Journal of Neuroscience Methods 196 (2011) 159–169

Contents lists available at ScienceDirect

Journal of Neuroscience Methods

journal homepage: www.elsevier.com/locate/jneumeth

Kalman filter mixture model for spike sorting of non-stationary data

Ana Calabrese[a,*], Liam Paninski[a,b]

[a] Center for Theoretical Neuroscience, Columbia University, 1051 Riverside Dr., New York, NY 10032, United States
[b] Department of Statistics, Columbia University, 1255 Amsterdam Ave., New York, NY 10027, United States

ARTICLE INFO

Article history:
Received 28 June 2010
Received in revised form 8 November 2010
Accepted 7 December 2010

Keywords:
Spike sorting
Non-stationarity
Kalman filter
Hidden Markov model
Mixture model
EM algorithm

ABSTRACT

Nonstationarity in extracellular recordings can present a major problem during in vivo experiments. In this paper we present automatic methods for tracking time-varying spike shapes. Our algorithm is based on a computationally efficient Kalman filter model; the recursive nature of this model allows for on-line implementation of the method. The model parameters can be estimated using a standard expectation-maximization approach. In addition, refractory effects may be incorporated via closely related hidden Markov model techniques. We present an analysis of the algorithm's performance on both simulated and real data.

1. Introduction

Despite decades of effort, spike sorting remains one of the frustratingly unsolved (or more accurately, half-solved) problems in statistical neuroscience: many spike sorting algorithms work quite well in some cases, but we still lack computationally efficient and robust methods that perform well in all desired settings. The difficulties here are well-known and include issues of nonstationarity, non-Gaussianity, temporal dependencies between spikes (e.g., refractoriness), and overlapping spike shapes due to synchronous activity in nearby neurons (spike "collisions"); see, e.g., Quian Quiroga (2007) for a brief overview, and Lewicki (1998) and Sahani (1999) for two in-depth reviews which remain fairly current over a decade after their original publication.

In this brief note we do not attempt anything so ambitious as a full solution of the spike sorting problem; instead, we focus on developing computationally efficient methods for the somewhat more tractable subproblem of automatically tracking time-varying (nonstationary) spike waveform shapes.[1] These nonstationarities are quite common in long experiments in vivo, and are most often due to drifts in the position of the recording electrode relative to the cell body, but can also be due to changes in the health of the cell over the course of the experiment, for example.

These nonstationarity issues have been addressed previously, and in fact with models that are more powerful in some respects than the approach we propose here (Pouzat et al., 2004; Bar-Hillel et al., 2006; Gasthaus et al., 2009). Our approach has some advantages in terms of simplicity and computational efficiency; we will discuss these issues at more length below. To summarize briefly, we introduce a simple Kalman-filter model of the nonstationary spike sorting problem; given this model, it is straightforward to adapt efficient recursive algorithms to perform inference and clustering. Indeed, the recursive nature of the Kalman-based algorithms makes it easy to implement these techniques in an on-line manner. In addition, we can adapt conceptually similar hidden Markov methods to incorporate simple refractory effects in the model, which can enhance clustering accuracy in the case of low-SNR, high firing rate recordings.

We begin by briefly reviewing the classic mixture-of-Gaussians (MoG) model for spike sorting (Section 2.1), along with the associated expectation-maximization (EM) algorithms for parameter inference in this model. We then extend this basic MoG approach to include nonstationary behavior in Section 2.2, and discuss online implementations of the parameter estimation algorithm in Section 2.2.1. Finally, we incorporate Markovian refractory effects in Section 2.3, and demonstrate the application of the methods to simulated and real data in Section 3.

* Corresponding author at: Columbia University, Schermerhorn Hall, 1190 Amsterdam Ave., New York, NY 10027, United States. Tel.: +1 212 854 5448.
E-mail address: amc2257@columbia.edu (A. Calabrese).

[1] Note that we will not address the important case of temporary changes in spike shape due to partial inactivation of sodium channels during the relative refractory period (Fee et al., 1996; Lewicki, 1998; Quirk and Wilson, 1999; Pouzat et al., 2004); instead, we have longer-lasting, slower nonstationarities in mind here.

0165-0270/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2010.12.002

2. Methods

2.1. The Gaussian mixture model and the EM algorithm

In this section we will briefly introduce our setting, along with some basic ideas and notation which will be useful in the remainder of the paper. Let us first review a few basic steps which are common to almost all spike-sorting approaches: we assume first that the the continuous raw extracellular voltage data has been band-pass filtered, to discard both slow fluctuations in the voltage signal as well as high-frequency noise. Second, putative spikes are detected, usually using simple threshold-crossing methods, but perhaps via more sophisticated spike-by-spike "peeling" methods capable of resolving multiple overlapping spike waveforms (Lewicki, 1998; Segev et al., 2004). Third, relevant features of the spike shapes are extracted, often via principal components or factor analysis or some other dimensionality reduction method. Finally, these features are the input of a clustering algorithm that performs the classification of the spike waveforms. Our emphasis in this paper will be almost entirely on this final clustering step, although we should note that one frequently alternates between the spike detection and clustering phases of these algorithms (using the inferred mean cluster parameters to improve the spike detector), and the nonstationary methods we focus on here could certainly be employed in an identical iterative manner.

Perhaps the simplest and most common probabilistic model underlying the clustering process is the mixture-of-Gaussians (MoG) model, in which each observed voltage waveform snippet V is a sample from a mixture distribution

$$p(V) = \sum_{z=1}^{J} \alpha_z p_z(V),$$

with J denoting the number of distinct units present in the recording, z indexing the different units, and the individual distributions given by the multivariate Gaussian

$$p_z(V) = \mathcal{N}_{\mu_z, C_z}(V), \tag{1}$$

with mean μ_z and covariance matrix C_z. The mixture probabilities α_z satisfy $\sum_z \alpha_z = 1$ and $\alpha_z \geq 0$ for all z. Thus the parameter vector θ summarizing this model includes our information about the underlying mixture components and weights:

$$\theta = \{(\mu_z, C_z, \alpha_z)_{1 \leq z \leq J}\}.$$

If the identities z of the mixture components were observed, in order to estimate the model parameters we would begin by writing down the likelihood $p(V|\theta, z)$ of the observed spike data V given the model parameters θ and the observed identities z, and then we could employ standard likelihood optimization methods to obtain the maximum likelihood (ML) solutions for θ. However, we do not observe z directly, and our likelihood is of the marginalized form

$$p(V|\theta) = \sum_z p(V, z|\theta),$$

with the mixture identities z treated as unobserved ("latent," or "hidden") data. Note that a direct approach towards maximizing this likelihood requires that we marginalize out z, and a direct optimization of this marginal probability may be difficult in general. The expectation-maximization (EM) algorithm (Dempster et al., 1977) was developed as a method for estimating θ without having to compute this integral. For brevity, we will not review the derivation of the EM algorithm for the MoG model here (see, e.g., Bilmes (1997) for a highly legible tutorial); however, we will build on these basic ideas as we develop our nonstationary model below.

2.2. Kalman filter mixture model for spike sorting

Consider now the situation in which the mean voltage waveforms μ_j are non-stationary: for each cluster j, instead of taking μ_j to be constant in time, we model this parameter with a random drift

$$\mu_j^{t+1} = \mu_j^t + \varepsilon_j^t, \quad \varepsilon_j^t \sim \mathcal{N}(0, C_j^\mu). \tag{2}$$

Here ε represents additive Gaussian noise, and the discrete sequence of time steps $t = 1, \ldots, T$ corresponds to the experiment time index. We assume that only one observation Y^t occurs per time step t:

$$Y^t = \begin{cases} V^t & \text{if a spike occurs at } t \\ \varnothing & \text{if no spike occurs at } t, \end{cases} \tag{3}$$

and we retain the Gaussian model (Eq. (1)) for the observations V_t given z^t and μ_j^t:

$$V^t = \mu_{z^t}^t + \eta_{z^t}^t, \quad \eta_{z^t}^t \sim \mathcal{N}(0, C_{z^t}^V). \tag{4}$$

Thus, given the sequence of cluster identities z^t, Eqs. (2) and (4) correspond to a classical Kalman filter (Roweis and Ghahramani, 1999; Durbin and Koopman, 2001), if we identify the vector of time-varying means μ^t as our hidden Markov state variable and the voltage data V^t as our observations. Of course, in practice, z_t are unobserved (these are exactly the variables we are trying to infer), and so we need to marginalize over these latent variables; thus we are left with a mixture-of-Kalman filters (MoK), instead of the simpler mixture-of-fixed-Gaussians model.[2] Finally, in order to exploit the computational efficiency of Kalman filters methods, we make the approximation that the observation covariances $C_{z^t}^V$ do not change in time (see below and Section 4 for a brief description of related approaches that take into account non-stationarities in the observation covariances). See Fig. 1 for an illustration.

