Parallel Computing of Kernel Density Estimates with MPI

Szymon Łukasik

Department of Automatic Control, Cracow University of Technology,
ul. Warszawska 24, 31-155 Cracow, Poland
Szymon.Lukasik@pk.edu.pl

Abstract. Kernel density estimation is nowadays a very popular tool for nonparametric probabilistic density estimation. One of its most important disadvantages is computational complexity of calculations needed, especially for data-based bandwidth selection and adaptation of bandwidth coefficient. The article presents parallel methods which can significantly improve calculation time. Results of using reference implementation based on *Message Passing Interface* standard in multicomputer environment are included as well as a discussion on effectiveness of parallelization.

Keywords: kernel density estimation, plug-in method, least squares cross-validation, adaptive bandwidth, parallel algorithms, MPI.

1 Introduction

Nonparametric methods find increasing number of applications in data mining problems. One of its main tools is kernel density estimation (KDE) introduced by Rosenblatt in 1956 [1] and Parzen in 1962 [2]. It has been successfully used in various applications like image processing [3], medical monitoring [4] and market analysis [5]. Theoretical and practical aspects of the estimation itself are covered in detail in numerous publications, e.g. popular monographs [6] and [7].

For n-dimensional probabilistic variable X with sample x_i of length m, kernel K and bandwidth h, the kernel density estimation evaluated for x is defined as a function:

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) . \tag{1}$$

It can be seen that computation of KDE is of complexity $O(nm)$. Thus, long samples and large number of points p for which kernel density estimates are calculated can seriously influence the estimation time. The evaluation of density estimates can be made even more resource exhausting when one wishes to use bandwidth variation as suggested by Abramson [8]. Computational complexity of additional calculations is $O(nm^2)$ in this case.

Y. Shi et al. (Eds.): ICCS 2007, Part III, LNCS 4489, pp. 726–733, 2007.

Methods of data-based optimal bandwidth calculation have also high computational demands. Second order plug-in method [9], which involves estimating second derivative of density function from given sample, is of $O(nm^2)$ complexity. Also least squares cross-validation method (LSCV) [10,11], where selecting optimal bandwidth is based on minimizing objective function *g(h)*, has the same polynomial time complexity.

Few approximation techniques have been proposed to deal with the problem of time-consuming calculations of kernel density estimates. The first of them, proposed by Silverman [12], uses fast Fourier transform (FFT). The other one applies Fast Gauss Transform (FGT) as suggested by Elgammal [13].

An alternative to those methods could use parallel processing for obtaining density function estimates. The pioneer paper in this subject is due to Racine [14]. It proves the usability of parallel computations for kernel density estimation but covers in detail only the estimation itself. Parallelization is done by dividing set of points where estimator is to be evaluated. Each processor obtains the density function estimation for approximately $\frac{p}{c}$ points (where c denotes number of CPUs involved).

The aim of this paper is to verify how parallel processing can be applied for kernel estimation, bandwidth selection and adaptation. All presented algorithms have been implemented and tested in a multicomputer environment using *MPI (Message Passing Interface)* [16]. A possibility of parallelization at the sample level, by distributing the sample and calculating approximately $\frac{m}{c}$ sums in (1) separately on each of the c CPUs for every evaluated point x, is also discussed.

2 Parallelization Methods

Following subsections will briefly present routines for obtaining kernel density estimates using parallel processing. All algorithms will be included in the form of pseudo-code with group operations treated as collective ones.

In the presented code listings subsequent notation was used: *proc_rank* is current processor number (including 0 as a root/master processor), *proc_no* - overall number of processors involved c, *sample[k]* - k*th* sample element, *sample_length* - sample length m, *x_eval[k]* - k*th* evaluation point, *f[k]* - the estimator value for *x_eval[k]* and *ev_points* - the overall number of points p where estimator is to be evaluated.

2.1 Kernel Density Estimation

Parallelization of estimation process can be achieved either by dividing set of evaluation points or sample data. Both of suggested parallelization schemes were presented using code attached below:

Parallel KDE at evaluation level

```
Broadcast(sample);
shift:=proc_rank*ev_points/proc_no;
/* evaluation of KDE for x[k+shift] using all sample elements */
for k:=1 to ev_points/proc_no do
  f[k+shift]:=Compute_KDE(x[k+shift],sample,1,sample_length);
Gather_at_root(f);
return f;
```

Parallel KDE at sample level

```
Scatter(sample);
shift:=proc_rank*sample_length/proc_no + 1;
/* evaluation of KDE for x[k] using sample starting from element
sample[shift] to element indexed by shift+sample_length/proc_no */
for k:=1 to ev_points do
  f[k]:=Compute_KDE(x[k],sample,shift,shift+sample_length/proc_no);
Aggregate_at_root(f);
return f;
```

In the first case (presented already in [14]) whole sample should be distributed among processors taking part in calculations. In the end evaluated kernel estimator values should be gathered at the master machine. The alternative is to implement parallel model at the sample level - each CPU calculates all estimator values but considering only the part of a sample. Results are globally aggregated at the master machine. In this version of the algorithm there is no need to broadcast all sample points.

