A Bayesian Clustering Method for Tracking Neural Signals Over Successive Intervals

Michael T. Wolf*, *Member, IEEE*, and Joel W. Burdick, *Member, IEEE*

***Abstract*—This paper introduces a new, unsupervised method for sorting and tracking the action potentials of individual neurons in multiunit extracellular recordings. Presuming the data are divided into short, sequential recording intervals, the core of our strategy relies upon an extension of a traditional mixture model approach that incorporates clustering results from the preceding interval in a Bayesian manner, while still allowing for signal nonstationarity and changing numbers of recorded neurons. As a natural byproduct of the sorting method, current and prior signal clusters can be matched over time in order to track persisting neurons. We also develop techniques to use prior data to appropriately seed the clustering algorithm and select the model class. We present results in a principal components space; however, the algorithm may be applied in any feature space where the distribution of a neuron's spikes may be modeled as Gaussian. Applications of this signal classification method to recordings from macaque parietal cortex show that it provides significantly more consistent clustering and tracking results than traditional methods based on expectation–maximization optimization of mixture models. This consistent tracking ability is crucial for intended applications of the method.**

***Index Terms*—Bayesian classification, clustering, expectation–maximization (EM), multitarget tracking, neuron tracking, spike sorting.**

I. Introduction

THE NEED to reliably identify and track the activities of a particular neuron in multiunit extracellular recordings is a common problem in basic electrophysiological studies and engineered neural interfaces. Extracellular neural recordings are obtained by positioning the tip of an electrode near enough to a neuron to detect and localize in time the occurrence of the neuron's *action potentials* or *spikes*, which are the basis for neural communication and information processing. However, the electrode tip may happen to be within the "listening sphere" of multiple neurons, thus causing the activity of several neurons to be recorded on a single electrode. In general, the interpretation of extracellular recordings requires a process to associate the spikes in the experimental data with the individual neurons that generated them, a task commonly referred to as "spike sorting" (see [1] for a review). Here, we address the challenge of autonomously classifying spikes according to their generating neurons and tracking the identities of the neuronal sources over time, even as their signal characteristics may change during a recording.

Specifically, this paper considers the following spike sorting and tracking problem. Starting at time t_1, an electrode signal $\mathcal{S}$ is sampled over an interval T_1 of duration Δ. After some preprocessing steps (the spikes in $\mathcal{S}$ are detected and temporally aligned), the spike waveforms found in T_1 are projected onto an d-dimensional feature space (e.g., a 2-D principal component analysis (PCA) basis) so that each waveform is represented as a point. These points must then be "clustered" into sets, with each assumed to be associated with a unique neuron in the multiunit signal. Additional signal samples are taken across successive intervals $T_2 = (t_2, t_2 + \Delta]$, $T_3 = (t_3, t_3 + \Delta]$, and so on.[1] Our goal is to accurately associate the spikes in each T_k to their generating neurons, and then, track the clusters representing individual neurons across successive sampling intervals, as well as to discover the appearance or disappearance of neurons.

This spike sorting and tracking problem arises, for example, in the authors' related work on algorithms for autonomously positioning electrodes to obtain high-quality extracellular recordings [2]–[5]. In the algorithm's main loop, the electrode's signal is periodically sampled for a brief interval (e.g., $\Delta = 10$ s) and analyzed to determine if positional adjustments will improve the quality of a given neuron's signal. To accomplish this, the detected spikes in each interval must be sorted according to their generating neurons, and previously identified neurons must be reidentified in the current recording interval ("tracked"), despite possible changes in the amplitude, phase, and numbers of neuronal signals.

Our method of spike sorting in successive sampling intervals may also be useful in offline processing of lengthy recordings gathered in basic scientific studies. During these recordings, the spike waveforms often evolve over time due to electrode drift and other causes, even without active electrode movement [6]. Dividing these long recordings into short time intervals for analysis can improve spike sorting results, as the data are apt to be effectively stationary over these brief intervals [7], [8].

Unsupervised spike sorting has long been an important and difficult problem in electrophysiology. Many traditional clustering procedures have been adapted for sorting neuronal spike waveforms, including hierarchical [9], k-means [10], [11],

Manuscript received March 31, 2008; revised December 22, 2008 and March 4, 2009. First published July 28, 2009; current version published October 16, 2009. This work was supported in part by the National Institutes of Health (NIH) under Grant R01 EY015545 and in part by the Rose Hills Foundation. *Asterisk indicates corresponding author.*

*M. T. Wolf was with the Division of Engineering and Applied Sciences, California Institute of Technology, Pasadena, CA 91125 USA. He is now with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: wolf@jpl.nasa.gov).

J. W. Burdick is with the Division of Engineering and Applied Sciences, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jwb@robotics.caltech.edu).

Digital Object Identifier 10.1109/TBME.2009.2027604

[1]For simplicity, we assume the lengths of successive sampling intervals are the same, but this is not necessary, nor must the intervals be adjacent. We only assume that Δ is sufficiently long to capture a nontrivial number of spikes.

neural networks [12], superparamagnetic [13], template matching [14], and density grids [15]. The optimization of a (typically Gaussian) mixture model has been shown to be a particularly effective approach [1], [7], [16]–[19].

Most existing techniques, however, are designed for offline batch processing of a single large dataset. Due to our interest in autonomous electrode positioning algorithms, we require an unsupervised method that may be used in real time[2]; thus, noncausal methods such as [7] and [8] are not applicable. Additionally, the short, successive intervals inherent to our problem can complicate the clustering operation because fewer data points (spikes) are available to process, thus exacerbating issues of volatility and variations due to noise. This volatility increases the *inconsistency* of the clustering results across intervals—i.e., spikes sampled in neighboring intervals from the same neurons may, using traditional methods, be sorted into drastically different clusters. (Examples of inconsistency issues are illustrated in Section V; for instance, compare time $k = 7$ versus $k = 8$ of the maximum likelihood (ML) or k-means rows of Fig. 2.) The inconsistency of conventional clustering methods' output over successive intervals is a crucial issue, as it inhibits the association of clusters (i.e., the tracking of neurons) across intervals.

Our method, founded on optimization of a Gaussian mixture model (GMM) via expectation–maximization (EM) [20], [21], is specifically designed to increase the consistency of clusters across successive intervals and track the clusters of persisting neurons. The key idea is to incorporate the available information over time, using the clustering results from interval T_{k-1} to improve the clustering of the data from interval T_k in several ways. Most importantly, we have constructed a special Bayesian prior for the clustering of data in interval T_k. While utilizing priors is common practice in statistical inference, we specifically derive a novel "mixture prior" that aids in tracking individual components of GMMs and still permits the number and identity of neurons to change over successive intervals. We also provide here the closed-form formulas for the corresponding EM calculations, which require introducing a new set of latent variables. Not only does our procedure provide more consistent clustering results, but we also show that it provides neuron tracking (data association across recording intervals) "for free," by virtue of these latent variables.

Additionally, we develop techniques to use the model's statistics from the preceding interval to provide initial values (or *seed clusters*) for EM, and describe a Bayesian model selection method (estimating the number of neurons recorded in T_k) that is more accurate than other commonly used criteria. Thus, clustering is effected as a maximum *a posteriori* (MAP) method (using priors to determine both the model parameters and the model order) rather than the ML method. Employing this MAP method and matching clusters across many successive intervals, neurons can be tracked through large, gradual changes of waveform shape over long periods of time (e.g., hours). (However, like all other methods based on waveform shape, this approach may fail if abrupt, discontinuous changes in a neuron's waveform occur.) We reported a preliminary version of this algorithm in [22].

[2]*Real time* here implies that analysis occurs immediately after each short sampling interval and does not require analysis to be continuously online.

Note that other authors [16], [23] have imposed a general spike waveform prior on all clusters but not one that aided in discriminating waveforms of specific neurons or in tracking these neurons over time. Bar-Hillel *et al.* [7] used results from neighboring intervals for initialization of GMM/EM, but only for the same model order, and still employed a standard ML EM method.

