Parallelized Sequential Minimal Optimization for Enhanced Support Vector Clustering Efficiency

DRID Abou Bakr Seddik
Computer science departement
Mohamed Khider university
LESIA Laboratory
Biskra, Algeria
drid.abs@univ-biskra.dz

Pr. Djeffal Abdelhamid
Computer science departement
Mohamed Khider university
LESIA Laboratory
Biskra, Algeria
abdelhamid.djeffal@univ-biskra.dz

***Abstract*—Support Vector Clustering (SVC) is a valuable tool for unsupervised data analysis, but its computational demands have limited its application to large datasets. In this study, we propose a parallelized implementation of SVC based on an adapted Sequential Minimal Optimization (SMO) algorithm. Our approach aims to reduce training time while maintaining clustering accuracy. Through a series of simulations on a single computer, we demonstrate the significant efficiency gains achieved through parallelization. Notably, we find that tuning the kernel width parameter (q) can strike a balance between execution time and clustering precision. These preliminary results hold promise for the scalability of SVC, with opportunities for further parameter optimization. This work contributes to the field by offering an efficient and scalable SVC solution, opening doors to its broader utilization across domains.**

***Index Terms*—Clustering, Support vector clustering (SVC), support vector machine (SVM), sequential minimal optimization (SVM), Parallelization.**

I. Introduction

Support Vector Clustering (SVC), is a relatively new kernel-based algorithm, proposed in 2000 by Ben-Hur et al. [1], mainly inspired by the Support Vector Machine (SVM) method by Vapnik [2], it accurately groups data point into clusters based on two main steps training and labeling. The SVC training step uses the kernel trick to find and train a mathematics that form in the hyperspace a minimum hypersphere that enclose most of the data points, then, by mapping back to the input space, the obtained hypersphere generates the cluster boundaries. Labeling step compute the adjacency matrix based on the direct connection test between data points, then it labels the whole data points in terms of the adjacency matrix [1]. Due to its ability of generating complex cluster boundaries, and its smoothness on dealing with outliers, also, the no necessity of predefining the number of clusters, the SVC method outperforms clustering conventional methods. However, dealing with large data sets shows a significant time consumption, both in training and labeling step, and it presents great challenges. In addition, the resulting clustering is very sensitive to the selection of some parameters, which are basically done in a supervised way.

Trying to solve these bottlenecks, many works have been proposed in the literature with various optimization concepts. In a comprehensive survey conducted by Li et al. [3], they systematically summarized and categorized a significant body of research conducted up until mid-2014. Upon analyzing these works, it becomes evident that the majority of them, owing to the algorithm's inherent nature, heavily rely on mathematical concepts, often overlooking straightforward optimization solutions, such as parallelization. In this study, we aim to address this gap by proposing a parallelized version of the SMO algorithm for the SVC method. Our primary goal is to reduce the time cost associated with the training step. We believe that by the parallelization this algorithm, we can significantly enhance its efficiency and scalability.

II. A review of Support Vector Clustering

Based on Support Vector Machine (SVM) by Vapnik [2], support vector domain description (SVDD) by Tax and Duin [4] and Estimating the Support of a High-Dimensional Distribution by Bernhard Scholkopf [5], Ben-Hurr proposed a robust mathematical kernel-based method called support vector clustering (SVC) [1]. The method task is to accurately label a set of data points in an unsupervised way, throw two main steps. Training step to construct a trained kernel radius function, and the label step to assign a cluster index for each data point. In this section, we present an overview of SVC method principals.

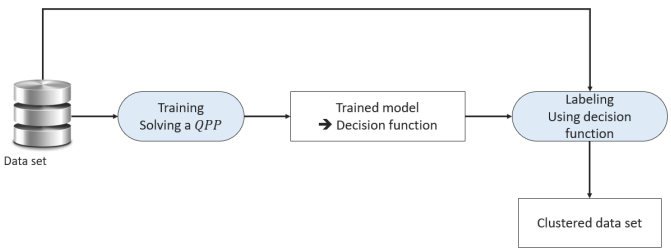

Fig. 1: SVC main steps.

A. SVC training

Two major approaches are proposed in the literature to define the domain of novelty in this step, the large margin

hyperplane (LMH) [5] and the minimum englobing sphere (MES) [1], [4].

1) LMH: This approach tries to define the domain of novelty by learning an optimal hyperplane that can separate the data samples from the origin with the maximum distance.

2) MES: Is the mainly used technique in literature, it aims to find a hypersphere in the hyperspace that can enclose all data points with a minimal radius, then, by mapping back to the input space, the obtained hypersphere generates the cluster boundaries.

