2018 30th International Symposium on Computer Architecture and High Performance Computing

Effect Of Network Topology On The Performance Of ADMM-based SVMs

1st Shirin Tavara
Information Technology
University of Borås
Borås, Sweden
shirintavara@hotmail.com

2nd Alexander Schliep
Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
alexander.schliep@cse.gu.se

***Abstract*—Alternating Direction Method Of Multipliers (ADMM) is one of the promising frameworks for training Support Vector Machines (SVMs) on large-scale data in a distributed manner. In a consensus-based ADMM, nodes may only communicate with one-hop neighbors and this may cause slow convergence. In this paper, we investigate the impact of network topology on the convergence speed of ADMM-based SVMs using expander graphs. In particular, we investigate how much the expansion property of the network influence the convergence and which topology is preferable. Besides, we supply an implementation making these theoretical advances practically available. The results of the experiments show that graphs with large spectral gaps and higher degrees exhibit accelerated convergence.**

***Index Terms*—ADMM, SVMs, Expander Graphs, Distributed Optimization, Convergence**

I. Introduction

Distributed optimization methods are key for solving large-scale machine learning problems due to the exponential growth of digital data. Serial methods are no longer capable of solving today's large problems due to the lack of scalability. Even centralized parallel optimization methods may perform poorly due to communication overheads. Therefore, decentralized distributed optimization methods play an important role in solving problems with big data. Alternating Direction Method Of Multipliers (ADMM) is one of such successful distributed methods since it is robust, distributedly parallelizable, and it has convergence guarantees. However, even ADMM may suffer from slow convergence in specific circumstances [1, 2]. In decentralized ADMM through a network, distributed agents/nodes with the knowledge of local data solve local optimization problems and only communicate with their neighboring nodes with the common goal of reaching consensus. Franca and Bento [2] point out that the network topology has impact on the convergence rate of ADMM in the context of a specific consensus problem. This leads to the natural question, whether their observation also arises in different circumstances. In this paper, we investigate the impact network topology has on ADMM-based Support Vector Machines (SVMs) [3]. In particular, we investigate how much the expansion property and connectivity of the network influence the convergence and which topology is preferable. We also supply an implementation making these theoretical advances practically available. The outline of the paper is as follows. We briefly discuss the basics of network topology in section II. In section III, we explain expander graphs and their properties followed by a brief summary of SVMs and ADMM in section IV and V. We in section VI describe the method and corresponding materials used in this article. The results of the experiments are presented in VII and we analyze the results in section VIII. Finally, the summary and conclusion of the paper are briefly described in section IX.

II. Network Topology

Network topology and the connectivity of the underlying graph have impact on the performance of network-based distributed algorithms in terms of convergence and the number of iterations until convergence. Such impact is shown for solving linear equations [1] and for ADMM with a specific optimization problem not related to SVMs [2]. In this paper, we investigate the impact of the network topology and the connectivity of the underlying graph on the performance of ADMM-based SVMs in terms of convergence. Consider the network of distributed agents/nodes as a graph $G(V, E)$, where $V = \{1, 2, 3, \ldots, n\}$ is the set of the nodes and $E \subseteq V \times V$ is the set of edges between nodes. We assume that G is an undirected and connected graph and that there are no multiple edges between any two nodes. The network topology of the graph is shown by the corresponding adjacency matrix $A(G) = [a_{ij}]_{n \times n}$, where

$$a_{ij} = \begin{cases} 1 & \text{for}(i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

By definition, the adjacency matrix of an undirected graph is symmetric and has real eigenvalues. The spectral gap/connectivity of a graph is related to the expansion property of the graph. In order to define the spectral gap and connectivity of a network, we first define the Lagrangian matrix corresponding the adjacency matrix $A(G)$. The Laplacian matrix of an adjacency matrix, denoted $L(A)$, and has -1 for a connected pair nodes and the degree of each node on its diagonal, i.e.,

$$l_{ij} = \begin{cases} -1 & \text{for}(i,j) \in E \quad \& \quad i \neq j \\ k_i & \text{for} i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

