Journal of Neuroscience Methods 174 (2008) 126–134

Contents lists available at ScienceDirect

Journal of Neuroscience Methods

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

Spike sorting with hidden Markov models

Joshua A. Herbst, Stephan Gammeter, David Ferrero, Richard H.R. Hahnloser*

Institute of Neuroinformatics UZH/ETH Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland

ARTICLE INFO

Article history:
Received 3 January 2007
Received in revised form 19 May 2008
Accepted 10 June 2008

Keywords:
Tetrode
Impedance
Songbird
Spike train
Code
Population
Sleep

ABSTRACT

The ability to detect and sort overlapping spike waveforms in extracellular recordings is key to studies of neural coding at high spatial and temporal resolution. Most spike-sorting algorithms are based on initial spike detection (e.g. by a voltage threshold) and subsequent waveform classification. Much effort has been devoted to the clustering step, despite the fact that conservative spike detection is notoriously difficult in low signal-to-noise conditions and often entails many spike misses.

Hidden Markov models (HMMs) can serve as generative models for continuous extracellular data records. These models naturally combine the spike detection and classification steps into a single computational procedure. They unify the advantages of independent component analysis (ICA) and overlap-search algorithms because they blindly perform source separation even in cases where several neurons are recorded on a single electrode. We apply HMMs to artificially generated data and to extracellular signals recorded with glass electrodes. We show that in comparison with state-of-art spike-sorting algorithms, HMM-based spike sorting exhibits a comparable number of false positive spike classifications but many fewer spike misses.

1. Introduction

Owing to its high spatio-temporal resolution, extracellular recording of neural activity is the electrophysiological technique of choice in intact animals. Typically, extracellular electrodes pick up electrical currents associated with action potentials from several cells, giving rise to the spike-sorting problem, i.e., the problem of detecting and classifying spike waveforms. Among the noise sources that hamper spike sorting is electrical noise (for example radiative pickup and ground-loop noise) (Bankman et al., 1993), electrode drift (Bar-Hillel et al., 2006), intrinsic spike waveform variability (for example due to movement of the animal or spike bursts), and background signals from other nearby neurons. Also, when two neurons have similar morphologies and distances to the electrode, their spike waveforms will look very similar, and so spike sorting is likely to be affected by confusions.

Many spike-sorting techniques have been introduced (Lewicki, 1998; Brown et al., 2004). Sorting results can vary significantly depending on the technique used and the human operator (Harris et al., 2000; Wood et al., 2004). In most existing algorithms, candidate spike waveforms are first detected by a simple thresholding operation and are then classified using sophisticated data clustering or classification techniques (Fee et al., 1996; Rinberg et al., 1999; Shoham et al., 2003; Delescluse and Pouzat, 2006). The problem with spike detection is that to simplify waveform classification, high thresholds are chosen, leading to many spike misses. There have been efforts to improve the spike-detection step for low signal-to-noise recordings (Kim and Kim, 2003a; Nenadic and Burdick, 2005), yet essentially no algorithm can do without some form of spike detection.

In brain areas such as hippocampus where average firing rates of cells are low, the spike trains of more than three neurons can be extracted from single-electrode recordings (Gray et al., 1995). When recording with tetrodes, this number increases up to 5–20 neurons (Wilson and McNaughton, 1993). However, in brain areas with high cell density and firing rates, spike overlaps can be prohibitive for most (classical) spike-sorting techniques. With high average firing rates and highly synchronized activity, spike waveforms from different neurons frequently overlap, leading to waveform superpositions that look quite different from their constituents, illustrated in Fig. 1. Very few algorithms have been designed to handle spike overlaps, mainly because overlaps are assumed to be rare events (which may be justified in many cases), or because they are difficult to deal with. Classification of spike overlaps cannot be performed by simple clustering algorithms, because overlapping spikes should not be considered as prototypes of a new class, but as joint occurrences of two existing prototypes.

Partially overlapping spikes can be sorted using clustering techniques based on highly localized features such as wavelet coefficients (Hulata et al., 2002). Most overlap-permissive algorithms suffer from the intractability of exhaustive searching. This intractability is overcome by limiting the number of spikes used to

* Corresponding author. Tel.: +41 44 635 3060; fax: +41 44 635 3053.
E-mail address: rich@ini.phys.ethz.ch (R.H.R. Hahnloser).

0165-0270/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2008.06.011

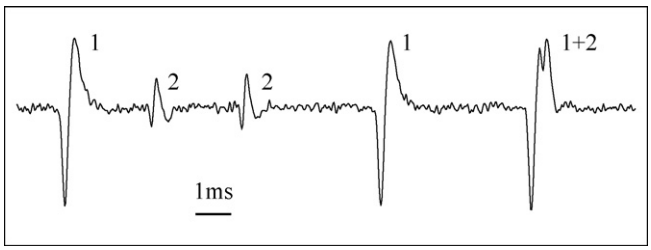

Fig. 1. The spike overlap problem. The spike waveforms of two neurons (1 and 2) appear on the same electrode. The rightmost waveform labeled '1 + 2' represents a spike overlap that is misclassified by many clustering algorithms.

explain an overlap (Atiya, 1992; Lewicki, 1994), or by searching for overlapping spikes only in those cases, where a fit by single spikes fails (Rinberg et al., 2003; Zhang et al., 2004; Wang et al., 2006). However, as most overlap algorithms restrict the overlap search to small data windows identified by a spike-detection step, these algorithms tend to also suffer from biases introduced by the detection step. Note that independent component analysis (ICA) is in principle able to unmix spike waveforms without a prior detection step (Brown et al., 2001); however, there is no guarantee that the independent components correspond to different neurons. Therefore, the unmixed data has yet to be sorted (Takahashi et al., 2003a, b), making ICA a useful pre-processing but not a self-contained spike-sorting solution. Furthermore, ICA is severely limited, as with a tetrode or multitrode of n electrodes, at most n neurons can be separated, but not more.

We apply probabilistic hidden Markov models (HMMs) to the spike-sorting problem. Such models have been applied to just the classification problem before (Sahani et al., 1998), but not as generative models to continuous data streams. There are two independent algorithms to our spike-sorting method, none of which relies on a spike detection step. In the first algorithm, the spike templates of the various cells, their firing probabilities, and the Gaussian noise parameters are learned. Reminiscent of blind source separation algorithms, spike templates can be learned even in the extreme case when all spikes overlap and neurons never fire in isolation. In the second algorithm, learned spike templates are superposed to produce an optimal fit to the raw data, thus yielding spike-time information. The fitting algorithm is in a sense independent of the learning algorithm; it can be applied as a post-processing step to any existing spike-sorting algorithm and provides the benefits of few spike misses and good sorting performance on data with spike overlaps. We illustrate our algorithm on extracellular signals from sleeping songbirds and assess its performance on artificial data taken from (Quiroga et al., 2004).

2. Results

Our spike-sorting technique is based on hidden variables that each represent the phase of an action potential cycle of a cell. For didactic purposes, first we introduce a one-variable HMM for single cell recordings. Then we address the multi-neuron case by generalizing our model to many variables and introduce the two algorithms used for learning and curve fitting. Finally, we explore a simple method for trading off false positive spike classifications against false negatives (misses).