The quest of finding the optimal hyperplane (in LMH) or the minimal hypersphere (in MES) equates to solving a Quadratic Programming Problem (QPP).The QPP of the training step can be solved by the Sequential Minimal Optimization (SMO) algorithm of Platt [5] which was proposed as an efficient tool for SVM training in the supervised case, and adapted to the SVC problem by Scholkopf [5].

The result of solving this QPP serves as the decision function within the SVC algorithm.

$$f(x) = K(x,x) - 2\sum_{i=1}^{N}\beta_i K(x_i,x) + \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K(x_i,x_j) \tag{1}$$

B. SVC labeling (cluster assignment):

The obtained decision function 1 indicates only if the tested data point is inside one of the clusters or not, it does not differentiate between points that belong to different clusters [1]. To do so, a simple graphical direct connection test can be used to assign each sample to the appropriate cluster. For any two points x_i, x_j, and using the function (3), we check the m segmers on the line segment that connect their images in the hyperspace, if all the m segmers lies in the hypersphere, x_i, x_j should be labeled with the same cluster, otherwise, they will be assigned to two different clusters.

III. Most important current works on SVC

As it is mentioned in the introduction, the main drawback of SVC method is the computational time requirement in both, training and labeling step. Thus, almost of the theoretical research works on the literature are focusing on minimizing time consumption with the preservation -or even improvement in some propositions- of the method's accuracy.

H. Li et al. [3] have classified the theoretical contributions into four main classes: data space reduction, solving dual problem, improving cluster labeling methods and parameter selection and optimization.

A. Data space reduction

Given that we are dealing with huge amount of data, reducing it or selecting the most relevant data points shown that it has a great effect on reducing computational time requirement. According to [3], [4] only a few data points are needed to define $f(x)$, these points are the SVs. So, the aim is to pinpoint these SVs and use only them to train $f(x)$. Many algorithms proposed to eliminate a large data point proportion and use only the remaining small part –which implicitly contains the SVs- to train the model.

Based on local geometrical and statistical information method proposed Y. Li et al. [6], H. Li et al. [3] proposed a border-edge pattern selection (BEPS) method to identify the boundaries points. Their algorithm tries to localize points with all of their neighbors or almost of them are locating on one side (upside or downside) of the tangent plane passing throw that points, depending on the curvature of the surface.

Applying BEPS result on a reduced data set, which significantly reduce the amount of training time. However, the necessity to set some parameters as: the thresh hold γ which is used to control the curvature of the surface, and test if a data point is a boundary point, and the number of neighbors *k*, can affect the accuracy of the method.

Authors in [7], suggested another method to eliminate unnecessary data, they proposed, as first step, to eliminate the noise data points using the shared nearest neighbor (SNN) algorithm, then identify and eliminate the core data points using the unit vectors concept and only keep the boundary points to use in the training step. However, this method can't effectively detect and eliminate all core points. In addition, they need to define a thresh hold δ test the similarity between data points, and a constant *k* for minimum number of neighbors.

B. Solving dual problem

Although reducing data space can allow a great performance improvement, finding an alternative problem to the original dual problem attracted the main researchers' attention due its high computational complexity.

The first attempt to find an alternative problem to the SVC dual problem was in [1], authors proposed to use the sequential minimal optimization (SMO) algorithm. It decomposes the original dual problem into a series of small-scale convex quadratic programming problems. In each problem, only two samples are required as the working set. Thus, in order to obtain a globally optimal solution, the algorithm starts with two heuristically selected factors β while the others are fixed [3]. Despite the improvement that SMO guarantied, it still requires a high time complexity.

In the aim to make SVC consume less, Y. Ping et al. [8] proposed to use the dual coordinate descent method (DCD), which was firstly proposed by C.-J. Hsieh et al. [9] to solve large scale linear SVM. Firstly, they reformulated the original dual problem to a linear one similar to SVM, however it still an unsupervised model. Thus, they extended it to an iterative algorithm to compute the β_i coefficients. The resulting model shows a flexibility when dealing with storage resources. It offers two ways to compute and store kernel values, search on demand or calculate on demand, depending on the used platform capabilities. If the platform has sufficient storage resources, one can calculate and store the whole kernel matrix in the memory, then use the search on demand policy. Otherwise, we compute only the needed block of the kernel matrix on

demand. This strategy can reduce the storage complexity of the SVC and allow individuals to use their limited platform resources to deal with large-scale data.

T. Pham et al. [10] applied the stochastic gradient descent (SGD) to the LMH version of the SVC. Their SGD-LMSVC algorithm iteratively train an optimal hyperplane. In each iteration t, it uniformly samples a single data point from the training data set to form a new hyperplane based on the updated information from the previous one.

$$w_{t+1} = (1 - \frac{1}{t})w_t + \frac{C}{t}\mathbb{I}_{[w_t^T \phi(x_{n_t}) \leq 1]}\phi(x_{n_t})$$

Where $\mathbb{I}_A$ is an indicator function, it returns 1 if A is true and 0 otherwise. This kind of solution allows its use in a dynamic mode and optimize memory usage. However, the use of the kernels can lead to a considerable model growth, which can slower the computation rate and cause a potential memory overflow.