978-1-5386-7769-8/18/$31.00 ©2018 IEEE
DOI 10.1109/SBAC-PAD.2018.00067

Here, k_i is the degree of node i. The eigenvalues of $A(G)$ satisfy

$$k_{max} = \mu_1^{'} \geqslant \mu_2^{'} \geqslant ... \geqslant \mu_n^{'}, \tag{3}$$

and the eigenvalues of $L(G)$ satisfy

$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n \leq 2k_{max}. \tag{4}$$

Here, k_{max} is the largest degree of all nodes.
The algebraic connectivity or the spectral gap of a graph network is related to the non-trivial eigenvalues of $A(G)$ and $L(G)$, i.e., it relates to $\mu_2 = max\{\mu_2^{'}, |\mu_n^{'}|\}$ and λ_2 [4]. The first smallest eigenvalue of $L(G)$ is trivial ($\lambda_1 = 0$) and corresponds to the first largest and trivial eigenvalue of $A(G)$, i.e., $\mu_1 = k_{max}$ and in some cases $|\mu_n| = |-k_{max}|$ [5].

A small spectral gap relates to a small number of edges required to be taken away to generate a bipartite graph. In contrast, a large spectral gap relates to non-modularity of the corresponding graph. For details refer to [4].

III. Expander Graphs

A group of well-studied connected graphs are expander graphs, in which any subset expands through all nodes in a robust manner, i.e., any subset of the graph efficiently connects to many nodes. Properties such as effective communication, high- and well-connectivity, and sparseness make expander graphs good choices for designing efficient networks. The expansion property of an expander graph can be defined by the Cheeger or isoperimetric constant [4]. The Cheeger constant shows that whether the graph has bottlenecks, i.e., whether there are two large subsets of vertices connected by only few edges. A large Cheeger constant indicates many edges between the two large subsets of vertices. In contrast, a small constant shows there is a bottleneck between the two subsets of vertices and they are connected with only few edges. The Cheeger constant of graph $G(V, E)$ is denoted as $h(G)$ and it can be defined as follows,

$$h(G) = min_{S \subseteq V, |S| \leqslant \frac{|V|}{2}} \frac{|\partial S|}{|S|}. \tag{5}$$

Here, $\partial S = \{(e, e^{'}) \in E : e \in S, e^{'} \in V \setminus S\}$. The Cheeger constant is related to the spectral gap by Cheeger inequalities, i.e.,

$$\frac{\lambda_2}{2} \leq h(G) \leq \sqrt{2d\lambda_2}. \tag{6}$$

The expansion properties can be enhanced towards increasing the spectral gap. In this regard, d-regular random graphs in which each node is connected to d other nodes are expanders if and only if the corresponding spectral gap is lower bounded [4]. In this paper, we study the impact of d-regular graphs on the convergence performance of the ADMM-based SVMs.

IV. SVMs

SVMs are a set of supervised machine learning techniques developed from statistical learning theory to solve classification and regression problems. The basic idea of SVMs in a simple binary classification problem is to search for a hyperplane that is the farthest to the closest training data points from both sides of the hyperplane. This process has two phases, training and testing. In the training phase, the machine is trained to find a plane that separates the given data samples into two classes. After the machine is trained, the training model is extracted and then the testing phase is carried out. In the testing phase, the SVMs model predicts which class label a new unseen test sample should have [6]. SVMs give a good generalization performance [7] and minimize the upper bound of the generalization error [8]. SVMs have special characteristics that can be used to implement efficient parallel algorithms in terms of time and memory. One characteristic is the sparsity of solutions [9], i.e., the solution is obtained by only a few samples called support vectors that determine the maximum margin separating hyperplane [10]. Another characteristic of SVMs is to perform the nonlinear mapping without knowing the mapping function using predefined functions called kernels for calculating the inner product of mapping functions [10]. Other characteristics of SVMs are the simple structure of SVMs constraints and the definition of the kernel function in a linear case, i.e., the inner product is a simple dot product in a linear case [11]. The optimization problem addressed by SVMs can be written as follows,

$$\begin{aligned} &Primal: \\ &\quad min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ &s.t. \quad \forall i: \quad y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad , i = 1, 2, ..., N \\ &\qquad \forall i: \quad \xi_i \geq 0 \quad , i = 1, 2, ..., N. \end{aligned} \tag{7}$$