The remainder of this paper is organized as follows. Section II reviews classical clustering based on GMM/EM, so that our extensions can be more clearly delineated. Section III details our method for sequential clustering based on Bayesian parameter estimation and model selection, while Section IV discusses how neurons can be tracked using this clustering process. Applications of this method to neural recordings in macaque cortex are presented in Section V, where we provide characterizations of our method and comparisons to other techniques. Concluding remarks are given in Section VI.

II. ML Optimization of Mixture Models via EM

The classical clustering technique based on ML optimization of a mixture model [20], [21] has been the basis for several spike sorting algorithms. This approach assumes that each neuron produces spikes whose waveform features vary according to a probability distribution, and thus, each generating neuron may be represented as a component in a mixture model. For example, if the ith data point (spike sample in feature space) $y_i \in \mathbb{R}^d$ was generated by the gth neuron (belongs to component, or "cluster," $\mathcal{C}_g$) with associated distribution parameters θ_g, then it is governed by the probability density $p\big(y_i \mid i \in \mathcal{C}_g, \theta_g\big)$. If we assume a Gaussian density, which is denoted by $f_{\mathcal{N}}$, the distribution parameters are the mean and covariance matrix $\theta_g = \{\mu_g, \Sigma_g\}$, and the density is

$$p(y_i \mid i \in \mathcal{C}_g, \theta_g) = f_{\mathcal{N}}(y_i \mid \mu_g, \Sigma_g) \equiv \frac{1}{\sqrt{\det(2\pi\Sigma_g)}} \exp\left(-\frac{1}{2}(y_i - \mu_g)^T \Sigma_g^{-1} (y_i - \mu_g)\right).$$

Including all N data points in the recording interval and all mixture components $g = 1, \dots, G_m$, the *mixture likelihood* $\mathcal{L}_M$ of the model parameters given the data is

$$\mathcal{L}_M(\Theta_m) = p\big(Y \mid \Theta_m, \mathcal{M}_m\big) = \prod_{i=1}^{N} \sum_{g=1}^{G_m} \pi_g f_g\big(y_i \mid \theta_g\big), \quad (1)$$

where the parameters are defined as follows.

1) $Y = \{y_i\}_{i=1}^N$ is the set of all spike observations.
2) $\mathcal{M}_m$ is the mth model class under consideration in the current recording interval, which dictates the model order G_m (i.e., the number of neurons contributing to the signal), the form of the gth probability density f_g (e.g., Gaussian), and the form of the model parameters Θ_m, which include θ_g and π_g.
3) π_g is the mixture weight of component $\mathcal{C}_g$, i.e., the prior probability that an observed spike was generated by the gth source neuron, with $\pi_g \geq 0$ and $\sum_{g=1}^{G_m} \pi_g = 1$.

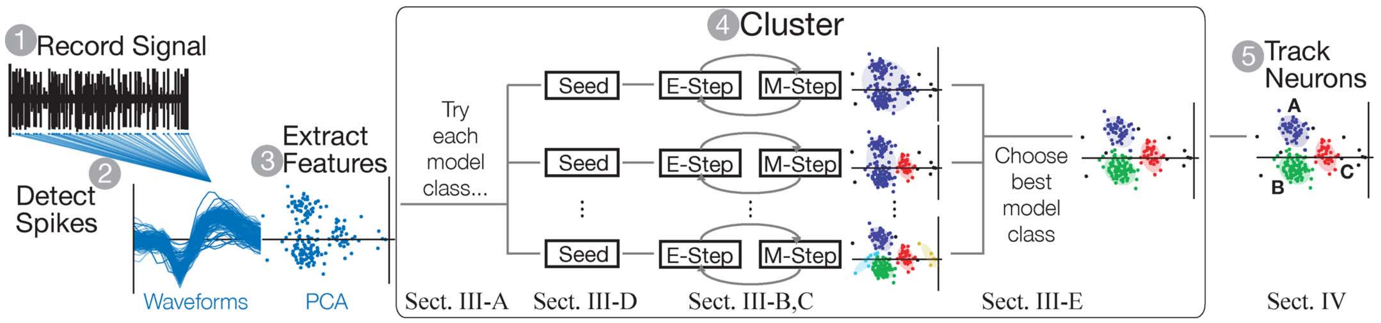


Fig. 1. Structure of the algorithm. Before clustering takes place: 1) the electrode signal is recorded for a brief sampling interval; 2) neuronal spike waveforms are detected in this voltage trace and aligned by their minimum; and 3) spikes are projected onto an appropriate feature space, such as a 2-D PCA basis. 4) Next, these data points are clustered using EM optimization of a GMM, over several possible model classes. 5) Finally, neurons are tracked by associating the clusters from the current interval to the previous interval. This entire process is repeated for every sampling interval.

The EM algorithm [24] is typically applied to estimate the mixture parameters by log-likelihood maximization. At the same time, the EM procedure assigns data points to mixture components, thereby clustering the spikes. To apply this technique, the data Y are considered "incomplete" and are augmented by Z, which is the set of spike membership indicator variables $z_i = (z_{i1}, \ldots, z_{iG_m})$

$$z_{ig} = \begin{cases} 1, & \text{if spike waveform } y_i \text{ belongs to cluster } \mathcal{C}_g \\ 0, & \text{otherwise.} \end{cases}$$

Incorporating Z, one can derive the corresponding *complete-data log-likelihood*

$$l_{CD}(\Theta_m \mid Y, Z, \mathcal{M}_m) = \sum_{i=1}^{N}\sum_{g=1}^{G_m} z_{ig} \log[\pi_g f_g(y_i \mid \theta_g)]. \quad (2)$$

EM iterates between an E-step to calculate the conditional expectation $\hat{z}_{ig} = E[z_{ig} \mid y_i, \hat{\Theta}_m] \in [0,1]$ using the current parameter estimates, and an M-step to find the parameter estimates $\hat{\Theta}_m$ that maximize 2 given $\hat{z}_{ig}$.[3] This iteration guarantees (under weak conditions) a nondecreasing $\mathcal{L}_M$ (1) and is continued until a predetermined convergence criterion.

III. MAP Clustering for Neuron Tracking

Fig. 1 outlines the general flow of the spike sorting process, with particular attention to the clustering step. Note that several candidate model classes are attempted for every dataset (see Section III-A) and that the EM iterations must be initialized by "seed clusters," or an initial guess (see Section III-D).

Let us now incorporate the sequential nature of the data sampling intervals,[4] thus establishing a Bayesian framework for MAP parameter estimation (determining parameter estimates $\hat{\Theta}_m$ and cluster membership Z) and model selection (determining the most appropriate number of clusters $\hat{G}$). Let the superscript k denote the time interval index, such that $Y^k = \{y_i^k\}_{i=1}^{N^k}$ denotes all spike observations in the kth recording interval T_k, and let $Y^{1:k} = \{Y^1, \ldots, Y^k\}$ denote all data from the 1st through the kth recording intervals. The MAP parameter estimates can be naturally derived from Bayes' rule

$$p(\Theta_m^k \mid Y^{1:k}, \mathcal{M}_m) \propto \underbrace{p(Y^k \mid \Theta_m^k, \mathcal{M}_m)}_{\text{likelihood, (1) and (4)}} \underbrace{p(\Theta_m^k \mid Y^{1:k-1}, \mathcal{M}_m)}_{\text{prior, Section III-B}}, \quad (3)$$

where Θ_m^k denotes the mixture model parameters for the mth model during T_k.[5]

A. Model Classes

While many model classes are possible within our framework, we focus on model classes that yielded the best results for neuronal signals in a PCA basis. Because the EM algorithm assumes the number of clusters G_m is known (but the number of neurons in the current signal is *a priori* unknown), we employ a typical workaround of applying EM to several candidate model classes $\mathcal{M}_m, m = 1, \ldots, \bar{M}$, varying $G_m = 1, \ldots, G_{\max}$ among them, and then selecting the best afterward. In each model class, Gaussian distributions account for the variability in each neuron's signals. There are many parsimonious models of the covariance matrices Σ_g of Gaussian distributions; we choose a shared-volume model so that all clusters are approximately the same "size" in feature space.[6]

Commonly, nonspike events are included in the data Y^k due to mistakes made by upstream components of neural signal analysis (e.g., spike detection and alignment) and must be identified as *outliers*. To capture these outlier observations, a uniform "background" distribution f_0 is added to

[3]The symbol "ˆ" will be used to denote an estimated quantity.

