北京大学

本科生毕业论文

题目：半定规划及其在组合优化中的应用

姓　　名：李东晨

学　　号：1900010681

院　　系：数学科学学院

专　　业：信息与计算科学

指导教师：邵嗣烘

二〇二三年六月

版权声明

摘要

半定规划问题是由线性规划推广而来的一类重要的凸优化问题, 在组合优化、控制、通信、统计、机器学习等应用领域都有重要的应用. 本文首先介绍了半定规划问题相关的基本概念, 如矩阵运算、近似理论、对偶理论等, 随后总结了历史上提出的各种算法的分类, 介绍了在理论、应用、未来发展等方面具有重要作用的四个多项式近似算法, 即椭球法, 内点法, 乘法加权更新法和近似条件梯度增广拉格朗日算法. 接下来, 本文重点介绍了半定规划在组合优化问题中的应用, 包括不同类型问题的松弛方法, 舍入技巧等, 同时也简单列举了半定规划问题在其它领域的应用和参考资料. 最后, 本文对半定规划算法的未来发展做了展望.

关键词: 半定规划, 凸规划, 组合最优化, 算法

Semidefinite Programming: Algorithms and Applications in Combinatorial Optimization

Dongchen Li (Information and Computing Sciences)
Directed by Sihong Shao

ABSTRACT

Semidefinite Programming (SDP) is an essential class of problems in convex optimization. It enjoys great importance in different applications, such as combinatorial optimization, control, communication, statistics and machine learning. Firstly, we introduce basic concepts about SDP, such as matrix operations, approximation theory and theory of duality. Secondly, we introduce four algorithms that play important roles in the theory, application and further developments of SDP chronologically. They are the ellipsoid method, the interior point method, the multiplicative weight update method and the sketchyCGAL algorithm. Then, we list several applications of SDP. The focus is mainly on combinatorial optimization. We detail the formulation of SDP relaxation for different problems and rounding skills. We also list some references for other applications. Finally, we state further possible development for SDP algorithms.

KEY WORDS: Semidefinite Programming, Convex Programming, Combinatorial Optimization, Algorithm

目录

第一章 引言 **1**
1.1 半定规划历史简介 1
1.2 论文结构 1

第二章 基础知识 **3**
2.1 数学字符对照表 3
2.2 线性代数 4
2.2.1 矩阵内积 4
2.2.2 矩阵的标准分解与半正定性 4
2.2.3 向量范数和矩阵范数 5
2.2.4 矩阵的奇异值分解, 低秩近似, 广义逆 5
2.3 半定规划 7
2.3.1 半定规划的标准形式 7
2.3.2 半定规划的对偶理论 8
2.4 凸集与凸函数 10

第三章 半定规划问题的算法 **13**
3.1 半定规划算法简述 13
3.2 椭球法 14
3.2.1 简述 14
3.2.2 算法原理 14
3.2.3 总结 18
3.3 内点法 19
3.3.1 简述 19
3.3.2 算法原理 19
3.3.3 总结 22
3.4 乘法加权更新算法 22
3.4.1 简述 22
3.4.2 算法原理 22
3.4.3 总结 26
3.5 sketchyCGAL 算法 26

3.5.1 简述 26
3.5.2 基础知识 26
3.5.3 算法原理 29

第四章 半定规划问题在组合优化中的应用 **33**
4.1 直接把组合优化问题建模为半定规划问题 33
4.1.1 图的最小单位距离表示问题 33
4.1.2 图的 Lovász theta 函数 33
4.2 导出组合优化问题的半定规划近似 34
4.2.1 组合优化问题的松弛 35
4.2.2 松弛解的舍入 40
4.3 半定规划问题的其他应用 42

第五章 结论与展望 **43**

参考文献 **45**

致谢 **51**

第一章　引言

1.1　半定规划历史简介

半定规划问题是著名的线性规划问题的推广, 也是最常见的凸优化问题之一. 对半定规划问题的深入研究从 20 世纪 80 年代开始, 其早期发展是与线性规划紧密相关的. Khachiyan[1]在 1979 年基于椭球法给出了线性规划的第一个多项式时间算法, 紧随其后, Grötschel, Lovász 和 Schrijver[2]在 1981 年也基于椭球法给出了半定规划问题的第一个多项式时间近似算法. 基于椭球法的算法尽管是多项式时间的, 但效率太低, 并不实用. 1984 年, Karmarkar 提出了线性规划的内点算法, 也是线性规划的第一个比较实用的多项式时间算法. 随后, Alizadeh[3], Kamath 和 Karmarkar[4-5]等人在 1992 年左右给出了半定规划问题的第一个多项式时间内点算法, 同样非常实用.

在实用算法出现后, 半定规划问题的理论和应用研究开始蓬勃发展, 其应用主要集中在组合最优化问题、系统设计和控制论中. 1995 年左右, 半定规划问题的研究进入了高峰, 大量的结果涌现出来, 包括半定规划问题最著名的应用 GW 算法（Goemans 和 Williamson[6]）以及若干不同结构, 效率更高的内点算法.

在 20 世纪初期后, 半定规划问题的研究进展逐渐减缓. 这一趋势的原因在于, 半定规划主要用于对相对困难的问题进行松弛, 以获得原问题的近似解. 然而, 这种松弛的代价是将问题的规模从 $O(n)$ 扩展到 $O(n^2)$, 导致求解过程的复杂度很高, 在大规模问题上应用困难. 随着越来越多基于迭代法和启发式方法的, 基本不改变问题规模的高效算法的提出, 半定规划松弛的应用变得更加有限.

2010 年以来, 半定规划问题的研究再次受到广泛关注. 机器学习的浪潮带来了大量新的半定规划问题, 例如最小二乘问题, 主成分分析等. 因此, 许多机器学习中常见的方法, 如低秩近似[7], 正交分解, Nyström 方法[8]等, 也被引入到半定规划问题中, 人们提出了很多新的高效近似半定规划算法, 在应用中取得了良好效果, 半定规划也逐渐重新引发应用领域的关注.

1.2　论文结构

在第二章中, 我们介绍与半定规划问题相关的一些基础知识, 如半正定矩阵, 对偶理论等.

在第三章中, 为了讨论半定规划问题算法的发展规律, 我们按照历史上发现的顺序介绍半定规划问题的几个著名算法, 如椭球法（1981）、内点法（1992）、乘法加权更新

算法（2007）和近期的近似条件梯度增广拉格朗日算法（2019）等, 并且同时穿插同类算法设计的通用思想, 以指导后面的进一步研究.

在第四章中, 我们讨论半定规划问题在组合最优化问题中的一些典型应用. 受限于篇幅和兴趣, 我们无法讨论半定规划问题的全部应用, 但是会简单介绍其它应用, 并给出若干参考资料.

在第五章中, 我们总结本文内容, 并提出对半定规划问题未来的若干展望.

第二章　基础知识

2.1　数学字符对照表

$\mathbf{A}$	矩阵
$\mathbf{b}$	向量
$\mathbf{b_i}$	第 i 个向量
b_i	向量 $\mathbf{b}$ 的第 i 个分量
$\mathbf{f}$	向量值函数
f	实值函数
$\mathbf{x} \geq \mathbf{y}$	对任意分量 i, $x_i \geq y_i$
$\mathbf{c}_n$	长度为 n 的全 c 向量
$\cdot$	向量内积
$\bullet$	矩阵内积(2.2.1)
Tr	迹函数
rank	矩阵的秩
$^\top$	转置
Diag($\mathbf{x}$)	以向量 $\mathbf{x}$ 为对角线的矩阵
diag($\mathbf{X}$)	取矩阵 $\mathbf{X}$ 的对角线上的向量
$\succeq$	矩阵上的矩阵上的 Löwner 序(2.2.4)
$\|\cdot\|$	向量范数和矩阵范数(2.2.3)
$[m]$	集合$\{1, 2, \cdots m\}$
∇	函数的梯度
∇^2	函数的 Hessian 矩阵
E	数学期望

2.2 线性代数

2.2.1 矩阵内积

一个矩阵 $\mathbf{A} \in \mathbb{R}^{m\times n}$ 称为 (n 阶) 方阵, 如果 $m = n$. 对于方阵 $\mathbf{A} \in \mathbb{R}^{n\times n}$, 定义其迹为 $\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{n} A_{ii}$, 即矩阵的对角元素之和. 进一步, 定义矩阵内积 $\mathbf{A} \bullet \mathbf{B} \triangleq \mathrm{Tr}\,\mathbf{A^T B} = \sum_{i,j=1}^{n} A_{ij}B_{ij}$ (有些文献中也记做 $\langle \mathbf{A}, \mathbf{B}\rangle$). 由于矩阵内积相当于把矩阵按列展开为向量做内积, 因此容易验证其确实是一种内积. 容易验证迹算子满足交换律, 即 $\mathrm{Tr}\,\mathbf{AB} = \mathrm{Tr}\,\mathbf{BA}$, 且 $\mathrm{Tr}\,\mathbf{A} = \mathrm{Tr}\,\mathbf{A}^\top$.

