Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Combining ADMM and the Augmented Lagrangian Method for Efficiently Handling Many Constraints

Joachim Giesen[1] and **Sören Laue**[*1,2]
[1]Friedrich-Schiller-Universität Jena
[2]Data Assessment Solutions

Abstract

Many machine learning methods entail minimizing a loss-function that is the sum of the losses for each data point. The form of the loss function is exploited algorithmically, for instance in stochastic gradient descent (SGD) and in the alternating direction method of multipliers (ADMM). However, there are also machine learning methods where the entailed optimization problem features the data points not in the objective function but in the form of constraints, typically one constraint per data point. Here, we address the problem of solving convex optimization problems with many convex constraints. Our approach is an extension of ADMM. The straightforward implementation of ADMM for solving constrained optimization problems in a distributed fashion solves constrained subproblems on different compute nodes that are aggregated until a consensus solution is reached. Hence, the straightforward approach has three nested loops: one for reaching consensus, one for the constraints, and one for the unconstrained problems. Here, we show that solving the costly constrained subproblems can be avoided. In our approach, we combine the ability of ADMM to solve convex optimization problems in a distributed setting with the ability of the augmented Lagrangian method to solve constrained optimization problems. Consequently, our algorithm only needs two nested loops. We prove that it inherits the convergence guarantees of both ADMM and the augmented Lagrangian method. Experimental results corroborate our theoretical findings.

1 Introduction

Optimization problems with many constraints typically arise from large data sets. An illustrative example is the core vector machine [Tsang *et al.*, 2007], where the smallest enclosing ball for a given set of data points has to be computed. The objective function here is the radius of the ball that needs to be minimized, and every data point contributes a convex constraint, namely the distance of the point from the center must be at most the radius.

The increasing availability of distributed hardware suggests to address such problems by distributing the constraints on different compute nodes. Unfortunately, to the best of our knowledge, algorithmic schemes for distributing convex constraints are only known in a few special cases. Such a scheme has not even been discussed for the well researched smallest enclosing ball (core vector machine) problem. The situation is vastly different when there is not a large number of constraints but a large number parameters. For instance, the alternating direction method of multipliers (ADMM) that was proposed by [Glowinski and Marroco, 1975] and by [Gabay and Mercier, 1976] already decades ago obtained considerable attention, because it allows to solve convex optimization problems with a large number of parameters in a distributed setting [Boyd *et al.*, 2011]. For instance, the parameters of any log-likelihood maximization problem like logistic or ordinary least squares regression are just the data points. The loss function of such problems, that is, the negative log-likelihood function, is the sum of the losses for each data point. In this case ADMM lends itself to a distributed implementation where the data points are distributed on different compute nodes.

Surprisingly, so far no general convex inequality constraints have been considered directly in the context of ADMM. Although, in principle, standard ADMM can also be used for solving constrained optimization problems. A distributed implementation of the straightforward extension of ADMM leads to non-trivial constrained optimization subproblems that have to be solved in every iteration. Solving constrained problems is typically transformed into a sequence of unconstrained problems. Hence, this approach features three nested loops, the outer loop for reaching consensus, one loop for the constraints, and an inner loop for solving unconstrained problems. Alternatively, one could use the standard augmented Lagrangian method, originally known as the method of multipliers [Hestenes, 1969], that has been specifically designed for solving constrained optimization problems. Combining the augmented Lagrangian method with ADMM allows to solve general constrained problems in a distributed fashion by running the augmented Lagrangian method in an outer loop and ADMM in an inner loop. Again, we end up

*Contact Author

with three nested loops, the outer loop for the augmented Lagrangian method and the standard two nested inner loops for ADMM. Thus, one could assume that any distributed solver for constrained optimization problems needs at least three nested loops: one for reaching consensus, one for the constraints, and one for the unconstrained problems. The key contribution of our paper is showing that this is not the case. One of the nested loops can be avoided by merging the loops for reaching consensus and dealing with the constraints. Our approach, that only needs two nested loops, combines ADMM with the augmented Lagrangian method differently than the direct approach of running the augmented Lagrangian method in the outer and ADMM in the inner loop. The latter combination, that to our surprise has not been discussed in the literature before, still provides us with a good baseline in the experimental section.

Related Work

To the best of our knowledge, our extension of ADMM is the first distributed algorithm for solving general convex optimization problems with no restrictions on the type of constraints or assumptions on the structure of the problem. The only special case, that we are aware of, are quadratically constrained quadratic problems that has been addressed by [Huang and Sidiropoulos, 2016]. However, their approach, that builds on consensus ADMM, does not scale, since every constraint gives rise to a new subproblem.

