This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

Sparse Coding and Compressive Sensing for Overlapping Neural Spike Sorting

Haifeng Wu, Kai Yang and Yu Zeng

***Abstract*—Spike sorting is one of key techniques to understand brain activity. With the development of modern electrophysiology technology, some recent multi-electrode technologies have been able to record the activity of thousands of neuronal spikes simultaneously. In this scenario, however, the recorded activity may be the overlap of multi-neuron spikes, which will degrade the sorting performance of existing cluster-based algorithms. In this paper, we introduce methods for overlapping spike sorting. The introduced methods start from a convolution model, where a sparse vector could be obtained via sparse coding or compressive sensing. Then, we use a maximum a posteriori (MAP) estimate to optimize the sparse vector, which make the overlapped spike sorting completed successfully. The advantage of the introduced method is that, it performs better than traditional methods when the waveforms of the spikes are similar. In experiments, some synthetic and real spike data are used to testify the methods. The experiment results show that the introduced methods' average sorting detection, defined as the ratio of successfully sorted spikes to the total spikes is nearly 4% higher than traditional methods, under the condition of the experimental data with similar waveforms.**

***Index Terms*—spike sorting; overlapping spike; sparse coding; compressive sensing**

I. Introduction

Neuronal spikes are basic units of a neural communication system. Neurons in a brain communicate with each other through the spikes. Hence, the study of neuron spikes is a key to understand brain activity. Spike sorting is a very important problem in neuroscience since the correct classification of different spikes means that the neurons that fire the different spikes would be distinguished. For example, spike sorting could distinguish the spike signals about memory from the spike signals about perception and learning [1].

Usually, neuronal spikes are collected from electrodes [2]. The aim of spike sorting is to find when the spikes occur and which neuron the spikes belong to. For traditional sorting algorithms, cluster and various cluster-based algorithms are very popular [2]-[10] and have already played an important role. With the development of modern electrophysiology technology, however, the number of spikes simultaneously collected by electrodes increases dramatically. Now, the multi-electrode technology has been able to record the activity of thousands of neuronal spikes simultaneously [11], [12]. In this scenario, the recorded activity may be the overlap of multiple neuronal spikes. Moreover, the neurons are adjacent and thus their waveform may be similar. All of these will degrade the performance of the traditional cluster-based sorting algorithms [13-15], and bring many challenges to the sorting algorithms. Some new methods have been proposed to solve the overlapping spike sorting [8]-[12], [16]-[17]. Compared with the cluster-based algorithms, the new proposed methods enhance the performance of the overlapping spike sorting. However, the mthods for the overlapping spike are not yet well done under some condisitons, e.g. sorting under the condition that different sorts of spikes have similar waveforms.

This paper introduces a sparse coding and a compressive sensing method for overlapping spike sorting. The sparse coding and compressive sensing are not new. Sparse coding could reduce the dimension and redundancy of observed data, and has been successfully applied to high-dimensional data classification with noise interference [18], [19]. Compressive sensing could map a signal to a low-dimensional space if the signal is compressible, and the signal could be reconstructed from a small of number of projections in the compressive signals [20]-[22].

In this paper, however, we adopt the sparse expression of the two methods to sort overlapping spikes, instead of signal dimensional reduction. Both of the introduced sparse coding and compressive sensing are based on a convolution model [18], [23], where the overlap of multiple spikes could be expressed as a sum of the respective spike convolutions with a sparse signal. Through the sparse signal vector, then, an overlapping spike could be sorted. Besides, a posteriori (MAP) estimate is used to optimize the sparse vector for overlapping spike sorting. In experiments, some synthetic and real data are used to verify the introduced methods, and the methods are compared with the existing methods. The experimental results show that the number of sorting errors in the introduced methods is less than that in the traditional methods under the condition of overlapping spikes with similar waveforms.

The rest of this paper is organized as follows. Section II describes related works. Section III gives a convolution model for overlapping spikes. Section IV introduces a sparse coding and compressive sensing method for overlapping spike sorting. In Section V, we provide experiment results to demonstrate the performance of the introduced methods. Finally, conclusions are drawn in Section VI.

II. Related work for overlapping spike sorting

The earliest spike sorting method is matched filtering [24], which can identify a neuron spike from signals collected by a single electrode. The idea of matched filtering is to match the observed signal to a neuronal spike template. If the matched errors are not beyond a threshold, the matched spike would be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

considered as the neuron's spike. Today, the matched filtering is still widely applied in single-cell electrophysiology [24]. When the observed signals are from multi-neuron signals collected by multi-electrodes, however, the method does not work well [25] because the observed signals may be from other neurons. In this case, the matched errors will occur and the threshold needs to be manual adjustment for better performance.

A cluster method is a popular spike sorting method, which works better in the case of multi-neuron spikes collected by multiple electrodes. The cluster method usually involves three steps [3]. First, segment a raw signal into some chips containing spikes via threshold parameters, such as absolute values [26], square values [4] and Teager energy [27], or some nonlinear operators [28]. Second, extract the eigenvalues of the segmented spikes [2], [4], [5]. Third, cluster the extracted eigenvalues via some clustering method, such as *k*-means [3], superparamagnetic clustering (SPC) [2] and other hybrid cluster-based method [6, 13-15]. However, the performance of the cluster method will degrade when the overlap of spikes occurs [7]. The reason for this is that the eigenvalue point of an overlapped spike may have large Euclid distances from all of the centers of clusters. This will make the point discarded or misclassified.

The misclassification for the overlapping spikes may lead to errors in some measurement results, e.g. neuronal firing rates [8], [25] and a correlation between a neuron and the activity of the neuron spike [29], [30]. With the development of multi-electrode array technology, especially, the activity of a large number of neuron spikes could be recorded simultaneously and the overlap of the spikes occurs frequently. Thus, sorting overlapping spikes correctly will become more important [16]. At present, there are several new proposed cluster-based methods to solve the overlapping spikes [8], [9]. Since the methods need to separate the overlapping spikes before sorting or greedily find the best fitting waveform for the overlap, their computational complexity increases with the number of overlapping neurons. As the number of neurons increases, their complexity would be unavailable.

Independent Component Analysis (ICA) is a good idea because it adopts blind separation to sort overlapping spikes [31]. Unfortunately, its accuracy in overlapping spike sorting is not high. Continuous Basis Pursuit (CBP) [16], [32] is also a method for overlapping spike sorting. Since CBP algorithm has lower computational complexity and higher sorting accuracy, the algorithm has become one of the best sorting algorithms for overlapping spikes in the past two years. However, CBP still has some problems. When the waveforms of overlapping spikes are similar, the penalty of a cost function in CBP will increase with the amplitude of the waveforms and produce performance degradation.

III. System Model

Before sorting spikes, we need to preprocess the raw voltage trace recorded by electrodes. The preprocessing steps are high-pass filtering, whitening and peaks detecting [8], [16].

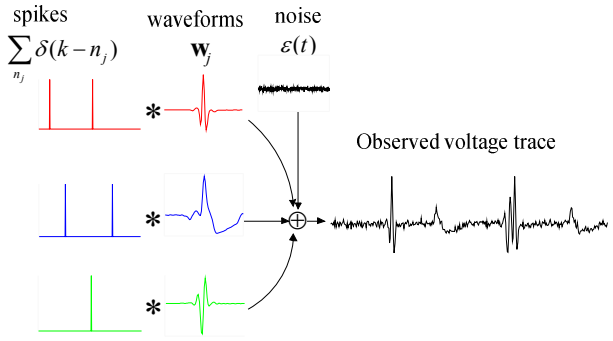


Fig. 1. Spikes signal system model where an overlapped spike is the sum of convolutions between several spike template signals and their respective time-shifted impulse functions.