[4]A few comments are in order about the selection of the interval duration Δ. A short duration helps to minimize complicating nonstationarity effects; however, short durations may also result in very few spike samples per cluster, which decreases the chance of proper clustering as well as the confidence of the cluster parameter estimates (though the MAP approach helps mitigate these issues). As a rule of thumb, for 2-D feature spaces, we aim for a minimum of 10–20 spikes per neuron in the interval (corresponding to average firing rates as low as 1–2 Hz in the $\Delta = 10$ s duration used in Section V). An appropriate duration of "nonstationarity" must be determined by the user based on the experimental conditions, such as acute versus chronic electrodes, dimensions of electrodes and guide tubes, settling time, etc.

[5]Since no prior data are available in the first time interval, for T_1 we employ the ML version of GMM/EM as in Section II.

[6]Celeux and Govaert [25] thoroughly discuss covariance models. They parameterize the covariance matrix via an eigenvalue decomposition $\Sigma_g = \lambda_g D_g A_g D_g^T$, with factors describing the volume (λ_g), shape (A_g), and orientation (D_g) of the corresponding constant-deviation ellipsoids. Some or all of these factors may be constrained to be equal across clusters; for a shared-volume model, $\Sigma_g = \lambda D_g A_g D_g^T$.

the mixture model: $f_0(y_i^k \mid V^k) = 1/V^k$, with data volume $V^k = \prod_{j=1}^{d}(\max_i y_{i,j}^k - \min_i y_{i,j}^k)$.

Thus, the mixture likelihood (1) can be rewritten as

$$p(Y^k \mid \Theta_m^k, \mathcal{M}_m) = \prod_{i=1}^{N^k}\left(\pi_0^k f_0\big(y_i^k \mid \theta_0^k\big) + \sum_{g=1}^{G_m} \pi_g^k f_{\mathcal{N}}\big(y_i^k \mid \mu_g^k, \Sigma_g^k\big)\right), \quad (4)$$

where θ_0^k contains the (constant) parameter(s) of the outlier distribution and $\pi_0^k = 1 - \sum_{g=1}^{G_m} \pi_g^k$ since the mixture weights must sum to unity. The set of independent GMM parameters is then $\Theta_m^k = \{\mu_g^k, \Sigma_g^k, \pi_g^k\}_{g=1}^{G_m}$.[7]

B. Prior on Cluster Location

Next, we construct an appropriate prior on the model parameters Θ_m^k based on the clustering results from interval T_{k-1}. The model parameters are assumed to be independent across mixture components and across each parameter; therefore,

$$p(\Theta_m^k \mid \cdot) = \prod_{g=1}^{G_m}\big[p(\mu_g^k \mid \cdot)\, p(\Sigma_g^k \mid \cdot)\, p(\pi_g^k \mid \cdot)\big].$$

Since the cluster covariance Σ_g^k and the mixture weight π_g^k associated with a given neuron may vary substantially across sampling intervals, we choose diffuse priors for these less informative model elements. Most important to the practical issue of neuron tracking is the location of each cluster center μ_g^k. To establish priors on these locations, we look for the gth cluster mean μ_g^k in T_k to be near to *any* of the cluster centers found in T_{k-1}, without regard to which one, and thus utilize a Gaussian mixture to represent the cluster means found in T_{k-1}. To allow for the possibility that $\mathcal{C}_g^k$ represents a *new* neuron that was not recorded in T_{k-1}, a uniform distribution component is included as well. Thus, the prior on the gth mean combines a uniform component for new neurons and Gaussian components for all $\hat{G}^{k-1}$ clusters estimated in interval T_{k-1}:

$$p\big(\mu_g^k \mid Y^{1:k-1}, \mathcal{M}_m\big) = \frac{\omega_0^k}{V^k} + \sum_{j=1}^{\hat{G}^{k-1}} \omega_j^k f_{\mathcal{N}}\big(\mu_g^k \mid \hat{\mu}_j^{k-1}, S_j^{k-1}\big). \quad (5)$$

The parameter $\hat{\mu}_j^{k-1}$ is the estimated value of the jth cluster mean in T_{k-1}, and S_j^{k-1} is the covariance associated with the estimation that the current mean μ_g^k is in the same location as the prior mean $\hat{\mu}_j^{k-1}$. In this model, $S_j^{k-1} = R_j^{k-1} + Q^{k-1}$, where $R_j^{k-1} = (1/n_j^{k-1})\Sigma_j^{k-1}$ is the measurement covariance matrix associated with the estimation of $\hat{\mu}_j^{k-1}$ (n_j is the number of data points in cluster $\mathcal{C}_j$) and the empirically determined covariance matrix Q^{k-1} accounts for effects, such as electrode movement, which cause a cluster to drift around in the feature space.

The mixture weight ω_j^k, which represents the prior probability of assigning a cluster to the jth component, is defined as

$$\omega_j^k = \begin{cases} \dfrac{1}{c}\lambda_0^k, & j = 0 \\ \dfrac{1}{c}\mathsf{P}_{\mathsf{d},j}^k, & j = 1, \ldots, \hat{G}^{k-1}, \end{cases} \quad (6)$$

where λ_0^k is the expected number of newly appearing neurons and spurious clusters in the recording interval, $\mathsf{P}_{\mathsf{d},j}^k$ is the probability of redetecting the jth neuron found in T_{k-1}, and c is a normalizing constant.[8]

Note that the PCA feature space is recalculated at every interval to find the best PC features for that dataset. Thus, the spike waveforms from T_{k-1} must be projected to the PCA space of T_k, and then, the prior clusters' statistics are calculated in this space.

C. Extending EM to Account for Cluster Location Priors

Note that the prior (5) resembles the mixture likelihood (1) and would, in fact, share the same difficulty of maximization. We, therefore, introduce *cluster association indicators* $\mathcal{Z}^k = \{\zeta_{gj}^k\}$, which indicate hidden data that specify whether the cluster $\mathcal{C}_j^{k-1}$ found in T_{k-1} is related to the current cluster $\mathcal{C}_g^k$ in T_k, or, ideally,

$$\zeta_{gj}^k = \begin{cases} 1, & \text{if } \mu_g^k \text{ and } \hat{\mu}_j^{k-1} \text{ represent the same neuron} \\ 0, & \text{otherwise.} \end{cases}$$

Based on this approach, instead of using (5) directly, we employ the following *complete-data* log prior on the means:

$$\log p\big(\mu^k, \mathcal{Z}^k \mid Y^{1:k-1}, \mathcal{M}_m\big) = \sum_{g=1}^{G_m}\sum_{j=0}^{\hat{G}^{k-1}} \zeta_{gj}^k \log\big[\omega_j^k f_j\big(\mu_g^k \mid \psi_j^{k-1}\big)\big], \quad (7)$$

where ψ_j^{k-1} denotes the parameters of the jth mixture component in the prior ($\psi_j^{k-1} = \{\hat{\mu}_j^{k-1}, S_j^{k-1}\}$ for Gaussian components).

Rewriting (3) to include the hidden variables, taking the logarithm, and using (2) and (7) result in

$$\log p\big(\Theta_m^k, \mathcal{Z}^k \mid Y^{1:k}, Z^k, \mathcal{M}_m\big) = \sum_{i=1}^{N^k}\sum_{g=0}^{G_m} z_{ig}^k \log\big[\pi_g^k f_g\big(y_i^k \mid \theta_g^k\big)\big] + \sum_{g=1}^{G_m}\sum_{j=0}^{\hat{G}^{k-1}} \zeta_{gj}^k \log\big[\omega_j^k f_j\big(\mu_g^k \mid \psi_j^{k-1}\big)\big] + \mathcal{C}, \quad (8)$$

where $\mathcal{C}$ is a constant. The EM algorithm operates on the complete-data posterior (8) with the formulas given later.

[7]The parameter set includes only the independent elements of the symmetric matrix Σ_g^k, which will depend on the chosen parsimonious covariance model. We will treat the matrix as a single parameter for brevity.