Here $\mathbf{w}$ is the weight vector for the hyperplane, $\mathbf{x}$ is a vector of observations, y_i are the class labels with $y_i \in \{+1, -1\}$, b is the bias parameter and $\Phi(\mathbf{x})$ is the map function. In the case that data can be classified by a linear classifier, $\Phi(\mathbf{x}) = \mathbf{x}$, but real-life data cannot always be classified by a linear classifier [8]. In non-linear cases, one can map the data from the input space into a high dimensional feature space using a non-linear transformation, i.e., $\Phi(\mathbf{x})$ maps the input vector $\mathbf{x}$ to the feature space [8]. In the feature space, the data can be linearly separable. Consequently, the dual form of equation (7) is represented in equation (8),

$$\begin{aligned} &Dual: \\ &min \quad D(\alpha) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ &s.t. \quad \sum_{i=1}^{N} y_i\alpha_i = 0 \\ &\forall i: \quad 0 \leq \alpha_i \leq C \quad , i = 1, 2, ..., N. \end{aligned} \tag{8}$$

Here $\boldsymbol{\alpha}$ is Lagrangian multipliers and $\alpha_i \in \boldsymbol{\alpha}$, $\mathbf{1}^T$ is a vector of ones and Q is a matrix of size $N \times N$ and $Q_{ij} = y_i y_j \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$.

V. ADMM

Distributed methods are one of the important approaches for solving large-scale machine learning problems. One of the popular distributed methods is ADMM since it has properties such as robustness, scalability of solving problems with big data, easily distributable and parallelizable. The robustness of ADMM refers often to no requirement of differentiability of the objective function. ADMM guarantees the convergence of convex functions [2] which is the case in SVMs. The convergence rate of ADMM is $O(1/N)$ for convex functions, where N is the number of iterations [12, 13]. Note in practice the convergence rate of ADMM is still not well-understood [2].

The optimization problems of the following form can be solved using ADMM.

$$\begin{aligned} min \quad & f(w) \\ s.t. \quad & w - v = 0 \end{aligned} \tag{9}$$

In order to solve the problem (9) in a distributed manner, one can reformalize the optimization problem in the form of (10). This problem is solvable using ADMM because of the decomposability of ADMM in which each node in a network has own independent objective function and constraints. ADMM can solve consensus optimization in which nodes only communicate with one-hop neighboring nodes. To do this, consensus constraints are defined to force local variables to agree across neighbors. The distributed consensus-based optimization can be formalized as follows,

$$\begin{aligned} min \quad & \sum_{i=1} f_i(w_i) \\ s.t. \quad & w_i - w = 0, \quad i = 1, 2, ..., n. \end{aligned} \tag{10}$$

Here, w is the consensus variable across the neighboring nodes. For detailed information, refer to [14, 15]. In this paper, we have followed the approach given by Forero, Cano and Giannakis in [14] for solving consensus ADMM-based SVMs.

VI. Material and Method

We designed experiments with datasets gathered by LIBSVM Data [16] from several machine learning data repositories such as UCI [17]. Table I shows the datasets with the corresponding number of training and testing instances and features we used in the experiments. In order to study the impact of network topologies on the number of iterations until convergence, we implemented several random d-regular expander graphs. For each graph, upper and lower bounds of the spectral gap are calculated using the formula given by Joel Friedman [18]. The formula is given for the second largest eigenvalue of the adjacency matrix $\mu_2 = max\{\mu_2^{'}, |\mu_n^{'}|\}$. We adapted the formula for the second smallest eigenvalue of the Lagrangian matrix λ_2 as follows. For $\epsilon > 0$,

$$d - 2\sqrt{d-1} - \epsilon \leq \lambda_2 \leq d + 2\sqrt{d-1} + \epsilon. \tag{11}$$

Note $\lambda_i = d - \mu_i^{'}$, $i = 1, 2, \ldots, N$. This holds for every random d-regular graph of size N for sufficiently large Ns. We implemented a special type of regular graphs called Ramanujan graphs. A d-regular graph is Ramanujan if and only if $\mu_2 \leq 2\sqrt{d-1}$ holds, where μ_2 is the second largest eigenvalue of the adjacency matrix of the graph as defined above.