After detecting the peaks, we find the time when the neuron fires an action potential. Then, the filtered and whiten voltage trace is segmented into some chips containing the peaks. The segmented spikes are just what we need to sort. If $v(m)$ is denoted as the potential amplitude of the *m*th sampling point in a segmented spike, it could be expressed as [8], [16]

$$v(m)=\sum_{j=1}^{J} w_j(m) * \delta(m-n_j)+\varepsilon(m) \qquad (1)$$

where

$m \in \{1, 2, \cdots, M\}$,
M denotes the length of the spike segment,
$w_j(m)$ denotes the potential amplitude of the *j*th neural spike template at the *m*th point,
$\delta(\cdot)$ denotes a unit impulse function,
$*$ represents a convolution,
J denotes the number of neurons,
$n_j \in \{-M+1, -M+2, \cdots, M\}$ denotes the time shifted,
$\varepsilon(m)$ is an additive noise.

Eq. (1) means that an overlapped spike can be expressed as the sum of convolutions between several spike template signals and their respective time-shifted impulse functions, shown in Fig. 1. Note that the spike template signals $w_j(m)$ in (1) could be from the center of clusters via methods, e.g. *k*-means. For spikes collected from multiple electrodes, besides, we could constitute them one-by-one into a sequence signal. Here, we only give the case of a single electrode for simplified expression.

IV. Algorithm

A. Sparse Coding and Compressive Sensing Method

As shown in last section, an overlapped spike is in fact the superposition of several time-shifted spike templates. Hence, the overlapped spike could be correctly sorted if we know which spikes are superposited. For this, we change (1) into a matrix form as

$$\mathbf{V}=\mathbf{W}\boldsymbol{\Delta}+\boldsymbol{\varepsilon} \qquad (2)$$

where

$$\mathbf{V}=[v(1), v(2), \cdots, v(M)]^{\mathrm{T}} \qquad (3)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

TABLE 1
ACTIVE-SET ALGORITHM FOR SPARSE CODING

Algorithm steps

Input: Segmented signal V
Known parameter: Toeplitz matrix $\mathbf{W}$.
Computation

$$\mathbf{H}_\Phi = \mathbf{W}_\Phi{}^{\mathrm{T}}\mathbf{W}_\Phi,\ \mathbf{g}_\Phi = -\mathbf{W}_\Phi{}^{\mathrm{T}}\mathbf{V}_\Phi,$$

where $\mathbf{W}_\Phi$ and $\mathbf{V}_\Phi$ are the corresponding Linear kernel mapping of $\mathbf{W}$ and $\mathbf{V}$, respectively.
Output: sparse vector $\hat{\mathbf{\Delta}}$.
Steps: 1.Initialization $\mathbf{X} = \left[\left(\mathbf{H}_\Phi\right)^{-1}\left(-\mathbf{g}_\Phi\right)\right]_+$, where $\mathbf{X} = [\mathbf{Y}]_+$ is defined that if $y_i > 0$, then $x_i = y_i$, otherwise $x_i = 0$. In addition, x_i represents the i-th elements of $\mathbf{X}$.
$\mathcal{R} = \{i | x_i = 0\}$;
$\mathcal{P} = \{i | x_i > 0\}$;
$\boldsymbol{\mu} = \mathbf{H}_\Phi\mathbf{X} + \mathbf{g}_\Phi$;
2. If $\mathbf{R} \neq 0$ and $\min_{i\in\mathcal{R}}(\mu_i) < -e$, where e is the convergence threshold, iteration from step 3 to step 8; otherwise, iteration completed and let $\mathbf{X} = \hat{\mathbf{\Delta}}$;
3. $j = \arg\min_{i\in\mathcal{R}}(\mu_i)$
$\mathcal{P} = \mathcal{P} + \{j\}$; $\mathcal{R} = \mathcal{R} - \{j\}$
$\mathbf{t}_\mathcal{P} = \left(\mathbf{H}_\mathcal{P}\right)^{-1}\left(-\mathbf{g}_\mathcal{P}\right)$; $\mathbf{t}_\mathcal{R} = \mathbf{0}$
4. If $\min \mathbf{t}_\mathcal{P} \le 0$, iteration from step 5 to step 6; otherwise, iteration completed and jumping to step 7;
5. $\alpha = \min x_i / (x_i - t_i)$, where $i \in \mathcal{P}, t_i \le 0$;
$\mathcal{K} = \arg\min x_i / (x_i - t_i)$, where $i \in \mathcal{P}, t_i \le 0$;
$\mathbf{X} = \mathbf{X} + \alpha(\mathbf{t} - \mathbf{X})$;
$\mathcal{P} = \mathcal{P} - \mathcal{K}$; $\mathcal{R} = \mathcal{R} + \mathcal{K}$;
$\mathbf{t}_\mathcal{P} = \left(\mathbf{H}_\mathcal{P}\right)^{-1}\left(-\mathbf{g}_\mathcal{P}\right)$; $\mathbf{t}_\mathcal{R} = \mathbf{0}$;
6. Return to step 4;
7. $\mathbf{X} = \mathbf{t}$; $\boldsymbol{\mu} = \mathbf{H}_\Phi\mathbf{X} + \mathbf{g}_\Phi$;
8. Return to step 2;

TABLE 2
BAYESIAN LAPLACE PRIOR ALGORITHM FOR COMPRESSIVE SENSING

Algorithm steps

Input: Segmented signal $\mathbf{V}$.
Known parameter: Toeplitz matrix $\mathbf{W}$.
Computation

$$p(\mathbf{X} | \mathbf{V}, \boldsymbol{\gamma}, \beta, \lambda) = N(\mathbf{X} | \boldsymbol{\mu}, \mathbf{\Sigma}) \quad \text{(T-1)}$$

where

$$\boldsymbol{\mu} = \mathbf{\Sigma}\beta\mathbf{W}^T\mathbf{V} \quad \text{(T-2)}$$
$$\mathbf{\Sigma} = \left[\beta\mathbf{W}^T\mathbf{W} + \mathbf{\Lambda}\right]^{-1} \quad \text{(T-3)}$$
$$\mathbf{\Lambda} = \mathrm{diag}(1/\gamma_i) \quad \text{(T-4)}$$
$$\lambda = (N - 1 + \theta/2)/(\textstyle\sum_i \gamma_i/2 + \theta/2) \quad \text{(T-5)}$$
$$\beta = \left(N/2 + a^\beta\right)\Big/\left(\left\langle\|\mathbf{V} - \mathbf{W}\mathbf{X}\|^2\right\rangle/2 + b^\beta\right) \quad \text{(T-6)}$$
$$\ln(\theta/2) + 1 - \psi(\theta/2) + \ln\lambda - \lambda = 0 \quad \text{(T-7)}$$
$$\mathbf{C} = \beta^{-1}\mathbf{I} + \textstyle\sum_i \gamma_i w_i w_i^T = \beta^{-1}\mathbf{I} + \sum_{i'\neq i}\gamma_{i'} w_{i'} w_{i'}^T + \gamma_i w_i w_i^T = \mathbf{C}_{-i} + \gamma_i w_i w_i^T \quad \text{(T-8)}$$
$$p_i = w_i^T\mathbf{C}_{-i}^{-1}w_i \quad \text{(T-9)}$$
$$q_i = w_i^T\mathbf{C}_{-i}^{-1}\mathbf{V} \quad \text{(T-10)}$$

$$\gamma_i = \begin{cases} \dfrac{-p_i(p_i + 2\lambda) + p_i\sqrt{A}}{2\lambda_i p_i^2} & \text{if } q_i^2 - p_i > \lambda \\ 0 & \text{otherwise} \end{cases} \quad \text{(T-11)}$$

