Fast ADMM Algorithm for Distributed Optimization with Adaptive Penalty

Changkyu Song, Sejong Yoon and **Vladimir Pavlovic**

Rutgers, The State University of New Jersey
110 Frelinghuysen Road
Piscataway, NJ 08854-8019
{cs1080, sjyoon, vladimir}@cs.rutgers.edu

Abstract

We propose new methods to speed up convergence of the Alternating Direction Method of Multipliers (ADMM), a common optimization tool in the context of large scale and distributed learning. The proposed method accelerates the speed of convergence by automatically deciding the constraint penalty needed for parameter consensus in each iteration. In addition, we also propose an extension of the method that adaptively determines the maximum number of iterations to update the penalty. We show that this approach effectively leads to an adaptive, dynamic network topology underlying the distributed optimization. The utility of the new penalty update schemes is demonstrated on both synthetic and real data, including an instance of the probabilistic matrix factorization task known as the structure-from-motion problem.

Introduction

The need for algorithms and methods that can handle large data in a distributed setting has grown significantly in recent years. Specifically, such settings may arise in two prototypical scenarios: (a) induced distributed data: distribute and parallelize computationally demanding optimization tasks to connected computational nodes using a data distributed model and (b) intrinsically distributed data: data is collected across a connected network of sensors (e.g., mobile devices, camera networks), where some or all of the computation can be performed in individual sensor nodes without requiring centralized data pooling. Several distributed learning approaches have been proposed to meet these needs. In particular, the alternating direction method of multiplier (ADMM) (Boyd et al. 2010) is an optimization technique that has been very often used in computer vision and machine learning to handle model estimation and learning in either of the two large data settings (Liu et al. 2012; Zhuang et al. 2012; Elhamifar et al. 2013; Zeng et al. 2013; Wang et al. 2014; Lai et al. 2014; Boussaid and Kokkinos 2014; Miksik et al. 2014).

In the distributed optimization setting, the distributed nodes process data locally by solving small optimization

problems and aggregate the result by exchanging the (possibly compressed) local solutions (e.g., local model parameter estimates) to arrive at a consensus global result. However, the nature of distributed learning models, particularly in the fully distributed setting where no network topology is presumed, inherently requires repetitive communications between the device nodes. Therefore, it is desirable to reduce the amount of information exchanged and simultaneously improve computational efficiency through faster convergence of such distributed algorithms. Our methods can be applied to any arbitrary distributed settings as well as parallel computation that requires a certain centralized connection (e.g. a star topology).

To this end, the contributions of this paper are three folds: (a) We propose two variants of ADMM for the consensus-based distributed learning faster than the standard ADMM. Our method extends an acceleration approach for ADMM (He, Yang, and Wang 2000) by an efficient variable penalty parameter update strategy. This strategy results in improved convergence properties of ADMM and also works in a fully distributed fashion. (b) We extend our proposed method to automatically determine the maximum number of iterations allocated to successive updates by employing a budget management scheme. This strategy results in adaptive parameter tuning for ADMM, removing the need for arbitrary parameter settings, and effectively induces a varying network communication topology. (c) We apply the proposed method to a prototypical vision and learning problem, the distributed PPCA for structure-from-motion, and demonstrate its empirical utility over the traditional ADMM.

Problem Description and Related Works

The problem we consider in this paper can be formulated as a consensus-based optimization problem (Bertsekas and Tsitsiklis 1989). A general consensus-based optimization problem can be written as

$$\arg\min_{\theta_i} \quad \sum_{i=1}^{J} f_i(\theta_i), \quad s.t. \quad \theta_i = \theta_j, \forall i \neq j \quad (1)$$

where we want to find the set of optimal parameters $\theta_i, i = 1..J$ that minimizes the sum of convex objective functions $f_i(\theta_i)$, where J denotes the total number of the functions.
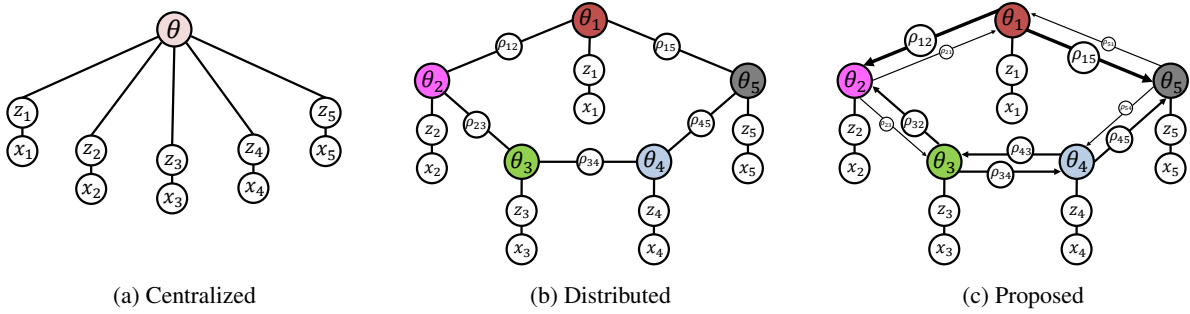
Figure 1: Centralized, distributed, and the proposed learning model in a ring network. The bigger size of ρ_{ij} means that corresponding constraint is more penalized.

