ParKerC: Toolbox for Parallel Kernel Clustering Methods

Sandrine Mouysset*, Ronan Guivarch*

*University of Toulouse, IRIT,
{sandrine.mouysset,ronan.guivarch}@irit.fr

Keywords: Kernel methods, overlapping domain decomposition, clustering, parallel computing.

Abstract

A large variety of fields such as biology, information retrieval, image segmentation needs unsupervised methods able to gather data without a priori information on shapes or locality. By investigating a parallel strategy based on overlapping domain decomposition, we present a toolbox which is a parallel implementation of two fully unsupervised kernel methods respectively based on density-based properties and spectral properties in order to treat large data sets in fields of pattern recognition.

1 Introduction

Many fields from Social Science to Medicine and Biology generate large amount of data to analyze. Clustering aims at partitioning data sets in clusters in order to group data points with a similarity (or affinity) measure. Kernel clustering methods have become an increasingly popular tool for machine learning [1]. These methods rely on the use of positive definite kernel functions which enable them to operate in a high-dimensional feature space and provide, in particular, interesting spectral properties [2]. The kernel representation implies defining a fully similarity matrix which can be computational costly when dealing with large amount of data. Parallel implementation are thus considered. Some efficient implementations were investigated for several supervised kernel methods such as Support Vector Machines or Gaussian Processes [3, 4] and we propose to extend this purpose for unsupervised kernel methods gathered in a Toolbox called ParKerC.

In this paper, we first focus on the parallel strategy of the ParKerc toolbox based on domain decomposition to treat large data sets. Then, a fully unsupervised version of two widely used kernel based clustering methods [5] - Mean shift [6] and Spectral clustering [2] - based respectively on kernel density estimation and eigen decomposition of similarity matrix is proposed. Finally, some tests on both benchmark data sets and image segmentation are performed.

2 ParKerC Toolbox

In this section, we present the principle of the proposed toolbox: the parallel strategy of domain decomposition with overlapping and how to set the overlapping bandwidth.

The code of the ParKerC toolbox is written in FORTRAN 90 using MPI library to handle the communication between processors.

2.1 Principle

The principle of the parallel toolbox is based on domain decomposition with overlaps. Let consider a data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$. This data set is included in a domain. We divide the domain in q sub-domains, thus defining q sub-sets. By assigning a sub-domain to each processor, a processor applies independently the clustering algorithm on the corresponding sub-set and provides a local partition.

For each sub-domain, the number of clusters is automatically determined, within a maximum number of clusters. This heuristic avoids us to fix the targeted number of clusters.

The final number of clusters, noted k, will be provided after the grouping step.

This grouping step is dedicated to link the local partitions from the sub-domains thanks to the overlap and the following transitive relation: $\forall x_{i_1}, x_{i_2}, x_{i_3} \in S$,

$$\begin{array}{ll} \text{if} & x_{i_1}, x_{i_2} \in C^1 \text{ and } x_{i_2}, x_{i_3} \in C^2 \\ \text{then} & C^1 \cup C^2 = P \text{ and } x_{i_1}, x_{i_2}, x_{i_3} \in P \end{array} \quad (1)$$

where C^1 and C^2 are two distinct clusters and P is a larger cluster which includes both C^1 and C^2. By applying this transitive relation (1) on the overlap, the connection between subsets of data is established and provides a global partition.

We can implement this algorithm using a Master-Slave paradigm as summarized in Algorithms 1 and 2.

2.2 Overlapping bandwidth

The idea is to consider an uniform distribution where the n data points are equidistant. To define the uniform distance between the points, we consider both the dimension of the problem as well as the density of points in the given p-th dimensional data set. In fact, the data set S is included in a p-dimensional box bounded by ρ_d the largest distance between pairs of points in each dimension d of S:

$$\rho_d = \max_{1 \leq i,j \leq n} |x_{id} - x_{jd}|, \ \forall d \in \{1, .., p\}.$$

Algorithm 1 Parallel Algorithm: Master

1: Pre-processing step
   1.1 read the global data and the parameters
   1.2 compute the uniform distance δ (see equation 2)
   1.3 compute the overlapping bandwidth α
   1.4 split the data into q sub-sets
2: Send δ and the data sub-sets to the slaves
3: Perform the Clustering Algorithm on its sub-set
4: Receive the local partitions and the number of clusters from each slave
5: Grouping Step
   5.1 Gather the local partitions in a global partition with the transitive relation (1)
   5.2 Output a partition of the whole data set S and the final number of clusters k

Algorithm 2 Parallel Algorithm: Slave

1: Receive δ and its data sub-set from the Master
2: Perform the Clustering Algorithm on its sub-set
3: Send its local partition and its number of clusters to the Master

So the uniform distance, noted δ, could be defined as follows:

$$\delta = \left(\frac{\prod_{i=1}^{p} \rho_i}{n} \right)^{\frac{1}{p}}. \tag{2}$$

From this distance, the overlapping bandwidth α is set as a multiple of δ in order to be able to merge the clusters with the transitive relation.