$\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \cdots, \gamma_N]$
w_i represents the ith column of $\mathbf{W}$,
$\psi(\theta/2)$ is the derivative of $\ln\Gamma(\theta/2)$,
$A = (p_i + 2\lambda)^2 - 4\lambda(p_i - q_i^2 + \lambda)$.
Output: sparse vector $\hat{\mathbf{\Delta}}$.
Steps: 1. Initialization $\boldsymbol{\gamma} = \mathbf{0}$, $\lambda = 0$.
2. If greater than a maximum number or smaller than a threshold, iteration completed and let $\mathbf{X} = \hat{\mathbf{\Delta}}$; otherwise, iteration from step 3 to step 10;
3. Choose a γ_i, $i \in [1, N]$.
4. If $q_i^2 - p_i > \lambda$ and $\gamma_i = 0$, then add γ_i to the model.
5. If $q_i^2 - p_i > \lambda$ and $\gamma_i > 0$, then re-estimate γ_i via (T-11).
6. If $q_i^2 - p_i < \lambda$, the ith column w_i in the $\mathbf{W}$ is pruned and let $\gamma_i = 0$.
7. Update $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ via (T-2) and (T-3), respectively.
8. Update p_i and q_i via (T-9) and (T-10), respectively.
9. Update λ, β, θ from (T-5) to (T-7), respectively.
10. Return to step 2.

and $\boldsymbol{\varepsilon} = [\varepsilon(1), \varepsilon(2), \cdots, \varepsilon(M)]^1$. Since (1) represents convolutions between spike templates and impulse functions, the matrix $\mathbf{W}$ in (2) should be a Toeplitz matrix. Moreover, $\mathbf{\Delta}$ is a vector composed of the unit impulse function $\delta(\cdot)$. Hence, the vector $\mathbf{\Delta}$ should be a sparse vector. Here, we will find a solution for $\mathbf{\Delta}$ through the method of sparse coding and compressive sensing. From the solution for $\mathbf{\Delta}$, what sort of spikes constitute the overlapped spike could be known and the classification would be completed.

For this aim, we firstly establish the Toeplitz matrix $\mathbf{W}$ as

$$\mathbf{W} = [\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \cdots, \bar{\mathbf{W}}_J] \quad (4)$$

where,

$$\bar{\mathbf{W}}_j = [\mathbf{w}_j(-r_j), \mathbf{w}_j(-r_j + 1), \cdots, \mathbf{w}_j(s_j - 1)] \quad (5)$$

$$\mathbf{w}_j(m) = \begin{cases} \left[\mathbf{0}_{1\times m}, w_j(1), w_j(2), \cdots, w_j(M - m)\right]^{\mathrm{T}} \\ \quad \text{if } 0 \le m \le s_j - 1 \\ \left[w_j(-m + 1), w_j(-m + 2), \cdots, w_j(M), \mathbf{0}_{1\times(-m)}\right]^{\mathrm{T}} \\ \quad \text{if } -r_j \le m \le -1 \end{cases} \quad (6)$$

From (4-6), the matrix $\bar{\mathbf{W}}_j$ is composed of several column vectors $\mathbf{w}_j(m)$, $m = -r_j, -r_j + 1, \cdots, s_j - 1$, which is from the jth neuronal spike template $w_j(m)$, $m = 1, 2, \cdots, M$ through left or right time-shifted. Therefore, the Toeplitz matrix $\mathbf{W}$ could be obtained from J time-shifted neuronal spike templates. Besides, r_j and s_j represent the maximum left and right shifts, respectively. The choice for r_j and s_j is decided by the waveform amplitude of the spike templates and ensure $0 < r_j < M$, $0 < s_j < M$. Finally, the dimension of $\mathbf{W}$ is $M \times N$ if $u_j = r_j + s_j$, $N = \sum_{j=1} u_j$.

Next, we establish the sparse vector $\mathbf{\Delta}$. From (3-6), we have

$$\mathbf{\Delta} = [\boldsymbol{\delta}^{\mathrm{T}}(n_1), \boldsymbol{\delta}^{\mathrm{T}}(n_2), \cdots, \boldsymbol{\delta}^{\mathrm{T}}(n_J)]^{\mathrm{T}} \quad (7)$$

where

$$\boldsymbol{\delta}(n_j) = \begin{cases} [\mathbf{0}_{1\times(r_j + n_j)}, 1, \mathbf{0}_{1\times(s_j - 1 - n_j)}]^{\mathrm{T}} & \text{if } -r_j \le n_j \le s_j - 1. \\ \mathbf{0}_{u_j\times 1} & \text{otherwise.} \end{cases} \quad (8)$$

From (1) and (7-8), the impulse function $\delta(m - n_j)$ of the jth neuron spike is changed into the vector $\boldsymbol{\delta}(n_j)$, where only the $r_j + n_j + 1$-th coefficient is 1 and the others are all 0. If $\boldsymbol{\delta}(n_j)$ is an all-zero vector, the overlapped spike will not contain the jth neuron spike. In addition, $\mathbf{\Delta}$ should be an $N \times 1$ vector since $\boldsymbol{\delta}(n_j)$ is an $u_j \times 1$ vector.

From the analysis above, if we obtain the vector $\mathbf{\Delta}$ from an overlapping spike, we will be able to know what sorts of spikes constitute the overlapped spike and how many shifts the spikes have. Thus, the overlapped spike would be sorted successfully. For example, if $J = 3$, $u_1 = u_2 = u_3 = 5$ and $\mathbf{\Delta} = [01000\ \ 00000\ \ 00010]^{\mathrm{T}}$, we will know that the overlapped signal is composed of the first sort of spike left-shifted by one bit and the third one right-shifted by one bit, respectively.

Since an overlapped spike signal can be expressed as a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

TABLE 3
OVERLAPPING SPIKE SORTING ALGORITHM

Algorithm steps
1. A raw voltage trace is preprocessed, i.e. whitened, filtered, peaks detected and segmented into a number of signal chips containing the peaks.
2. The eigenvalues of the signal segments are extracted through PCA and clustered into J clusters through k-means. Then, the centers of the clusters are used as J spike templates $\mathbf{w}_j(0)$, $j=1, 2, \cdots, J$.
3. The templates $\mathbf{w}_j(0)$, $j=1, 2, \cdots, J$ time-shifted constitute a Toeplitz matrix $\mathbf{W}$ in (4).
4. A Sparse vector $\hat{\boldsymbol{\Delta}}$ is obtained via sparse coding in Table 1 or compressive sensing in Table 2.
5. An optimized sparse signals $\boldsymbol{\Delta}^*$ is estimated via MAP in (11), where a search scope is from a set Ω^N in (12);
6. Judge whether $\boldsymbol{\Delta}^*((j-1)u_j : ju_j)$ has a coefficient 1 or not. If it does, the segmented signal contains the spike of the j th neuron. Otherwise, the segment does not contain the spike.
7. Repeat the step 4 to 6 until all signal segments are sorted.

Toeplitz matrix $\mathbf{W}$ multiplied by an impulse function vector $\boldsymbol{\Delta}$, i.e. a dictionary multiplied by a sparse signal, a solution for $\boldsymbol{\Delta}$ will be changed into

$$\hat{\boldsymbol{\Delta}} = \arg\min_{\mathbf{X}} \left\{ k \left\| \mathbf{V} - \mathbf{W}\mathbf{X} \right\|_2^2 + \tau \left\| \mathbf{X} \right\|_1 \right\} \tag{9}$$

where k and τ are both scale coefficients. For the solution in (9), we have the following schemes.

(1) When the coefficients k=1/2, τ =0 and the constraint condition $\mathbf{X} \geq 0$, the problem is actually a non-negative least squares (NNLS) sparse coding model. Table 1 gives an active-set algorithm for the sparse model [18], [33], [34].

(2) When k=1, the equation is changed into a compressive sensing problem [20], [22], [23]. Table 2 gives a Laplace prior Bayesian algorithm [23] for the compressive sensing problem.