C. Improving cluster labeling

Cluster label assignment operation is based on pair-wise testing and graph construction using the decision function $f(x)$, which shown a considerable time and storage complexity. Thus, most of researches in the literature, who are dealing with this step, are interested on how reducing data points used in both, pair-wise tests and graph construction.

D. Parameter tuning

The aim of training $f(x)$ step is to estimate the β_i coefficients. However, and as seen in the previous section, $f(x)$ results depends also on other parameters, such as: the kernel width q, the penalty factor C and the sample rate m -which has an influence on the labeling step-. Thus, many researches in the literature are interested on tuning those parameters due to their great influence on the clustering results. Following, we will discuss the influence of every parameter, and some important researchers work done about that parameter.

1) Kernel width q: The selected kernel function defines how the data points will be mapped to the new hyperspace. As well as, the most used one is the Gaussian kernel, the clusters boundary shape depends on the selected q value. In addition, and as we have seen previously, some of the boundary data points known as SVs, are with direct responsibility of defining the function $f(x)$, thus, increasing or decreasing the number of the SVs will obviously affect the cluster's boundary shape. Ben-Hur et al. [1] showed that the kernel width q is on direct relation with the SVs. The authors proposed to begin with small q value, equal to $1/\max_{i,j} \left\|x_i - x_j\right\|^2$, which leads to generate one cluster englobing all of the data points, then incrementally increasing it into obtaining the desired cluster split.

2) Penalty factor C (trade-off parameter): From the fact that working on real data sets, and trying to divide them into separated clusters is usually not allowed even with different q values, Ben-Hur et al [1] confirmed the necessity of allowing the Bounded Support Vectors (BSVs) for cluster separation.

3) Sample rate m: Another important parameter to tune, the sample rate m. It does not affect the cluster shape as the previous two parameters, but it has a great influence on the labeling step. However, not a lot of interest in the literature about this parameter.

IV. Proposed solution

The most common algorithm used to solve the Support Vector Quadratic Programming Problem (QPP) is SMO, which has a time complexity of $O(l^3)$. In our approach, we parallelize SVC using an adapted version of Scholkopf's SMO algorithm [5] with the same complexity. Our solution involves partitioning the data set into n sub-datasets, where n denotes the number of utilized machines, allowing for independent training of each sub-data set. This partitioning significantly reduces the computational complexity of each portion to $O((l/n)^3)$. Subsequently, the outcomes of these executions, which constitute subgroups of support vectors representing approximately 10% of every trained sub-data set, are consolidated and treated as a new data set. This new data set is then retrained on a server machine, resulting in a complexity of $O((l/10)^3)$ for the server. The overall complexity of our parallelized approach is expressed as $O((l/n)^3) + O((l/10)^3)$, which proves to be substantially more efficient than the original $O(l^3)$ complexity.

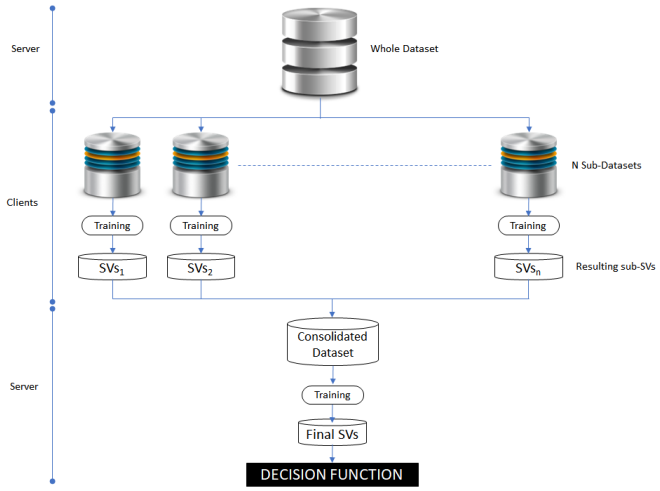

Fig. 2: Fig Proposed solution diagram.

V. Experiments

In this section, we present the results of our experiments conducted to evaluate the performance of our parallelized Support Vector Clustering (SVC) approach. These experiments aim to provide insights into the impact of parallelization on training efficiency. It's important to clarify that the experiments presented in this section were conducted through simulation on a single computer. In this simulation setup, sub-datasets were trained sequentially, and the highest execution time was considered for comparison purposes. Since the kernel

function parameter, q, plays a pivotal role in shaping clusters and notably influences the number of support vectors, our preliminary results involve varying this parameter to enhance the precision of the parallel version, as illustrated in Table I The process of experiment step is as following:

- We used a data set with 3031 instances and 2 attributes,
- Set q to 0.0007 for the mono-post version,
- Penalty factor C to 1 (No outlier allowed),
- We trained the algorithm on the mono-post version and we get the following results:
  - Execution time: 670
  - Number of SVs: 245

TABLE I: Preliminary experimental results.

