped as well but remained greater than 1 since the zero pattern (no spikes in any neuron) is not destroyed by shuffling and jittering and was better described by the pattern model.

Regularization was crucial to getting fits superior to that of independent neuron models. In Figure 8A, we show both unregularized and optimally regularized trees. Figures 8B and 8C show the patterns of the 12 most probable leaves belonging to each of these trees. The effect of the regularization is to bias the algorithm so that patterns that look similar via Hamming distance are grouped together. Common singlets, doublets, and triplets are
Figure 8: Regularization improves the fit of cat V1 patterns with high numbers of spikes. (A) Unregularized and regularized regression trees. (B) Member patterns for the 12 most probable leaves of the unregularized tree (C) and for a regularized tree. Singlets, doublets, and triplets are apparent in the leaves of the regularized tree. (D) Left: Percentage of the log likelihood accounted for by patterns with different numbers of spikes. Gray = no regularization, black = regularized. Right: Ratio of the plots in the upper panel. Regularization improves the fit of three spike patterns by more than a factor of two.
clearly visible in many of the leaves. Such structure is less apparent in the unregularized tree. Next, we explored which patterns fits were being improved by regularization. After breaking down the stimulus log likelihood as a function of the number of spikes in each pattern (see Figure 8D, left), we found that the increased fit came mostly from spikes with three or more patterns. Proportionally, the log likelihood more than doubled for patterns with three spikes (see Figure 8D, right). Thus, the effect of regularization was to stabilize the fits of patterns that occurred rarely (had high numbers of spikes).

The bagged regression trees constitute a probability model for each pattern that depends on both the grating direction and time since stimulus onset. We calculated tuning curves for all of the observed patterns and also how their probabilities varied with time since onset (temporal profile). Most tuning curves and temporal profiles changed little when calculated from the regression trees versus the independent neuron models, but a minority did. We show three examples in Figure 9. We investigated whether these changes could be used for improved decoding of grating direction (over independent neuron models) using a maximum a posteriori decoding
scheme. We found slight improvement (41/60 versus 39/60 test data trials decoded the grating direction accurately), but this was not statistically significant ($p = 0.34$; bootstrap test of proportions). This lack of significance is not surprising given that the improvement in fit (stimulus log likelihood) was only 4% for this data set. It is possible that if the population size was larger, the decoding could have been improved further.

5 Discussion

With the advent of multielectrode arrays (Nicolelis, 2008) it has become possible to simultaneously record the action potentials of hundreds, and potentially thousands, of neurons. Such experiments provide an unprecedented opportunity to study how computations are expressed as spatiotemporal patterns of neural activity. Unfortunately, statistical methodologies for analyzing neural population data have not kept pace with the experiments (Brown, Kass, & Mitra, 2004; Macke, Opper, & Bethge, 2011). One reason is combinatorial; the sheer number of patterns a large neural population can express precludes any sort of direct modeling approach. Another reason is that neuronal populations, or at least the small subsets of the network that can be recorded from, are highly stochastic (Schiller, Finlay, & Volman, 1976; Tolhurst, Movshon, & Dean, 1983; Vogels, Spileers, & Orban, 1989; Mainen & Sejnowski, 1995; Chelaru & Dragoi, 2008; London, Roth, Beeren, Hausser, & Latham, 2010). In this article, we presented a probabilistic model of encoding by population activity, and an algorithm for fitting it, which circumvents these two difficulties.

To circumvent the combinatorial problem, we used a logistic regression tree to group patterns into clusters, the members of which convey equivalent information about the stimulus being presented. The regression tree was generated via an expectation-maximization-type, divisive clustering algorithm that iteratively split the expressed patterns into subgroups with similar encoding properties. At each split, a logistic regression model was fit (E step) and the patterns were then reassigned between groups so as to maximize the data likelihood (M step). To circumvent the stochasticity problem, we regularized the regression tree to bias the clustering toward grouping patterns that were similar by Hamming distance. This requires fitting trees for multiple values of the regularization parameter and choosing the optimum value by cross-validation. While regularization is computationally intensive, it can substantially improve generalizability to independent test data. Regularization tends to improve model fit for patterns with several (two or more) spikes. These patterns occur rarely and are harder to assign by likelihood maximization alone.