[Mosk-Aoyama *et al.*, 2010] have designed and analyzed a distributed algorithm for solving convex optimization problems with separable objective function and linear equality constraints. Their algorithm blends a gossip-based information spreading, iterative gradient ascent method with the barrier method from interior-point algorithms. It is similar to ADMM and can also handle only linear constraints.

[Zhu and Martínez, 2012] have introduced a distributed multiagent algorithm for minimizing a convex function that is the sum of local functions subject to a global equality or inequality constraint. Their algorithm involves projections onto local constraint sets that are usually as hard to compute as solving the original problem with general constraints. For instance, it is well known via standard duality theory that the feasibility problem for linear programs is as hard as solving linear programs. This holds true for general convex optimization problems with vanishing duality gap.

In principle, the standard ADMM can also handle convex constraints by transforming them into indicator functions that are added to the objective function. However, this leads to subproblems that need to be solved in each iteration that entail computing a projection onto the feasible region. This entails the same issues as the method by [Zhu and Martínez, 2012] since computing these projections can be as hard as solving the original problem.

The recent literature on ADMM is vast. Most papers on ADMM stay in the standard framework of optimizing a function or a sum of functions subject to linear constraints. Exemplarily for many others, we just mention [Zhang and Kwok, 2014] who provide convergence guarantees for asynchronous ADMM and [Ghadimi *et al.*, 2015] who study optimal penalty parameter selection.

2 Alternating Direction Method of Multipliers

Here, we briefly review the alternating direction method of multipliers (ADMM) and discuss how it can be adapted for dealing with distributed data, before we extend it for handling convex constraints in the next section.

ADMM is an iterative algorithm that in its most general form can solve convex optimization problems of the form

$$\begin{array}{ll} \min_{x,z} & f_1(x) + f_2(z) \\ \text{s.t.} & Ax + Bz - c = 0, \end{array} \tag{1}$$

where $f_1 : \mathbb{R}^{n_1} \to \mathbb{R} \cup \{\infty\}$ and $f_2 : \mathbb{R}^{n_2} \to \mathbb{R} \cup \{\infty\}$ are convex functions, $A \in \mathbb{R}^{m \times n_1}$ and $B \in \mathbb{R}^{m \times n_2}$ are matrices, and $c \in \mathbb{R}^m$.

ADMM can obviously deal with linear equality constraints, but it can also handle linear inequality constraints. The latter are reduced to linear equality constraints by replacing constraints of the form $Ax \leq b$ by $Ax + s = b$, adding the slack variable s to the set of optimization variables, and setting $f_2(s) = \mathbf{1}_{\mathbb{R}^{\mathbf{m}}_{+}}(\mathbf{s})$, where

$$\mathbf{1}_{\mathbb{R}^{\mathbf{m}}_{+}}(\mathbf{s}) = \begin{cases} 0, & \text{if } s \geq 0 \\ \infty, & \text{otherwise,} \end{cases}$$

is the indicator function of the set $\mathbb{R}^m_+ = \{x \in \mathbb{R}^m | x \geq 0\}$. Note that f_1 and f_2 are allowed to take the value ∞.

Recently, ADMM regained a lot of attention, because it allows to solve problems with separable objective function in a distributed setting. Such problems are typically given as

$$\min_x \quad \textstyle\sum_i f_i(x),$$

where f_i corresponds to the i-th data point (or more generally i-th data block) and x is a weight vector that describes the data model. This problem can be transformed into an equivalent optimization problem, with individual weight vectors x_i for each data point (data block) that are coupled through an equality constraint,

$$\begin{array}{ll} \min_{x_i,z} & \sum_i f_i(x_i) \\ \text{s.t.} & x_i - z = 0 \quad \forall\, i, \end{array}$$

which is a special case of Problem 1 that can be solved by ADMM in a distributed setting by distributing the data.

