A Penalty Alternating Direction Method of Multipliers for Decentralized Composite Optimization

Jiaojiao Zhang

Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong

Joint work with
Anthony Man-Cho So (CUHK)
Qing Ling (SYSU)

ICASSP 2020

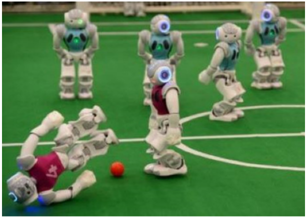
Outline

Outline

1. Background
2. Penalized Approximation Formulation
3. Algorithm Development
4. Convergence and Rate of Convergence
5. Numerical Experiments
6. Conclusions

Background

Distributed Network





Background

- Consider the composite optimization problem:

$$\hat{x}^* = \arg\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} \big(f_i(x) + g_i(x)\big)$$

- Such a finite sum problem is common in machine learning. For example, f_i represents loss function and g_i represents regularization.

Background

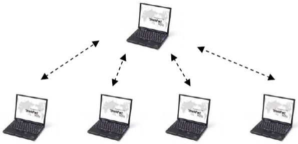


Fig 1. Centralized network

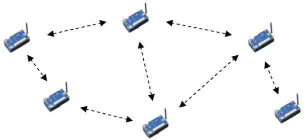


Fig 2. Decentralized network

- Centralized network: master collects/broadcasts x.
- Decentralized network: equivalent formulation

$$\{\hat{x}_i^*\} = \arg\min_{x_i \in \mathbb{R}^p} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i) \right),$$
$$\text{s.t. } x_1 = \cdots = x_n.$$

Each node holds its own x_i and only communicates with its neighbors.

Related Work

- Primal domain
  - Decentralized Gradient Descent (DGD) [Yuan 2016]
  - Second-order methods for the penalized approximation [Mokhtari 2016]
- Dual domain
  - Alternating Direction Method of Multipliers (ADMM) for smooth problem with linear rate [Shi 2014]
  - Proximal decentralized linearized ADMM with ergodic rate $O(1/k)$ [Aybat 2017]
  - The augmented Lagrangian method (ALM) for smooth problem wih linear rate [Jakovetic 2014]
- Other methods
  - PG-EXTRA and NIDS for nonsmooth problem with rate $O(1/k)$ and $o(1/k)$ [Shi 2016, Li 2019]
  - gradient tracking [Lorenzo 2016, Scutari 2019]
  - PGA [Alghunaim 2019] and SONATA [Sun 2019] with linear rate under the assumption that nonsmooth term is common

Outline

Penalized Approximation Formulation

Aggregated variables and functions:

- stack all x_i into matrix $\mathbf{x}$

$$\mathbf{x} \triangleq \begin{pmatrix} - & x_1^T & - \\ & \vdots & \\ - & x_n^T & - \end{pmatrix} \in \mathbb{R}^{n \times p}$$

  ,

- define $f(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$ and $g(\mathbf{x}) = \sum_{i=1}^n g_i(x_i)$.

Consensus constraint:

- (Assumption 1) Introduce the mixing matrix W with elements $w_{ij} \geq 0$. $w_{ij} = 0$ if and only if $j \notin \mathcal{N}_i \cup \{i\}$. Further, $W^T = W$, $W\mathbf{1}_{n\times 1} = \mathbf{1}_{n\times 1}$ and $null(I - W) = span(\mathbf{1}_{n\times 1})$.
- $x_1 = \cdots = x_n$ is equivalent to $(I - W)\mathbf{x} = 0$. Since $I - W \succeq 0$, $(I - W)^{\frac{1}{2}}\mathbf{x} = 0$.

Penalized Approximation Formulation

Consider the penalized approximation:

$$\hat{x}^* = \arg\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} \left(f_i(x) + g_i(x)\right)$$

Equivalent

$$\hat{\mathbf{x}}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \ f(\mathbf{x}) + g(\mathbf{x}) \quad \text{s.t. } (I - W)^{\frac{1}{2}}\mathbf{x} = 0$$

Penalized Approximation Formulation

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \ f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|(I - W)^{\frac{1}{2}}\mathbf{x}\|_{\mathcal{F}}^2$$

- $\epsilon > 0$ is penalty parameter.
- smaller ϵ brings higher accuracy, i.e. $\mathbf{x}^*$ is close to $\hat{\mathbf{x}}^*$.