In the following, we present two widely used kernel clustering methods, check their suitability with a domain decomposition strategy and adapt their inherent parameters. The first method, Mean Shift in section 3, relies on a non-parametric estimator of density gradient for locating the maxima of the density function called mode. The second method, spectral clustering in section 4, based on eigen-decomposition of kernel affinity matrix is used in pattern recognition or image segmentation to cluster non-convex domains without a priori on the shapes.

3 Mean shift

Introduced by Fukunaga and Hostetler [6], mean shift method considers the points in the feature space as a probability density function. Dense regions in feature space corresponds to local maxima (or mode). The clusters are then associated with the modes.

3.1 Algorithm

Mean shift associates each data point in $\mathbb{R}^p$ with the nearby peak of the data set's probability density function. For each data point, mean shift defines a window around it and computes the mean of the data points which belong to this window. Then it shifts the center of the window to the mean and repeats the algorithm till it converges. In other words, the window shifts to a more denser region of the data set.

Algorithm 3 Mean shift Algorithm

Input: data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$, bandwidth h
At the iteration (t), for each data point $x_i \in S$,

1. Compute mean shift vector $m(x_i)^{(t)}$
2. Move the density estimation window to $m(x_i)^{(t)}$
3. repeat till convergence i.e $\|m(x_i)^{(t+1)} - m(x_i)^{(t)}\| \leq threshold$

3.2 Justification

Mean shift relies on kernel density estimation. Kernel density estimation [7] (also called the Parzen window technique [8]) is the most popular non parametric density estimation method.

Given a kernel K, a bandwidth parameter h, kernel density estimator for a given set of n p-dimensional points is:

$$f(x) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(h, x - x_i\right) \tag{3}$$

Mean shift is based on Gradient ascent on the density contour [9]. So, for each data point, we perform gradient ascent on the local estimated density until convergence.

So:

$$\nabla f(x) = \frac{1}{nh^p} \sum_{i=1}^{n} K'\left(h, x - x_i\right) \tag{4}$$

where $K'(h, x)$ is the derivative of $K(h, x)$. The stationary points obtained via gradient ascent represent the modes of the density function. All points associated with the same stationary point belong to the same cluster. By assuming that $g(h, x) = -K'(h, x)$, the following quantity $m(x)$, called mean shift, is computed as follows:

$$m(x) = \frac{\sum_{i=1}^{n} g\left(h, x - x_i\right) x_i}{\sum_{i=1}^{n} g\left(h, x - x_i\right)} - x \tag{5}$$

With this strategy of searching the maximum of local density, this method does not require to fix the number of clusters.

This implies that mean shift can be run in sub-sets of S and if a cluster relies on several sub-domains then the transitive relation (1) will merge the clusters.

3.3 Tuning parameters

As said in the previous section, the number of clusters is automatically defined. But Mean Shift is sensitive to the selection of bandwidth h. A small h can slow down the convergence whereas a larger one can speed up the convergence and merge two modes. We can define it automatically by considering the bandwidth h as a multiple of the uniform distance δ defined by (2). The initialization of Mean Shift is done by choosing randomly observation in data set.

4 Spectral clustering

Spectral clustering uses eigenvectors of a matrix, called Gaussian affinity matrix, in order to define a low-dimensional space in which data points can be clustered.

4.1 Algorithm

Assume that the number k of targeted clusters is known (we will see how to automatically determine it). Algorithm 4 presents the different steps of spectral clustering. First, the spectral clustering consists in constructing the affinity matrix based on the Gaussian affinity measure between points of the data set S. After a normalization step, the k largest eigenvectors are extracted. So every data point x_i is plotted in a spectral embedding space of $\mathbb{R}^k$ and the clustering is made in this space by applying K-means method. Finally, thanks to an equivalence relation (*step 6.*), the final partition of data set is defined from the clustering in the embedded space.