3 ADMM Extension

Adding convex inequality constraints to Problem 1 does not destroy convexity of the problem, but so far ADMM cannot deal with such constraints. Note that the problem only remains convex, if all equality constraints are induced by affine functions. That is, we cannot add convex equality constraints in general without destroying convexity. Hence, here we consider convex optimization problems in its most general form

$$\begin{array}{ll} \min_{x,z} & f_1(x) + f_2(z) \\ \text{s.t.} & g_0(x) \leq 0 \\ & h_1(x) + h_2(z) = 0, \end{array} \tag{2}$$

where f_1 and f_2 are as in Problem 1, $g_0 : \mathbb{R}^{n_1} \to \mathbb{R}^p$ is convex in every component, and $h_1 : \mathbb{R}^{n_1} \to \mathbb{R}^m$ and $h_2 : \mathbb{R}^{n_2} \to \mathbb{R}^m$ are affine functions. In the following we assume that the problem is feasible, i.e., that a feasible solution

exists, and that strong duality holds. A sufficient condition for strong duality is that the interior of the feasible region is non-empty. This condition is known as Slater's condition for convex optimization problems [Slater, 1950].

Our extension of ADMM for solving Problem 2 and its convergence analysis works with an equivalent reformulation of Problem, where we replace $g_0(x)$ by

$$g(x) = \max\{0,\, g_0(x)\}^2,$$

with componentwise maximum, and turn the convex inequality constraints into convex equality constraints. Thus, in the following we consider optimization problems of the form

$$\begin{array}{ll} \min_{x,z} & f_1(x) + f_2(z) \\ \text{s.t.} & g(x) = 0 \\ & h_1(x) + h_2(z) = 0, \end{array} \tag{3}$$

where $g(x) = \max\{0,\, g_0(x)\}^2$, which by construction is again convex in every component. Note, though, that the constraint $g(x) = 0$ is no longer affine. However, we show in the following that Problem 3 can still be solved efficiently.

Analogously to ADMM our extension builds on the Augmented Lagrangian for Problem 3 which is the following function

$$L_\rho(x, z, \mu, \lambda) = f_1(x) + f_2(z) + \frac{\rho}{2} \left\|g(x)\right\|^2 + \mu^\top g(x) + \frac{\rho}{2} \left\|h_1(x) + h_2(z)\right\|^2 + \lambda^\top \left(h_1(x) + h_2(z)\right),$$

where $\mu \in \mathbb{R}^p$ and $\lambda \in \mathbb{R}^m$ are Lagrange multipliers, $\rho > 0$ is some constant, and $\|\cdot\|$ denotes the Euclidean norm. The Lagrange multipliers are also referred to as dual variables.

Algorithm 1 is our extension of ADMM for solving instances of Problem 3. It runs in iterations. In the $(k+1)$-th iteration the primal variables x^k and z^k as well as the dual variables μ^k and λ^k are updated.

Algorithm 1 ADMM for problems with non-linear constraints

input: instance of Problem 3
output: approximate solution $x \in \mathbb{R}^{n_1}, z \in \mathbb{R}^{n_2}, \mu \in \mathbb{R}^p, \lambda \in \mathbb{R}^m$
initialize $x^0 = 0, z^0 = 0, \mu^0 = 0, \lambda^0 = 0$, and ρ to some constant > 0
repeat
  $x^{k+1} := \operatorname{argmin}_x L_\rho(x, z^k, \mu^k, \lambda^k)$
  $z^{k+1} := \operatorname{argmin}_z L_\rho(x^{k+1}, z, \mu^k, \lambda^k)$
  $\mu^{k+1} := \mu^k + \rho g(x^{k+1})$
  $\lambda^{k+1} := \lambda^k + \rho \left(h_1(x^{k+1}) + h_2(z^{k+1})\right)$
until convergence
return $x^k, z^k, \mu^k, \lambda^k$

4 Convergence Analysis

From duality theory we know that for all $x \in \mathbb{R}^{n_1}$ and $z \in \mathbb{R}^{n_2}$

$$L_0(x^*, z^*, \mu^*, \lambda^*) \leq L_0(x, z, \mu^*, \lambda^*), \tag{4}$$

where L_0 is the Lagrangian of Problem 3 and x^*, z^*, μ^*, and λ^* are optimal primal and dual variables. Note, that x^*, z^*, μ^*, and λ^* are not necessarily unique. Here, they refer just to one optimal solution. Also note that the Lagrangian is identical to the Augmented Lagrangian with $\rho = 0$. Given that strong duality holds, the optimal solution to the original Problem 3 is identical to the optimal solution of the Lagrangian dual.

We need a few more definitions. Let $f^k = f_1(x^k) + f_2(z^k)$ be the objective function value at the k-th iterate (x^k, z^k) and let f^* be the optimal function value. Let $r_g^k = g(x^k)$ be the residual of the nonlinear equality constraints, i.e., the constraints originating from the convex inequality constraints, and let $r_h^k = h_1(x^k) + h_2(z^k)$ be the residual of the linear equality constraints in iteration k.