2.1. Single neuron model—circular hidden variable

The data samples of the extracellular voltage signal are denoted by O_t $(t = 1, \ldots, T)$, where T is the total number of available samples. Our model associates sample O_t with no spike (silent state) when the hidden (random) variable S equals one at that time, $S_t = 1$. The next data sample O_{t+1} is associated with a spike onset, if the hidden variable jumps into state 2, $S_{t+1} = 2$. The state space of S has a ring structure: once a spike is initiated, S must step through all $K - 1$ spike states before jumping back into the silent state. State transitions are governed by a single parameter, which is the probability p of stepping out of the silent state and into the spike-onset state 2 (Fig. 2A):

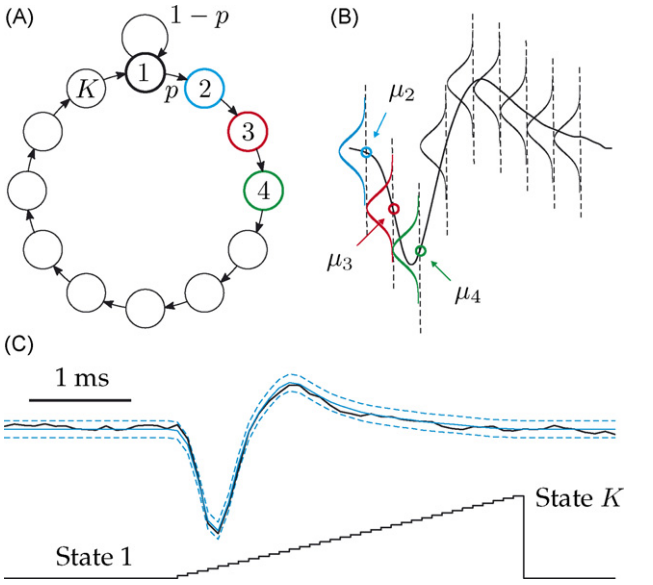

Fig. 2. Generative model of a spike. (A) The extracellular signal is modeled as a random (hidden) variable with a ring structure. The hidden state labeled 1 is the silent state in which the neuron is inactive. With probability p per sample time, the hidden variable leaves the silent state and jumps into the first state of a spike waveform (state 2). Thereafter, the variable steps through states 3 to K with probability one. (B) When the variable is in state i, the extracellular voltage is modeled by a Gaussian random variable (colored and black bell-shaped curves) with mean μ_i. (C) An example data fit. Top: The raw data O (black line) is fit by the model output (blue line), the dashed blue lines delimit $\pm 2\sigma$. The bottom plot shows the most likely hidden sequence S^* associated with the fit on top, it revolves once around the ring.

$$\mathrm{P}(S_{t+1} = 2 | S_t = 1) = p. \tag{1}$$

Conservation of probabilities implies that $\mathrm{P}(S_{t+1} = 1 | S_t = 1) = 1 - p$ and the ring structure implies that $\mathrm{P}(S_{t+1} = m | S_t = m - 1) = 1$ for $2 < m \leq K$. As a function of K, the model can implement a refractory period. For example, for a sampling rate of 30 kHz, a refractory period of 1 ms can be enforced in rings with $K = 30$ different hidden states.

We assume there exists a source of Gaussian white noise that affects observed samples (Fig. 2B). The mean μ of the Gaussian density associated with sample O_t depends on the state S_t of the hidden variable. We define the state-conditional sample probability distribution as

$$\mathrm{P}(O_t | S_t = k) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(O_t - \mu_k)^2\right], \tag{2}$$

where the Gaussian means $\mu_1, \ldots, \mu_K$ form the spike template and σ^2 is the fixed noise variance.

To work with the hidden Markov model just defined we face the following two problems.

- The first problem is the parameter estimation problem, also known as the learning problem. From given data $O = (O_1, \ldots, O_T)$ we must find the unknown spike-onset probability p, the unknown spike template $\mu = (\mu_1, \ldots, \mu_K)$, and the unknown noise variance σ^2.

- The second problem is the reconstruction problem. From given data O and given model parameters $\Phi = (p, \mu, \sigma^2)$, we must determine the most likely hidden sequence $S^* = (S_1^*, \ldots, S_T^*)$, i.e. find all spike-onset times.

A well known solution to the learning problem is the Baum–Welch algorithm (Baum et al., 1970). In this algorithm, starting from initial guesses, parameters estimates are iterated in such a way that the likelihood of the observed data monotonically increases until it reaches a local maximum. Each iteration is based on the computation of forward and backward probabilities, requiring the storage of $2TK$ numbers.

The reconstruction problem can be solved using dynamic programming (also known as the Viterbi algorithm (Forney, 1973)). This algorithm finds the set of spike times that maximizes the joint probability $\mathrm{P}(O, S|\Phi)$ of samples and hidden states given model parameters. Briefly, this algorithm comprises a forward pass in which for each sample O_t, the state likelihoods of the hidden variable are evaluated (remembering for each state and each time step the most likely state visited one step earlier). After completion of the forward pass, the most likely final state is selected and the most likely preceding hidden states are determined iteratively by backtracking. The forward pass of this algorithm requires a total of TK^2 multiplications and the memorization of KT previously visited hidden states. The learning and reconstruction algorithms are presented in more detail in Section 2.3.

We illustrate the learning and reconstruction algorithms on extracellular data from a single neuron recorded with a glass electrode in a sleeping songbird (for the protocols and procedures of head-fixed, sleeping bird preparation, see (Hahnloser et al., 2006)). Model parameters were learned from an extracellular trace containing several spikes. After three iterations of the Baum–Welch algorithm, the Gaussian means (the spike template) corresponded well to the spike waveform of the recorded cell, Fig. 2B. The spike-onset probability p converged to the fraction of data samples associated with a spike onset (in general, with firing rates smaller than 100 Hz and a sample rate of 30 kHz, p is typically smaller than 1/300). The learned noise variance was very small, reflecting the high signal-to-noise ratio of the recording. After reconstructing the data, spike-time information could be simply read out from the optimal hidden sequence S^* (the most likely hidden sequence visited state 2 right at the spike onset in Fig. 2C). Mathematically, the reconstruction algorithm is equivalent to computing a least-squares fit to the data, subject to a spiking cost of $-\log p$. This equivalence stems from the fact that maximizing the log-likelihood of data and hidden sequences $\log \mathrm{P}(O, S|\Phi)$ is equivalent to finding the sequence S^* that minimizes

$$S^* = \arg\min_S \left[\frac{1}{\sigma^2} \sum_t |O_t - \mu_{S_t}|^2 + C(S) \right], \quad (3)$$

which is the squared distance between the data and the model fit (in units of variance), plus a cost $C(S)$ that depends only on the hidden sequence, but not the data. This cost is such that each spike contributes a value of $-\log p + K \log(1-p)$ to the total cost. Because spike probabilities are small ($p \ll 1$), we can neglect the second term and find that the cost per spike in the sense of a least-squares fit is roughly given by $-\log p$. With firing rates in the range 1–100 Hz, the cost per spike is in the range $-\log p =$ 6–10. We will see that this cost typically is small compared to the cost imposed by bad fits. Next we turn to more realistic situations in which many cells are recorded on one or several nearby electrodes.

2.2. Full model

2.2.1. Many electrodes

There are two important generalizations of the simple model in Section 2.1. First, when dealing with tetrode recordings, we may want to search for spike signals on all electrodes simultaneously. Accordingly, the t th data sample is now represented as a sample vector $\underline{O}_t = \begin{pmatrix} O_t^1 & O_t^2 & \ldots & O_t^D \end{pmatrix}$ of dimension $D \times 1$, where D is the number of electrodes. Each element of an observation sequence $\underline{O} = \{\underline{O}_1, \ldots, \underline{O}_T\}$ is then modeled by a multi-dimensional Gaussian distribution of constant covariance C. The probability distribution of sample vector $\underline{O}_t$ conditional on the hidden state S_t is defined by:

$$\mathrm{P}(\underline{O}_t|S_t = k) = \frac{1}{\sqrt{(2\pi)^D |C|}} \exp\left[-\frac{1}{2} (\underline{O}_t - \underline{\mu}_k)^{\mathrm{T}} C^{-1} (\underline{O}_t - \underline{\mu}_k) \right], \quad (4)$$

where $\underline{\mu}_k = \begin{pmatrix} \mu_k^1 & \mu_k^2 & \ldots & \mu_k^D \end{pmatrix}$ is of dimension $D \times 1$, and μ_k^d represents the mean voltage of the d th electrode when the hidden variable is in state k. The expression $|C|$ stands for the determinant of C and superscript T denotes the transpose of a matrix.