Algorithm 4 Spectral Clustering Algorithm

Input: data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$, number of clusters k

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by:

$$A_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{(\sigma/2)^2}\right) \text{ if } i \neq j, \\ 0 \text{ otherwise,} \end{cases} \quad (6)$$

2. Construct the normalized matrix: $L = D^{-1/2} A D^{-1/2}$ with $D_{i,i} = \sum_{j=1}^{n} A_{ij}$,
3. Assemble the matrix $X = [X_1 X_2 .. X_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors associated with the k largest eigenvalues of L,
4. Form the matrix Y by normalizing each row in the $n \times k$ matrix X,
5. Treat each row of Y as a point in $\mathbb{R}^k$, and group them in k clusters via the K-means method,
6. Assign the original point x_i to cluster j when row i of matrix Y belongs to cluster j.

4.2 Justification

The sub-domain decomposition implies to study if the decomposition will have an impact on the final partition. From the definitions of both the Gaussian affinity A_{ij} between two data points x_i and x_j and the Heat kernel $K(t, x) = (4\pi t)^{-\frac{p}{2}} \exp\left(-\|x\|^2/4t\right)$ in free space $\mathbb{R}_+^* \times \mathbb{R}^p$, we can interpret the Gaussian affinity matrix defined by (6) as discretization of heat kernel by the following equation:

$$A_{ij} = (2\pi\sigma^2)^{\frac{p}{2}} K\left(\sigma^2/2, x_i - x_j\right). \quad (7)$$

So, we can prove that eigenfunctions for bounded and free space Heat equation are asymptotically close [10]. With Finite Elements theory, we can also prove that the difference between eigenvectors of A and discretized eigenfunctions of K_t is of an order of the distance between points include inside the same cluster. This means that applying spectral clustering into subdomains resumes in restricting the support of these L^2 eigenfunctions which have a geometrical property: their supports are included in only one connected component.

Thus, the overlapping domain decomposition does not alter the global partition because the eigenvectors carry the geometrical property and so, the clustering property.

4.3 Tuning parameters

Spectral clustering depends on two parameters: the Gaussian affinity parameter σ and the number of clusters k. The Gaussian affinity matrix (6) is widely used and depends on a free parameter σ. It is known that this parameter affects the results in spectral clustering and spectral embedding [11]. From the definition of δ defined by (2) in which we consider the case of an uniform distribution in the sense that all pair of points are separated by the same distance δ in the box of edge size D_{max}, we can state that clusters may exist if there are points that are at a distance no more than a fraction of δ.
The number of clusters k is determined by partitioning the data set from 2 to a maximum number, denoted *k_max* and then the similarity matrix is reordered per cluster. So k is defined from a measure based on the ratio of the Frobenius norms of the affinity measure between distinct clusters and within clusters [12]. The value of k that minimizes this ratio becomes the optimal number of clusters.

4.4 Efficient method to compute dominant eigenvectors for the Spectral Clustering Algorithm

In step 3 of algorithm 4, we have to compute the *k_max* eigenvectors corresponding to the *k_max* dominant eigenvalues of the matrix L (normalization of the affinity matrix A). To determine these *k_max* dominant eigenvectors, one can use `LAPACK` library, especially the routine `DSYEV` that computes all the eigenvalues and eigenvector of a symmetric matrix.

But why compute all the eigenvectors when we need only the *k_max* dominant ones? An alternative to the routine `DSYEV` is the subspace iteration method [13]. There is different algorithms corresponding to this method. Algorithm 5 is an efficient and practically one.

This method can be seen as the power method that computes the dominant eigenpair extended to a set of vectors.

An important parameter of this method is the size m of the subspace: a larger m accelerates the convergence (decreases the number of iterations) but requires a larger memory size but above all a more expensive computation of the spectral decomposition of the Rayleigh quotient H of size $m \times m$.

Numerical comparisons between the subspace iteration method and the use of `DSYEV` are presented in section 5.1.