To perform inference in this model, we need to adapt the familiar EM method from the MoG model for this MoK case. Lets begin by more explicitly casting this inference problem in terms of the EM framework. The parameters we want to infer are $\theta = \{(\mu_j^t, C_j^\mu, C_j^V, \alpha_j)\}$. We have incorporated a Gaussian prior $p(\mu)$ on the vector of means μ_j^t, from Eq. (2); note that this prior is improper (does not integrate to one), since we have not constrained the initial value of μ_j. In addition, we should emphasize that even though the prior over the μs corresponds to a random walk, more generally it can be interpreted as a term that penalizes big changes in μ_j from one time step to the next. In particular, we show later in Section 3.1 that even when $\mu_j(t)$ changes more systematically than a random walk, the method works well (see Fig. 5). Finally, for now, assume that our prior on the remaining elements of θ is flat, though of course this may be generalized if we have additional prior information that constrains the structure of the model. Now we want to optimize the marginal log-posterior

$$\log p(\theta|Y) = \log p(\theta) + \log p(Y|\theta) = \log p(\mu) + \log \sum_z p(Y, z|\theta).$$

The first step in the derivation of any EM algorithm is to write out the complete log-posterior. In this case, we have:

[2] This model may be seen as a special case of the "switching Kalman filter" model (Wu et al., 2004), which models an observed time series whose dynamics and observation processes change randomly according to a Markov process; here we have $J+1$ processes with fixed dynamics, and our observation is switching between these in an i.i.d. manner (which is a special case of a Markov process). It turns out that inference in the general switching Kalman model is significantly more difficult than in the special mixture-of-Kalmans model discussed here.

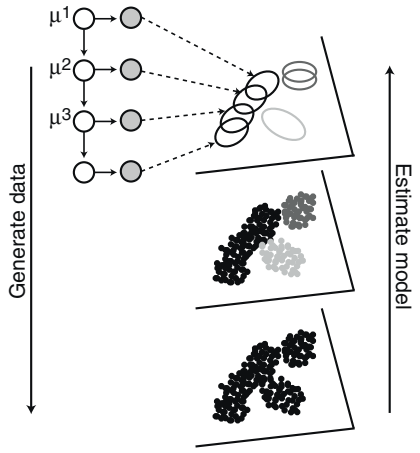

Fig. 1. A schematic illustration of the mixture-of-Kalman filters clustering approach. Time-varying cluster means are generated by Eq. (2), which in turn are combined with noise (Eq. (4), as indicated by the ellipses in the figure) to give rise to the observed data (indicated by the grayscaled dots on the two-dimensional feature plane); from the combined (unlabeled) data we would like to recover the time-varying cluster means and other model parameters (M step) and correctly assign labels to each spike (E step).

$$\log p(Y,z|\theta)+\log p(\mu) = \log p(z|\theta)+\log p(Y|z,\theta)+\log p(\mu)$$
$$= \sum_{t=1}^{T}\log p(z^t|\theta)+\sum_{t=1}^{T}\log p(Y^t|z^t,\theta) + \sum_{j=1}^{J}\sum_{t=2}^{T}\log p(\mu_j^t|\mu_j^{t-1}),$$

where $p(z^t|\theta)=\alpha^t$ corresponds to the mixture weights and the observation density $p(Y^t|z^t,\theta)$ is given by

$$p(Y^t|z^t,\theta)=\begin{cases}\mathcal{N}_0 & 0\\ \vdots & \vdots\\ \mathcal{N}_J & 0\\ 0 & 1\end{cases}; \tag{5}$$

the first column corresponds to the case $Y^t=V^t$ and the second column handles the case that no data were observed at this time point, $Y^t=\varnothing$; we have set $p(V^t|z^t=j,\theta)=\mathcal{N}_j$ (where $\mathcal{N}_j$ denotes the Gaussian observation density as defined in Eqs. (1) and (4)), $p(V^t|z^t=J+1,\theta)=0$, $p(\varnothing|z^t=j,\theta)=0$ and $p(\varnothing|z^t=J+1,\theta)=1$. Note that we have added an additional "background model" Gaussian ($\mathcal{N}_0$) to the mixture. During the EM process only the weight and covariance of this Gaussian will be allowed to change, in order to model variable degrees of background noise; we will fix the mean μ_0, for simplicity.

Thus the complete log-posterior can be written as

$$\log p(Y,z|\theta)+\log p(\mu)$$
$$=\sum_{t=1}^{T}\log\alpha_{z^t}+\sum_{t=1}^{T}\log\mathcal{N}_{\mu_{z^t}^t,C_{z^t}^V}(V^t)+\sum_{j=1}^{J}\sum_{t=2}^{T}\log p(\mu_j^t|\mu_j^{t-1})$$
$$=\sum_{t=1}^{T}\left(\log\alpha_{z^t}-\frac{1}{2}(\log|C_{z^t}^V|+(V^t-\mu_{z^t}^t)^T(C_{z^t}^V)^{-1}(V^t-\mu_{z^t}^t)\right)$$
$$-\frac{1}{2}\sum_{j=1}^{J}\sum_{t=2}^{T}(\mu_j^t-\mu_j^{t-1})^T(C_j^\mu)^{-1}(\mu_j^t-\mu_j^{t-1})+\text{const.}$$

The next step is to write out the expected complete log-posterior; computing this expectation over the conditional distribution of the latent variable z constitutes the E-step of the EM algorithm, while the maximization of the resulting terms (considered as a function of the parameters $\theta=\{\alpha_j,C_j^V,\mu_j^t\}$) corresponds to the M-step. The expected complete log-posterior here is

$$E_{p(z|V,\hat\theta)}\log p(V,z|\theta)+\log p(\mu)$$
$$=E_{p(z|V,\theta)}\left(\sum_{t=1}^{T}\log p(z^t|\theta)+\sum_{t=1}^{T}\log p(V^t|z_t,\theta)\right)$$
$$+\sum_{j=1}^{J}\sum_{t=2}^{T}\log p(\mu_j^t|\mu_j^{t-1})=\sum_{j=0}^{J}\sum_{t=1}^{T}p(z^t=j|V^t,\theta)$$
$$\times\left(\log\alpha_j-\frac{1}{2}(\log|C_j^V|+(V^t-\mu_j^t)^T(C_j^V)^{-1}(V^t-\mu_j^t))\right)$$
$$-\frac{1}{2}\sum_{j=1}^{J}\sum_{t=2}^{T}(\mu_j^t-\mu_j^{t-1})^T(C_j^\mu)^{-1}(\mu_j^t-\mu_j^{t-1})+\text{const.},$$

with

$$p(z^t=j|V^t,\hat\theta)=\frac{e^{\log\hat\alpha_j-(1/2)(\log|\hat C_j^V|+(V^t-\hat\mu_j^t)^T(\hat C_j^V)_j^{-1}(V^t-\hat\mu_j^t))}}{\sum_{j'=0}^{J}e^{\log\hat\alpha_{j'}-(1/2)(\log|\hat C_{j'}^V|+(V^t-\hat\mu_{j'}^t)^T(C_{j'}^V)^{-1}(V^t-\hat\mu_{j'}^t))}}. \tag{6}$$

Thus the E-step turns out to be exactly the same as in the standard MoG model (Bilmes, 1997) and corresponds to a probabilistic assignment of cluster identity to each sample, given the parameters $\hat\theta$ from the previous iteration.

Now, in the M-step we have to optimize $E_{p(z|V,\theta)}\log p(V,z|\theta)$ as a function of θ:

$$\arg\max_\theta\{E_{p(z|V,\theta)}\log p(V,z|\theta)\}$$
$$=\arg\max_\theta\left\{\sum_{j=0}^{J}\sum_{t=1}^{T}p(z_t=j|\theta')[\log\alpha_j-\frac{1}{2}(\log|C_j^V|\right.$$
$$+(V^t-\mu_j^t)^T(C_j^V)^{-1}(V^t-\mu_j^t))]$$
$$\left.-\frac{1}{2}\sum_{j=1}^{J}\sum_{t=2}^{T}(\mu_j^t-\mu_j^{t-1})^T(C_j^\mu)^{-1}(\mu_j^t-\mu_j^{t-1})\right\}. \tag{7}$$

Since this objective function is a sum of simpler functions, we may optimize each piece in turn: we have one optimization involving α, J optimizations involving μ_j (since we have fixed $\mu_0=0$), and finally $J+1$ optimizations involving C_j^V.