2.2.2 矩阵的标准分解与半正定性

定义方阵 $\mathbf{A}$ 对称, 如果 $\mathbf{A} = \mathbf{A}^\top$. $\mathbb{R}^{n\times n}$ 中的所有对称矩阵的集合记为 $\mathbb{S}^n$. 定义方阵 $\mathbf{A}$ 正交, 如果 $\mathbf{A}^\top\mathbf{A} = \mathbf{I}$. 特别地, 取其列向量为 $\mathbf{a_1}, ..., \mathbf{a_n}$, 则其两两正交, 构成 $\mathbb{R}^n$ 的一组正交基.

定义 2.2.1. 矩阵的对角化

任意方阵 $\mathbf{A} \in \mathbb{R}^{n\times n}$ 都可以分解为 $\mathbf{PDQ}^{-1}$ 的形式, 其中 $\mathbf{P}, \mathbf{Q}$ 是可逆矩阵, $\mathbf{D} = \mathrm{Diag}(\lambda_1, ..., \lambda_n)$ 是由矩阵 $\mathbf{A}$ 的特征值 $\lambda_1 \geq ... \geq \lambda_n$ 构成的对角矩阵, 该分解也称为矩阵的谱分解.

定义 2.2.2. 对称矩阵的对角化

任意对称矩阵 $\mathbf{A} \in \mathbb{S}^n$ 可以进一步分解为 $\mathbf{P}\,\mathrm{Diag}(\lambda_\mathbf{1}, ..., \lambda_\mathbf{n})\mathbf{P}^\top$ 的形式, 其中 $\mathbf{P}$ 为正交矩阵, $\lambda_1 \geq ... \geq \lambda_n$ 为矩阵 $\mathbf{A}$ 的特征值, 之后为了对不同矩阵的特征值做区分, 也记作 $\lambda_i(\mathbf{A})$.

推论 2.2.1. 对于矩阵 $\mathbf{A} \in \mathbb{S}^n$, $\mathrm{Tr}\,\mathbf{A} = \mathrm{Tr}\,\mathbf{PQP}^\top = \mathrm{Tr}\,\mathbf{PP}^\top\mathbf{Q} = \mathrm{Tr}\,\mathbf{Q} = \lambda_1 + ... + \lambda_n$.

定义 2.2.3. 半正定 (semidefinite)

定义一个矩阵 $\mathbf{A}$ 半正定, 若其满足以下任意 (相互等价的) 条件.

- 其所有特征值都大于等于 0, 或 $\lambda_n \geq 0$.
- 对任意向量 $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^\top\mathbf{Ax} \geq 0$.
- 存在矩阵 $\mathbf{C} \in \mathbb{R}^{n\times n}$, $\mathbf{A} = \mathbf{C}^\top\mathbf{C}$.

定义 2.2.4. Löwner 序

对任意方阵 $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$, 定义 $\mathbf{A} \succeq \mathbf{B}$ 当且仅当 $\mathbf{A} - \mathbf{B}$ 半正定. 该方阵上的二元关系称为 Löwner 序. 特别地, 若 $\mathbf{A}$ 是半正定的, 则记 $\mathbf{A} \succeq 0$. 可以证明, 这是一个偏序关系.

所有 n 维对称半正定矩阵的集合记为 $\mathbb{S}^n_+$. 半正定矩阵是本文的主要研究对象.

2.2.3　向量范数和矩阵范数

定义 2.2.5. 范数

在线性空间 E 上, 满足下列性质的实值函数 $||\cdot||$ 称为范数.

对任意 $\mathbf{x}, \mathbf{y} \in E, \alpha \in \mathbb{R}$:

$$\|\mathbf{x}\| \geq 0, \|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = 0,$$
$$\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|,$$
$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|.$$

根据内积的定义, 线性空间上的任意内积 $\cdot$ 都诱导出一个满足 $||\mathbf{x}|| = \sqrt{\mathbf{x} \cdot \mathbf{x}}, \forall x \in E$ 的范数 $||\cdot||$,

特别地, 定义在向量空间上的范数称为向量范数. 常见的向量范数为 p-范数, 记作 $||\cdot||_p$, 即 $||\mathbf{x}||_p = (x_1^p + \cdots + x_n^p)^{1/p}$. 根据该定义, 向量内积诱导的范数称为二范数, 记为 $||\mathbf{x}||_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}}$, 由于是最常用的范数, 因此也简记为 $||\mathbf{x}||$.

定义在矩阵空间上的范数称为矩阵范数. 同样地, 之前定义的矩阵内积诱导的范数称为 Frobenius 范数, 即 $||\mathbf{A}||_F = \sqrt{\mathbf{A} \bullet \mathbf{A}} = \sqrt{\mathrm{Tr}(\mathbf{A}^\top \mathbf{A})}$.

向量范数可以诱导出矩阵范数. 假设已有向量范数 $||\cdot||$, 那么对任意 $\mathbf{A} \in \mathbb{R}^{n \times m}$, 定义矩阵范数 $||\mathbf{A}|| = \max_{||\mathbf{x}||=1} ||\mathbf{A}\mathbf{x}||$. 根据定义可以验证其确实是一个矩阵范数. 因此, p-向量范数可以诱导 p-矩阵范数. 特别地, 2-向量范数诱导的 2-矩阵范数满足 $||\mathbf{A}||_2 = \max_{||\mathbf{x}||_2=1} ||\mathbf{A}\mathbf{x}||_2 = \sqrt{\lambda_1(\mathbf{A}^\top \mathbf{A})}$. 由于与特征值相关, 该范数也称为谱范数.

2.2.4　矩阵的奇异值分解, 低秩近似, 广义逆

矩阵的奇异值分解 (SVD) 可以视作矩阵对角化的推广. 对于 $m \times n$ 矩阵 $\mathbf{A}$, $\mathbf{A}$ 可以分解为 $\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^\top$ 的形式, 其中 $\mathbf{P}, \mathbf{Q}$ 为 $m \times m, n \times n$ 维正交矩阵, $\mathbf{\Sigma}$ 为 $m \times n$ 维对角矩阵 (只有 $\Sigma_{ii}, i \in [\min\{m, n\}]$ 非零).

此时, 有 $\mathbf{A}\mathbf{A}^\top = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^\top\mathbf{Q}\mathbf{\Sigma}^\top\mathbf{P}^\top = \mathbf{P} \wedge_{\mathbf{1}} \mathbf{P}^\top$, 同理有 $\mathbf{A}^\top\mathbf{A} = \mathbf{Q} \wedge_{\mathbf{2}} \mathbf{Q}^\top$. 这两个式子恰好是这两个对称矩阵 $\mathbf{A}\mathbf{A}^\top, \mathbf{A}^\top\mathbf{A}$ 的正交分解. 因此, $\mathbf{P}$ 的列向量是 $\mathbf{A}\mathbf{A}^\top$ 的特征向量, 称为右奇异向量, $\mathbf{Q}^\top$ 的行向量是 $\mathbf{A}^\top\mathbf{A}$ 的特征向量, 称为左奇异向量. Σ 的对角线上的元素称为奇异值.

