2019 年 8 月 计 算 数 学 第 41 卷第 3 期
Aug., 2019 MATHEMATICA NUMERICA SINICA Vol.41, No.3

青年评述

无中心优化的算子分裂方法*[1)]

印卧涛
(阿里巴巴 (美国) 达摩院)

摘 要

在某些多智能体系统中, 由于受到通讯等因素的限制, 单个智能体只能进行本地计算, 再与相邻智能体交换数据. 与传统的并行和分布式计算不同, 这种数据交换方式不再使用中心节点或者共享内存, 而仅限于相邻节点之间. 这种通过局部数据交换而实现全网目标的方式叫做无中心计算. 比如, 从任意的多个数开始, 所有智能体通过不断地计算其局部平均, 就都能收敛到这些数的平均值. 无中心计算有不易形成通讯和计算瓶颈的优点, 更适合分布的节点, 因此受到一些应用的欢迎.

本文介绍求解一致最优化问题的若干无中心算法. 一致最优化问题的目标是全网所有节点的变量收敛到同一个、并使所有目标函数之和最小的值. 我们可以通过推广求平均的无中心方法去实现这个目标, 但是得到算法比普通 (有中心的) 优化算法收敛得更慢, 有阶数差距. 近年来, 一些新的无中心算法弥补了这个阶数差距. 本文采用算子分裂的统一框架, 以比这些算法原文更为简单的形式介绍这些方法.

关键词: 无中心算法, 一致优化, 算子分裂, 单调算子

MR (2010) 主题分类: 68Q85, 90C25, 93A14

1. 前 言

多智能体系统 (multi-agent system) 由多个智能体 (agents) 组成. 每个智能体拥有自己的传感器、存储器、计算单元. 它们不但有计算功能, 还可以与环境进行交互, 与相邻的其他智能体交换数据. 这样一个系统可以提供单个智能体不可能提供的功能. 常见的例子包括多玩家游戏、机器人群、无人机群、可以彼此通信的自动驾驶汽车、无线网络、智能电网, 当然还有人类社会. 此外, 因为模型规模太大或者数据量太大, 大型机器学习的模型训练, 尤其是深度学习的模型训练, 往往需要使用到多台计算机.

多智能体系统在博弈论、机器人技术、信息论、数据挖掘、机器学习、经济学、政治学等多个领域得到了广泛的研究. 虽然不同领域赋予多智能体系统不同的目标、约束、行为集, 但是大多数问题仍然可以表示为数学方程或者最优化模型.

最简单的多智能体问题是计算 n 个向量 $a_1,\dots,a_n\in\mathbb{R}^d$ 的平均值, 其中每个智能持有一个 a_i. 如果存在一个中心智能体, 能够收集所有其他智能体的数字, 那么再求一下平均就有答

* 2019 年 7 月 9 日收到.

1) 作者简介: 印卧涛, 哥伦比亚大学大学运筹学博士, 现任阿里巴巴 (美国) 达摩院机器智能技术研究员, 加州洛杉矶大学 (UCLA) 数学系终身教授 (on leave). 主要从事计算理论、优化算法、机器学习、信号处理等专业的创新理论和应用研究. 曾获得 2008 年 NSF CAREER Award, 2009 年 Alfred P. Sloan 研究奖, 2016 年晨兴应用数学金奖, 以及若干会议及年度最佳论文奖.

案了. 如果我们不允许这样一个中心存在, 而只允许相邻智能体交换数据, 这个看似简单的问题就变得有些挑战了.

我们用一个无向图 $G=(V,E)$ 刻画智能体之间的互联; 其中 $V=\{1,\dots,n\}$ 是节点集合, 每一个节点代表一个智能体 (下文将一致采用 "节点", 不再使用 "智能体"); E 是边的集合, 每条边是一个通信链接. 当没有中心节点去收集数据的情况下, 我们依然可以通过多次通讯迭代来获得平均值, 这个方法叫做 average consensus (平均一致) 法. 具体说, 节点 i 从 $x_i^0=a_i$ 开始, 产生迭代序列 $x_i^1,\dots,x_i^k,\dots$; 具体迭代形式为

$$x_i^{k+1}=w_{ii}x_i^k+\sum_{j\in N_i}w_{ij}x_j^k,\quad i=1,\dots,n. \tag{1.1}$$

这里 $N_i=\{j:(j,i)\in E\}$ 是节点 i 的相邻节点的集合. 实现 (1.1) 的过程中, 每个节点首先获取所有相邻节点 j 的 x_j^k, 然后按照 (1.1) 做加权求和, 其中 w_{ij} 是求和系数. 为了公式简洁, 我们引入

$$\mathbf{x}=\begin{bmatrix}— & x_1^T & — \\ & \vdots & \\ — & x_n^T & —\end{bmatrix}\in\mathbb{R}^{n\times d},$$

(对于多数算法的设计和分析而言, $d=1$ 和 $d\geq 2$ 之间没有本质区别, 所以我们不使用 X 而仅仅使用更简单的 $\mathbf{x}$) 且将迭代 (1.1) 写成矩阵形式

$$\mathbf{x}^k=W\mathbf{x}^{k-1}=\cdots=W^k\mathbf{x}^0.$$

(注意: k 是向量 $\mathbf{x}$ 的迭代次数, 也是矩阵 W 的幂次.) 我们希望 $W^k\mathbf{x}^0$ 的极限是所有节点的一致平均, 即

$$\lim_{k\to\infty}W^k\mathbf{x}^0=\begin{bmatrix}\vdots \\ \frac{1}{n}\sum_{i=1}^n\mathbf{x}_i^0 \\ \vdots\end{bmatrix}=\frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{x}^0.$$

我们把矩阵 $\frac{1}{n}\mathbf{1}\mathbf{1}^T$ 称作求平均算子.

为了实现无中心求平均这个目标, W 需要满足以下条件:

- 假设 $x_1=x_2=\cdots=x_n$ 已经一致了, 简写为 $\mathbf{x}=x_1\mathbf{1}$, 我们要求 $W\mathbf{x}=\mathbf{x}$, 即不破化一致性. 这就得到 $W\mathbf{1}=\mathbf{1}$. 因此, 我们要求 W 有特征值 1, 并对应特征向量 $\mathbf{1}$. 如果再要求 w_{ij} 为非负, 这样的矩阵是**右随机矩阵**.

- 我们也不希望 W 改变 n 个数的总和, 即对任意 $\mathbf{x}$, 要求满足 $\mathbf{1}^TW\mathbf{x}=\mathbf{1}^T\mathbf{x}$, 从而有 $\mathbf{1}=W^T\mathbf{1}$. 因此, 我们要求 W^T 也有特征值 1 和对应的特征向量 $\mathbf{1}$. (当然, W 和 W^T 是共享特征值的.) 这样的矩阵是**左随机矩阵**.