Our goal in this section is to prove the following theorem.

Theorem 1. *When Algorithm 1 is applied to an instance of Problem 3, then*

$$\lim_{k\to\infty} r_g^k = 0, \quad \lim_{k\to\infty} r_h^k = 0, \quad \textit{and} \quad \lim_{k\to\infty} f^k = f^*.$$

The theorem states primal feasibility and convergence of the primal objective function value. Note, however, that convergence to primal optimal points x^* and z^* cannot be guaranteed. This is the case for the original ADMM as well. Additional assumptions on the problem, like, for instance, a unique optimum, are necessary to guarantee convergence to the primal optimal points. However, the points x^k, z^k will be primal optimal and feasible up to an arbitrarily small error for sufficiently large k.

The proof of Theorem 1 can be found in the full version of this paper [Giesen and Laue, 2016].

5 Distributing Constraints

Finally, we are ready to discuss the main problem that we set out to address in this paper, namely solving general convex optimization problems with many constraints in a distributed setting by distributing the constraints. That is, we want to address optimization problems of the form

$$\begin{array}{lll} \min_x & f(x) & \\ \text{s.t.} & g_i(x) \leq 0 & i = 1 \ldots p \\ & h_i(x) = 0 & i = 1 \ldots m, \end{array} \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}^{p_i}$ are convex functions, and $h_i : \mathbb{R}^n \to \mathbb{R}^{m_i}$ are affine functions. In total, we have $p_1 + p_2 + \ldots + p_p$ inequality constraints that are grouped together into p batches and $m_1 + m_2 + \ldots + m_m$ equality constraints that are subdivided into m groups. For distributing the constraints we can assume without loss of generality that $m = p$. That is, we have m batches that each contain p_i inequality and m_i equality constraints.

Again it is easier to work with an equivalent reformulation of Problem 5, where each batch of equality and inequality constraints shares the same variables x_i, namely problems of the form

$$\begin{array}{lll} \min_{x_i,z} & \sum_{i=1}^m f(x_i) & \\ \text{s.t.} & \max\{0,\, g_i(x_i)\}^2 = 0 & i = 1 \ldots m \\ & h_i(x_i) = 0 & i = 1 \ldots m \\ & x_i = z, & \end{array} \tag{6}$$

where all the variables x_i are coupled through the affine constraints $x_i = z$. To keep our exposition simple, the objective function has been scaled by m in the reformulation.

For specializing our extension of ADMM to instances of Problem 6 we need the Augmented Lagrangian of this problem, which reads as

$$\begin{aligned} L_\rho(x_i, z, \mu_{i,g}, \mu_{i,h}, \lambda) = \\ \sum_{i=1}^{m} f(x_i) + \frac{\rho}{2} \sum_{i=1}^{m} \| \max\{0,\, g_i(x_i)\}^2 \|^2 \\ + \sum_{i}^{m} (\mu_{i,g})^\top \max\{0,\, g_i(x_i)\}^2 \\ + \frac{\rho}{2} \sum_{i=1}^{m} \|h_i(x_i)\|^2 + \sum_{i}^{m} (\mu_{i,h})^\top h_i(x_i) \\ + \frac{\rho}{2} \sum_{i=1}^{m} \|x_i - z\|^2 + \sum_{i}^{m} (\lambda_i)^\top (x_i - z), \end{aligned}$$

where $\mu_{i,g}, \mu_{i,h}$, and λ_i are the Lagrange multipliers (dual variables).

Note that the Lagrange function is separable. Hence, the update of the x variables in Line 5 of Algorithm 1 decomposes into the following m independent updates

$$\begin{aligned} x_i^{k+1} = \operatorname{argmin}_{x_i} \; & f(x_i) + \frac{\rho}{2} \| \max\{0,\, g_i(x_i)\}^2 \|^2 \\ & + (\mu_{i,g}^k)^\top \max\{0,\, g_i(x_i)\}^2 \\ & + \frac{\rho}{2} \|h_i(x_i)\|^2 + (\mu_{i,h}^k)^\top h_i(x_i) \\ & + \frac{\rho}{2} \|x_i - z^k\|^2 + (\lambda_i^k)^\top (x_i - z^k), \end{aligned}$$

that can be solved in parallel once the constraints $g_i(x_i)$ and $h_i(x_i)$ have been distributed on m different, distributed compute nodes. Note that each update is an unconstrained, convex optimization problem, because the functions that need to be minimized are sums of convex functions. The only two summands where this might not be obvious, are

$$\frac{\rho}{2} \| \max\{0,\, g_i(x_i)\}^2 \|^2 \quad \text{and} \quad (\mu_{i,g}^k)^\top \max\{0,\, g_i(x_i)\}^2.$$

For the first term note that the squared norm of a non-negative, convex function is always convex again. The second term is convex, because it can be shown by induction that the $\mu_{i,g}^k$ are always non-negative.