[8]This mixture weight follows from methods in conventional target tracking [26], employing a model where each neuron is detected according to a Bernoulli trial and the number of new and spurious clusters in an interval is Poisson-distributed.

1) E-Step: As in the classical EM algorithm, given the parameter estimates from the M-step, the expectation of each spike membership indicator $\hat{z}_{ig}^k$ is

$$\hat{z}_{ig}^k = \frac{\hat{\pi}_g^k \, f_g\left(y_i^k \mid \hat{\theta}_g^k\right)}{\sum_{n=0}^{G_m} \hat{\pi}_n^k \, f_n\left(y_i^k \mid \hat{\theta}_n^k\right)}. \tag{9}$$

Recall that $f_g(y_i^k \mid \hat{\theta}_g^k)$ is a Gaussian distribution with parameters $\hat{\theta}_g^k = \{\hat{\mu}_g^k, \hat{\Sigma}_g^k\}$ for the components $g = 1, \ldots, G_m$ and an outlier density for the zeroth mixture component. The expectation of the other hidden data, the cluster association indicators, i.e., $\hat{\zeta}_{gj}^k = E[\zeta_{gj}^k \mid Y^{1:k}, \hat{\Theta}_m^k]$, has an analogous form

$$\hat{\zeta}_{gj}^k = \frac{\omega_j^k \, f_j\left(\hat{\mu}_g^k \mid \psi_j^{k-1}\right)}{\sum_{l=0}^{\hat{G}^{k-1}} \omega_l^k \, f_l\left(\hat{\mu}_g^k \mid \psi_l^{k-1}\right)}. \tag{10}$$

2) M-Step: Since the prior term in (8) is independent of the parameters π_g and Σ_g, these estimates remain the same as the classical ML clustering version. For the mixture weights

$$\hat{\pi}_g^k = \frac{n_g^k}{N^k}, \tag{11}$$

where $n_g^k = \sum_{i=1}^{N^k} \hat{z}_{ig}^k$, and for the shared-volume form of the covariance matrix[9] [25]

$$\hat{\Sigma}_g^k = \lambda^k \frac{W_g^k}{|W_g^k|^{1/d}}, \tag{12}$$

where $\lambda^k = \sum_{g=1}^{G_m^k} |W_g^k|^{1/d}/N^k$ and $W_g^k = \sum_{i=1}^{N^k} \hat{z}_{ig}^k (y_i^k - \hat{\mu}_g^k)(y_i^k - \hat{\mu}_g^k)^T$. Maximizing (8) with respect to μ_g^k results in the estimate

$$\hat{\mu}_g^k = \left[\sum_{i=1}^{N^k} \hat{z}_{ig}^k (\hat{\Sigma}_g^k)^{-1} + \sum_{j=1}^{\hat{G}^{k-1}} \hat{\zeta}_{gj}^k (S_j^{k-1})^{-1}\right]^{-1} \times \left[\sum_{i=1}^{N^k} \hat{z}_{ig}^k (\hat{\Sigma}_g^k)^{-1} y_i^k + \sum_{j=1}^{\hat{G}^{k-1}} \hat{\zeta}_{gj}^k (S_j^{k-1})^{-1} \hat{\mu}_j^{k-1}\right], \tag{13}$$

in contrast to the ML estimation of the cluster center location $\hat{\mu}_g^k = \sum_{i=1}^{N^k} \hat{z}_{ig}^k y_i^k / \sum_{i=1}^{N^k} \hat{z}_{ig}^k$. Note that (13) has the form of a weighted average of the data points y_i^k and the prior means $\hat{\mu}_j^{k-1}$, with the weights governed by their (fuzzy) association to cluster $\mathcal{C}_g^k$ and the respective covariance matrices. A minor drawback is that (13) is a function of the parameters $\hat{\Sigma}_g^k$, thus implying the need to simultaneously solve for both $\hat{\mu}_g^k$ and $\hat{\Sigma}_g^k$. However, one may approximate $\hat{\Sigma}_g^k$ by its value from the previous EM iteration ($\hat{\Sigma}_g^k$ varies little across consecutive EM iterations) to solve (13), and then, update $\hat{\Sigma}_g^k$ using (12).

D. Generating Seed Clusters

The EM algorithm requires initial values to seed its iterations. The choice of these seed clusters is a key issue, as the EM algorithm is highly susceptible to finding local optima near its initial values. Assuming again that the clusters found in T_{k-1} provide a good starting point, an obvious seeding strategy is to group the current data points according to the closest prior cluster. For this purpose, we use the (squared) Mahalanobis distance between the ith data point y_i^k in T_k and the jth cluster center estimated from T_{k-1}

$$d_j^2(y_i^k) = (y_i^k - \hat{\mu}_j^{k-1})^T (\hat{\Sigma}_j^{k-1})^{-1} (y_i^k - \hat{\mu}_j^{k-1}). \tag{14}$$

However, recall that the EM algorithm is applied to a range of candidate model classes, with varying model order (numbers of clusters). A complication arises in cases where the candidate model order G_m is different from $\hat{G}^{k-1}$, which is the model order estimated in T_{k-1}. Such differences can arise, for example, when neurons go silent or new neural signals are introduced between sampling intervals. Later, we outline our approach for each of the three possible relations between G_m and $\hat{G}^{k-1}$.

1) Case $G_m = \hat{G}^{k-1}$: The seed assignment process assigns each observation to the closest prior cluster: each y_i^k is assigned to the jth cluster, where j is the index that minimizes $d_j^2(y_i^k)$ in (14).

2) Case $G_m < \hat{G}^{k-1}$: The goal is to produce good clustering seeds when $\Delta G = \hat{G}^{k-1} - G_m$ neuron(s) disappear between sampling intervals. All $\binom{\hat{G}^{k-1}}{G_m}$ combinations of the $\hat{G}^{k-1}$ prior clusters are evaluated to determine which set of G_m prior clusters minimizes the sum of the squared Mahalanobis distance. The left column of Fig. 4 displays a seeding example with $\hat{G}^{k-1} = 3$ and $G_m = 2$.

3) Case $G_m > \hat{G}^{k-1}$: In this case, $\Delta G = G_m - \hat{G}^{k-1}$ "extra" seed clusters must be generated. Such a situation can occur when ΔG new neurons have been detected and a new cluster must be created for each. The spikes from T_k are first assigned to the $\hat{G}^{k-1}$ prior clusters, as in the first case before, after which we wish to divide the cluster that is most likely to contain multiple neurons (see the right column of Fig. 4 for an example with $\hat{G}^{k-1} = 1$ and $G_m = 2$). Since such a group is likely to have a larger data spread, the group with the largest average point-to-centroid Euclidean distance is chosen. This cluster's points are projected onto its principal axis, and then split between the adjacent points that have the largest distance between them [see Fig. 4(f) and (g)]. This is essentially a one-step divisive hierarchical clustering technique. The aforementioned identification and splitting of groups are repeated as necessary for $\Delta G > 1$.