Regression trees constitute a true pattern-encoding model that identifies how the probability of each pattern covaries with an arbitrary stimulus instead of merely identifying that patterns are present beyond chance. We demonstrated our algorithm on both simulated data and population data
recorded in area 17 of anesthetized cat driven by a grating-type stimulus. The simulation studies showed that our algorithm can accurately deduce pattern probabilities even when the data contain thousands of unique patterns and deduce when neurons are being driven collectively, versus individually, even if the collective drive is relatively weak. The analysis of the cat data showed that certain visual stimuli are encoded collectively as patterns, at least to some degree. Regarding this result, we note that the population we used was small (20 neurons), and it could well be that the influence of patterns would grow if a larger population was used.

Population encoding is most commonly studied under an independent neuron assumption (Nirenberg, Carcieri, Jacobs, & Latham, 2001; Petersen, Panzeri, & Diamond, 2001). The population vector model is perhaps one of the simplest examples, merely averaging firing rates (Georgopoulos, Schartz, & Kettner, 1986), but more sophisticated approaches are also commonly used (Truccolo et al., 2005). Most methods that move beyond independence have attempted to determine the existence of correlations between neurons, the joint peristimulus time histogram being one of the earlier efforts (Gerstein & Perkel, 1969). More recently, researchers have used second-order models to couple neurons together with pairwise interactions. However, none of the current methods includes both stimuli and interactions between neurons in the same time bin. That is, they do not describe the encoding properties of patterns. The Ising model couples neurons bidirectionally, so that each neuron in a pair influences each other in the same manner. It has been used in demonstrate the existence of second-order correlations between neurons (Schneidman et al., 2006; Tang et al., 2008). Although Ising models do include a constant bias (mean firing rate) term for each neuron, they do not allow for a time-varying, perhaps stimulus-driven, firing rate because this makes the Ising partition function stimulus dependent and such models become computationally intractable. In contrast, cross-coupled generalized linear models do allow for independent stimulus drive to each neuron, but here the interactions between neurons are time lagged by at least one bin. That is, each neuron’s spike probability is conditioned on the previous spiking history of other neurons in the population (Okatan et al., 2005; Pillow et al., 2008; Truccolo et al., 2010; Gerhard, Pipa, Lima, Neuenschwander, & Gerstner, 2011). Thus, GLMs do not model spatial (in the same time bin) patterns.

Thus, to understand how populations encode, it is important to develop methods that can describe how patterns are driven by stimuli. Such methods are also important for determining whether correlations between neurons between neurons are driven by each other or by a common input (Kulkarni & Paninski, 2007; Lawhern, Wu, Hatsopoulos, & Paninski, 2010; Macke et al., 2011). Certain authors have attempted to address whether patterns are important by looking at the information between patterns and stimuli (Oram, Hatsopoulos, Richmond, & Donoghue, 2001; Nirenberg, Carcieri, Jacobs, & Latham, 2001), others by comparing the performance of decoding
models when correlations are, and are not, included (Barbieri et al., 2004; Pillow, Ahmadian, & Paninski, 2011). We took a different approach, directly generating an encoding model of the pattern probability. One of the authors (E.B.) previously addressed encoding by spike patterns of small populations of thalamic barreloid neurons by directly modeling each pattern’s probability of expression (Ba, Temereanca, & Brown, 2012). However, this brute force approach becomes intractable as the population size grows.

It is an experimental reality that one simply does not, and likely never will, have enough data to model each pattern independently. Instead, guided by the data, we clustered the patterns so that only the pattern statistics present in the data were modeled. This clustering does not depend on patterns “looking the same,” although we found that reintroducing this prior helped in practice. Instead patterns are clustered together if they are collectively more probable in some subset of the stimulus space. Our divisive clustering approach is similar in spirit to the causal state splitting reconstruction (CSSR) algorithm that generates hidden Markov models of time series by divisively clustering past histories into equivalence classes that are maximally predictive of the future evolution of the time series (Shalizi & Klinkner, 2004). In this case the patterns are temporal, and the clustering is based entirely on the internal dynamics of the time series. CSSR performs the clustering completely nonparametrically and hence requires large amounts of data. Still, one of the authors (R.H.) adapted and applied this technique for calculating the computational complexity of spike trains (Haslinger, Klinkner, & Shalizi, 2010). Our regression tree approach reduces the amount of data required by employing parametric logistic regression models and also allows for the effects of external stimuli to be included.