2.2.2. Many neurons

When we wish to sort the spikes from many simultaneously recorded neurons, we introduce a hidden (random) variable S^n for each neuron n, leading to what is formally known as a factorial HMM (Ghahramani and Jordan, 1997). Assuming N neurons, the states of the hidden variables can be summarized by the vector $\underline{S}_t = \begin{pmatrix} S_t^1 & \ldots & S_t^N \end{pmatrix}$, where S_t^n is a number between 1 and K, representing the state of the n-th hidden variable associated with data sample t. The transition probability from state $\underline{S}_{t-1}$ to $\underline{S}_t$ follows from the assumption that spike onset probabilities are independent among neurons:

$$\mathrm{P}(\underline{S}_t|\underline{S}_{t-1}) = \prod_{n=1}^{N} \mathrm{P}(S_t^n|S_{t-1}^n). \quad (5)$$

For the n-th neuron, the probability of stepping from the silent state into the spike state is given by the parameter $p^n = \mathrm{P}(S_{t+1}^n = 2|S_t^n = 1)$ (cf. Eq. (1)).

The key assumption in our factorial HMM is that the contributions of individual neurons to the extracellular voltage sum up linearly. That is, we replace the mean $\underline{\mu}_k$ of our Gaussian model in Eq. (4) by the sum

$$\underline{\mu}(k_1, k_2, \ldots, k_N) = \sum_{n=1}^{N} \underline{\mu}_{k_n}^n, \quad (6)$$

where $\underline{\mu}_{k_n}^n$ is the vector of means (dimension $D \times 1$) that applies when the n-th hidden variable is in state k_n.

In summary, in the many-electrodes and many-neurons case, we have to learn the N spike probabilities $\underline{p} = \begin{pmatrix} p^1 & p^2 & \ldots & p^N \end{pmatrix}$, we have to learn the $D \times D$ covariance matrix C of the correlated noise model, and we have to learn the N spike templates $\begin{pmatrix} \bar{\mu}^1 & \bar{\mu}^2 & \ldots & \bar{\mu}^N \end{pmatrix}$, where $\bar{\mu}^n = (\underline{\mu}_1^n, \underline{\mu}_2^n, \ldots, \underline{\mu}_K^n)$ is the template of the n-th neuron of dimension $D \times K$. The reconstruction problem is the problem of finding the most likely sequence of hidden state vectors $\underline{S}^* = (\underline{S}_1^*, \underline{S}_2^*, \ldots, \underline{S}_T^*)$.

There is one practical problem of using our algorithm to sort the spikes of many neurons (more than 2), which is the associated computational cost. This issue is discussed in the following.

2.2.3. The curse of dimensionality

The amount of computer memory as well as the computational cost required to run the learning and reconstruction algorithms

grows exponentially with the number of neurons N. For example, for a recording of 1 s duration, to sort the spikes of three neurons calls for 20 GB of memory (assuming a sampling rate of 40 kHz, $K = 40$, and double precision arithmetic). To sort $N = 4$ neurons would require 800 GB of memory, a storage capacity no PC is able to accommodate in random access memory (RAM) today.

Concerning parameter learning, there are two ways how to bypass this inherent intractability. First, we can avoid using the Baum–Welch algorithm for learning the model parameters. For example, any traditional spike-sorting algorithm can be used to find spike templates $\bar{\mu}^n$ for the different neurons; and, spike probabilities p^n can be estimated from typical firing rates of cells. Finally, the covariance matrix of the Gaussian model can be estimated by computing the covariance matrix of a patch of data containing no or only few spikes. A second solution to the intractability of the learning algorithm is to run the Baum–Welch algorithm on a restricted state space, where spike overlaps are forbidden (this works well if overlaps are not too frequent).

For reconstruction, there is no good alternative to the Viterbi algorithm. Again, the computational cost of reconstruction can be reduced by imposing restrictions on the allowable hidden state space. For example, if we allow not more than R neurons to simultaneously spike, we reduce the memory load from $K^N \times T$ to about $\binom{N}{R} K^R \times T$, where $\binom{N}{R}$ represents the binomial factor "N choose R". For example for $N = 3$, we can reduce the memory requirement from 20 GB to 1.5 GB when we allow for single overlaps only ($R = 2$). Such restriction of the overlap space has also been applied in (Atiya, 1992; Lewicki, 1994). The error introduced by ignoring overlaps of more than two templates is often negligible.

Note that one additional method to reduce the computational load of both parameter learning and data reconstruction is by a variational approximation method presented in (Ghahramani and Jordan, 1997). An evaluation of this method for spike sorting goes beyond the scope of this work.

2.3. Descriptions of the Baum–Welch and Viterbi algorithms

We present a detailed description of the two algorithms used for learning and reconstruction. The methods are based on two classical algorithms: the Baum–Welch algorithm for learning the optimal model parameters given the data (Baum et al., 1970), and the Viterbi algorithm for computing the most likely states of hidden variables given the data and the model parameters (Forney, 1973). We provide the formulas for these algorithms in the context of spike sorting, but omit their mathematical derivation. For an introduction to HMMs see (Rabiner, 1989).

2.3.1. Glossary of variables

N	number of neurons (i.e. hidden variables)
K	number of states per neuron
D	dimensionality of observations (number of channels per tetrode)
T	number of observations (data samples)
$\underline{O}_t$	observation variables at time t (dimension $D \times 1$)
$\underline{S}_t$	hidden variables at time t (dimension $K^N \times 1$)
Φ	the set of model parameters $(\bar{\mu}^1, \ldots \bar{\mu}^N, p^1, \ldots, p^N, C)$
$\bar{\mu}^n$	n-th spike template (dimension $D \times K$)
$\bar{\mu}^{1,\ldots,N}$	concatenation of spike templates (dimension $D \times NK$)
p^n	spike-onset probability of neuron n
C	covariance of Gaussian noise model (dimension $D \times D$)
$\underline{\alpha}_t$	forward probabilities at time t (dimension $K^N \times 1$)
$\underline{\beta}_t$	backward probabilities at time t (dimension $K^N \times 1$)
$\underline{\gamma}_t$	posterior probabilities at time t (dimension $K^N \times 1$)
$\underline{e}_i$	$\underline{e}_{i,j} = \delta ij$(dimension $K^N \times 1$). The Kronecker delta is defined as $\delta ij = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$,
ς_t^n	marginal posterior probabilities of hidden variable n at time t (dimension $K \times 1$)
$\varsigma_t^{1,\ldots,N}$	concatenation of the marginal posterior probabilities (dimension $NK \times 1$)

2.3.2. The Baum–Welch algorithm

To find optimal model parameters $\Phi = (\bar{\mu}^1, \ldots \bar{\mu}^N, p^1, \ldots, p^N, C)$ we want to maximize the log-likelihood of the data $\sum_t \log \mathrm{P}(\underline{O}_t|\Phi)$, which is hard. The idea of the Baum–Welch algorithm is to iteratively maximize the simpler function $B(\Phi^r, \Psi)$. That is, from the current estimated parameters Φ^r, the improved parameters Φ^{r+1} are determined from

$$\begin{aligned} \Phi^{r+1} &= \arg\max_{\Psi} B(\Phi^r, \Psi) \\ &= \arg\max_{\Psi} \sum_t \sum_{\underline{S}_t} \mathrm{P}(\underline{S}_t|\underline{O}_t, \Phi^r) \log \mathrm{P}(\underline{O}_t, \underline{S}_t|\Psi), \end{aligned} \tag{7}$$

where, the maximization in Eq. (7) is based on the expectation step and the maximization step.