E. Selecting the Model Class $\mathcal{M}_m$

The model selection step, which estimates how many clusters exist in the signal, is based on a Bayesian approach as well, by taking the model with highest probability

$$P(\mathcal{M}_m \mid Y^{1:k}) = \frac{1}{\mathcal{D}} p(Y^k \mid Y^{1:k-1}, \mathcal{M}_m) P(\mathcal{M}_m \mid Y^{1:k-1}), \tag{15}$$

where $\mathcal{D}$ is a normalizing constant. This probability (15) is difficult to compute because the evidence $p(Y^k \mid Y^{1:k-1}, \mathcal{M}_m)$ theoretically requires an integration over all possible parameters. However, Laplace's method for asymptotic approximation of

[9]The equation assuming a fully variable covariance is $\hat{\Sigma}_g^k = (1/n_g^k) \sum_{i=1}^{N^k} \hat{z}_{ig}^k (y_i^k - \hat{\mu}_g^k)(y_i^k - \hat{\mu}_g^k)^T$.

integrals [20], [27][10] can be employed to estimate a value of the evidence term while evaluating only at the MAP parameters $\hat{\Theta}_m^k$

$$p(Y^k \mid Y^{1:k-1}, \mathcal{M}_m) \approx p(Y^k \mid \hat{\Theta}_m^k, \mathcal{M}_m) \times p(\hat{\Theta}_m^k \mid Y^{1:k-1}, \mathcal{M}_m)(2\pi)^{\eta_m/2} |\mathbf{H}_m(\hat{\Theta}_m^k)|^{-1/2}, \quad (16)$$

where η_m is the number of independent parameters in model $\mathcal{M}_m$. The first factor is the well-known likelihood of the Gaussian mixture (4). The other factors, collectively known as the Ockham factor since they penalize the complexity of the model parameterization, include the parameter prior (5) and the Hessian matrix

$$\mathbf{H}_m(\hat{\Theta}_m^k) = -\nabla\nabla|_{\hat{\Theta}_m^k} \log p\big(Y^k|\Theta_m^k, \mathcal{M}_m) p(\Theta_m^k|Y^{1:k-1}, \mathcal{M}_m\big),$$

which has an analytical expression for the model classes under consideration [28]. Most popular model selection approaches, such as the Akaike information criterion (AIC) and Bayes information criterion (BIC), are essentially approximations to (16) and specific to the ML method [27]. For our application, the Laplace approach naturally incorporates the prior on Θ_m^k.

The model class prior $P(\mathcal{M}_m \mid Y^{1:k-1})$ in (15) is the model selection result from T_{k-1}, under the assumption that the model class is constant. However, since the model class may change (e.g., neural signal sources appear or disappear), we use a weighted mixture of the previous result and a uniform prior: $P(\mathcal{M}_m \mid Y^{1:k-1}) \leftarrow \alpha P(\mathcal{M}_m \mid Y^{1:k-1}) + (1-\alpha)(1/\bar{M})$, where $\bar{M}$ is the total number of model classes under consideration. The parameter $\alpha \in [0,1]$ (we use $\alpha = 0.95$) imposes a "forgetting factor" on the prior, which ensures a nontrivial probability of each model class at every sampling interval.

IV. Tracking Clusters Across Intervals

Ultimately, our goal is to "track" individual neurons—i.e., to associate specific neurons with specific signal clusters over time. Viewing this as a data association task on the means, the quantity $\hat{\zeta}_{gj}^k$ already encodes the probability that current cluster $\mathcal{C}_g^k$ is associated with prior cluster $\mathcal{C}_j^{k-1}$, relative to all $\hat{G}^{k-1}+1$ components in the prior (5). Each current cluster $\mathcal{C}_g^k$ is, therefore, matched to a prior cluster $\mathcal{C}_{j^*}^{k-1}$ via $j^* = \arg\max_j \hat{\zeta}_{gj}^k$. Thus, at the completion of the EM iterations, in addition to the model parameters $\hat{\Theta}_m^k$ and the cluster memberships $\hat{z}_{ig}^k$, the algorithm also yields cluster associations $\hat{\zeta}_{gj}^k$ for tracking.

New neurons are identified when $\mathcal{C}_g^k$ is matched to the uniform distribution, thus highlighting the importance of a uniform component in the prior. Disappearing neurons are identified when prior clusters are not matched to any current clusters. Note that a time interval may include both addition and subtraction of neurons, thus changing the identities of the neurons even when the number of neurons remains the same. Additionally, *multiple* current clusters $\mathcal{C}_g^k$ may match the same prior cluster $\mathcal{C}_j^{k-1}$. While a single-match nearest neighbor approach could be used, we wish to allow for "splits" of the neuronal signal components (when the signals of two (or more) previously indistinguishable neurons are now separated).

V. Experimental Results

The proposed MAP algorithm was applied to recordings from macaque parietal cortex, collected in acute recording sessions with platinum–iridium, 1.5 MΩ impedance electrodes in a microdrive controlled by our autonomous electrode positioning algorithm [2]–[4]. Spikes were detected in the recorded voltage stream via a wavelet matching approach [29], aligned by their minimum, and projected onto a 2-D PCA space prior to clustering.

As noted earlier, EM optimization of a GMM with ML parameters has shown its effectiveness in many clustering applications [20] and, specifically, spike sorting [1], [7], [16]–[19]. Thus, we compare our proposed MAP method to such a technique, which we have also used in hundreds of recording sessions. We previously chose this method due to its success compared to other spike sorting options. In the implementation of the contrasting ML approach, seed clusters are generated from a standard hierarchical agglomerative technique and model order is selected according to Bayesian information criterion (BIC),[11] following the suggestions of [30]. Both the MAP and ML implementations use the same models for the components' covariance matrices and the same uniform "background" mixture component to capture outliers.

A. Detail: Sequence of Consecutive Recording Intervals

Fig. 2 displays clustering results over a sequence of 12 consecutive recording intervals, chosen to highlight how the MAP algorithm enables neuron tracking, especially as compared to alternatives. Each sampling interval lasts 10 s, with separating intervals of about 25 s during which no signals are sampled (while the control algorithm repositions the electrodes). Each interval's data were clustered using its own PCA features; however, for consistent visualization, these datasets are plotted in the same PCA feature space (PCs from $k = 7$). Although it is impossible to know the actual spike–neuron associations conclusively, the results are compared to a best-effort manual clustering, as determined by an expert's thorough examination of *both* the spikes' full waveforms and their PCA features (whereas the automated clustering uses only PCA features). While manual sorting is not foolproof, it provides the most common and basic baseline. In addition to MAP and ML algorithm results, a k-means clustering result is also presented, with the number of clusters k manually selected to match the number of clusters in the expert results. Listed for each interval in Fig. 2 is the percentage of spikes

[10]For *globally identifiable* functions (having a single maximizing point at $\hat{\Theta}_m^k$), Laplace's method states that the right-hand side (RHS) of (16) approaches $p(Y^k \mid Y^{1:k-1}, \mathcal{M}_m)$ asymptotically as the amount of data Y^k increases. Although we know only that $\hat{\Theta}_m^k$ is a local maximum (and do not show global identifiability), the approximation offers a valuable measure with fewer assumptions than prevailing information criteria.

[11]BIC $\equiv 2l_M(\hat{\Theta}_m^k \mid Y^k, \mathcal{M}_m) - \eta_m \log N^k$, for maximized mixture log-likelihood l_M, and number of independent model parameters η_m.