Another algorithm with similarities to our own is the k-means algorithm for clustering Euclidean data. In k-means, the data points, defined in some Euclidean space, are randomly assigned to k-clusters, the cluster’s centroids are calculated, and the data points are reassigned to the cluster with the closest centroid. Calculating the centroids is analogous to our fitting a logistic regression model at each tree branching, while reassigning the data points between clusters is similar to our reassigning patterns between child nodes. In both k-means and our algorithm, what is missing from the EM procedure is the knowledge of which cluster (leaf) a data point (pattern) is a member of.

Regression trees have long been used to model complex functions since their introduction by Breiman and colleagues in 1984 with the CART (classification and regression tree) algorithm (Breiman et al., 1984), and the later C4.5 algorithm of Quinlan (1993). Trees are unbiased estimators but notoriously variable due to the recursive partitioning; if two classes of observations are separated by a split near the root of the tree, they can never be remerged (Ripley, 1996). A number of strategies have been developed to not only handle but also take advantage of this—most notably, bagging and boosting. Bagging averages the response (here, the pattern probability) from many trees to achieve a more reliable estimator (Breiman, 1996). We
used this technique to improve significantly goodness of fit and generaliz-
ability. In contrast, boosting combines many weak classifiers (such as small
trees or tree stubs) to obtain an optimal one by iteratively refitting model
residuals (Schapire, 1990; Freund, 1995). Boosting tends to produce models
that generalize extremely well to test data, so we view it as a promising
avenue for future work.

Other algorithmic improvements suggest themselves. First, we currently
fit logistic regression models at each branching using all of the stimulus
covariates. At times, more optimal models can be achieved by fitting only
a subset of the covariates at each branching (Hastie et al., 2009). Second,
we calculated the BIC at each branching and kept the branching only if
it minimized the BIC. However, an initial branching that leads to a poor
improvement in fit may lead to a subsequent branching with substantial fit
improvement. For this reason, many studies, including the original work of
Breiman et al. (1984) grow their trees “large,” continuing the splitting until
each pattern is in its own leaf. They then “prune” the branches of the tree,
from bottom to top, until an optimally sized tree is found. We have yet to
fully explore either of these strategies in the context of neuronal patterns,
although we note that both will likely increase computation time. For both
the simulated and cat data, a tree could be fit using a standard work station
in only a few minutes.

Perhaps the most significant drawback of the current formalism is that
the pattern model is not nested (McCullagh & Nelder, 1989). That is, it
would be ideal to use the independent neuron case as the null model and
build the regression tree–based pattern model from that starting point, per-
haps by boosting off it (see above). This would have two advantages. First, it
would facilitate the comparison of pattern-based encodings with indepen-
dent neuron-based encodings. Second, since correlations between neurons
tend to be weak and sparse, the encoding properties of most observed pat-
terns are likely best described using an independent neuron assumption.
A nested model would allow one to identify exactly which patterns are, in
contrast, best described collectively. We are currently investigating how a
nested model could be constructed, but such a model is beyond the scope
of this work.

It is commonly held that neuronal networks store and process informa-
tion in spatiotemporal patterns of spiking activity, but exactly how they do
this is hotly debated (Averbeck, Latham, & Pouget, 2006; Jacobs et al., 2009;
Abbott & Dayan, 1999). If the functional role of neuronal correlations and
patterns of population spiking activity are to be fully understood, it is crucial
to develop methodologies capable of characterizing how the collective spik-
ing statistics of neuronal populations depend on stimuli and other variables
of interest. Our approach has the advantage of being extremely general,
allowing any variable to be included in the covariate matrix. For example,
although we restricted ourselves in this article to analyzing how patterns
depend on external stimuli, information about the internal dynamics of
the system, such as spike history dependence, or macroscopic state, such as local field potentials, could also be included. Regarding this generality, it is interesting to note that recent theoretical work on reservoir computing models has shown that recurrent networks can store vast amounts of information in overlapping patterns of activity, and that different linear readouts can reliably access different encodings (Buonomano & Maass, 2009). In principle, regression trees could allow one to experimentally investigate if different encodings are simultaneously present in a true population code of arbitrary correlative order. Beyond neuroscience, we note that our algorithm may also be useful for describing how patterns of activity in other complex systems, such as weather dynamics, ecosystems, spin glasses, and social phenomena, are generated by external driving forces.