The expectation step: To compute the posterior probability distribution $\mathrm{P}(\underline{S}_t|\underline{O}_t, \Phi^r)$ in Eq. (7), i.e., the distribution of hidden variables $\underline{S}_t$ given the observation $\underline{O}_t$ (and model parameters Φ^r), we use the forward–backward algorithm. In this algorithm, for each time t we define forward and backward variables $\underline{\alpha}_t$ and $\underline{\beta}_t$. The dimension of these variables is $K^N \times 1$, corresponding to an orderly arrangement of the K^N different state combinations of hidden variables $\underline{S}_t$. The ith component $\underline{\alpha}_{t,i}$ of the forward variable $\underline{\alpha}_t$ corresponds to the joint probability of the observation sequence $(\underline{O}_1, \ldots, \underline{O}_t)$ and that the hidden variables $\underline{S}_t$ are in the ith state combination, given the model parameters Φ^r. Similarly, the ith component $\underline{\beta}_{t,i}$ of the backward variable $\underline{\beta}_t$ corresponds to the joint probability of the future observation sequence $(\underline{O}_{t+1}, \ldots, \underline{O}_T)$ and that hidden variables $\underline{S}_t$ are in the ith state combination and, given the same model parameters.

After suitable initialization, we iteratively calculate the forward probabilities $\underline{\alpha}$ and backward probabilities $\underline{\beta}$ as follows:

Forward probability $\underline{\alpha}$:

$$\begin{aligned} &\underline{\alpha}_1 = \underline{\delta}_1, \\ &\underline{\alpha}_{t+1} = \mathrm{P}(\underline{O}_{t+1}|\underline{S}_{t+1}, \Phi^r) \sum_{\underline{S}_t} \mathrm{P}(\underline{S}_{t+1}|\underline{S}_t) \underline{\alpha}_t \\ &\text{Rescaling: } \underline{\alpha}_{t+1}{}^{\text{rescaled}} = \frac{\underline{\alpha}_{t+1}}{|\underline{\alpha}_{t+1}|_1}. \end{aligned} \tag{8}$$

Backward probability $\underline{\beta}$:

$$\begin{aligned} &\underline{\beta}_1 = \underline{\delta}_1, \\ &\underline{\beta}_{t-1} = \sum_{\underline{S}_t} \mathrm{P}(\underline{O}_t|\underline{S}_t, \Phi^r) \mathrm{P}(\underline{S}_t|\underline{S}_{t-1}) \underline{\beta}_t \\ &\text{Rescaling: } \underline{\beta}_{t-1}{}^{\text{rescaled}} = \frac{\underline{\beta}_{t-1}}{|\underline{\beta}_{t-1}|_1}. \end{aligned} \tag{9}$$

Here, $|\cdot|_1$ denotes the L_1-norm ($|x|_1 = \sum_i |x_i|$); the rescaling is required to prevent numerical underflow. By our model assumption, $\mathrm{P}(\underline{O}_t|\underline{S}_t, \Phi^r) = \mathcal{N}_D(\underline{\mu}, C)$ is a multivariate Gaussian density with covariance C and mean $\underline{\mu} = \underline{\mu}(k_1, \ldots, k_N)$ (cf. Eq. (6)).

We write the transition probability $\mathrm{P}(\underline{S}_{t+1}|\underline{S}_t)$ as a $(K\times K)^N$ transition probability matrix:

$$\mathrm{P}(\underline{S}_{t+1}|\underline{S}_t) = \begin{bmatrix} 1-p^1 & 0 & 0 & \dots & 0 & 1 \\ p^1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1-p^N & 0 & 0 & \dots & 0 & 1 \\ p^N & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{10}$$

The posterior probability $\mathrm{P}(\underline{S}_t|\underline{O}_t, \Phi^r)$ in Eq. (7) is defined by elementwise multiplication of $\underline{\alpha}_t$ and $\underline{\beta}_t$, and rescaling:

Posterior probability $\underline{\gamma}$:

$$\underline{\gamma}_t = \mathrm{P}(\underline{S}_t|\underline{O}_t, \Phi^r) = \frac{\underline{\alpha}_t \underline{\beta}_t}{|\underline{\alpha}_t \underline{\beta}_t|_1}. \tag{11}$$

Note on the restricted state space: When we restrict the state space, i.e. when we allow not more than R neurons to simultaneously spike (constrained spike overlaps, see Section 2.2.3), we have to make some adaptations. For example, if $R=2$, then the number of possible hidden states is reduced from K^N to $1+N(K-1)+N(N-1)(K-1)^2/2$ (corresponding to zero, one or two active neurons). The dimensions of $\underline{\alpha}$ and $\underline{\beta}$ are also reduced, and transitions to and from forbidden states disappear from the transition matrix. With this dimensionality reduction, the computation of posterior probabilities is faster and requires less memory.

The maximization step: Given the posterior probabilities $\underline{\gamma}_t$, we maximize the auxiliary function $B(\Phi^r, \psi)$ in Eq. (7) by differentiation. We thus find for the updated templates:

Update of $\bar{\mu}^{1,\dots,N}$:

$$\bar{\mu}^{1,\dots,N}_{r+1} = \left(\sum_{t=1}^{T} \underline{O}_t \underline{\varsigma}_t^{1,\dots,N\mathrm{T}}\right)\left(\sum_{t=1}^{T} \underline{\varsigma}_t^{1,\dots,N} (\underline{\varsigma}_t^{1,\dots,N})^{\mathrm{T}}\right)^{\dagger}, \tag{12}$$

where $\bar{\mu}^{1,\dots,N}_{r+1}$ is the concatenation of the matrices $\bar{\mu}^n$ (of dimension $D\times NK$) and $\dagger$ denotes the Moore-Penrose pseudo-inverse. $\underline{\varsigma}_t^{1,\dots,N}$ is the concatenated vector of the marginal posterior probabilities $\underline{\varsigma}_t^n$, defined elementwise by:

$$\varsigma^n_{t,k} = \mathrm{P}(S^n_t = k|\underline{O}_t, \Phi^r) = \sum_{\substack{j,\text{with} \\ S^n_t = k}} \gamma_{t,j}. \tag{13}$$

Similarly, the spike-onset probabilities p^n for the neurons $n=1,\dots,N$ are found by maximizing the auxiliary function $B(\Phi^r, \Psi)$ subject to the constraint that probabilities must sum to one, leading to:

Update of p^n:

$$p^n_{r+1} = \frac{\sum_{t=2}^{T} \varsigma^n_{t,2}}{\sum_{t=2}^{T} \varsigma^n_{t-1,1}}, \tag{14}$$

The last parameter to be updated is the covariance matrix:

Update of C:

$$C_{r+1} = \frac{1}{T}\sum_{t=1}^{T} \underline{O}_t \underline{O}_t^{\mathrm{T}} - \frac{1}{T}\sum_{t=1}^{T}\sum_{n=1}^{N} \underline{\mu}^n \underline{\varsigma}_t^n \underline{O}_t^{\mathrm{T}}. \tag{15}$$

Eqs. (12), (14), and (15) constitute the parameter updates.