The optimization of

$$\sum_{j=0}^{J}\sum_{t=1}^{T}p(z_t=j|\hat\theta)\log\alpha_{z_t},$$

as a function of α_j (under the constraint that α forms a proper probability mass function) leads to the intuitive solution

$$\alpha_j^{\text{new}}=\frac{\sum_{t=1}^{T}p(z_t=j|\hat\theta)}{T}; \tag{8}$$

i.e., the updated mixture probability is just the average fraction of samples assigned to index j under the parameter setting $\hat{\theta}$; again, this is exactly as in the MoG case.

Next we need to optimize

$$-\frac{1}{2}\left(\sum_{t=1}^{T} p(z_t = j|\hat{\theta})(V^t - \mu_j^t)^T (C_j^V)^{-1}(V^t - \mu_j^t) + \sum_{t=2}^{T}(\mu_j^t - \mu_j^{t-1})^T (C_j^\mu)^{-1}(\mu_j^t - \mu_j^{t-1})\right),$$

with respect to μ_j, for each j between 1 and J (again, each optimization can be computed separately). Here, for the first time, we exploit the Kalman nature of our model, and note that each of these optimizations are quadratic in μ_j, with only nearest-neighbor dependence between μ_j^t and μ_j^{t-1}, and may be computed directly via a straightforward forward–backward Kalman recursion. For the forward step, the mean and covariance of the forward distribution $p(\mu^t|Y^{1:t})$ are given by

$$(C_j^{t+1})^F \doteq \mathrm{Cov}(\mu_j^{t+1}|Y^{1:t+1}) = ([(C_j^t)^F + C_j^\mu]^{-1} + p(z_t{=}j|Y^t, \hat{\theta})(C_j^V)^{-1})^{-1}$$

$$(\mu_j^{t+1})^F \doteq E(\mu_j^{t+1}|Y^{1:t+1}) = (C_j^{t+1})^F([(C_j^t)^F + C_j^\mu]^{-1}(\mu_j^t)^F + p(z_t = j|Y^t, \hat{\theta})(C_j^V)^{-1}Y^t), \quad (9)$$

with $p(z^t = j|Y^t = \varnothing) = 0$, $\forall 1 \le j \le J$. These recursions can be derived by the usual complete-the-squares argument, as in the standard Kalman filter setting; see, e.g., Durbin and Koopman (2001) for details.

The backward recursion is exactly the same as in the standard Kalman filter model, since the backward step does not depend on the observed data except through the sufficient forward statistics $E(\mu_t|Y^{1:t})$ and $\mathrm{Cov}(\mu_t|Y^{1:t})$:

$$\begin{aligned}(\mu_j^t)^S &= E(\mu_j^t|Y^{1:T}) = (\mu_j^t)^F + J_j^t[(\mu_j^{t+1})^S - (\mu_j^t)^F]\\ (C_j^t)^S &= \mathrm{Cov}(\mu_j^t|Y^{1:T}) = (C_j^t)^F + J_j^t[(C_j^{t+1})^S - ((C_j^t)^F + C_j^\mu)](J_j^t)^T\\ J_j^t &= (C_j^t)^F[(C_j^t)^F + C_j^\mu]^{-1}\end{aligned} \quad (10)$$

The idea then is to run the forward algorithm to obtain $(\mu_j^t)^F$, $(C_j^t)^F$ for $1 \le t \le T$, then initialize the recursion

$$\begin{aligned}(\mu_j^T)^S &= (\mu_j^T)^F\\ (C_j^T)^S &= (C_j^T)^F,\end{aligned}$$

and propagate backwards for $t = T, T-1, \ldots, 1$. When the backwards recursion concludes at $t = 1$, update $\mu_j^T \leftarrow (\mu_j^T)^S$, and we are done with the update for μ_j.

Finally, optimizing

$$-\frac{1}{2}\sum_{t=1}^{T} p(z_t = j|\hat{\theta})[\log|C_j^V| + (V^t - \mu_{z_t}^t)^T (C_j^V)^{-1}(V^t - \mu_j^t)]$$

with respect to C_j^V, leads to the following update rule for C_j^V:

$$(C_j^V)^{\mathrm{new}} = \frac{\sum_{t=1}^{T} p(z_t = j|\hat{\theta})(V^t - \mu_j^t)^T(V^t - \mu_j^t)}{\sum_{t=1}^{T} p(z_t = j|\hat{\theta})}, \quad (11)$$

again echoing the solution in the MoG case.

Table 1
MoK algorithm.

Initialization: initialize a mixture-of-Gaussians (using e.g. J-means to determine centers $\mu_1, \ldots, \mu_J$ of J components. (Predetermine or cross-validate over J.) Use the output of the mixture-of-Gaussians algorithm to initialize the mixture-of-Kalman filters model.
Repeat
  E Step
    Update memberships z_{ij} (Eq. (6))
  M Step
    For $j = 0{:}J$
      Update α_j (Eq. (8))
    End For
    For $j = 1{:}J$
      Update μ_j^t (forward–backward method)
    End For
    For $j = 0{:}J$
      Update C_j^V (Eq. (11))
    End For
Until convergence

To summarize, the full MoK algorithm (Table 1) consists of the standard MoG E-step to obtain the probabilistic assignments $p(z^t = j|V^t, \hat{\theta})$, followed by the standard MoG M-step to update the mixture parameters α then a slightly modified Kalman forward–backward sweep for maximizing Eq. (7), and finally a standard MoG-like update for the covariance parameters C^V.

We should note before continuing that it is straightforward in principle to also allow the mixture weights to vary with time by introducing a state-space model for these parameters, and applying an approximate EM approach along the lines of Smith and Brown (2003) to iteratively update these parameters. For that matter, we could also introduce a state-space model for the observation covariances C_j^V, although this is somewhat more complicated (since we must maintain the positive definiteness of C_j^V). We do not pursue this direction further here; again, see Bar-Hillel et al. (2006) and Gasthaus et al. (2009) for some related approaches. Finally, we should note that our algorithm does not include an EM update rule for the dynamics noise C_j^μ (recall from Eq. (2) that this parameter determines how much the mean waveform shape μ_j may change from one timestep to the next). Instead, we assume a fixed value for C_j^μ, and show later (see Section 3.2) that varying this parameter over several orders of magnitude has only a weak effect on spike classification for two different real data sets. Even though we do not pursue this direction further here, it is possible to derive an EM update for C^μ directly from the output of the forward–backward algorithm; see Shumway and Stoffer (2006) for details.

2.2.1. Online clustering

In many applications of spike sorting, spike sorting must be carried out in close to real time. In neuronal prosthetic applications, for example, neuronal activity simultaneously recorded from hundreds of electrodes needs to be transformed into a "motor" action on a time scale of hundreds of milliseconds (Donoghue, 2002; Zumsteg et al., 2005). To satisfy such demands, a spike sorting algorithm must be able to estimate the model parameters in an "on-line" fashion.

One major advantage of the MoK is that the forward–backward method introduced above for updating the cluster means μ_j permits a direct on-line implementation: as each new spike is observed at a given time t_o, we may update the required quantities online in the following manner. First, compute the weight $p(z_{t_o} = j|V^{t_o}, \hat{\theta})$; then run the filter backwards from time t_o (incorporating this new piece of information V^{t_o}) to time $t_o - s$, where s is the number of time steps required for the updated values of μ^t to effectively settle back down to the values obtained before the data at time t_o were observed. Finally, update the weights $p(z_t = j|V^t, \hat{\theta})$ for

$t_o - s \le t \le t_o$. Each of these steps requires a fixed amount of computation time (i.e., the computation time does not scale as a function of t_o, or more generally T), and therefore we may in principle repeat these partial EM updates once or a few times per observed spike, to ensure accurate online assimilation of each new data point.

It is also possible to update the estimate for the observation noise C^V in an online manner. Recall that our estimate for the noise variance in the offline version of the algorithm is given by Eq. (11). To apply this equation online, we just need to note that, for time indices $t < s$, the means μ_j^t and the responsibilities $p_{tj} = p(z_t = j|V^t, \hat{\theta})$ do not change, so we just need to keep track of the sufficient statistics

$$A_{sj} = \sum_{i<s} (V^t - \mu_j^t)^T (V^t - \mu_j^t)$$
$$B_{sj} = \sum_{i<s} p_{tj} \quad ,$$

and then update our estimate as

$$(C_j^V)^{\text{new}} = \frac{A_{sj} + \sum_{t \ge s} (V^t - \mu_j^t)^T (V^t - \mu_j^t)}{B_{sj} + \sum_{t \ge s} p_{tj}}. \tag{12}$$

2.3. Including Markovian refractory effects

In the model discussed in Section 2.2, each spike waveform is generated independently of all others. One simple feature that has not been accounted for in our model is the ocurrence of the refractory period, a short period after each action potential during which the cell will not fire again. In this section, we account for this effect by introducing a simple hidden Markov model (HMM) for neural refractoriness. Again, related approaches have been employed previously (Sahani, 1999; Herbst et al., 2008), as we will discuss at more length below.