注意到, $\mathbf{\Sigma}\mathbf{\Sigma}^\mathbf{T} = \wedge_1$ 是半正定矩阵 AA^T 的特征矩阵, 因此奇异值的平方与 AA^T 的特征值是一一对应的, 我们约定奇异值都非负. 同时, 不妨设 $\mathbf{\Sigma}$ 的奇异值从大到小排列. 矩阵 $\mathbf{A}$ 的秩为 r 当且仅当其有 r 个奇异值非零, 因此我们不妨将非零奇异值记为

$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$, 矩阵 $\mathbf{\Sigma}$ 可以分块为

$$\begin{bmatrix} \mathrm{Diag}(\sigma_1, \cdots, \sigma_r) & \mathbf{0_{r\times(n-r)}} \\ \mathbf{0_{(m-r)\times r}} & \mathbf{0_{(m-r)\times(n-r)}} \end{bmatrix}.$$

注意到, $\mathbf{A} = \mathbf{P\Sigma Q}^\top = \sum_{i=1}^{\min\{m,n\}} \sigma_i \mathbf{p_i q_i}^\top = \sum_{i=1}^{r} \sigma_i \mathbf{p_i q_i}^\top$, 因此, 对于秩为 r 的矩阵 A, 其奇异值分解 $\mathbf{P\Sigma Q}^\top$ 的大小实际上只需要是 $m \times r, r \times r, r \times n$ 维的.

不过, 完全计算一个矩阵的奇异值分解的时间复杂度很高, 为 $O(mn(m+n))$. 因此, 在实际应用中为了保证效率, 一般只求一部分奇异值分解, 例如只求最大的奇异值.

利用奇异值分解, 我们可以给出矩阵的低秩近似. Eckart-Young-Mirsky 定理表明, 这样构造的低秩近似在某种意义下是最优的.

定义 2.2.6. 低秩近似

定义矩阵 $\mathbf{A}$ 的秩 k 近似 ($k < r$) 为 $\mathbf{A_k} = \sum_{i=1}^{k} \sigma_i \mathbf{p_i q_i}^\top$, 显然其秩为 k.

定理 2.2.1. *(Eckart-Young-Mirsky 定理, 1936)*

在矩阵的谱范数 $||\cdot||_2$ 和 Frobenius 范数 $||\cdot||_F$ 下, $\mathbf{A}_k$ 是和矩阵 $\mathbf{A}$ 距离最近的秩 k 矩阵.

对于非奇异方阵 $\mathbf{X}$, 我们可以定义逆矩阵 $\mathbf{X}^{-1}$, 其满足 $\mathbf{X}^{-1}\mathbf{X} = \mathbf{X}\mathbf{X}^{-1} = \mathbf{I}$. 其它的矩阵则不能定义满足这样性质的矩阵. 但是, 参考逆矩阵的形式, 我们可以定义广义逆矩阵. 注意到, 非奇异方阵 $\mathbf{X}$ 可以做对角化分解 $\mathbf{P}\,\mathrm{Diag}(\lambda_1, \cdots, \lambda_n)\mathbf{Q}^{-1}$, 而其非奇异当且仅当 $\lambda_1, \cdots, \lambda_n$ 均非零. 此时, 其逆矩阵可以表示为 $\mathbf{Q}\,\mathrm{Diag}(\lambda_1^{-1}, \cdots, \lambda_n^{-1})\mathbf{P}^{-1}$. 因此, 一个自然的想法是不对值为 0 的特征值取逆, 并且把对角化分解更改为奇异值分解.

定义 2.2.7. 广义逆

对于一般的秩为 r 的 $m \times n$ 维矩阵 $\mathbf{A}$, 若其奇异值分解为

$$\mathbf{P}\begin{bmatrix} \mathrm{Diag}(\sigma_1, \cdots, \sigma_r) & \mathbf{0_{r\times(n-r)}} \\ \mathbf{0_{(m-r)\times r}} & \mathbf{0_{(m-r)\times(n-r)}} \end{bmatrix}\mathbf{Q}^\top,$$

则定义其广义逆 $\mathbf{A}^*$ 为

$$\mathbf{Q}\begin{bmatrix} \mathrm{Diag}(\sigma_1^{-1}, \cdots, \sigma_r^{-1}) & \mathbf{0_{r\times(n-r)}} \\ \mathbf{0_{(m-r)\times r}} & \mathbf{0_{(m-r)\times(n-r)}} \end{bmatrix}\mathbf{P}^\top.$$

定理 2.2.2. *(Moore-Ponrose, 1955)*

$\mathbf{A}^*$ 是满足以下方程组的唯一解.

$$\begin{aligned}
\mathbf{AXA} &= \mathbf{A},\\
\mathbf{XAX} &= \mathbf{X},\\
(\mathbf{AX})^\top &= \mathbf{AX},\\
(\mathbf{XA})^\top &= \mathbf{XA}.
\end{aligned}$$

2.3　半定规划

2.3.1　半定规划的标准形式

半定规划 (Semidefinite Programming, SDP) 是线性规划问题的推广. 因此, 在介绍半定规划问题之前, 我们先介绍线性规划问题.

线性规划问题的标准形式为

$$\begin{aligned}
\min \quad & \mathbf{c}\cdot\mathbf{x}\\
\text{s.t. } \forall j\in[m], \quad & \mathbf{a_j}\cdot\mathbf{x} = b_j\\
& \mathbf{x}\geq\mathbf{0}_n,
\end{aligned}$$

其中, $\mathbf{x}, \mathbf{a_j}, \mathbf{c}\in\mathbb{R}^n, \mathbf{b}\in\mathbb{R}^m$.

相对应地, 半定规划问题则相当于把线性规划问题推广到矩阵上. 我们需要把向量内积推广为矩阵内积, 并且把 $\geq\mathbf{0}$ 的约束推广为矩阵的半正定性 $\succeq 0$. 同时, 为了保证对称性, 我们要求涉及的矩阵都是对称矩阵 (在目标函数和约束都是线性的情况下没有影响), 由此, 我们得到半定规划问题的标准形式

$$\begin{aligned}
\min \quad & \mathbf{C}\bullet\mathbf{X}\\
\text{s.t. } \forall j\in[m], \quad & \mathbf{A_j}\bullet\mathbf{X} = b_j\\
& \mathbf{X}\succeq 0,
\end{aligned}\tag{2.3.1}$$

其中, $\mathbf{A_1},\cdots,\mathbf{A_m}, \mathbf{C}, \mathbf{X}\in\mathbb{S}^n$.

另一个常见的半定规划标准形式为

$$\begin{aligned}
\max \quad & \mathbf{b}^\top\mathbf{y}\\
\text{s.t.} \quad & \textstyle\sum_{i=1}^m y_i\mathbf{A_i}\preceq\mathbf{C},
\end{aligned}\tag{2.3.2}$$

其中, $\mathbf{A_1},\cdots,\mathbf{A_m}, \mathbf{C}, \mathbf{X}\in\mathbb{S}^n$. 简便起见, 除非特别说明, 此后我们默认讨论的都是对称矩阵, 不再写这个条件.

我们在下一节会介绍这两个形式的关系 (互为对偶). 这里, 我们先讨论半定规划问题的一般性.

半定规划问题的可扩展性很强, 很多常见的问题都可以转化为半定规划问题.