2.3.3. The Viterbi algorithm

The Viterbi algorithm is a trellis algorithm to find the single most probable state sequence $\underline{S}^* = (\underline{S}^*_1, \dots, \underline{S}^*_T)$ for a given observation sequence $\underline{O} = (\underline{O}_1, \dots, \underline{O}_T)$,

$$\underline{S}^* = \underset{\underline{S}}{\mathrm{argmax}} \mathrm{P}(\underline{S}, \underline{O}|\Phi). \tag{16}$$

We define the best-sequence probability to state combination i at time t by

$$V_{t,i} = \max_{\underline{S}_t,\dots,\underline{S}_{t-1}} \mathrm{P}(\underline{O}_1, \dots, \underline{O}_t, \underline{S}_t, \dots, \underline{S}_{t-1}, \underline{S}_t = \underline{e}_i|\Phi). \tag{17a}$$

Starting with $V_1 = \underline{e}_1$, we find the best-sequence probabilities at later times by induction:

$$V_{t+1,j} = \max_i [V_{t,i}\mathrm{P}(\underline{S}_{t+1} = \underline{e}_j|\underline{S}_t = \underline{e}_i)]\,\mathrm{P}(\underline{O}_{t+1}|\underline{S}_{t+1} = \underline{e}_j),$$
$$\begin{cases} 1 & \le i \le K^N \\ 2 & \le t \le T. \end{cases} \tag{17b}$$

To prevent underflow in this calculation, there are two possibilities; we can do all the calculations in the log-domain or include a rescaling step:

$$V^{\mathrm{rescaled}}_{t,i} = \frac{V_{t,i}}{|V_t|_1}. \tag{18}$$

We keep track of the argument that maximizes Eq. (17b) with the backtracking matrix $B = (B_{t,j})$:

$$B_{t+1,j} = \underset{i}{\mathrm{argmax}} [V_{t,i}\mathrm{P}(\underline{S}_{t+1} = \underline{e}_j|\underline{S}_t = \underline{e}_i)], \qquad \begin{cases} 1 & \le j \le K^N \\ 2 & \le t \le T. \end{cases} \tag{19}$$

The best sequence $\underline{S}^*$ is now found by tracing back:

$$\underline{S}^*_T = \underline{e}_k \ \text{ where } k = \mathrm{argmax}_i [V_{T,i}] \text{ and} \tag{20a}$$

$$\underline{S}^*_T = \underline{e}_{B_{t+1,k}}, \quad \text{where } k \text{ is such that } \underline{S}^*_{t+1} = \underline{e}_k, \quad 1 \le t \le T-1. \tag{20b}$$

Sample code that implements the learning and reconstruction algorithms can be downloaded from http://www.ini.uzh.ch/ rich/software/index.html. For didactic purposes, the code sorts the spikes from just a single neuron and is written in Matlab (Mathworks Inc.).

2.4. Practical considerations and fully automated spike sorting

Our algorithm is straightforward to use except that in practice some skills are needed for initial parameter selection (including the number of neurons N). For completeness, we present a fully automated scheme that allows for spike sorting without any user interference.

Initialization: We found that fastest parameter convergence in the Baum–Welch algorithm was achieved by choosing as initial spike templates manually selected example spikes (the parameters typically converged in two iterations). However, to allow for fully automated operation of our algorithm we worked mainly with generic initial templates consisting of zeros and 1.5 periods of a sine wave between states 6 and 15 ($K=30$, 24 kHz sampling rate). Low sampling rates can introduce a non-negligible amount of jitter to spike waveforms (Blanche and Swindale, 2006). For noisy files we

improved results when we up-sampled the data to 48 kHz using cubic splines (necessitating a doubling of K). The amplitude of the sine wave was randomly chosen from a uniform distribution in the interval [0,5*std(data)]. Remaining initial parameters were $N = 8$, $C = \mathrm{cov}(\mathrm{data})$ and $p^i = 10^{-3}$ for all neurons.

Learning: Usually not all initial templates converged to an actual spike waveform, but some to noisy signals or to single outliers. We took care of these cases in the following manner. We ran 8 iterations of the Baum–Welch algorithm (with a $\binom{N}{1}$-state-space, see Section 2.2.3) on 4 s of data ($T = 96000$, 24 kHz sampling rate, band-pass filtered between 100 Hz and 6 kHz). Templates that converged to outlier events were removed after each iteration by inspecting the learned parameters $p^1, \ldots, p^8$: if for some i the probability p^i corresponded to a firing rate of 0.5 Hz or less, we re-initialized the template i by randomly picking a template j with larger parameter p^j and multiplying this template by 0.99. When the signal-to-noise ratio of the recording is very low, all templates tend to converge to patches of noise. In these cases we re-initialized spike probabilities to $p^i = 10^{-10}$ after each learning iteration; this procedure prevents the templates from converging to noise.

Template condensation: After eight iterations, the spike templates were condensed as follows. To test whether, for example, templates one and two were identical, we up-sampled templates (using cubic spline with 10 times finer mesh), aligned them by maximum cross-correlation, and down-sampled them to result in perfectly aligned candidate templates $\bar{\mu}^{1'}$ and $\bar{\mu}^{2'}$. If two candidate templates corresponded to the same neuron, their differences $\underline{X}_k = \underline{\mu}_k^{1'} - \underline{\mu}_k^{2'}$ should be a sequence of Gaussian distributed random variables with mean zero and covariance matrix $C_1 + C_2$ (to be determined). Accordingly, $\theta = \sum_{k=1}^{K} (\underline{X}_k)^{\mathrm{T}} (C_1 + C_2)^{-1} (\underline{X}_k)$ should be a random variable obeying a chi-square distribution with KD degrees of freedom. We thus combined two candidate templates depending on the outcome of the chi-square test statistic $\theta < \chi^2_\alpha(KD)$ with confidence level α. To estimate the covariance $C_1 + C_2$ we realized that candidate templates are never truly independent of each other, because the learning algorithm tries to maximally distribute spike-shape variability between any two templates. We therefore used as our estimate of covariance the following lower-bounded heuristic expression: $C_i = C \cdot \max((4)/(p^i T), (1)/(12))$. When the candidate templates were identical on the 1 ‰-level, we combined them by calculating the weighted mean $\bar{\mu}^{\mathrm{new}} = (p^1 \bar{\mu}^{1'} + p^2 \bar{\mu}^{2'})/(p^1 + p^2)$. This procedure was repeated, until no templates were identical on the 1 ‰-level. When two or more templates were combined, additional learning iterations were performed, because combined templates usually were not completely stable solutions and could still change after re-learning.

Final template selection: If no templates could be combined in a condensation step, we stopped the learning procedure and eliminated templates that presumably correspond to noise events. These are templates with energy smaller than twice the expected energy of a noise patch of K samples length and with unreasonably high spiking probability p (i.e. firing rates larger than 100 Hz). Removing these templates at the end gave better results than removing them at an earlier stage of the learning procedure. To increase the speed of the reconstruction algorithm, it is possible to reduce the number K by removing redundant states at the beginning and the end of a template. The automated procedure is illustrated in Fig. 3.

2.5. The tradeoff between false positive and false negative spikes

We applied the automated sorting algorithm to single glass-pipette recordings from three neurons. For reconstruction we

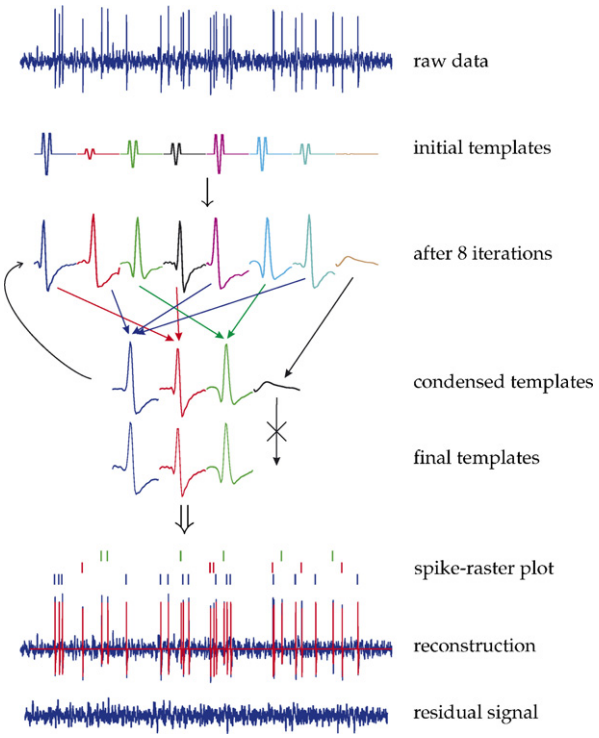

Fig. 3. Illustration of the fully automated algorithm: we start with raw data and eight random initial templates. After eight iterations of the Baum–Welch algorithm, templates are condensed according to mutual similarities and re-learned. After condensation and relearning, noise templates are removed (black cross). The remaining final (three) spike templates are fed into the Viterbi algorithm, and the optimal reconstruction (red line) is computed (minimal residual signal). The rasters depict the sorted spikes of the three neurons (with colors matching the final template colors).

reduced the state space to at most two simultaneously active neurons ($R = 2$, $N = 3$). Occasional spike overlaps were correctly identified, as assessed from the small residual signal (Fig. 4). One of the three neurons (Neuron 1) produced very small spikes. We have simultaneously recorded from that neuron with a second glass electrode. On this separate electrode, signals were very well isolated and could be used to identify sorting errors. We found that approximately 30% of the detected spikes in Neuron 1 were false positives, with an example shown in the right part of Fig. 4. Interestingly, this large number of false positive spikes in Neuron 1 was not caused by small spike amplitudes (the signal-to-noise ratio was 4.3), but by the shift similarity between spike Template 3 and Templates 3 and 1, illustrated in Fig. 5. Thus, false positive spikes may not only pose a problem when spike waveforms are small, but also when the temporal derivative of a waveform looks like another waveform on the same electrode.