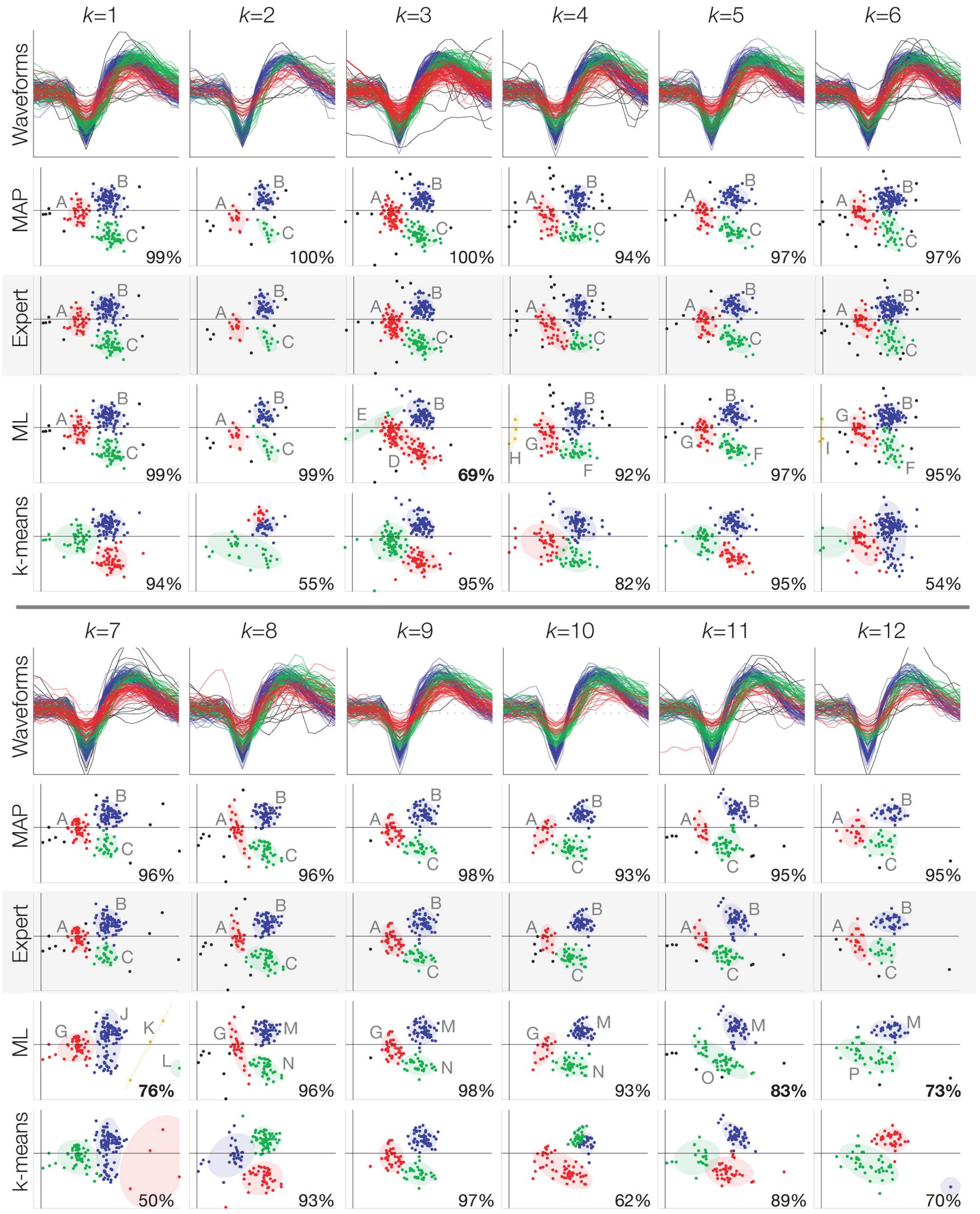


Fig. 2. Cluster results over 12 consecutive recording intervals, displayed in a common PCA space. Rows: 1) extracted, aligned waveforms from the interval (colored by MAP result); 2) results from our MAP algorithm; 3) manual sorting by expert; 4) results from the baseline (ML) algorithm; and 5) results from k-means, with $k = 3$. Shaded ellipses indicate $\sigma = 2$ for each cluster; percentile is of spikes classified similarly to the expert; capital letters label neuron ID; black points indicate classification as outliers.

TABLE I
CLUSTER STATISTICS OF SELECTED INTERVALS FROM FIG. 2

k	MAP					ML				
	g	Exp.[a]	n_g[b]	Err.[c]	ΔFR[d]	g	Exp.	n_g	Err.	ΔFR
3	1	A	73	0%	0%	1	A	76	4%	4%
	2	B	53	0%	0%	2	—	5	n/a	n/a
	3	C	65	0%	0%	3	C	115	86%	77%
4	1	A	73	4%	1%	1	A	74	3%	3%
	2	B	32	23%	23%	2	B	32	23%	23%
	3	C	36	17%	12%	3	C	36	17%	12%
						4	—	4	n/a	n/a
7	1	A	74	4%	4%	1	A	104	35%	35%
	2	B	29	7%	7%	2	—	1	n/a	n/a
	3	C	48	7%	7%	3	C	53	18%	18%
						4	—	3	n/a	n/a
11	1	A	49	9%	9%	1	A	54	0%	0%
	2	B	41	14%	14%	2	B	53	53%	47%
	3	C	16	7%	7%					
12	1	A	39	0%	0%	1	A	40	3%	3%
	2	B	22	10%	10%	2	C	41	105%	105%
	3	C	20	20%	0%					

[a]Exp.: expert cluster ID to which the gth cluster was matched.
[b]n_g: number of spikes in the gth cluster.
[c]Err.: error versus the expert; Err. $=$ (MC+FP)/$n_{g,\text{expert}}$, where MC is the number of missed classifications and FP is the number of false positives.
[d]ΔFR: percent difference in firing rate from the expert cluster.

classified similarly to the expert's results,[12] and each cluster is labeled with a "neuron ID," thus indicating the neuron that it tracks.[13] Table I provides a detailed view of the intervals where the MAP and ML results differed significantly.

Ostensibly, spikes from the same three generating neurons (labeled A, B, and C) persist through the 12 sampling intervals of Fig. 2, as determined in the expert clusters. The clustering challenge is difficult, however, as the spike waveform features are not highly separated and the firing rates (and thus numbers of data points) are sometimes low. Note that the MAP algorithm consistently identifies three clusters in roughly the same PCA position. The ML algorithm often provides good results, but some intervals show incongruous (though statistically sound) results, seemingly more susceptible to noise variations. Meanwhile, the k-means solution is unreliable, even with the advantage of knowing the model order *a priori*.

Even a small number of intervals with poor results significantly impacts our ability to track neurons over time. For example, in T_3 (i.e., $k = 3$), the ML method groups most spikes from neurons A and C into a single cluster, whose mean is relatively distant from the means of T_2. When attempting to associate the clusters across these intervals, this result is interpreted as the loss of neurons A and C and the appearance of a new neuron (D) in T_3 (rather than tracking neurons A and C from T_2, as the MAP method does). Next, when the spikes from neurons A and C are (mostly) correctly classified by the ML method at $k = 4$, they are considered as "new" neurons G and F since their mean locations are appreciably removed from the prior mean in T_3. Such errors, occurring also at $k = 7$, $k = 11$, and $k = 12$, prevent neuron tracking in the ML method. Note that our MAP algorithm results in one track per neuron lasting across all 12 intervals, whereas the ML algorithm cannot track the three neurons and also generates many spurious tracks.

[12]For this calculation, each cluster is matched to the expert cluster sharing the most spikes, and the number of spikes these clusters have in common is considered to be classified similarly.

[13]Because the ML method does not include a natural data association process, the following procedure was used to test its neuron tracking ability. The clusters from T_1 are assigned neuron IDs (A, B, C, ...). Thereafter, a cluster in T_k is associated with the nearest cluster in T_{k-1}, provided its mean lies within 2 standard deviations of the prior mean location, using the same covariance Q^k (discussed in Section III-B) as used in the MAP algorithm. If no match is found, a new track is created (new ID assigned). Note that the presented MAP tracking results are identical when using this procedure or when using the procedure in Section IV.

B. Consistency and Tracking Over Longer Time Frames

Fig. 3 provides comparisons between the MAP and ML algorithms' consistency and tracking ability in two sessions lasting over an hour (over 100 sampling intervals), displaying for each sampling interval the locations of all cluster means in the first principal component (again, a common PCA space is chosen for display only). The connected "neuron tracks" (using the procedure in Section V-A for tracking ML clusters) are labeled by neuron ID (A–Z, a–z, AA–AZ, ...). The unbroken, single-ID tracks in the MAP results indicate our algorithm has tracked putative neurons over the entire duration of the recordings, exhibiting the algorithm's ability to consistently cluster many consecutive sampling intervals and successfully associate clusters across each interval. By contrast, the ML tracks are often broken, even though it appears that the clusters should persist over time. Also, the ML method produces many spurious clusters, thus resulting in a large number of (presumably) false neuron tracks.

Broadening time frames further, we can attempt to quantify the notion of clustering consistency via the change in the number of clusters from interval to interval. Computing $\Psi = \sum_{s=1}^{S} \sum_{k=2}^{K_s} |\hat{G}^k - \hat{G}^{k-1}|$ over all time intervals of each recording session provides a quantitative measure of "inconsistency"—however, many changes in the value of G are correct, as the number of recorded neurons may vary over the recording session. Examining a set of 100 consecutive recording sessions, comprising about one month of recording trials and 21 914 total sampling intervals, $\Psi = 3516$ for the MAP algorithm, compared to $\Psi = 17\,646$ for the ML algorithm, which is equivalent to an 80% decrease.

