A Discrete-Time Parallel Update Algorithm for Distributed Learning

Tansu Alpcan and Christian Bauckhage
Deutsche Telekom Laboratories, 10587 Berlin, Germany
{*tansu.alpcan, christian.bauckhage*}*@telekom.de*

Abstract

We present a distributed machine learning framework based on support vector machines that allows classification problems to be solved iteratively through parallel update algorithms with minimal communication overhead. Decomposing the main problem into multiple relaxed subproblems allows them to be simultaneously solved by individual computing units operating in parallel and having access to only a subset of the data. A sufficient condition is derived under which a synchronous, discrete-time gradient update algorithm converges to the approximate solution. We apply the proposed distributed learning framework in the context of automatic image tagging as a first processing layer. Initial results from corresponding experiments indicate that he proposed framework has favorable properties including efficiency, configurability, robustness, suitability for online learning, and low communication overhead.

1 Introduction

Recent advances in networking, multiprocessor systems, and multicore processors have made distributed computing widely accessible. Various distributed computing schemes are today applicable to a variety of problems that have been traditionally addressed by centralized approaches. Decentralized schemes are particularly suitable to address machine learning (ML) problems in networked systems such as wireless sensor networks as well as in parallel computing systems consisting of multicore processors. For example, in a variety of ML problems data collection and storage processes are already distributed.

We study in this paper a novel distributed machine learning (DML) framework based on support vector machines (SVMs) for addressing classification problems. Unlike sequential or centralized approaches in the literature [2, 4–6, 9], we focus exclusively on parallel update schemes for DML allowing individual processing units to do simultaneous computations. Most of these contributions investigate training SVMs either sequentially or in parallel first and then fusing them into a centralized classifier. In [5], a decentralized learning scheme is described where a large training set is partitioned into smaller parts. However, as they do not consider a client server architecture, the communication overhead between the individual units is significant. Our approach, on the other hand, is especially useful when each unit has access to a different subset of the overall dataset which may be time-varying, has its own computing resources, and is under communication constraints. It can be applied to networked or multiprocessor systems as well as multicore processors, where each processing core is independent and has its own cache memory (e.g. Cell processor, GPUs).

We apply the proposed distributed learning framework on the first processing layer of a decentralized image classification and tagging system for online applications. Initial experiments with images of unconstrained natural scenes demonstrate that the SVMs trained according to our procedure yield robust results which will be forwarded to higher level processing units.

2 Model

Assume a given set of labeled data $S := \{(x_1, y_1), \dots, (x_N, y_N)\}$, where $x_d \in \mathbb{R}^L$ and $y_d \in \{\pm 1\}$, $d = 1, \dots, N$. We consider the following optimal margin nonlinear support vector machine (SVM) classification problem:

$$\max_{\alpha_d} \sum_{d=1}^{N} \alpha_d - \frac{1}{2} \sum_{d=1}^{N} \sum_{e=1}^{N} \alpha_d \alpha_e q_{d,e}$$

$$\text{such that } \alpha_d \geq 0 \; \forall d \text{ and } \sum_{d=1}^{N} \alpha_d y_d = 0. \tag{1}$$

where the α_d are the Lagrange multipliers of the corresponding *support vectors* (SVs) and $q_{d,e}$

are the entries of the positive definite matrix $Q := [\left(y_d y_e\, k(x_d, x_e)\right)_{d,e}]_{N \times N}$. The decision function classifying an input x is then $f(x) = \mathrm{sgn}\left(\sum_{l=1}^{N_D} \alpha_l y_l k(x, x_l) + b\right)$, where b is the bias term and $N_D < N$ is the number of support vectors [7].

Let us define a set of $\mathcal{M} := \{1, 2, \ldots, M\}$ separate *processing units* with access to different (possibly overlapping) subsets, $S_i,\ i \in \mathcal{M}$, of the labeled training data such that $S = \bigcup_{i=1}^{M} S_i$. The initial partition of the data S can be due to the nature of the specific problem at hand or as a result of a partitioning scheme in preprocessing. Given the partition, we define the vectors $\{\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(M)}\}$ with the ith one having a size of N_i (elements). In order to devise a distributed algorithm that solves the optimization problem (1), we first relax it by substituting the constraint $\sum_{d=1}^{N} \alpha_d y_d = 0$ by a quadratic penalty function, $0.5 M\beta(\sum_{d=1}^{N} \alpha_d y_d)^2$, where $\beta > 0$. We next impose an upper-bound α_{max} on α_d such that $\alpha_d < \alpha_{max}\ \forall d$. This upper-bound can be chosen to derive a soft margin hyperplane (i.e. maximizing the margin). Alternatively, it can be chosen very large to minimize the training error ignoring the margin. Thus, we define the constrained optimization problem