For simplicity, we focus on the second smallest eigenvalue λ_2 of the Laplacian matrix as the spectral gap instead of using the second largest eigenvalue of the adjacency matrix μ_2 since the spectral gap is defined as $\lambda_2 = d - \mu_2^{'}$, where $\mu_2 = max\{\mu_2^{'}, |\mu_n^{'}|\}$.

For a Ramanujan graph, formula (11) appears as follows [19],

$$d - 2\sqrt{d-1} \leq \lambda_2 \leq d + 2\sqrt{d-1}. \tag{12}$$

To run experiments, we used several random regular graphs with different degrees and number of graph nodes depending on the size of training datasets. The first group of graphs, denoted $G1$, consists of d-regular expander graphs with low degrees designed for training small datasets, where $d = \{3, 5, 7, 9, 11, 13, 15\}$ and the second group of graphs, denoted $G2$, consists of d-regular expander graphs with higher degrees for training large datasets, where $d = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

In this paper, we have focused on binary classifications since a multi-class classification can be transformed into several binary classifications using the one-versus-all technique. The classification accuracy of a consensus ADMM-based SVM is influenced by parameters such as ρ in ADMM and JC and γ in SVMs, where J is the number of graph nodes and γ is the RBF kernel parameter. To estimate sufficiently good values of the parameters, we used a grid search with cross-validation. Besides, we use normalizing and standardizing scaling techniques to further improve the classification accuracy if needed. To evaluate the results, we used standard statistics such as true positive/negative rate, positive/negative predictive rate, and accuracy metrics.

To measure the impact of expander graph topology, the algorithm trains the datasets for each d-regular expander graph using shared memory parallel programming. Thereafter, the number of iterations and the corresponding elapsed time are measured until convergence. To stabilize our analysis for some of the datasets, we shuffled the training data 10 times. Each shuffled data are trained and the number of iterations and the elapsed time are measured and then we calculated the average value for the final analysis.

TABLE I
Dataset information

Datasets	Training Points	Testing Points	Features
ionosphere	300	51	34
svmguid	3089	4000	4
phishing	11055	1655	68
a9a	32561	16281	123
ijcnn	35000	14990	22
skinNonskin	37492	5000	3

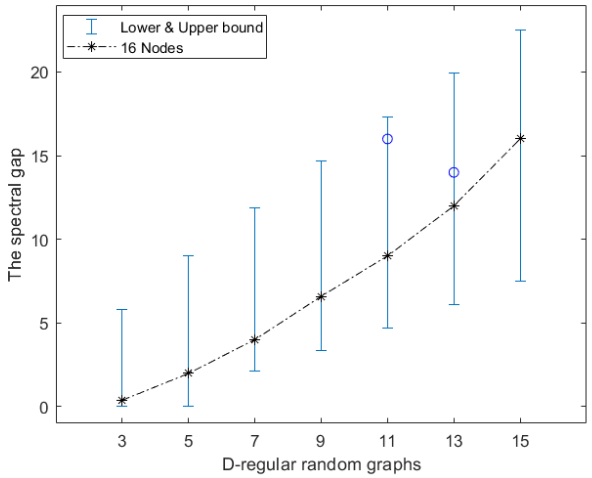

Fig. 1. The upper and lower bounds of the spectral gap for random d-regular expander graphs with 16 graph nodes, where $d = \{3, 5, 7, 9, 11, 13, 15\}$. "*" symbols show the spectral gaps of the d-regular graphs implemented in the experiments. "o" symbols show that a random 11-regular graph can have a higher spectral gap than a random 13-regular graph.

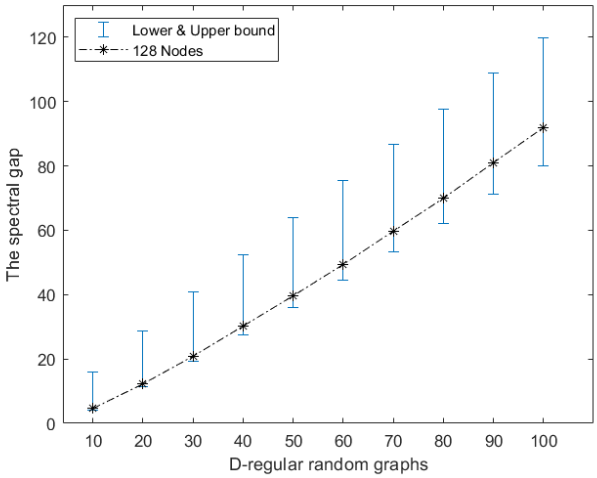

Fig. 2. The upper and lower bounds of the spectral gap for random d-regular/Ramanujan graphs with 128 graph nodes and $d = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