- 最后, 我们要求

$$\lim_k\|W^k-\frac{1}{n}\mathbf{1}\mathbf{1}^T\|=0. \tag{1.2}$$

即迭代的极限效果是一致平均. 由于 $W^k\mathbf{1} = W^{k-1}\mathbf{1} = \cdots = \mathbf{1}$, 我们有 $(W^k - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{1} = \mathbf{1} - \mathbf{1} = 0$, 因此

$$\begin{aligned}\|W^k - \frac{1}{n}\mathbf{1}\mathbf{1}^T\| &= \max\left\{\|(W^k - \frac{1}{n}\mathbf{1}\mathbf{1}^T)u\| : \|u\| = 1\right\}\\ &= \max\{\|(W^k - \frac{1}{n}\mathbf{1}\mathbf{1}^T)u\| : \|u\| = 1, \mathbf{1}^T u = 0\}\\ &= \max\{\|Wu\|^k : \|u\| = 1, \mathbf{1}^T u = 0\}.\end{aligned}$$

那么, 若要得到 (1.2), 我们首先要求 W 的特征值 1 的重数为 1, 并且要求其余的特征值属于 $(-1, 1)$, 即

$$1 = \lambda_1(W) > \lambda_2(W) \geq \cdots \geq \lambda_n(W) > -1. \tag{1.3}$$

总结一下, 在非负前提下, W 是随机矩阵 (既是左随机的又是右随机的) 并且满足条件 (1.3).

得到了 W 的谱性质后, 我们再来看看 W 的结构性质. 如果节点 j 无法直接向节点 i 传送它的 x_j, 根据无中心化要求, 节点 i 无法取得 x_j, 则 w_{ij} 必为 0. 因此, 通信网络的拓扑结构决定了 W 的稀疏结构.

如果通讯网络随着时间变化, 那么 W 也要随时间变化, 这时 $W(t)$ 需要满足的条件变得更加复杂. 但不难看出, (离散时间) 下的无中心平均法的核心是对矩阵 W 和其乘积的收敛性研究.

1.1. 一致最小问题

一致平均问题可以写成一个简单的优化模型:

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} \frac{1}{2}\|x_i - a_i\|^2, \quad \text{subject to } x_i = x_j, \ \ \forall \text{ 节点 } i, j \in V.$$

一个有广泛应用、更一般的问题是**一致最小问题**, 也叫做**共识优化问题**, 其基本模型是:

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} f_i(x_i), \quad \text{subject to } x_i = x_j, \ \ \forall \text{ 节点 } i, j \in V. \tag{1.4}$$

这个问题要求所有节点的变量均一致, 在此基础上最小化所有目标函数的和. 求解 (1.4) 必须既考虑一致性又考虑最小化. 如果不同的 f_i 不能被同一个点 x 最小化, 则一致性和最小化相互间存在矛盾.

虽然 (1.4) 与 $\text{minimize}_{\mathbf{x}} \sum_{i=1}^{n} f_i(x)$ 在数学上等价, 但是出于无中心化计算的要求, 我们必须使用类似于 $W\mathbf{x}^k$ 这样的分布式通讯和迭代. 一个较为直接的想法是把 (1.1) 与梯度下降结合, 得到算法:

$$\mathbf{x}^{k+1} = W\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k), \tag{1.5}$$

$$\text{即} \quad \begin{cases} y_i^k = w_{ii}x_i^k + \sum_{j\in N_i} w_{ij}x_j^k, \\ x_i^{k+1} = y_i^k - \alpha\nabla f_i(x_i^k), \end{cases} \qquad i = 1, \ldots, n.$$

一次迭代过程以通讯开始, 得到 y_i^k 后, 每个节点再通过本地梯度下降得到 x_i^{k+1}.

但是, (1.5) 并不能收敛到 (1.4) 的最优解. 事实上, 其极限 $\mathbf{x} = \lim_k \mathbf{x}^k$ 是不一致的, 不能保证 $x_1 = \cdots = x_n$. 这是因为假设有 $x_1 = \cdots = x_n$, 我们就有 $\mathbf{x} = W\mathbf{x}$, 再从 (1.5) 的极限

$\mathbf{x} = W\mathbf{x} - \alpha\nabla f(\mathbf{x})$, 我们得到 $\nabla f(\mathbf{x}) = 0$, 即所有的 ∇f_i 在同一点 x_1 取值为 0 (向量); 但是, (1.4) 的最优条件是在同一点, 所有 ∇f_i 之和为 0, 并不要求它们同时为 0; 进一步说, 能使得所有 ∇f_i 同时为 0 的点一般而言是不存在的.

类似但不同的迭代, $\mathbf{x}^{k+1} = W\left(\mathbf{x}^k - \alpha\nabla f(\mathbf{x})\right)$ (最早在 [1,2] 中出现), 也不能收敛到一致的解.

假设原问题的最优解是 $\mathbf{x}^*$; 令 $x^* = x_1^* = \cdots = x_n^*$. 我们用两个标准衡量解的质量:

1. 不一致程度: $\|\mathbf{x}\|_{I-W}$; $\mathbf{x}$ 满足 $x_1 = \cdots = x_n$ 当且仅当 $\|\mathbf{x}\|_{I-W} = 0$.
2. 全局目标误差: 给定 $\mathbf{x}$ (无论是否一致), 利用其平均 $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ 计算 $\sum_{i=1}^n f_i(\bar{x}) - \sum_{i=1}^n f_i(x^*)$ (当下文还有函数 g_i 的时候, 以 $f_i + g_i$ 代替 f_i).

回头看 (1.5). 假设所有梯度 ∇f_i 是 Lipschitz 的. 文献 [3] 证明了在固定步长 α 下, 迭代 (1.5) 收敛到原问题解的 $O(\alpha)$ 邻域内一个点, 即不一致程度是 $O(\alpha)$, 全局目标误差是 $O(\frac{\alpha}{1-\zeta})$; 这里

$$\zeta = \max\{|\lambda_2(W)|, |\lambda_n(W)|\} < 1,$$

$\lambda_i(W)$ 是矩阵 W 的第 i 大特征值.

为了解决 (1.5) 无法收敛到精确解的问题, [4,5] 使用了递减步长和梯度加速技巧. 其中, [4] 使用了 $\alpha_k = \frac{1}{\sqrt{k}}$ 并在有界梯度的条件下证明了全局目标误差的收敛率 $O(\ln k/k)$. 而 [5] 使用了内外双层循环, 其中内层循环会做多次 $W\mathbf{x}$ 的迭代以提高一致性; 如果不使用内层循环的话, 递减步长 $\alpha_k = \frac{1}{k^{1/3}}$ 同样给出了全局目标误差的收敛率 $O(\ln k/k)$. [6] 研究了一族递减步长 $O(1/k^\epsilon)$, $\epsilon \in (0,1]$, 得到了不一致程度的收敛速度 $O(1/k^\epsilon)$, 以及相应全局目标误差的收敛率

$$\begin{cases} O(\frac{\ln k}{\sqrt{k}}), & \text{if } \epsilon = 1/2, \\ O(\frac{1}{\ln k}), & \text{if } \epsilon = 1, \\ O\left(k^{-\min\{\epsilon, 1-\epsilon\}}\right), & \text{otherwise.} \end{cases}$$

由此可见, ϵ 接近 0 的时候, 步长递减慢, 导致一致性收敛慢; ϵ 接近 1 的时候, 一致性收敛速度接近 $O(1/k)$, 但是全局目标误差收敛接近非常慢的 $O(1/\ln k)$. 当 $\epsilon = 1/2$ 的时候, 一致性收敛和全局目标误差收敛的速度均是 $O(1/\sqrt{k})$.