We tested whether false positive spikes can be remedied by decreasing spike-onset probabilities p^i. The idea is that the cost-per-spike determines how much better a fit must be using a certain number of spike templates in order to be preferred over another fit using fewer templates. Interestingly, the reduction in log-likelihood that results from a missed spike in Neuron 1 (i.e., to fit the spike waveform with a sequence of silent states) is on the order of 100, a cost that largely outweighs the cost per spike of $-\log p^1 = 7$. Thus, typical values of spike onset probabilities entail the risk of overfitting the data. We found that substantially decreasing p^1 from 10^{-3} to 10^{-20} reduced the percentage of false positive spikes in Neuron 1 from 30% down to 1.7%, whereas the number of missed spikes increased only weakly from 1.1% to 3.3%. Hence, by manipulating spike-onset probabilities p^i, we can effectively trade off false positive against false negative spikes.

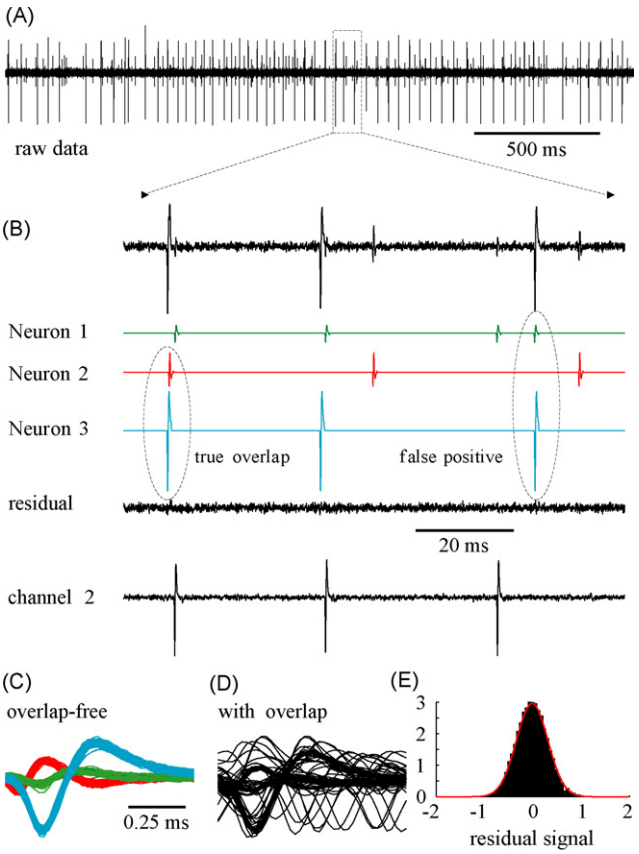

Fig. 4. Sorting the spikes of three neurons on a single glass electrode (3–5 M Ω). (A) Excerpt of raw data. (B) From the raw data (black line, top) three spike trains are extracted (Neurons 1–3, shown in color below), leading to a small residual signal (black line, below). Spike overlaps are indicated by the dashed ellipses. We recorded from Neuron 2 with a separate electrode (raw trace in the bottom), revealing that the overlap on the right contains a false positive spike of Neuron 1. (C) Aligned raw signals of all isolate spikes. (D) Aligned raw signals of all overlapping spikes. (E) The histogram of residual signals (black) is well fit by a Gaussian curve of variance σ^2 (σ^2 was derived in the learning algorithm). The units are arbitrary. For reconstruction, a $\binom{3}{2}$ hidden state space was used (see Section 2.2.3).

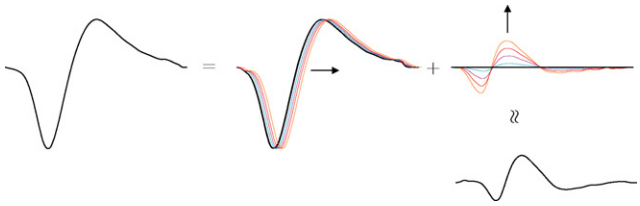

Fig. 5. Shift similarity of spike waveforms in Fig. 4: The spike template of Neuron 3 (left) can be similar to shifted copies of itself (middle) added to the template of Neuron 1 (bottom right). The colors blue to orange indicate different time shifts. For large shifts, the residual signal (orange curve on the right) strongly resembles the spike template of Neuron 1 below.

How to determine p^i in practice is non-trivial, because the answer depends on whether false positives or false negatives are worse. As a compromise, one may want to minimize the total number of misclassified spikes. When we did this, we counted confused spikes (spikes assigned to the wrong neuron) twice: once as false positive and once as false negative. For the glass-pipette recordings, the global minimum of total classification error (the sum of false positives and negatives) was very flat and was located at around $p^i \simeq 10^{-20}$ (Fig. 6A). We found that to implement a sparse prior independent of the data, a reasonable choice of p^i was to set the cost of spiking to three times the log-likelihood ratio of K noisy samples and K noiseless samples, leading to

$$p^i = 2^{-3KD/2}. \tag{21}$$

For the glass-pipette recordings, this choice implied $p^i = 10^{-18}$, which was close to the minimum at $p^i \simeq 10^{-20}$. We evaluated this choice of p^i also on a different data set, presented in the next section.

2.6. Comparison with other techniques

We evaluated our algorithm on artificial data sets made available by R. Quiroga at www.vis.caltech.edu/ rodri and we compared the sorting performance to that of WaveClus (Quiroga et al., 2004), a state-of-art technique based on voltage-thresholded spike detection and super-paramagnetic clustering of spike features. The artificial data comprises four sets of simulated extracellular signals. For each set a unique triple of spike waveforms was mixed with noise of four or eight different variances and written to data files of 60 s each. On these data, WaveClus outperforms conventional

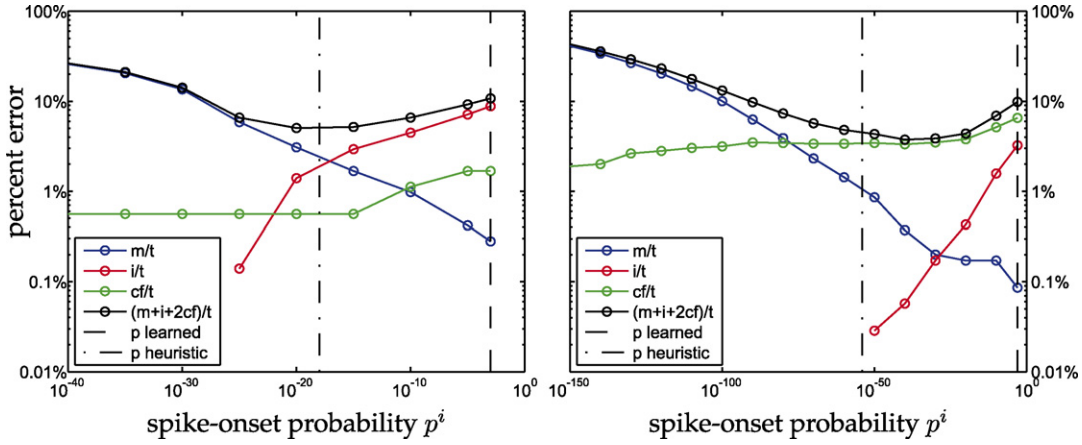

Fig. 6. Fractional classification errors for missed (m), falsely introduced (i), confused spikes (cf), and the sum of false negatives and false positives ($f-$) + ($f+$) relative to the total number of spikes (t), where $f+ = (i + cf)/t$ and $f- = (m + cf)/t$. For (A) we used the same glass-electrode data as in Fig. 4. The p^i heuristic was derived from Eq. (21). For (B), we used the simulated data set d2(0.20) (cf. Section 2.6).