2.2 Bandwidth Selection Using Plug-In Method

Let us consider commonly used two-stage direct plug-in method of Sheather-Jones [7]. Proposed parallel algorithm for plug-in bandwidth selection with parallelization at the sample level is introduced below:

Parallel plug-in bandwidth selection

```
Broadcast(sample);
shift:=proc_rank*sample_length/proc_no + 1;
/* estimation of plug-in psi functional of order i using sample starting
from element sample[shift] of length sample_length/proc_no */
for i:=6 and 4 do
  psi[i]:=Compute_SJF(i,sample,shift,shift+sample_length/proc_no);
  Aggregate_at_all(psi[i]);
h=Select_Bandwidth_at_root();
return h;
```

2.3 Bandwidth Selection Using Least Squares Cross-Validation Method

The problem of minimizing score function *g(h)* can be approached using direct evaluation of the function in selected range with a required step or by applying

some optimization technique. Underlying evaluation of the score function can be easily parallelized at the sample level. Due to lack of space, only the algorithm of direct parallel minimization of *g(h)* will be presented here. Experimental results were obtained using golden section rule [15].

Parallel LSCV bandwidth selection

```
Broadcast(sample);
shift:=proc_rank*sample_length/proc_no + 1;
g_min:=infinity, h_min:=0;
for h:=h_start to h_stop do
  /* calculation of g(h) using sample starting from
  element sample[shift] of length sample_length/proc_no */
  g:=Compute_G(h,sample,shift,shift+sample_length/proc_no);
  Aggregate_at_root(g);
  if g<g_min then g_min:=g, h_min:=h;
return h_min;
```

2.4 Adaptive Bandwidth

In order to reduce computational burden of bandwidth modifying coefficients calculation using Abramson method [8] parallel processing can be applied as presented below:

Parallel bandwidth adaptation

```
Broadcast(sample);
shift:=proc_rank*sample_length/proc_no;
/* first obtain unmodified estimates for sample points
taking into consideration all sample elements */
for k:=1 to sample_length/proc_no do
  f[k+shift]:=Compute_KDE(sample[k+shift],sample,1,sample_length);
Gather_at_all(f);
/* then calculate geometric mean of estimates using
sample_length/proc_no KDE values starting from f[1+shift] */
mean=Compute_Mean(f,1+shift,1+shift+sample_length/proc_no);
Product_at_all(mean);
/* obtain modifying coefficients */
for k:=1 to sample_length/proc_no do
  s[k+shift]:=(f[k+shift]/mean)^(-0.5);
Gather_at_all(s);
return s;
```

Modifying coefficients s_i can be easily incorporated into variable bandwidth parallel kernel density estimation (with $h_i = h \cdot s_i$) similarly as it was presented in Section 2.1.

3 Experimental Results

Reference implementation of proposed parallel algorithms was prepared to examine their efficiency. As a measure of parallelization performance following efficiency function was chosen:

$$E(c) = \frac{T(c)}{T(1)} \frac{1}{c} 100\% \, . \tag{2}$$

where: $T(c), T(1)$ are computation times for c CPUs and one CPU accordingly. Presented results were obtained for multiple execution of algorithms in Ethernet standard computer network of 6 Pentium®4 machines with MPICH-2 library used as a parallelization tool. The estimation was performed with radial normal kernel function for randomly generated one-dimensional samples. Some sample lengths correspond to those considered in [14], the others were chosen to create broader view on an average parallel estimation execution time (given in seconds). Due to space limitations only selected results were presented - full set can be obtained from author's web site (`http://www.pk.edu.pl/~szymonl`).

3.1 Parallel Kernel Density Estimation

First, the performance of two proposed parallelization schemes were tested for varying sample size and number of evaluation points. During the testing procedure $m = p$ was assumed (like in [14]). Obtained results (presented in Fig. 1) include, for reference, computation times for sequential ($c = 1$) version of the KDE.

It can be seen that results for small sample lengths are not encouraging, taking into consideration both calculation time and efficiency. The overhead, which include times of communication, synchronization and input/output operations, has a significant influence on effectiveness of multicomputer parallel system. Nevertheless, increasing problem size leads to speed-up which close to linear.

