Contents lists available at SciVerse ScienceDirect

Journal of Neuroscience Methods

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

Computational Neuroscience

# Performance comparison of extracellular spike sorting algorithms for single-channel recordings

Jiri Wild[a,*], Zoltan Prekopcsak[c], Tomas Sieger[a,b], Daniel Novak[a,**], Robert Jech[b]

[a] *Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Karlovo nam. 13, 121 35 Praha 2, Czech Republic*
[b] *Department of Neurology, 1st Faculty of Medicine and General Teaching Hospital, Charles University in Prague, Katerinska 30, 128 21 Praha 2, Czech Republic*
[c] *Department of Telecommunications and Media Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Magyar Tudosok Korutja 2, H-1117 Budapest, Hungary*

A R T I C L E  I N F O

*Article history:*
Received 24 March 2011
Received in revised form 14 October 2011
Accepted 17 October 2011

*Keywords:*
Spike sorting
Action potential
WaveClus
OSort
MClust
KlustaKwik
Mutual information
Micro-recording
Neuronal activity

A B S T R A C T

Proper classification of action potentials from extracellular recordings is essential for making an accurate study of neuronal behavior. Many spike sorting algorithms have been presented in the technical literature. However, no comparative analysis has hitherto been performed. In our study, three widely-used publicly-available spike sorting algorithms (WaveClus, KlustaKwik, OSort) were compared with regard to their parameter settings. The algorithms were evaluated using 112 artificial signals (publicly available online) with 2–9 different neurons and varying noise levels between 0.00 and 0.60. An optimization technique based on Adjusted Mutual Information was employed to find near-optimal parameter settings for a given artificial signal and algorithm. All three algorithms performed significantly better ($p < 0.01$) with optimized parameters than with the default ones. WaveClus was the most accurate spike sorting algorithm, receiving the best evaluation score for 60% of all signals. OSort operated at almost five times the speed of the other algorithms. In terms of accuracy, OSort performed significantly less well ($p < 0.01$) than WaveClus for signals with a noise level in the range 0.15–0.30. KlustaKwik achieved similar scores to WaveClus for signals with low noise level 0.00–0.15 and was worse otherwise. In conclusion, none of the three compared algorithms was optimal in general. The accuracy of the algorithms depended on proper choice of the algorithm parameters and also on specific properties of the examined signal.

## 1. Introduction

Classifying neuronal action potentials is a technical challenge that is a prerequisite for studying many types of brain function. Accurate detection of the activity of individual neurons can be difficult to achieve due to the large amount of background noise and the complexity in distinguishing the action potentials of one neuron from others. Even if the activity of several neurons is recorded with only a single electrode, spike sorting allows the researcher to measure the activity of the individual neurons separately. Although there are many spike sorting software packages (including commercial packages), we are not aware of any objective comparison of them that discusses adjustments to their parameters and their impact on spike sorting accuracy.

### 1.1. Spike sorting algorithms

Most unsupervised spike sorting algorithms employ three principal steps (Fig. 1). In the first step, spikes are detected with an automatic spike detection method. In the second step, a set of features is extracted from each spike – principal component analysis (PCA) in Adamos et al. (2008) or the wavelet transform (Quiroga et al., 2004) are usually used in this step. Finally, the spikes represented by their features are assigned to different neurons by an unsupervised learning algorithm (e.g., a clustering algorithm). We should mention that these steps are sometimes combined (Franke et al., 2009; Herbst et al., 2008), but most spike sorting algorithms handle the three steps independently.

We focus on stages 2 and 3, as there are already a number of comparative studies in the field of spike detection (Lewicki, 1998; Adamos et al., 2008; Gibson et al., 2008), and because the studied spike sorting algorithms are modular, thus allowing the researcher to choose freely which spike detection algorithm to use. The spike detection part was omitted by providing the algorithms with reference spike times.

The idea of recording multiple neurons and then grouping the action potentials by the source neuron is not new. It was first

* Corresponding author. Tel.: +420 224357666.
** Principal corresponding author.
*E-mail addresses:* wildjiri@fel.cvut.cz, jirka.wild@gmail.com (J. Wild), prekopcsak@tmit.bme.hu (Z. Prekopcsak), siegetom@fel.cvut.cz (T. Sieger), xnovakd1@labe.felk.cvut.cz (D. Novak), jech@cesnet.cz (R. Jech).