首先, 如果我们取 $\mathbf{C} = -\mathbf{C}$, 可知问题也可写为最大化目标函数的形式

$$\begin{aligned} \max \quad & \mathbf{C} \bullet \mathbf{X} \\ \text{s.t. } \forall j \in [m], \quad & \mathbf{A_j} \bullet \mathbf{X} = b_j \\ & \mathbf{X} \succeq 0. \end{aligned}$$

其次, 半正定的约束使得我们得以包含不等式关系. 注意到, 通过引入新变量 t, $a \le b$ 等价于 $a + t^2 = b$. 因此, 对于不等式约束 $\mathbf{A}_j \bullet \mathbf{X} \le b_j$, 把 $\mathbf{A}_j$ 扩展两行两列得到

$$\mathbf{A}'_j = \begin{bmatrix} \mathbf{A}_j & \mathbf{0}_{n\times 1} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & 0 & 1 \\ \mathbf{0}_{1\times n} & 1 & 0 \end{bmatrix}.$$

此时, $\mathbf{A}'_j \bullet \mathbf{X}' = b_j$ 等价于 $\mathbf{A}_j \bullet \mathbf{X} \le b_j$. 因此, 如果能够求解标准形式的半定规划问题, 那么通过提升问题的维度, 也能求解等式、不等式约束混合的问题

$$\begin{aligned} \min \quad & \mathbf{C} \bullet \mathbf{X} \\ \text{s.t. } \forall j \in [m], \quad & \mathbf{A_j} \bullet \mathbf{X} \le b_j \\ \forall i \in [t], \quad & \mathbf{A_i} \bullet \mathbf{X} = b_i \\ & \mathbf{X} \succeq 0. \end{aligned} \tag{2.3.3}$$

最后, 常见的二次约束二次规划 (Quadratic Constrained Quadratic Programming) 问题也可以表示为半定规划问题. 标准的二次约束二次规问题为

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & \mathbf{x}^\top \mathbf{Q_0} \mathbf{x} + \mathbf{q_0}^\top \mathbf{x} + c_0 - \theta \le 0 \\ \forall i \in [m], \quad & \mathbf{x}^\top \mathbf{Q_i} \mathbf{x} + \mathbf{q_i}^\top \mathbf{x} + c_i \le 0, \end{aligned} \tag{2.3.4}$$

其中, $\mathbf{Q_i} \succeq 0$. 由于 $\mathbf{Q_i} \succeq 0$, 我们可以做分解 $\mathbf{Q_i} = \mathbf{M_i}^\top \mathbf{M_i}$. 因此,

$$\begin{pmatrix} \mathbf{I} & \mathbf{M_i x} \\ \mathbf{x}^\top \mathbf{M_i}^\top & -c_i - \mathbf{q_i}^\top \mathbf{x} \end{pmatrix} \succeq 0 \quad \Leftrightarrow \quad \mathbf{x}^\top \mathbf{Q_i} \mathbf{x} + \mathbf{q_i}^\top \mathbf{x} + c_i \le 0.$$

利用这个方法, 我们可以把二次约束二次规划问题转化为 $2n$ 维, $m + 1$ 个约束的半定规划问题.

2.3.2 半定规划的对偶理论

半定规划算法的对偶理论研究已经相当成熟. 我们这里只对对偶理论做最基础的介绍, 以满足之后讨论的需要. 对于更多对偶理论的成果, 可以参考 [9] 第二章.

我们首先介绍普通的对偶理论 (参考了 [10]). 对于一般的约束优化问题

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & c_i(\mathbf{x}) \le 0, i \in \mathcal{I}, |\mathcal{I}| = m \\ & c_i(\mathbf{x}) = 0, i \in \mathcal{E}, |\mathcal{E}| = p, \end{aligned}$$

其中 $\mathbf{x} \in \mathbb{R}^n$, 设其最优值为 p^*. 其定义域为

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \mid c_i(\mathbf{x}) \le 0, i \in \mathcal{I} \text{ 且} c_i(\mathbf{x}) = 0, i \in \mathcal{E}\}.$$

其拉格朗日函数 $L : \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p \to \mathbb{R}$ 定义为

$$L(\mathbf{x}, \lambda, \nu) = f(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i c_i(\mathbf{x}) + \sum_{i \in \mathcal{E}} \nu_i c_i(\mathbf{x}).$$

其拉格朗日对偶函数 $g : \mathbb{R}^m_+ \times \mathbb{R}^p \to \mathbb{R}$ 定义为

$$\begin{aligned} g(\lambda, \nu) &= \inf_{\mathbf{x} \in \mathbb{R}^n} L(\mathbf{x}, \lambda, \nu) \\ &= \inf_{\mathbf{x} \in \mathbb{R}^n} \left(f(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i c_i(\mathbf{x}) + \sum_{i \in \mathcal{E}} \nu_i c_i(\mathbf{x}) \right). \end{aligned}$$

下面我们看到, 对偶函数的一个重要作用是给出原问题的下界.

定理 2.3.1. *弱对偶定理 (参考 [11])*

若 $\lambda \ge \mathbf{0}$, 则 $g(\lambda, \nu) \le p^*$.

证明. 若 $\tilde{\mathbf{x}} \in \mathcal{X}$, 则

$$g(\lambda, \nu) = \inf_{\mathbf{x}} L(\mathbf{x}, \lambda, \nu) \le L(\tilde{\mathbf{x}}, \lambda, \nu) \le f(\tilde{\mathbf{x}}).$$

对 $\tilde{\mathbf{x}}$ 取下界得

$$g(\lambda, \nu) \le \inf_{\tilde{\mathbf{x}} \in \mathcal{X}} f(\tilde{\mathbf{x}}) = p^*.$$

□

根据弱对偶定理, $\sup_{\lambda \ge \mathbf{0}, \mu} g(\lambda, \mu) = \sup_{\lambda \ge \mathbf{0}, \mu} \inf_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu)$ 给出了原问题的一个下界, 这个问题称为原问题的对偶问题, λ, μ 称为对偶变量.

假设对偶问题的最优值为 q^*, 则 $q^* \le p^*$, $p^* - q^* \ge 0$ 称为对偶间隙. 对偶问题和原问题的解相等当且仅当对偶间隙为 0, 此时亦称问题满足强对偶性. 实际上, 可以证明, 线性规划问题的对偶间隙必然为 0, 而半定规划问题的对偶间隙可以任意大.

不过, 如果想要对半定规划问题导出对偶, 我们还需要先对广义不等式 $\succeq$ 引入对偶理论. 我们的思路仍然是定义乘子. 但是, 由于我们的不等式是在矩阵空间定义的, 因此下面的运算推广为矩阵内积.

线性空间的子集 K 称为锥, 若其对伸缩变换封闭, 即对任意 $\mathbf{C} \in K$, 任意 $\alpha > 0$, $\alpha\mathbf{C} \in K$. 对于锥 K, 定义其在内积 $\bullet$ 下的对偶锥为

$$K^* \triangleq \{\mathbf{B} : \mathbf{B} \bullet \mathbf{C} \geq 0, \forall \mathbf{C} \in K\}.$$

同时, 如果锥 K 在内积 $\bullet$ 下满足 $K^* = K$, 则称其在内积 $\bullet$ 下自对偶.

注意到, 非负向量空间 $\mathbb{R}^n_+ = \{\mathbf{x} : \mathbf{x} \geq \mathbf{0}\}$ 是一个锥, 且在向量内积下自对偶. 对称半正定矩阵空间 $\mathbb{S}^n_+$ 也是一个锥, 且在矩阵内积下自对偶. 这说明, ≥ 0 和 $\geq \mathbf{0}$ 具有对偶意义上的相似性, 揭示了半定规划作为矩阵空间中线性规划的推广的深层原因.

由此, 我们导出半定规划问题2.3.1的对偶形式. 其拉格朗日函数为

$$L(\mathbf{X}, \mathbf{y}, \mathbf{S}) = \mathbf{C} \bullet \mathbf{X} - \sum_{i=1}^{m} y_i \left(\mathbf{A_i} \bullet \mathbf{X} - b_i\right) - \mathbf{S} \bullet \mathbf{X}, \quad \mathbf{S} \geq 0,$$

其对偶函数为

$$g(\mathbf{y}, \mathbf{S}) = \inf_{\mathbf{X}} L(\mathbf{X}, \mathbf{y}, \mathbf{S}) = \begin{cases} \mathbf{b^T y}, & \sum_{i=1}^m y_i \mathbf{A_i} - \mathbf{C} + \mathbf{S} = 0, \\ -\infty, & \text{其他}. \end{cases}$$

因此, 我们得到了对偶问题

$$\begin{aligned} \max_{\mathbf{y} \in \mathbb{R}^m} \quad & \mathbf{b}^\top \mathbf{y} \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i \mathbf{A_i} - \mathbf{C} + \mathbf{S} = 0 \\ & \mathbf{S} \geq 0, \end{aligned}$$

恰好就是2.3.2的形式.