算法 (1.5) 的收敛速度和精度都低于普通 (中心化) 梯度算法. 具体说, 在相同假设下, 后者没有一致性问题并有全局目标误差的速度 $O(1/k)$ 和经过 Nesterov 加速的 $O(1/k^2)$.

本文将介绍如何利用现代的最优化和算子理论, 系统地得到无中心一致优化问题的一阶算法, 并且获得更快的收敛速度.

2. 无中心优化的基本模型

无中心优化算法的设计起点是一些等价但形式不同的模型.

为了更一般化, 我们让每个节点 i 拥有 Lipschitz 可导的函数 f_i 和 proximable (有易于计算的邻近点算子, 即 proximal 算子) 的函数 g_i, 每个节点的目标函数是 $f_i + g_i$. 对应的一致最

小问题的**基本模型**是:

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i)\right), \quad \text{subject to } x_i = x_j, \ \ \forall \text{ 节点 } i, j \in V. \tag{2.1}$$

我们要求所有节点的变量均一致, 在此基础上最小化所有目标函数的和.

由于图是连通的, 所以我们可以把约束 $x_i = x_j$, $\forall$ 节点 $i, j \in V$, 简化为 $x_i = x_j$, $\forall$ 边 $e = (i, j) \in E$. 在此基础上, 对每边 e 引入额外变量 y_e, 则得到了 **ADMM 适配模型**:

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i)\right) \\ &\text{subject to } y_e = x_i, \ y_e = x_j, \ \forall \text{ 边 } e = (i, j). \end{aligned} \tag{2.2}$$

通过定义**合适的矩阵** G **和** H, 我们可以把 (2.2) 写为

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i)\right) + 0(\mathbf{y}) \\ &\text{subject to } G\mathbf{x} + H\mathbf{y} = 0. \end{aligned} \tag{2.3}$$

对这个模型可以运用 ADMM 或者 function-linearized ADMM 法.

引入信息混合矩阵 W, 我们得到**等价约束模型**

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i)\right), \quad \text{subject to } (I - W)\mathbf{x} = 0. \tag{2.4}$$

这里 I 是单位矩阵. W 满足第一节中描述的谱性质和结构性质. 模型 (2.4) 的优势将在后面的讨论中凸显, 它能够给出目前最快的一类无中心化方法.

最后一个是**无约束惩罚模型**:

$$\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} \left(f_i(x_i) + g_i(x_i)\right) + \frac{1}{2\rho}\|\mathbf{x}\|_{I-W}^2. \tag{2.5}$$

这里 $\|\mathbf{x}\|_{I-W}^2$ 是一个 semi-norm, 其定义是

$$\|\mathbf{x}\|_{I-W}^2 = \mathbf{x}^T(I - W)\mathbf{x} \geq 0.$$

模型 (2.5) 比较简单, 但不精确. 一方面, 由于没有约束, 我们可以使用最简单的梯度法. 另一方面, 对约束 $(I - W)\mathbf{x} = 0$ 的松弛使得其最优解并非原问题 (2.1) 的最优解. 这里, 我们用参数 ρ 调节松弛的力度.

在之后的讨论中, 为了公式简洁, 我们令

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{n} f_i(x_i), \\ g(\mathbf{x}) &= \sum_{i=1}^{n} g_i(x_i). \end{aligned}$$

拥有 f, g 这种结构的函数叫做完全可分函数.

3. 算子分裂法介绍

这一节里, 我们介绍基本的算子分裂法和收敛结果. 本文中所有的无中心化方法均能快速地使用算子分裂法得到, 并且得到收敛保证.

在介绍一般算子的时候, 为了一般性, 我们让 x, y, z, u, v 来代表 $\mathbb{R}^n$ 中的点, 而不使用 $\mathbf{x}, \mathbf{y}$ 等我们已经特殊定义的量.

3.1. 三步流程

先介绍算子分裂法的三步流程. 第一步, 我们将一个问题写成等价的算子等式或者属于式:

$$0 \in A(x), \tag{3.1a}$$

$$\text{或} \quad 0 \in A(x) + B(x), \tag{3.1b}$$

$$\text{或} \quad 0 \in A(x) + B(x) + C(x).$$

这里, A, B, C 不是矩阵而是抽象的算子, 每个算子把 $\mathbb{R}^n$ 中的点 x 映射到 $\mathbb{R}^n$ 中的点或者集合. "+" 是 Minkowski 加法. 如果在某点 x, $A(x)$ 或 $B(x)$ 或 $C(x)$ 为 $\emptyset$, 则和为 $\emptyset$. 考虑集合映射而不限制为单射的目的是适用非光滑函数的次梯度以及约束等式的 normal cone. 比如, 前述一致优化模型中的凸函数 g_i 不可导, 所以 ∂g_i 是集合函数. (一个凸函数 f 在点 x 处, 对所有点 y 都满足 $f(y) \geq f(x) + \langle u, y - x\rangle$ 的向量 u 叫做偏导数; 凸函数 f 在点 x 处如果可导, 则 u 为其倒数 $\nabla f(x)$; 如果不可导, 偏导数 u 不唯一, 它们的集合写作 $\partial f(x)$.)

为了简单, 本文全篇忽略算子的定义域, 即在 $\mathbb{R}^n$ 上定义所有算子. 为了避免误解, 我们强调问题本身可以带有约束. 约束作为 δ 函数融入 g_i.

从一个问题得到 (3.1) 并不难, 通常写出优化问题的一阶最优条件就可以. 但是, 由于凸非光滑函数 $f(x), g(x)$ 和 $h(x) = f(Hx)$ (这里 H 是矩阵) 一般只有

$$\partial(f + g)(x) \supseteq \partial f(x) + \partial g(x), \quad \partial h(x) \supseteq H^T \partial f(Hx).$$

因此为了简单起见, 我们**假设有正则条件**使得出现类似情形时, 我们总有

$$\partial(f + g)(x) = \partial f(x) + \partial g(x), \quad \partial h(x) = H^T \partial f(Hx).$$

第二步, 我们把 (3.1) 化成一个等价的不动点算子 T ($\mathbb{R}^n$ 上点到点的映射) :

$$z = T(z).$$

这个转化一般直接套用一个标准的分裂形式. 这个过程中要考虑 A, B, C 的性质, 使得计算 $T(z)$ 可以分解为对 A, B, C 的独立计算, 达到化繁为简的目的. 另外, 还需要 T 的迭代能够收敛到一个不动点. 这里的 z 在相对简单的分裂形式中就是 x 本身; 在一些较复杂的分裂形式中, z 是 x 与其对偶变量的某种组合.

第三步, 选取起始点 z^0, 迭代获得

$$z^{k+1} = T(z^k), \quad k = 0, 1, \ldots.$$

这是最基本迭代方法. 此外, 还有在此基础上建立的外推法、随机坐标法、异步坐标法、融入二阶信息的方法, 等等.

3.2. 单调算子

我们最关心两类单调算子. 第一类, **单调最大算子** (maximally monotone operator)：如果一个算子 A 满足

$$\langle u-v, x-y\rangle \geq 0, \quad \forall x, y \in \mathbb{R}^n,\ u \in A(x), v \in A(y),$$

则我们说 A 是**单调**的. 它把一维单调函数拓展到了高维映射. 如果一个算子 A 满足

$$\langle u-v, x-y\rangle \geq \mu\|x-y\|^2, \quad \forall x, y \in \mathbb{R}^n,\ u \in A(x), v \in A(y),$$

则我们说 A 是 μ-**强单调**的, $\mu > 0$.