This problem is typically a reformulation of a centralized optimization task $\arg\min f(\theta)$ with a decomposable objective $f(\theta) = \sum_{i=1}^{J} f_i(\theta)$. Given the consensus formulation, the original problem can be solved by decomposing the problem into J subproblems so that J processors can cooperate to solve the overall problem by changing the equality constraint to $\theta_i = \bar{\theta}$ where $\bar{\theta}$ denotes a globally shared parameter. The optimization can be approached efficiently by exploiting the alternating direction method of multiplier (ADMM) (Boyd et al. 2010).

The above consensus formulation is particularly suitable for many optimization problems that appear in computer vision. For instance, since $f_i(\theta_i)$ can be any convex function, we can also consider a probabilistic model with the joint negative log likelihood $f_i(\theta_i) = -\log p(x_i, z_i | \theta_i)$ between the observation x_i and the corresponding latent variable z_i. Assuming (x_i, z_i) are independent and identically distributed, finding the maximum likelihood estimate of the shared parameter $\bar{\theta}$ can then be formulated as the optimization problem we described above for many exponential family parametric densities. Moreover, the function need not be a likelihood, but can also be a typical decomposable and regularized loss that occurs in many vision problems such as denoising or dictionary learning.

It is often very convenient to consider the above consensus optimization problem from the perspective of optimization on graphs. For instance, the centralized i.i.d. Maximum Likelihood learning can be viewed as the optimization on the graph in Fig. 1a. Edges in this graph depict functional (in)dependencies among variables, commonly found in representations such as Markov Random Fields (Miksik et al. 2014) or Factor Graphs (Bishop 2006). In this context, to fully decompose $f(\cdot)$ and eliminate the need for a processing center completely, one can introduce auxiliary variables ρ_{ij} on every edge to break the dependency between θ_i and θ_j (Forero, Cano, and Giannakis 2011; Yoon and Pavlovic 2012) as shown in Fig. 1b. This generalizes to arbitrary graphs, where the connectivity structure may be implied by node placement or communication constraints (camera networks), imaging constraints (pixel neighborhoods in images or frames in a video sequence), or other contextual constraints (loss and regularization structure).

In general, given a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the nodes $i, j \in \mathcal{V}$ and the edges $e_{ij} = (i, j) \in \mathcal{E}$, the consensus optimization problem becomes

$$\min \sum_{i \in \mathcal{V}} f_i(\theta_i), \quad s.t. \quad \theta_i = \rho_{ij}, \rho_{ij} = \theta_j, j \in \mathcal{B}_i. \quad (2)$$

Solving that problem is equivalent to optimizing the augmented Lagrangian $\mathcal{L}(\boldsymbol{\Theta}) = \sum_{i \in \mathcal{V}} \mathcal{L}_i(\boldsymbol{\Theta}_i)$,

$$\mathcal{L}_i(\boldsymbol{\Theta}_i) = f_i(\theta_i) + \sum_{j \in \mathcal{B}_i} \left\{ \lambda_{ij1}^\top (\theta_i - \rho_{ij}) + \lambda_{ij2}^\top (\rho_{ij} - \theta_j) \right\}$$
$$+ \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \|\theta_i - \rho_{ij}\|^2 + \|\rho_{ij} - \theta_j\|^2 \right\}, \quad (3)$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_i : i \in \mathcal{V}\}$, $\boldsymbol{\Theta}_i = \{\theta_i, \rho_i, \lambda_i\}$ are parameters to find, $\lambda_i = \{\lambda_{ij1}, \lambda_{ij2} : j \in \mathcal{B}_i\}$, $\lambda_{ij1}, \lambda_{ij2}$ are Lagrange multipliers, $\mathcal{B}_i = \{j | e_{ij} \in \mathcal{E}\}$ is the set of one hop neighbors of node i, $\eta > 0$ is a fixed scalar penalty constraint, and $\| \cdot \|$ is induced norm. The ADMM approach suggests that the optimization can be done in coordinate descent fashion taking gradient of each variable while fixing all the others.

Convergence Speed of ADMM

The currently known convergence rate of ADMM is $O(1/T)$ where T is the number of iterations (He and Yuan 2012). Even though $O(1/T)$ is the best known bound, it has been observed empirically that ADMM converges faster in many applications. Moreover, the computation time per each iteration may dominate the total algorithm running time. Thus many speed up techniques for ADMM have been proposed that are application specific. One way is to come up with a predictor-corrector step for the coordinate descent (Goldstein et al. 2014) using some available acceleration method such as the accelerated gradient method (Nesterov 1983). It guarantees quadratic convergence for strongly convex $f_i(\cdot)$. Another way is to replace the gradient descent optimization with a stochastic one (Ouyang et al. 2013; Suzuki 2013). This approach has recently gained attention as it greatly reduces the computation per iteration. However, these methods usually require the coordinating center node thus may not readily applicable to the decentralized setting. Moreover, we want to preserve the application range of ADMM and avoid introducing additional assumptions on $f_i(\cdot)$.

