Distributed Training of Structured SVM

Ching-pei Lee, Kai-Wei Chang, Shyam Upadhyay, Dan Roth
University of Illinois at Urbana-Champaign
{cleel49, kchang10, upadhy3a, danr} @illinois.edu

Abstract

Training structured learning models is time-consuming and current approaches are limited to using a single machine. Thus, the advantage of more computing power and the capacity for larger data set has not been exploited for structured learning. In this work, we propose two efficient algorithms for distributedly training large-scale structured support vector machines. One is based on alternating direction method of multipliers, while the other is based on a recently proposed distributed block-coordinate descent method. Theoretical and experimental results both indicate that our methods are efficient for training structured models.

1 Introduction

Many tasks in natural language processing and computer vision can be formulated as structured prediction problems. The goal is to assign values to multiple interdependent variables, where the interdependencies constitute the "structure". Examples include image segmentation, scene interpretation, dependency parsing of sentences, and co-reference resolution of entities in a document. To fully exploit the rich representation of the structures, it is essential to use large amount of data. However, in practice, only a limited amount of data can be used to train a structured model because most current approaches for structured learning are confined to a single machine, and the limit is imposed by the memory and disk capacity. The inability of training structured prediction models with larger volume of data limits the prediction performance. In the context of "standard" linear classification (i.e., binary classification), this problem has been addressed by developing distributed training algorithms (see, for example, [25, 11, 10, 23, 1]). However, little work except for structured perceptron [14] has been done to develop distributed algorithms for structured learning.

We notice that there are many difficulties in training structured models in a distributed fashion. First, the features vectors are generated using both the output structures and the input instances. When this information is stored across different machines, features generated on different machines could represent different structures. Second, in distributed structured learning, both communication and inference are expensive. In order to design an efficient distributed algorithm, it is important to consider both factors. This is in contrast to binary classification, where reduction in the communication overhead alone significantly reduces the training time.

In this work, we address these challenges and propose two efficient distributed algorithms for training structured support vector machines (SSVM) [20]. The first algorithm, solving the dual problem, is inspired by the recently proposed distributed box-constrained quadratic optimization algorithm [10]. The other one is based on the alternating direction method of multipliers (ADMM) [3] and directly deal with the primal problem. We provide theoretical analysis and show in both algorithms, convergence is guaranteed. Especially, in one of the algorithm proposed, global linear convergence rate can be obtained. Moreover, we show that the per-machine local sub-problems in both algorithms can be formed as small SSVM problems, which can be efficiently solved by off-the-shelf solvers. This enables us to leverage the well-studied single-machine structured learning methods such as the dual coordinate decent algorithm [5]. Experiments show that our algorithms are very fast in practice and are suitable for training large-scale structured models.
This paper is organized as follows. Section 2 briefly introduces the distributed structured learning problem. Our algorithms are presented in Sections 3-4. Their analysis and important implementation issues are discussed in Section 5. Section 6 describes related work, and the experimental results are presented in Section 7. Section 8 concludes this work.

## 2 Structured Support Vector Machine

Given a set of observations $D = \{(x_i, y_i)\}_{i=1}^l$, where $x_i \in \mathcal{X}$ are instances with their annotated structure $y_i \in \mathcal{Y}_i$, and $\mathcal{Y}_i$ is the set of all feasible structures for $x_i$. SSVM solves the following optimization problem

$$
\min \limits_{w, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^l \ell(\xi_i)
$$

subject to $w^T \phi(y, y_i, x_i) \geq \Delta(y, y_i) - \xi_i, \forall y, \forall y \in \mathcal{Y}_i,$

where $C > 0$ is the penalty parameter of the loss term specified by the user,

$$
\phi(y, y_i, x_i) = \Phi(x_i, y_i) - \Phi(x_i, y),
$$

$\Phi(x, y)$ is the feature vector generated by considering the input $x$ and the structure $y$ together, $\ell(\xi)$ is the loss term to be minimized, and the loss function $\Delta(y, y_i) \geq 0$ is a metric that represents the distance between structures. When we use $\ell(x) = x$ and $\ell(y) = y^2$, (1) is referred to as L1-loss and L2-loss SSVM, respectively. Due to the space limitations, we focus our discussion on L2-loss SSVM, but our algorithms and analysis are applicable to L1-loss SSVM as well.  