如果不存在另一个不同的单调算子 B 使得 $A(x) \subseteq B(x)$, $\forall x$, 则 A 是**最大**的. 投影到一个非空的闭凸集的算子是单调最大的. 对于一个凸函数 (且 proper 和 closed), ∂f 也是单调最大的. 此外, 对称半正定矩阵和反对称矩阵也是单调最大算子.

第二类, **余强制算子** (cocoercive operator)：如果一个算子 A 是单值的并且满足

$$\langle A(x)-A(y), x-y\rangle \geq \beta\|A(x)-A(y)\|^2, \quad \forall x, y \in \mathbb{R}^n,\ \beta > 0,$$

则我们说 A 是 β-**余强制**的. 余强制算子总是单调的, 并且是单映射. 谱半径为 $1/\beta$ 的对称半正定矩阵是 β- 余强制的. 对于一个可导的凸函数 f (且 proper 和 closed), 如果其倒数的 Lipschitz 常数是 $1/\beta$, 则算子 ∇f 是 β- 余强制的.

函数 f 是	凸的	μ- 强凸的	凸的, 且有 β^{-1}-Lipschitz 连续倒数
算子 ∂f 是	单调的	μ- 强单调的	β- 余强制的

一个单调算子的入门介绍是 [7], 一个利用二维作图刻画单调算子及其收敛性质的文章是 [8], 一本研究单调算子的参考书是 [9].

当问题中有多个算子相加的时, 两个余强制算子的和可以当成一个余强制算子看待, 只需要适当的增加余强制的参数, 至多是单个算子 β 之和. 但是, 是否合并两个自然形成的两个最大单调算子 A, B 要看是否能够有效计算替换为 $(A+B)$ 的反算子 (下面定义).

3.3. 第一步：建立算子包含问题

结合具体的无中心优化模型, 我们进行第一步. 下面我们就一致优化模型举一些例子. 假设

- 所有函数都是 proper 和 closed 的凸函数;
- W 满足第一节的谱和结构条件, 并且 $W = W^T$.

(由于篇幅关系, 针对 $W \neq W^T$ 的 Push-Sum 及其更一般的无中心优化方法, 如 [10,11], 不在本文做介绍.)

模型 (2.5) 的最优条件是目标函数倒数为零, 即

$$0 \in \nabla f(\mathbf{x}^*) + \partial g(\mathbf{x}^*) + \frac{1}{\rho}(I-W)\mathbf{x}^*.$$

引入算子 $A = \partial g(\mathbf{x}^*)$ 和 $B = \nabla f + \frac{1}{\rho}(I-W)$, 我们就得到了形式 (3.1b). 这里, A 是最大单调的, B 是 β 余强制的, 并有 $1/\beta \leq L_f + \frac{1}{\rho}\lambda_{\max}(I-W)$ (其中 L_f 代表 ∇f 的 Lipschitz 常数).

对于模型 (2.3), 我们假设 $f_i \equiv 0$ (加入 f_i 并使用 ∇f_i 会需要更复杂的变形), 再引入任意凸函数 h, 因此得到一般形式

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}}\, g(\mathbf{x}) + h(\mathbf{y}), \quad \text{subject to } G\mathbf{x} + H\mathbf{y} = 0. \tag{3.2}$$

对这样的模型可以直接使用 ADMM. 但是, 不用 ADMM 而直接得到形式 (3.1b) 也很容易: 引入

$$g_G(\mathbf{z}) = \inf\{g(\mathbf{x}) : G\mathbf{x} = \mathbf{z}\}.$$

这里, $\mathbf{z}$ 是 g 的间接输入. 如果矩阵 G 可逆, 则 $g_G(\mathbf{z}) = g(G^{-1}\mathbf{z})$; 否则, $G\mathbf{x} = \mathbf{z}$ 可能有无穷多解, 我们在所有满足 $G\mathbf{x} = \mathbf{z}$ 的 $\mathbf{x}$ 里面选择使得 $g(\mathbf{x})$ 最小的那个 (当然, 正如 ADMM 的子问题一样, (3.2) 中的最小不一定取到, 所以用 inf). 这个函数叫做 infimal-postcomposition. 利用这个函数定义, 我们可以把模型 (3.2) 写成

$$\underset{\mathbf{z}}{\text{minimize}}\, g_G(\mathbf{z}) + h_H(-\mathbf{z}).$$

因为 g, h 是凸函数, 可以证明 g_G, h_H 也是凸函数, 在合适的正则条件下, 上述个问题的最优条件拥有形式 (3.1b):

$$0 \in \partial g_G(\mathbf{z}^*) + \partial_{\mathbf{z}} h_H(-\mathbf{z}^*), \tag{3.3}$$

(请注意 $\partial_{\mathbf{z}}$ 的下标.) 并且 ∂g_G 和 $\partial_{\mathbf{z}} h_H(-\cdot)$ 都是最大单调的. 对 (3.3) 直接应用后面将介绍的 Douglas-Rachford 分裂法又会得到 ADMM 法 [12, 13]. 对于凸问题, 这两种方法是等价的.

为了清晰起见, 我们给明 h_H. 在 (2.3) 里, h 是零函数, 所以 $h_H(-\mathbf{z}^*) = \delta_{\text{span}(H)}$ (即子空间 $\text{span}(H)$ 的 delta 函数: 当 $\mathbf{z}^* \in \text{span}(H)$, 等于 0; 否则, 等于 ∞). 并且, 当 $\mathbf{z}^* \in \text{span}(H)$, $\partial_{\mathbf{z}} h_H(-\mathbf{z}^*)$ 等于子空间 $\text{span}(H^T)$; 否则为空集 $\emptyset$.

最后, 我们利用拉格朗日鞍点, 给出模型 (2.4) 的等价最优条件. 首先由于 $(I - W)$ 是对称半正定的, 存在方阵 U 使得

$$U^T U = \frac{1}{2}(I - W). \tag{3.4}$$

这里加上 $1/2$ 仅是为了之后的推导更简洁. 另外, $(I - W)\mathbf{x} = 0$ 当且仅当 $U\mathbf{x} = 0$, 因此后者可以作为线性约束代替前者.

定义拉格朗日函数

$$L(\mathbf{x}; \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{\alpha}\mathbf{y}^T U\mathbf{x}.$$

这里 $\alpha > 0$ 是任意数. L 对 $\mathbf{x}$ 是凸的, 对 $\mathbf{y}$ 是线性的. 在一定的正则假设下, (2.4) 的最优条件是

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \begin{bmatrix} \partial_{\mathbf{x}} L(\mathbf{x}; \mathbf{y}) \\ \partial_{\mathbf{y}}(-L(\mathbf{x}; \mathbf{y})) \end{bmatrix},$$

即

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \Bigg(\underbrace{\begin{bmatrix} \nabla f & 0 \\ 0 & 0 \end{bmatrix}}_{B} + \underbrace{\begin{bmatrix} \partial g & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{\alpha}U^T \\ -\frac{1}{\alpha}U & 0 \end{bmatrix}}_{A} \Bigg) \underbrace{\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}}_{\mathbf{z}}, \tag{3.5}$$

即我们又得到了 (3.1b) 的形式. 这里 α 是一个自由参数. 我们将在之后解释为什么使用矩阵 U 并将 B 定义为两个算子之和.