C. Changing Numbers of Clusters

It is necessary to recognize appearing and disappearing neurons in the recorded signal, a goal somewhat at odds with an increase in clustering consistency. The columns of Fig. 4 present a detailed view of two examples involving transitions to fewer and to more clusters. In both cases, although $\hat{G}^{k-1}$ is different from the number of neurons at time k, the MAP algorithm determines the correct number of clusters $\hat{G}^k$, as sufficient evidence exists that the number of neurons in T_k is "inconsistent" with the previous interval T_{k-1}. The numbers quantifying this evidence are provided at the bottom of the figure, showing the difference in log probability of the chosen model class versus the model class selected in T_{k-1}. Also listed are three

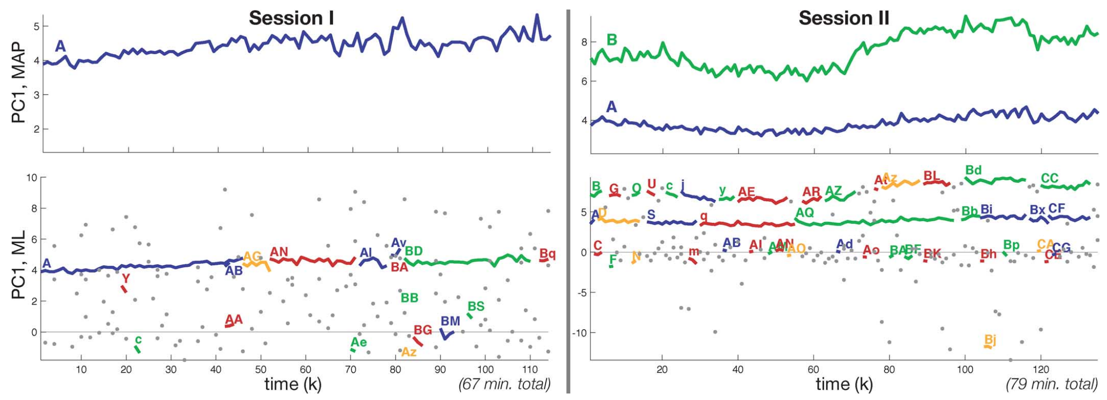
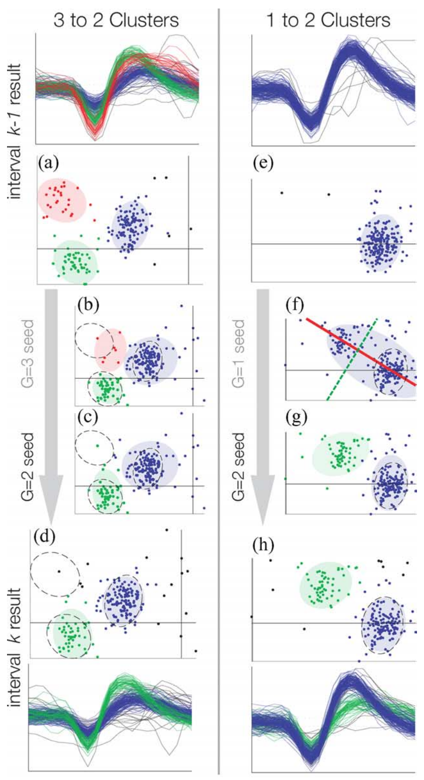


Fig. 3. Graphical comparison of the tracking ability of the MAP versus ML algorithms in two sessions, each lasting over an hour. Each plot depicts the first principal component of the cluster means versus time. Neuron tracks are represented by lines and labeled by letter ID. Gray dots indicate a neuron ID lasting only one time step (for which the letter label is suppressed). Note that the MAP algorithm (upper graphs) produces unbroken tracks with single IDs, whereas the ML algorithm (lower graphs) clusters the data inconsistently, losing the tracks frequently, and generating many spurious clusters.



Interval k Model Selection

$\mathcal{M}_2$ vs. $\mathcal{M}_3$		$\mathcal{M}_2$ vs. $\mathcal{M}_1$
–5.0	Δlog(data likelihood)	63.8
15.6	Δlog(Ockham factor)	–18.1
–4.6	Δlog(model prior)	–4.6
6.0	Δlog(model probability)	41.1

Fig. 4. Examples of consecutive recording intervals where the number of neurons change across intervals. (Left column) Example showing decrease from $\hat{G}^{k-1} = 3$ to $\hat{G}^k = 2$ clusters. (Right column) Example showing increase from $\hat{G}^{k-1} = 1$ to $\hat{G}^k = 2$ clusters. Black dashed 2-sigma ellipses show locations of prior clusters. (b) and (c) Seed clusters for $G = 2$ are chosen by keeping best two of the three prior clusters. (f) and (g) Seed clusters for $G = 2$ are formed by splitting the one prior cluster along its principal axis (think red line) at the point of largest gap (dashed green line). Below the plots are the quantities used to select the model class for interval k. We list the difference between values for the chosen model class (with two clusters, dubbed $\mathcal{M}_2$) and the model class selected in the previous interval (at $k-1$). Recall that the model probability may be decomposed into three key factors, whose values are listed here as well (see Section III-E, particularly (15) and 16 for the representative formulas).

main components of the model class probability (15)—the data likelihood, Ockham factor, and model prior—to show the relative contribution of these terms. Recall that the model evidence (16), $p(Y^k \mid Y^{1:k-1}, \mathcal{M}_m)$, is the combination of the data likelihood (which generally increases with the number of mixture components) and the Ockham factor (which penalizes such added model complexity). In each depicted scenario, the $\hat{G}^k = 2$ model class was chosen with probability greater than 99%. Also shown in this figure are plots of the seed clusters for the selected model class and how these seed clusters are generated from the T_{k-1} result.

VI. Conclusion

We have detailed a Bayesian clustering algorithm to optimize a mixture model via EM, and our results show how this MAP algorithm provides more consistent clustering and improves tracking of neurons over time. In addition to constructing a novel "mixture prior" on the cluster locations, we have introduced a new set of latent variables and derived the resulting expressions for incorporating the prior into a MAP EM algorithm. We have also created a new process for generating seed clusters and proposed a suitable model class selection method. From an electrophysiology perspective, this neuron tracking algorithm decreases the corruption of neuronal statistics, such as firing rate, caused by misclassification and can increase the number of scientifically useful neurons identified on the signal. From the perspective of building an autonomous electrode positioning algorithm that tries to maximize the SNR of a particular neuron, consistently tracking the neurons' identities is essential in determining appropriate electrode control.

Although we have focused on providing more consistent results, our algorithm also performs well when the prior is not similar to the current clusters (as in Section V-C). The prior's construction as a mixture of densities effectively influences the posterior cluster locations but assumes neither a certain number of clusters nor the *a priori* association of *particular* current and prior clusters. Thus, our algorithm is not unduly biased by the prior when evidence suggests the appearance (or disappearance) of neurons. These same properties of the algorithm allow it to recover from errors made in clustering the previous interval.

Our algorithm is more likely to avoid poor local optima because of our superior seeding method and because the mixture prior on the cluster means better guides the EM process. (Although the result is not guaranteed to be globally optimal, it tends to be the *desired* solution.) Also, our model selection procedure is quite effective because: 1) the model evidence increases when the parameters are near those of the last interval, as influenced by our MAP EM approach; 2) the model prior biases the result toward a consistent number of clusters; and 3) Laplace's asymptotic approximation is better than other methods at distinguishing between "close calls."

In deconstructing the algorithm's performance, the contribution from good seed clusters dominates when there are many data points and/or when the covariance S_j^{k-1} is large (both more likely for longer recording intervals). For short sampling intervals (with relatively few data points but effectively stationary signals), the use of cluster location priors during EM plays a stronger role and enables the same clusters to be identified with few data points, as the parameter prior increases the posterior probability that cluster means lie in the same place and influences the model evidence.

Because we aim for real-time applications to autonomous electrode positioning and brain–machine interfaces, computational considerations are important. The total processing time for each sampling interval in the results of Section V averaged $\sim$2 s using nonoptimized MATLAB code on a 3.2-GHz Pentium D processor, which is well within the needs of our current application. The main computational burden is the calculation of the Hessian matrix, which may be removed by using the BIC to approximate model evidence (instead of Laplace's method) while maintaining most benefits of our approach. In this case, the average time per interval drops to about 0.25 s, which is, in fact, about 40% faster than the ML method—although the MAP method is more complex, it usually requires fewer EM iterations to converge.