Because (1) is non-differentiable and hard to optimize directly, most efficient algorithms for SSVM solve its dual problem. Let $\alpha$ be the vector of the dual variables of dimension $\prod |\mathcal{Y}_i|$, and $\otimes$ be the Kronecker product, the dual problem of L2-loss SSVM can be written as,

$$
\min \limits_{\alpha \geq 0} f(\alpha) \equiv \frac{1}{2} \alpha^T (Q + \frac{1}{2C} A) \alpha - w^T \alpha,
$$

where

$$
Q_{(i,y_1), (j,y_2)} = \phi(y_1, y_i, x_i)^T \phi(y_2, y_j, x_j), \forall 1 \leq i, j \leq l, \forall y_1 \in \mathcal{Y}_i, \forall y_2 \in \mathcal{Y}_j,
$$

$$
A = (I \otimes e)^T (I \otimes e),
$$

$$
v_{(i,y)} = \Delta(y, y_i), \forall 1 \leq i \leq l, \forall y \in \mathcal{Y}_i,
$$

and $e$ is a vector of ones.

From the KKT conditions, the respective optimal solutions $w^*$ and $\alpha^*$ for (1) and (2) satisfy

$$
w^* = \sum_{i,y} \alpha^*_{i,y} \phi(y, y_i, x_i).
$$

The key challenge when solving (2) is that for most applications, the size of $\mathcal{Y}_i$ and thus the dimension of $\alpha$ is exponentially large (with respect to the length of $x_i$), so optimizing over all variables is unrealistic. Efficient dual methods maintain a working set of dual variables to optimize such that variables not in the working set are fixed to be zero. These methods then iteratively enlarge the working set until the problem is well-optimized [20, 5, 9]. In augmenting the working set at each iteration, given the current $w$ and any $i$, we add to the working set the dual variable $\alpha_{i,y}$ corresponds to the structure $y$

$$
y = \text{arg max}_{y \in \mathcal{Y}_i} w^T \phi(y, y_i, x_i) - \Delta(y, y_i).
$$

After the dual problem over the working set is optimized, (3) is used to update the current $w$.  

---

1 L1-loss SSVM has an additional linear constraint, which is seemingly difficult to fit in the box-constrained optimization framework of [10]. However, the algorithm is still applicable and the convergence rate analysis is still valid because this constraint can be viewed as a polyhedral feasible set, and the error bound used in [10] still holds in this case. For ADMM, extending it to L1-loss SSVM is straightforward.
Algorithm 1: A box-constrained quadratic optimization algorithm for solving (1)

1. \( w \leftarrow 0, \alpha \leftarrow 0 \).
2. For \( t = 0, 1, \ldots \) (outer iteration)
   2.1. Use the current \( w \) to compute \( \nabla f(\alpha) \) by (7).
   2.2. Solve (5) to get \( d \) distributedly in \( K \) machines.
   2.3. Use allreduce to obtain \( \Delta w \) in (8) and (9)-(10).
   2.4. Compute \( \eta \) by (11) by another \( O(1) \) communication.
   2.5. \( \alpha \leftarrow \alpha + \eta d \).
   2.6. \( w \leftarrow w + \eta \Delta w \).

In this work we solve SSVM in a distributed environment, where the training data is disjointly stored in \( K \) machines. We call the set of instances stored in the \( k \)-th machine the \( k \)-th partition, and index them by \( J_k \). We have

\[
\bigcup_{i=1}^{K} J_i = D, J_i \cup J_k = \phi, \forall i \neq k.
\]

We call the step of computing (4) “inference”, and call the part of optimizing (2) over a fixed working set “learning”. The key difficulty of training SSVM distributedly is that usually both communication required in the learning step and inference are expensive. Therefore, we need to consider methods that ensure fewer rounds of both parts.