The update of the z variable in Line 6 of Algorithm 1 amounts to solving the following unconstrained optimization problems

$$\begin{aligned} z^{k+1} &= \operatorname{argmin}_z \sum_{i=1}^{m} \frac{\rho}{2} \|x_i^{k+1} - z\|^2 + \sum_{i=1}^{m} (\lambda_i^k)^\top (x_i^{k+1} - z) \\ &= \frac{\rho \sum_{i=1}^{m} x_i^{k+1} + \sum_{i=1}^{m} \lambda_i^k}{\rho \cdot m}, \end{aligned}$$

and the updates of the dual variables μ_i and λ_i are as follows

$$\begin{aligned} \mu_{i,g}^{k+1} &= \mu_{i,g}^k + \rho \, \max\{0,\, g_i(x_i^{k+1})\}^2, \\ \mu_{i,h}^{k+1} &= \mu_i^k + \rho \, h_i(x_i^{k+1}), \\ \lambda_i^{k+1} &= \lambda_i^k + \rho \left(x_i^{k+1} - z^{k+1}\right). \end{aligned}$$

That is, in each iteration there are m independent, unconstrained minimization problems that can be solved in parallel on different compute nodes. The solutions of the independent subproblems are then combined on a central node through the update of the z variables and the Lagrange multipliers. Actually, since the Lagrange multipliers $\mu_{i,g}$ and $\mu_{i,h}$ are also local, i.e., involve only the variables x_i^{k+1} for any given index i, they can also be updated in parallel on the same compute nodes where the x_i^k updates take place. Only the variables z and the Lagrange multipliers λ_i need to be updated centrally.

Looking at the update rules it becomes apparent that Algorithm 1 when applied to instances of Problem 6 is basically a combination of the standard Augmented Lagrangian method [Hestenes, 1969; Powell, 1969] for solving convex, constrained optimization problems and ADMM for solving convex optimization problems in a distributed fashion.

6 Experiments

We have implemented our extension of ADMM in Python using the NumPy and SciPy libraries, and tested this implementation on the robust SVM problem [Shivaswamy *et al.*, 2006] that has a second order cone constraint for every data point. In our experiments we distributed these constraints onto different compute nodes, where we had to solve an unconstrained optimization problem in every iteration. Since there is no other approach available that could deal with a large number of arbitrary constraints in a distributed manner we compare our approach to the baseline approach of running an Augmented Lagrangian method in an outer loop and standard ADMM in an inner loop. Note that this approach has three nested loops. The outer loop turns the constrained problem into a sequence of unconstrained problems (Augmented Lagrangians), the next loop distributes the problem using distributed ADMM, and the final inner loop solves the unconstrained subproblems using the L-BFGS-B algorithm [Morales and Nocedal, 2011; Zhu *et al.*, 1997] in our implementation.

Robust SVMs

The robust SVM problem has been designed to deal with binary classification problems whose input are not just labeled data points $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$, where the $x^{(i)}$ are feature vectors and the $y^{(i)}$ are binary labels, but a distribution over the feature vectors. That is, the labels are assumed to be known precisely and the uncertainty is only in the features. The idea behind the robust SVM is replacing the constraints (for feature vectors without uncertainty) of the standard linear soft-margin SVM by their probabilistic counterparts

$$\mathsf{Pr}\big[y^{(i)} w^\top x^{(i)} \geq 1 - \xi_i\big] \geq 1 - \delta_i$$

that require the now random variable $x^{(i)}$ with probability at least $1 - \delta_i \geq 0$ to be on the correct side of the hyperplane whose normal vector is w. Shivaswamy et al. show that the probabilistic constraints can be written as second order cone constraints

$$y^{(i)} w^\top \bar{x}^{(i)} \geq 1 - \xi_i + \sqrt{\delta_i/(1-\delta_i)} \, \big\|\Sigma_i^{1/2} w\big\|,$$

under the assumption that the mean of the random variable $x^{(i)}$ is the empirical mean $\bar{x}^{(i)}$ and the covariance matrix of