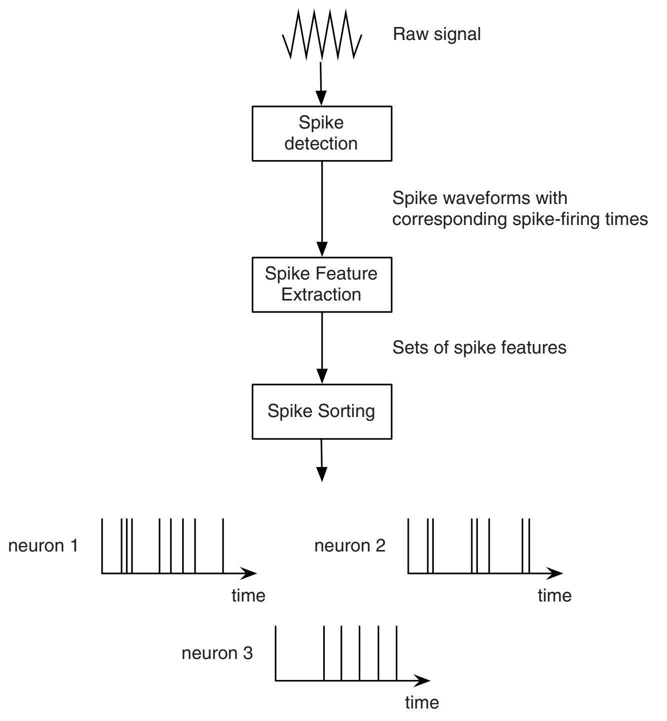

**Fig. 1.** Three principal stages of unsupervised spike sorting algorithms.

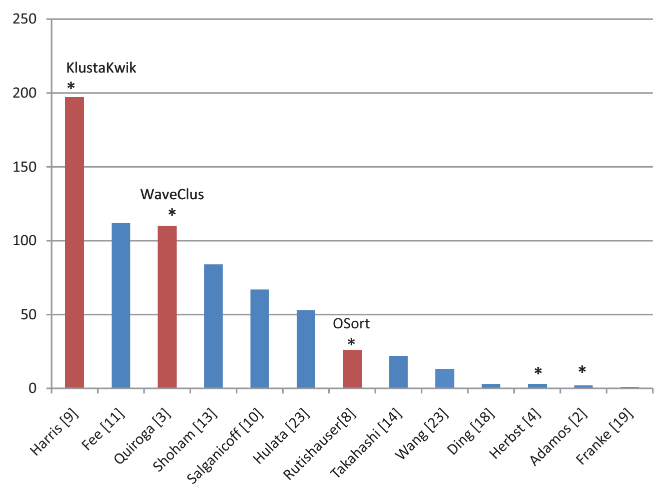

**Fig. 2.** Citation histogram of spike sorting algorithms as of January 2011. The approaches are tagged by an asterisk if the source code is available.

proposed in the 1960s (Gerstein and Clark, 1964), and since then numerous approaches to the problem have been developed.

Given a lower-dimensional representation of the spikes and disregarding the times at which the spikes occurred, the spike sorting problem reduces to a clustering problem. Therefore, most of the better known clustering algorithms have been applied to spike sorting: k-means clustering (Salganicoff et al., 1998), hierarchical clustering (Fee et al., 1996), superparamagnetic clustering (Quiroga et al., 2004), as well as mixtures of Gaussians (Sahani, 1999) and mixtures of t-distributions (Shoham et al., 2003). The method used in Fee et al. (1996) grouped multiple classes according to whether the interspike interval histogram of the group showed a significant number of spikes in the refractory period.

Takahashi et al. (2003a,b) combined independent component analysis (ICA) and the efficiency of an ordinary spike sorting technique (k-means clustering) to solve spike overlapping and nonstationarity problems of tetrode recordings with no limitation on the number of single neurons to be separated. Adamos et al. (2010) attempted to resolve overlapping spikes by introducing a hybrid scheme that combines the robust representation of spike waveforms to facilitate the reliable identification of contributing neurons with efficient data learning to enable the precise decomposition of coactivations.

Fee et al. (1997) described a procedure for efficiently sorting spikes in the presence of noise that is anisotropic, i.e., dominated by particular frequencies, and whose amplitude distribution may be non-Gaussian, such as occurs when spike waveforms are a function of the interspike interval. Support vector machines were used in Ding and Yuan (2008) to solve the superposition spike problem.

Herbst et al. (2008) combined the spike detection and classification steps into a single computational procedure using a Hidden Markov Model framework. Detection and classification was also merged in Franke et al. (2009), where a method of linear filters was inspected to find a new representation of the data and to optimally enhance the signal-to-noise ratio. By incorporating direct feedback, the algorithm adapted to nonstationary data. Delescluse and Pouzat (2006) used Markov chain Monte Carlo in order to estimate and make use of the firing statistics as well as the spike amplitude dynamics of the Purkinje cells. Online spike-sorting approaches suitable for HW implementation were addressed in Gibson et al. (2010) and Rutishauser (2006). Adamos et al. (2008) performed a comparative study focused on PCA using synthetic data on which correlated and white Gaussian noise processes are superimposed, and the KlustaKwik (Harris, 2000) clustering approach was used. Wang et al. (2006), proposed a robust approach employing an automatic overlap decomposition technique based on the relaxation algorithm that required simple fast Fourier transforms. Hulata et al. (2002) used a simple k-means technique for spike sorting while applying the wavelet packets decomposition framework in an extraction step.

The following approaches dealt with the quality of the spike sorting process. Schmitzer-Torbert et al. (2005) introduced two measures: L-ratio and Isolation Distance. The two measures quantified how well separated the spikes of one cluster were from other spikes. Joshua et al. (2007) described the isolation score, which measured the overlap between the noise (non-spike) and the spike clusters. The measure of Tankus et al. (2009) was based on visual features of the spike waveform and an automatic adaptive algorithm that learned the classification by a given human and could apply similar visual characteristics for classifying new data.

### 1.2. Comparative scheme

This paper describes a comparative analysis od the three most cited spike-sorting approaches with a publicly available source-code: WaveClus (Quiroga et al., 2004), OSort (Rutishauser, 2006) and KlustaKwik (Harris, 2000). The citation index was used as a measure for selecting the algorithm – see Fig. 2. Emphasis was put on involving one algorithm (Rutishauser, 2006) that can be used for real-time analysis.

The papers on WaveClus and KlustaKwik did not make direct comparisons with any other spike sorting method. They merely made comparisons between different versions of the same algorithm. OSort was compared with both methods, but from the perspective of online spike sorting (Rutishauser, 2006). We are convinced there is a need to evaluate them within a common framework, in order to determine which one to use for a specific task.

Lewicki (1998) presented an extensive review on spike sorting in 1998, but did not include any quantitative experiments, and dozens of new algorithms have been proposed since that review appeared. Gibson et al. (2008) compared several spike detection and feature extraction methods, but they did not include a

**Table 1**
Summary of the properties of each spike sorting algorithm.

| | WaveClus | KlustaKwik | OSort |
|---|---|---|---|
| Features | Wavelet transform | PCA | Raw data points |
| Clustering method | Superparamagnetic clustering | AutoClass | Template matching |
| User-tunable parameters | 20 | 10 | 2 |
| Real-time use | No | No | Yes |
| Open source | Yes | Yes | Yes |
| GUI available | Yes | Yes | Yes (Mclust) |
| Version tested | 2.0 | 1.6 | 2.1 |

comparison of the clustering algorithm, because the goal of the paper was only to reduce the data for hardware implementation.

In summary, very few quantitative comparisons of spike sorting methods have been made, and there are no standard criteria for evaluating them. We propose in this paper an evaluation framework aimed at providing a fair comparison of spike sorting methods in more optimal terms.