3 Distributed Box-Constrained Quadratic Optimization for SSVM

In this section, we propose an efficient distributed algorithm for training (2).

We note that (2) is in the form of a quadratic box-constrained optimization problem, and we can apply the recently proposed distributed box-constrained quadratic optimization framework [10] to solve it. Under this framework, at each iteration, given the current \( \alpha \), we compute

\[
d = \arg \min_{d: \alpha + d \geq 0} \nabla f(\alpha)^T d + \frac{1}{2} d^T H d
\]

for some symmetric positive definite \( H \). We then conduct a line search to decide a suitable step size \( \eta \) and update \( \alpha \leftarrow \alpha + \eta d \). For efficiently solving (5) without incurring communication, we consider block-diagonal matrices that each block uses only information from \( J_k \). To also approximate the real Hessian closely, we consider

\[
H \equiv \theta \bar{Q} + \frac{1}{2C} A + \lambda I,
\]

where \( \lambda > 0 \) is a small constant to ensure positive-definiteness, \( \theta > 0 \) can be tuned to decide how conservative the updates are, and

\[
\bar{Q}(i, y_1, j, y_2) = \begin{cases} 
0 & \text{if } i, j \text{ are not in the same partition,} \\
\phi(y_1, y_i, x_i)^T \phi(y_2, y_j, x_j) & \text{otherwise.}
\end{cases}
\]

Using this design, (5) can be decomposed into \( K \) disjoint sub-problems that can be solved locally. The form of the sub-problems is very similar to a smaller SSVM dual problem. Thus, one can adopt existing single-machine methods for SSVM to efficiently solve the sub-problems that have exponentially many variables. The choice of local solvers is discussed in Section 5. Using the form of (5), if \( w \) is available to all machines, we can obtain the part of \( \nabla f(\alpha) \) that corresponds to the coordinates in \( J_k \) by

\[
\nabla f(\alpha)(i, y) = w^T \phi(y, y_i, x_i) + \sum_{y' \in \mathcal{Y}_i} \alpha_{i, y'} - \Delta(y, y_i).
\]

After (5) is solved, we compute

\[
\Delta w \equiv \sum_{i, y} d_{i, y} \phi(y, y_i, x_i)
\]
by an allreduce operation that communicates information between machines. This information also synchronizes the model for conducting inferences to enlarge the working set. Using $\Delta w$ and two additional scalar in the communication, we can get

\[
\alpha^T Q d = w^T \Delta w, \\
d^T Q d = \Delta w^T \Delta w, \\
\alpha^T A d = \sum_i (\sum_y \alpha_{i,y} \sum_y d_{i,y}), \\
d^T A d = \sum_i (\sum_y d_{i,y} \sum_y d_{i,y}).
\]  

Thus an exact line search for deciding $\eta$ can be conducted.

\[
\frac{\partial f(\alpha + \eta d)}{\partial \eta} = 0 \Rightarrow \eta^* = \frac{-\nabla f(\alpha)^T d}{d^T (Q + A/2C)d} = -\frac{\alpha^T (Q + A/2C)d - w^T d}{d^T (Q + A/2C)d}.
\]

To ensure feasibility, we take
\[
\eta = \min(\max\{\eta' \mid \alpha + \eta' d \geq 0\}, \eta^*).
\]

Note that this value is obtained by another $O(1)$ communication. $\alpha$ and $w$ are then updated by
\[
\alpha \leftarrow \alpha + \eta d, \\
w \leftarrow w + \eta \Delta w.
\]

The detailed description is in Algorithm [\ref{alg:admm}].