3.4. 第二步：三个常用的算子分裂法

下面介绍一下第二步中的几个常用算子和分裂方法. 我们先给出它们, 然后给出已知的收敛条件.

一个最大单调算子 A 总有对应的**反算子** (resolvent)：

$$J_{\lambda A} = (I + \lambda A)^{-1}.$$

该定义里用到了**逆算子**的概念：一个算子由它的图 $\mathrm{graph}(A) = \{(x, y) : y \in A(x)\}$ 唯一定义; 逆算子 A^{-1} 则是由 $\mathrm{graph}(A^{-1}) = \{(y, x) : y \in A(x)\}$ 定义的算子. 利用逆算子, 我们可以得到

$$(3.1a) \iff x = J_{\lambda A}(x), \quad \forall \lambda > 0,$$

从而把一个求零点的问题转化为一个不动点问题 ($J_{\lambda A}$ 是一个点到点的映射). 我们把迭代 $x^{k+1} = J_{\lambda A}(x^k)$ 称作 **proximal-point 方法**或者 PPA.

对于一个凸 (且 proper 和 closed 的) 函数, 考虑算子 $\partial f(x) = \{g : f(y) \geq f(x) + \langle g, y - x\rangle \ \forall\ y\}$, 则其反算子就是邻近点算子：

$$J_{\lambda \partial f}(x) = \arg\min_w f(w) + \frac{1}{2\lambda}\|w - x\|_2^2.$$

这个算子所输出的 $J_{\lambda \partial f}(x)$ 位于 f 的定义域中, 在最小化 f 和靠近 x 之间取得平衡. 这样的算子我们也写作 $\mathbf{prox}_{\lambda f} = J_{\lambda \partial f}(x)$. 对于很多函数 f, $\mathbf{prox}_{\lambda f}$ 很容易计算, 在一阶算法中很常见; 详见 [14] [15, §2.2.7] [16].

即便两个最大单调算子 A, B 各自有易于计算的 $J_{\lambda A}, J_{\lambda B}, J_{\lambda(A+B)}$ 往往是不容易计算的. 所以我们使用分裂法将他们分开处理.

定义了反算子, 我们接下来定义正算子. 正算子要求输入的 A 是单映射 (点映射到点). A 的正算子定义为

$$F_{\lambda A} := (I - \lambda A).$$

一个常见的例子是 $A = \nabla f$, 对应的 $F_{\lambda A}$ 是梯度下降算子. 对于单映射 A, 显然

$$(3.1a) \iff x = F_{\lambda A}(x), \quad \forall \lambda > 0.$$

针对两个算子的问题 (3.1b), 在 A 是最大单调和 B 是单射的前提下, [17, 18] 分析了正反分裂算子：

$$(3.1b) \iff x = J_{\lambda A} F_{\lambda B}(x), \quad \forall \lambda > 0. \tag{3.6}$$

一个推导 "$\iff$" 的简单过程是

$$\begin{aligned} 0 \in A(x) + B(x) &\Leftrightarrow x - \lambda B(x) \in x + \lambda A(x) \\ &\Leftrightarrow (I - \lambda B)x \in (I + \lambda A)x \\ &\Leftrightarrow \underbrace{(I + \lambda A)^{-1}}_{J_{\lambda A}} \underbrace{(I - \lambda B)}_{F_{\lambda B}} x = x. \end{aligned}$$

基于 (3.6) 的迭代叫做正反算子分裂法：选定 $\lambda > 0$ 和 x^0, 迭代

$$x^{k+1} = J_{\lambda A} F_{\lambda B}(x^k). \tag{3.7}$$

去掉 B 为单射的条件, 现在假设 A, B 均为最大单调. [13] 提出了 Peaceman-Rachford 分裂 (PRS) 算子:

$$(3.1b) \iff z = (2J_{\lambda A} - I)(2J_{\lambda B} - I)z,\ x = J_{\lambda B}(z), \quad \forall \lambda > 0. \tag{3.8}$$

其推导可见 [7]. 将 $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$ 与 I 做平均所得的算子叫做 Douglas-Rachford 分裂 (DRS) 算子 [13]:

$$(3.1b) \iff z = \left(\frac{1}{2}I + \frac{1}{2}(2J_{\lambda A} - I)(2J_{\lambda B} - I)\right) z. \tag{3.9}$$

显然, PRS 和 DRS 两个算子拥有相同的不动点集合, 但是它们会产生不同的迭代序列. DRS 迭代

$$z^{k+1} = \frac{1}{2}z^k + \frac{1}{2}(2J_{\lambda A} - I)(2J_{\lambda B} - I)z^k \tag{3.10}$$

有更好的收敛性质 (但是, 如果 PRS 也收敛, PRS 可能更快) .

在下面的命题中, 我们总结一些已有的收敛结果.

命题 1.

1. 如果算子 (3.6),(3.8),(3.9) 均可定义 (即, A, B 均为最大单调, B 是单射) , 则对任意给定 $\lambda > 0$, (3.6) 的不动点集 S 与 (3.8) 和 (3.9) 的不动点集合 T, 有如下关系

   $$S = \{J_{\lambda B}z : z \in T\}.$$

   S 和 T 允许同为空集.

2. 假设 (3.6) 存在不动点, A 是最大单调的, B 是 β- 余强制的, 则对于步长 $\lambda \in (0, 2\beta)$ 和任意 x^0, (3.7) 收敛到一个不动点. 收敛速度是 $\|x^{k+1} - x^k\|^2 = o(1/k)$.

3. 假设 (3.9) 存在不动点, A, B 是最大单调的, 则对于步长 $\lambda \in (0, \infty)$ 和任意 x^0, (3.10) 收敛到一个不动点. 收敛速度是 $\|z^{k+1} - z^k\|^2 = o(1/k)$.

使用 $\|\cdot\|^2$ (而不是 $\|\cdot\|$) 是惯例, 也是因为 $\|\cdot\|^2$ 自然地出现在收敛分析中. 这里给出的速度是紧的 [19].

我们只归纳相对简单的条件和收敛结果, 不再讨论条件增强之后我们所能得到的收敛加速. 这些问题当然很重要; 有兴趣的读者请参见 [20].