One way to improve convergence speed of ADMM is through the use of different constraint penalty in each iteration. For example, a variant of ADMM with self-adaptive penalty (He, Yang, and Wang 2000) improved the convergence speed as well as made its performance less dependent on initial penalty values. The idea is to change the constraint penalty taking account of the relative magnitudes of *primal* and *dual* residuals of ADMM as follows

$$\eta^{t+1} = \begin{cases} \eta^t \cdot (1 + \tau^t) & \text{, if } \|r^t\|_2 > \mu \|s^t\|_2 \\ \eta^t \cdot (1 + \tau^t)^{-1} & \text{, if } \|s^t\|_2 > \mu \|r^t\|_2 \\ \eta^t & \text{, otherwise} \end{cases} \quad (4)$$

where t is the iteration index, $\mu > 1$, $\tau^t > 0$ are parameters, r^t and s^t are the primal and dual residuals, respectively (Please refer extended version of this paper (Song, Yoon, and Pavlovic 2015) for their definitions). The primal residual measures the violation of the consensus constraints and the dual residual measures the progress of the optimization in the dual space. This update converges when τ^t satisfies $\sum_{t=0}^{\infty} \tau^t < \infty$, i.e. we stop updating η^t after a finite number of iterations. Typical choice for parameters are suggested as $\mu = 10$ and $\tau^t = 1$ at all t iterations. The strength of this approach is that conservative changes in the penalty are guaranteed to converge (Rockafellar 1976; Boyd et al. 2010). However, like other ADMM speed up approaches mentioned above, this update scheme relies on the global computation of the primal and the dual residuals and requires the η^t stored in nodes to be homogeneous over entire network thus it is not a fully decentralized scheme. Moreover, the choice of parameters as well as the maximum number of iterations require manually tuning.

Proposed Methods

We present our proposed ADMM penalty update schemes in three steps. First, we extend the aforementioned update scheme of (4) to be applicable on fully decentralized setting. Next, we propose the novel penalty parameter update strategy for ADMM speed up that does not require manual tuning of τ^t. Finally, we extend the strategy so that we can automatically select the maximum number of penalty update iterations.

ADMM with Varying Penalty (ADMM-VP)

Throughout the paper, the superscript t in all terms with subscript i denote either the objective function or parameter at t-th iteration for node i. In order to extend (4) for a fully distributed setting, we first introduce η_i^t, the penalty for i-th node at t-th iteration. Next, we need to compute local primal and dual residuals for each node i. In the fully distributed learning framework of (Forero, Cano, and Giannakis 2011; Yoon and Pavlovic 2012), the dual auxiliary variable vanishes from derivation. However, to compute the residuals, we need to keep track of the dual variable, which is essentially the average of local estimates, explicitly over iterations. The squared residual norms for the i-th node are defined as

$$\|r_i^t\|_2^2 = \|\theta_i^t - \bar{\theta}_i^t\|_2^2, \quad \|s_i^t\|_2^2 = (\eta_i^t)^2 \|\bar{\theta}_i^t - \bar{\theta}_i^{t-1}\|_2^2, \quad (5)$$

where $\bar{\theta}_i^t = (1/|\mathcal{B}_i|) \sum_{j \in \mathcal{B}_i} \theta_j^t$. Note the difference from the standard residual definitions for consensus ADMM (Boyd et al. 2010), used in (4), where the dual variable is considered as a single, globally accessible variable, $\bar{\theta}^t$ instead of local $\bar{\theta}_i^t$. This allows each node to change its η_i^t based on its own local residuals. The penalty update scheme is similar to (4) but η^t, $\|r^t\|_2$ and $\|s^t\|_2$ are replaced with η_i^t, $\|r_i^t\|_2$ and $\|s_i^t\|_2$, respectively. Lastly, the original adaptive penalty ADMM (He, Yang, and Wang 2000) stopped changing η^t after $t > 50$. However, in ADMM-VP, if we stop the same way, we end up with heterogeneously fixed penalty values which impacts the convergence of ADMM by yielding heavy oscillations near the saddle point. Therefore we reset all penalty values in all nodes to a pre-defined value (e.g. η^0, the initial penalty parameter) after a fixed number of iterations. As we fix the penalty values homogeneously after a finite number of iterations, it becomes the standard ADMM after that point thus the convergence of ADMM-VP update is guaranteed.

ADMM with Adaptive Penalty (ADMM-AP)

We further extend η_i by introducing a bi-directional graph with a penalty constraint parameter η_{ij} specific to directed edge e_{ij} from node i to j. The modified augmented Lagrangian $\mathcal{L}_i$ is similar to (3) except that we replace η with η_{ij}. The penalty constraint controls the amount each constraint contributes to the local minimization problem. The penalty constraint parameter η_{ij} is determined by evaluating the parameter θ_j from node j with the objective function $f_i(\cdot)$ of node i as

$$\eta_{ij}^{t+1} = \begin{cases} \eta^0 \cdot (1 + \tau_{ij}^t) & \text{, if } t < t^{max} \\ \eta^0 & \text{, otherwise} \end{cases} \quad (6)$$

where t^{max} is the maximum number of iterations for the update as proposed in (He, Yang, and Wang 2000) and

$$\tau_{ij}^t = \frac{\kappa_i^t(\theta_i^t)}{\kappa_i^t(\theta_j^t)} - 1, \quad \kappa_i^t(\theta) = \left(\frac{f_i^t(\theta) - f_i^{min}}{f_i^{max} - f_i^{min}} + 1 \right), \quad (7)$$

$$f_i^{max} = \max\{f_i^t(\theta_i^t), f_i^t(\theta_j^t) : j \in \mathcal{B}_i\},$$
$$f_i^{min} = \min\{f_i^t(\theta_i^t), f_i^t(\theta_j^t) : j \in \mathcal{B}_i\}. \quad (8)$$