4 Alternating Direction Method of Multipliers for SSVM

In this section, we investigate a distributed approach of directly solving (1) based on ADMM. (1) can be reformulated as an unconstrained optimization problem:

\[
\min_w \frac{1}{2} w^T Q w + C \sum_i \max_{y \in Y_i} (\Delta(y, y_i) - w y^T \phi(y, y_i, x_i))^2.
\]

As mentioned in Section [2], we assume that the data is disjointly split, and the subset in the $k$-th machine is indexed by $J_k$. Then, we can rewrite (12) to a consensus minimization problem that is equivalent to (12):

\[
\min_{w = w_k, \ldots, w_K} \frac{1}{2} w^T Q w + C \sum_{k=1}^K \sum_{i \in J_k} \max_{y \in Y_i} (\Delta(y, y_i) - w_k y^T \phi(y, y_i, x_i))^2.
\]

The $k$-th machine maintains two vectors $w_k$ and $u_k$. Machines communicate with each other to obtain a consensus model $\bar{w}$. Following [3], we can derive an ADMM algorithm that updates the model using the following rules at iteration $t$:

\[
w^{t+1}_k \leftarrow \arg \min_{w_k} \frac{\rho}{2} \| w_k - (\bar{w}^t - u_k^t) \|^2 + C \sum_{i \in J_k} \max_{y \in Y_i} (\Delta(y, y_i) - w_k^t y^T \phi(y, y_i, x_i))^2,
\]

\[
\bar{w}^{t+1} \leftarrow \arg \min_w \frac{1}{2} \bar{w}^T Q \bar{w} + \frac{K \rho}{2} \left\| \bar{w} - \frac{1}{K} \sum_{k=1}^K w_k^{t+1} - \frac{1}{K} \sum_{k=1}^K u_k^t \right\|^2,
\]

\[
u_k^{t+1} \leftarrow \bar{w}_k - w_k - \bar{w}_k - u_k^t + u_k^{t+1}.
\]

The first step (14) requires solving a sub-problem. If we let $w^t_0 = \bar{w}^t - u^t_k$, intuitively, each machine solves a SSVM problem on a subset of data with a regularizer that ensures the solution to be close to the model $w^t_0$. Let $\bar{w}_k = w_k - w^t_0$, (14) is equivalent to:

\[
\min_{w_k} \frac{1}{2} \| \bar{w}_k \|^2 + C \rho \sum_{i \in J_k} \max_{y \in Y_i} (\Delta(y, y_i) - (\bar{w}_k + w^t_0)^T \phi(y, y_i, x_i))^2.
\]
Algorithm 2: ADMM for SSVM

1. \( \bar{w}^0 \leftarrow 0, u_k^0 \leftarrow 0, \forall k \).
2. For \( t = 0, 1, \ldots \) (outer iteration)
   1. Update \( w_k^{t+1} \), \( \forall k \) by (12) in parallel such that each thread or each machine solves SSVM on a partition of the training data.
   2. Communicate \( w_k^{t+1}, u_k^t \) to update \( \bar{w}^{t+1} \) by (18).
   3. Update \( u_k^{t+1}, \forall k \) in parallel using the updated \( \bar{w}^{t+1} \) and \( w_k^{t+1} \) by (16).

(17) is a SSVM problem with the penalty parameter \( \frac{C}{\rho} \), and the loss function \( \Delta(y, y_i) + w_0^T \phi(y, y_i, x_i) \). Therefore, it can be solved by off-the-shelf SSVM solvers.

By setting the first derivative to zero, the minimization problem in the second step (15) has a closed form solution:

\[
\bar{w}^{t+1} \leftarrow \frac{\rho}{K \rho + 1} \sum_{k=1}^{K} (w_k^{t+1} + u_k^t).
\]  

(18)

Similar to (8), in a distributed environment, the summation in (18) requires an allreduce operation to communicate information across machines. Intuitively, (18) adds the average of \( \bar{w}_k \) to the consensus model \( \bar{w} \), and rescales the whole weight vector.

Finally, the last step (16) calculates the difference between the current model \( w_k \) and the consensus model \( \bar{w} \). The detailed algorithm is described in Algorithm 2.

5 Analysis and Discussion

In this section, we discuss the time complexity of Algorithms 1 and 2 and important practical issues.

5.1 Iteration complexity and Sub-problem Solver

Following the analysis in [10], we can show the following convergence result for Algorithms 1.

Theorem 1. Algorithm 1 requires \( O(\log(1/\epsilon)) \) iterations to obtain an \( \epsilon \)-accurate solution for (2).