## 2. Materials and methods

The objective of the study is to compare the three most widely-used publicly-available spike sorting algorithms (WaveClus, KlustaKwik, OSort) with regard to their parameter settings. We observed that even a small change in the parameters of a spike sorting algorithm may have a dramatic impact on their accuracy. Therefore a comparison between spike sorting algorithms and non-optimal parameters could be biased. To overcome this weakness, we employ an optimization technique on artificial signals to find near-optimal parameter settings. Using these settings, we compared the algorithms on various types of artificial signals, focusing on single-channel recordings (similar to extracellular signals recorded using a single micro electrode).

### 2.1. Spike sorting algorithms

The most important properties of all three spike sorting algorithms selected in the previous section are summarized in Table 1. There follows. A more detailed description of the algorithms that have been used follows.

#### 2.1.1. WaveClus

WaveClus is an unsupervised spike detection and sorting algorithm that combines the wavelet transform (localizing distinctive spike features) with superparamagnetic clustering (SPC), which is a method used in statistical mechanics (Quiroga et al., 2004). It enables clustering of the data without assumptions such as low variance or Gaussian distributions. In the first step, spikes are detected with an automatic amplitude threshold on the high-pass filtered data. In the second step, a small set of wavelet coefficients from each spike is chosen as the input for the clustering algorithm. Finally, SPC classifies the spikes according to the selected set of wavelet coefficients (Quiroga et al., 2004). WaveClus is one of the most widely-used spike sorting algorithms, and it has a large number of parameters for fine-tuning the method (see Table 2 for details). WaveClus version 2.0 was used for the comparison.

#### 2.1.2. OSort

OSort is an implementation of a template-based, unsupervised online spike sorting algorithm. The estimation of the number of neurons present, as well as the assignment of each spike to a neuron, is based on a distance metric between two spikes (Rutishauser, 2006). Based on this distance, a threshold is used: (i) to decide how many neurons are present and (ii) to assign each spike uniquely to a neuron cluster, or to a noise cluster if unsortable. The threshold is calculated from the noise properties of the signal and is equal to the squared average standard deviation of the signal, calculated with a sliding window. The main advantage of OSort over its competitors is that it can be used online, thus enabling realtime spike sorting during an experiment (Rutishauser, 2006). OSort version 2.1 was used for the comparison.

#### 2.1.3. KlustaKwik

KlustaKwik is a software for unsupervised classification of multidimensional data. It is employed in the MClust toolbox, which enables both manual and automatic spike sorting on single-electrode, stereotrode and tetrode recordings. KlustaKwik fits a mixture of Gaussians with unconstrained covariance matrices and automatically chooses the number of mixture components. PCA is used to extract spike features for the clustering and a penalty term for selecting the number of clusters is implemented. The penalty is based on the ability to specify Bayesian information con-

**Table 2**
List of parameters impacting the spike sorting accuracy for each algorithm. The parameter names were taken directly from the original source code of each algorithm author.

| WaveClus | |
|---|---|
| force_auto | Automatically force membership of spikes assigned to noise cluster using template matching |
| inputs | Number of wavelet coefficients to use as features for clustering |
| KNearNeighb | Number of data points used for the nearest neighbors interactions in the SPC |
| min_clus_stop | Minimum size of a cluster (cluster will be deleted if the number of spikes it contains is lower than this value) |
| mintemp | SPC minimum temperature – a lower temperature value groups all data into a single cluster, while higher values allow the data to split into more clusters |
| scales | Number of wavelet decomposition levels used |
| SWCycles | Number of Monte Carlo iterations used by SPC |
| template_type | Type of template matching method used – template matching is used for spike sorting speed up in the case of large number of spikes or for assigning spikes in the noise cluster to the existing clusters (if force_auto is set) |
| KlustaKwik | |
| noDim | Number of PCA dimensions used for clustering |
| MinClusters | The random initial assignment will have no less than *MinClusters* clusters. The final number may be different, since clusters can be split or deleted during the course of the algorithm |
| PenaltyMix | Amount of Bayesian information content (BIC) or Akaike information content (AIC) to use as a penalty for more clusters. Default of 0 sets to use all AIC. Use 1.0 to use all BIC (this generally produces fewer clusters) |
| OSort | |
| minNrSpikes | Minimum size of a cluster (cluster will be deleted if the number of spikes it contains is lower than this value) |
| correctionFactorThreshold | Value correcting a signal noise estimate used as a clustering threshold |

