Speeding Up the Algorithm for Finding Optimal Kernel Bandwidth in Spike Train Analysis

P. Šanda[1]
[1]Institute of Physiology, Academy of Sciences of the Czech Republic
Supervisor: Doc. RNDr. Petr Lánský, CSc.

Summary

One of the important tasks in the spike train analysis is to estimate the underlying firing rate function. The aim of this article is to improve the time performance of an algorithm which can be used for the estimation.

As there is no unique way how to infer the firing rate function, several different methods have been proposed. A popular method how to estimate this function is the convolution of the spike train with Gaussian kernel with appropriate kernel bandwidth. The definition of what "appropriate" means remains a matter of discussion and a recent paper [1] proposes a method how to exactly compute optimal bandwidth under certain conditions. For large sets of spike train data the elementary version of the algorithm is unfortunately too inefficient in terms of computational time complexity.

We present a refined version of the algorithm which in turn allows us to use the original method even for large data sets.

The achieved performance improvement is demonstrated on a particular results and shows usability of proposed method.

Keywords: action potential, spike train, neural coding, firing rate, convolution, Gaussian kernel, kernel bandwidth, Brent's minimization, parallel computing, MPI

1. Introduction

Many neurophysiological studies are based on the assumption that the majority of information flow between neurons is provided by spikes. Spike trains are believed to form a neuronal code and many coding models successfully predict experimental stimuli features when only the resulting spike train is given. It has been shown that important aspects of the stimuli are coded by the neuron's firing rate, however, the exact procedure how to obtain such a rate from the experimental data differs from paper to paper and various methods were proposed [2].

Here we consider the method based on the convolution of a spike train with a fixed (Gaussian) kernel, which in result leads to a smooth estimate of firing rate and has been widely used in the past decades [3], [4], [5], [6], [7], [8]. The most difficult part of this method is the selection of the kernel bandwidth, because it affects substantially the "quality" of the estimate, while there is no obvious clue how the optimal bandwidth should be chosen. In [1] authors propose a kernel density estimator based on the mean integrated squared error principle (MISE) and formulate a precise algorithm how to infer optimal (fixed) kernel bandwidth.

For larger sets of recorded spike trains the time complexity of the straightforward version of the algorithm increases, so that it becomes unusable for online queries when studying the features of the method. Here we provide a solution, which improves the time complexity of the implementation. That at the end allows us to work with experimental data in a reasonable manner. It is also worth to note that the proposed solution does not interfere with the actual result of the original method - for the properties and comparison with other methods look in the original paper [1].

2. Methods

2.1 Original method

The firing rate is a non-negative function λ for which the integral $\int_b^a \lambda(t)dt$ gives the expected number of spikes during the time interval $[a,b)$. In the experimental recording we get only one or more trials of spike train data. The problem is how to assess the firing rate $\hat{\lambda}$, which will be as close as possible to the original λ, which is believed to stand in the background of spike discharges. The method is to convolve the spike train with a specific kernel, thus obtaining a smooth estimate of λ, for example see Fig. 1. In this case we use fixed Gaussian kernel

$$k_w(t) = \frac{1}{\sqrt{2\pi w}} \exp\left(-\frac{t^2}{2w^2}\right)$$

and the problem is reduced to the question how to select the optimal bandwidth w, so that the difference between λ and $\hat{\lambda}$ is minimal. The method itself is beyond the scope of this article, for details see [1]. What is important here is that the core of computation can be summarized in the following statement: find w^*, such that the formula (1) is minimal:

$$C(w) = \frac{1}{n^2}\sum_{i,j}\psi_w(t_i,t_j) - \frac{2}{n^2}\sum_{i\neq j}k_w(t_i-t_j) \quad (1),$$

where t_i is the time of i-th spike, n is the number of trials,

$$\psi_w(t_i,t_j) = \int_b^a k_w(t-t_i)\,k_w(t-t_j)\,dt, \quad [a,b]$$

defines the time range of the record and k_w is as the kernel used. Since we will study the Gaussian kernel the equation (1) can be rewritten as

$$2\sqrt{\pi}n^2 C(w) = \frac{N}{w} + \frac{2}{w}\sum_{i<j}\left(e^{-\frac{(t_i-t_j)^2}{4w^2}} - 2\sqrt{2}e^{-\frac{(t_i-t_j)^2}{2w^2}}\right) \quad (2),$$

where N is the number of spikes. Note that the term $2\sqrt{\pi}n^2$ is constant and has no effect on w^*. Let us denote the inner term of the sum in (2) as $E(w, t_i, t_j)$. We will denote $\mathcal{W}$ the set of possible values of w, in which we are searching. We denote the size $W = |\mathcal{W}|$ and assume that its points are equidistant.