Algorithm 5 Subspace iteration method with Raleigh-Ritz projection

1: Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$, *k_max* the number of required dominant eigenvectors, m, subspace size ($>= k\_max$)
2: Output: *k_max* dominant eigenvectors V_{out}
3: Generate an initial set of m orthonormal vectors $V \in \mathbb{R}^{n \times m}$;
4: **repeat**
5: Compute Y such that $Y = A \cdot V$
6: $V \longleftarrow$ orthonormalisation of the columns of Y
7: *Rayleigh-Ritz projection* applied on matrix A and orthonormal vectors V
7.1 compute the Rayleigh quotient $H = V^T \cdot A \cdot V$.
7.2 compute the spectral decomposition of H
$X \cdot \Lambda_{out} \cdot X^T = H$.
7.3 compute $V_{out} = V \cdot X$.
8: *Convergence analysis step*: save eigenpairs that have converged
9: **until** (not k_max_dominant_eigenvalues_computed)

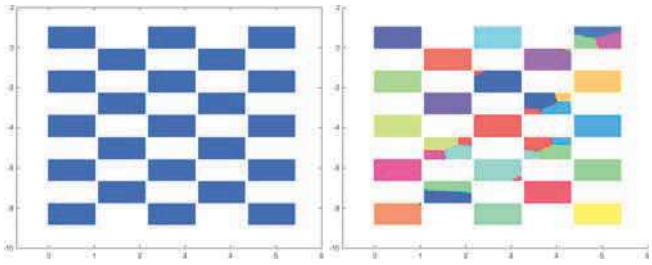


Figure 1. Data set (60^2 points per block) and K-means results (one color per cluster)

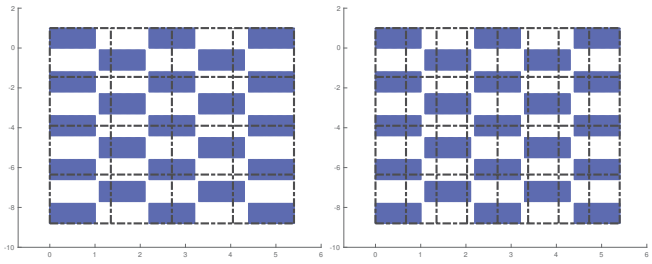


Figure 2. Data set for performance testing: decomposition by 4×4 and 4×8 domains in block dash-dot lines (60^2 points per block)

5 Results

We present in this section some results: first, in subsection 5.1, performance results to compare the subspace iteration method and `DSYEV` to compute the eigenvectors ; then, in subsections 5.2 and 5.3, clustering results to validate our parallel approach on both data sets and images.

5.1 Comparison between the subspace iteration method and `DSYEV`

To compare the two methods, the data sets is a checkerboard with 23 clusters (see Figure 1). There are 5 data sets where the number of points by block is 60^2, 70^2, 80^2, 90^2 and 100^2. The topology of this example is interesting because it presents clusters that are not easily linearly separated as shown in Figure 1 when K-means is applied on it by randomly choosing 23 observations from the dataset as initial centers.

Table 1 gives the time to perform the spectral clustering with `DSYEV` and the subspace iteration method for two decomposition of the domain: 4×4 and 4×8 (see Figure 2); we assign a processor by sub-domain. For each data set size and domain decomposition, we indicate the maximum number of points on a sub-domain; that gives us an indication of the maximum amount of computation by processor.

As we are looking for k_max clusters by sub-domain, we have set the size of the subspace in the subspace iteration method at $m = 10 \times k\_max$.

As we can observe in Table 1, the subspace iteration method gives better results with all the 4×4 decomposition examples and for two 4×8 decomposition ones when we reached a certain amount of points by sub-domain. We can also notice that this maximum number of sub-domain is just an indication: for a quasi-equivalent maximum number of points (6800 for the $\{60^2,\ 4 \times 4\}$ case and 7000 for $\{80^2,\ 4 \times 8\}$ ones), the computation times are not the same at all. We must take into account the properties of the affinity matrix that depend on the distribution of points in the sub-domain.

This is an interesting result, especially when we will consider image clustering, because with images, each pixel is considered as a point and then every sub-domain has the same large number of points to process.

5.2 Application to clustering data sets

To validate our domain decomposition approach, we present in Figure 3 some results on four different data sets from a clustering benchmark `https://cs.joensuu.fi/sipu/datasets/`. The characteristics of the data sets are summarized in the table 2.

Each problem is solved by using spectral clustering or mean shift methods, in sequential (1×1) and in parallel (2×2 square-partitioning). In these experiments, which compare the two methods, the number of sub-domains is small. Of course, this number can be increased and will depend essentially on the number of processors available.

We tuned the parameters as explained in the theoretical sections: δ is computed as described in subsection 2.2 ; we used it to define the Gaussian affinity parameter σ for the spectral clustering (subsection 4.3), the bandwidth for the mean shift as h (3.3) and the overlapping bandwidth α (2.2).