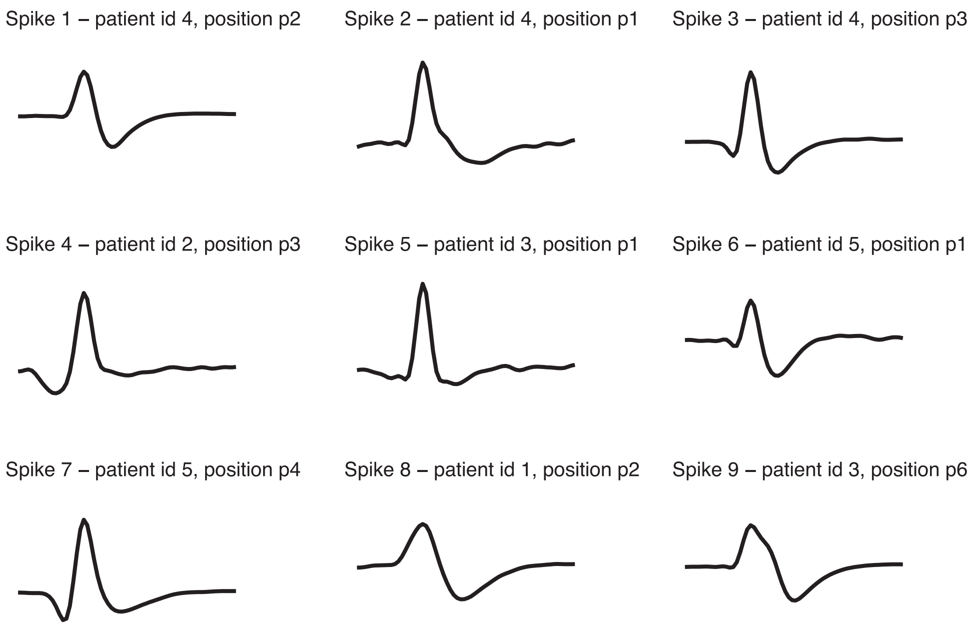

**Fig. 3.** Waveforms of 9 real spikes, used for artificial signal generation. Each spike represents a different neuron.

tent (Cheeseman and Stutz, 1996). KlustaKwik allows a variable number of clusters to be fitted. The program periodically checks if splitting any cluster would improve the overall score. KlustaKwik also checks to see if deleting any cluster and reallocating its points would improve the overall score. The splitting and deletion features often allow the program to escape from local minima, reducing sensitivity to the initial number of clusters, and reducing the total number of starts needed for a data set (Harris, 2000). KlustaKwik version 1.6 was used for the comparison.

### 2.2. Test data

For the purposes of comparison we used two sets of artificial data: previously published data (Quiroga et al. (2004), referred to as QQ after Quian Quiroga) and data generated by our own method (referred to as JW, publicly available online – http://nit.felk.cvut.cz/~wildj1/ssc). Both of these data sets were obtained simulating extracellular signals recorded using a single micro electrode.

Our artificial data was generated by superimposing real spikes at random times onto a noise background. Since several aspects of signals affect spike sorting, we used a wide range of signals of different characteristics (signal noise level, number of neurons) to maximize the objectivity and discriminability of our results.

A total of 9 real spikes (64 samples) shown in Fig. 3 were picked manually from extracellular tungsten micro-electrode recordings during a Deep Brain Stimulation operation from the sub-thalamus nuclei (STN) of 5 patients. Each spike was deduced from a different position in the STN, thus eliminating the possibility of extracting two separate spikes of the same neuron.

To generate a signal with $n$ neurons, spikes $1\ldots n$ were used as a template for each neuron. Each template was first scaled to 75–125% (uniform distribution) of its maximal amplitude to mimic the different spatial distance from each neuron to the electrode and was placed at random positions in the signal, while maintaining a neuronal refractory period of 3 ms. The contribution of different neurons was independent, such that spikes of different neurons might have coincided with each other in the signal, simulating the situation of several neurons firing at the same time.

The noise background for longer signals (60, 960 s) was generated in the same way as for the QQ data (Quiroga et al., 2004) using over 2000 different spikes (some of which might be from the same neuron), thus simulating the activity of many distant neurons in the brain. For shorter signals (20 s), a spike-less part of a raw signal recorded from STN was used as a noise background to approximate real signals more closely. The noise was then scaled, so that its standard deviation $\sigma$ lies within the range, and was then superimposed on the previously generated signal to get the final artificial record.

Twenty-two QQ signals (60 s) and another 90 JW signals with 2–9 neurons generated using the described procedure were used to evaluate the spike sorting algorithms on a large variety of signals with different properties. The JW signals were split into three groups according to their length – 40 short JW signals (20 s), 40 long JW signals (60 s) and 10 very long JW signals (960 s). The signals with the same number of neurons differed in the standard deviation of the noise that was superimposed on the signal element. However, as it was very difficult to estimate (and compare) the standard deviation of the noise component in the case of real signals, all the JW and QQ data was labeled using a straightforward noise estimation method (see Section 2.3).

### 2.3. Noise level estimation

The noise level $n_l$ was defined as the reciprocal value to the signal-to-noise ratio *SNR* (Smith, 1999)

$$n_l = \frac{1}{SNR} = \left(\frac{A_{\text{noise}}}{A_{\text{signal}}}\right)^2 \qquad (1)$$

where $A_{\text{signal}}$ represents the root mean square (RMS) amplitude calculated from all the spikes extracted using spike detection, and $A_{\text{noise}}$ accounts for RMS computed from the rest of the signal. As the estimated noise level was normalized, it was easier for comparison across signals with different amplitude ranges, as opposed to standard deviation. An inevitable drawback of the method described here was that the estimation was slightly biased as, in theory, $A_{\text{signal}}$ had to be calculated only from the useful signal (spikes), whereas in the real case the spikes themselves were corrupted by noise.

For illustration purposes, Fig. 4 depicts the same 250 ms long signal with four different noise levels (0.05, 0.15, 0.25, 0.35). On