DGD to Solve Penalized Problem

- When $g(\mathbf{x}) = 0$, penalized problem can be solved by GD:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma\big(\nabla f(\mathbf{x}^k) + \frac{1}{\epsilon}(I - W)\mathbf{x}^k\big),$$

  where $\gamma > 0$ is the step size.
- GD is not efficient, why?
  - When ϵ is very small, the Lipschitz constant is in the order of $O(\frac{1}{\epsilon})$. To converge, γ must be in the order of $O(\epsilon)$.
  - Setting $\gamma = \epsilon$ recovers the DGD update $\mathbf{x}^{k+1} = W\mathbf{x}^k - \epsilon\nabla f(\mathbf{x}^k)$.
  - Proximal DGD faces with the same problem.
- To tackle the unfavorable accuracy-speed tradeoff, we use ADMM-based method.

Outline

Introduce an auxiliary variable **z**:

$$\hat{x}^* = \arg\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} \left(f_i(x) + g_i(x)\right)$$

$\xleftrightarrow{\textit{Equivalent}}$

$$\hat{\mathbf{x}}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \; f(\mathbf{x}) + g(\mathbf{x})$$
$$\text{s.t. } (I - W)^{\frac{1}{2}}\mathbf{x} = 0$$

$\Big\downarrow$ ***Penalized Approximation Formulation***

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \; f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|(I - W)^{\frac{1}{2}}\mathbf{x}\|_{\mathcal{F}}^2$$

$\Big\updownarrow$ ***Equivalent***

$$(\mathbf{x}^*, \mathbf{z}^*) = \arg\min_{\mathbf{x}, \mathbf{z} \in \mathbb{R}^{n \times p}} \; f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|\mathbf{z}\|_{\mathcal{F}}^2$$
$$\text{s.t. } (I - W)^{\frac{1}{2}}\mathbf{x} = \mathbf{z}$$

Algorithm Development

- The augmented Lagrangian function is $L_\alpha(\mathbf{x}, \mathbf{z}, \Pi) = f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|\mathbf{z}\|_{\mathcal{F}}^2 + \left\langle \Pi, (I - W)^{\frac{1}{2}}\mathbf{x} - \mathbf{z} \right\rangle + \frac{\alpha}{2}\|(I - W)^{\frac{1}{2}}\mathbf{x} - \mathbf{z}\|_{\mathcal{F}}^2$.
  - $\Pi \in \mathbb{R}^{n \times p}$ is the Lagrange multiplier (also called dual variable).
  - $\alpha > 0$ is a parameter.
- Traditional ADMM iterates as

$$
\begin{aligned}
\mathbf{x}^{k+1} &= \arg\min_{x} L_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k), \\
\mathbf{z}^{k+1} &= \arg\min_{z} L_\alpha(\mathbf{x}^{k+1}, \mathbf{z}, \Pi^k) \\
&= \frac{1}{\alpha + \frac{1}{\epsilon}}\left[\Pi^k + \alpha(I - W)^{\frac{1}{2}}\mathbf{x}^{k+1}\right], \\
\Pi^{k+1} &= \Pi^k + \alpha\left[(I - W)^{\frac{1}{2}}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\right].
\end{aligned}
$$

- $\mathbf{x}^{k+1}$ does not have a closed form solution $\to$ *linearization*

Algorithm Development

- Separate the augmented Lagrangian function as

$$L_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k) \triangleq \underbrace{g(\mathbf{x})}_{\text{nonsmooth}} + \underbrace{\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)}_{\text{smooth}}.$$

- Replace the smooth part by a quadratic approximation

$$L_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k) \approx \underbrace{g(\mathbf{x})}_{\text{nonsmooth}}$$
$$+ \underbrace{\tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k) + \left\langle \nabla_{\mathbf{x}} \tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k), \mathbf{x} - \mathbf{x}^k \right\rangle + \frac{1}{2c} \|\mathbf{x} - \mathbf{x}^k\|_{\mathcal{F}}^2}_{\text{quadratic approximation}}.$$

- The primal update is the proximal mapping of g defined as $\text{prox}_{cg}(y) \triangleq \text{argmin}_x \{g(x) + \frac{1}{2c}\|x - y\|^2\}$, where $c > 0$ is a scalar.