B. MAP for Sorting

If we obtain $\hat{\boldsymbol{\Delta}}$ via Table 1 or 2, its expected form should be very sparse. For example, when an overlapped spike is from three neurons, an expected $\hat{\boldsymbol{\Delta}}$ of the overlapped spike should have only three coefficients equal to 1 and the others all equal to 0. Likewise, $\hat{\boldsymbol{\Delta}}$ should have only one coefficient equal to 1 when the spike contains only one kind of spike. Due to noises and the Toeplitz matrix with not-full-column rank, however, $\hat{\boldsymbol{\Delta}}$ may have the following cases. First, the actual vector has more nonzero coefficients than the expected one. Second, a maximum coefficient may not be 1. Third, the location of the maximum coefficient 1 does not match the time shift of a spike template. To optimize the sparse vector, we use an MAP estimate as

$$\Delta^* = \arg\max_{\mathbf{X} \in \Omega^N} p\left(\mathbf{X} \middle| \mathbf{V}; \mathbf{W} \right) \tag{10}$$

If the noise $\boldsymbol{\varepsilon}$ in (2) is Gaussian with mean zero, taking logarithm for (10) will yield

$$\Delta^* = \arg\min_{\mathbf{X} \in \Omega^N} \left\| \mathbf{V} - \mathbf{W}\mathbf{X} \right\|_2^2 \tag{11}$$

The key of solving (11) is to determine the search set Ω^N. If we merely consider Ω^N as an N-dimensional vector space, the search scope will be very large. Here, the results via sparse coding or compressed sensing algorithm could significantly narrow the search scope. Using the form of (8), the narrowed set is given by

$$\Omega^N = \{ [\boldsymbol{\delta}(l_1), \boldsymbol{\delta}(l_2), \cdots, \boldsymbol{\delta}(l_J)]^{\mathrm{T}} \mid l_j \in \mathrm{L}_j, j = 1, 2, \cdots, J \}$$
$$\mathrm{L}_j = \{ \ddagger_j^0, \ddagger_j^1, \cdots, \ddagger_j^I \} \tag{12}$$

where $\ddagger_j^i$, i =1, 2, ⋯, I denotes the shift corresponding to the location of the i th largest coefficient in $\boldsymbol{\Delta}((j-1)u_j : ju_j)$. Here, $\hat{\boldsymbol{\Delta}}$ (m_1:m_2) represents a vector composed of the coefficients from the m_1-th to m_2-th in $\hat{\boldsymbol{\Delta}}$. And, we specify that if i =0, then $\boldsymbol{\delta}(l_j = \ddagger_j^0) = \mathbf{0}_{u_j \times 1}$ which corresponds that the overlapped spike vector $\mathbf{V}$ does not have the j th neuron spike. Here, we give an example for the search set. If $\boldsymbol{\Delta}((j-1)u_j : ju_j)$ is [0, 0.15, 1, 0.1, 0]$^{\mathrm{T}}$ where u_j=5, r_j=2, s_j=3, I =3, then L_j = {$\ddagger_j^0$, 0, -1, 1}.

The set Ω^N in (12) is based on the following reasons.

(1) The vector $\boldsymbol{\Delta}((j-1)u_j : ju_j)$ may contain nonzero coefficients even if the overlapped spike does not have the j th neuron spike. Therefore, the set Ω^N will have the element $\ddagger_j^0$.

(2) When an overlap spike has the j th neuron spike shifted by n_j, due to inferences or noises, the coefficient whose location corresponds to n_j in $\hat{\boldsymbol{\Delta}}((j-1)u_j : ju_j)$ should be at least the I th largest though not the 1st largest. Therefore, the cardinality of L_j will be I +1.

C. Computational Complexity and Algorithm Summary

It is concluded from (12) that the number of searches for MAP is $(I+1)^J$. When the number of neurons J is fixed, the computational complexity will increase with I. If $\ddagger_j$ is the number of nonzero coefficients in $\boldsymbol{\Delta}((j-1)u_j : ju_j)$, I could be chosen an integer between min $\{\ddagger_j\}$ and max$\{\ddagger_j\}$, j =1, 2, ⋯, J, i.e.

$$\min\{\ddagger_j\} \leq I \leq \max\{\ddagger_j\}, \; j = 1, 2, \cdots, J \tag{13}$$

Since $\hat{\boldsymbol{\Delta}}$ is a sparse vector, the search scope could not be large even though I chosen as max$\{\ddagger_j\}$. In practice, the value of I could be determined via how to optimize the performance of spike sorting.

Finally, we summarize the introduced spike sorting algorithm in Table 3.

V. EXPERIMENT

In this experiment, we use three groups of synthetic data and two groups of real data to verify our sparse coding and compressive sensing method. And, our methods are compared with the traditional k-means, CBP and SPC algorithm. In the five groups of data, there are three groups of data where the waveforms of spikes are more similar. Next, we will give the experimental results for the five groups of data, respectively.

A. Synthetic data

In this sub-section, three groups of synthetic data are used to verify the introduced methods and the traditional algorithms, and are denoted by C_Easy1_noise015, C_Difficult1_noise02 and C_Difficult2_noise01, where the latter two groups have similar waveforms [2]. Their waveforms are from real environment and noises are chosen to be able to simulate realistic background activity. Some parameters in this experiment and the algorithms are shown in Table 4. More parameters for SPC could be found from [2] and a software, WaveClus version 2.0 downloaded from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

TABLE 4
SOME PARAMETERS FOR SYNTHETIC DATA

Parameter	Value
Number of neurons	J=3
Number of channel	1
Filtered	Cut-off frequency at 250 Hz with a Butterworth high pass filter of order 50.
Whiten	Whitened in time
Peaks detected	Mid-point-window method[16] and a threshold is 6
Length of signal segment	M=81
k-means	
Number of principal component	4
Contribution rate of principal components	90%
Distance measure	Euclidean distance
Clustering repeated	25
Selection of initial centroid	3 centroid randomly selected
CBP algorithm	
Minimum length of a signal segment	81
Maximum length of a signal segment	1001
Thresholds for identifying three different neurons	0.8871, 0.7065 and 0.5258
The number of iterations	200
The variance of noise	1
Sparse coding	
Sparse coding method	Non-negative least squares
Prediction method	k-nearest neighbor
Kernel function	Linear
Sparsity threshold	10^{-4}
Compressive sensing	
Spike templates	Clustering centroid via *k*-means
λ	Laplace priors
A threshold for termination	10^{-8}
Maximum iterations	1000 times

https://vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm, and more parameters for CBP found in literature [16].

Fig. 2 (a) shows the waveforms of three neuron spikes in the synthetic data C_Easy1_noise015. Fig. 2 (b) is a two-dimensional clustering figure where feature 1 and 2 are eigenvalues extracted from PCA. From the figure, there are three clusters. And, a number of points are far away from all center points of the three clusters. It could be verified that the points are indeed for some overlapped spikes.

