ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

SMO Lattices for the Parallel Training of Support Vector Machines

Markus Kächele, Günther Palm and Friedhelm Schwenker *

Ulm University - Institute of Neural Information Processing
89069, Ulm - Germany

Abstract. In this work, a method is proposed to train Support Vector Machines in parallel. The difference to other parallel implementations is that the problem is decomposed into hierarchically connected nodes and that each node does not have to fully optimize its local problem. Instead Lagrange multipliers are filtered and transferred between nodes during runtime, with important ones ascending and unimportant ones descending inside the architecture. Experimental validation demonstrates the advantages in terms of speed in comparison to other approaches.

1 Introduction

One of the most important tasks in data mining and machine learning is the classification of data into a set of given classes. Classification algorithms have far reaching application and devices that implement them have already started to be part of our life – be it as face detectors in cameras or smartphones, speech analysis, or by automatically supporting medical diagnoses such as the severity of depression [1]. In recent years, many classification algorithms have been proposed but still one of the most popular and most widely used classifiers is the Support Vector Machine (SVM) [2]. Very appealing theoretical characteristics such as the maximum margin property render it a practitioners first choice. However, there are also drawbacks such as the relatively high training complexity that render it especially for large scale problems impractical. Many researchers proposed solutions to these problems in the form of iterative algorithms [3, 4] or by breaking up the problem into smaller subproblems that are easier to solve [5]. In this work, a method is presented to train an SVM in parallel by decomposing the training set into smaller subsets. By taking individual optimization steps for the subproblems using the Sequential Minimal Optimization (SMO) algorithm [4], potentially important Lagrange multipliers are filtered and passed along a hierarchy to take their part in the solution of the whole problem.

This work is organized as follows: In the next Section, a short overview of SVMs and the SMO algorithm is given. The proposed approach is presented in Section 3 followed by experimental validation in Section 4. The work is concluded by a discussion and a summary of the presented ideas.

*This paper is based on work done within the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG). Markus Kächele is supported by a scholarship of the Landesgraduiertenförderung Baden-Württemberg at Ulm University.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

2 Background

Training of an SVM involves solving the convex optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad L(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{1}$$

with Lagrange multipliers α_i under the constraints $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$. $x_{\{i,j\}}$ and $y_{\{i,j\}}$ denote data points and labels, respectively. Standard quadratic programming (QP) methods can be utilized for solving Equation 1, however their complexity is too high for large datasets. Therefore, other algorithms have been developed that solve this task iteratively, such as Platt's SMO [4]. Based on the decomposition scheme by [3], the algorithm selects two Lagrange multipliers, solves the QP subproblem analytically and repeats the process until convergence is achieved. Convergence is achieved if a solution is found that fulfills the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i(\langle x_i, w \rangle + b) \geq 1 \\ 0 < \alpha_i < C &\Leftrightarrow y_i(\langle x_i, w \rangle + b) = 1 \\ \alpha_i = C &\Leftrightarrow y_i(\langle x_i, w \rangle + b) \leq 1 \end{aligned}$$

Here w and b denote the parameters of the hyperplane. By iteratively picking pairs that violate these conditions and optimizing them, SMO constructs a solution in which no violations occur and the global optimum is reached.

3 Lattices of SMO Nodes

The proposed approach builds on the decomposition of the data into independent subsets allowing their distribution to independent working nodes, which can naturally run in parallel. Each node contains an instance of the SMO algorithm and solves the problem for its subset. The workflow is composed of the two alternating phases *SMO computation* and *inter-node communication*. Periodically, inter-node communication steps are introduced to exploit the gathered knowledge of the computation phase by transferring filtered multipliers to connected nodes. Here, the architecture of the nodes is organized in a hierarchy with a varying degree of *importance* of a node depending on its level in the hierarchy. The local importance of the multipliers is estimated and important ones are passed to the upper nodes (i.e. parent), while unimportant ones are either kept or passed to lower nodes (children). The communication of the nodes is restricted to their parent and child nodes (apart from specific cases, see Section 4). The idea is that during the process, the support vectors will ascend in the hierarchy and finally reach the top node. The top node periodically checks if the KKT conditions *for the whole problem* are fulfilled, indicating that an optimal solution for the problem has been found. The idea to hierarchically filter for support vector candidates has already been employed in other works

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

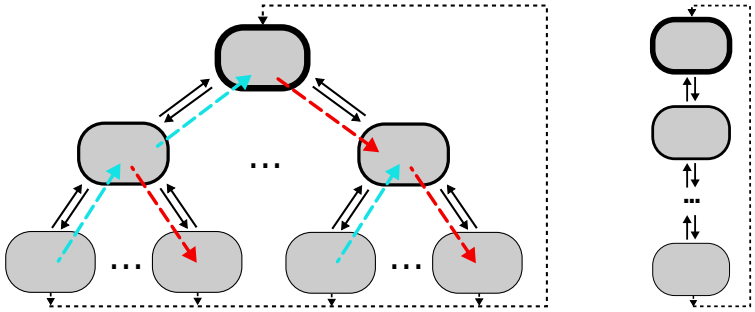

Fig. 1: Two possible architectures. Architectures can have arbitrary structure as long as there is a top node. The dashed line indicates the connection between the bottom and the top of the architecture (see "wrap around").

such as [5, 6]. While somewhat similar to this work, the unique characteristic of the presented approach is that it is not necessary to iteratively solve whole (sub-)optimization procedures, instead only a single node, the top node, will optimize until convergence.