For Algorithm 2 [16] proved that if the objective is convex, bounded from below, and the constraints satisfy some conditions, then the ADMM algorithm converges to an optimal solution. The consensus SSVM problem (13) is a convex problem and is bounded from below. Moreover, the constraints in (13) satisfy the conditions required in [16, Theorem 1]. Therefore, the ADMM algorithm for SSVM converges.

Solving (5) and (17) are the same as solving \( K \) smaller SSVM problems in parallel, and many efficient algorithms have been proposed [9, 20, 19, 5, 18]. These methods all rely on conducting (4) on instances to select the working set. They differ in the solver used for the sub-problems, and the strategy to update \( w \) after selecting the working set. We note that all these methods can be used as our sub-problem solver.

In practice, we also notice that solving (5) or (17) approximately (i.e., use less iterations in the sub-problem solver) may sometimes improve the training speed.

The communication steps in (8) and (13) require machines to communicate a vector of \( O(n) \). The actual cost of this communication depends on the network setting and usually grows with larger \( K \).

5.2 Model Consistency

Unlike binary classification, while learning a structured model, features are usually generated on-the-fly because the features depend on \( y \), and the set \( Y_i \) is decided by the structures the solver has seen so far. As a result, if each machine maintains its own feature mapping, the feature indices will be inconsistent across different machines. For example, when training a part-of-speech (POS) tagger, each machine observes different sets of words in different orders. Therefore, there is a need to ensure the feature indices associated with words are consistent in different machines.
One potential solution is to synchronize the feature mappings at each round. However, this approach incurs a huge communication overhead. To tackle this issue, we adapt a feature hashing strategy \[21\] in our framework. When a feature is generated, we use a unique hashing function to map the feature index into an integer value in \([0, 2^d]\), \(d \in \mathbb{N}\) and use it as the new feature index. Thus the size of the weight vector is at most \(2^d\). For tasks using string features, a hash function is used to map strings into integers. This strategy has been used in distributed environments \[2, 12\] mainly for dimension reduction and fast feature lookup. However, we note that this hashing technique is essential for keeping feature indices across machines consistent in the structured task, and thus making distributed training possible.

6 Related Works

Our algorithms are based on the framework of distributed linear classification. Algorithm \[1\] is related to the quadratic box-constrained optimization framework \[10\] and Algorithm \[2\] is inspired by ADMM \[3\]. Distributed algorithms for optimizing linear classification models have been studied extensively, however, most of these algorithms cannot be directly used for structured learning. Many efficient distributed algorithms for the primal problem \[25, 11, 2\] rely on the differentiability of the problems to guarantee fast convergence. However, since \((1)\) is not differentiable, these methods cannot be directly applied. On the other hand, several methods (for example, \[22, 17\]) have been proposed to solve the dual problems with pointwise loss terms in the form of

\[
\ell(w^T x_i, y_i).
\]  

(19)

These methods, too, cannot be directly applied to \((1)\) because of the complex loss function is not in the form of \((19)\). Fortunately, by leveraging the framework of \[10\], these methods can be adopted to solve \((2)\) with proper modifications. However, since \((2)\) is non-strongly-convex, most of these dual methods have only \(O(1/\epsilon)\) convergence rate. For the SSVM dual problem, only our method in Section 4 extending the framework in \[10\] provides global linear convergence rate \(O(\log(1/\epsilon))\). Note that because the outer iteration complexity affects solving the costly sub-problems \((5)\) and \((14)\) with many time-consuming inference steps, it is essential to use methods with lower iteration complexity.

Distributed version of the structured perceptron \[7\] was proposed in \[14\] using the map-reduce framework. To the best of our knowledge, this is the only existing work for distributed structured learning. Their approach is similar to our methods in synchronizing the model by communicating \(w\). However, the major difference is that in the SSVM case we are optimizing a certain objective, while in the structured perceptron case the approach is to simply average the update results on each machine, and this does not guarantee converging to the same model generated by using only a single machine. We also observe that the learning step of SSVM can effectively reduce the number of inferences required for achieving a stable accuracy. \[24\] modified the algorithm in \[14\] by using minibatch updates. However, their parallel algorithm is designed for a multi-core machine with shared memory. It is unclear how to extend their approach to a distributed setting.