Algorithm Development

- The proposed penalty ADMM (PAD) updates as

$$
\begin{aligned}
\mathbf{x}^{k+1} &= \mathrm{prox}_{cg}\left(\mathbf{x}^k - c\left[\nabla f(\mathbf{x}^k) + \alpha(I-W)^{\frac{1}{2}}((I-W)^{\frac{1}{2}}\mathbf{x}^k - \mathbf{z}^k + \frac{\Pi^k}{\alpha})\right]\right),\\
\mathbf{z}^{k+1} &= \frac{1}{\alpha + \frac{1}{\epsilon}}\left[\Pi^k + \alpha(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1}\right],\\
\Pi^{k+1} &= \Pi^k + \alpha\left[(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\right].
\end{aligned}
$$

- Simplify updates by introducing $\bar{\Pi}^k = (I-W)^{\frac{1}{2}}\Pi^k$ and $\bar{\mathbf{z}}^k = (I-W)^{\frac{1}{2}}\mathbf{z}^k$.

Algorithm 1 PAD run by agent i

Require: Choose the parameters ϵ, α and c. Initialize the local variables to x_i^0, $\bar{z}_i{}^0$ and $\bar{\pi}_i^0$.

1: **for** $k = 1, 2, \cdots$ **do**

2: Update local variable x_i^{k+1} by

$$x_i^{k+1} = \mathrm{prox}_{cg_i}\left(x_i^k - c\left[\nabla f_i(x_i^k) + \alpha(x_i^k - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^k - \bar{z}_i^k) + \bar{\pi}_i^k\right]\right).$$

3: Transmit x_i^{k+1}/ receive x_j^{k+1} from neighbors $j \in \mathcal{N}_i$.

4: Update local auxiliary variable $\bar{z}_i^{k+1}$ by

$$\bar{z}_i^{k+1} = \frac{1}{\alpha + \frac{1}{\epsilon}}\left[\bar{\pi}_i^k + \alpha(x_i^{k+1} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{k+1})\right].$$

5: Update local dual variable $\bar{\pi}_i^{k+1}$ by

$$\bar{\pi}_i^{k+1} = \bar{\pi}_i^k + \alpha\left[(x_i^{k+1} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{k+1}) - \bar{z}_i^{k+1}\right].$$

6: **end for**

Outline

Convergence and Rate of Convergence

For convergence, assume

- (Assumption 2) Each $g_i(x)$ is convex and nonsmooth. The proximal mapping of $g_i(x)$ can be computed easily.
- (Assumption 3) Each $f_i(x)$ is convex and differentiable with a Lipschitz continuous gradient such that

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_f \|x - y\|, \ \forall x, y \in \mathbb{R}^p,$$

where $L_f > 0$ is the Lipschitz constant.

For linear convergence rate, further assume

- (Assumption 4) Each $f_i(x)$ is strongly convex such that

$$\left\langle x - y, \nabla f_i(x) - \nabla f_i(y) \right\rangle \geq \mu_f \|x - y\|^2, \ \forall x, y \in \mathbb{R}^p,$$

where $\mu_f > 0$ is the strong convexity constant.

Convergence and Rate of Convergence

Theorem 1 (Convergence)

Under Assumptions 1, 2 and 3, if the parameters α and c are chosen such that $c < \frac{1}{L_f + \alpha\lambda_{max}(I-W)}$, then PAD from any initial points converges to the optimal solution $\mathbf{x}^*$ of the penalized approximation problem.

Theorem 2 (Convergence rate)

Under Assumption 1-4, if the parameters α and c are chosen such that $c < \frac{1}{\frac{L_f^2}{\mu_f} + \alpha\lambda_{max}(I-W)}$, then PAD from any initial points converges linearly to the optimal solution $\mathbf{x}^*$ of the penalized approximation problem.

- Since $L_f/\mu_f > 1$, the bound of Theorem 2 is smaller.
- Step sizes do not rely on ϵ. A sufficiently small ϵ can be used so that the penalized approximation error is negligible.

Outline

Numerical Experiments

1 Decentralized Logistic Regression

- $\min_x \frac{1}{n}\sum_{i=1}^n \left\{ \sum_{j=1}^{m_i} \ln\left(1+\exp(-(M_{(i)j}x)y_{(i)j})\right)\right\}$.
- $n = 30$, connectivity ratio $= 0.5$, $p = 10$ and $m_i = 5, \forall i$.
- Relative error $\|\mathbf{x}^k - \hat{\mathbf{x}}^*\|_{\mathcal{F}}/\|\mathbf{x}^0 - \hat{\mathbf{x}}^*\|_{\mathcal{F}}$.