3.1 Architecture setup

In Figure 1 two example architectures are illustrated. The bold outline indicates the importance of the node (depending on the level). The architectures are organized such that a parent node integrates the local information obtained from the subsets contained in its child nodes. The black arrows denote the communication paths of the nodes. The upwards path is reserved for potentially important multipliers (i.e. those that are relevant for the current subproblem), while the downward paths are for the ones, that do not seem to be important for the optimization. Depending on the local filter results, a node will pass multipliers to the parent and child nodes and in turn will receive multipliers from its parent and child nodes. Architectures do not have to comply to the two presented designs in Figure 1 as long as there is a node that optimizes for the KKT conditions of the whole problem (the so called top node).

For reasons grounded in theoretical considerations and convergence issues, the two following modifications are introduced:

- **Wrap-around**: To maintain a closed circle and to prevent that multipliers will sediment to the bottom of the architecture with no way up again the connection from the leaf nodes to the top node is made (as indicated by the dashed line in Figure 1).
- **Sweeping**: The movement direction (naturally either up or down) can only be changed in the top or leaf nodes. That means that once either direction is chosen, a multiplier will continue to take this direction until it either reaches the top node (directly or by wrap-around) or it reaches a leaf node, where the direction can be inverted to make its way up again.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

3.2 Theoretical considerations

In this Section, some insights into the theoretical setup of the approach will be presented. The validity of the presented approach is based on the following key assumptions:

1. Fulfilling the KKT conditions is a necessary and *sufficient* condition [4].
2. By adding and optimizing for a KKT-violating pair α_i, α_j an optimal solution of a subset $\boldsymbol{\alpha}^{\boldsymbol{S}}$ can be extended such that the value of the objective function will be increased: $L(\boldsymbol{\alpha}^{\boldsymbol{S}\cup\{\boldsymbol{i},\boldsymbol{j}\}}) > L(\boldsymbol{\alpha}^{\boldsymbol{S}})$ [3, 7].

Since the local behaviour does not reflect the global importance of a sample, measures have to be introduced to allow the non-greedy selection of samples and therefore enable convergence to the global optimum.

By combining assumption 1 and the fact that the top node checks for the fulfillment of the conditions for the whole problem and not only for its subproblem it follows that the algorithm will converge if all of the necessary Support Vectors will be accessible in the top node.

To get there, from each node, the locally important multipliers are handed to the parent nodes. However, there is no guarantee, that the greedy selection will suffice. Cases can occur in which the top node cannot progress any further because a necessary violating multiplier is hidden in the depths of the architecture, trapped in a constellation in which it is not handed over to higher levels because of its seeming unimportance. To overcome such issues, the aforementioned sweeping procedure is used. After a number of iterations without improvement in the top node, sweeping is utilized to change the otherwise greedy strategy of the nodes to one that *forces* new samples to the top node. For each multiplier, sweeping is turned off after having been examined once by the SMO in the top node. Note that a formal proof of convergence will still have to be presented, however the experiments suggest that globally optimal solutions are indeed achieved using these procedures.

4 Experimental Validation

For the validation of the proposed approach, a comparison with related algorithms such as CascadeSVM (CSVM) [5] and Distributed Parallel SVM (DPSVM) [6] is carried out on a number of datasets. The experiments are set up such that the dataset is fixed and variations are introduced by different architecture settings and initializations. As DPSVM is a more generalized version of CascadeSVM (see [6] for details) and both operate similarly, the comparisons will be conducted using this algorithm with different architecture settings.

For the experiments the two datasets *breast cancer* (from the UCI repository[1]) and *ring* (artificially generated concentric rings) are used. The architectures as depicted in Figure 1 are trained and compared. Each of the approaches

[1] `http://archive.ics.uci.edu/ml/`

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

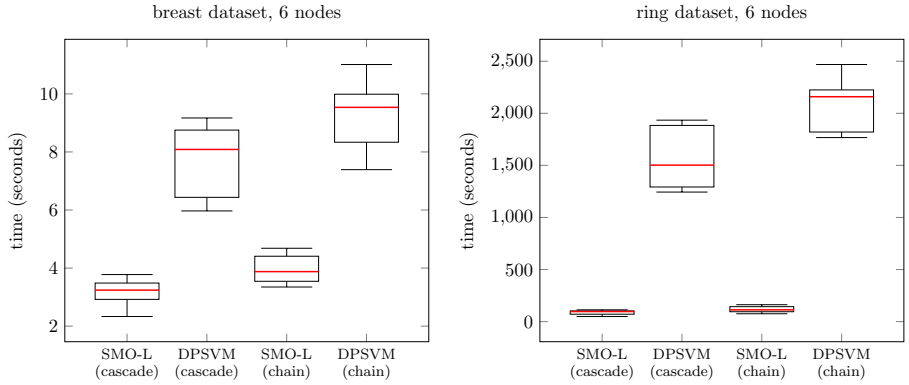

Fig. 2: Training time comparison. In both cases SMO-L is faster than DPSVM. It can be observed that the time gap between the algorithms increases drastically with larger training sets (breast cancer has 580 and ring 2000 samples).