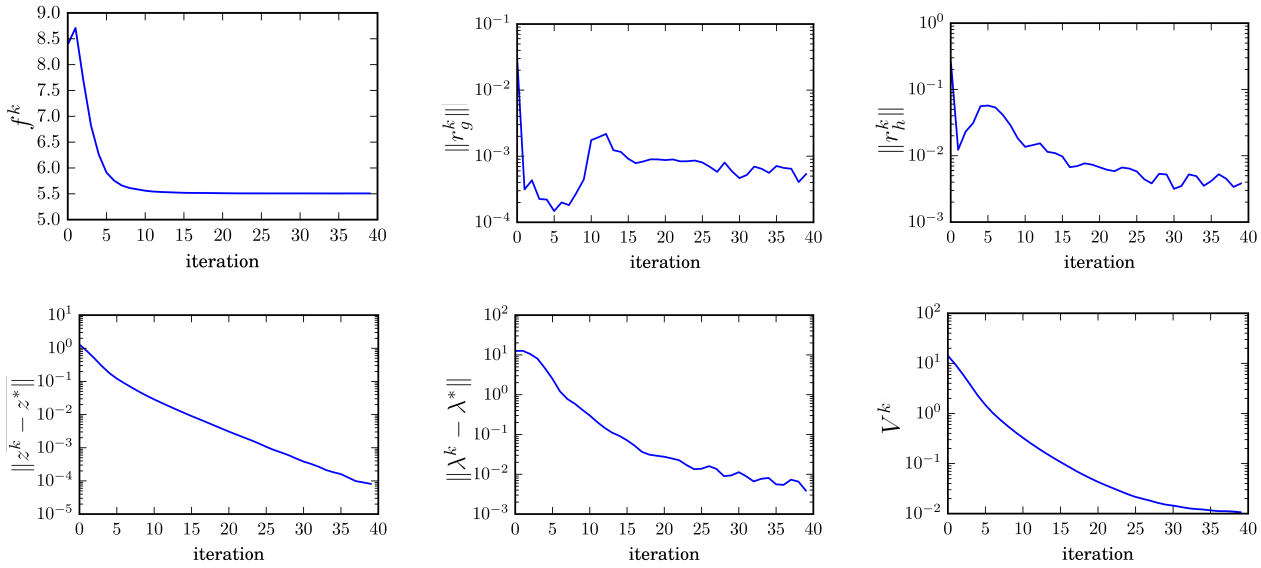

Figure 1: Various statistics for the performance of the distributed ADMM extension on an instance of the robust SVM problem. The convergence proof only states that the value V^k must be monotonically decreasing. This can be observed also experimentally in the figure on the bottom right. Neither the primal function value nor the residuals need to be monotonically decreasing, and as can be seen in the figures on the top, they actually do not decrease monotonically.

$x^{(i)}$ is Σ_i. The robust SVM problem is then the following SOCP (second order cone program)

$$\begin{aligned} &\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } \ y^{(i)}w^\top \bar{x}^{(i)} \geq 1 - \xi_i + \sqrt{\delta_i/(1-\delta_i)}\left\|\Sigma_i^{1/2}w\right\| \\ &\qquad \xi_i \geq 0, \quad i = 1 \ldots n. \end{aligned}$$

This problem can be reformulated into the form of Problem 6 and is thus amenable to a distributed implementation of our extension of ADMM.

Experimental Setup

We generated random data sets similarly to [Andersen *et al.*, 2012], where an interior point solver has been described for solving the robust SVM problem. The set of feature vectors was sampled from a uniform distribution on $[-1, 1]^n$. The covariance matrices Σ_i were randomly chosen from the cone of positive semidefinite matrices with entries in the interval $[-1, 1]$ and δ_i has been set to $\frac{1}{2}$. Each data point contributes exactly one constraint to the problem and is assigned to only one of the compute nodes.

In the following, the primal optimization variables are w and ξ, the consensus variables for the primal optimization variables w are still denoted as z, and also the dual variables are still denoted as λ for the consensus constraints and μ for the convex constraints, respectively.

Convergence Results

Figure 1 shows the primal objective function value f^k, the norm of the residuals r_g^k and r_h^k, the distances $\|z^k - z^*\|$, $\|\lambda^k - \lambda^*\|$, and the value V^k of one run of our algorithm for two compute nodes. Note, that only V^k must be strictly monotonically decreasing according to our convergence analysis. The proof does not make any statement about the monotonicity of the other values, and as can be seen in Figure 1, such statements would actually not be true. All values decrease in the long run, but are not necessarily monotonically decreasing.

