35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Neural spike sorting with a self-training semi-supervised support vector machine

Abdollah Ghanbari[1], Mohammad B. Shamsollahi[1], Vincent Vigneron[2], Abdessalam Kifouche[2, 3]

[1]*Biomedical Image and Signal Processing Laboratory (BiSIPL), Department of Electrical Engineering*
Sharif University of Technology, Tehran, Iran
[2]*University of Evry Val d'Esson, Evry, France*
[3]*Université Saad Dahlab Blida, Tunisia*
ghanbari@ee.sharif.edu, mbshams@sharif.edu, vincent.vigneron@ibisc.univ-evry.fr

***Abstract*—Brain decoding would be a replacement for some nerve injured patients to communicate motor functions with a prosthesis device. Decoding algorithms translate ensemble of firing rates to the intended function. Firing rates for each individual neuron are obtained from labeling the detected spikes. This labeling process -also known as spike sorting- could be done from the range of fully automated to a heavily operator dependent manners. On the other hand we could use merits of both automation and operator's watch in a semi-supervised approach. In this study we explored the application of a self-training SVM classifier algorithm to label spikes with a small training dataset. Result shows the proved monotonically increasing convergence and consequently the ability of this algorithm to significantly reduce the operator's effort for continuous supervision. It provides in addition a significant improvement with respect to the previously used SVMs.**

***Index Terms*—Firing rate, neural decoding, spike sorting, self-training SVM**

I. Introduction

THE brain is the most complex organ in the human body. Several approaches proposed to explore its functioning like EEG, MEG, fMRI and extracellular recording. In the expense of invasiveness, extracellular recording has the highest resolution among them in order of a single neuron [1]. Extracellular recording data usually has being recorded in an intracellular space with 10 to 100 neurons surrounding by an array of electrodes. Therefore each electrode records a superposition of spikes originated from nearby neurons [2]. Spike is a deformed version of an internal action potential recorded from its outside. It is needed to separate the activity of each neuron by assigning its corresponding spikes which is also called spike sorting. The output of the sorting step could be used in decoding through computing firing rates.

Spike sorting as a meddling stage could cause an exhaustive defect in the decoding if it is done inaccurately. This process could be done in a supervised, unsupervised or a semi-supervised manner. Supervised algorithms need labeled data which is done by an operator and this kind of labeling is almost impossible for a long time recording period. Some unsupervised algorithms such as those based on wavelet packets were proposed in [3]. But these unsupervised algorithms are designed to satisfy the real-time implementation and they are not recommended for testing a decoding algorithm. Testing a decoding algorithm needs a high accuracy spike sorting even it is done offline.

On the other hand semi-supervised methods use an operator to check or initiate the sorting algorithm. These "learning" approaches have different level of complexity based on the operator's dependence. Operator's contribution in the semi-supervised approaches could be in the selecting the number of clusters but also in assigning a small number of waveform to each cluster. In this study we proposed an approach to reduce the operator's effort and also gain a highest possible accuracy based on the knowledge that the operator gave us at the initial point in an offline processing manner.

The rest of paper is organized as follows. Section II gives a brief mathematical view of a standard SVM and the self-training semi-supervised algorithm. It also contains a mathematical description of our model selection approach. The results are exposed in section III and finally a brief discussion is given in IV.

II. Methods

Spike sorting is the task of assigning each waveform of detected spikes to the neuron it is originated from. This task is usually done by clustering. In this approach each spike waveform is represented as a vector in the feature space. By the assumption of waveform consistency during the extracellular recording process and well-chosen features it is possible to assign each waveform to its neuron.

In recent years several classifiers have been employed in spike sorting. Among them SVM has been modified several times to be used in this context because of its generalization power [4][5]. In addition Ding et al. [5] showed the ability of a multi-class clustering done by SVM to overcome the problem of superposed spikes. In their proposed method, they ignored the spikes that were not classified in their one-against-all approach.

A standard SVM originally is designed for two-class

classification problems. In a spike sorting study we usually counter with a mixture of neurons in each channel. One-against-all or one-against-one could be employed to expand a standard two-class SVM to a multi-class case. In [5] it is shown that one-against-all technique has a superiority over the others. In this paper we modified the self-training semi-supervised algorithm proposed by [6] and applied it to a spike sorting task using one-against-all technique.