Appendix A: Pattern Reassignment

Assume the patterns have each been initially assigned to one of two child nodes, denoted here by $-$ and $+$, and a logistic regression model has been fit such that the probability of each node is given by

$$P_{\pm|s_t} = \frac{e^{\pm[\beta_0 + X_t \beta]}}{1 + e^{\pm[\beta_0 + X_t \beta]}}$$  \hspace{1cm} (A.1)

where the $X_t$ is the $t$th column of the covariate matrix $X$ that parameterizes the stimulus $s$. The log likelihood of the observed patterns is then

$$\log \mathcal{L} = \sum_{t=1}^{T} \log P_{m|s_t} = \sum_{m=1}^{M} \sum_{t \in T_m} \log P_{m|s_t}. $$  \hspace{1cm} (A.2)

$T$ is the total number of observations, and $T_m$ is the subset of observations for which pattern $m$ is observed. $P_{m|s_t}$ is the probability of pattern $m$ at time $t$, and $P_{m|s_t}$ is the probability at time $t$ of the pattern $m$ that was actually observed at time $t$. The goal is to maximize the likelihood by reassigning the patterns $m$ between the two classes, while keeping the parameters $\beta_0$ and $\beta$ fixed (e.g., to perform the M step). To this end, it suffices to maximize each term in the sum over $m$ independently.

Thus, consider this component of the log likelihood:

$$\log \mathcal{L}_m^\pm = \sum_{t \in T_m} \log P_{m|s_t} \hspace{1cm} \left[ \frac{e^{\pm[\beta_0 + X_t \beta]} T_{m|s_t}}{1 + e^{\pm[\beta_0 + X_t \beta]} T_{\mp|s_t}} \right]. $$  \hspace{1cm} (A.3)
If we assume, without loss of generality, that the covariate matrix $X$ has been standardized so that each column has mean zero, then the fraction of observations belonging to each class $\pm$ is given by the bias parameter $\beta_0$:

$$
\frac{T_\pm}{T} = \frac{e^{\pm \beta_0}}{1 + e^{\pm \beta_0}}. \tag{A.4}
$$

Substituting this into equation A.3 and canceling terms, we obtain

$$
\log L_\pm^m = \sum_{t \in T_m} \log \left[ \frac{e^{\pm [\beta_0 + X_t \beta]} T_\pm T_m}{1 + e^{\pm [\beta_0 + X_t \beta]}} \right]
\quad = \sum_{t \in T_m} \pm X_t \beta + \log \left[ \frac{1 + e^{\pm \beta_0}}{1 + e^{\pm [\beta_0 + X_t \beta]}} \right] + \log \left[ \frac{T_m}{T} \right]. \tag{A.5}
$$

The node into which pattern $m$ should be placed is the one that maximizes equation A.5. The third term is the same for both nodes. The first depends on the sign of $\sum_{t \in T_m} X_t \beta$. We now show that the second term does as well.

Multiplying the top and bottom of the second term by $e^{\mp [\beta_0 + X_t \beta]}/2$, we obtain

$$
\log \left[ \frac{1 + e^{\pm \beta_0}}{1 + e^{\pm [\beta_0 + X_t \beta]}} \right] = \log \left[ \frac{e^{\mp \beta_0/2} + e^{\pm \beta_0/2}}{e^{\mp [\beta_0 + X_t \beta]/2} + e^{\pm [\beta_0 + X_t \beta]/2}} \right] \mp X_t \beta/2. \tag{A.6}
$$

Inserting equation A.3 into equation A.5 and combining terms, we obtain

$$
\log L_\pm^m = \sum_{t \in T_m} \pm \frac{X_t \beta}{2}
\quad + \sum_{t \in T_m} \log \left[ \frac{e^{\mp \beta_0/2} + e^{\pm \beta_0/2}}{e^{\mp [\beta_0 + X_t \beta]/2} + e^{\pm [\beta_0 + X_t \beta]/2}} \right] + \log \left[ \frac{T_m}{T_\pm} \right]. \tag{A.7}
$$

Only the first term is different between the two nodes.
Thus, so as to maximize the likelihood, pattern $m$ should be placed in the first ($-$) child node if $\sum_{t \in T_m} X_i \beta < 0$ and in the second ($+$) child node if $\sum_{t \in T_m} X_i \beta > 0$. This rule can be recursively applied for splitting each node of the tree.