A parallel algorithm for training \((1)\) in the multi-core setting was developed by \[4\]. Their idea is to decouple the learning and the inference part to different threads. Only one thread is used for optimizing \((2)\) and all other threads are dedicated to inference. We note that this approach cannot be directly adopted in the distributed setting because it assumes zero communication overhead between threads, whereas the communication overhead is substantial in distributed environments. However, if each node has multiple cores, we can combine this method with Algorithm \[1\] to obtain significant speedup. The speedup is possible because the expensive communication and the expensive inference can be conducted simultaneously without blocking each other.

7 Experiments

We perform experiments on two structured prediction tasks. They are part-of-speech tagging (POS) and dependency parsing (DP).
Figure 1: Comparison between different algorithms using eight nodes. Training time is in log scale. The horizontal line represents the result of simple average of models trained using only local data in each machine. This is obtained from the first iteration result of ADMM-STRUCT.

Figure 2: Performance of BQO-STRUCT using different number of machines. Time is in log scale. Results show that BQO-STRUCT leverages the distributed learning environment and obtains a reasonable model more quickly when using more machines.

7.1 Problem Settings

POS is a sequential labeling task, where we aim at learning part-of-speech tags assigned to each word in a sentence. Each tag assignment (there are 45 possible tag assignment) depends on the associated word, the surrounding words, and their part-of-speech tags. The inference in POS is solved by the Viterbi algorithm. We evaluate our model by the per-word tag accuracy.

For DP, the goal is to learn, for each sentence, a tree structure which describes the syntactic dependencies between words. We use the graph-based parsing formulation and the features described in [15], where we find the highest scoring parse using the Chu-Liu-Edmonds [6] algorithm. We evaluate the parsing accuracy using the unlabeled attachment score, which measures the fraction of words that have correctly assigned parents.

For both tasks, we use the Wall Street Journal portion of the Penn Treebank [13] with the standard split for training (section 02-21) and test (section 23). The training and test corpus contains 39,832 and 4,832 sentences, respectively.
7.2 Distributed Learning Algorithms for Structured Learning

We compare the following algorithms.

- **ADMM-STRUCT**: The algorithm we proposed in Section 4.
- **BQO-STRUCT**: The algorithm we proposed in Section 3.
- **DISTRIBUTED PERCEPTRON**: A parallel structured Perceptron algorithm described in [14].
- **Simple average**: Each machine trains a separate model using the local data. The final model is obtained by averaging all local models. This is equivalent to the first iteration of ADMM-STRUCT.

We use eight nodes in a local cluster to conduct our experiments. All algorithms are implemented in JAVA, and the distributed platform is open MPI [8]. Data sets are uniform randomly split into eight partitions.

For solving the sub-problems in ADMM-STRUCT and BQO-STRUCT, we use the dual coordinate descent solver in [5] that is shown to be empirically faster than other existing methods. Although we can alter the number of inferences between two rounds of communications, and fine-tuning this number may improve the training speed, we realize that it is not realistic for users to spend time on tuning parameters that do not affect the performance but just the training time. Thus, for all algorithms, we fix the number of inferences through all instances between any two rounds of communications to be one, so that the number of inference rounds is identical to the number of communication rounds. For BQO-STRUCT and ADMM-STRUCT, each time in solving the local sub-problem with a fixed working set, we let the local coordinate descent solver pass through the local instances ten times. We note that this number of iterations may also affect the training speed but we do not fine-tune this parameter for the same reason above. For both tasks, we set $C = 0.1$ for SSVM. For ADMM-STRUCT, the parameter $\rho$ also only affects the training speed. Instead of fine-tuning it, a fixed value $\rho = 1.0$ is used. Following the setting in [22], we use $\theta = K$ in (6) of BQO-STRUCT.