Parallelization at the evaluation level in most cases performs better than the scheme with distribution of sample elements. The time cost of global aggregation (using *MPI_Reduce*) in the multicomputer environment dominates the gain

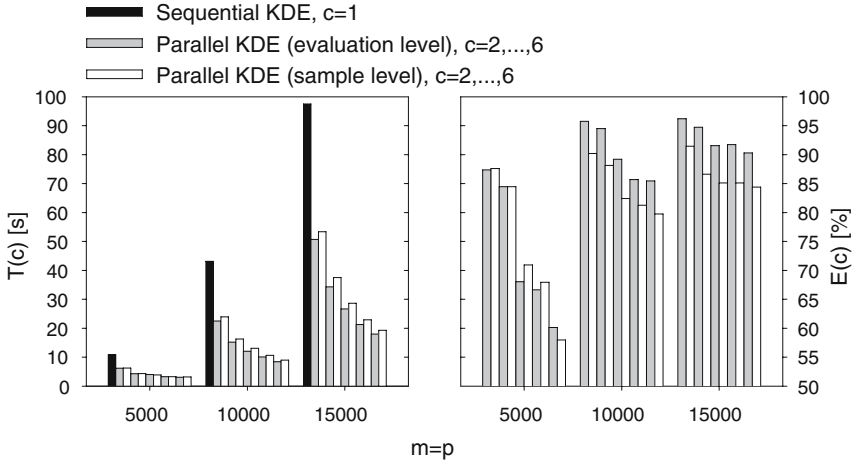

Fig. 1. Execution times and efficiency of parallel kernel density estimation methods

achieved by distributing smaller amount of data at the first phase of estimation. The proposed parallelization scheme can be applied successfully when $m \gg p$, though. The difference between performances of the presented methods is under these assumptions neglectable. For instance, if $m = 50000$, $p = 5000$ and $c = 3$ algorithm using parallelization at the sample level is only 0.3 s slower (which is a fraction of 107.8 s processing time for $c = 1$).

3.2 Parallel Bandwidth Selectors

Observing high computational complexity of data-based bandwidth selectors leads to preliminary conclusion, that parallel processing in the case of both plug-in and LSCV methods will give significant improvement in calculation time. Results of test performed for variable sample lengths presented in Fig. 2 confirm it. When ruling out cases where too many CPUs were assigned to a very small task, relatively high parallelization effectiveness can be also noticed. Advantage of using parallel processing is in case of least squares cross validation more considerable - it is a direct effect of substantial computational demands raised by this method.

3.3 Parallel Bandwidth Adaptation

Finally parallelized routine for bandwidth adaptation was under investigation. The test was conducted with $m = p$. Considered scheme of algorithm involved also parallelizing the density estimation itself at evaluation level. Results of the test are enclosed in Fig. 3.

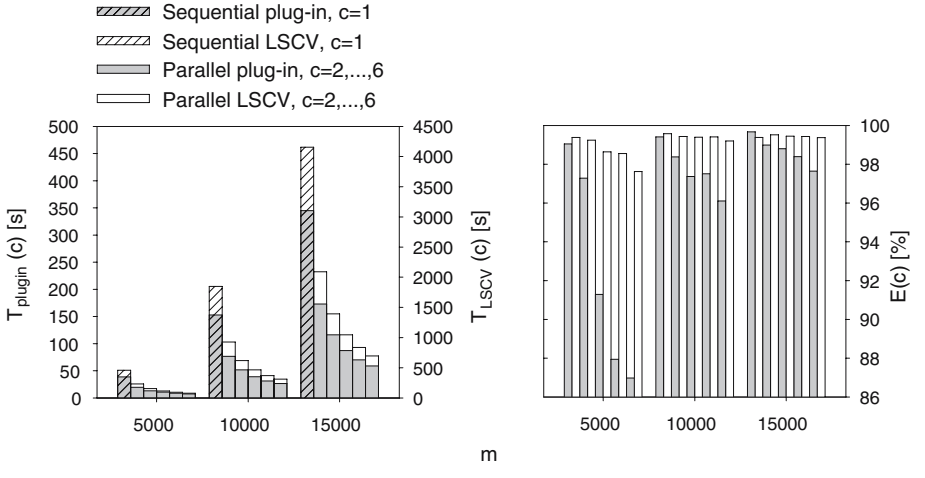

Fig. 2. Execution times and efficiency of parallel plug-in method and parallel least squares cross-validation