As before, we model each observed voltage waveform snippet V as a sample from a mixture distribution, where the individual distributions are given by multivariate Gaussians with mean μ_j^t and covariance C_j^V, and z denotes the hidden cluster identity of the observed snippet. In order to account for refractory effects, we now introduce an additional hidden variable q_j^t for each neuron, which represents the *potential* of the cell to generate a spike at time t. For clarity, we first introduce a one-variable HMM for single cell recordings. We then generalize to the multi-neuron case in Section 2.3.2.

2.3.1. Single neuron model

We assume a simple three-state Markov model for clarity (see Fig. 2); as will become clear below, this can be generalized easily. Our model associates an observation Y^t with a spike ($Y^t = V^t$) when the hidden random variable q equals one at that time, $q^t = 1$. We will refer to this state as the *spiking* state, and we assume that the system spends just one timestep in this state; i.e., q_t transitions deterministically away from the spiking state into $q^{t+1} = 2$, the *refractory* state. The neuron is unable to spike in this state (i.e., this state represents an absolute refractoriness); from here, q_t transitions with some rate to the *rest* state. The sum of the mean times the system will spend in the refractory and rest states is precisely equal to the mean inter-spike interval (ISI) for the cell. The state space of q has a ring structure: a spike is generated in state 1, and then q must jump through both remaining states before jumping back into the spiking state.

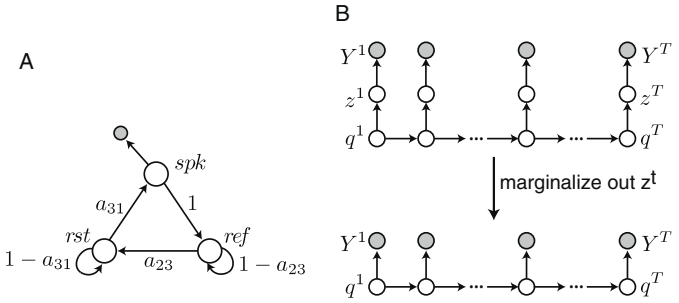

Fig. 2. (A) A simple hidden Markov model for spike trains. The spike-generation process is modeled as a random (hidden) variable with a ring structure. The hidden state labeled *spk* is the state in which the neuron fires. After one time step the hidden variable leaves the spiking state and jumps into the refractory state (*ref*), in which the neuron is unable to produce a spike. From this state the hidden variable transitions with rate a_{23} to the rest (*rst*) state, and after some time to the *spk* state with rate a_{31}, in which the cell fires again. The sum of the mean times the system will spend in the refractory and rest states corresponds to the mean ISI for the cell. (B) Full model of the data generation process. The complete model has the architecture of a slightly extended hidden Markov model: the hidden states q^t are coupled to the observed spike data Y^t only via their connection to the hidden cluster identities z^t. Once the hidden identities z^t are marginalized out, q^t and Y^t together form a hidden Markov model (bottom).

State transitions are determined by the state transition probability distribution $A = \{a_{ik}\}$, where $a_{ik} = p(q^t = k|q^{t-1} = i)$, $1 \le i, k \le 3$:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 - a_{23} & a_{23} \\ a_{31} & 0 & 1 - a_{31} \end{pmatrix}, \tag{13}$$

where as mentioned above the neuron's mean ISI is given by $1/a_{23} + 1/a_{31}$.

We can now write down an expression for the assignment of cluster identity to each sample (E-step):

$$\begin{aligned} p(z_t|Y^{1:T}) &= \sum_{q^t} p(z^t, q^t|Y^{1:T}) = \sum_{q^t} p(q^t|Y^{1:T}) p(z^t|q^t, Y^{1:T}) \\ &= \sum_{q^t} p(q^t|Y^{1:T}) p(z^t|q^t, Y^t) \\ &= \sum_{q^t} \frac{p(q^t|Y^{1:T}) p(z^t|q^t) p(Y^t|z^t)}{p(Y_t|q_t)}. \end{aligned} \tag{14}$$

The third term in Eq. (14) is given by Eq. (5). The second term is determined by the generative model and can be summarized as:

$$p(z^t|q^t) = \begin{pmatrix} 0 & 1 & 0 \\ p & 0 & 1 - p \\ p & 0 & 1 - p \end{pmatrix}, \tag{15}$$

where the first column corresponds to $z = 0$ and the second column to $z = 1$, and the third column to $z = \varnothing$, and where p represents the probability of the background Gaussian model, $\mathcal{N}_0$, emitting an observation. The term in the denominator is just

$$p(Y_t|q_t) = \sum_{z^t} p(Y^t|z^t) p(z^t|q^t). \tag{16}$$

Thus to compute the assignments $p(z_t|Y^{1:T})$ the only nontrivial part is to compute the conditional state probabilities $p(q^t|Y^{1:T})$. To handle these, note that once we marginalize out z_t via Eq. (16), q_t and Y_t together form a hidden Markov model, as summarized in Fig. 2. Thus we may employ the standard HMM forward–backward algorithm (as summarized, e.g., in Rabiner, 1989) to obtain $p(q^t|Y^{1:T})$. Once the assignments $p(z_t|Y^{1:T})$ are in hand, the remainder of the parameter updates (M-step) are straightforward: the Kalman terms remain the same as discussed

above, while the M-step for the transition probabilities $p(q^t|q^{t-1})$ (specifically a_{23} and a_{31}) is standard (Rabiner, 1989). We will refer to this combined mixture-of-Kalmans HMM model by the abbreviation MoKHMM below.

2.3.2. Full model – many neurons case

To sort the spikes from many simultaneously recorded neurons, we may in principle proceed as in the simpler single-neuron case. We introduce a hidden variable q_j for each neuron; assuming J neurons, the states of the hidden variables can be summarized by the vector $q^t = (q_1^t \dots q_J^t)$, where q_j^t is a number between 1 and 3, representing the state of the j-th hidden variable associated with data at time t. The EM iteration for the full model is completely analogous to the single neuron case if we replace the state transition probability distribution and the observation symbol probability distribution by the corresponding elements of the joint HMM for J cells. The state transition probability distribution $A = \{a_{ik}\}$ for the joint process may be written most simply in terms of the Kronecker product of the individual transition matrices $A^j = \{a_{ik}^j\}$:

$$A = A^1 \otimes A^2 \otimes \dots \otimes A^J,$$

due to the independence of each q^t. Note that this joint transition matrix A is fairly sparse, which enhances the efficiency of the forward–backward procedure, in which the computational time is dominated by repeated matrix–vector multiplications involving the transition matrix.

The joint observation probability distribution may be defined similarly. The conditional mixture identities $p(z^t|q^t)$ are constructed in the natural way: neurons for which the corresponding elements of q_t are in the spiking state are assigned as spiking in this time bin with probability one, and if no neurons are in the spiking state then we assign probability p to the background model and the remainder, $1-p$, to the null observation, just as in Eq. (15). For the observation density $p(Y^t|z^t)$, if one or fewer neurons are in the spiking state, then we revert to the single-neuron observation and plug in the Gaussian formulas discussed above. Conversely, if more than one element of z^t is labeled "on," then we have a simultaneous spike in this time bin, and we can modify the mean and covariance of our observation Gaussian accordingly, e.g., by summing the means of the spiking neurons. (The simulations and data considered in the Results section below deal largely with the case where the expected number of spike coincidences is negligible, and so we did not pursue this aspect of the model in great depth.)

This completes the specification of our algorithm. Of course the attentive reader will have noticed a problem: the dimensionality of the effective state variable q here scales exponentially with J.[3] Thus this exact approach is restricted to the setting of fairly small J in high-firing regimes. (To be clear, this concern only applies to the Markov refractory model described in this section; the computational complexity of the basic MoK discussed in Section 2.2 is not subject to any such exponential dependence on J.) Approximations to the exact E-step such as Gibbs sampling or variational mean field algorithms have been developed elsewhere (Jordan, 1999; Herbst et al., 2008); further exporation of these methods here remains a topic for further research.

[3] More precisely, the state dimensionality scales exponentially with the number of cells for which we implement a Markovian refractory model; for many low-firing cells, it will be simpler to just revert back to the original MoK model, to avoid such an explosion in computational complexity.