Appendix B: Algorithmic Benchmarking

To test the regression tree algorithm over a range of data lengths, true numbers of clusters, and numbers of patterns comprising each cluster, we generated multinomial processes from models of the form

$$P_{m|s_t} = P_{m|c} P_{c|s_t}, \quad (B.1)$$

where $P_{c|s_t}$ is a multinomial logit model,

$$P_{c|s_t} = \frac{e^{\beta_c^0 + X_t \beta_c}}{\sum_{c=1}^{C} e^{\beta_c^0 + X_t \beta_c}}, \quad (B.2)$$

and

$$P_{m|c} = \frac{N_m}{\sum_{m \in c} N_m}. \quad (B.3)$$

The parameters $\{\beta_c^0, \beta_c\}$ for each cluster were randomly chosen within $[-1, 1]$. Likewise the covariate matrix $X$ was random with values drawn from within $[-1, 1]$. We chose $X$ to have 24 columns (24 plus 1 bias parameter gives 25 parameters total for each class) and a variable number $T$ of observations (rows). Each class had a variable number of patterns $M$ labeled by ordinal numbers. The probabilities of drawing any given pattern $m \in 1, \ldots, M$ given the cluster $c$ were equal. That is, $P_{m|c} = 1/M$. In the absence of Hamming regularization, the regression tree algorithm can operate on any multinomial process, not only one where the observations are defined as patterns of spikes.

We then simulated data for different numbers of clusters $C \in [5, 10, 20, 50]$ and different numbers of patterns per class $M \in [5, 10, 20, 50]$, and also different numbers of observations $T \in [10, 20, 100, 200] \times 10^3$. These numbers of observations were chosen to reflect experiments at 1 ms resolution, which are 10, 20, 100, and 200 seconds long, respectively. For each triplet $(C, M, T)$, 20 simulated data sets were generated and the logistic tree algorithm fit to the data.

We calculated three measures of goodness of fit. The first was the number of leaf nodes found by the algorithm normalized by the true number of
The second measure was one of pattern misassignment. The individual patterns comprising each leaf node can, due to insufficient data, belong to different true clusters. We assigned each leaf node to a true equivalence class $C$ by choosing the class to which the majority of patterns in the leaf node belong to and calculated the mixing fraction as

$$G = \frac{N_{m \notin C}}{N_{meleaf}}.$$  \hspace{1cm} (B.5)

Third, we calculated the percentage of log likelihood that the regression tree accounted for compared to the true model. We used the null model, for which the probability of any given pattern $m$ is simply its occupation probability, to define a baseline log likelihood,

$$\log L_{\text{null}} = \sum_m N_m \log \frac{N_m}{T},$$  \hspace{1cm} (B.6)

and calculated

$$\Delta L = \frac{\log L^{\text{tree}} - \log L^{\text{null}}}{\log L^{\text{true}} - \log L^{\text{null}}},$$  \hspace{1cm} (B.7)

where $\log L^{\text{tree}}$ is computed with the time-varying probability of each pattern $m$ as implicitly defined by the logistic regression tree, and $\log L^{\text{true}}$ is the log likelihood of the true model used to generate the simulated data. A value of $\Delta L$ close to 1 implies that the tree model is almost as good as the true model, while a value of $\Delta L$ close to 0 implies a poor fit.

The results are shown in Figure 10. A few trends are apparent. First, the number of leaves (first column) tends to be slightly higher than the true number of clusters. This trend increases with the total number of patterns and does not diminish as the simulation time grows. In contrast, the mixing fraction (second column) decreases as the data size grows, indicating that the leaves, although more numerous than they should be, comprise patterns that should be grouped together. These trends arise because if patterns are misassigned by an initial branching, they cannot be remerged. Thus, the algorithm has to make additional leaves to achieve a well-fit model. We often find two or more leaves containing patterns that belong to the same cluster but only that cluster. Although not as parsimonious as the multinomial logit model used for the simulation, the regression tree does describe the data
reasonably well. The log likelihood fraction (third column) is high for all simulations, although it does decrease moderately as the number of clusters grows.

Appendix C: Experimental Methods

Anesthesia was induced with ketamine and maintained with a mixture of 70% $N_2O$ and 30% $O_2$, supplemented with halothane (0.4 0.6%). The animal was paralyzed with pancuronium bromide (pancuronium, organon, 0.15 mg kg$^{-1}$ h$^{-1}$). All the experiments were conducted according to the guidelines of the American Physiological Society and German law for the protection of animals, approved by the local governments’ ethical committee and overseen by a veterinarian.