The interpretation of this update strategy is straightforward. In each iteration t, each i-th node will evaluate its objective using its own estimate of θ_i^t and the estimates from other nodes θ_j^t (we use ρ_{ij}^t instead of actual θ_j^t to retain locality of each node from the neighbors). Then, we assign more weight to the neighbor with better parameter estimate for the local $f_i(\cdot)$ (i.e. larger penalty η_{ij}^t if $f_i(\theta_j) < f_i(\theta_i)$) with the above update scheme. The intuition behind the ADMM-AP update is to emphasize the local optimization during early stages and then deal with the consensus update at later, subsequence stages. If all local parameters yield similarly valued local objectives $f_i(\cdot)$, the onus is placed on consensus. This makes ADMM-AP different from pre-initialization that does the local optimization using the local observations and ignores the consensus constraints.

Note that unlike the update strategy of (4), we do not need to specify τ^t and the update weight is automatically chosen

according to the normalized difference in the local objective evaluation among neighboring parameters. The proposed algorithm also emphasizes the objective minimization over the minimization that solely depends on the norms of primal and dual residuals of constraints. The hope is that we not only achieve the consensus of the parameters of the model but also a *good* estimate with respect to the objective.

On the other hand, the convergence property of (He, Yang, and Wang 2000) still holds for the proposed algorithm. Following Remark 4.2 of (He, Yang, and Wang 2000), the requirement for the convergence is to satisfy the update ratio to be fixed after some $t^{\max} < \infty$ iterations.

Moreover, the proposed update ensures bounding by $\eta_{ij}^{t+1}/\eta_{ij}^{t} \in [0.5, 2]$, which matches with the increase and decrease amount suggested in the literature (He, Yang, and Wang 2000; Boyd et al. 2010). One may use $t^{\max} = 50$ as in (He, Yang, and Wang 2000).

ADMM with Network Adaptive Penalty (ADMM-NAP)

To extend the proposed method for automatically deciding the maximum number of penalty updates, the penalty update for the ADMM becomes

$$
\eta_{ij}^{t+1} = \begin{cases} \eta^0 \cdot (1 + \tau_{ij}^t) & \text{, if } \sum_{u=1}^{t} |\tau_{ij}^u| < \mathcal{T}_{ij}^t \\ \eta^0 & \text{, otherwise.} \end{cases} \quad (9)
$$

Fig. 1c depicts how the proposed model have different structures from centralized and traditional distributed models, and how nodes share their parameters via network.

In addition to the adaptive penalty update, the inequality condition on the summation of $\tau_{ij}^u, u = 1..t$ encodes the spent *budget* that the edge e_{ij} can change η_{ij}. All nodes have its upper bound $\mathcal{T}_{ij}^t$ and everytime it makes a change to η_{ij}, it has to *pay* exactly the amount they changed. If the edge has changed too much, too often, the update strategy will block the edge from changing η_{ij} any more.

The update scheme is guaranteed to convergence if $\mathcal{T}_{ij}^t$ is simply set to constant $\mathcal{T}$ for all i, j, t or if $\tau_{ij}^t = 0$ for $t > t^{max}$. However, with a different objective function and different network connectivity, a different upper bound should be imposed. This is because a given upper bound $\mathcal{T}$ or maximum iteration t^{max} could be too small for a certain node to fully take an advantage of our adaptation strategy or they could be too big so that it converges much slowly because of the continuously changing η_{ij}^t. To this end, we propose updating strategy for $\mathcal{T}_{ij}^t$ as following:

$$
\mathcal{T}_{ij}^{t+1} = \begin{cases} \mathcal{T}_{ij}^t + \alpha^n \mathcal{T} & \text{, if } \sum_{u=1}^{t} |\tau_{ij}^u| \geq \mathcal{T}_{ij}^t \text{ and} \\ & \quad |f_i(\theta_i^t) - f_i(\theta_i^{t-1})| > \beta \\ \mathcal{T}_{ij}^t & \text{, otherwise} \end{cases} \quad (10)
$$

where $\mathcal{T}_{ij}^0$ is set by an initial parameter $\mathcal{T}$ and $\alpha, \beta \in (0, 1)$ are parameters. Whenever $\mathcal{T}_{ij}^{t+1} > \mathcal{T}_{ij}^t$, we increase n by 1. Once $\sum_{u=1}^{t} |\tau_{ij}^u| \geq \mathcal{T}_{ij}^t$ but its objective value is still significantly changing, i.e. $|f_i(\theta_i^t) - f_i(\theta_i^{t-1})| > \beta$, $\mathcal{T}_{ij}^{t+1}$ is increased by $\alpha^n \mathcal{T}$. Note that the independent upper bound

$\mathcal{T}_{ij}^t$ for each η_{ij}^t update on the edge e_{ij} makes it sensitive to the various network topology, but it still satisfies the convergence condition because $\lim_{t \to \infty} \mathcal{T}_{ij}^t \leq \sum_{n=1}^{\infty} \alpha^{n-1} \mathcal{T} = \frac{1}{1-\alpha} \mathcal{T}$.