Because BQO-STRUCT solves the dual problem [2], ADMM-STRUCT optimizes the primal problem [1] and DISTRIBUTED PERCEPTRON considers a different problem, the only criterion we can use to fairly compare these algorithms is the test performance.

Experimental results are shown in Figure 1. BQO-STRUCT performs the best in both tasks, while ADMM-STRUCT is worse than DISTRIBUTED PERCEPTRON in DP, but better in POS. The superior result of BQO-STRUCT confirms its theoretical linear convergence rate, while we notice ADMM-STRUCT converges slower in DP and hence leads to inferior performance.

We further investigate the speedup of the best solver BQO-STRUCT in Figure 2. We notice that for the time-consuming task DP, the speedup is significant because a large portion of the training time is spent on inference. Parallelizing this part can achieve nearly linear speedup. While for POS, because the training time using a single machine is already fast enough, using multiple machines does not improve the training time much.

Surprisingly, though there is the additional communication cost, yet our training time seems to be less than that of existing multi-core methods using more cores reported in [4]. A possible reason is that the speedup of multi-core algorithms is limited by the memory access bandwidth, which is not an issue when using multiple machines.

8 Conclusion

This work addresses the challenge of training structured SVM problems in a distributed setting and proposes two efficient algorithms with fast convergence rate in this setting. Experiments show that our methods significantly outperform the only existing distributed structured learning method and enable training structured SVM with a number of instances that goes beyond the capacity of a single machine. We hope that this work will inspire more applications of structured learning with large volume of training data and this supports the performance improvement on structured learning tasks.

Based on this work, a software will be released for public use.
References

[1] A. Agarwal, O. Chapelle, M. Dudik, and J. Langford. A reliable effective terascale linear learning system. Journal of Machine Learning Research, 15:1111–1133, 2014.
[2] A. Agarwal, O. Chapelle, M. Dudik, and J. Langford. A reliable effective terascale linear learning system. Journal of Machine Learning Research, 2014.
[3] 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, pages 1–122, 2011.
[4] K.-W. Chang, V. Srikumar, and D. Roth. Multi-core structural SVM training. In ECML, 2013.
[5] M.-W. Chang and W.-T. Yih. Dual coordinate descent algorithms for efficient large margin structural learning. Transactions of the Association for Computational Linguistics, 2013.
[6] Y. J. Chu and T. H. Liu. Relational dependency networks. Science Sinica, 1965.
[7] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In EMNLP, 2002.
[8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squiryes, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In European PVM/MPI Users' Group Meeting, pages 97–104, 2004.
[9] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. Machine Learning, 2009.
[10] C.-P. Lee and D. Roth. Distributed box-constrained quadratic optimization for dual linear SVM. In ICML, 2015.
[11] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin. Large-scale logistic regression and linear support vector machines using Spark. In Proceedings of the IEEE International Conference on Big Data, pages 519–528, 2014.
[12] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. Proc. VLDB Endow., 5(8), 2012.
[13] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. Computational Linguistics.
[14] R. McDonald, K. Hall, and G. Mann. Distributed training strategies for the structured Perceptron. In ACL, 2010.
[15] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In EMNLP, 2005.
[16] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. Technical report, 2011.
[17] D. Pechyony, L. Shen, and R. Jones. Solving large scale linear SVM with distributed block minimization. In NIPS 2011 Workshop on Big Learning: Algorithms, Systems, and Tools for Learning at Scale, 2011.
[18] S. K. Shevade, B. P., S. Sundararajan, and S. S. Keerthi. A sequential dual method for structural SVMs. In SDM, 2011.
[19] C. H. Teo, S. Vishwanathan, A. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. Journal of Machine Learning Research, 11:311–365, 2010.
[20] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research, 2005.
[21] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In ICML, 2009.
[22] T. Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In NIPS, 2013.
[23] C. Zhang, H. Lee, and K. G. Shin. Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In AISTATS, 2012.

[24] K. Zhao and L. Huang. Minibatch and parallelization for online large margin structured learning. In NAACL, pages 370–379, 2013.

[25] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. Distributed Newton method for regularized logistic regression. In PAKDD, 2015.