The straightforward implementation will find the minimum value via evaluating this term in three nested loops:

(11) for $w \in \mathcal{W}$
(12) for $j \in [1,...N]$
(13) for $i \in [1,...,j-1]$
$Tmp=E(w, t_i, t_j)$
$if\ (min>tmp) min=tmp\ ,\ w^*=w$

thus obtaining the time complexity $O(N^2W)$. The selection of $\mathcal{W}$ is dependent on the interval $[a, b)$ and required precision of the optimal value. In a typical case $w^* << b - a$ we can select the upper bound of $\mathcal{W}$ to $log(b - a)$, in the case of bisection (see below) the upper bound is not so vital.

2.2 Bisecting

Now we will use that in a typical case where C forms an unimodal function, see Fig 2., though this cannot be asserted in general (such a problem can, for example, occur when searching for bandwidths is smaller than the sampling resolution of input data). Having the unimodal function and a sensible estimate of lower and upper bounds we can use any of the extremum search algorithms based on sectioning the domain. This will reduce loop (*I1*) time complexity from a linear to a logarithmic factor and as a result we obtain the complexity of $O(N^2log(W))$. As hinted above while this method helps a lot certain attention needs to be paid before its use.

2.3 Parallelization

Because the evaluation of $E(w, t_i, t_j)$ is independent of the previous computations, it is a natural target for parallelization.

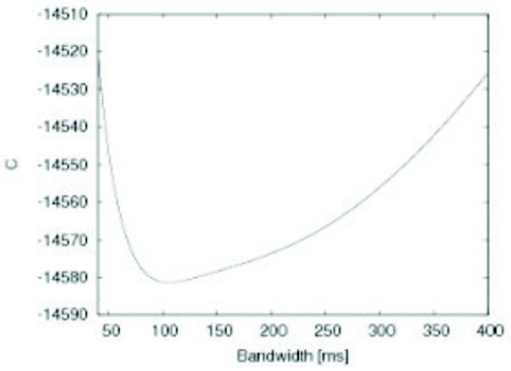

Fig. 2. Typical shape of the function for experimental spike train data.

Splitting the task for the parallel execution at the loop (*I1*) level will not allow us to use parallelization in case of the bisection run, thus we will split the task on p parts at the level (*I2*). That will give us the final estimate for the time complexity of $O(N^2log(W)/p)$.

Let us stick with the implementation details now.

2.3.1 Splitting

Since the upper bound in the loop (*I3*) is not constant, trivial splitting of (*I2*) will produce p subtasks $[1,...,N/p),[N/p,...,2N/p),...,[(p-1)N/p,...,N]$ with increasing time complexity of subtasks. At the end this would produce a situation where the first subtasks are completed having the relevant CPUs idling while the last subtasks would still be in computation.

There are more ways how to solve it - (1) move the splitting of task into (*I3*), (2) splitting p tasks in (*I2*) in a proportional way, so that each subtask has the same computational cost or (3) split (*I2*) into many small subtasks which are successively distributed to CPUs according to their load. In real-life implementation we have chosen (3) because (1) tends to produce high overhead of the parallelization engine and (2) assumes that the underlying CPUs are equivalent in performance and accessibility (that breaks in many distributed environments).

3. Results

3.1 Tuning parameters

The algorithm was implemented based on the sections above, allowing all the strategies - exhaustive search or bisecting both in sequential and parallelized versions. The language used was C++, for parallelization openMPI implementation [9] of MPI standard was chosen, for bisection we used Brent minimization algorithm [10]. In order to find the proper splitting of the subtasks we analyzed measured time demands for a different fragmentation of the tasks, see Fig. 3.

We ran the optimality search for two sets of 1000 and 18000 spikes. In the case of the larger set we could see that taking any value below f = 500 gives approximately the same time demands. On the very beginning there is a visible peak caused by the growth of the load by the parallelization maintenance (i.e. the cost of distributing subtasks starts to be larger than computation of subtasks themselves). From f = 500 we could see gradual growth caused by the insufficient fragmentation (i.e. some CPUs are needlessly idle and waiting for other unfinished subtasks).

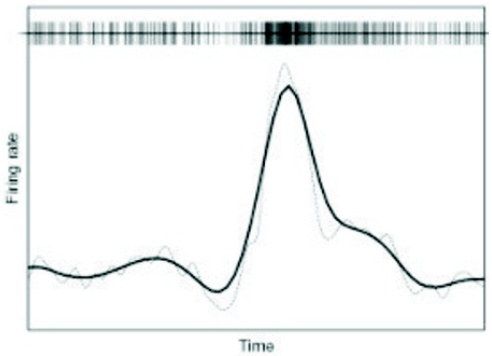

Fig.1. Illustration of the problem. The thick line is the original λ, the top line shows experimentally measured spike train generated from this function, the thin line is firing rate $\hat{\lambda}$, which we try to optimize.