Combined Update Strategies (ADMM-VP + AP, ADMM-VP + NAP)

Observing (4) and the proposed update schemes (6) and (9), one can easily come up with a combined update strategy by replacing τ^t in (4) with τ_{ij}^t. Based on preliminary experiments, we found that this replacement yields little utility. Instead, we suggest another penalty update strategy combining ADMM-VP and ADMM-AP as

$$
\eta_{ij}^{t+1} = \begin{cases} \eta_{ij}^t \cdot (1 + \tau_{ij}^t) \cdot 2 & \text{, if } \|r_i^t\|_2 > \mu \|s_i^t\|_2 \\ \eta_{ij}^t \cdot (1 + \tau_{ij}^t) \cdot (1/2) & \text{, if } \|s_i^t\|_2 > \mu \|r_i^t\|_2 \\ \eta_{ij}^t & \text{, otherwise} \end{cases} \quad (11)
$$

which we denote as ADMM-VP + AP. We reset $\eta_{ij}^t = \eta^0$ when $t > t^{\max}$. In order to combine ADMM-VP and ADMM-NAP, we consider the summation condition of τ_{ij}^t as in (9). We denote this strategy as ADMM-VP + NAP.

The Choice of the Proposed Methods

The key difference between ADMM-AP and ADMM-NAP is that the latter does not require $t_{\max}$ to be decided in advance. If the best $t_{\max}$ is known for a certain application, there is no significant benefit of ADMM-NAP over ADMM-AP. However, in many real world problems, $t_{\max}$ is not known and ADMM-NAP can be an effective option as demonstrated in Fig. 3c.

Distributed Maximum Likelihood Learning

In this section, we show how our method can be applied to an existing distributed learning framework in the context of distributed probabilistic principal component analysis (D-PPCA). D-PPCA can be viewed as fundamental approach to a general matrix factorization task in the presence of potentially missing data, with many applications in machine learning.

Probabilistic Principal Component Analysis

The Probabilistic PCA (PPCA) (Tipping and Bishop 1999) has many applications in vision problems, including structure from motion, dictionary learning, image inpainting, etc. We here restrict our attention to the linear PPCA without any loss of generality. The centralized PPCA is formulated as the task of projecting the source data $\mathbf{x}$ according to $\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$ where $\mathbf{x} \in \mathbb{R}^D$ is the observation column vector, $\mathbf{z} \in \mathbb{R}^M$ is the latent variable following $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{W} \in \mathbb{R}^{D \times M}$ is the projection matrix that maps $\mathbf{x}$ to $\mathbf{z}$, $\boldsymbol{\mu} \in \mathbb{R}^D$ allows non-zero mean, and the Gaussian observation noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, a^{-1}\mathbf{I})$ with the noise precision a. When $a^{-1} = 0$, PPCA recovers the standard PCA. The posterior estimate of the latent variable $\mathbf{z}$ given the observation $\mathbf{x}$ is $p(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^{\top}(\mathbf{x} - \boldsymbol{\mu}), a^{-1}\mathbf{M}^{-1})$, where $\mathbf{M} = \mathbf{W}^{\top}\mathbf{W} + a^{-1}\mathbf{I}$. The parameters $\mathbf{W}$, $\boldsymbol{\mu}$, and a can be estimated using a number of methods, including SVD and Expectation Maximization (EM) algorithm.

Distributed PPCA

The distributed extension of PPCA (D-PPCA) (Yoon and Pavlovic 2012) can be derived by applying ADMM to the centralized PPCA model above. Each node learns its local copy of PPCA parameters with its set of local observations $\mathbf{X}_i = \{\mathbf{x}_{in}|n = 1..N_i\}$ where $\mathbf{x}_{in}$ denotes the n-th observation in i-th node and N_i is the number of observations available in the node. Then, they exchange the parameters using the Lagrange multipliers and impose consensus constraints on the parameters. The global constrained optimization is

$$\min_{\boldsymbol{\Theta}_i} -\log p(\mathbf{X}_i|\boldsymbol{\Theta}_i) \quad s.t. \quad \boldsymbol{\Theta}_i = \rho_{ij}^{\boldsymbol{\Theta}}, \rho_{ij}^{\boldsymbol{\Theta}} = \boldsymbol{\Theta}_j, \quad (12)$$

where $\boldsymbol{\Theta}_i = \{\mathbf{W}_i, \boldsymbol{\mu}_i, a_i\}$ is the set of local parameters and $\rho_{ij}^{\boldsymbol{\Theta}} = \{\rho_{ij}^{\mathbf{W}}, \rho_{ij}^{\boldsymbol{\mu}}, \rho_{ij}^a\}$ is the set of auxiliary variables for the parameters. For the details regarding how the decentralized model is optimized, see (Yoon and Pavlovic 2012).

D-PPCA with Network Adaptive Penalty

The augmented Lagrangian applying the proposed ADMM with Network Adaptive Penalty is similar to (Yoon and Pavlovic 2012) except that η becomes η_{ij}. with λ_i, γ_i, β_i are Lagrange multipliers for the PPCA parameters for node i. The adaptive penalty constraint η_{ij}^t controls the speed of parameter propagation dynamically so that the overall optimization empirically converges faster than (Yoon and Pavlovic 2012). One can solve this optimization using the distributed EM approach (Forero, Cano, and Giannakis 2011). The E-step of the D-PPCA is the same as centralized counterpart (Tipping and Bishop 1999). The M-step is similar to (Yoon and Pavlovic 2012) except we use separate η_{ij} for each edge. The update formulas for the three parameters are similar and an example update for $\boldsymbol{\mu}_i$ can be found in (Song, Yoon, and Pavlovic 2015). Once all the parameters and the Lagrange multipliers are updated, we update η_{ij} and $\mathcal{T}_{ij}$ using (9) and (10), respectively. The overall algorithmic steps for the D-PPCA with Network Adaptive Penalty is summarized in (Song, Yoon, and Pavlovic 2015).