A few elements may be considered for future work. The tracking algorithm may be made more robust (for example, for temporarily silent neurons) by incorporating prior information from several time intervals, and perhaps, implementing a multiple hypothesis tracking approach (see [28]). Also, choices of feature space other than PCA, as well as a neuron's "dynamics" in this space, may be considered further.

Acknowledgment

The authors thank R. Andersen and his laboratory, particularly G. Mulliken, for collaboration and test data, and Z. Nadasdy and N. Hudson for several helpful discussions.

References

[1] M. S. Lewicki, "A review of methods for spike sorting: The detection and classification of neural action potentials," *Netw.: Comput. Neural Syst.*, vol. 9, pp. R53–R78, 1998.

[2] J. G. Cham, E. A. Branchaud, Z. Nenadic, B. Greger, R. A. Andersen, and J. W. Burdick, "Semi-chronic motorized microdrive and control algorithm for autonomously isolating and maintaining optimal extracellular action potentials," *J. Neurophysiol.*, vol. 93, pp. 570–579, Jan. 2005.

[3] Z. Nenadic and J. W. Burdick, "A control algorithm for autonomous optimization of extracellular recordings," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 5, pp. 941–955, May 2006.

[4] E. A. Branchaud, "An algorithm for the autonomous isolation of neurons in extracellular recordings," Ph.D. dissertation, California Inst. Technol., Pasadena, Jun. 2006.

[5] J. G. Cham, M. T. Wolf, R. A. Andersen, and J. W. Burdick, "Miniature neural interface microdrive using parylene-coated layered manufacturing," in *Proc. IEEE/RAS-EMBS Int. Conf. Biomed. Robot. Biomechatron. (BioRob)*, Feb. 2006, pp. 721–726.

[6] R. K. Snider and A. B. Bonds, "Classification of non-stationary neural signals," *J. Neurosci. Methods*, vol. 84, no. 1/2, pp. 155–166, Oct. 1998.

[7] A. Bar-Hillel, A. Spiro, and E. Stark, "Spike sorting: Bayesian clustering of non-stationary data," *J. Neurosci. Methods*, vol. 157, no. 2, pp. 303–316, Oct. 2006.

[8] A. Emondi, S. Rebrik, A. Kurgansky, and K. Miller, "Tracking neurons recorded from tetrodes across time," *J. Neurosci. Methods*, vol. 135, pp. 95–105, 2004.

[9] M. S. Fee, P. P. Mitra, and D. Kleinfeld, "Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability," *J. Neurosci. Methods*, vol. 69, no. 2, pp. 175–188, Nov. 1996.

[10] M. Salganicoff, M. Sarna, L. Sax, and G. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. Algorithms and implementation," *J. Neurosci. Methods*, vol. 25, no. 3, pp. 181–187, Oct. 1988.

[11] E. Hulata, R. Segev, and E. Ben-Jacob, "A method for spike sorting and detection based on wavelet packets and Shannon's mutual information," *J. Neurosci. Methods*, vol. 117, no. 1, pp. 1–12, May 2002.

[12] F. Ohberg, H. Johansson, M. Bergenheim, J. Pedersen, and M. Djupsjobacka, "A neural network approach to real-time spike discrimination during simultaneous recording from several multi-unit nerve filaments," *J. Neurosci. Methods*, vol. 64, no. 2, pp. 181–187, Feb. 1996.

[13] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Comput.*, vol. 16, no. 8, pp. 1661–1687, 2004.

[14] T. I. Aksenova, O. K. Chibirova, O. A. Dryga, I. V. Tetko, A.-L. Benabid, and A. E. P. Villa, "An unsupervised automatic method for sorting neuronal spike waveforms in awake and freely moving animals," *Methods*, vol. 30, no. 2, pp. 178–187, Jun. 2003.

[15] C. Vargas-Irwin and J. P. Donoghue, "Automated spike sorting using density grid contour clustering and subtractive waveform decomposition," *J. Neurosci. Methods*, vol. 164, no. 1, pp. 1–18, 2007.

[16] M. S. Lewicki, "Bayesian modeling and classification of neural signals," *Neural Comput.*, vol. 6, pp. 1005–1030, 1994.

[17] S. Shoham, M. R. Fellows, and R. A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *J. Neurosci. Methods*, vol. 127, no. 2, pp. 111–122, Aug. 2003.

[18] K. H. Kim and S. J. Kim, "Method for unsupervised classification of multiunit neural signal recording under low signal-to-noise ratio," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 4, pp. 421–431, Apr. 2003.

[19] F. Wood, M. Fellows, J. Donoghue, and M. J. Black, "Automatic spike sorting for neural decoding," in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc. (EMBS)*, 2004, pp. 4009–4012.

[20] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.

[21] P. Cheeseman and J. Stutz, "Bayesian classification (AutoClass): Theory and results," in *Advances in Knowledge Discovery and Data Mining*, U. Fayyard, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Cambridge, MA: AAAI/MIT Press, 1996, ch. 6, pp. 61–83.

[22] M. T. Wolf and J. W. Burdick, "Spike clustering and neuron tracking over successive time windows," in *Proc. IEEE EMBS Conf. Neural Eng. (NER)*, 2007, pp. 659–665.

[23] F. Wood and M. J. Black, "A nonparametric Bayesian alternative to spike sorting," *J. Neurosci. Methods*, vol. 173, no. 1, pp. 1–12, 2008.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Stat. Soc. Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[25] G. Celeux and G. Govaert, "Gaussian pasimonious clustering models," *Pattern Recognit.*, vol. 28, pp. 781–793, 1995.

[26] T. Kurien, "Issues in the design of practical multitarget tracking algorithms," in *Multitarget-Multisensor Tracking: Advanced Applications*. Boston, MA: Artech House, 1990, pp. 43–83.

[27] J. L. Beck and K.-V. Yuen, "Model selection using response measurements: Bayesian probabilistic approach," *J. Eng. Mech.*, vol. 130, no. 2, pp. 192–203, 2004.
[28] M. T. Wolf, "Target tracking using clustered measurements, with applications to autonomous brain–machine interfaces," Ph.D. dissertation, California Inst. Technol., Pasadena, Jun. 2008.
[29] Z. Nenadic and J. W. Burdick, "Spike detection using the continuous wavelet transform," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 1, pp. 74–87, Jan. 2005.
[30] C. Fraley and A. E. Raftery, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *Comput. J.*, vol. 41, no. 8, pp. 578–588, 1998.

Michael T. Wolf (M'07) received the B.S. degree in mechanical engineering from Stanford University, Stanford, CA, and the M.S. and Ph.D. degrees in mechanical engineering from California Institute of Technology, Pasadena.

Since 2008, he has been with the National Aeronautics and Space Administration Jet Propulsion Laboratory, California Institute of Technology. His current research interests include medical and assistive robotics, robot motion planning, and multitarget tracking.

Dr. Wolf received the Best Thesis Award in Mechanical Engineering from California Institute of Technology in 2008 for his work on neuronal signal tracking for neural prosthetics.

Joel W. Burdick (M'05) received the Undergraduate degree in mechanical engineering from Duke University, Durham, NC, and the M.S. and Ph.D. degrees in mechanical engineering from Stanford University, Stanford, CA.

Since May 1988, he has been with the Department of Mechanical Engineering, California Institute of Technology, Pasadena. His current research interests include sensor-based robot motion planning, multifingered grasping, neural prosthetics, and rehabilitation of spinal cord injuries.

Dr. Burdick has been a finalist for the Best Paper Award for the IEEE International Conference on Robotics and Automation in 1993, 1999, 2000, and 2005. He was appointed an IEEE Robotics Society Distinguished Lecturer in 2003. He received the National Science Foundation Presidential Young Investigator Award, the Office of Naval Research Young Investigator Award, and the Feynman Fellowship from California Institute of Technology.