Table 1
Comparison between WaveClus and HMM for spike-onset probabilities determined based on a heuristic derived in Section 2.5

Data set (noise level)	t	HMM						WaveClus					
		c	m	cf	i	$f-$ (%)	$f+$ (%)	c	m	cf	i	$f-$ (%)	$f+$ (%)
e1(0.05)	3514	3510	4	0	2	0.1	0.1	2929	585	0	0	16.6	0.0
(0.10)	3522	3521	1	0	0	0.0	0.0	3065	457	0	0	13.0	0.0
(0.15)	3477	3476	1	0	0	0.0	0.0	2849	628	0	1	18.1	0.0
(0.20)	3474	3463	10	1	0	0.3	0.0	2271	1213	0	8	34.6	0.2
(0.25)	3298	3230	68	0	1	2.1	0.0	1458	1839	1	8	55.8	0.3
(0.30)	3475	3265	210	0	8	6.0	0.2	879	2595	1	6	74.7	0.2
(0.35)	3534	3135	398	1	30	11.3	0.9	520	3003	11	10	85.3	0.6
(0.40)	3386	3092	294	0	390	8.7	10.9	270	3115	1	9	92.0	0.3
e2(0.05)	3410	3409	1	0	0	0.0	0.0	2873	537	0	0	15.7	0.0
(0.10)	3520	3518	2	0	0	0.1	0.0	3006	514	0	0	14.6	0.0
(0.15)	3411	3400	8	3	0	0.3	0.1	2510	898	3	0	26.4	0.1
(0.20)	3526	3508	5	13	5	0.5	0.5	1622	1895	9	0	54.0	0.3
d1(0.05)	3383	3377	5	1	1	0.2	0.1	2931	452	0	0	13.4	0.0
(0.10)	3448	3443	0	5	1	0.1	0.2	2982	465	1	0	13.5	0.0
(0.15)	3482	3464	2	6	0	0.5	0.2	2725	737	10	0	21.7	0.3
(0.20)	3414	3411	1	2	0	0.1	0.1	1188	2200	6	0	65.2	0.2
d2(0.05)	3364	3361	3	0	0	0.1	0.0	2852	512	0	0	15.2	0.0
(0.10)	3462	3457	5	0	0	0.1	0.0	3012	447	3	0	13.0	0.1
(0.15)	3440	3433	4	3	1	0.2	0.1	1821	1610	9	0	47.1	0.3
(0.20)	3493	3395	39	59	0	2.8	1.7	1111	2268	114	0	68.2	3.3

For each data file, the number of correctly classified (c), missed (m), confused (cf), and falsely introduced (i) spikes is reported. The percentage of false negatives relative to the total number of spikes (t) is calculated as $f- = (m + cf)/t$, and the percentage of false positives as $f+ = (cf + i)/t$.

methods based on principal components analysis or clustering techniques such as K-means.

Waveclus and many other spike-sorting algorithms make use of the wavelet transform for spike detection (Kim and Kim, 2003a, b; Nenadic and Burdick, 2005) and feature extraction (Zouridakis and Tam, 1997; Hulata et al., 2002; Quiroga et al., 2004). To be comparable to these algorithms, we formed a second data channel with coefficients of a continuous wavelet transform of the raw data. The first channel contained the original raw trace and the second channel the coefficients of the continuous Daubechies "db2" wavelet transform (for a detailed discussion of wavelet transforms of neural data, we refer to the literature). For both our algorithm and for WaveClus, we upsampled the data from 24 kHz to 48 kHz with a cubic spline (because it is known that low sampling rates can introduce a non-negligible amount of jitter to spike waveforms (Blanche and Swindale, 2006)). We chose $K = 60$, which according to Eq. (21) led to $p^i = 10^{-54}$; for the data sets e1 we chose $K = 80$ to cover the entire spike waveform (leading to $p^i = 10^{-72}$). The dependence of false positives and false negatives on p is illustrated for a particular file in Fig. 6B.

With the automated procedure described in Section 2.4, our algorithm robustly found the correct number of neurons in all cases, except for two files (e1(0.35) and e1(0.40)), for which the correct number was found in about 90% percent of cases (depending on random initial conditions). Note that WaveClus did not always find the correct number of neurons either. In these cases, we manually corrected the number to three (this was the case for files e1(0.30) – 2 clusters found, e1(0.35) – 2 clusters, e1(0.40) – 2 clusters, d1(0.10) – 5 clusters, and d2(0.20) – 2 clusters).

Waveclus and the HMM sorter produced similar numbers of false positives. However, the number of false negatives in the case of the HMM was typically between one and two orders of magnitudes smaller. Detailed numbers are reported in Table 1.

The small number of false negatives cannot merely be explained by overlapping spikes (which sum up to less than 15% and are ignored by WaveClus).

We also tested our algorithm on a data set in which spike amplitudes are variable, which can happen for example when electrodes drift in the brain. Spike sorting can be made robust in this case by dividing the raw data into small segments that are each sorted with a unique set of spike templates (Bar-Hillel et al., 2006). We adapted this approach to our algorithm by re-learning model parameters between reconstructions of adjacent data segments. We tested this procedure on a simulated file where the amplitude of one spike continuously decreases to 30% of its original value throughout the file. This file was part of the data set made available by (Quiroga et al., 2004). We sorted this file of 60 s duration by cutting it into 2-s pieces and repeatedly re-learning the parameters using one iteration of the Baum–Welch algorithm and the previous parameters as initial conditions. For re-learning, $p^i = 10^{-54}$ was fixed, cf. Eq. (21), in order to ensure that the template of the spike with the decreasing amplitude did not converge to patches of noise. Relearned spike templates always corresponded to the same neuron as in the previous segment. For reconstruction, we set $p^i = 10^{-54}$. We found that more than 90% of all errors occurred in the last third of the file, where the spike amplitude fell below 3σ. The resulting percentages of false negatives $f-$ and false positives $f+$ were $f- = 6.5\%$ and $f+ = 1.5\%$, comparable to WaveClus: $f- = 24.4\%$ and $f+$ 2.7%.

3. Summary and discussion

We have proposed a new solution to the spike-sorting problem in terms of a full generative model of extracellular data. Our solution consists of a learning algorithm for finding model parameters and a reconstruction algorithm for producing a fit to raw data. Both algorithms allow for spike overlaps to occur. In the past, given the speed and memory limitations of personal computers, such generative algorithms have been impractical.

In our algorithm there is no initial spike-detection step; parameter learning is not restricted to pre-selected data patches. Instead, parameters are estimated by summing over data samples weighted by their posterior probability, Eq. (12). Thanks to this summing property, parameter estimation is free of spike selection and is not affected by overlapping spikes. For simplicity, we did not include pre-processing steps into our algorithm such as whitening the raw data (in order to get rid of noise autocorrelation, (Bankman et al., 1993)) or independent component analysis (for multi-electrode recordings, (Brown et al., 2001)). However, we expect that some performance improvement could be achieved

using such techniques. For example, whitening the data from auto-correlated noise could increase spike-sorting performance because one of our model assumptions is presence of Gaussian white noise. Furthermore, independent component analysis could lead to more distinct spike waveforms on the de-mixed data channels, which might also improve sorting performance.