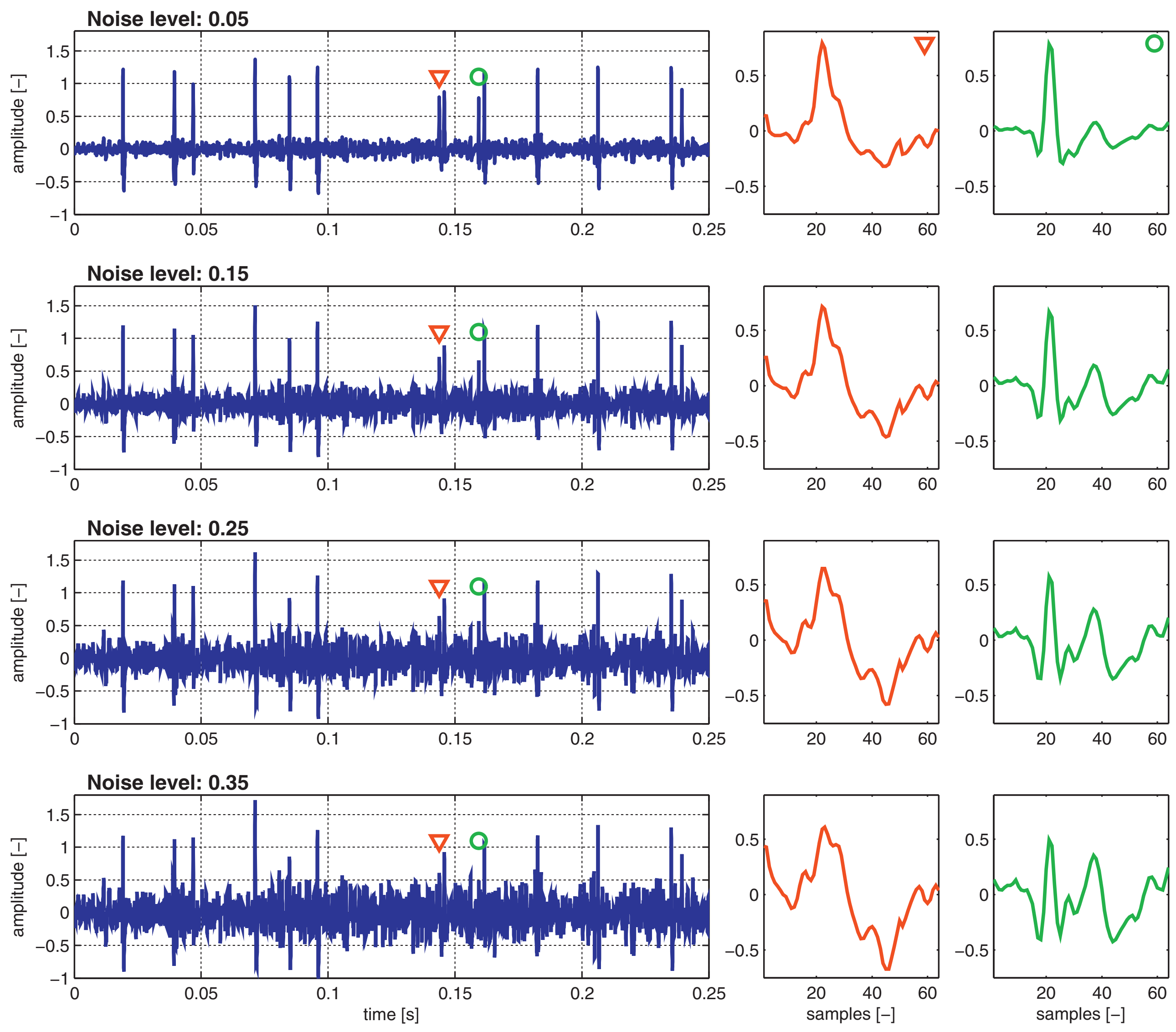

**Fig. 4.** Example of the same 250 ms-long signal with different noise levels ranging from 0.05 to 0.35. The spikes marked in the signal by a triangle and a circle each belonged to a different neuron and are shown in greater detail on the right side – in the case of a higher noise level at 0.25 and 0.35, a new noisy spike could be misleadingly detected.

the right side of each signal there is a detail of two spikes (marked in the signal by a triangle and a circle), each belonging to a different neuron. This is an example to illustrate of how much the noise affected the shape of the spikes.

### 2.4. Performance rating function

In order to asses the accuracy of different spike sorting algorithms and to provide an objective function for optimization, a performance measure was needed. As the experiments were performed using artificial data, the true clustering of the spikes was available. In machine learning research, many measures have been proposed for this type of clustering evaluation task (Warrens, 2008; Vinh et al., 2009, 2010), and some of them have already been used for spike sorting evaluation (Kretzberg et al., 2009; Gasthaus, 2008). Recently, Vinh et al. (2010) showed that Adjusted Mutual Information (AMI) had the best properties among all these clustering evaluation measures, so this measure was used for the evaluation.

AMI is an information theoretic measure which usually provides a value between 0 and 1. The value is 0 if the clustering provides information about the true clustering just by chance, and it is 1 if all information is revealed, meaning that the two clusterings are the same. Hence, AMI can be considered as the ratio of true information in a spike sorting result. Several AMI values and their corresponding clustering are shown in Fig. 5.

### 2.5. Spike sorting parameters

All of the spike sorting algorithms discussed in this paper have a number of parameters (OSort – 2 parameters; KlustaKwik – 9 parameters; WaveClus – 13 parameters) that can be adjusted in order to improve the spike sorting accuracy. However, it was very difficult to set these parameters correctly using manual methods.

Although all the parameters were documented, it was an almost impossible task to find out how to operate them so that the algorithm would perform better on a given signal. The parameter search was thus formulated as an automatic optimization problem: given a set of algorithm parameters $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, find a solution for $\arg\max_{\mathbf{x}} f(\mathbf{x})$, where the $f(\mathbf{x})$ objective function is the value of the performance rating function (the AMI score) for the spike sorting results obtained with parameter vector $\mathbf{x}$. As artificial signals were used in this study, the AMI could be calculated for the parameter space and the optimal solution could be identified by an exhaustive search. Gradient descent and genetic algorithms were

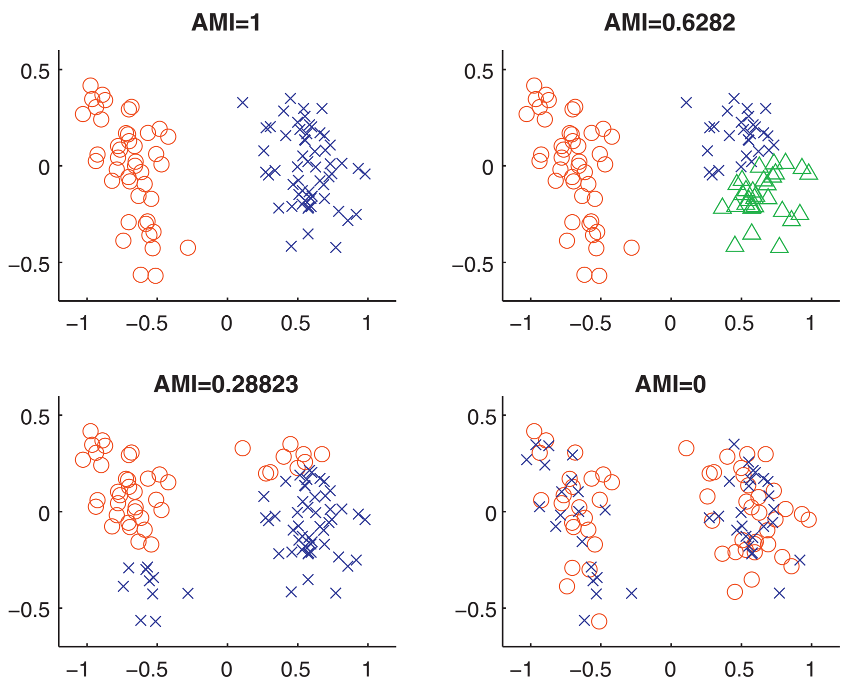

**Fig. 5.** Several clustering results with their corresponding AMI values. The correct clustering is presented at the top on the left with different shapes for each cluster. At the top on the right, one cluster is further split, so AMI is reduced. At the bottom on the left, the number of clusters is correct, but there is a wrong split. At the bottom on the right, there is random clustering, so the AMI value is zero.

also considered, but the objective function changed significantly with only a small change in the parameters, so only an exhaustive search guaranteed finding the global optima.