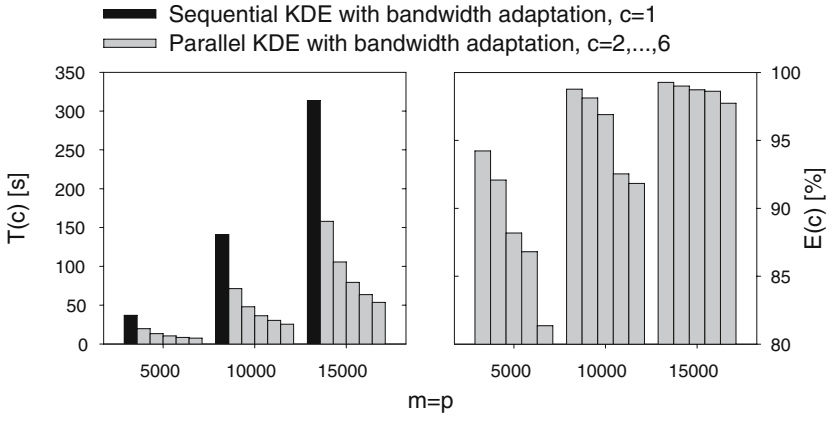

Fig. 3. Execution times and efficiency of parallel KDE with bandwidth adaptation

In contrast to parallel kernel estimation with fixed bandwidth, performing parallel bandwidth adaptation can be judged as highly effective, even when relatively small sample is under consideration. The obtained speed-up of computation, with increasing CPUs number, is in this case close to linear.

4 Conclusion

The article positively verifies the possibility of applying parallel algorithms for speeding up time-consuming process of kernel density estimation. Beneficial effects of parallelization, already proved for KDE [14] itself, were also confirmed for popular data dependent bandwidth selectors and bandwidth adaptation. Moreover, for fixed sample size, gain of using parallel calculation is in those cases more significant.

The option of using the alternative parallelization scheme for kernel density estimation - at the sample level - was also under consideration. Empirical studies proved that the proposed method is, in general case, not as effective as parallel evaluation of density estimates.

As in every parallel processing application it is important to note, that proper care has to be taken of granularity of calculation if one wishes to obtain effective use of engaged resources. Parallel bandwidth selectors and bandwidth adaptation involve substantial number of collective operations executed between calculation cycles so it would be also advisable to use load-balancing in order to eliminate overhead generated by MPI tasks synchronization routine (*MPI_Barrier*).

Acknowledgments. I would like to thank Prof. Zbigniew Kokosiński for his support and comments on an earlier version of this paper. I'm also grateful for valuable suggestions of anonymous reviewers.

References

1. Rosenblatt, M.: Remarks on Some Nonparametric Estimates of a Density Function. Annals of Mathematical Statistics **27** (1956) 832–837
2. Parzen, E.: On Estimation of a Probability Density Function and Mode. Annals of Mathematical Statistics **33** (1962) 1065–1076
3. Mittal, A., Paragios, N.: Motion-Based Background Subtraction Using Adaptive Kernel Density Estimation. IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2** (2004) 302–309
4. Cerrito, P.B., Barnes, G.R.: The Use of Kernel Density Estimators to Monitor Protocol Compliance. Proceedings of the Twenty-Fifth Annual SAS®Users Group International Conference **SUGI25** (2000) paper no. 273
5. Donthu N., Rust, R.T.: Estimating Geographic Customers Densities using Kernel Density Estimation. Marketing Science **8** (1989) 191–203
6. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
7. Wand, M.P., Jones, M.C.: Kernel Smoothing. Chapman and Hall, London (1995)
8. Abramson, I.: On Bandwidth Variation in Kernel Estimates - a Square Root Law. The Annals of Statistics **10** (1982) 1217-1223
9. Sheather, S.J., Jones, M.C.: A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation. Journal of the Royal Statistical Society. Series B. Methodological **53/3** (1991) 683–690
10. Rudemo, M.: Empirical Choice of Histograms and Kernel Density Estimators. Scandinavian Journal of Statistics **9** (1982) 65-78
11. Bowman, A.W.: An Alternative Method of Cross-Validation for the Smoothing of Density Estimates. Biometrika **71** (1984) 353–360
12. Silverman, B.W.: Algorithm AS 176: Kernel Density Estimation Using the Fast Fourier Transform. Applied Statistics **31/1** (1982) 93–99
13. Elgammal, A., Duraiswami, R., Davis, L.S.: Efficient Kernel Density Estimation Using the Fast Gauss Transform with Applications to Color Modeling and Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25/11** (2003) 1499–1504
14. Racine, J.: Parallel distributed kernel estimation. Computational Statistics and Data Analysis **40/2** (2002) 293–302
15. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, London (1981)
16. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra J.: MPI: The Complete Reference. The MIT Press, Cambridge (1996)