VII. Results

The lower and upper bounds of the spectral gap (the second smallest eigenvalue of the Lagrangian matrix) λ_2 for low degree regular graphs $G1$ is shown in Fig. 1. The same concept for high degree regular graphs $G2$ is shown in Fig. 2. We use the low degree expander graphs for training small datasets and the high degree expander graphs for training larger datasets. The vertical bars show the bounds calculated using formula (12).

In Fig. 1, " $*$ " indicates the spectral gap of the Ramanujan graphs implemented for 16 graph nodes in the experiments. Similarly, in Fig. 2, " $*$ " shows the spectral gaps of the Ramanujan graphs implemented for 128 graph nodes. As the figures show, the spectral gaps of all of the expander graphs we used in the experiments for 16 and 128 graph nodes are inside the allowed bounds. As both figures show, the connectivity of the regular/Ramanujan graphs increases as the degree of the graphs becomes larger.

Fig. 3 shows the number of iterations and the elapsed time for training `ionosphere` and `svmguid` datasets until convergence using group $G1$ of d-regular graphs with 16 graph nodes. As shown in the figure, the number of iterations decreases as the degree of the graph increases. The trend is most visible between 3 to 9 regular graphs for `svmguid` and between 7 to 11 for `ionosphere`. In contrast, the decrease in the number of iterations saturates as the degree of the graphs is close to the number of graph nodes, i.e., 16 nodes in this case. This is visible between 11 to 15 regular graphs for both datasets.

Fig. 4 shows the number of iterations and the elapsed time for training `phishing`, `a9a`, `skinNonskin`, and `ijcnn` datasets using group $G2$ of d-regular graphs with 128 graph

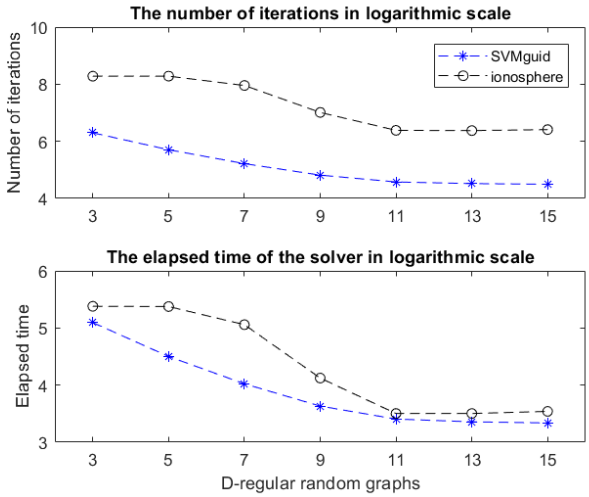

Fig. 3. Impact of d-regular graphs on the number of iterations and time for `svmguid` and `ionosphere` datasets using 16 graph nodes.

nodes. The results highlight the performance improvement in terms of the number of iterations as the spectral gap and the connectivity of the regular graphs increases with the fixed number of nodes (128 nodes). This trend is more stable for `phishing` and `a9a` datasets throughout all degrees and it is most noticeable between 30 to 70 regular graphs for `skinNonskin` and `ijcnn` datasets. Note for the degree (d) close to the number of graph nodes, the performance improvement saturates or becomes negligible. This is most visible for degree between 90 to 100 for all datasets.