We can notice that spectral clustering gives expected results for all problems both in sequential and in parallel (the number of exhibited clusters are between parenthesis). In contrary, mean shift have good results when the clusters are convex but

number of points by sub-domain	4×4			4×8		
	max	DSYEV	SubIter	max	DSYEV	SubIter
60^2	6800	476 s	**426 s**	4000	**137 s**	191s
70^2	9100	921 s	**725 s**	5500	**355 s**	404 s
80^2	11800	2098 s	**1328 s**	7000	**677 s**	761 s
90^2	15000	5106 s	**2525 s**	9000	1554 s	**1523 s**
100^2	18400	10244 s	**5278 s**	11000	3955 s	**2619 s**

Table 1. Comparisons between DSYEV and SubIter: time in seconds, max is the maximum number of points on a sub-domain

	unbalance	spiral	Compound3	jain
Nb points	6500	312	219	373
Nb Clusters	8	3	3	2

Table 2. Number of points and number of clusters of the four selected examples

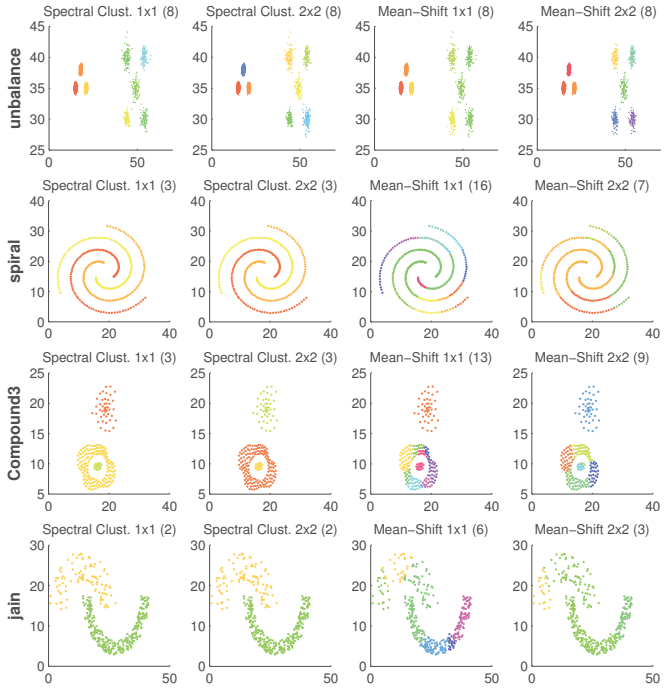


Figure 3. Examples of data set segmentation

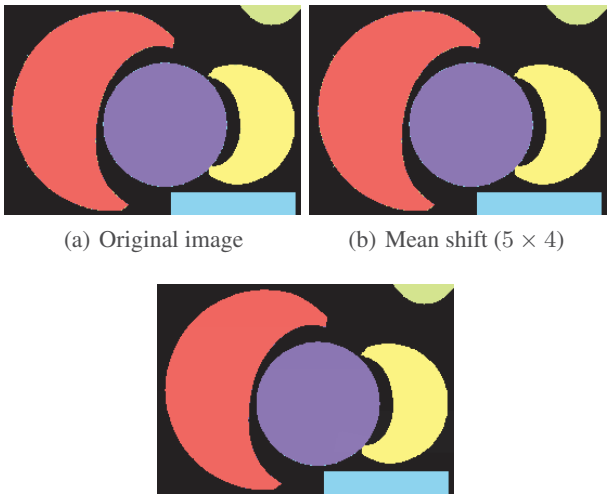
poor ones with non-convex clusters; we tried different values of $h = C \times \delta$ and, among the results, we present the best ones. With no post-treatment (for instance, merging using transitivity), the results of mean shift method on non convex shapes are not exploitable.

5.3 Application to image segmentation

For image segmentation, the domain decomposition is applied geometrically on the image and also on the brightness distribution (or color levels). Thus, the kernel function is applied on geometrical and brightness/color data. The kernel function K at a pixel x will be decomposed according to the spatial and color vectors as:

$$K_{h_r,h_s}(x) = K(h_s, x^s)\, K(h_r, x^r) \tag{8}$$

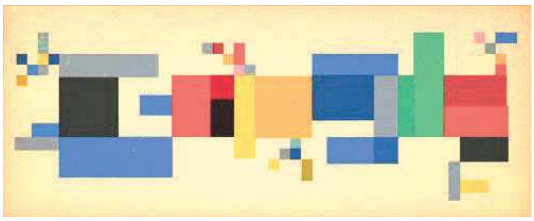
where $x^s \in \mathbb{R}^2$ is the spatial vector of the pixel x and $x^r \in \mathbb{R}^3$ is the 3D color level vector of x and h_s and h_r are respectively the spatial and color parameters. We apply in parallel both spectral clustering and mean shift on a geometrical picture as shown in Figure 4 and Figure 5. From the partition of both methods, we compute the mean color of each cluster (the average color of all the points which belong to the same cluster) and we plot this mean color at the corresponding geometrical pixels. Figure 4 represents distinct colored geometrical shapes which are well separated. Mean Shift and Spectral Clustering segment all the shapes and color easily. The second example inspired by the works of the Swiss artist Sophie Taeuber-Arp has different colored geometrical blocks which are not well separated. We can see that the segmentation is not perfect and some colored clusters are merged. Nevertheless the main shapes of the painting can be distinguished. For image application, a post processing step must be investigated to merge clusters that share the same geometrical and/or color information to reduce the number of clusters.



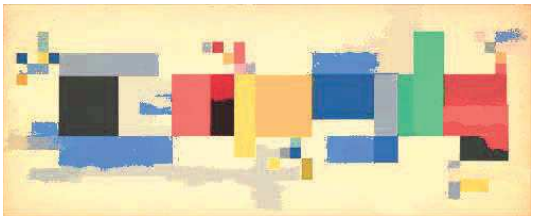
(a) Original image (b) Mean shift (5×4)

(c) Spectral clust. (5×4)

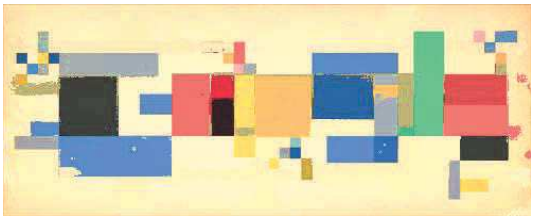
Figure 4. Results of parallel clustering methods on an image (275×194)



(a) Original painting



(b) Mean shift (5×4) {1605 clusters}



(c) Spectral clustering (5×4) {1400 clusters}

Figure 5. Results of parallel clustering methods on a painting (500×200)

6 Conclusion and perspectives

We have validated the two parallel methods, both with data sets and images. We will continue our tests on larger problems to fully validate our FORTRAN code. These kernel methods offer different clustering analysis. Spectral clustering, based on connected components, can partition clusters with uniform distribution and circular shapes. Mean shift, relied on a density-based approach, can separate clusters even when they are connected by few points.

On the theoretical part, we want to investigate some other expressions of δ to better approximate the data distribution (for instance replace the euclidean distance by $L1-$norm) for a better estimation of the tuning parameters. We also plan to investigate different kernel functions such as sigmoïd and polynomial kernels and add other kernels methods in the toolbox such as Kernel K-means [14].

Acknowledgements

This work was performed using HPC resources from CALMIP (Grant 2018-[p0989]).

References

[1] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, 2008.

[2] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," *Proc.Adv.Neural Info.Processing Systems*, 2002.

[3] R. Díaz-Morales and Á. Navia-Vázquez, "Efficient parallel implementation of kernel methods," *Neurocomputing*, vol. 191, pp. 175–186, 2016.

[4] R. Langone, R. Mall, C. Alzate, and J. A. Suykens, "Kernel spectral clustering and applications," in *Unsupervised Learning Algorithms*, pp. 135–161, Springer, 2016.

[5] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[6] F. K. and H. L.D., "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, 1975.

[7] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, 1956.

[8] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, 1962.

[9] D. Comaniciu and P. Meer, "Mean shift analysis and applications," *Proceedings of the 7th IEEE International Conference on Computer Vision*, 1999.

[10] S. Mouysset, J. Noailles, D. Ruiz, and C. Tauber, "Spectral clustering: interpretation and gaussian parameter," *Data Analysis, Machine Learning and Knowledge Discovery*, 2014.

[11] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[12] S. Mouysset, R. D. Noailles, J., and R. Guivarch, "On a strategy for spectral clustering with parallel computation," *High Performance Computing for Computational Science: 9th International Conference*, 2010.

[13] G. W. Stewart, *Matrix Algorithms: Volume 2, Eigensystems*. Society for Industrial and Applied Mathematics (SIAM), 2001.

[14] S. Wang, A. Gittens, and M. W. Mahoney, "Scalable kernel k-means clustering with nyström approximation: relative-error bounds," 2017.