While employing the exhaustive search, only some of the algorithm parameters proved to have an impact on the spike sorting accuracy. The Table 2 summarizes the names of these parameters for all three algorithms. A complete annotated list of all parameters is available online at http://nit.felk.cvut.cz/~wildj1/ssc or at each algorithm author's website.

In order to make a fair comparison between algorithms with different numbers of parameters, all signals were split into two parts. The first part was used for optimization to find the ideal parameters, and the second part was utilized to evaluate the spike sorting accuracy with these parameters.

### 2.6. *Technical equipment used*

All calculations and statistical analyses were performed using MatLab (Mathworks, Natick, MA). The spike sorting results for the different algorithms were calculated using a Dell Precision workstation running 32-bit Linux Mint with a 2.13 GHz Intel Core 2 Duo E6400 2.13 GHz and 2 GB of DDR2 RAM.

### 2.7. *Statistical methods*

For each artificial signal the AMI scores were calculated for each spike sorting algorithm, using either optimized or default parameters. For the spike sorting evaluation, the signals and their corresponding AMI scores were grouped according to the algorithm used and the signal noise level. Each group was visualized as a simplified boxplot showing the median and the lower and upper quartiles. The range between these quartiles is referred to as the spread. Differences between group medians were assessed using the two-sided Wilcoxon signed-rank test. Bonferroni corrections for multiple comparisons were applied whenever appropriate.

For the comparison between the optimized parameters, and the default parameters the AMI scores were grouped according to the algorithm and parameters that were used (either optimized or default). For visualization, the simplified boxplots were used as described above. Significant differences between the medians of the groups were assessed in the same way as for the spike sorting evaluation, using the two-sided Wilcoxon signed rank test.

## 3. Results and discussion

The algorithms were compared in two main aspects. First, the spike sorting accuracy was measured with AMI (one AMI score for each signal and algorithm). The results correspond to the evaluation part of the signals, unless otherwise stated. Second, the speeds of these algorithms were compared to give some impression of the number of spikes that can be processed within a reasonable time.

### 3.1. *Optimized parameters*

As was already discussed in Section 2.5, the parameters were optimized on one part of the signal and evaluated on the other half. It was important to see whether this optimization really yielded better results than the default parameters of the algorithm. Fig. 6 shows the spike sorting accuracy results using near-optimal parameters in comparison with the results employing the default parameters. JW short, long and QQ signals with noise levels ranging from 0.0 to 0.6 were used for this comparison. Although the spread of the AMI values was quite high, mainly due to the noise level diversity of signals used, it could be clearly seen that the optimization improved all three algorithms ($p < 0.01$).

### 3.2. *Spike sorting accuracy*

Our main assumption was that increasing noise levels have a negative effect on spike sorting accuracy. We therefore present our results depending on noise levels. Fig. 7 shows the spike sorting accuracy of WaveClus, KlustaKwik and OSort on short (10s) JW signals with noise levels ranging from 0.0 to 0.6. For signals with noise level between 0.00 and 0.15, WaveClus was the most accurate

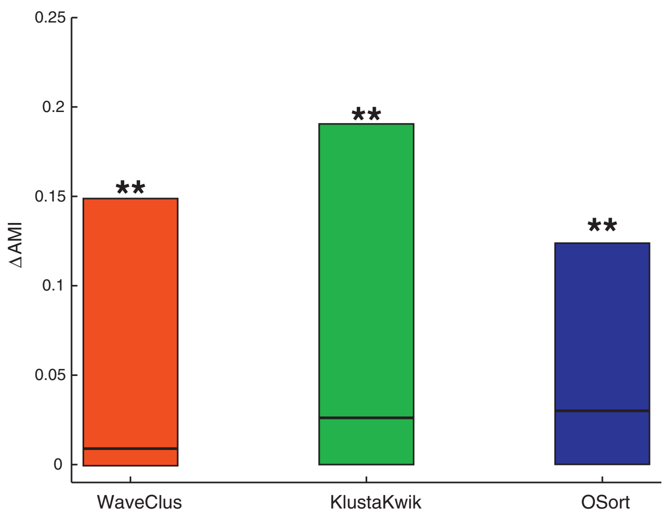

**Fig. 6.** Comparison of the accuracy of the algorithms when used with default and optimized parameters. The *y*-axis represents the difference between the achieved AMI score while using optimized parameters and while using default parameters). Symbol ** indicates that the medians of the marked boxplots are significantly different from zero ($p < 0.01$).