Multiunit activity was recorded from a region of area 17 corresponding to the central part of the visual field. Single-unit activity was extracted by offline sorting. For recording, we used two SI-based multielectrode probes.
(16 channels per electrode) supplied by the Center for Neural Communication Technology at the University of Michigan (Michigan probes) with an intercontact distance of 200 μm (0.3–0.5 MΩ impedance at 1000 Hz). The probes were inserted in the cortex approximately perpendicular to the surface to record from neurons at different cortical depths and along an axis tangential to the cortical surface. The software used for visual stimulation was Active-STIM (www.ActiveSTIM.com). The stimulus consisted of a drifting sinusoidal grating, spanning 15 degrees of visual angle (spatial frequency, 3 degrees per cycle; temporal frequency of the drift, 3.6 degrees per second), which was sufficient to cover the receptive fields of all the recorded cells simultaneously and to stimulate also the regions surrounding the receptive fields.

Appendix D: Parameterizing the Grating Stimulus  

As discussed in the main text, neurons in area 17 were stimulated by sinusoidal grating of a fixed temporal frequency and in different directions over repeated trials. These neurons mean firing rates were strongly modulated by grating direction and also periodically at the frequency of the grating. This periodic modulation had both amplitudes and phase shifts that were also grating direction dependent. Thus we had to account for all of these effects in parameterizing the stimulus as a covariate matrix $X$ linearly weighted by a parameter vector $\beta$. Defining the argument of the logistic regression models as $\Lambda = X\beta$, we used the form

$$
\Lambda(\phi, t) = \mu + A \cos(\phi - \phi_1) + B \cos(2\phi - 2\phi_2) + C(\phi) \cos(\omega t - \Psi(\phi)).
$$

(D.1)

$\mu$ is a fitted parameter that quantifies the mean firing rate. The second and third terms model how this rate is modulated by grating direction $\phi$, and $A$, $B$, $\phi_1$, and $\phi_2$ are parameters to be fit. These terms can be rewritten using a trigonometric identity so that they are linear in fitted parameters, for example,

$$
A \cos(\phi - \phi_1) = A(\cos(\phi_1) \cos(\phi) + \sin(\phi_1) \sin(\phi))
= \alpha_1 \cos(\phi) + \alpha_2 \cos(\phi),
$$

(D.2)

where $\alpha_1 = A \cos(\phi_1)$ and $\alpha_2 = A \sin(\phi_2)$. The third term can similarly be linearized. This model form is commonly used to model the effect of angular variables such as grating direction.

The fourth term modulates the response as a sinusoidal function of time since stimulus onset. The phase shift of this sinusoid $\Psi(\phi)$ depends on the grating direction, as does the magnitude $C(\phi)$ of the sinusoidal response.
We first split this term analogous to the above:

\[ C(\phi) \cos(\omega t - \Psi(\phi)) = [C(\phi) \cos(\Psi(\phi))] \cos(\omega t) \]
\[ + [C(\phi) \sin(\Psi(\phi))] \sin(\omega t). \]  
\[ \text{(D.3)} \]

We then parameterized each bracketed term similarly to the directional dependence of the firing rate,

\[ C(\phi) \cos(\Psi(\phi)) = \gamma_1 \cos(\phi) + \gamma_2 \sin(\phi) + \gamma_3 \cos(2\phi) + \gamma_4 \sin(2\phi), \]  
\[ \text{(D.4)} \]

and \( C(\phi) \cos(\Psi(\phi)) \) is similar. \( \Lambda \) is now written as a linear sum of weighted functions dependent on the grating direction and time since stimulus onset and can be recast as a 13-column covariate matrix \( X \) multiplied by a parameter vector \( \beta \).

**Acknowledgments**

We thank Demba Ba, Zhe Chen, and Cosma Shalizi for helpful conversations regarding the research presented in this article. This work was supported by NIH grants K25 NS052422-02 (R.H.), DP1 OD003646-0, MH59733-07 (E.N.), and the Hertie Foundation (G.P.). Experiments with cat recordings (D.N.) were supported by a Deutsche Forschungsgemeinschaft Grant NI 708/2-1, Hertie Stiftung, Max-Planck Society, and the Frankfurt Institute for Advanced Studies.

**References**


Received April 17, 2012; accepted January 29, 2013.