Table 5 gives sorting results for C_Easy1_noise015, C_Difficult1_noise02 and C_Difficult2_noise01 through five methods, sparse coding (denoted by SC), compressive sensing (denoted by CS), *k*-means, CBP and SPC, respectively. Except the parameters listed in Table 4, we specify some parameters for sparse coding and compressive sensing as follows. In sparse coding, a maximum left shift and right shift r_j, s_j of the Toeplitz matrix in (4-6) are set to $r_j = s_j$=42, j=1, 2, 3 for C_Easy1_noise015, and I in (27) is set to I=25. In compressive sensing, r_j and s_j are the same as in sparse coding, but I is set to I=1. For C_Difficult1_noise02 and C_Difficult2_noise01, sparse coding and compressive sensing have the similar values r_j s_j and I. We specify that, if the difference between spike time in experiment and ground truth is within 4ms, it will be concluded that the time in experiment and in ground truth match. When the true spike time is not found to match the experimental one, we regard it as "miss". When the experimental one does not match the true one, on the other hand, we regard it as "false positive". From the result of C_Easy1_noise015, the misses by SPC is the highest, arriving

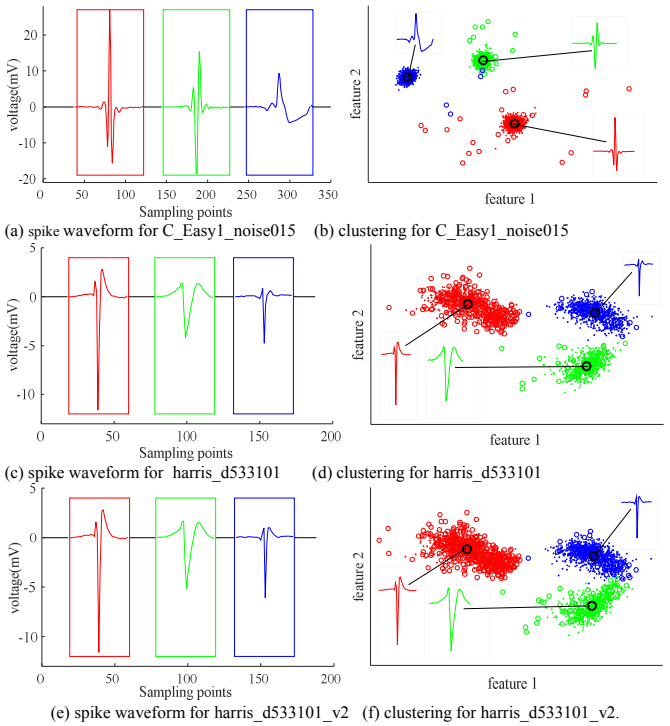


Fig. 2 Spike waveform and clustering figure

at 246 and those by *k*-means are also higher, arriving at 244. On the contrary, compressive sensing and CBP miss fewer spikes, both 13 and the fewest misses are for sparse coding. The reason why the misses by *k*-means and SPC are higher is that, clustering methods will make the eigenvalue point for an overlapped spike be far away from all centers of clusters. Note that, *k*-means and SPC use the same pre-processing method as CBP algorithm and our algorithm, such as whitening, filtering, and peak detection to eliminate the effect of different pre-processing. Finally, we can also see from the table that, the false positives by sparse coding and compressive sensing are both 10, slightly higher than 4 of CBP but not more than 1% of the total spikes. If we define the number of errors as a sum of misses and false positives, however, the number of errors by sparse coding is 17, the same as CBP.

For the results of C_Difficult1_noise02 in Table 5, sparse coding's false positives and classification errors are both fewer than CBP, and compressive sensing's misses, false positives and classifications errors are all fewer than CBP. In addition, sparse coding and compressive sensing's misses and classification errors are fewer than *k*-means and SPC. In Fig.3, we give the results of sorting detection (SD), which is defined as $SD(\%) = 100 * K/T$ where K is the number of successfully sorted spikes and T is the total number of spikes fired by the intracellular recorded neuron. From the figure, compressive sensing's *SD* for C_Difficult1_noise02 is higher than sparse coding, CBP, *k*-means and SPC. For the results of C_Difficult2_noise01 in Table 5, although sparse coding's misses and false positives are slightly higher than CBP, compressive sensing's misses and false positives are the same as CBP. Similarly, sparse coding and compressive sensing's misses are lower than *k*-means and SPC. From the results of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

TABLE 5
RESULTS OF SORTING SPIKE IN FIVE GROUPS OF DATA BY FIVE METHODS

Experimental data	SC		CS		*k*-means		CBP		SPC		Number of Spikes
	Misses	FP	Misses	FP	Misses	FP	Misses	FP	Misses	FP	
C_Easy1_noise015	7\0.94	10	13\1.7	10	244\32.8	2	13\1.7	4	246\33.0	4	3477//744/21.4
harris_d533101	2	27	3	27	29	23	2	24	30	33	621
harris_d533101_v2	48	53	48	54	83	48	121	26	90	62	777
C_Difficult1_noise02	158\20.5	64	54/7	52	229\29.9	21	144\18.8	107	231\30.1	9	3414//767/22.5
C_Difficult2_noise01	12\1.6	22	7\0.93	5	204\27.1	21	7\0.93	3	206\27.3	18	3462//754/21.8
Average	45	35	25	30	158	23	57	33	161	25	2350

Note: SC, CS and FP are denoted by sparse coding, compressive sensing and false positives, respectively. The notaitons of \n, /n and //n is expressed as the ratio of misses to the overlapping spikes, the ratio of overlapping spikes to total spikes and the number of overlapping spikes, respectively.

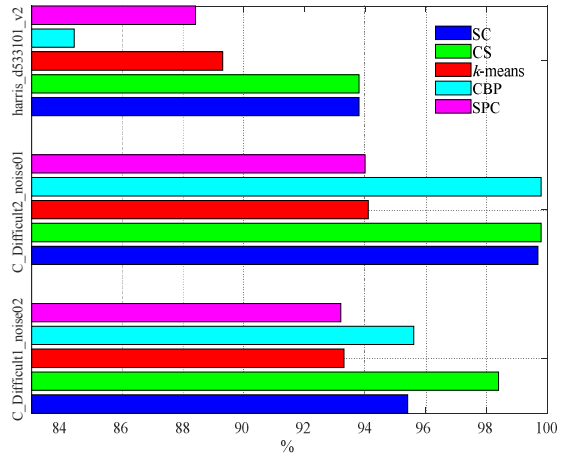


Fig. 3 *SD* (%) for three groups of similar-waveform data, harris_d533101_v2, C_Difficult1_noise02 and C_Difficult2_noise01

C_Difficult2_noise01 in Fig.3, sparse coding and compressive sensing's *SD* is similar to CBP, and higher than *k*-means and SPC. For the three groups of synthetic data, we also give the number of overlapped spikes and the ratio of misses to the overlapping (RMO) spikes in Table 5. From the table, Compressive Sensing's average RMO is the slightly lower than CBP and Sparse coding's average RMO is slightly higher than CBP, but both of them are much lower than *k*-means and SPC.

Next, we analyze the impact of several key parameters on the performance of sparse coding and compressive sensing. Fig. 4 shows results for the impact of I on sparse coding and compressive sensing, respectively. From C_Easy1_noise015 in sparse coding, we can see that the errors of spikes decrease with the value of I until I=25. When $I \geq 25$, the errors are almost unchanged, always 17. From C_Easy1_noise015 in compressive sensing, the errors are the least when I=1 and increase with the value of I until I=13. Then, the errors are almost unchanged. The results show that, the performance of compressive sensing cannot be enhanced even if I increasing, but the performance of sparse coding can be enhanced. For the data C_Difficult1_noise02 and C_Difficult2_noise01, the value of I has similar impact on sparse coding and compressive sensing.

Fig. 5 shows the impact of r_j and s_j on the performance of sparse coding and compressive sensing, respectively. From

TABLE 6
SOME PARAMETER FOR DATA HARRIS_D533101

Parameter	**Value**
Number of channels	4
Whiten	Whitened in time and space.
Length of signal segment	M=41
k-means	
No. of principal components	74
CBP	
Thresholds for three neurons	0.8194, 0.4806 and 0

C_Easy1_noise015 in sparse coding, the errors decrease from 181 to 17, with the value of r_j and s_j from about 50 to 42. Then, the errors will increase with r_j and s_j from about 42 to 18. From C_Easy1_noise015 in compressive sensing, the errors decrease from 47 to 23, with the value of r_j and s_j from about 50 to 40. Then, the errors will increase rapidly when r_j and s_j are below about 40. In addition, we can also see that the misses and false positives cannot arrive at the least value simultaneously whatever r_j and s_j are. Thus, the choice for the value of r_j and s_j may be only a compromise between the two performances. For the data C_Difficult1_noise02 and C_Difficult2_noise01, the values of r_j and s_j have similar impact on the performance of sparse coding and compressive sensing.