3. Results

3.1. Clustering of synthetic data

We began by testing the algorithm (a MATLAB version of the code, as well as several example synthetic data sets is publicly available at http://www.columbia.edu/amc2257/ana/Code.html) on simulated data; this allowed us to quantify the performance in cases where ground truth information was available over a variety of noise levels and degrees of nonstationarity. As discussed in, e.g., Quian Quiroga (2007), a complete spike sorting system includes modules for spike detection, feature representation, and clustering. The focus of our work is in the clustering stage, and we will use pre-existing methods for spike detection and representation (see Section 3.2 for details). We therefore restrict our attention here to simulations in which we model dynamic mixtures in a simple two-dimensional setting; we have in mind the common approach of projecting putative spike waveforms onto a plane spanned by the first two principal components of the observed voltage segment data (the choice of a two-dimensional representation here is merely for visualization purposes. Both the MoK and MoKHMM can handle higher-dimensional representations of the data; see online for an example). To test the complete model (MoKHMM), we further restricted the analysis to mixture models with just $J = 2$ components. However, if the Markov refractory model is ignored (as would be appropriate in low firing rate settings, where refractory violations are less of a concern), the number of component units can be easily generalized to an arbitrary J (see Section 2.3.2 for further details).

We test our model on two different scenarios. In Section 3.1.1 we assess the performance of the algorithm on synthetic data that was generated according to the model described in Section 2.3.2. In Section 3.1.2 we relax some of the assumptions of the MoK model (namely the Gaussian model for the observations Y^t in Eq. (1) and the Gaussian random drift of μ_j in Eq. (2)) and test the performance of our model on a more challenging data set that deviates from these assumptions.

3.1.1. Synthetic data set 1: dynamic mixture of Gaussians

This data set mimics two nonstationary clusters corresponding to two different cells. Initially, the clusters are well-separated but after some time the movements in the cluster centers induce overlaps. The data were generated according to the model described in Section 2.3.2. Fig. 3A (right panel) shows the resulting synthetic data set. We compared the performance of our MoK with a Markov model for neural refractoriness (MoKHMM) against a standard mixture-of-Gaussians model (MoG). Both methods were applied to the same spike data, and were supplied with the correct number of clusters J.

The clustering solutions for the MoG and MoKHMM are shown in Fig. 3A (left and center panels, respectively). As can be seen clearly from the figure, in this scenario it is very difficult to estimate the correct clustering based on a "stationary" view of the data (in which temporal information is discarded). The MoG, which assumes stationarity of the data, fails to provide an accurate estimate of the true underlying labels. The MoK approach, on the other hand, allows us to successfully track the clusters' centers over time (see Fig. 3B), and produces a solution that is much closer to the underlying ground truth (fraction of spikes correctly classified was 0.9 for the MoKHMM, versus 0.72 for the MoG).

With this data set we intended to simulate the case of high firing rate neurons, where refractory period violations (i.e., the occurrence of two spike observations in a time window of length 1 ms) are often observed in the data. There are 41 refractory period violations in this data set, out of which our model is able to detect 40. Fig. 4 shows the true and estimated hidden state sequence for

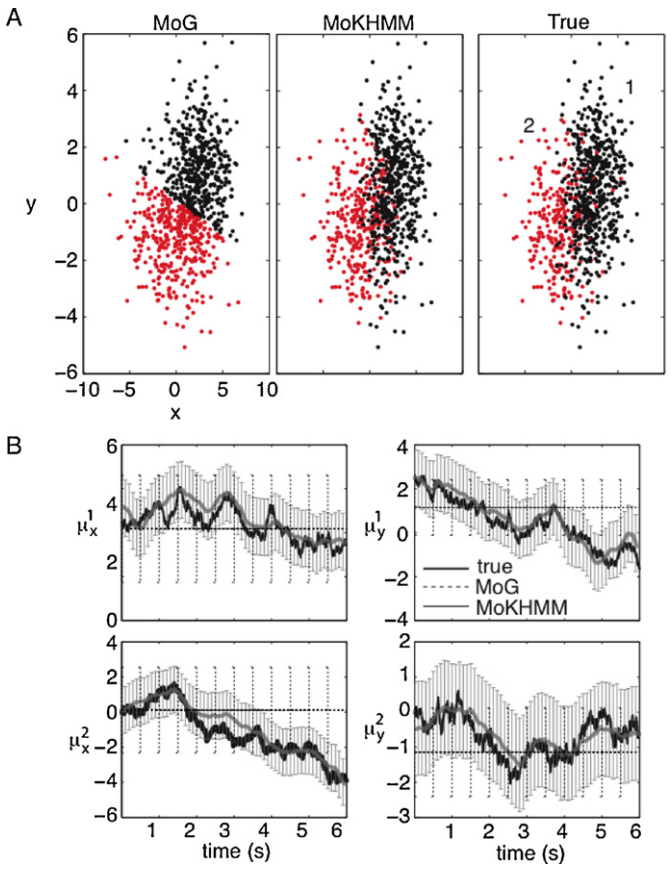

Fig. 3. Clustering results obtained for an example synthetic dataset. Simulated spikes in a two-dimensional feature space were sampled from a mixture of two bivariate Gaussian distributions. In addition, both cluster centers (each associated with a different neuron) move constantly according to Eq. (2), and for both clusters the spike-generation process follows the three-state model described in Section 2.3.1 (see Fig. 2A). (A) Clustering solution for a Gaussian mixture model (left), clustering solution for the Kalman-filter mixture model with an HMM for neural refractoriness (center), and true underlying mixtures (right). (B) Estimated and true underlying 2D movement of the centers of the clusters (μ_1 and μ_2) as a function of time. Error bars represent posterior s.d. for μ (computed by taking the square root of the diagonal elements of the forward–backward covariance $(C_j^t)^S$ from Eq. (10)). The position of the center of the clusters as estimated by the MoG has been plotted for comparison (dashed lines), with the errorbars in this case indicating the s.d. of the observation noise C_j^V. We found that the MoKHMM significantly outperforms an MoG and, is able to accurately track the clusters over time when applied to this example synthetic dynamic mixture-of-Gaussians with substantial cluster overlap.

a subset of the data. While the tracking of the complete sequence is not perfect (we find that correctly estimating the hidden states becomes more difficult when the clusters overlap stongly) the model is able to detect most of the refractory period violations, and taking these refractory effects into account improves the performance of our model. When we try to sort the same data with a model that does not include the Markov model for refractoriness (MoK) we find that the fraction of spikes correctly classified drops to 0.86 (as opposed to 0.9 for the full model). These results were replicated consistently over many similar numerical experiments (data not shown).

3.1.2. *Synthetic data set 2: robustness with respect to relaxation of the model assumptions*

We next tested the robustness of our algorithm by relaxing some of the underlying assumptions of the model. How well does the model sort data that was not generated according to a Gaussian model? How accurately is the model able to track cluster movements that do not correspond to a simple Kalman filter?

To answer these questions we generated synthetic data for which spikes (again, as represented in a two-dimensional plane for simplicity of visualization) were sampled from a mixture of two bivariate *t* distributions (with four degrees of freedom) as opposed to a mixture of Gaussians (Shoham et al., 2003). In this data set, one of the clusters (associated with a neuron) moves with a constant, deterministic velocity towards the other cluster (which is stationary and associated with noise). Note that this scenario is different from the one described in Section 3.1.1, where both clusters in the mixture were associated with neurons and both had Gaussian temporal dynamics. In addition, the center of cluster 1 (neuron) experiences a large discontinuous jump about halfway through the simulated experiment (Fig. 5A, right panel, black line). The spike-generation process for cluster 1 is given by the model described in Section 2.3.1 (see also Fig. 2A). For cluster 0 (noise), spikes were generated at random with a fixed probability $p = 0.15$.

Fig. 5A shows the clustering solution for a MoG and the MoKHMM, together with the posterior standard deviation for the estimates of the center of each cluster at different times through the length of the simulation. As with the previous data set, our method outperforms a simple MoG (fraction of spikes correctly classified was 0.96 for the MoKHMM and 0.88 for the MoG). Even though the underlying 2D movement of the center of cluster 1 ($\mu 1$) deviates significantly from a random walk, our model is still capable of accurately tracking its trajectory (Fig. 5B). Note that towards the end of the simulation, when both clusters overlap, it becomes harder to accurately estimate the position of cluster 1 (Fig. 5B, left panel). We observe a similar effect for the estimation of the hidden state q (not shown).

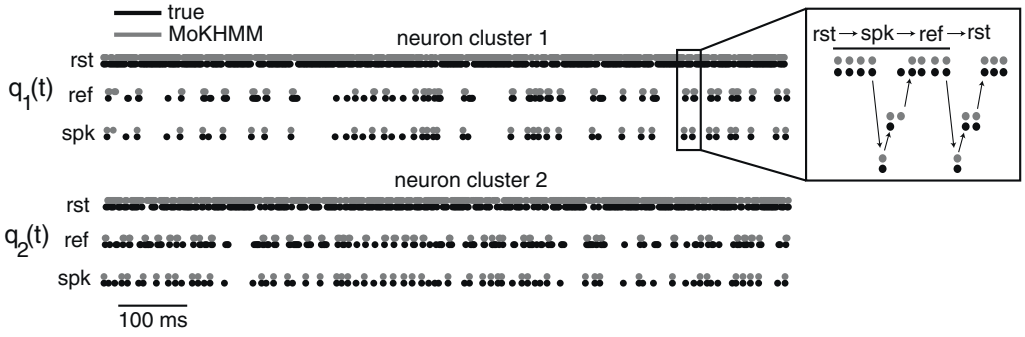

Fig. 4. True (black) and estimated (gray) hidden states for a subset of the synthetic dataset of Fig. 3. The inset shows an expanded version of a small portion of the sequence of hidden states for cluster 1 to facilitate visualization of the transitions. While the tracking of the complete state sequence is not perfect, the MoKHMM is able to detect most of the refractory period violations in the example synthetic data set of Fig. 3.