3.5. 度量加持的算子分裂

在上面的讨论中, 我们一直使用欧式距离, 比如用欧式距离定义了邻近点算子. 我们可以把欧式距离换成一个对称正定矩阵 (对称半正定矩阵) M 导出的模 (半模 semi-norm):

$$\|\mathbf{x}\|_M = \sqrt{\mathbf{x}^T M \mathbf{x}}.$$

在算子分裂中, 使用 $\|\mathbf{x}\|_M$ 有两个主要作用: 一是做预条件加速, 这时 M 往往是 Hessian 矩阵或者是它的一个易于计算的近似; 二是把原对偶的计算拆分开来. 我们这里需要的是第二

个功能. 首先, 我们仍然推导一下带 M 的正反分裂算子 (由于下面将要在 $\mathbf{z}$ 上使用该算子, 我们不用 $\mathbf{x}$ 了):

$$\begin{aligned} 0 \in A(\mathbf{z}) + B(\mathbf{z}) &\Leftrightarrow M\mathbf{z} - B(\mathbf{z}) \in M\mathbf{z} + A(\mathbf{z}) \\ &\Leftrightarrow (I - M^{-1}B)\mathbf{z} \in (I + M^{-1}A)\mathbf{z} \\ &\Leftrightarrow \underbrace{(I + M^{-1}A)^{-1}}_{J_{M,A}}\underbrace{(I - M^{-1}B)}_{F_{M,B}}\mathbf{z} = \mathbf{z}. \end{aligned} \tag{3.11}$$

由此推导出的 M 加持的正反分裂迭代是

$$\mathbf{z}^{k+1} = J_{M,A}F_{M,B}\mathbf{z}^k.$$

我们暂时不用关心这个迭代的具体实现, 实际计算会用一个合适的 M 再直接使用 (3.11) 求解

$$M\mathbf{z}^k - B(\mathbf{z}^k) \in (M + A)\mathbf{z}^{k+1}. \tag{3.12}$$

对于 (3.5), 我们面临的问题是原对偶被那个反对称矩阵捆绑在一起, 如果直接使用正反分裂算子的话, 必对 $\mathbf{x}$ 和 $\mathbf{y}$ 同时更新, 由于反对称矩阵并非 cocoercive, 因此必然导致矩阵求逆. 然而, 我们无法利用无中心计算一次性完成网络上的矩阵求逆. 这时, 一个巧妙 M 就能派上用场, 将 $\mathbf{x}$ 和 $\mathbf{y}$ 更新解耦. 取

$$M = \frac{1}{\alpha}\begin{bmatrix} I & U^T \\ U & I \end{bmatrix},$$

对于度量 M 加持的正反分裂算子和 (3.5) 中给出的 A, B, 我们从 (3.12) 得到

$$\left(\frac{1}{\alpha}\begin{bmatrix} I & U^T \\ U & I \end{bmatrix} - \begin{bmatrix} \nabla f & 0 \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}^k \\ \mathbf{y}^k \end{bmatrix} \in \left(\frac{1}{\alpha}\begin{bmatrix} I & U^T \\ U & I \end{bmatrix} + \begin{bmatrix} \partial g & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{\alpha}U^T \\ -\frac{1}{\alpha}U & 0 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}^{k+1} \\ \mathbf{y}^{k+1} \end{bmatrix}.$$

注意到右侧第二行可以消去 U, 并且第二行左右均为单值, 故左右同乘 α 后得到

$$\begin{aligned} (I - \alpha\nabla f)\mathbf{x}^k + U^T\mathbf{y}^k &\in (I + \alpha\partial g)\mathbf{x}^{k+1} + 2U^T\mathbf{y}^{k+1}, \\ U\mathbf{x}^k + \mathbf{y}^k &= \mathbf{y}^{k+1}. \end{aligned}$$

这个算法在优化里叫做 Chambolle-Pock [21], 在算子分裂里叫做 Condat-Vu [22, 23].

现在, 计算 $\mathbf{y}^{k+1}$ 不再依赖于 $\mathbf{x}^{k+1}$ 了, 所以可以首先计算 $\mathbf{y}^{k+1}$ 再带入后计算 $\mathbf{x}^{k+1}$; 更进一步, 令

$$\mathbf{w}^k = U^T\mathbf{y}^k,$$

再结合 (3.4) 我们就得到了适应 W 的原对偶分裂法:

$$\begin{aligned} \mathbf{w}^{k+1} &= U^TU\mathbf{x}^k + \mathbf{w}^k = \mathbf{w}^k + \frac{1}{2}(I - W)\mathbf{x}^k, \\ \mathbf{x}^{k+1} &= \underbrace{(I + \alpha\partial g)^{-1}}_{\mathbf{prox}_{\alpha g}}\left(W\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k) - \mathbf{w}^k\right). \end{aligned}$$

每一次迭代只需要计算 $W\mathbf{x}$, $\alpha\nabla f(\mathbf{x})$, $\mathbf{prox}_{\alpha g}$ 各一次.

4. 无中心优化的算法

现在算子分裂法已经准备就绪, 这节中利用这些方法获得无中心优化的算法.

4.1. 正反分裂法

这一节里面, 我们将正反分裂方法运用在模型 (2.5) 上.

首先, 正反分裂方法在无约束优化问题上给出了 proximal-gradient 法. 假设函数 s 可导, t 为 proximable, 考虑问题

$$\underset{x}{\text{minimize}}\ \ s(x)+t(x).$$

(无约束与有约束并没有本质区别, 因为我们可以将约束 $x\in C$ 以 delta 函数 $\delta_C(x)$ 融入目标中.) proximal-gradient 法从任意 x^0 出发, 迭代形式是

$$x^{k+1}=\mathbf{prox}_{\lambda t}\left(x^k-\lambda\nabla s(x^k)\right),\quad k=0,1,\ldots. \tag{4.1}$$

迭代公式中的算子是 (3.6) 的特殊形式. 具体来说, 在 (3.6) 中我们取 $B=\nabla s$, $A=\partial t$, 就得到了上述 proximal-gradient 算子.

对于无约束模型 (2.5), 我们令

$$\begin{aligned} s(\mathbf{x})&:=\sum_{i=1}^n f_i(x_i)+\frac{1}{2\rho}\|\mathbf{x}\|^2_{I-W},\\ t(\mathbf{x})&:=\sum_{i=1}^n g_i(x_i), \end{aligned}$$

采用 (4.1) 并取步长 $\lambda=\rho$ (简化起见) 即得到

$$\mathbf{x}^{k+1}=\mathbf{prox}_{\rho g}\left(\mathbf{x}^k-\rho W\nabla f(\mathbf{x}^k)\right),\quad k=0,1,\ldots.$$

由于 f,g 有完全可分结构, 计算 ∇f 和 $\mathbf{prox}_{\alpha g}$ 的工作分解到每个节点独立完成. 给定 $\nabla f(\mathbf{x}^k)$ 时, $W\nabla f(\mathbf{x}^k)$ 可以通过相邻节点间的通讯实现. 具体实现过程为:

$$s_i=\nabla f_i(x_i^k),\qquad i=1,\ldots,n, \tag{4.2a}$$

$$t_i=w_{ii}s_i+\sum_{j\in N_i}w_{ij}s_i,\qquad i=1,\ldots,n \tag{4.2b}$$

$$x_i^{k+1}=\arg\min_{x_i} g_i(x_i)+\frac{1}{2\rho}\|x_i-(x_i^k-\rho t_i)\|^2,\qquad i=1,\ldots,n. \tag{4.2c}$$

这里, (4.2a) 和 (4.2c) 为每个节点的独立计算; (4.2b) 需要相邻节点之间的通讯.

在 $g_i\equiv 0$ 时, 步骤 (4.2c) 可以化简为 $x_i^{k+1}=x_i^k-\rho t_i,\ i=1,\ldots,n$. 这个方法就是前述的方法 (1.5), 出自 [24], 叫做无中心梯度下降 (decentralized gradient descent) 法. 收敛速度我们已经在 (1.1) 小节里讨论了. 对于 g_i 存在的情况, 乃至一些非凸情况, [6] 做了收敛分析.