$$\max_{\alpha_d \in [0, \alpha_{max}]} F(\alpha) = \sum_{d=1}^{N} \alpha_d - \frac{1}{2} \sum_{d=1}^{N} \sum_{e=1}^{N} \alpha_d \alpha_e q_{d,e} - \frac{M\beta}{2} \left(\sum_{l=1}^{N} \alpha_l y_l \right)^2 \qquad (2)$$

which approximates (1) with a quadratic penalty function. Note that the objective function $F(\alpha)$ is strictly concave in all its arguments and the constraint set $X := [0, \alpha_{max}]^N$ is convex, compact, and nonempty.

We next partition the convex optimization problem (2) into M subproblems through *Lagrangian decomposition*. The i^{th} units optimization problem is

$$\underset{\alpha_d^{(i)} \in [0, \alpha_{max}]}{\text{maximize}}\ F_i(\alpha) = \sum_{d \in S_i} \alpha_d - \frac{1}{2} \sum_{d \in S_i} \sum_{e=1}^{N} \alpha_d \alpha_e q_{d,\,e} - \frac{\beta}{2} \left(\sum_{l=1}^{N} \alpha_l y_l \right)^2 \qquad (3)$$

Again, the individual objective functions $F_i(\alpha)$ are strictly concave in their all arguments and the respective constraint sets are convex, compact, and nonempty for all i.

3 Parallel Update Algorithm

We present a discrete-time parallel update scheme to solve (2) through the partitioning (3). Clearly, solving all unit problems at the same time is equivalent to finding the solution of the relaxed problem (2). One possible way of achieving this objective is to utilize a gradient algorithm that converges to the unique maximum, α^*, of (2) which closely approximates the one of the original optimization problem. Then, every unit implements the following gradient projection algorithm for each of its training samples

$$\begin{aligned} \alpha_d(n+1) = &\ [\alpha_d(n) + \kappa_d G_d(\alpha)]^+ \ \forall d, \text{ where} \\ G_d(\alpha(n)) := &\ 1 - \frac{1}{2}\left(\alpha_d(n) q_{dd} - \textstyle\sum_{e=1}^{N} \alpha_e(n) q_{d,e}\right) \\ &\ -\beta y_d \textstyle\sum_{l=1}^{N} \alpha_l(n) y_l\,. \end{aligned} \qquad (4)$$

Here n denotes the update instances and the notation $[\cdot]^+$ represents the orthogonal projection of a vector onto the convex set X defined by $[\alpha]^+ := \arg\min_{z \in X} \|z - \alpha\|_2$, where $\|\alpha\|_2$ is the Euclidean norm. In this special case, the projection of α_d onto X can be computed in a straightforward way by mapping α_d onto $[0, \alpha_{max}]$ for each d. This enables us to implement the parallel algorithm easily.

The function $F(\alpha)$ in (2) is a polynomial and hence continuously differentiable in its arguments. Furthermore, there exists a scalar constant C such that

$$\|\nabla F(\gamma) - \nabla F(\delta)\|_2 \le C \,\|\gamma - \delta\|_2\,,\ \forall \gamma, \delta \in X,$$

where ∇F is the gradient operator. Define $z := \gamma - \delta$. Then,

$$\|\nabla F(\gamma) - \nabla F(\delta)\|_2^2 = z^T\, A^T A\, z,$$

where

$$A := \frac{1}{2}\mathrm{diag}(Q) + \frac{1}{2} Q + \beta y\, y^T. \qquad (5)$$

The matrix $\mathrm{diag}(Q)$ contains the diagonal elements of Q with all its off-diagonal elements set to zero. The scalar constant C is then given by $C = \sqrt{\max \lambda(A^T A)}$, where $\max \lambda(\cdot)$ is the maximum eigenvalue. Therefore, the gradient of the objective function $F(\alpha)$, ∇F, is Lipschitz continuous. Moreover, it is bounded from above on X.