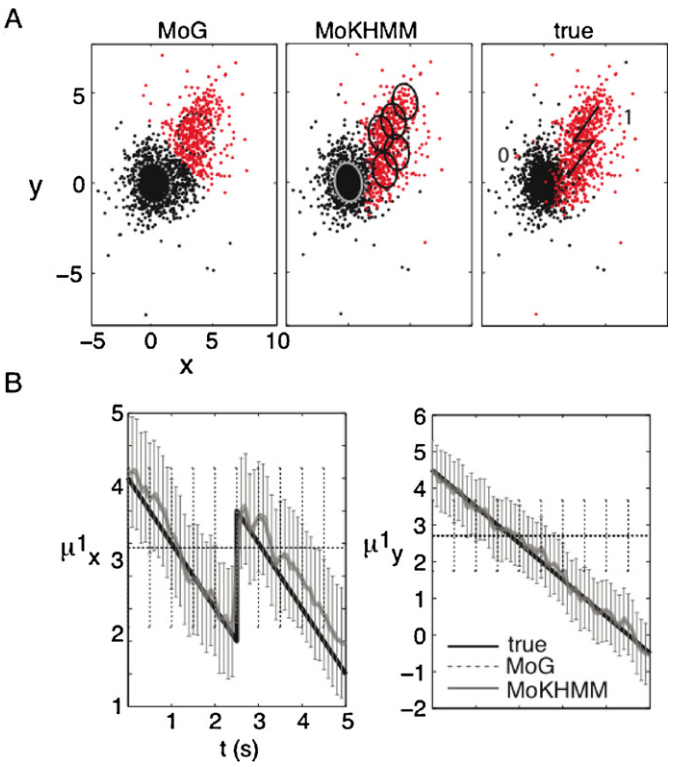

Fig. 5. Robustness of the MoK to relaxation of the model assumptions. Spikes in a two-dimensional feature space were sampled from a mixture of two bivariate t distributions, as opposed to a mixture of Gaussians. In addition, cluster 1 (associated with a neuron, red) moves deterministically and with constant velocity towards cluster 0 (associated with noise, black), but experiencing a discontinuous jump about halfway through the experiment (panel A, right panel, black line). The spike-generation process for cluster 1 is given by the three-state model described in Section 2.3.1 (see Fig. 2A). (A) Left: MoG clustering solution. Dashed lines represent noise covariance ellipses computed from C_j^V for each cluster. Center: clustering solution for the MoKHMM. Full lines represent estimated observation noise covariance ellipses centered at the inferred cluster means at a few representative times through the length of the simulation. Covariance ellipses for the observation noise C_j^V in the MoG model are replotted for comparison. Right: true underlying mixtures and trajectory of cluster 1 (red). (B) Estimated and true underlying 2D movement of the center of cluster 1 (μ_1) as a function of time. Error bars represent the posterior s.d. of μ, computed as in Fig. 3. The position of the center of cluster 1 as estimated by the MoG has been plotted for comparison (dashed lines ± estimated observation noise s.d.). Even in a case in which the data set deviates significantly from the model assumptions (see Section 3.1.2 for details), we found that the Mok has a higher clustering performance than the MoG and is able to accurately track the position of the centers as a function of time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

3.2. Clustering of real data

Finally, we tested our method on two different real data sets. The first data set, for which the "ground-truth" labels are partially known, consists of simultaneous intracellular and extracellular (tetrode) recordings of cells in the hippocampus of anesthetized rats (see Harris et al., 2000) for a detailed description of this data set). The second data set, for which the "ground-truth" labels are unknown, corresponds to extracellular (single electrode) recordings of cells in the midbrain of awake songbirds. In each case, the MoK appears to qualitatively outperform the MoG in tracking the non-stationarity visible in this data.

3.2.1. Data set with known "ground truth" labels

We used a 4-min subset of the data set here (in particular, we used data set d533101 at http://crcns.org/data-sets/hc/hc-1/about), in which recordings from an extracellular tetrode and an intracellular (IC) electrode were made simultaneously. Action potentials detected on the extracellular channels may include the action potentials generated by the intracellularly recorded cell (which are recorded with sufficiently high SNR to be treated as ground truth here), but typically include spiking activity from other cells as well. The data were recorded at 10 kHz and bandpass filtered (300 Hz–3 kHz), and spikes on the intracellular channel were detected as the local maxima near which the first derivative of the voltage exceeded a threshold. Putative spikes on the extracellular channels were determined as the local maxima near voltage excursions exceeding 6 median absolute deviations in magnitude. We extracted 40-dimensional spike waveforms from around each spike (19 samples before and 20 samples after the peak). The positions of the spike waveforms were aligned by upsampling, shifting and then down-sampling the waveforms. The extracellular spikes corresponding to action potentials from the identified neuron were determined as the spikes occurring within 0.1 ms of the IC spike. PCA dimensionality reduction was performed on the spike waveforms from one of the four tetrode channels. The first two principal components were used as the input to our spike sorting algorithm. The dataset consists of 2491 putative spikes, 786 of which were also detected on the IC channel.

As shown in Fig. 6A, there is a clearly visible change in waveform shape of the identified neuron over time. This can also be seen from the non-Gaussian shape of the clusters on the right panel of Fig. 6B (in particular, see cluster 1, red), which shows the underlying labeling of the identified cell (red) and presumably additional cells and background noise (black), for one channel of the tetrode. The inferred labels are illustrated on the left panel of Fig. 6B. For comparison, the same data set was also sorted using the MoG (which does not make use of any information about the occurrence times of the spikes). We found that our algorithm outperforms the MoG on this data set (fraction of correctly classified spikes ± binomial proportion confidence interval for MoK was 0.91 ± 0.006 as opposed to 0.89 ± 0.006 for MoG); this difference is modest here, largely because the clusters remain fairly well separated even if temporal information is discarded. Note that here we use the MoK model instead of the full MoKHM model to sort this data, since this data set does not contain any refractory period violations and sorting the data with the full model leads to the same classification performance. Turning off the HMM for neural refractoriness here has an additional advantage in terms of computational efficiency: instead of taking the time index t to be the experimental time, we may take t to be the spike number. This straightforward modification makes the algorithm significantly faster (time for a single EM iteration in the MoK setting is ∼1.4 s as opposed to ∼90 s for the full MoKHM model).

The positions of the clusters' centers (μ_1 and μ_2) as a function of time are shown in Fig. 6C. In agreement with Fig. 6A, we found that the center of cluster 1 (identified cell) drifts considerably with time, whereas the center of cluster 2 is mostly static. Our results did not depend strongly on the value of C_j^μ used; recall that this quantity controls the variability in the drift of the cluster means μ_j. We experimented with settings of C_j^μ over several orders of magnitude; classification performance remained at 90% over this range.