As shown in Fig. 4, `ijcnn` did not converge for 10 and 20 regular graphs in less than 10000 iterations which we put as the maximum iteration for the convergence. This might

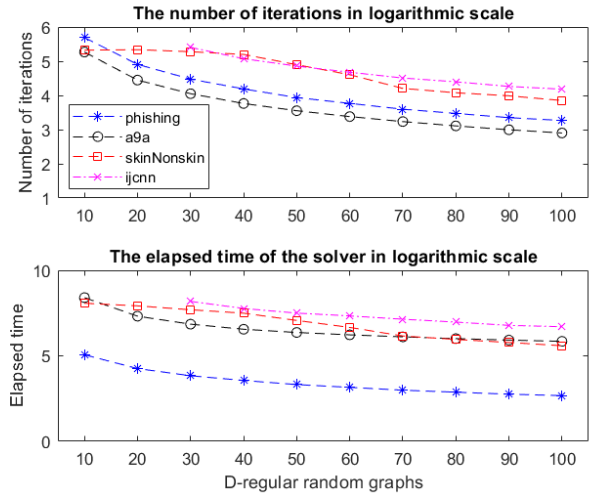

Fig. 4. Impact of d-regular graphs on the number of iterations and time for `phishing`, `a9a`, `skinNonskin`, and `ijcnn` datasets using 128 graph nodes.

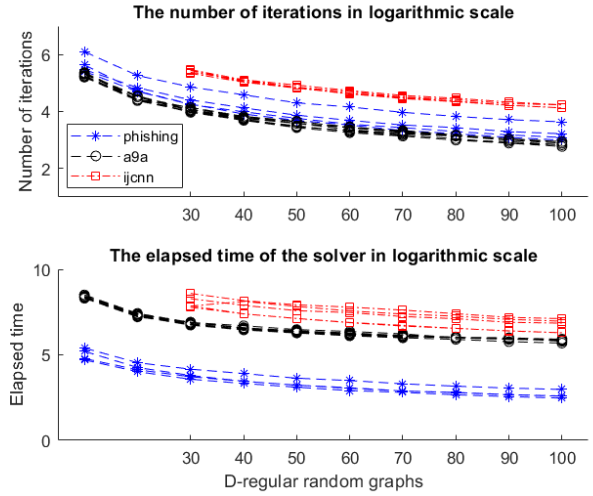

Fig. 5. Several rounds of shuffling data. Shuffling increases the classification accuracy while keeping the trend of decreasing the number of iterations and time for d-regular graphs.

be explained due to inefficient communication between graph nodes when the degree of the graph is low.

Fig. 5 shows the number of iterations for several rounds of shuffling data. Shuffling data follows the trend of decreasing the number of iterations and time, besides improving the classification accuracy and leading to a stable classifier.

VIII. Disussion

The performance of a network-based ADMM implementation of SVMs was improved based upon the efficient connection between neighbouring nodes using expander/Ramanujan graphs. The graphs with high degrees and spectral gaps and consequently many neighbouring nodes exhibit accelerated convergence (Fig. 3 and 4). This finding is consistent with that of Cao et al. [1] whose theoretical analysis suggested networks with higher mean degree. Although their approach was not tested for network-based ADMM algorithms, nor were tested for nonlinear systems, the authors reported the performance improvement of a distributed algorithm in terms of convergence.

The bounds of the spectral gap expanded as the degree of the graph and consequently connectivity increased for the fixed number of nodes, thus efficient communication between neighboring nodes. Note that a random d-regular graph may have better connectivity than a random $d^{'}$-regular graph even if $d < d^{'}$. This is likely if the spectral gap of d-regular graph is larger than $d^{'}$-regular graph, e.g., as shown in Fig. 1 by "o" symbols, a random 11-regular graph has a higher spectral gap than a random 13-regular graph. Therefore, increasing the degree of a graph will not necessarily lead to better connectivity.

The results of our experiments showed accelerated convergence for increasing the degree of expander graphs, however the decrease of the number of iterations was saturated or became very small as the degree of the graph got closer to the number of graph nodes (Fig. 3 and 4). This may be explained due to the increasing number of neighbors, although the convergence is guaranteed, many graph nodes should communicate and exchange their local results to reach the consensus. This leads to slow convergence.

Based on the line of reasoning above, it is interesting to find an appropriate degree of regular graphs with regards to the number of graph nodes in which the degree should be sufficiently high that leads to efficient communication and sufficiently low that leads to good connectivity.