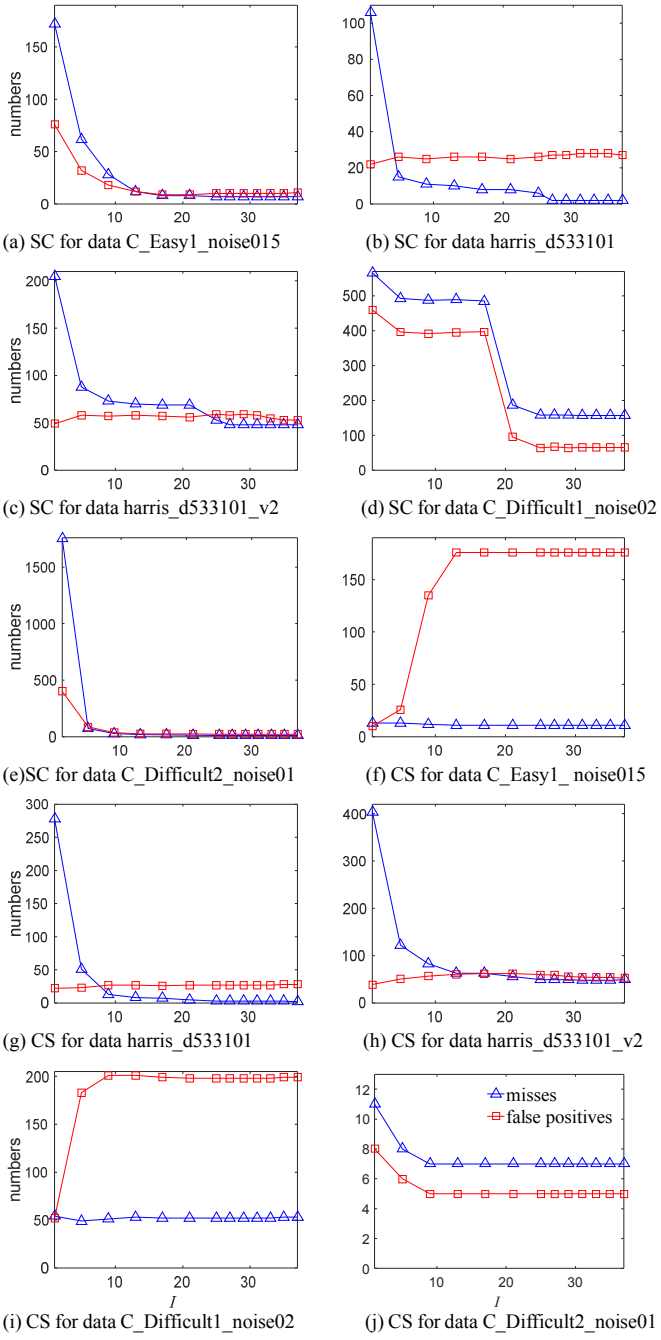
B. Electrode recording data in rat hippocampus

In this sub-section, a group of real data are used to verify the introduced algorithms and the traditional algorithms. The data denoted by harris_d533101 are from CA1 region in anesthetized rat hippocampus [5]. The data recorded by electrodes are from the intracellular and extracellular, respectively. The intracellular is used as ground truth and the extracellular is used for test. Most parameters for harris_d533101 are the same as in Table 4 except some ones listed in Table 6.

Fig. 2 (c) shows the waveform of three neuron spikes in this data. It is seen from the figure that the peaks and shapes of the three spikes are also different. Fig. 2 (d) is a two-dimensional clustering figure for the data. From the figure, there are also three clusters and the points for overlapped spikes are far away from all centers of the clusters.

Table 5 gives the results for sorting harris_d533101. There are some parameters not listed in Tables, which is specified as follows. A maximum left shift and right shift in sparse coding

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering



(a) SC for data C_Easy1_noise015 (b) SC for data harris_d533101
(c) SC for data harris_d533101_v2 (d) SC for data C_Difficult1_noise02
(e)SC for data C_Difficult2_noise01 (f) CS for data C_Easy1_ noise015
(g) CS for data harris_d533101 (h) CS for data harris_d533101_v2
(i) CS for data C_Difficult1_noise02 (j) CS for data C_Difficult2_noise01

Fig. 4. Impact of I on the performance of sparse coding and compressive sensing.

are r_j =20, s_j =19, j=1, 2, 3 and I=27. The parameters r_j and s_j in compressive sensing are consistent with sparse coding, but I=25. For this multi-channel data, we use the SPC public software to handle single channel data. Then, we choose the best performance of a channel from multiple channels in harris_d533101 as final experimental results. We can see from this group of data that, the misses by SPC is the highest, arriving at 30 and those by *k*-means are also higher, arriving at 29. On the contrary, misses by compressive sensing is only 3, and both of sparse coding and CBP are 2. The reason for higher misses by *k*-means and SPC is still that they are cluster-based methods. In addition, sparse coding and compressive sensing have the same number of false positives, 27 slightly higher than 24 of CBP.

TABLE 7
SOME PARAMETERS FOR DATA HARRIS_D533101_V2

Parameter	Value
k-means	
Number of principal components	77
CBP	
Thresholds for three neurons	0.8194, 0.7516 and 0

Next, we analyze several parameters having the impact on the performance in this group of data. Fig. 4 shows the impact of I on the performance of sparse coding and compressive sensing, respectively. We can see from harris_d533101 in sparse coding that when I =27, the number of errors is the least, only 29. When I>27, the errors tend to be stable. From compressive sensing, the number of errors is the least when I =25, only 30. Then, the errors also tend to be stable. The results for this group of real data show that, the errors can be reduced through increasing I reasonably. However, an excessive large I could not always make the errors decrease.

Fig. 5 gives the impact of r_j and s_j on sparse coding and compressive sensing, respectively. From harris_d533101 in sparse coding, the errors decrease from about 900 to 29 of a minimum, with r_j and s_j from 50 to about 21. Then, the errors will be almost unchanged even if r_j and s_j decrease. From harris_d533101 in the compressive sensing, the errors decrease from about 700 to 30 of a minimum, with r_j and s_j from 50 to about 21. Then, the errors will be almost unchanged even if r_j and s_j decrease. Similar with the synthetic data above, the results indicate that both of misses and false positives cannot arrive at a minimum simultaneously.

C. *Electrode recording data in locust*

In this sub-section, another real data are used to verify the introduced algorithms and the traditional algorithms in this section. The data denoted by harris_d533101_v2 are from locust in vivo [10] and also have the intracellular and the extracellular data. Most parameters for harris_d533101 _v2 are consistent with of Table 4 and 6 except some ones listed in Table 7.

Fig. 2 shows the waveforms of three neural spikes. From the figure, we can see that the shapes of the second and the third one are some similar except their widths. Fig. 2(f) shows a two-dimensional clustering figure for this data. There are also three clusters and some points far away from all of the cluster centers. From the figure, moreover, the cluster of the second neuron is closer to that of the third one due to their similar shapes[2, 16]. Hence, the points in the intersection of the two clusters may be difficult to be sorted correctly.