我们还可以计算问题的对偶间隙. 假设原问题的最优解为 $\tilde{\mathbf{X}}$, 对偶问题的最优解为 $\tilde{\mathbf{y}}$, 则 $p^* - d^* = \mathbf{C} \bullet \tilde{\mathbf{X}} - \mathbf{b} \cdot \tilde{\mathbf{y}} = \mathbf{C} \bullet \tilde{\mathbf{X}} - \sum_{i=1}^m y_i \mathbf{A_i} \bullet \tilde{\mathbf{X}} = (\mathbf{C} - \sum_{i=1}^m y_i \mathbf{A_i}) \bullet \tilde{\mathbf{X}} = \tilde{\mathbf{S}} \bullet \tilde{\mathbf{X}}$. 由于 $\tilde{X}$ 和 $\tilde{S}$ 均为半正定矩阵, 有 $\tilde{\mathbf{S}} \bullet \tilde{\mathbf{X}} \geq 0$.

2.4 凸集与凸函数

对于有限维 Hilbert 空间 (含内积的完备向量空间) 中的非空集合 C, 如果对其中任意两点 $\mathbf{x}, \mathbf{y}$, 任意参数 $\alpha \in [0, 1]$, 有 $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in C$, 则称 C 为一个凸集.

凸集的一个重要性质是凸集分离定理.

定理 2.4.1. *凸集分离定理 (参考 [12])*

1. 对于空间中的两个至多交于一点的凸集 C_1, C_2, 存在超平面 $\mathbf{a}^\top \mathbf{x} = b$ 分离 C_1, C_2, 即 $\forall x \in C_1$, $\mathbf{a}^\top \mathbf{x} \geq b$, $\forall x \in C_2$, $\mathbf{a}^\top \mathbf{x} < b$.

2. 对于空间中的凸集 C 和任意向量 $\mathbf{x}$, 要么 $\mathbf{x} \in C$, 要么存在超平面分离 $\mathbf{x}$ 和 C, 分离的定义同上.

凸集分离定理提供了向量 $\mathbf{x}$ 是否属于 C 的一种二择一性质, 是凸优化问题的经典理论之一, 在优化问题的最优性条件、KKT 条件等性质的导出中占据核心作用.

设 C 是 $\mathbb{R}^n$ 中的凸集, 称函数 $f : C \to \mathbb{R}$ 为 C 上的凸函数, 若 $\lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \geq f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}), \forall \lambda \in [0,1]$. 一个等价的定义是, 函数 f 的上方图 $\{(\mathbf{x}, z) : \mathbf{x} \in C, z \geq f(\mathbf{x})\}$ 是一个凸集.

凸函数决定了函数微分的一些性质.

定理 2.4.2. 凸函数与微分

1. 如果 f 连续可微, 则 f 凸当且仅当 $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot \mathbf{y} - \mathbf{x}, \forall \mathbf{x}, \mathbf{y} \in C$.
2. 若 f 二阶连续可微, 则 f 凸当且仅当 $\nabla^2 f(\mathbf{x}) \geq 0, \forall \mathbf{x} \in C$.

与对偶理论类似, 我们可以把凸函数的概念推广到对称矩阵空间中. 我们定义一个函数 $\mathbf{g} : C \to \mathbb{S}^n$ 是矩阵凸函数, 若对任意 $\mathbf{x}, \mathbf{y} \in C$, $\lambda \mathbf{g}(\mathbf{x}) + (1-\lambda)\mathbf{g}(\mathbf{y}) \geq \mathbf{g}(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}), \forall \lambda \in [0,1]$.

第三章　半定规划问题的算法

3.1　半定规划算法简述

对于半定规划问题, 我们只讨论近似算法, 因为半定规划问题可能不存在有理数解甚至代数数解, 无法使用确定数量的比特表示精确解. 例如, $x^2 = 2$ 就是只有无理数解的半定规划问题. 一般使用参数 ϵ 来度量算法的近似程度.

由于本文章篇幅有限, 不能详述半定规划问题的所有算法. 笔者在这里大致列出一个目前基于各种方法的半定规划算法的概述和参考文献列表, 供读者参考.

- **一阶方法** 即基于梯度的方法, 算法的复杂度与问题的近似度关系为 $\mathrm{poly}(1/\epsilon)$, 收敛较慢, 但单步求解很快.
  - **乘法加权更新算法** 这是 Arora 和 Kale[13]提出的基于原始对偶方法的算法. 该算法在半定规划问题的量子算法中应用广泛, 参考 Huang 等人[14]的概述.
  - **增广拉格朗日方法** 这种方法基于约束规划问题中常用的增广拉格朗日方法. 一般和锥优化的方法结合, 如基于交替方向乘子法的 SDPNAL[15], SDPNAL+[16], , 基于条件梯度法的 CGAL[17]等.
- **二阶方法** 即基于 Hessian 矩阵的方法, 算法的复杂度与问题的近似度关系为 $\mathrm{poly}(\log(1/\epsilon))$, 收敛很快, 但单步求解较慢.
  - **切平面法** 这种方法的思路是在每步迭代中利用分离神谕将定义域分割成两部分, 并且判定最优解在哪一部分, 以逐渐缩小最优解的可能区域.
    目前最优的算法的复杂度为 $m\left(mn^2 + m^2 + n^\omega\right)$[18].
    参考文献 [1], [19], [20], [21], [18].
  - **内点法** 这种方法的思路是利用障碍函数定义一系列优化问题, 构造一条趋向最优解的中心路径, 并且利用障碍函数将迭代过程限制在中心路径附近.
    目前最优的算法的复杂度为 $\sqrt{n}\left(mn^2 + m^2 + n^\omega\right)$[22].
    参考文献 [3], [4], [5], [23], [22].
- **其它方法** 例如, 把问题转化为非光滑优化问题, 然后使用谱束法 (Spectral Bundle Method)[24]. 利用低复杂度运算加速求解的近似方法[25]等.

半定规划问题是一类很有代表性的凸规划问题, 因此上述方法往往也适用于其它类似问题, 具有很高的理论价值. 但需要注意的是, 半定规划问题的算法在理论和实践上差别是比较大的. 首先, 很多半定规划问题的算法是启发式算法, 如 SDPNAL+[16], 除

了必要的收敛性保证以外缺乏复杂度分析的结果, 和其它算法的比较基本只能基于数值实验进行. 其次, 半定规划问题的评价指标非常多元, 从终止准则, 近似解是否能够突破约束再到测试样例的类型都有不同的理论和实验. 最后, 在面对特定的算例时, 各种算法仍然有很大的优化空间, 理论结果也不能完全预测, 例如, 半定规划的一种测试样例是基于最大割问题生成的 (我们在第四章中给出了其形式4.2.2), 但是其输入是非常稀疏的, 利用稀疏矩阵的若干技巧, 问题有很大的优化空间. 近年来, 也有一部分文章分析了问题对稀疏形式输入的效果, 如 [18,22] 等, 但是仍然不能完全拟合实验结果.

随后, 我们按照历史顺序, 在四个小节中介绍四个半定规划的有代表性的算法:

- **椭球法** 二阶切平面方法, 半定规划问题的第一个多项式近似算法, 具有重要的理论价值.
- **内点法** 二阶内点方法, 半定规划问题的第一个实用的多项式近似算法, 至今仍是最主流的半定规划求解算法.
- **乘法加权更新算法** 一阶方法, 结构比较特殊的半定规划算法, 在半定规划问题的量子算法设计里起着核心作用.
- **sketchyCGAL 算法** 一阶近似方法, 近年提出的算法, 融合了大量低秩近似的技巧, 提出了一个牺牲精度但效率极大提高的算法, 在大规模求解上有很高的潜力.

其中, 前三个算法都是多项式时间近似算法, 最后一个算法是比较特殊的低秩近似算法. 这些算法的理论和实际上的效率保证均随着历史发展逐渐提高.

3.2 椭球法

3.2.1 简述

本节的组织参考了 [26].

Khachiyan[1]在 1979 年提出椭球法, 并用它为线性规划问题给出了第一个多项式时间的精确算法. 很快, Grötschel, Lovász 和 Schrijver[2]在 1981 年将其推广, 给出了半定规划问题的第一个多项式时间近似算法.