Experiments

We first analyze and compare the proposed methods (ADMM-VP, ADMM-AP, ADMM-NAP, ADMM-VP + AP, ADMM-VP + NAP) with the baseline method using synthetic data. Next, we apply our method to a distributed structure from motion problem using two benchmark real world datasets. For the baseline, we compare with the ADMM-based D-PPCA (Yoon and Pavlovic 2012) denoted as ADMM with fixed penalty $\eta^t = \eta^0$. Unless noted otherwise, we used $\eta^0 = 10$. To assess convergence, we compare the relative change of (12) to a fixed threshold (10^{-3} in this case) for the D-PPCA experiments as in (Yoon and Pavlovic 2012).

Synthetic Data

We generated 500 samples of 20 dimensional observations from a 5-dim subspace following $\mathcal{N}(\mathbf{0}, \mathbf{I})$, with the Gaussian measurement noise following $\mathcal{N}(\mathbf{0}, 0.2 \cdot \mathbf{I})$. For the distributed settings, the samples are assigned to each node evenly. All experiments are ran with 20 independent random initializations. We measured the number of iterations to convergence and the maximum subspace angle error versus the ground truth defined as the maximum of subspace angles between each node's projection matrix and the ground truth projection matrix. We examined the impact of different graph topologies and different graph sizes. We tested three network topologies: complete, ring and cluster (a connected graph consists of two complete graphs linked with an edge). For the graph size, we tested on 12, 16 and 20 nodes settings.

Top three plots in Fig. 2 depict results over varying number of nodes while fixing the graph topology as the complete graph. We plot the median result out of the 20 independent initializations. We observed that the speed up with the proposed method, particularly for ADMM-VP and its variants, becomes more significant as the number of nodes increases. This suggests the proposed method can be of particular use as the size of an application problem increases. Fig. 2c to Fig. 2e in the figure show the performance in the context of different network topologies. Our proposed methods converge faster or at the same rate as the standard ADMM. In some cases, either the standard ADMM or our methods could converge to a local optima, e.g. some of our methods in Fig. 2c prematurely converged, however, they still have very good performance that is less than 2 degree of subspace angle. The proposed method works most robustly in the complete graph setting. In other words as the graph connectivity increases, the convergence property of the proposed method improves. Note also that ADMM-VP works best in complete graph while ADMM-AP / NAP are better than the ADMM-VP in weakly connected networks (e.g. a ring which exhibits the sparsest connectivity resulting in long (error) propagation effects and, subsequently, much variable behavior). This makes sense as ADMM-VP depends on residual computation and the proposed local residual computation become less accurate compared to the complete graph when the global residual can be computed.

Distributed Affine Structure from Motion

We tested the performance of our method on five objects of Caltech Turntable (Moreels and Perona 2007) and Hopkins 155 (Tron and Vidal 2007) dataset as in (Yoon and Pavlovic 2012). The goal here is to jointly estimate the 3D structure of the objects as well as the camera motion, however in a distributed camera network setting. The input measurement matrix is defined as $2 \times F$ by N where F denotes the number of frames and N denotes the number of points. By applying PCA, we can decompose the input into the camera pose $\mathbf{W}_i$ and the 3D structure $\mathbb{E}[\mathbf{z}_{in}], n = 1..N_i$. For the detailed experimental setting, refer to (Tron and Vidal 2011; Yoon and Pavlovic 2012). As the performance measure, we used the maximum subspace angle error versus the centralized SVD-reconstructed structure. The network setting assumes five cameras on a complete graph.