Table 5 gives the results for sorting harris_d533101_v2. Some parameters unlisted in tables are as follows. The values of r_j , s_j and I in sparse coding are set to r_j =19, s_j =18, j=1, 2, 3, and I =35, respectively. The parameters r_j , s_j in compressive sensing are the same as sparse coding but I=33. It is worth noting that the errors are significantly increased due to the similar waveforms. For SPC, we still chose a channel data performing the best. The misses by SPC are the highest, arrive at 90. And, the misses by CBP are 121, even higher than 83 of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSRE.2018.2848463, IEEE Transactions on Neural Systems and Rehabilitation Engineering

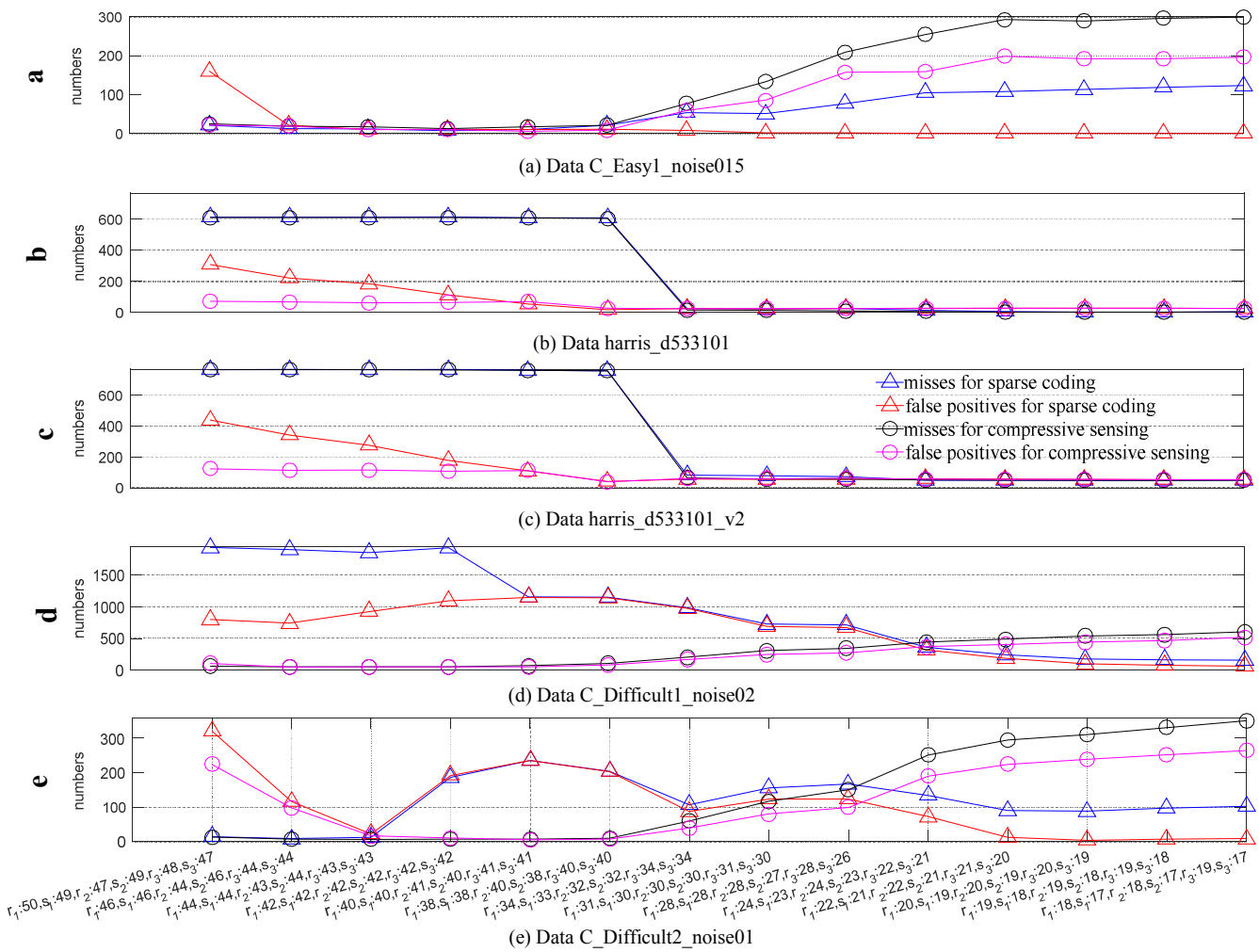


Fig. 5 Impact of maximum right shift and left shift on the performance of sparse coding and compressive sensing.

k-means. On the contrary, misses by sparse coding and compressive sensing are the least, both 48. The reason why CBP performs worse is the similar waveforms of the two neuron spikes. On the other hand, the false positives by sparse coding or compressed sensing are some higher than the other three methods, but the sum of misses and positives are lower. Thus, from the three groups of similar-waveform data, harris_d533101_v2 C_Difficult1_noise02 and C_Difficult2_ noise01, the results of Table 5 and Fig. 3 show that two groups of data display higher classification performance of our algorithm and the other group display the same performance.

Next, we give the impact of several parameters on the performance. Fig. 4 shows the change of *I* on performance of sparse coding and compressive sensing, respectively. From harris_d533101_v2 in sparse coding, the number of errors, 101 is the least when *I*=35. From harris_d533101_v2 in compressive sensing, likewise, the number of errors, 102 is the least when *I* =33. For both of the two algorithms, it is seen that the errors tend to be stable even *I* increase from 27 to 37. The results show that an excessive large *I* could not always make the errors decrease. This is also consistent with the harris_d533101 data.

Fig. 5 shows the impact of r_j and s_j on the performance of sparse coding and compressive sensing, respectively. From harris_d533101_v2 in sparse coding, the errors decrease from about 1200 to about 100 of a minimum, with r_j and s_j from about 50 to about 20. From harris_d533101_v2 in compressive sensing, the errors decrease from about 800 to about 100 of a minimum, with r_j and s_j from about 50 to about 20. Similar with the previous two data, both of misses and false positives cannot arrive at a minimum simultaneously.

D. Analysis for some parameters

From the results above, the value of *I* has an impact on the performance of sparse coding and compressive sensing, and an excessive large or an excessive small *I* cannot produce an optimal sorting performance. The former will increase the computational complexity and could not always have better spike identification. On the other hand, the latter will result in more sorting errors.

In addition, the establishment of a dictionary i.e. a measurement Toeplitz matrix also has an impact on the results. Likewise, excessive large or small shifts, r_j and s_j are not good candidates for sorting spikes. All of the groups of data

could indicate this point.

VI. Conclusion

This paper intorduces a sparse coding and a compressive sensing method to solve a problem for overlapped spike sorting. The introduced methods start from a convolutional model, where a Toeplitz matrix is established from several spike templates time-shifted. The spike templates could be obtained via the centroids of clusters in traditional *k*-means algorithm. Through a dictionary i.e. the Toeplitz matrix, sparse coding or compressive sensing yields a sparse vector. Further, MAP will optimize the sparse vector, through which an overlapped spike sorting is completed.

In the experiments above, three groups of synthetic and two groups of real data are used to testify the introduced and traditional methods. From the experimental results, the introduced method's average sorting detection is nearly 4% higher than traditional methods for the three groups of data with similar spike waveforms. On the other hand, the introduced and traditional methods have similar sorting performance for the other two groups of data with different waveforms. This indicates that our algorithm has some advantages in classification when the waveforms of neuron spikes are similar.

Although our methods have better sorting performance for the overlapped spikes with similar waveforms, some improvement needs to be done. For example, when a Toeplitz matrix is established, we need to choose a maximum right shift and left shift which have an important impact on the performance of sorting spike. How to choose reasonable values will be a key technique. One of feasible solutions is that, we first choose a group of training data and then try several values of the maximum shift in the training data. Thus, each shift value will have a classification result, and the value of shift corresponding to the best classification result would be an optimal value.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant No. 61762093, The17th batches of Young and Middle-aged Leaders in Academic and Technical Reserved Talents Project of Yunnan Province under Grant No. 2014HB019, The Program for Innovative Research Team (in Science and Technology) in University of Yunnan Province.

References

[1] H. G. Rey, M. Ahmadi, and R. Q. Quiroga, "Single trial analysis of field potentials in perception, learning and memory," *Current Opinion in Neurobiology*, vol. 31, pp. 148-155, April. 2015.