3.2.2 算法原理

椭球法的基本思路来源于凸集分离定理 (2.4.1). 在给定的连续紧凸集 $K \subset \mathbb{R}^n$ 上, 文献 [2] 考虑了下面两种问题:

定义 3.2.1. 强优化问题

给定 $\mathbf{c} \in \mathbb{R}^n$, 求 $\mathbf{x} = \arg\max_{\mathbf{x}\in K} \mathbf{c} \cdot \mathbf{x}$.

定义 3.2.2. 强分离问题

给定 $\mathbf{x}$, 判定 $\mathbf{x}$ 是否属于 K, 若否, 给出 $\mathbf{x}$ 和 K 的分离超平面 $\mathbf{a} \cdot \mathbf{x} = \mathbf{b}$.

但是, 大量的优化问题 (包括半定规划) 不存在精确解, 我们往往只考虑近似解. 因此, 文献 [2] 又定义了近似的版本:

定义 3.2.3. 弱优化问题

给定 $\mathbf{c} \in \mathbb{R}^n$ 和近似参数 $\epsilon > 0$, 求 $\hat{\mathbf{x}}$, 使得 $d(\mathbf{x}, K) \leq \epsilon$, 且 $\mathbf{x}$ 是 $\mathbf{c} \cdot \mathbf{x}$ 的 ϵ-近似最优解, 即对任意 $\mathbf{y} \in K$, $\mathbf{c} \cdot \mathbf{y} \leq \mathbf{c} \cdot \mathbf{x} + \epsilon$.

定义 3.2.4. 弱分离问题

给定 $\mathbf{x}$ 和近似参数 $\delta > 0$, 要么断言 $d(\mathbf{x}, K) \leq \delta$, 要么给出 $\mathbf{x}$ 和 K 的近似分离超平面 $\mathbf{a}$, 使得 $\|a\| \geq 1$, 且对任意 $\mathbf{y} \in K$, $\mathbf{a} \cdot \mathbf{y} \leq \mathbf{a} \cdot \mathbf{x} + \delta$.

随后, 文献 [2] 提出了" 凸体" (convex body) 的概念:

定义 3.2.5. 凸体

定义凸体为一个五元组 $(K, n, \mathbf{a_0}, r, R)$, 其中 $n \geq 2, K \subseteq \mathbb{R}^n$ 为凸集, $\mathbf{a_0} \in K$, 且:

$$B(\mathbf{a_0}, r) \subseteq K \subseteq B(\mathbf{a_0}, R),$$

其中 $B(\mathbf{p}, r)$ 为一个半径为 r, 球心为 $\mathbf{p}$ 的闭球.

亦即, 要求凸集 K 被夹在两个 n 维球中间.

文献 [2] 证明, 对任意的定义域 K 为凸体的问题, 弱优化问题和弱分离问题的多项式算法的存在性是等价的. 如果其中一个问题一个有多项式算法, 则另一个也有. 因此, 我们现在要证明可以多项式求解弱优化问题, 只要证明可以多项式求解弱分离问题.

实际上, 我们直接证明可以求解强分离问题. 考虑标准形式的半定规划问题 2.3.1, 注意到其可行域 K 是一个凸集. 因此, 对任意矩阵 $\mathbf{X}$ 和 K 的强分离问题, 只需要分别验证 $\mathbf{X}$ 是否满足下列条件

- $\mathbf{X}$ 是否对称. 用 $O(n^2)$ 时间比较 $\mathbf{X}$ 和 $\mathbf{X}^\top$ 的元素即可.
- $\mathbf{X}$ 是否半正定. 用 $O(n^3)$ 时间计算 $\mathbf{X}$ 的最小特征值 $\lambda_{\min}$(高斯消元) 和最小特征向量 $\mathbf{v}$(求解线性方程), 如果 $\lambda_{\min} \geq 0$, 则其为半正定矩阵; 否则, 有 $\mathbf{v}^\top \mathbf{X} \mathbf{v} = \lambda_{\min} \mathbf{v}^\top \mathbf{v} < 0$, 从而 $\mathbf{v}^\top \mathbf{X} \mathbf{v} = \operatorname{Tr} \mathbf{v}^\top \mathbf{X} \mathbf{v} = \mathbf{X} \bullet \mathbf{v}\mathbf{v}^\top < 0$, 但是对任意 $\mathbf{Y} \in \mathbb{S}_+^n$, $\mathbf{V} \bullet \mathbf{v}\mathbf{v}^\top \geq 0$, 因此 $vv^\top$ 就是把 $\mathbf{X}$ 和 K 分离的超平面.
- 用 $O(mn^2)$ 的时间计算 $\mathbf{A_i} \bullet \mathbf{X}$ 的值, 如果都等于 b_i, 则其属于 K; 否则, 使其不成立的 $\mathbf{A_i}$ 就是一个分离超平面.

综上, 我们证明了半定规划问题存在多项式算法.

为了更加细致地研究问题的计算复杂度, 我们下面给出 $\mathbb{R}^n$ 上椭球法的细节 (对于半定规划问题的椭球法是类似的, 把向量空间提升为矩阵空间, 向量内积改为矩阵内积

即可). 给定一个凸体 K 和上面的强分离问题的多项式算法 (简称强分离神谕), 我们希望求优化问题 $\min_{\mathbf{x}\in K} \mathbf{c}\cdot\mathbf{x}$ 的近似解.

我们已知集合 K 是凸体, 故 K 属于某个有界椭球. 我们的思路是利用分离问题带来的信息, 在保证最优解属于椭球的情况下, 逐渐缩小该椭球, 直到椭球的体积充分小, 即可得到问题的一个近似解.

令 $\mathbf{z}\in\mathbb{R}^n$, $\mathbf{A}\in\mathbb{S}_+^n$ 为一个正定矩阵, 其决定一个 n 维欧式空间中的椭球

$$\ell(z,A)=\left\{\mathbf{x}\in\mathbb{R}^n:(\mathbf{x}-\mathbf{z})^\top\mathbf{A}^{-1}(\mathbf{x}-\mathbf{z})\le 1\right\},$$

其中 $\mathbf{z}$ 是椭球的球心. 特别地, $\ell(z,I)$ is 就是球心为 $\mathbf{z}$ 的单位球.

下面描述椭球法的运算过程.

从凸体的定义中给定的包含 K 的椭球 $\ell(\mathbf{a_0},R\mathbf{I})$ 开始, 每次选择当前凸体的球心 $\mathbf{x}^k$, 对其计算分离问题, 如果其属于 K, 那么我们取超平面参数 $\mathbf{a}^k=\mathbf{c}$; 如果其不属于 K, 则询问分离神谕, 得到分离超平面 $\mathbf{d}\cdot\mathbf{x}=e$, 取超平面参数 $\mathbf{a}^k=\mathbf{d}$. 这样, $\mathbf{a}^k$ 总是定义了一个经过 $\mathbf{x}^k$ 的超平面, 优化问题的最优解必然在超平面的某一侧. 如果 $\mathbf{x}^k$ 属于 K, 那么 K 在超平面上方的点的目标函数值都大于 $\mathbf{x}^k\cdot\mathbf{c}$; 如果 $\mathbf{x}^k$ 不属于 K, 则根据分离超平面的定义, K 本身就都包含在超平面的一侧. 总之, 我们把最优解存在的区域缩小到了经过球心的" 半个" 椭球中. 这样, 我们把椭球更新为仅包含这半个椭球的最小椭球, 就能不断缩小椭球的体积, 迭代过程如图3.1所示.

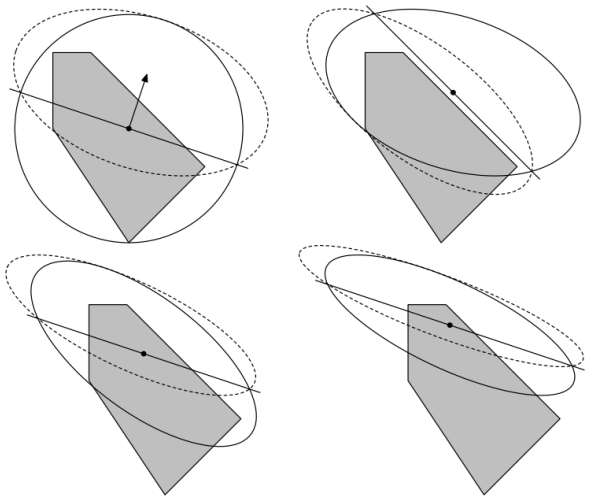

图 3.1　椭球法的四步迭代, 来源于 [26]

使用下面的引理, 我们可以证明算法的收敛性.