The impact of many communicating neighbouring nodes is minor in the shared memory parallelism. In contrast, in the distributed memory parallelism, the communication between many nodes reaching the consensus might cause communication overhead. Thereby, increasing the degree of graphs close to complete graphs may not be efficient as in the shared memory parallelism. We will further investigate the impact of expander graphs on the performance of our algorithm using distributed parallelism in the future work.

Beside the discussion of graph topology, several factors showed their importance for the classification accuracy. One factor was the balance between the two classes. To keep the balance between two classes, we randomly shuffled data and it exhibited higher classification accuracy versus unshuffled data while keeping the trend of decreasing the number of iterations and time. This might not possible for networks that supply their data from distributed sources and do not allow to combine and shuffle the data due to privacy issues. Another factor was data cleansing and improving data quality. Although we did not investigate different techniques of data cleansing, simply removing missing and corrupted data exhibited higher accuracy. Our implemented algorithm is able to solve the dense and sparse formats of data and we plan to further improve the performance in regards to special treatment of sparsity.

IX. Summary and conclusion

A consensus-based ADMM implementation of SVMs involves a consensus agreement among neighboring nodes, consequently the number of neighbors and the way they are connected impact the performance of the algorithm in terms of convergence and communication. Thereby, the network topology used for communication of neighboring nodes plays an important role in performance improvement of the distributed algorithm.

Based on the line of reasoning above, complete graphs are well-known for their connectivity, however, high connectivity does not come cheap in particular using distributed memory parallelism, i.e., all nodes need to communicate with each other and this may increase the number of iterations until convergence and increase the communication complexity. Random d-regular expander graphs are good sparse approximations of complete graphs [20] in which good connectivity property is inherited with efficient communication between nodes. Note the better the expander property of regular graphs is, the faster nodes communicate.

Acknowledgment

This work is supported by universities of Borås and Gothenburg.

References

[1] H. T. Cao, T. E. Gibson, S. Mou, and Y. Y. Liu, "Impacts of network topology on the performance of a distributed algorithm solving linear equations," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 1733–1738.

[2] G. França and J. Bento, "How is Distributed ADMM Affected by Network Topology?" *ArXiv e-prints*, Oct. 2017.

[3] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Research, Tech. Rep. MSR-TR-98-14, April 1998. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=69644

[4] L. Donetti, F. Neri, and M. A. Muoz, "Optimal network topologies: expanders, cages, ramanujan graphs, entangled networks and all that," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2006, no. 08, p. P08007, 2006. [Online]. Available: http://stacks.iop.org/1742-5468/2006/i=08/a=P08007

[5] Y. T. Chow, W. Shi, T. Wu, and W. Yin, "Expander graph and communication-efficient decentralized optimization," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, Nov 2016, pp. 1715–1720.

[6] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[7] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[8] H. Byun and S.-W. Lee, "Applications of support vector machines for pattern recognition: A survey," in *Pattern recognition with support vector machines*. Springer, 2002, pp. 213–236.

[9] A. Carpenter, "cusvm: A cuda implementation of support vector classification and regression," *patternsonscreen. net/cuSVMDesc. pdf*, 2009.

[10] O. Ivanciuc, "Applications of support vector machines in chemistry," *Reviews in computational chemistry*, vol. 23, p. 291, 2007.

[11] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," *Parallel computing*, vol. 29, no. 4, pp. 535–551, 2003.

[12] B. He and X. Yuan, "On the $o(1/n)$ convergence rate of the douglasrachford alternating direction method," *SIAM Journal on Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012. [Online]. Available: https://doi.org/10.1137/110836936

[13] R. Monteiro and B. Svaiter, "Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 475–507, 2013. [Online]. Available: https://doi.org/10.1137/110849468

[14] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *The Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[17] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[18] J. Friedman, "A proof of alon's second eigenvalue conjecture and related problems," *CoRR*, vol. cs.DM/0405020, 2004. [Online]. Available: http://arxiv.org/abs/cs.DM/0405020

[19] M. Morgenstern, "Existence and explicit constructions of q + 1 regular ramanujan graphs for every prime power q," *Journal of Combinatorial Theory, Series B*, vol. 62, no. 1, pp. 44 – 62, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0095895684710549

[20] C. Williamson, "Spectral graph theory, expanders, and ramanujan graphs," *University of Washington*, 2014.