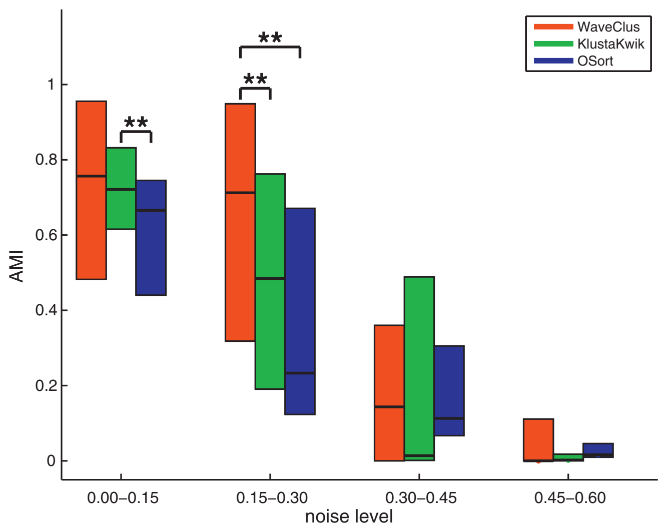

**Fig. 8.** Performance of spike sorting algorithms using long (30 s) artificial JW and QQ signals with noise levels binned and optimized parameters. The *y*-axis represents the AMI score of each algorithm along with its spread. Symbol ** indicates that the medians of the marked boxplots are significantly different ($p < 0.01$ corrected for 3 comparisons).

algorithm, with a median AMI of 0.7. However, because of its large spread the difference between WaveClus and KlustaKwik or OSort was not significant.

With added noise, the median AMI of all respective algorithms decreased, with both WaveClus and KlustaKwik proving to be significantly better than OSort ($p < 0.05$ and $p < 0.01$ at noise level 0.15–0.30), both having a better AMI score than OSort for 80% of 10 s signals within the respective noise level range. For signals with noise levels above 0.30 all three algorithms had very poor accuracy, indicating that signals with such a high noise level were beyond their capabilities.

Fig. 8, which depicts the same experiment as Fig. 7, only with longer JW and QQ signals (30 s), gave us somewhat similar results for WaveClus and OSort. WaveClus performed best in all cases, though it was significantly better ($p < 0.01$) than both of its competitors only for noise level 0.15–0.30 (it had a better AMI score for 89% of the respective signals). KlustaKwik was significantly better than OSort for noise level 0.00–0.15, though with higher noise levels KlustaKwik had a larger spread than OSort. Again, noise level above 0.30 was too high for the algorithms to give reasonable results.

Some spikes were visually investigated in order to explain the effect of the noise levels. Judging from Fig. 4, the spike shape (in this case) remained almost unchanged for noise levels 0.05 and 0.15, but at 0.25 and 0.35 the spike shape did not seem like the shape at 0.05. This had a direct negative effect on the spike sorting accuracy of OSort, as shown in Figs. 7 and 8, in comparison with WaveClus, because OSort used raw spike shapes (without any filtering) and a simple distance measure for sorting.

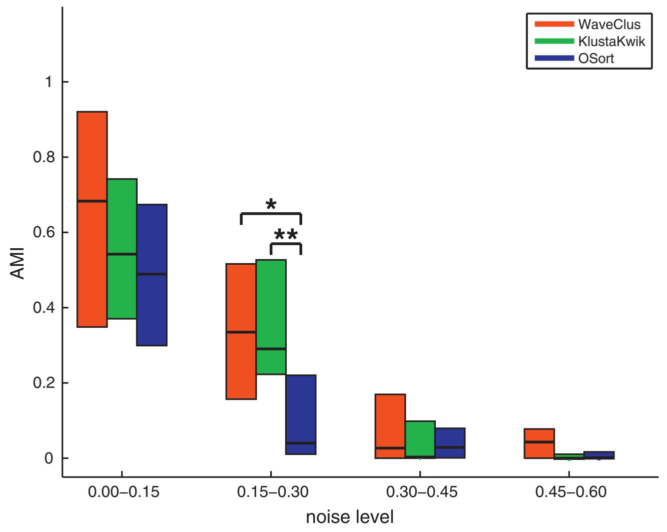

**Fig. 7.** Performance of spike sorting algorithms using short (10 s) artificial JW signals with noise levels binned and optimized parameters. The *y*-axis represents the AMI score of each algorithm along with its spread. Symbols * and ** indicate that the medians of the marked boxplots are significantly different ($p < 0.05$ and $p < 0.01$, corrected for 3 comparisons).

### 3.3. *Spike sorting time complexity*

In a real world scenario, the speed of an algorithm may be of considerable importance. For example, if a certain algorithm can be run online, it will help researchers to gather sorted spiking data from micro electrodes in real time. Of these three algorithms, only OSort is online, which means that it processes spikes one-by-one as they come. For the other two algorithms, the whole spike sorting process needs to be re-run with all previous data to cluster the new spikes, so they are more targeted for offline analysis when new spikes are not coming in. Even for large-scale offline analysis, it would be good to know the computational demand of the algorithms.

Ten very long signals (960 s) with noise level 0.15 were used for evaluating the time complexity of each algorithm. The 960 s signals were cut into shorter signals, with the number of spikes varying from 100 to 19,460. The parameters for each spike sorting algorithm were optimized on the first part (1400 spikes) of each 960 s signal, and remained unchanged for all the other parts originating from this signal. As only the speed of the algorithms was measured and not their actual accuracy, parameter optimization of each individual signal part was unnecessary.

The results of the speed test are shown in Fig. 9. OSort was the fastest algorithm, with an average speed of 1100 spikes/s, whereas the average speed for KlustaKwik and WaveClus was 200 and 100 spikes/s respectively.

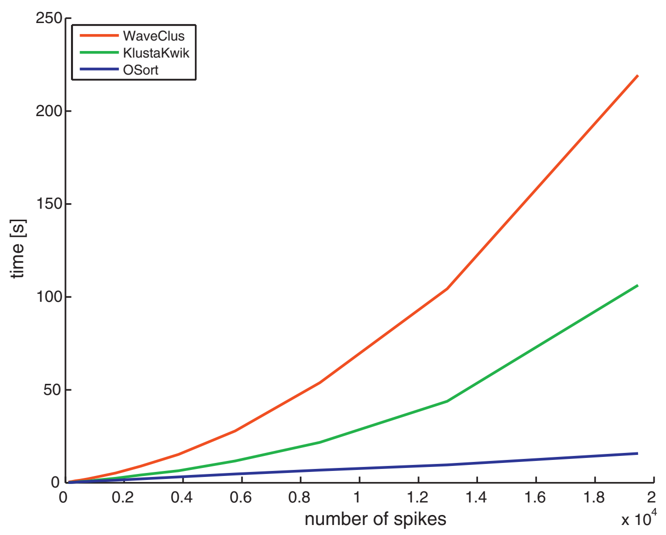

**Fig. 9.** Relation between number of spikes and time needed for the spike sorting algorithms to run. Long JW signals (960 s, noise level 0.15) cut into several parts were used. The parameters were optimized using the first signal segment with 1400 spikes and remained the same for all other segments.