引理 3.2.1. *([26], 引理 2)*

对于椭球 $\ell(\mathbf{z},\mathbf{A})$ 和任意超平面 $H=\{\mathbf{x}\in\mathbb{R}^n:\mathbf{a}^\top\mathbf{x}\ge\mathbf{a}^\top\mathbf{z}\}$, 包含 $\ell(\mathbf{z},\mathbf{A})\cap H$ 的唯一最小椭球是 $\ell(\mathbf{z}',\mathbf{A}')$, 其中

$$\mathbf{z}' = \mathbf{z} + \frac{1}{n+1} \cdot \frac{\mathbf{Aa}}{\sqrt{\mathbf{a}^\top \mathbf{Aa}}}, \tag{3.2.1}$$

$$\mathbf{A}' = \frac{n^2}{n^2-1}\left(\mathbf{A} - \frac{2}{n+1} \cdot \frac{\mathbf{Aaa}^\top \mathbf{A}}{\mathbf{a}^\top \mathbf{Aa}}\right). \tag{3.2.2}$$

并且,

$$\frac{\mathrm{vol}(\ell(\mathbf{z}', \mathbf{A}'))}{\mathrm{vol}(\ell(\mathbf{z}, \mathbf{A}))} < e^{-1/(2n+2)}.$$

该性质保证了包含最优解 $\mathbf{z}^*$ 的椭球体积至少以固定比例减少, 从而保证了算法在多项式时间内收敛.

下面列出算法的详细步骤.

Algorithm 1 半定规划的椭球算法[26]

输入: 凸体 $(K, n, \mathbf{a_0}, r, R)$, $\mathbf{c} \in \mathbb{R}^n$, $\epsilon > 0$.

输出: ϵ 弱优化问题的解.

// 初始化

定义 $N = \left(2n^2 + 2n\right)\left\lceil \ln \frac{2R^2\|c\|}{r\varepsilon} \right\rceil$.

定义 $\mathbf{x_0} = \mathbf{a_0}, \mathbf{A_0} = R\mathbf{I}$.

$k = 0$, max=$\mathbf{c}^\top \mathbf{x_0}$ (当前可行点给出的最大目标函数值), **maxvec=**$\mathbf{x_0}$.

// 迭代

for $k < N$ **do**

　对 $\mathbf{x_k}$ 使用强分离神谕.

　if $\mathbf{x_k} \in K$ **then**

　　$\mathbf{a} = \mathbf{c}$.

　else

　　神谕输出分离超平面的法向量 $\mathbf{d}$.

　　$\mathbf{a} = -\mathbf{d}$.

　end if

　利用性质3.2.1, 基于 $\mathbf{x_k}, \mathbf{A_k}$ 定义的椭球和 $\mathbf{a}, \mathbf{x_k}$ 定义的超平面计算出 $\mathbf{x_{k+1}}, \mathbf{A_{k+1}}$.

　if $\mathbf{x_k} \in K$ 且 $\mathbf{c}^\top \mathbf{x_k} \geq$ max **then**

　　max=$\mathbf{c}^\top \mathbf{x_k}$, **maxvec=**$\mathbf{x_k}$.

　end if

end for

输出 **maxvec**.

定理 3.2.1. *([26], 第三节)*

椭球算法在多项式时间内收敛.

证明. 核心思想仍然是控制体积, 但是用到了凸集的性质和一些技巧.

根据凸体的定义, 以 $\mathbf{x_0}$ 为球心的球 $B(\mathbf{x_0}, r)$ 属于 K, 从而其 $n-1$ 维截面 $S = B(\mathbf{x_0}, r) \cap \{\mathbf{x} : \mathbf{c}^\top \mathbf{x} = \mathbf{c}^\top \mathbf{x_0}\}$ 也属于 K. 记问题的最优解为 $\mathbf{x}^* \in K$, 那么以 $\mathbf{x}^*$ 为顶, 以 S 为底的 n 维圆锥 C 也属于 K.

假设算法的输出是 $\hat{\mathbf{x}}$, 那么, 根据算法的过程, 一方面, 对于所有 $\mathbf{x_k} \in K$, 它们给出的目标函数值都小于等于 $\hat{\mathbf{x}}$ 的目标函数值; 另一方面, 对于所有 $\mathbf{x_k} \notin K$, 其给出的分离超平面都不影响 K 的内部. 总之, 我们知道, 圆锥 C 和半空间 $\{\mathbf{x} : \mathbf{c}^\top \mathbf{x} \geq \mathbf{c}^\top \hat{\mathbf{x}}\}$ 的交一定没有被分离过, 一直和最优解 x^* 留在算法的分离过程中 K 剩下的部分里.

由于圆锥 C 的底面的法向量正是 $\mathbf{c}$, 因此 $C \cap \{\mathbf{x} : \mathbf{c}^\top \hat{\mathbf{x}} \leq \mathbf{c}^\top \mathbf{x}\}$ 的体积可以表示为

$$\frac{\operatorname{vol} B_{n-1} r^{n-1} \left(c^\top x^* - c^\top x_0\right)}{n\|c\|} \left(\frac{c^\top x^* - c^\top \hat{x}}{c^\top x^* - c^\top x_0}\right)^n.$$

另一方面, 定理3.2.1控制了算法在第 N 步时确定 x^* 所在的区域, 即 $\ell(\mathbf{x_N}, \mathbf{A_N})$ 的体积上界. 我们有

$$\operatorname{vol}(\ell(\mathbf{x_N}, \mathbf{A_N})) \leq e^{-N/(2n+2)} R^n \operatorname{vol}(B_n).$$

由于前者包含于后者, 因此我们得到

$$\mathbf{c}^\top \mathbf{x}^* - \mathbf{c}^\top \hat{\mathbf{x}} \leq e^{-N/\left(2n^2+2n\right)} R \left(\frac{n \operatorname{vol}(B_n)}{\operatorname{vol}(B_{n-1})}\right)^{1/n} \left(\frac{\mathbf{c}^\top \mathbf{x}^* - \mathbf{c}^\top \mathbf{x_0}}{r}\right)^{\frac{n-1}{n}} \|c\|^{1/n}.$$

代入 $\mathbf{c}^\top \mathbf{x}^* - \mathbf{c}^\top \mathbf{x_0} = \mathbf{c}^\top \left(\mathbf{x}^* - \mathbf{x_0}\right) \leq \|\mathbf{c}\| \, \|\mathbf{x}^* - \mathbf{x_0}\| \leq R\|\mathbf{c}\|$, 即可证明 $0 \leq \mathbf{c}^\top \mathbf{x}^* - \mathbf{c}^\top \hat{\mathbf{x}} \leq \epsilon$, 从而 $\hat{\mathbf{x}}$ 是一个 ϵ-近似解. □

当然, 实际上算法在运算过程中的精度是有限的, 不可能完全精确地存储. 为了对抗舍入误差, 可以稍微扩大算法中迭代计算的椭球体积, 最后得到类似的结论, 详见 [26] 第四节.

3.2.3 总结

椭球法最大的问题是时间复杂度太高. 根据3.2.1, 算法需要迭代 $O(n^2)$ 次, 但是, 每次迭代都需要使用一次强分离神谕, 根据我们先前的断言, 这需要计算 $O((m+n)n^2)$ 次. 总体的计算复杂度为 $O((m+n)n^4)$, 虽然使用更加先进的方法可以略微降低复杂度, 但是与实践要求比起来依然太高. 而且, 在迭代过程中, 椭球法需要存储椭球的系数矩阵, 因此空间复杂度为 $O(n^2)$, 对于大规模问题而言还是过大.