In an attempt to improve the overall performance of both algorithms, we tried sorting the data by combining the signals from the four tetrode channels. We appended the 4 voltage snippets extracted from each of the channels into a single vector, and then performed PCA dimensionality reduction on the combined data. This procedure led to a slight (∼2%) increase in the performance of the algorithms. Adding more dimensions to the spike representation (e.g., three PC instead of two) had an even smaller effect. Increasing the number of clusters in the mixture (e.g., $J = 3$), did not

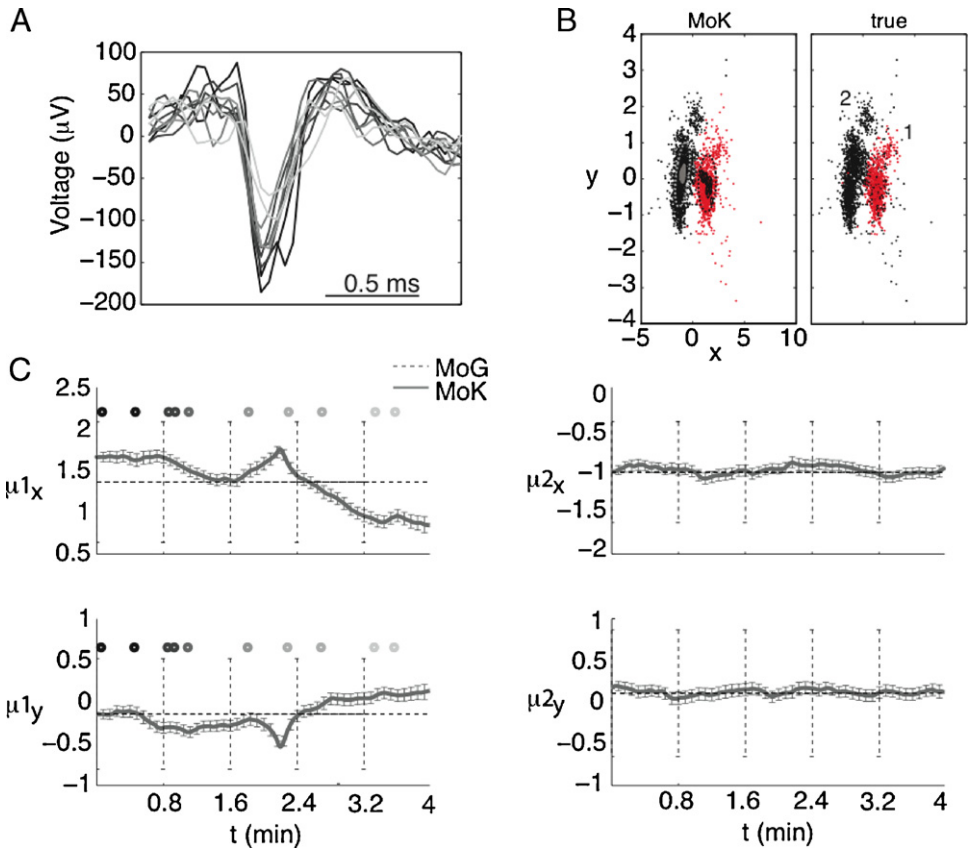

Fig. 6. Clustering results and mean-tracking for data for which the ground truth labeling of one neuron is known (see Section 3.2.1 for details). (A) Subset of the extracted voltage snippets from a single extracellular electrode for the identified cell (cluster 1 in panel B). Different levels of gray represent different occurrence times (dark gray corresponds to the beginning of the recording session and light gray to the end). The amplitude of the detected waveforms changes visibly with time (see also panel C). (B) Left: clustering solution for the MoK. Full lines represent the estimates of the center of each cluster at diferent times through the length of the experiment. The corresponding observation noise covariance ellipses for C_j^y in the MoG are plotted for comparison. Right: true underlying labels of the recoded data, as determined by intracellular recording. (C) Estimated cluster centers (μ_1 and μ_2) as a function of time, ±posterior s.d., as in Figs. 3 and 5. Estimated MoG means, ±estimated observation noise s.d., plotted for comparison (dashed). Gray circles represent the occurrence time of the waveforms plotted in panel A. The use of a MoK on this data set allows us to successfully track nonstationarities in the clusters' positions.

increase the number of spikes of the labeled cell that were correctly classified.

3.2.2. Data set with unknown "ground truth" labels

We used a 20-min data set, in which recordings from a single extracellular electrode were made from the auditory midbrain of an awake, restrained zebra finch, during playback of different types of auditory stimuli. The data were recorded at 24 kHz, amplified (1000×), and filtered (300–5000 Hz; A-M Systems). Spike times were detected online using a spike threshold discriminator and spike waveforms were saved for off-line sorting. Waveforms consisted of 293-dimentional vectors, and we used the first two principal components of each vector as the input to our algorithm. The data set consists of 14713 putative action potentials.

Fig. 7A shows the first two PCs of the extracted voltage snippets color-coded according to occurrence order. While one of the clusters (cluster 1) stays mostly static throughout the recording session, the other one (cluster 2) drifts considerably over time. The inferred labels using a mixture-of-Gaussians are shown in Fig. 7B, and those assigned by our algorithm are illustrated in Fig. 7C and D, for two different values of the dynamics noise parameter (C_j^μ), respectively. Note that even though this new data set contains substantial drift, the two cells still remain fairly well separated, which leads to similar clustering assignments for the MoK and MoG. Since the true labels of the data are not known, we compare the MoG and the MoK in terms of their associated observation error ellipses, which are a measure of the uncertainty in the observations. Smaller error ellipses mean decreased uncertainty and in this sense we found that the MoK outperforms the MoG.

As before, we found that the clustering assignments for the MoK did not depend strongly on the value of C_j^μ used (c.f. Fig. 7C and D), as expected. Varying the dynamics noise parameter has a more visible effect on the estimation of the clusters' position over time. This can be seen clearly from Fig. 8: smaller values of C_j^μ lead to decreased variability when tracking the position of the clusters' centers as a function of time (c.f. panels A and B of Fig. 8). In agreement with Fig. 7A, we found that the center of cluster 1 stays relatively stationary, while the center of cluster 2 drifts considerably over time. The MoK seems to do a good job overall of tracking the nonstationarity evident in Fig. 7A.

4. Discussion

We have introduced a simple Bayesian method for tracking nonstationary spike shapes in extracellular recordings. The basic model we employ can be seen as an extension of the usual mixture-of-Gaussians model for spike sorting to the case that the cluster means are allowed to vary with time. The Gaussian nature of the model leads to efficient computation based on standard Kalman filter methods; in particular, the required forward and backward recursions can be adapted for on-line implementation, as discussed in Section 2.2.1. Despite the simplicity and constrained nature of the

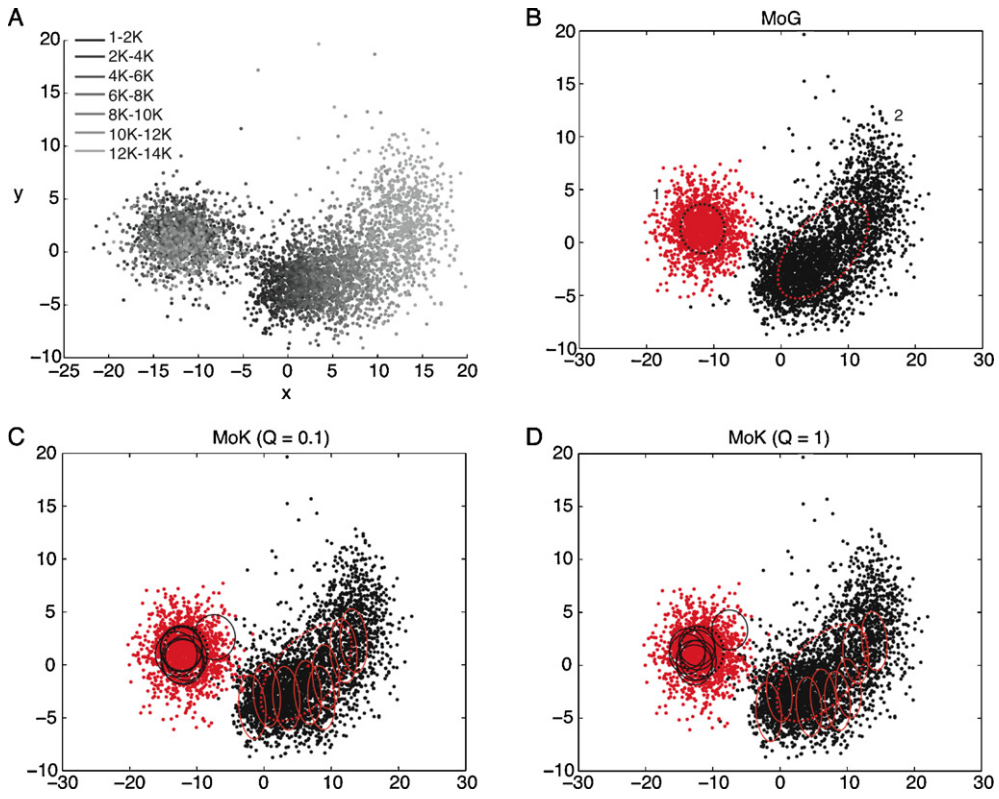

Fig. 7. Clustering results for a data set for which the ground truth labeling is not known(see Section 3.2.2 for details). (A) Extracted voltage snippets in a two-dimensional feature space color coded according to occurrence order (dark gray corresponds to the beginning of the recording session and light gray to the end). (B) Clustering assignments for a MoG model. Dashed lines represent the covariance ellipses for the observation noise C_j^V. (C) Clustering assignments for the MoK, for $C_j^\mu = 1$. Noise covariance ellipses for the MoK (full lines) and MoG (dashed lines) as in Fig. 5. (D) Clustering assignments for the MoK, for $C_j^\mu = 0.1$. Noise covariance ellipses for the MoK and MoG as in panel C. Even though the labeling of the data produced by the two methods is similar, the use of a MoK rather than a MoG results in decreased uncertainty in the observations as measured by the associated observation noise covariance ellipses.