A. *Self-training Semi-supervised Algorithm*

Objective function of a two-class standard *C*-SVM is defined as follow:

$$f(\mathbf{w}, \boldsymbol{\varepsilon}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=0}^{N}\xi_i$$

where we aim to find the normal vector of the separating hyperplane, $\boldsymbol{w}$, y-intercept, b, and ξ_i which measures the misclassification rate to minimize the objective function subject to:

$$y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}} + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0 \;\; i = 1,2,\dots,N$$

where $x_i \in \mathbb{R}^l$ is a feature vector from the training set, $y_i \in \{1, -1\}$ is the label of corresponding samples and *C* is the penalty parameter. In a standard *C*-SVM, *C* is a constant value during the minimization process, later we will discuss how we will choose this constant value in each iteration.

A self-training semi-supervised algorithm uses this *C*-SVM classifier. Suppose that F_I is a small training set that contains N_I sample $\{\mathbf{x}_i, i = 1,2,\dots,N_I\}$ which have been labeled by an operator $[y^{(0)}(1),\dots,y^{(0)}(N_I)]$, and a test set F_T containing N_T samples $\{\mathbf{x}_{N_I+i}, i = 1,2,\dots,N_T\}$ with unknown labels. The steps of the self-training semi-supervised *C*-SVM algorithm are as follows:

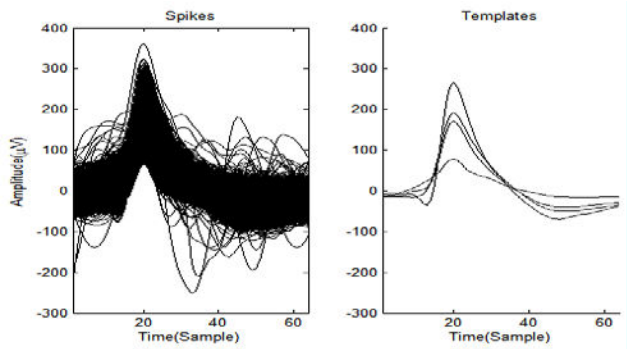
- In the first step we train a standard SVM with the labeled data and use the designed classifier to label the test set F_T. The parameters of the SVM classifier are denoted $\mathbf{W}^{(0)} \in \mathbb{R}^l, \varepsilon^{(0)} \in \mathbb{R}^{N_I}$ and $b^{(0)} \in \mathbb{R}$, the predicted labels are $[y^{(0)}(N_I+1),\dots,y^{(0)}(N_I+N_T)]$. The upper subscript denotes the iteration number.
- The Second step is a loop. In each iteration, *k,* we define a new training set $F_N = F_I + F_T$, with the labels predicted earlier, $[y^{(k-1)}(N_I+1),\dots,y^{(k-1)}(N_I+N_T)]$. Then by this augmented training set, we train a SVM and perform a new classification on F_T. We do this step for all *C* values in a pre-defined subset and find the optimal *C* as it is described later on model selection. For this optimum *C* value, the parameters of the SVM are denoted as $\mathbf{W}^{(\mathrm{k})} \in \mathbb{R}^l, \varepsilon^{(k)} \in \mathbb{R}^{N_I+N_T}$ and $b^{(k)} \in \mathbb{R}$. We next apply this classifier to the data and update the labels of the test set $[y^{(k)}(N_I+1),\dots,y^{(k)}(N_I+N_T)]$. Train set labels will remain the same in each iteration. Finally in this step we calculate the objective function by obtained parameters in each iteration:

$$f\left(\mathbf{w}^{(\mathrm{k})}, \boldsymbol{\xi}^{(k)}\right) = \frac{1}{2}\left\|\mathbf{w}^{(\mathrm{k})}\right\|^2 + C\sum_{i=0}^{N_I+N_T}\xi_i^{(k)}$$

- In the third step we check the termination criteria by calculating the absolute value of the two consecutive iterations and the algorithm will be stopped if this criteria is less than a predefined positive value:

$$\left|f\left(\mathbf{w}^{(\mathrm{k})}, \boldsymbol{\xi}^{(k)}\right) - f\left(\mathbf{w}^{(\mathrm{k}-1)}, \boldsymbol{\xi}^{(k-1)}\right)\right| < \delta_0$$

where δ_0 is the least change we expect from our objective function. This also means that the algorithm reaches its local minimum.



(a) (b)

Fig. 1. (a) Overall view of the spikes and (b) the corresponding templates

The convergence of the original algorithm has been theoretically proved in [6]. This convergence can be observed in our result. Another modification of this algorithm has been proposed by [7] which enters the scattering matrix into the SVM's objective function and claimed a better result.