## 4. Conclusion

Three widely-used publicly-available spike sorting algorithms were compared (WaveClus, KlustaKwik, OSort) with regard to their parameter settings, using single-channel artificial data with different noise levels and different number of neurons. To avoid biased results, an optimization technique was employed based on Adjusted Mutual Information to find near-optimal parameter settings for our artificial signals. When using the near-optimal parameters, each algorithm improved its spike sorting accuracy as opposed to when only the default parameters were used ($p < 0.01$). Using these settings, an objective comparison of the three algorithms was made.

WaveClus was the best performing spike sorting algorithm. The accuracy of KlustaKwik was comparable to that of WaveClus at a lower noise level (0.00–0.15), and worse otherwise. Although OSort performed less well than both WaveClus and KlustaKwik, it sorted spikes at more than five times faster, and can thus be recommended for real-time signal processing with a low amount of noise present (below noise level 0.15). Where there is high noise (noise level greater than 0.3), none of the three algorithms provided reasonable results.

As our artificial data is publicly available online, we believe that our framework can be extended to further spike sorting algorithms, thus providing an objective comparison platform for neuroscience researchers.

## Acknowledgements

This work has been supported by Czech Science Foundation research program 309/09/1145 and by Grant Agency of the Czech Technical University in Prague Grant no. SGS10/279/OHK3/3T/13. The second author has been supported by Visegrad Fund scholarship no. 51000739.

## References

Adamos D, Kosmidis E, Theophilidis K. Performance evaluation of PCA-based spike sorting algorithms. Comput Methods Prog Biomed 2008;91:232–44.

Adamos D, Laskaris N, Kosmidise E, Theophilidis G. Nass: an empirical approach to spike sorting with overlap resolution based on a hybrid noise-assisted methodology. J Neurosci Methods 2010;190(1):129–42.

Cheeseman P, Stutz J. Bayesian classification (autoclass): theory and results. AAAI Press, MIT Press; 1996.

Delescluse M, Pouzat C. Efficient spike-sorting of multi-state neurons using interspike intervals information. J Neurosci Methods 2006;150(1):16–29.

Ding W, Yuan J. Spike sorting based on multi-class support vector machine with superposition resolution. Med Biol Eng Comput 2008;46:139–45.

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.

Fee MS, Mitra PP, Kleinfeld D. Erratum: automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-gaussian variability. J Neurosci Methods 1997;71(2):233.

Franke F, Natora M, Boucsein C, Munk MHJ, Obermayer K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. J Comput Neurosci 2009;29:127–48.

Gasthaus J. Spike sorting using time-varying DIRICHLET process mixture models. Master's thesis. University College London; 2008.

Gerstein GL, Clark W. Simultaneous studies of firing patterns in several neurons. Science 1964;143(20):1325–7.

Gibson S, Judy J, Markovic D. Comparison of spike-sorting algorithms for future hardware implementation. In: Engineering in medicine and biology society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE. IEEE; 2008. p. 5015–20.

Gibson S, Judy J, Markovic D. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. IEEE Trans Neural Syst Rehabil Eng 2010;18(5):469–78.

Harris KD. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. J Neurophysiol 2000;84(1):401–14.

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

Hulata E, Segev R, Ben-Jacob E. A method for spike sorting and detection based on wavelet packets and shannons mutual information. J Neurosci Methods 2002;117:1–12.

Joshua M, Elias S, Levine O, Bergmana H. Quantifying the isolation quality of extracellularly recorded action potentials. J Neurosci Methods 2007;163:267–82.

Kretzberg J, Coors T, Furche J. Comparison of valley seeking and T-distributed EM algorithm for spike sorting. BMC Neurosci 2009;10(1):47.

Lewicki M. A review of methods for spike sorting: the detection and classification of neural action potentials. Network-Comp Neural 1998;9(4):53–78.

Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput 2004;16(8):1661–87.

Rutishauser U. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. J Neurosci Methods 2006;154:204–24.

Sahani M. Latent variable models for neural data analysis. Ph.D. thesis. Pasadena, California: California Institute of Technology; 1999.

Salganicoff M, Sarna M, Sax L, Gerstein GL. Unsupervised waveform classification for multineuron recordings: a real-time, software-based system. J Neurosci Methods 1998;25(3):181–7.

Schmitzer-Torbert N, Jackson J, Henze DHK, Redish A. Quantitative measures of cluster quality for use in extracellular recordings. Neuroscience 2005;131:1–11.

Shoham S, Fellows MR, Normann RA. Robust, automatic spike sorting using mixtures of multivariate T-distributions. J Neurosci Methods 2003;127(2): 111–22.

Smith SW. The scientist and engineer's guide to digital signal processing. San Diego, California: California Technical Publishing; 1999.

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 2003a;46(3):265–72.

Takahashi S, Anzai Y, Sakurai Y. Automatic sorting for multineuronal activity recorded with tetrodes in the presence of overlapping spikes. J Neurophysiol 2003b;89(4):2245–58.

Tankus A, Yeshurun Y, Fried I. An automatic measure for classifying clusters of suspected spikes into single cells versus multiunits. J Neural Eng 2009;6(5): 056001.

Vinh N, Epps J, Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th annual international conference on machine learning. ACM; 2009. p. 1073–80.

Vinh N, Epps J, Bailey J. Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. JMLR 2010;11:2837–54.

Wang G, Zhou Y, Chen A, Zhang P, Liang P. A robust method for spike sorting with automatic overlap decomposition. IEEE Trans Bio-Med Eng 2006;53(6): 1195–8.

Warrens M. On the equivalence of Cohen's kappa and the Hubert-Arabie adjusted Rand index. J Classif 2008;25(2):177–83.