我们无法直接将 proximal-gradient 法运用在其余的模型 (2.1)–(2.4) 上. 这是因为这些模型都带有约束, 而约束在 proximal-gradient 法中只能用 proximal 处理. 这就造成了两个问题, 首先, 我们已经对 $\sum_{i=1}^n g_i(x_i)$ 使用了 proximal 算子, 它与约束是 proximal 不兼容的. 其次, 即使单独对 (2.1)–(2.4) 里的约束使用 proximal 算子, 我们得到的是对 $\{\mathbf{x}: x_1=x_2=\cdots=x_n\}$ 的投影 (那三个问题中的约束是等价的) , 但这个投影需要全局信息, 无法通过相邻节点间的一两次通讯直接完成. 所以, 即使对 (2.1)–(2.4) 使用支持两个不同 proximable 算子的三算子分裂法 [25], 也不能得到一个无中心算法.

4.2. 无中心 ADMM 法

我们将 Linearized ADMM 直接运用在问题 (2.2) 上就能得到一个无中心化方法.

为了表达简便, 我们使用等价模型 (2.3). Linearized ADMM 给我们

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \langle \nabla f(\mathbf{x}^k), \mathbf{x}\rangle + g(\mathbf{x}) + \frac{\rho}{2}\|G\mathbf{x} + H\mathbf{y}^k + \mathbf{z}^k\|_F^2 + \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^k\|_F^2, \tag{4.3}$$

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} \frac{\rho}{2}\|G\mathbf{x}^{k+1} + H\mathbf{y} + \mathbf{z}^k\|_F^2, \tag{4.4}$$

$$\mathbf{z}^{k+1} = G\mathbf{x}^{k+1} + H\mathbf{y}^{k+1} + \mathbf{z}^k.$$

因为 x_i 定义在每个节点上, y_e 定义在每条边上, 所以考虑到 f, g 的节点可分结构以及 G, H 的结构, 第一步 (4.3) 分化为每个节点 i 的独立计算, 第二步 (4.4) 分化为每条边上的通讯和计算, 第三步 (4.3) 分化为每条边上的计算. 第二、三步由每边两端的节点共同完成.

文献 [26–29] 最早分别在无中心优化上使用了 ADMM (虽然模型的设计与这里有区别).

由于 (4.4) 简化为线性方程组, 并且矩阵 H 带有很强的结构, 所以我们可以使用变量代换消去一些变量, 得到更为简单但等价的算法. 在 $f_i \equiv 0, \beta = 0$ 的情况下的具体化简过程和结果, 可以参见 [30]. 这里不在赘述.

除了使用算子收敛的一般理论外, 文献 [31, 32] 也在一定意义下证明了 ADMM 方法的 $O(1/k)$ 收敛速度. 此外, 文献 [30] 在一定条件下证明了线性收敛. 对于 (中心化的) ADMM 的凸分析, 包括收敛速度、速度的下界等, 参考 [19, 第 8 节] 和 [20].

4.3. EXTRA 和 PG-EXTRA 法

回顾一下, 我们对模型 (2.4) 使用了鞍点最优条件, 再使用矩阵 M 引导出的度量所加持的正反算子分裂, 最终得到了算法 (3.13). 该算法中, 变量 w_i 定义在每个节点上, 每轮迭代中通讯和计算一次 $W\mathbf{x}^k$, 其余的计算利用 f, g 的可分性质, 分解成每个节点上的独立计算. 这个算法由 [33] 首先提出, 取名 EXTRA, 其中字母 EX 代表 "exact" 以强调可以使用一个常数 α 获得精确的收敛. EXTRA 没有考虑 g_i, 后在 [34] 里拓展成 PG-EXTRA (PG 是 proximal-gradient 的缩写). 这两个算法的收敛速度是 $O(1/k)$, 此外作者还首次证明了, 只要目标函数之和为强凸 (单个凸函数不需要强凸), 则有线性收敛.

相比 [33, 34] 里的推导, 度量加持的正反算子分裂更加一般和简洁, 提供对 PG-EXTRA 的一个系统性刻画. 算法分裂的拓展与增强可以自然地运用到 PG-EXTRA 上.

5. 其他方法

[35] 使用了对偶分裂法, [36] 引入了梯度跟踪技术, [37–40] 得到了无需不同节点相互协调的梯度步长, [41] 在动态图上获得线性收敛, [42, 43] 有与 [39] 类似的更新方法但更加支持左随机矩阵, [44] 给出了带有光滑强凸目标函数的无中心优化的梯度和通讯下界, [44, 45] 提出了加速的算法, [46, 47] 引入了来由双层循环的加速算法, [46, 48] 使用对偶梯度上升法并做了加速, [49] 把 PG-EXTRA 扩展到异步和带时延的情形, [50, 51] 将随机梯度法融入无中心优化. 由于近年来无中心优化方法在不同方向上得到了提高和应用, 相关文章大量涌现. 由于篇幅和作者知识的限制, 本文没有介绍和引用很多优秀的无中心优化方法, 谨此向同行致歉.

6. 总结

本文介绍了无中心一致优化问题及其相关的算法, 重点阐述了问题中由于去中心化带来的邻近节点间的通信以及节点内的计算. 通过引入算子分裂方法, 本文将求解无中心一致优化问题的算法进行了较为统一的表达, 给出了相关的理论背景和算法的推导步骤. 算子分裂框架可以简洁、准确地梳理了一些近年来求解无中心一致优化问题的新算法. 借此作者希望向读者提供研究和求解该类问题的有效途径.

致谢. 本文作者感谢两位匿名审稿人的建议和指正, 并特别感谢密歇根州立大学的严明教授对本文内容的帮助.

参 考 文 献

[1] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed. A diffusion RLS scheme for distributed estimation over adaptive networks. In 2007 IEEE 8th Workshop on Signal Processing Advances in Wireless Communications. 2007, 1–5.

[2] F. S. Cattivelli and A. H. Sayed. Diffusion LMS strategies for distributed estimation. IEEE Transactions on Signal Processing[J]. 2010, 58(3):1035–1048.

[3] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. SIAM Journal on Optimization[J]. 2016, 26(3):1835–1854.

[4] Annie I.-An Chen. Fast Distributed First-Order Methods. Thesis, Massachusetts Institute of Technology, 2012.

[5] Dusan Jakovetic, Joao Xavier, and Jose M. F. Moura. Fast distributed gradient methods. IEEE Transactions on Automatic Control[J]. 2014, 59(5):1131–1146.

[6] Jinshan Zeng and Wotao Yin. On nonconvex decentralized gradient descent. IEEE Transactions on Signal Processing[J]. 2018, 66(11):2834–2848.

[7] Ernest K. Ryu and Stephen Boyd. Primer on monotone operator methods. Technical report, Stanford University, 2015.

[8] Ernest Ryu, Robert Hannah, and Wotao Yin. Scaled relative graph: Nonexpansive operators via 2D Euclidean geometry. arXiv:1902.09788, 2019.

[9] Heinz H. Bauschke and Patrick L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces. CMS Books in Mathematics. Springer New York, New York, NY, 2011.