Overlap-permissive spike-sorting algorithms are at risk of detecting false positive spikes. In our algorithm, we found that reconstructions using learned spike-onset probabilities p^i introduced many false positive spikes either because of shift similarity or because of low signal-to-noise ratios. We have shown that false positive spikes can be effectively suppressed by decreasing p^i, i.e. by increasing the cost of spiking. Notice that our cost-of-spiking term $-\log p^i$ is derived from first-order principles (maximum likelihood estimation) and is comparable to the heuristic punishment term λ in the spike-sorting algorithm of (Segev et al., 2004). In other spike-sorting algorithms, false positives are avoided by increasing the number of spike templates only when a goodness-of-fit test fails (Zhang et al., 2004). In clustering-based algorithms, the number of false positive classifications is intrinsically minimized because outliers tend to remain unclassified (Quiroga et al., 2004), or because they are not separable when spike times are too close (Hulata et al., 2002). In summary, many methods for avoiding false positives exist and it may not matter which method is used as long as the problem is given adequate consideration.

Recently there has been growing interest in the structure of higher-order spike correlations (Schneidman et al., 2006; Weber and Hahnloser, 2007). Because correlation analyses are based on population recordings and so may necessitate advanced spike-sorting methods, it is important that no spurious correlations arise from spike-sorting errors, such as correlations between real spikes and false positive spikes. Overlap-permissive algorithms such as ours are at risk of producing spurious peaks in cross-correlation functions near zero time lag. Though this risk may represent a drawback of elaborate spike-sorting algorithms, our preliminary analysis indicates that correlations arising from false positive spikes are extremely tight (their width is on the order of a few tens of microseconds) and so can be easily distinguished from true peaks in cross-correlation functions (which typically are at least 1 ms wide). In fact, one may turn this possible drawback into an advantage and use the ultra precision of spurious correlations as a criterion for detecting false positive spikes, to be explored.

Last but not least, one of the key features of our algorithm is that many fewer spikes are missed (false negatives) than in many other algorithms. Our algorithm is thus very useful in cases in which spike misses are highly undesirable. In practice, missed spikes may pose a problem during spike bursts or during synchronous spiking events. We ascribe the low false-negative detection rate of our algorithm to absence of an explicit spike-detection step, which is one of the main advantages of full generative models of extracellular records.

Acknowledgements

This work was supported by a professorship grant to R.H. from the Swiss National Science Foundation. J.H. was supported by a Ph.D. grant from the Neuroscience Center Zurich (ZNZ).

References

Atiya AF. Recognition of multiunit neural signals. IEEE Trans Biomed Eng 1992;39(7):723–9.
Bankman IN, Johnson KO, Schneider W. Optimal detection, classification, and superposition resolution in neural waveform recordings. IEEE Trans Biomed Eng 1993;40(8):836–41.
Bar-Hillel A, Spiro A, Stark E. Spike sorting: Bayesian clustering of non-stationary data. J Neurosci Methods 2006;157(2):303–16.
Baum LE, Petrie T, Soules G, Weiss N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann Math Stat 1970;41(1):164–71.
Blanche TJ, Swindale NV. Nyquist interpolation improves neuron yield in multiunit recordings. J Neurosci Methods 2006;155(1):81–91.
Brown EN, Kass RE, Mitra PP. Multiple neural spike train data analysis: state-of-the-art and future challenges. Nat Neurosci 2004;7(5):456–61.
Brown GD, Yamada S, Sejnowski TJ. Independent component analysis at the neural cocktail party. Trends Neurosci 2001;24(1):54–63.
Delescluse M, Pouzat C. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. J Neurosci Methods 2006;150(1):16–29.
Fee MS, Mitra PP, Kleinfeld D. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability. J Neurosci Methods 1996;69(2):175–88.
Forney GD. The Viterbi algorithm. Proc IEEE 1973;61(3):268–78.
Ghahramani Z, Jordan MI. Factorial hidden Markov models. Mach Learn 1997;29(2–3):245–73.
Gray CM, Maldonado PE, Wilson M, McNaughton B. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. J Neurosci Methods 1995;63(1–2):43–54.
Hahnloser RHR, Kozhevnikov AA, Fee MS. Sleep-related neural activity in a premotor and a basalganglia pathway of the songbird. J Neurophysiol 2006;96(2):794–812.
Harris KD, Henze DA, Csicsvari J, Hirase H, Buzski G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. J Neurophysiol 2000;84(1):401–14.
Hulata E, Segev R, Ben-Jacob E. A method for spike sorting and detection based on wavelet packets and Shannon's mutual information. J Neurosci Methods 2002;117(1):1–12.
Kim KH, Kim SJ. Method for unsupervised classification of multiunit neural signal recording under low signal-to-noise ratio. IEEE Trans Biomed Eng 2003a;50(4):421–31.
Kim KH, Kim SJ. A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio. IEEE Trans Biomed Eng 2003b;50(8):999–1011.
Lewicki MS. Bayesian Modeling and Classification of Neural Signals. Neural Comput 1994;6(5):1005–30.
Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Network 1998;9(4):R53–78.
Nenadic Z, Burdick JW. Spike detection using the continuous wavelet transform. IEEE Trans Biomed Eng 2005;52(1):74–87.
Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput 2004;16(8):1661–87.
Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 1989;77(2):257–86.
Rinberg D, Bialek W, Davidowitz H, Tishby N. Spike sorting in the frequency domain with overlap detection. http://arxiv.org/abs/physics/0306056v2; 2003:1–31.
Rinberg D, Davidowitz H, Tishby N. Multi-electrode spike sorting by clustering transfer functions. Advances in neural information processing systems, vol. II. Cambridge: MIT Press; 1999. p. 146.
Sahani M, Pezaris JS, Andersen RA. On the separation of signals from neighboring cells in tetrode recordings. In: Advances in neural information processing systems, vol. 10. Cambridge: MIT Press; 1998. p. 222.
Schneidman E, Berry MJ, Segev R, Bialek W. Weak pairwise correlations imply strongly correlated network states in a neural population. Nature 2006;440:1007–12.
Segev R, Goodhouse J, Puchalla J, Berry II MJ. Recording spikes from a large fraction of the ganglion cells in a retinal patch. Nat Neurosci 2004;7(10):1155–62.
Shoham S, Fellows MR, Normann RA. Robust, automatic spike sorting using mixtures of multivariate t-distributions. J Neurosci Methods 2003;127(2):111–22.
Takahashi S, Anzai Y, Sakurai Y. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. J Neurophysiol 2003a;89(4):2245–58.
Takahashi S, Anzai Y, Sakurai Y. A new approach to spike sorting for multi-neuronal activities recorded with a tetrode—how ICA can be practical. Neurosci Res 2003b;46(3):265–72.
Wang GL, Zhou Y, Chen AH, Zhang PM, Liang PJ. A robust method for spike sorting with automatic overlap decomposition. IEEE Trans Biomed Eng 2006;53(6):1195–8.
Weber AP, Hahnloser RHR. Spike correlations in a songbird agree with a simple Markov population model. PLoS Comput Biol 2007;3:e249.
Wilson MA, McNaughton BL. Dynamics of the hippocampal ensemble code for space. Science 1993;261(5124):1055–8.
Wood F, Black MJ, Vargas-Irwin C, Fellows M, Donoghue JP. On the variability of manual spike sorting. IEEE Trans Biomed Eng 2004;51(6):912–8.
Zhang PM, Wu JY, Zhou Y, Liang PJ, Yuan JQ. Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem. J Neurosci Methods 2004;135(1–2):55–65.
Zouridakis G, Tam DC. Multi-unit spike discrimination using wavelet transforms. Comput Biol Med 1997;27(1):9–18.