Theorem 1. *The gradient projection algorithm (4) converges to the unique maximum, α^*, of the objective function F in (2), if the step-size constant κ_d satisfies*

$$0 < \kappa_d < \frac{2}{\sqrt{\max \lambda(A^T A)}},\ \forall d.$$

Proof. The proof follows directly from the upper-boundedness and strict concavity of the function F and Lipschitz continuity of its gradient ∇F. We refer to Propositions 3.3 and 3.4 in [1, pp. 213] for details. □

Theorem 2. *Assume that the conditions in Theorem 1 hold. Then, the gradient projection algorithm (4) converges to the unique maximum α^* of (2) geometrically.*

Proof. This result follows directly from Proposition 3.5 [1, p. 215], if there exists a constant $c > 0$ such that

$$(\nabla F(\gamma) - \nabla F(\delta))^T(\gamma - \delta) \geq c \, \|\gamma - \delta\|_2^2 \, , \; \forall \gamma, \delta \in X.$$

Let us again define $z := \gamma - \delta$. Then,

$$\begin{aligned}(\nabla F(\gamma) - \nabla F(\delta))^T(\gamma - \delta) &= z^T A^T z \\ &\geq z^T \max \lambda(A^T) z,\end{aligned}$$

$\forall \gamma, \delta \in X$, where the matrix A is defined in (5). We note that the matrix A is the sum of two positive definite matrices $\mathrm{diag}(Q)$, Q and a positive semidefinite one, $y\,y^T$. It is hence positive definite and all of its eigenvalues are positive. Hence, there exists a positive constant c which satisfies the sufficient condition for the theorem to hold. □

4 Active Set Algorithm

Although the update algorithm presented in Section 3 provably converge to the unique solution of (2), and hence approximately solve the original binary classification problem (1), it often results in a large number of support vectors due to the relaxation involved. This is undesirable not only for efficiency reasons but also due to the communication overhead it brings to the system. To address this problem, we resort to *active set* methods [8].

For the relaxed problem (2), we have only non-negativity constraints on α. In this case, the vectors with $\alpha > 0$ constitute the set of support vectors. Hence, the set of support vectors is the complement of the active set and both sets are mutually exclusive. Furthermore, the union of both sets gives the (universal) set of all feature vectors. We consider, specifically, the greedy algorithm summarized in Fig. 1.

We note that the communication overhead of the active set-based approach is significantly lower than the plain algorithm presented in Section 3. The information flow within the distributed system depicted in Fig. 1 results, thus, in an efficient communication scheme. Furthermore, the iterative algorithm presented is very suitable for online learning where training data is dynamic.

5 Experiments

We present results of experiments with 256 pictures of animals which are downloaded from *flickr.com*. The animals are from 11 different species and are depicted in various sizes, body postures, and contexts. We consider the task of automatic image tagging on this dataset. Assume that a user tags only a single image per species and that different users tag different images. Fig. 2 shows images of tigers together with brush strokes indicating the location of the animals. Corresponding pairs of images and marks form the training data for our algorithm.

Given an image of an animal of a certain species, our system determines the bounding box of the brush strokes and randomly samples 100 image patches from within this bounding box. In order to characterize these patches, we apply color correlograms [3]. Correlograms extend the idea of histograms in that they count co-occurences of pixels of the same color at horizontal and vertical distances Δx and Δy, respectively. Therefore they also encode spatial information. 250 correlograms that serve as counter examples for training are created at random. In all experiments reported here, images patches are of size 100×60 pixels, Δx and Δy are set to 5 pixels and we apply color quantization using a

Algorithm 1 Active set-based greedy training algorithm.

```
Input: Datasets $S_1, \ldots, S_M$ and SVM kernel $k(\cdot)$
Parameters: $\beta$, $\kappa$, max updates
Output: SVs, corresponding $\alpha$ values, and bias $b$
Initialize active set $\mathcal{A}_0$ (randomly or problem specific)
while Number of updates $n <$ max updates do
    Make $\mathcal{A}(n)$ the current active set
    Compute gradients of the data points $x \in \mathcal{A}(n)$
    Remove data with max. gradient from $\mathcal{A}(n)$
    repeat
        Update $\alpha_d(t)$ $\forall d \notin \mathcal{A}(n)$ using Eq. (4)
        Find min. $\alpha$ and add respective SV to $\mathcal{A}(n+1)$
    until $\max_d(\alpha_d(t+1) - \alpha_d(t)) < \varepsilon$
end while
Determine bias term $b$
```

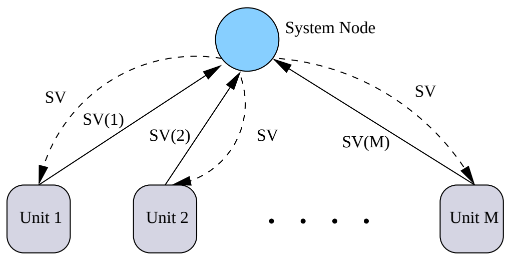

Figure 1. The information flow within the distributed system.