Decentralized Logistic Regression

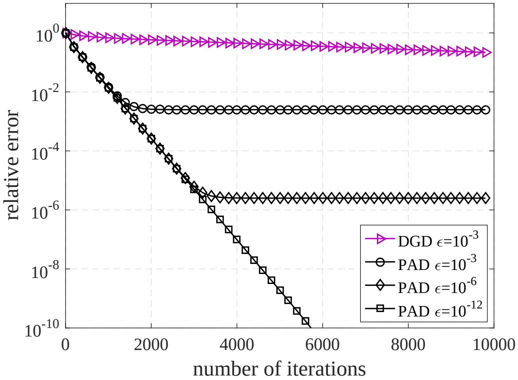


Fig 3. Relative error of DGD (with $\epsilon = 10^{-3}$) and PAD (with $\epsilon = 10^{-3}$, $\epsilon = 10^{-6}$ and $\epsilon = 10^{-12}$, respectively). For DGD, its step size is $\gamma = \epsilon$. For PAD, $\alpha = 0.6$ and $c = 0.032$ according to the condition in Theorem 1.

2 Decentralized Quadratic Programming

- $\min_x \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} x^T Q_i x + h_i^T x,\ s.t. a_i^T x \leq b_i, \forall i.$
- $g_i(x) = \begin{cases} 0 & \text{if } a_i^T x \leq b_i \\ +\infty & \text{otherwise} \end{cases}$
- $n = 10$, connectivity ratio $= 0.4$ and $p = 50$.
- Relative error $\|\mathbf{x}^k - \hat{\mathbf{x}}^*\|_{\mathcal{F}} / \|\mathbf{x}^0 - \hat{\mathbf{x}}^*\|_{\mathcal{F}}$.

Decentralized Quadratic Programming

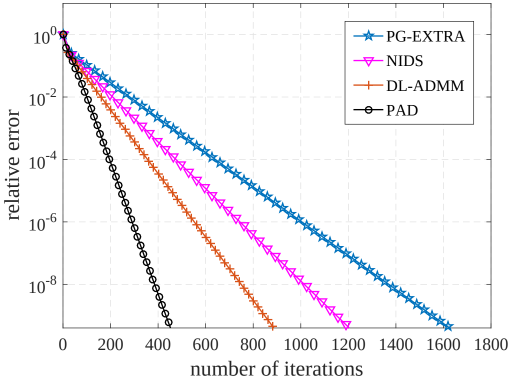


Fig 4. Relative error of PG-EXTRA, NIDS, DL-ADMM and PAD. For PAD, $\epsilon = 10^{-12}$, $\alpha = 1.2$ and $c = 0.2$ as suggested in Theorem 2. For PG-EXTRA, $c = 2\lambda_{\min}((I + W)/2)/L_f$. For NIDS, $c = 1.9/L_f$. The parameters of DL-ADMM are hand-optimized.

3 Application in Classification for Breast Cancer Data

- $\min_x \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\ln\left(1+\exp\left(-\left(M_{(i)j}x\right)y_{(i)j}\right)\right)+\frac{1}{n}\sum_{i=1}^{n}\lambda_i\|x\|_1.$
- $n=50$, connectivity ratio $=0.5$, $p=10$ and $\lambda_i=\frac{0.1}{n}$. Each node i holds $m_i=10$ training samples.
- Percentage of correct predictions.

Application in Classification for Breast Cancer Data

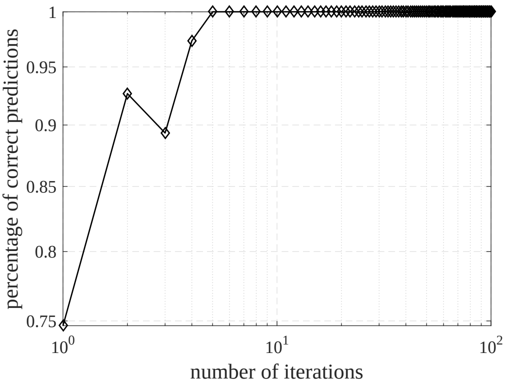


Fig 5. Percentage of correct predictions of PAD. We set $\epsilon = 10^{-12}$, $\alpha = 0.2$ and $c = 0.9$ by hand-optimization.

Outline

Conclusions

- Consider a penalized approximation of the decentralized composite problem. The penalty parameter can be very small.
- By linearization, PAD has low computational costs.
- PAD is provably convergent under convexity, and linearly convergent under strong convexity.

Thank you !