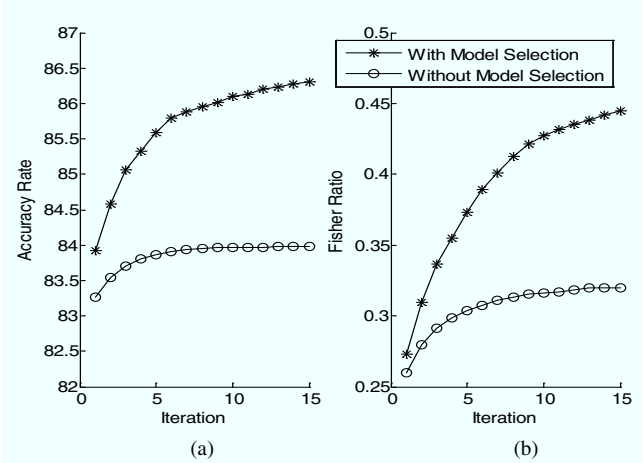
B. *Model Selection*

In the algorithm described in the previous subsection we should set the penalty parameter *C* of SVM through model selection. In [6] it was suggested to use an one-dimensional Fisher ratio and it was not mentioned how we can employ it to a high-dimensional space. Here we used a robust Fisher ratio which can be used in our high-dimensional feature space. It is also common to use cross-validation on training data set and search in a specific range for *C* to find the best possible *C* which gives us the maximum accuracy in our training set. Here because our training set is so small we could not rely on these results. In [7] cross-validation has been used in self-training semi-supervised algorithm however it was prohibited in [6].

Now we search in a finite set of *C* values in a pre-defined subset $\{C_1,\dots,C_L\}$ and simultaneously calculate the Fisher discriminant ratio:

$$\mathrm{FDR}(k, C_i) = \frac{\boldsymbol{u}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{u}}{\boldsymbol{u}^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\boldsymbol{u}}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote the mean and covariance of each class respectivly, and $\boldsymbol{u}$ is a discriminant vector. The lower



(a) (b)

Fig. 2. (a) Accuracy comparison and (b) discriminant measurement of the simulated spike data.

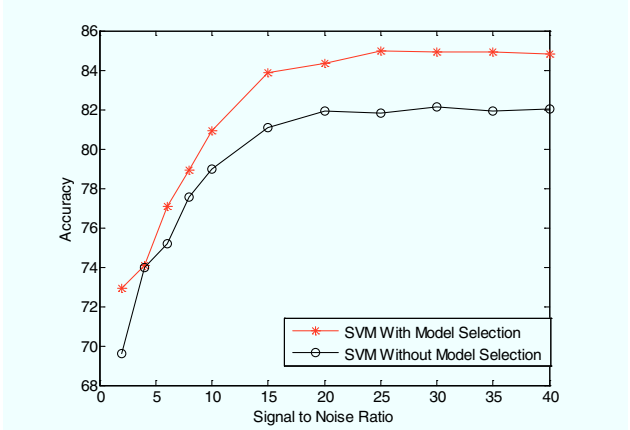


Fig. 3. Accuracy measurement of the proposed method and the method without the model selection in different SNRs.

subscript denotes the class label. A $\boldsymbol{u}$ that gives the maximum discriminant is [8]:

$$\boldsymbol{u} = (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

After computing Fisher ratio for all C_l we define C as:

$$C_{opt}^{(k)} = \max_{C_i} \text{FDR}(k, C_i)$$

where $C_{opt}^{(k)}$ is the penalty parameter applied to augmented training set in the kth iteration in the second step of the algorithm.

This model selection method is based on the fact that the original Fisher ratio represents the reparability of the clusters, i.e., the Fisher ratio increase implies high reparability. Therefore in each iteration we choose a C that gives us a higher discrimination.

III. Dataset & Results

A. Dataset

The data came from a 30' multiunit recording from a human epileptic patient. This data was collected at the lab of Itzhak Fried at UCLA, using a Neuralynx system (Tucson, Arizona). The spikes have been detected and also clustered using Waveclus toolbox by University of Lecister and this clustering has been accepted as our ground truth [9]. The cluster membership number for spikes also contains some zero values which were considered as spikes and were not assigned to any cluster by the Waveclus contributors. An overall view of the detected spikes is represented by a 64 sample point and their corresponding templates are plotted on Fig. 1. These templates are calculated by averaging of the spikes in the same class. The template with the lowest peak corresponds to the non-clustered spikes.

B. Results

Since we had a multiclass problem in spike sorting in one hand and a two-class classifier on the other hand we had to choose between one-against-one, one-against-all and Directed Acyclic Graph SVM [10]. In [5] Ding et al. claimed one-against-all showed a better performance in neural activity gathered by micro-electrode arrays. Accordingly we employed this approach by combining three binary classifiers. We could have considered the unlabeled data as a separate class but we ignored it in our results.