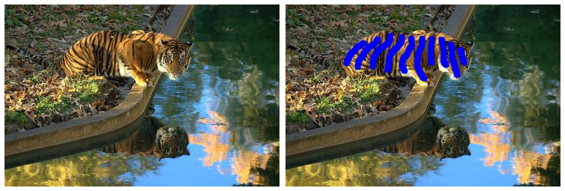

Figure 2. Example pictures of tigers and brush marks indicating the location of the object of interest.

palette of 256 colors. During runtime the test images are processed with a step size of 8 pixels. The corresponding local correlograms are classified as to whether or not they depict the animal in question.

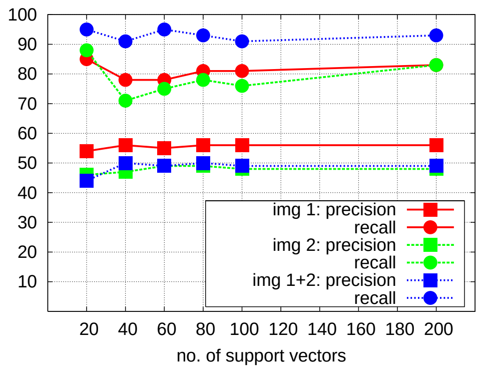

Figure 3. Precision and recall in experiments on tiger detection.

Fig. 3 shows quantitative results for three prototypical cases. It displays the precision and recall curves resulting from classification based on various numbers of support vectors obtained from training with only image 1, image 2, and image 1+2, respectively. Fig. 4 exemplifies cases of successful and less successful classification obtained after distributed training. Overall, these results demonstrate the good performance of the proposed approach. What is noticeable from Fig. 3 is that the classifier that was trained with twice the amount of input data achieves higher recall than the classifiers trained from only a single image while precision remains similar. It is noteworthy that the number of support vectors does not significantly influence the performance. With respect to the time-critical interactive application considered, this is an encouraging result, since it allows for reducing classification efforts without harming the performance.

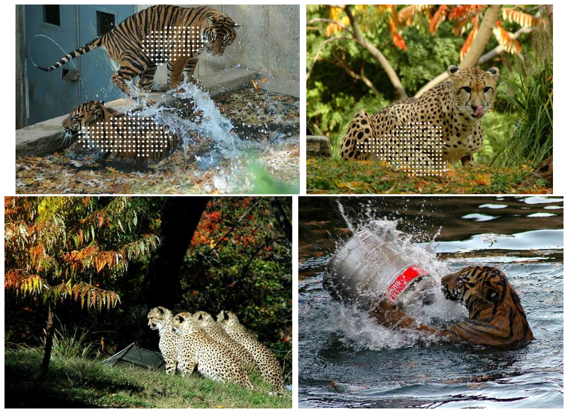

Figure 4. Examples of (clockwise from top left) true positive, false positive, false negative and true negative results from our tiger detection experiments.

References

[1] D. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Compuation: Numerical Methods*. Prentice Hall, Upper Saddle River, NJ, 1989.

[2] H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel Support Vector Machines: The Cascade SVM. In *Proc. NIPS*, pages 521–528, 2005.

[3] J. Huang, S. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image Indexing Using Color Correlograms. In *Proc. CVPR*, pages 762–768, 1997.

[4] T. Joachims. *Making Large-Scale Support Vector Machine Learning Practical*, chapter 11, pages 169–184. Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge, MA, USA, 1998.

[5] Y. Lu and V. Roychowdhury. Parallel Randomized Support Vector Machine. In *Proc. PAKDD*, pages 205–214, 2006.

[6] R. U. Pedersen. *Using Support Vector Machines for Distributed Machine Learning*. PhD thesis, University of Copenhagen, August 2004.

[7] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002.

[8] M. Vogt and V. Kecman. *Active-Set Methods for Support Vector Machines*, pages 133–158. Support Vector Machines: Theory and Applications. Springer-Verlag, Berlin, Heidelberg, August 2005.

[9] G. Zanghirati and L. Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Comput.*, 29(4):535–551, 2003.