Cli	q	SV cons	SV mat	Exec T	SV acc	Time rep
9	0.0007	189	141	38,37	58%	06%
9	0.001	214	165	32,37	67%	05%
9	0.002	233	188	137,14	77%	20%
9	0.003	236	192	272,85	78%	41%
9	0.004	239	202	397,58	82%	59%
9	0.005	235	206	275,76	84%	41%
9	0.006	243	213	304,48	87%	45%
9	0.007	240	213	315,04	87%	47%
9	0.008	245	218	360,53	89%	54%
9	0.009	241	217	492,58	89%	74%

Notes:

- Cli: represents the number of client machine,
- q: represents the kernel width,
- SV Cons: represents the number of support vectors in the new consolidated data set,
- SV mat: represents the number of matched SVs to the mono post results,
- Exec T: represent the highest execution time of a client machine,
- SV acc: represent the accuracy in term of matching SVs to the mono post results,
- Time rep: represent the time report to the execution time of the mono post results,

A. Result discussion

As previously mentioned, the kernel width significantly shapes the cluster structure and affects the number of support vectors. Our results underscore the influence of varying this parameter on the accuracy and time complexity of our proposed solution. Notably, we observed that setting q = 0.008 led to a 50% reduction in execution time while maintaining acceptable accuracy.

It's essential to emphasize that these results are preliminary. Other parameters remain available for fine-tuning to further enhance the accuracy of our solution and achieve additional time complexity gains. These parameters include:

- The penalty factor, C, to assess the influence of outliers on our solution.
- The number of clients employed in our approach.

By optimizing these parameters, we anticipate further improvements in both accuracy and efficiency.

VI. Conclusion

In this study, we have presented a parallelized approach to Support Vector Clustering (SVC) and conducted a series of experiments to evaluate its performance. Our approach, based on an adapted version of the Sequential Minimal Optimization (SMO) algorithm, aims to reduce the time cost associated with training while maintaining clustering accuracy. Through simulation on a single computer, we demonstrated that parallelization significantly improves training efficiency, making SVC applicable to larger datasets.

Our experiments also highlighted the importance of parameter tuning, particularly the kernel width (q), in achieving the desired balance between execution time and accuracy. Notably, a parameter setting of q = 0.008 led to a substantial reduction in execution time while still delivering acceptable clustering results.

While these preliminary results are promising, there is room for further optimization and exploration of additional parameters such as the penalty factor (C) and the number of clients used. These factors may provide avenues for improving the accuracy and efficiency of our solution.

In summary, our work contributes to the field of unsupervised machine learning by introducing a parallelized SVC approach and shedding light on the significance of parameter tuning. We believe that our findings pave the way for more efficient and scalable SVC implementations, with potential applications in various domains. Future research can delve deeper into parameter optimization and real-world deployment, building upon the foundations laid in this study.

References

[1] Ben-Hur, Asa, et al. "Support vector clustering." Journal of machine learning research 2.Dec (2001): 125-137.

[2] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[3] H. Li and Y. Ping, "Recent advances in support vector clustering: Theory and applications," Int. J. Pattern Recognit. Artif.

[4] D. M. J. Tax and R. P. W. Duin, "Support vector domain description," Pattern Recognit. Lett., vol. 20, no. 11–13, pp. 1191–1199,

[5] B. Sch, J. C. Platt, J. Shawe-taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional

[6] Y. Li, S. Member, and L. Maguire, "Selecting Critical Patterns Based on Local Geometrical and Statistical Information," vol. 33, no. 6, pp. 1189–1201, 2011.

[7] Wang, Jeen-Shing, and Jen-Chieh Chiang. "An Efficient Data Preprocessing Procedure for Support Vector Clustering." J. UCS 15.4 (2009): 705-721.

[8] Y. Ping, Y. F. Chang, Y. Zhou, Y. J. Tian, Y. X. Yang, and Z. Zhang, "Fast and scalable support vector clustering for large-scale data analysis," Knowl. Inf. Syst., vol. 43, no. 2, pp. 281–310, 2015.

[9] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear svm, in: Proceedings of the 25th International Conference on Machine Learning (ICML '08), ACM, 2008, pp. 408–415.

[10] T. Pham, H. Dang, T. Le, and H. Le, "Stochastic Gradient Descent Support Vector Clustering," pp. 88–93, 2015.