As can be seen in Figure 1 (top-left), the function value f^k is actually increasing for the first few iterations, while the residuals r_g^k for the inequality constraints become very small, see Figure 1 (top-middle). That is, within the first iterations each compute node finds a solution to its share of the data that is almost feasible but has a higher function value than the true optimal solution. This basically means that the errors ξ_i for the data points are over-estimated. After a few more iterations the primal function value drops and the inequality residuals increase meaning that the error terms ξ_i as well as the individual estimators w_i converge to their optimal values.

In the long run, the local estimators at the different compute nodes converge to the same solution. This is witnessed in Figure 1 (top-right), where one can see that the residuals r_h^k for the consensus constraints converge to zero, i.e., consensus among the compute nodes is reached in the long run.

Finally, it can be seen that the consensus estimator z^k converges to its unique optimal point z^*. Note that in general we cannot guarantee such a convergence since the optimal point does not need to be unique. In the special case that the optimal point is unique we always have convergence to this point.

Scalability Results

Figure 2 shows the scalability of our extension of ADMM in terms of the number of compute nodes, data points, and approximation quality, respectively. All running times were

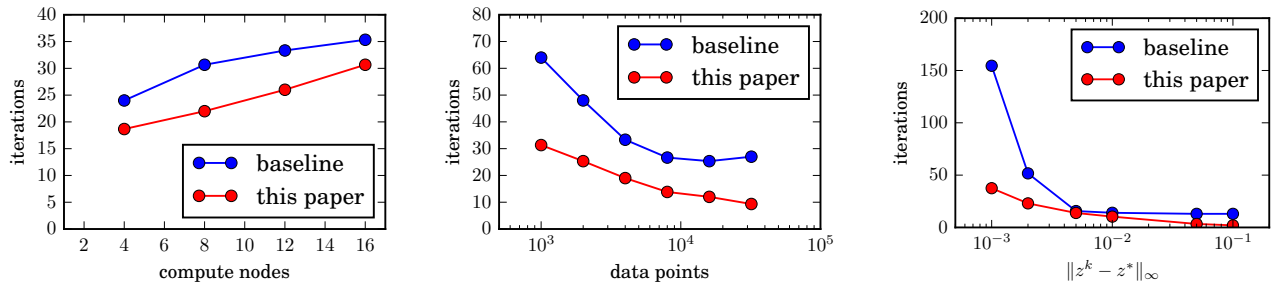

Figure 2: Running times of the algorithm on the robust SVM problem. The figure on the left shows that the number of iterations increases mildly with the number of compute nodes. The middle picture shows that the number of iterations is decreasing with increasing number of data points. The figure on the right shows the dependency of the distance of the consensus estimator z^k in iteration k to the optimal estimator z^*. It can be seen that our extension of ADMM outperforms the baseline approach with three nested loops.

measured in terms of iterations and averaged over runs for ten randomly generated data sets. For the baseline we used the straightforward three nested loops approach.

(1) For measuring the scalability in terms of employed compute nodes, we generated 10,000 data points with 10,000 features. As stopping criterion we used $\|z^k - z^*\|_\infty \leq 5 \cdot 10^{-3}$, i.e., the predictor z^k had to be close to the optimum. Here we use the infinity norm to be independent from the number of dimensions. The data set was split into four, eight, twelve, and 16 equal sized batches that were distributed among the compute nodes. Note that every batch had much fewer data points than features, and thus the optimal solutions to the respective problems at the compute nodes were quite different from each other. Nevertheless, our algorithm converged very well to the globally optimal solution. Only the convergence speed was affected by the diversity of the local solutions at the different compute nodes. Since we kept the total number of data points in our experiments fixed, the diversity was increasing with the number of compute nodes that were assigned fewer data points each. Hence it was expected that the convergence speed decreases, i.e., the number of iterations increases, with a growing number of compute nodes. The expected behavior can be seen in Figure 2 (left).

(2) For measuring the scalability in terms of the number of data points we increased the number of data points but kept the number of features fixed at 200. The stopping criterion for our algorithm was again $\|z^k - z^*\|_\infty \leq 5 \cdot 10^{-3}$. We used eight compute nodes to compute the solutions. Again, the points were distributed equally among the compute nodes. This time one would expect a decreasing running time with an increasing number of data points, because the number of data points per machine is increasing and thus also the diversity of the local solutions at the different compute nodes is decreasing. That is, with an increasing number of data points it should take fewer iterations to reach an approximate consensus about the global solution among the compute nodes. The results of the experiment that are shown in Figure 2 (middle) confirm this expectation. The number of iterations indeed decreases with a growing number of data points. This is similar to [Shalev-Shwartz and Srebro, 2008] who have also observed that an increasing number of data points can decrease the work required for a good predictor.