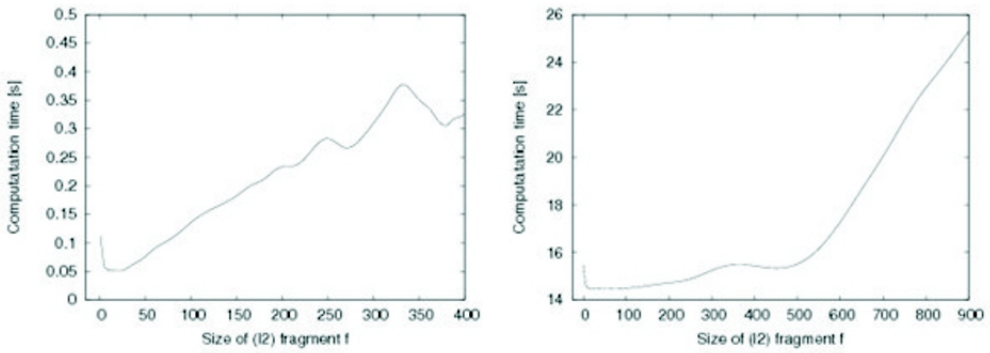

Fig. 3. Figure shows how splitting affects time performance of the computation. The left panel presents the case, where the input data were 1000 spikes, while the input for the right panel was 18000 spikes. Both sets were taken from the real experimental data, bisection was used in this case. f is the size of one (fixed) subtask, that is the number of (I3) iterations.

Tab. 1. Comparison of time demands.

Method	Time (min:sec)
Sequential search	58:41
Parallel search	4:14
Sequential bisection	2:43
Parallel bisection	0:09

As the number of spikes in the input set will decrease, this value will also decrease, as the results for the set of 1000 spikes show. Here we could see similar properties as far as the shape is concerned, but the total time needed for computation is now negligible.

This leads to the final choice of f = 100, which will be always sufficient for any larger input sets. As it can be seen in the left panel of Fig. 3, it is a reasonable value even for small sets, but that is not so important due to small total time demands. This value is, of course, dependent on the particular computational setting - in our case all tests have been done on a small cluster with 20 CPU cores.

3.2 Real time demands

For the comparison of real-time improvements we offer the table below. The input data and parameters were the same for all the tasks: 18000 spikes, [1;400] ms range for bandwidth, precision of 1 ms (W=400/1).

4. Discussion

We have proposed and implemented a parallel algorithm for optimal kernel bandwidth search which has better time performance than its "straightforward" version. Moreover when the function (1) is unimodal on the given range, we can use the bisecting version, which reduces the time even more drastically. To check the reasonable ranges, one can do the first trial run which uses only few sampling points (and in fact cannot be omitted even in normal case).

This performance boost does not play an important role in the case of small input sets of spikes, however, it is significant in case of large sets. The whole work was motivated by real demands, when experimental sets of ~20000 spikes were evaluated and, moreover, their subsets also needed to be evaluated the approximate knowledge of the function (1) shape reduced the need for a slow version of the algorithm.

At the end we proposed tuning parameters for an example cluster configuration and provided actual results of the performance improvement.

Acknowledgments

The work was supported by the grant SVV-2010-265 513.

References

[1] Shimazaki H., Shinomoto S:. Kernel bandwidth optimization in spike rate estimation. Journal of Computational Neuroscience 2010; 29: 171182.

[2] Cunningham J.P., Gilja V., Ryu S.I., Shenoy, K.V.: Methods for estimating neural firing rates, and their application to brain-machine interfaces. Neural Networks 2009; 22 (9): 12351246.

[3] Sanderson A.C.: Adaptive Filtering of Neuronal Spike Train Data. IEEE Transactions on Biomedical Engineering 1980; 27 (5): 271 274.

[4] Richmond B.J., Optican L.M., Podell M., Spitzer H.: Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. I. Response characteristics. J Neurophysiol 1987; 57 (1): 132146.

[5] Richmond B.J., Optican L.M., Spitzer H.:. Temporal encoding of twodimensional patterns by single units in primate primary visual cortex. I. Stimulus-response relations. J Neurophysiol 1990; 64 (2): 351369.

[6] Paulin M.G.: Digital filters for firing rate estimation. Biological cybernetics 1992; 66 (6): 525531.

[7] Paulin M.G., Hoffman L.F.: Optimal firing rate estimation. Neural Networks 2001; 14 (6-7): 877 881.

[8] Nawrot M., Aertsen A., Rotter S.: Single-trial estimation of neuronal firing rates: From single-neuron spike trains to population activity. Journal of Neuroscience Methods 1999; 94 (1): 81 92.

[9] Gabriel E., Fagg G.E., Bosilca G., Angskun T., Dongarra J.J., Squyres J.M. et al:. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In: Proceedings, 11th European PVM/MPI Users' Group Meeting 2004; 97104.

[10] Brent R.P.: Algorithms for minimization without derivatives. Dover Pubns; 2002.

Contact
Mgr. Pavel Šanda
Institute of Physiology,
Academy of Sciences of the Czech Republic
Vídeňská 1083
142 20 Prague 4
Czech Republic
e-mail: ps@ucw.cz