uses a setup with 6 nodes with the data being randomly split among them. All the architectures (also DPSVM) are closed loops (i.e. with wrap-around). Both architectures were trained until convergence up to a tolerance of $\epsilon = 0.0001$. As DPSVM has no such thing as a top node, in each iteration, the convergence of every node is tested and if any of the nodes converged, the training is stopped (as opposed to the original paper in which convergence is stated as *all nodes have converged*). For SMO-L, two additional modifications are introduced: A variable iteration length, that increases over time to favor many quick multiplier exchanges in the beginning and optimization for the exact values in the later run of the algorithm. The second modification is the so called *early out* heuristic, which allows the top node to instantaneously gather every violating sample from anywhere inside the architecture. It is used if convergence has almost been achieved (i.e. when only a few percent of the KKT conditions are violated). Each training run has been repeated 10 times with random initialization to ensure validity. The time utilized for training is measured and plotted for each training set and architecture. The results are illustrated in Figure 2. A comparison between the number of iterations it took to converge can be seen in Figure 3. As can be seen, the SMO-L approach clearly outperforms DPSVM for both architecture sets and both datasets while converging to the same solution.

5 Conclusion

In this work, a method has been presented to train an SVM based on iteratively filtering Lagrange multipliers based on their relative importance in the optimization function. The key aspects of the algorithm are that multiple nodes can be combined to build a filter hierarchy and that all of them can be run in parallel. Inter-layer communication and propagation of multipliers assure that

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

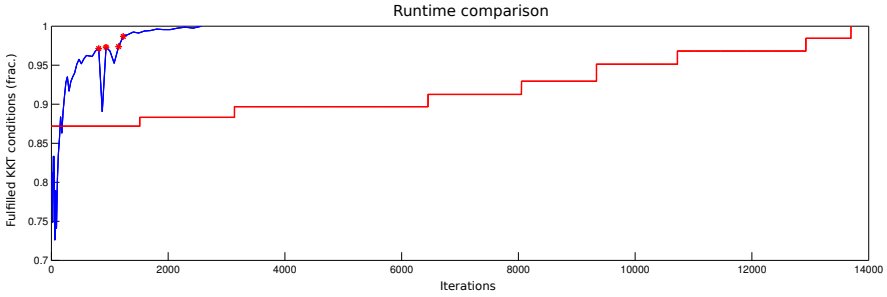

Fig. 3: Comparison of number of iterations. As SMO-L does not require to completely converge each SMO sub-algorithm, much fewer iterations are needed (for SMO-L the optimal solution is reached after 2570 iterations). The red lines indicate the iterations of each step in the DPSVM algorithm. The compared architectures were chains with 6 nodes each. The red stars indicate points where the *early out* heuristic has been used.

relevant information is passed from nodes to nodes in different layers where the information is used to compute an optimization step before weight adaptations take place and the updated information is provided for the other layers. It has been shown that the performance of the proposed method is comparable with established approaches.

There is a multitude of possibilities for future work. The most interesting point would be an in-depth theoretical analysis of the convergence properties. Other directions could include investigations to analyze the initialization phase (e.g. use of overlapping subsets or cluster-based initialization). Also a focus could be set on additional heuristics for the acceleration or stabilization of the algorithm.

References

[1] Markus Kächele, Martin Schels, and Friedhelm Schwenker. Inferring Depression and Affect from Application Dependent Meta Knowledge. In *Proceedings of the 4th International Workshop on Audio/Visual Emotion Recognition*, AVEC '14, pages 41–48. ACM, 2014.

[2] Vladimir N. Vapnik. *Statistical Learning Theory*, volume 2. Wiley, 1998.

[3] Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for Support Vector Machines. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, Sep 1997.

[4] John C. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[5] Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel Support Vector Machines: The Cascade SVM. In *NIPS*, 2004.

[6] Yumao Lu, Vwani Roychowdhury, and Lieven Vandenberghe. Distributed Parallel Support Vector Machines in Strongly Connected Networks. *IEEE Transactions on Neural Networks*, 19(7):1167–1178, July 2008.

[7] S. Sathiya Keerthi and Elmer G. Gilbert. Convergence of a Generalized SMO Algorithm for SVM Classifier Design. *Machine Learning*, 46(1-3):351–360, Mar 2002.