(3) For measuring the scalability in terms of the approximation quality, we generated 8000 data points in 200 dimensions. Again, eight compute nodes were used for the experiments whose results are shown in Figure 2 (right). As expected the number of iterations (running time) increases with increasing approximation quality that was again measured in terms of the infinity norm. In this paper we are not providing a theoretical convergence rate analysis, which we leave for future work, but the experimental results shown here already provide some intuition on the dependency of the number of iterations in terms of the approximation quality: It seems that our extension of ADMM can solve problems to a medium accuracy within a reasonable number of iterations, but higher accuracy requires a significant increase in the number of iterations. Such a behavior is well known for standard ADMM without constraints. In the context of our example application, robust SVMs, medium accuracy usually is sufficient as often higher accuracy solutions do not provide better predictors, a phenomenon that is also known as regularization by early stopping.

7 Conclusions

Despite the vast literature on ADMM, to the best of our knowledge, no scheme for distributing general convex constraints has been studied before. Here we have closed this gap by combining ADMM and the augmented Lagrangian method for solving general convex optimization problems with many convex constraints. The straightforward combination of ADMM and the augmented Lagrangian method entails three nested loops, an outer loop for reaching consensus, one loop for the constraints, and an inner loop for solving unconstrained problems. Our main contribution is showing that the loops for reaching consensus and for handling constraints can be merged, resulting in a scheme with only two nested loops. We provide the first convergence proof for such a lazy algorithmic scheme.

Acknowledgments

This work has been supported by the DFG grant GI-711/5-1 within the Priority Program 1736 Algorithms for Big Data.

References

[Andersen *et al.*, 2012] Martin Andersen, Joachim Dahl, Zhang Liu, and Lieven Vandenberghe. *Interior-Point Methods for Large-Scale Cone Programming*, pages 55–84. MIT Press, 2012.

[Boyd *et al.*, 2011] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[Gabay and Mercier, 1976] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17 – 40, 1976.

[Ghadimi *et al.*, 2015] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems. *IEEE Trans. Automat. Contr.*, 60(3):644–658, 2015.

[Giesen and Laue, 2016] Joachim Giesen and Sören Laue. Distributed convex optimization with many convex constraints. *CoRR*, abs/1610.02967, 2016.

[Glowinski and Marroco, 1975] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[Hestenes, 1969] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[Huang and Sidiropoulos, 2016] Kejun Huang and Nicholas D Sidiropoulos. Consensus-admm for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing*, 64(20):5297–5310, 2016.

[Morales and Nocedal, 2011] José Luis Morales and Jorge Nocedal. Remark on "algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 38(1):7:1–7:4, 2011.

[Mosk-Aoyama *et al.*, 2010] Damon Mosk-Aoyama, Tim Roughgarden, and Devavrat Shah. Fully distributed algorithms for convex optimization problems. *SIAM Journal on Optimization*, 20(6):3260–3279, 2010.

[Powell, 1969] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14(1):224–248, 1969.

[Shalev-Shwartz and Srebro, 2008] Shai Shalev-Shwartz and Nathan Srebro. SVM optimization: inverse dependence on training set size. In *International Conference on Machine Learning (ICML)*, pages 928–935, 2008.

[Shivaswamy *et al.*, 2006] Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, and Alexander J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

[Slater, 1950] Morton Slater. Lagrange multipliers revisited. Cowles Foundation Discussion Papers 80, Cowles Foundation for Research in Economics, Yale University, 1950.

[Tsang *et al.*, 2007] Ivor W. Tsang, András Kocsor, and James T. Kwok. Simpler core vector machines with enclosing balls. In *International Conference on Machine Learning (ICML)*, pages 911–918, 2007.

[Zhang and Kwok, 2014] Ruiliang Zhang and James T Kwok. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning (ICML)*, pages 1701–1709, 2014.

[Zhu and Martínez, 2012] Minghui Zhu and Sonia Martínez. On distributed convex optimization under inequality and equality constraints. *IEEE Trans. Automat. Contr.*, 57(1):151–164, 2012.

[Zhu *et al.*, 1997] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 1997.