Fig. 3 shows the result on the Caltech Turntable dataset. First, we compare Fig. 3a and Fig. 3b. One can see that when the graph is less connected (Fig. 3a), the proposed adaptive penalty method can boost ADMM-VP which cannot utilize the full residual information of fully connected case (Fig. 3b), as explained in synthetic data experiments. Next, we compare Fig. 3b and Fig. 3c. The network topolo-
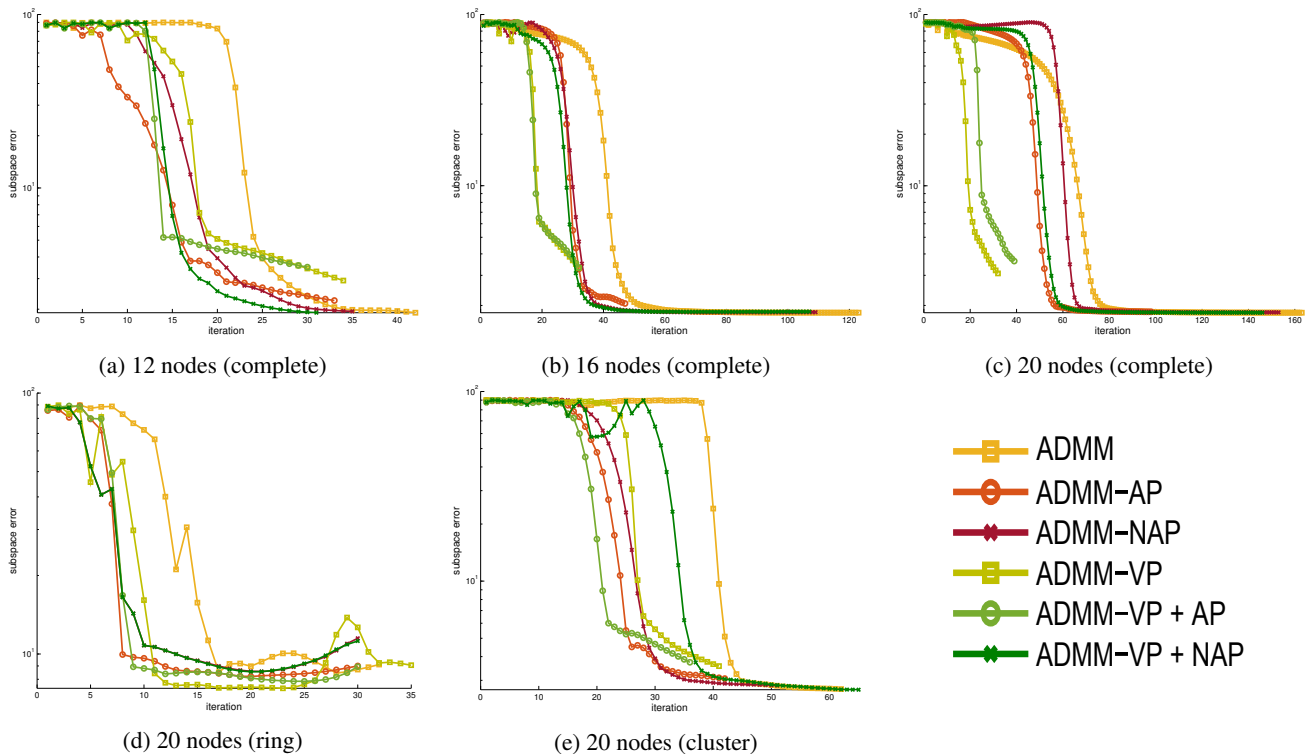
Figure 2: The comparison of proposed methods and the baseline ADMM using the subspace angle error of the projection matrix with (a-c) different graph size and (c-e) different network topology.

gies are the same (complete) but $t^{\max}$ value required for ADMM-VP, ADMM-AP, ADMM-VP + AP is different in these two groups of experiments. When $t^{max} = 50$ (Fig. 3b), all methods can accelerate throughout the iterations. However, when $t^{max} = 5$ (Fig. 3c), the methods that depend on t^{max} cannot accelerate after 5 iterations thus showing behavior similar to the baseline ADMM. On the other hand, ADMM-NAP based methods can accelerate by adaptively modifying the maximum number of penalty updates. Note that one can choose any small value of $\mathcal{T}$ and $\mathcal{T}_{ij}$ is increased automatically using (10).

For the Hopkins 155 dataset, we compared methods on 135 objects using the same approach as (Yoon and Pavlovic 2012). For each method considered, we computed the mean number of iterations until convergence. Since some objects in the dataset are point trajectories of non-rigid structure, it is inevitable for simple linear models to fail for those objects. Thus we omitted objects yielded more than 15 degrees when calculating the mean. For each object, we tested 5 independent random initializations. For ADMM-AP, ADMM-NAP and ADMM-VP + NAP, we found no significant speed up over the baseline ADMM. For ADMM-VP and ADMM-VP + AP, we could obtain 40.2%, 37.3% speed up, respectively if we use complete network. In ring network, the amount of improvement becomes smaller. This small or no improvement of speed is mainly due to the fact that the baseline ADMM converges fast enough (typically < 100 iterations)

thus there is little room for the proposed methods to speed up the optimization. As observed from the synthetic experiments and Caltech dataset, the acceleration of the proposed methods occurs at the earlier iterations of the optimization. Thus if one can come up with a better convergence checking criterion depending on the application, the proposed methods can be a very viable choice due to its parameter-free nature.

Conclusion

We introduced a novel adaptive penalty update methods for ADMM that can be applied to consensus distributed learning frameworks. Contrary to previous approaches, our adaptive penalty update methods, ADMM-AP and ADMM-NAP does not depend on the parameters that require manual tuning. Using both synthetic and real data experiments, we showed the empirical effectiveness of the methods over the baseline. In addition, we found that the performance of ADMM-VP decreases with weakly connected graphs, and in those cases, ADMM-AP and ADMM-NAP can be useful.

The proposed methods do leave some room for improvements. For the problems when the standard ADMM can converge fast enough, the proposed methods may show less than significant gains. A better convergence criterion may help stop the proposed algorithms at earlier iterations (e.g. a criterion that can stop algorithms to remove long tails in Fig. 2b or Fig. 2c).