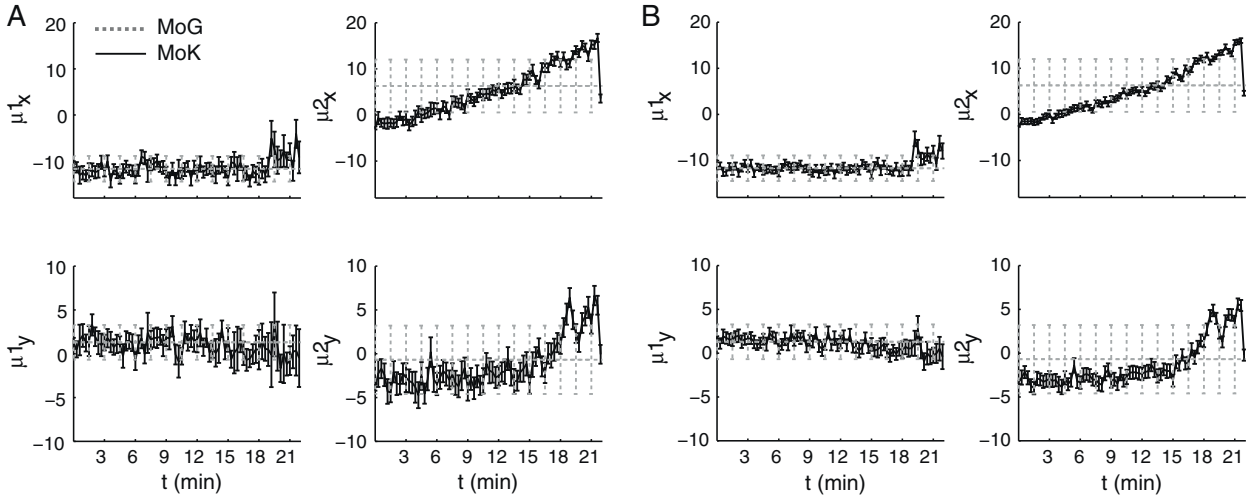

Fig. 8. Estimated cluster centers (μ_1 and μ_2) for the data set in Fig. 7 as a function of time, ±posterior s.d., as in Figs. 3, 5 and 6. Estimated MoG means, ±estimated observation noise s.d., plotted for comparison (dashed). (A) $C_j^\mu = 1$, (B) $C_j^\mu = 0.1$. A smaller value of C_j^μ leads to decreased variability in the estimate of μ_j as a function of time (compare panels A and B). In both cases, while cluster 1 stays mostly stationary over time, cluster 2 drifts considerably (c.f. Fig. 7).

model, we have found that these methods are effective and fairly robust when applied to simulated and real data. In addition, it is possible to incorporate refractory effects into the model via closely related hidden Markov model techniques.

As emphasized in Section 1, our work is far from the first to address these issues. For example, Monte Carlo-based methods for incorporating non-stationarity have been proposed which are more powerful and flexible but more computationally expensive (Pouzat et al., 2004; Gasthaus et al., 2009). In addition, more powerful nonparametric Bayesian methods can be employed to perform automatic model selection and averaging (i.e., a proper quantification of our uncertainty about the number of cells present), a topic

we do not address at all here[4]; see, e.g., Wood and Black (2008) and Gasthaus et al. (2009) for further details. Bar-Hillel et al. (2006) took a similar approach to nonstationarity tracking as ours; in fact their model for nonstationarity was more general, in that Bar-Hillel et al. (2006) allow both the covariance and the mean of the distribution of spike shapes to change with time. We make the approximation that only the cluster mean varies significantly in time, in order to exploit the more computationally efficient Kalman filter methods emphasized here. Finally, Wolf et al. (2009) recently employed an online Gaussian-based iterative tracking scheme which is similar in spirit to the forward recursion of the Kalman filter; these authors went further, implementing online hardware control of the electrode position to stabilize the recording quality. It would be interesting to explore whether the online expectation-maximization-Kalman approach we have presented here for tracking multiple clusters could lead to a more accurate and robust method for electrode stabilization.

Related approaches to incorporating refractory effects in spike sorting algorithms have also been pursued. Sahani (1999) discusses a number of these approaches. At one end of the spectrum we can simply discard spike pairs which cause a "refractory violation"; this is done semi-manually in many experimental labs, though more principled expectation-maximization-based approaches are available (Sahani, 1999). On the other hand a number of works have pursued more intricate hidden Markov models, in which the observed voltage depends on the precise submillisecond spiking state of the neuron, or on the cell's longer-term spike history (Sahani, 1999; Fee et al., 1996; Pouzat et al., 2004; Herbst et al., 2008). Our work represents a computationally efficient compromise position between these two extremes. As emphasized in Section 2.3.2, an important direction for future work will be to develop more efficient and scalable methods in the setting of large numbers of isolable units.

Acknowledgments

We thank A. Vyas for providing the songbird data set, and J. Pillow, J. Schulman, J. Shlens, S. Shoham, E.Simoncelli, J. Vogelstein, and F. Wood for helpful discussions. LP is supported by an NSF CAREER and a McKnight Scholar award.

References

Bar-Hillel A, Spiro A, Stark E. Spike sorting: Bayesian clustering of non-stationary data. J Neurosci Methods 2006;157:303–16.

Bilmes J. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report TR-97-021. ICSI; 1997.

Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 1977;39:1–38.

Donoghue J. Connecting cortex to machines: recent advances in brain interfaces. Nat Neurosci 2002;5:1085–8.

Durbin J, Koopman S. Time series analysis by state space methods. Oxford University Press; 2001.

Fee M, Mitra P, Kleinfeld D. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability. J Neurosci Methods 1996;69:175–88.

Gasthaus J, Wood F, Gorur D, Teh Y-W. Dependent Dirichlet process spike sorting. NIPS 2009:497–504.

Harris K, Henze D, Csicsvari J, Hirase H, Buzsaki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. J Neurophys 2000;84:401–14.

Herbst JA, Gammeter S, Ferrero D, Hahnloser RH. Spike sorting with hidden Markov models. J Neurosci Methods 2008;174(1):126–34.

Jordan MI, editor. Learning in graphical models. Cambridge, MA, USA: MIT Press; 1999.

Lewicki M. A review of methods for spike sorting: the detection and classification of neural action potentials. Network: Comput Neural Syst 1998;9:R53–78.

Pouzat C, Delescluse M, Viot P, Diebolt J. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a Markov chain Monte Carlo approach. J Neurophys 2004;91:2910–28.

Quian Quiroga, R. Spike sorting. Scholarpedia, http://www.scholarpedia.org/article/Spike_sorting; 2007.

Quirk MC, Wilson MA. Interaction between spike waveform classification and temporal sequence detection. J Neurosci Methods 1999;94:41–52.

Rabiner L. A tutorial on Hidden Markov Models and selected applications in speech recognition. Proc IEEE 1989;77:257–86.

Roweis S, Ghahramani Z. A unifying review of linear Gaussian models. Neural Comput 1999;11:305–45.

Sahani M. Latent variable models for neural data analysis. PhD thesis. California Institute of Technology; 1999.

Segev R, Goodhouse J, Puchalla J, Berry M. Recording spikes from a large fraction of the ganglion cells in a retinal patch. Nat Neurosci 2004;7:1154–61.

Shoham S, Fellows M, Normann R. Robust, automatic spike sorting using mixtures of multivariate *t*-distributions. J Neurosci Methods 2003;127:111–22.

Shumway R, Stoffer D. Time series analysis and its applications. Springer; 2006.

Smith A, Brown E. Estimating a state-space model from point process observations. Neural Comput 2003;15:965–91.

Wolf MT, Cham JG, Branchaud EA, Mulliken GH, Burdick JW, Andersen RA. A robotic neural interface for autonomous positioning of extracellular recording electrodes. Int J Robotics Res 2009;28:1240–56.

Wood F, Black M. A nonparametric Bayesian alternative to spike sorting. J Neurosci Methods 2008;173:1–12.

Wu W, Black MJ, Mumford D, Gao Y, Bienenstock E, Donoghue J. Modeling and decoding motor cortical activity using a switching Kalman filter. IEEE Trans Biomed Eng 2004;51:933–42.

Zumsteg Z, Kemere C, O'Driscoll S, Santhanam G, Ahmed R, Shenoy K, et al. Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems. IEEE Trans Neural Syst Rehabil Eng 2005;13:272–9.

[4] Standard methods, such as cross-validation, can be readily applied to the Kalman setting to determine the number of clusters *J*. The development of more powerful model selection methods remains a subject of future work.