[10] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS '03, pages 482–, Washington, DC, USA, 2003. IEEE Computer Society.

[11] Angelia Nedic and Alex Olshevsky. Distributed optimization over time-varying directed graphs. IEEE Transactions on Automatic Control[J]. 2015, 60(3):601–615.

[12] 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[J]. 1975, 9(R2):41–76.

[13] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. SIAM Journal on Numerical Analysis[J]. 1979, 16(6):964–979.

[14] Neal Parikh and Stephen Boyd. Proximal algorithms. Foundations and Trends in Optimization[J]. 2014, 1(3):127–239.

[15] Hao-Jun Michael Shi, Shengyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. UCLA CAM Report 16-67, 2016.

[16] Giovanni Chierchia, Emilie Chouzenoux, Patrick L. Combettes, and Jean-Christophe Pesquet. The proximity operator repository (user's guide). Technical Report version 0.1, proximity-operator.net, 2017.

[17] Ronald E. Bruck Jr. An iterative solution of a variational inequality for certain monotone operators in Hilbert space. Bulletin of the American Mathematical Society[J]. 1975, 81(5):890–893.

[18] Gregory B. Passty. Ergodic convergence to a zero of the sum of monotone operators in Hilbert space. Journal of Mathematical Analysis and Applications[J]. 1979, 72(2):383–390.

[19] Damek Davis and Wotao Yin. Convergence rate analysis of several splitting schemes. In Roland Glowinski, Stanley Osher, and Wotao Yin, editors, Splitting Methods in Communication, Imaging, Science and Engineering, Chapter 4, pages 115–163. Springer, 2016.

[20] Damek Davis and Wotao Yin. Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions. Mathematics of Operations Research[J]. 2017, 42(3):783–805.

[21] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision[J]. 2011, 40(1):120–145.

[22] Laurent Condat. A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. Journal of Optimization Theory and Applications[J]. 2013, 158(2):460–479.

[23] B. C. Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. Advances in Computational Mathematics[J]. 2011, 38(3):667–681.

[24] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. IEEE Transactions on Automatic Control[J]. 2009, 54(1):48–61.

[25] Damek Davis and Wotao Yin. A three-operator splitting scheme and its optimization applications. Set-Valued and Variational Analysis[J]. 2017, 25(4):829–858.

[26] Dimitri P. Bertsekas and John N. Tsitsiklis. Parallel and Distributed Computation: Numerical Methods. Prentice Hall, Englewood Cliffs, N.J, 1989.

[27] Gonzalo Mateos, Juan Andrés Bazerque, and Georgios B. Giannakis. Distributed sparse linear regression. IEEE Transactions on Signal Processing[J]. 2010, 58(10):5262–5276.

[28] Ioannis D. Schizas, Alejandro Ribeiro, and Georgios B. Giannakis. Consensus in ad hoc WSNs with noisy links - part I: Distributed estimation of deterministic signals. IEEE Transactions on Signal Processing[J]. 2008, 56(1):350–364.

[29] Qing Ling and Zhi Tian. Decentralized sparse signal recovery for compressive sleeping wireless sensor networks. IEEE Transactions on Signal Processing[J]. 2010, 58(7):3816–3827.

[30] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the ADMM in decentralized consensus optimization. IEEE Transactions on Signal Processing[J]. 2014. 62(7):1750–1761.

[31] Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus ADMM. IEEE Transactions on Signal Processing[J]. 2015, 63(2):482–497.

[32] E. Wei and A. Ozdaglar. On the $o(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In 2013 IEEE Global Conference on Signal and Information Processing, 2013, 551–554.

[33] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for

decentralized consensus optimization. SIAM Journal on Optimization[J]. 2015, 25(2):944–966.

[34] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. IEEE Transactions on Signal Processing[J]. 2015, 63(22):6013–6023.

[35] Håkan Terelius, Ufuk Topcu, and Richard M. Murray. Decentralized multi-agent optimization via dual decomposition. IFAC Proceedings Volumes[J]. 2011, 44(1):11245–11251.

[36] Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. Automatica[J]. 2010, 46(2):322–329.

[37] A. Nedić, A. Olshevsky, W. Shi, and C. A. Uribe. Geometrically convergent distributed optimization with uncoordinated step-sizes. In 2017 American Control Conference (ACC), 2017, 3950–3955.

[38] Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In 2015 54th IEEE Conference on Decision and Control (CDC), 2055–2060, Osaka, 2015. IEEE.

[39] Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. IEEE Transactions on Signal Processing, early access, 2019.

[40] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. IEEE Transactions on Control of Network Systems[J]. 2018, 5(3):1245–1260.

[41] Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. SIAM Journal on Optimization[J] 2017, 27(4):2597–2633.

[42] Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H. Sayed. Exact diffusion for distributed optimization and learning — Part I: Algorithm development. IEEE Transactions on Signal Processing[J]. 2019, 67(3):708–723.

[43] Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H. Sayed. Exact diffusion for distributed optimization and learning — Part II: Convergence analysis. IEEE Transactions on Signal Processing[J]. 2019, 67(3):724–739.

[44] Kevin Scaman, Francis Bach, Sebastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In Advances in Neural Information Processing Systems 31, pages 2740–2749. Curran Associates, Inc., 2018.

[45] Guannan Qu and Na Li. Accelerated distributed Nesterov gradient descent. arXiv:1705.07176, 2017.

[46] César A. Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. arXiv:1809.00710, 2018.

[47] Huan Li, Cong Fang, Wotao Yin, and Zhouchen Lin. A sharp convergence rate analysis for distributed accelerated gradient methods. arXiv:1810.01053, 2018.

[48] Kevin Seaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In Proceedings of the 34th International Conference on Machine Learning 70, pages 3027–3036, 2017.

[49] Tianyu Wu, Kun Yuan, Qing Ling, Wotao Yin, and Ali H. Sayed. Decentralized consensus optimization with asynchrony and delays. IEEE Transactions on Signal and Information Processing over Networks[J]. 2018, 4(2):293–307.

[50] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan,

and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5330–5340. Curran Associates, Inc., 2017.

[51] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In International Conference on Machine Learning, pages 3043–3052, 2018.

OPERATOR SPLITTING METHODS FOR DECENTRALIZED OPTIMIZATION

Yin Wotao

(*DAMO Academy, Alibaba US;*

(*on leave from*) *Department of Mathematics, University of California, Los Angeles, US*)

Abstract

Many problems in multi-agent systems, due to communication restrictions, need to be solved in a decentralized manner. There is no data fusion center, so we must rely on short-distance communication between adjacent nodes to achieve the goal of the whole network. Compared with traditional (centralized) computing, decentralized computing is more suitable for distributed data, less subject to communication and computing bottlenecks, and easier to realize in some applications.

This article overviews the formulations and methods of decentralized consensus optimization. The objective of consensus optimization is that all the variables of the nodes converge to the same vector that minimizes the sum of their objective functions. This problem is solved by calculations at each node and data exchanges between adjacent nodes. Naive decentralized algorithms are much slower than their centralized counterparts. In order to make up for this gap, we review some recent methods through a unified framework of operator splitting.

Keywords: decentralized algorithms, consensus optimization, operator splitting, monotone operator

2010 Mathematics Subject Classification: 68Q85, 90C25, 93A14