(a) $t^{max} = 50$ (ring)   (b) $t^{max} = 50$ (complete)   (c) $t^{max} = 5$ (complete)
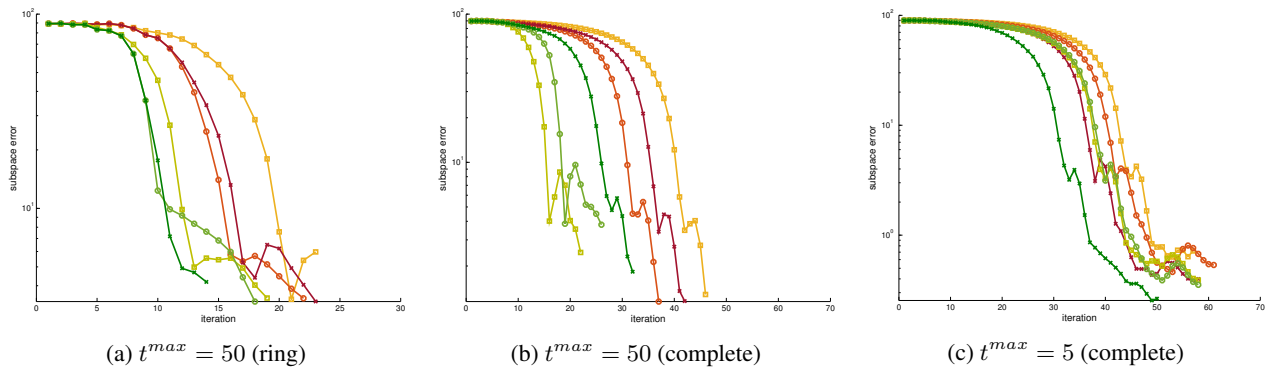
Figure 3: The comparison of proposed methods and the baseline ADMM using the subspace angle error of the reconstructed 3D structure with one object in Caltech dataset (Standing). See Fig. 2 for the plot labels. More results on the other four objects can be found in (Song, Yoon, and Pavlovic 2015).

References

Bertsekas, D. P., and Tsitsiklis, J. N. 1989. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. Springer.

Boussaid, H., and Kokkinos, I. 2014. Fast and Exact: ADMM-Based Discriminative Shape Segmentation with Loopy Part Models. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2010. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* 3(1):1–122.

Elhamifar, E.; Sapiro, G.; Yang, A. Y.; and Sastry, S. S. 2013. A Convex Optimization Framework for Active Learning. In *IEEE International Conference on Computer Vision, (ICCV)*, 209–216.

Forero, P. A.; Cano, A.; and Giannakis, G. B. 2011. Distributed Clustering Using Wireless Sensor Networks. *IEEE Journal of Selected Topics in Signal Processing* 5(4).

Goldstein, T.; O'Donoghue, B.; Setzer, S.; and Baraniuk, R. 2014. Fast Alternating Direction Optimization Methods. *SIAM Journal of Imaging Science* 7(3):1588–1623.

He, B., and Yuan, X. 2012. On the $O(1/n)$ Convergence Rate of the Douglas-Rachford Alternating Direction Method. *SIAM Journal of Numerical Analysis* 50(2):700–709.

He, B.; Yang, H.; and Wang, S. 2000. Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities. *Journal of Optimization Theory and Applications* 106(2):337–356.

Lai, K.-T.; Liu, D.; Chen, M.-S.; and Chang, S.-F. 2014. Recognizing complex events in videos by learning key static-dynamic evidences. In *Proceedings of the European Conference on Computer Vision (ECCV), 2014*, volume 8691 of *Lecture Notes in Computer Science*. 675–688.

Liu, R.; Lin, Z.; De la Torre, F.; and Su, Z. 2012. Fixed-rank representation for unsupervised visual learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 598–605.

Miksik, O.; Vineet, V.; Pérez, P.; and Torr, P. H. S. 2014. Distributed Non-Convex ADMM-inference in Large-scale Random Fields. In *British Machine Vision Conference (BMVC)*.

Moreels, P., and Perona, P. 2007. Evaluation of Features Detectors and Descriptors based on 3D Objects. *International Journal of Computer Vision* 73(3):263–284.

Nesterov, Y. 1983. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Math. Dokl.* 27:372–376.

Ouyang, H.; He, N.; Tran, L.; and Gray, A. 2013. Stochastic alternating direction method of multipliers. In *In Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Rockafellar, R. T. 1976. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization* 14:877.

Song, C.; Yoon, S.; and Pavlovic, V. 2015. Fast ADMM Algorithm for Distributed Optimization with Adaptive Penalty. *CoRR* http://arxiv.org/abs/1506.08928.

Suzuki, T. 2013. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Tipping, M. E., and Bishop, C. M. 1999. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society, Series B* 61:611–622.

Tron, R., and Vidal, R. 2007. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Tron, R., and Vidal, R. 2011. Distributed Computer Vision Algorithms Through Distributed Averaging. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 57–63.

Wang, C.; Wang, Y.; Lin, Z.; Yuille, A. L.; and Gao, W. 2014. Robust estimation of 3d human poses from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2369–2376.

Yoon, S., and Pavlovic, V. 2012. Distributed probabilistic learning for camera networks with missing data. In *Advances in Neural Information Processing Systems (NIPS)*.

Zeng, Z.; Xiao, S.; Jia, K.; Chan, T.; Gao, S.; Xu, D.; and Ma, Y. 2013. Learning by associating ambiguously labeled images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 708–715.

Zhuang, L.; Gao, H.; Lin, Z.; Ma, Y.; Zhang, X.; and Yu, N. 2012. Non-negative low rank and sparse graph for semi-supervised learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*.