To do the clustering we need to extract features from our spike waveforms and project each spike to the feature space. Here we used Principal Component Analysis (PCA) and project the spike signal to an orthogonal space as the first component have the biggest projection of the signal. The succeeding components is also have the most value under the constraint that it be orthogonal the preceding components. The original spikes were sampled in 64 points, so we selected the first 25 components of PCA and used it as our features [1], [2]. These 25 components contain almost 90 percent of the spike's energy. To apply the algorithm to the dataset here we chose 20 labeled samples as our training set, F_I. For the test set, F_T, we used 980 spikes to role in the iterative scheme in the second step. Choosing all the 10,000 remained spikes in our 30' data set although could improve the accuracy but it causes a severe computational burden. To implement the standard SVM in the algorithm we used LIBSVM [11].

Fig. 2a illustrates the accuracy rate of the algorithm in each iteration. The result is the mean value for 100 run of the algorithm with random permutation of training and test sets. It can be seen form the curves that the algorithm converges in 10 iterations. Fig. 2b demonstrates the monotonic increase of the Fisher discriminant ratio. The accuracy rate which we obtained here is relative to the Waveclus classification. Therefore if we assume the classification done by Waveclus is correct, which is not, we showed that we reached a better classification in each iteration. For better comparison of the method we should use an intracellular recording along with the extracellular recording for all the neurons located nearby the recording area. In that case we have the actual ground truth and as it is shown here the algorithm increase the accuracy and converge to the actual ground truth by iterating.

Extracellular recording of neural activity usually suffers from noise existence. Here to model this noise in the

recording we assumed it doesn't harm our spike detection process. So we added different level of noise to our already detected spikes to have different SNR. Fig. 3 shows that in our particular spikes with a peak value around 300 μV both SVMs work properly in their ideal situation till the SNR is less than 20. For noise power bigger than this value the relative accuracy of the algorithm drops till the random cluster numbers. This figure shows that the SVM with the model selection outperforms the other method without model selection.

IV. Discussion

In this paper, we proposed to use a modified version of a self-training semi-supervised SVM in spike sorting. Semi-supervised approaches have the ability to train a classifier with small amount of training data set. This method reduces the efforts of human expert to label the whole dataset manually. Here we search a classifier that gives the maximum discrimination between classes. This method showed its superiority on standard SVM approaches in a small train dataset. It is worth to mention that using the Waveclus as our ground truth does not mean that we compare our approach to that method. Here we just wanted to show that the approach could increase the accuracy of a standard SVM by iterating through the self-training method described earlier. The self-training approach with the proposed model selection showed that it could increase the accuracy of a standard SVM up to 4 percent compared to the approach without using this model selection which only increases the accuracy less than one percent.

ACKNOWLEDGEMENT

The authors would like to thanks to Dr. Emad Fatemizadeh and also the anonymous reviewers for their comments.

References

[1] S. Gibson, J. W. Judy, and D. Markovi, "Spike Sorting: The first step in decoding the brain," no. December 2011, pp. 124–143, 2012.

[2] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials.," *Network (Bristol, England)*, vol. 9, no. 4, pp. R53–78, Nov. 1998.

[3] J. C. Letelier and P. P. Weber, "Spike sorting based on discrete wavelet transform coefficients.," *Journal of neuroscience methods*, vol. 101, no. 2, pp. 93–106, Sep. 2000.

[4] R. Jacob Vogelstein, K. Murari, P. H. Thakur, C. Diehl, S. Chakrabartty, and G. Cauwenberghs, "Spike sorting with support vector machines.," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, vol. 1, pp. 546–9, Jan. 2004.

[5] W. Ding and J. Yuan, "Spike sorting based on multi-class support vector machine with superposition resolution.," *Medical & biological engineering & computing*, vol. 46, no. 2, pp. 139–45, Feb. 2008.

[6] Y. Li, C. Guan, H. Li, and Z. Chin, "A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1285–1294, 2008.

[7] Y. Jin, C. Huang, and L. Zhao, "A Semi-Supervised Learning Algorithm Based on Modified Self-training SVM," *Journal of Computers*, vol. 6, no. 7, pp. 1438–1443, Jul. 2011.

[8] S. Kim, A. Magnani, and S. Boyd, "Robust fisher discriminant analysis," *Advances in Neural Information*, vol. 1, 2006.

[9] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering.," *Neural computation*, vol. 16, no. 8, pp. 1661–87, Aug. 2004.

[10] C. Hsu and C.-J. Lin, "A comparison on methods for multi-class support vector machines," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, pp. 415–425, 2002.

[11] C. Chang and C. Lin, "LIBSVM : A Library for Support Vector Machines," pp. 1–39, 2001.