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

[3] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network Computation in Neural Systems*, vol. 9, no. 4, pp. 53–78, Jul. 1998.

[4] U. Rutishauser, E. M. Schuman, A. N. Mamelak, "Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo," *Journal of Neuroscience Methods*, vol. 154, no. 1-2, pp. 204–224, June. 2006.

[5] K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, G. Buzsaki, "Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements," *Journal of Neurophysiology*, vol. 84, no. 1, pp. 401–414, July. 2000.

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

[7] C. Pouzat, O. Mazor, G. Laurent, "Using noise signature to optimize spike sorting and to assess neuronal classification quality," *Journal of Neuroscience Methods*, vol. 122, no. 1, pp. 43–57, Dec. 2002.

[8] J. W. Pillow, J. Shlens, E. J. Chichilsnisky, E. P. Simoncelli, "A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings," *PLoS ONE*, vol. 8, no. 5, article e622123, May. 2013.

[9] B. Chen, D. E Carlson, L. Carin, "On the analysis of multi-channel neural spike data," *Advances in Neural Information Processing Systems 24(NIPS 2011)*. Cambridge, MA, 2011, pp. 936-944.

[10] M. Wehr, J. S. Pezaris, M. Sahani, "Simultaneous paired intracellular and tetrode recordings for evaluating the performance of spike sorting algorithms," *Neurocomputing*, vol. 26–27, pp. 1061–1068, June. 1999.

[11] D. Khodagholy, J. N. Gelinas, T. Thesen, *et al*., "NeuroGrid: recording action potentials from the surface of the brain," *Nature Neuroscience*, vol. 18, no. 2, pp. 310-315, Dec. 2014.

[12] H. G. Rey, C. Pedreira and R. Q. Quiroga, "Past, present and future of spike sorting techniques," *Brain Research Bulletin*," vol. 119, no. Part B, pp. 106-117, Oct. 2015.

[13] Tiganj Z, Mboup M. A non-parametric method for automatic neural spike clustering based on the non-uniform distribution of the data [J]. *Journal of neural engineering*, 2011, 8(6): 066014.

[14] Ghanbari Y, Spence L, Papamichalis P. A graph-Laplacian-based feature extraction algorithm for neural spike sorting[C]//*Engineering in Medicine and Biology Society*, 2009. EMBC 2009. Annual International Conference of the IEEE. IEEE, 2009: 3142-3145.

[15] Chah E, Hok V, Della-Chiesa A, et al. Automated spike sorting algorithm based on Laplacian eigenmaps and k-means clustering [J]. *Journal of neural engineering*, 2011, 8(1): 016006.

[16] C. Ekanadham, D. Tranchina, E. P. Simoncelli, "A unified framework and method for automatic neural spike identification," *Journal of Neuroscience Methods*, vol. 222, pp. 47-55, Jan. 2014.

[17] F. Franke, M. Natora, C. Boucsein, M. H. Munk, K. Obermayer, "An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes," *Journal of Computat-ional Neuroscience*, vol. 29, no. 1-2, pp. 127–148, Aug. 2010.

[18] Li. Y, A. Ngom, "Sparse representation approaches for the classification of high-dimensional biological data," *BMC Systems Biology*, vol. 7, no. 4, pp. S6, Oct. 2013.

[19] Yi S, Lai Z, He Z, et al. Joint sparse principal component analysis [J]. *Pattern Recognition*, 2017, 61: 524-536.

[20] Yang. J, Liao. X, Yuan. X, "Compressive Sensing by Learning a Gaussian Mixture Model From Measurements," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 106-119, Oct. 2015.

[21] B. Adcock, A. C. Hansen, "Generalized Sampling and Infinite-Dimensi-onal Compressed Sensing," *Foundations of Computational Mathematics*, vol. 16, no. 5, pp. 1263-1323, Oct. 2016.

[22] K. Arai, C. Belthangady, H. Zhang, *et al.*, "Fourier magnetic imaging with nanoscale resolution and compressed sensing speed-up using electronic spins in diamond," *Nature Nanotechnology*, vol. 10, no. 10, pp. 859-864, Aug. 2015.

[23] S. D. Babacan, R. Molina, A. K. Katsaggelos, "Bayesian Compressive Sensing Using Laplace Priors," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 53-63, Jan. 2010.

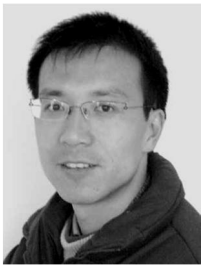
[24] G. Turin, "An introduction to matched filters. Information Theory," *IRE Transactions on Information Theory*, vol. 6, no. 3, pp. 311–329, July. 1960.

[25] I. B. Gad, Y. Ritov, E. Vaadia, H. Bergman, "Failure in identification of overlapping spikes from multiple neuron activity causes artificial correlations," *Journal of neuroscience methods*, vol. 107, no. 1-2, pp. 1–13, May. 2001.

[26] I. Obeid, P. D. Wolf, "Evaluation of spike-detection algorithms for a brain–machine interface application," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 905–911, July. 2004.

[27] J. H. Choi, H. K. Jung, T. Kim,"A new action potential detector using the MTEO and its effects on spike sorting systems at low signal-to-noise ratios," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 4, pp. 738–46, Apr. 2006.

[28] S. P. Rebrik, B. D. Wright, A. A. Emondi, K. D. Miller, "Cross-channel correlations in tetrode recordings: implications for spike-sorting," *Neurocomputing*, vol. 26–27, pp. 1033–1038, June. 1999.
[29] R. Rosenbaum, M. A. smith, A. Kohn, *et al.*, "The spatial structure of correlated neuronal variability," *Nature Neuroscience*, vol. 20, no. 1, Jan. 2017.
[30] G. Vinci, V. Ventura, M. A. Smith, *et al.*, "Separating Spike Count Correlation from Firing Rate Correlation," *Neural Computation*, vol. 28, no. 5, pp. 849-881, May. 2016.
[31] Tiganj Z, Mboup M. Neural spike sorting using iterative ICA and a deflation-based approach [J]. *Journal of neural engineering*, 2012, 9(6): 066002.
[32] C. Ekanadham, D. Tranchina, E. P. Simoncelli,"Recovery of sparse translation-invariant signals with continuous basis pursuit," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4735–4744, Oct. 2011.
[33] A. Bemporad, "A Quadratic Programming Algorithm Based on Nonne-gative Least Squares With Applications to Embedded Model Predictive Control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1111 – 1116, July. 2015.
[34] He Z, Yi S, Cheung Y M, et al. Robust object tracking via key patch sparse representation [J]. *IEEE transactions on cybernetics*, 2017, 47(2): 354-364.
[35] A. Calabrese, L. Paninski, "Kalman filter mixture model for spike sorting of non-stationary data," *Journal of Neuroscience Methods*, vol. 196, no. 1, pp. 159–169, Mar. 2011.



Haifeng Wu received the M.S. degree in electrical engineering from Yunnan University, Kunming, China, in 2004, and the Ph.D. degree in electrical engineering from Sun Yat-Sen University, Guangzhou, China, in 2007. He is currently an professor at the Department of Information Engineering at the Yunnan Minzu University. Prior to that, he was a postdoctoral scholar at the Kunchuan Institute of Technology from 2007 to 2009. His research interests include machine learning, neural signal processing and mobile communications.

Kai Yang is now pursuing the M.S. degree in electrical engineering from Yunnan University, Kunming, China. His interests include neural system and machine learning.

Yu Zeng received the M.S. degree in electrical engineering from Yunnan University, Kunming, China, in 2006. She is currently an assistant professor at the Department of Information Engineering at the Yunnan Minzu University. Prior to that, she was an electrical engineer in Kunming Institute of Physics from 2006 to 2009. Her research interests include wireless network and mobile communications.