3.3　内点法

3.3.1　简述

1984 年, Karmarkar 提出了线性规划的内点算法, 虽然应用上仍然不如单纯形法, 但是是第一个比较实用的多项式保证的算法. 1988 年, Nesterov 和 Nemirovsky[27]提出了自协调障碍函数的概念, 为内点算法应用到一般的凸优化问题中打下了基础. 1992 左右年, Alizadeh[3], Kamath 和 Karmarkar[4-5]等人给出了半定规划问题的第一个多项式时间内点算法. 随后, 关于内点算法的研究呈井喷式发展, 大量高效的内点算法被提出, 其中最有代表性的是 Vandenberghe 和 Boyd 在 1996 年撰写的综述[23], 系统地总结了内点法的理论和应用.

3.3.2　算法原理

内点算法的原理是, 在可行域内从一初始点出发, 从内部沿着一条可行点的迭代路径逼近最优解. 在内点法提出后, 半定规划问题逐渐进入了人们的视野, 人们基于半定规划发展了大量的应用.

我们只以 [23] 为基础给出内点算法的一个例子, 以研究其基本思想. 实际上, 与常见的基于优化的搜索算法类似, 内点算法有大量的变种. 例如, 使用不同的搜索方向, 距离函数等等. 要详细了解各种变种, 可以参考 [28], [29] 等综述.

为了保证在迭代过程中不会移动到可行域外, 一个通用的办法是增加一个障碍函数, 让可行域外的目标函数值非常大, 从而防止迭代移动到可行域外. 常见的罚函数有倒数障碍函数和对数障碍函数. 这两种函数都使得函数值在接近定义域边界时急剧增长, 迫使迭代不会越过边界. 1988 年, Nesterov 和 Nemirovsky[30]提出了自协调障碍函数的概念, 并证明, 对于任何凸优化问题, 只要能够找到一个自协调障碍函数, 就能用内点法给出多项式时间近似算法.

很快, 人们发现了半定规划问题2.3.1和2.3.2的自协调障碍函数, 即 $-\log\det(\mathbf{X})$ 和 $\phi(\mathbf{y}) = -\log\det(-\sum_{i=1}^{m} y_i\mathbf{A}_i + \mathbf{C})$. 下面, 我们简记 $\mathbf{A}(\mathbf{y}) = -\sum_{i=1}^{m} y_i\mathbf{A}_i + \mathbf{C}$.

下面的定理给出了上述自协调障碍函数的导数.

定理 3.3.1. *([23], 4.1 节)*

$$
\begin{aligned}
(\log\det(\mathbf{X}))' &= (\mathbf{X}^{-1})^{\top},\\
(\nabla\phi(\mathbf{y}))_i &= -\operatorname{Tr}\mathbf{A}(\mathbf{y})^{-1}\mathbf{A}_i = -\operatorname{Tr}\mathbf{A}(\mathbf{y})^{-1/2}\mathbf{A_i}\mathbf{A}(\mathbf{y})^{-1/2},\\
\left(\nabla^2\phi(\mathbf{y})\right)_{ij} &= \operatorname{Tr}\mathbf{A}(\mathbf{y})^{-1}\mathbf{A_i}\mathbf{A}(\mathbf{y})^{-1}\mathbf{A_j}\\
&= \operatorname{Tr}(\mathbf{A}(\mathbf{y})^{-1/2}\mathbf{A_i}(\mathbf{A}(\mathbf{y})^{-1/2})(\mathbf{A}(\mathbf{y})^{-1/2}\mathbf{A_j}(\mathbf{A}(\mathbf{y})^{-1/2}).
\end{aligned}
$$

与常见的障碍函数类似, 对于问题的定义域 $\mathbf{X} \succeq 0$ 而言, $\mathbf{X}$ 接近边界意味着其有特

征值接近 0, 从而导致其行列式接近 0. 此时, $\log\det(\mathbf{X})$ 趋于无穷大, 保证了该障碍函数不会越界. 图3.2为该函数在某个半定规划问题上的等高线图示.

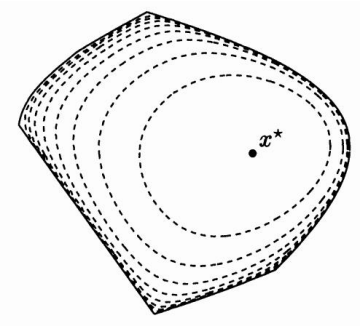

图 3.2　等高线图, 来源于 [23]

定义了障碍函数后, 最简单的方法就是利用加权的障碍函数直接构造迭代过程. 假设我们为障碍函数加权 μ, 并且在每次迭代中以上次迭代的结果为初始点, 近似求解下面的问题 (一般使用梯度下降 + 线搜索)

$$\min_{\mathbf{X}} \{\mathbf{C}\bullet\mathbf{X} - \mu\log\det(\mathbf{X}) : \mathbf{A_i}\bullet\mathbf{X} = b_i (i = 1, \ldots, m)\}.$$

理想状况下, 若每一步都能精确求解该问题, 并且在迭代中令 μ 逐渐趋于 0, 我们就能得到一系列在定义域内逐渐逼近最优解的解. 这实质上相当于沿着一条 μ 趋近于 0 的路径逼近最优值点, 这条关于 μ 的路径称为迭代的中心路径.

这种方法的研究可以参考 [31-33] 等工作, 其主要缺点在于, 需要谨慎设置迭代步长, 因为步长过大时可能会导致迭代越过定义域的边界, 导致算法无法收敛. 为了防止该现象, 人们提出可以在迭代时同时求解原始问题和对偶问题. 这样, 由于原始、对偶问题相互控制界限和最优性条件, 我们可以避免越界情况的发生. 这种方法称为原始-对偶内点法.

在第二章中, 我们讨论了半定规划问题的对偶理论和对偶间隙. 半定规划的原始-对偶内点法的基本思想就是同时对原始问题和对偶问题做迭代, 得到一系列原始-对偶解 $(\mathbf{X}^k, \mathbf{y}^k)$, 并且逐渐减小其对偶间隙. 迭代中, 原始问题对应的目标函数值为 $p^k = \mathbf{C}\bullet\mathbf{X}^k$, 而对偶问题对应的目标函数值为 $d^k = \mathbf{b}\cdot\mathbf{y}^k$. 定义其差为当前的对偶间隙. 因此, 当对偶间隙充分小时, 根据对偶原理, 我们得到的就是原问题的一组近似解. 例如, 假设算法的终止准则设置为对偶间隙小于等于 ϵ, 即 $p^k \le d^k + \epsilon$. 根据对偶理论, 原始问题的最优解 p^* 大于等于对偶问题的最优解 d^*. 由于迭代过程中得到的均为可行解, 我们有 $p^k \ge p^* \ge d^* \ge d^k$. 由 $p^k \le d^k + \epsilon$, 显然有 $p^k \le p^* + \epsilon$, 因此 $\mathbf{X^k}$ 必然是原问题的 ϵ-近似解.

原始-对偶内点法需要求解下面的问题

$$\min_{\mathbf{X},\mathbf{y},\mathbf{S}} \left\{ \mathbf{X} \bullet \mathbf{S} - \mu \log\det(\mathbf{XS}) : \mathrm{Tr}\,(\mathbf{A_i X}) = b_i (i = 1, \ldots, m), \sum_{i=1}^{m} y_i \mathbf{A_i} + \mathbf{S} = \mathbf{C} \right\},$$

其中, 根据第2.3.2节的讨论, $\mathbf{X} \bullet \mathbf{S}$ 是原始问题和对偶问题的对偶间隙. $-\mu \log\det(XS)$ 是加权的障碍函数. 剩余部分则是原始和对偶问题的约束. 而且, 使用一阶最优性条件, 我们还有 $\mathbf{X} \bullet \mathbf{S} = \mu I$.

每一步迭代中, 我们使用梯度下降的方法. 假设当前的解是 $\mathbf{X}, \mathbf{y}, \mathbf{S}$, 我们现在需要找到梯度 $\delta\mathbf{X}, \delta\mathbf{y}, \delta\mathbf{S}$, 使得上述约束规范条件在梯度下降后仍然成立, 也就是

$$\begin{aligned} \mathbf{X} + \delta\mathbf{X} \succeq 0, \mathbf{S} + \delta\mathbf{S} \succeq 0, \\ \mathbf{A_i} \bullet \delta\mathbf{X} = 0, \quad i = 1, \ldots, m, \\ \sum_{i=1}^{m} \delta y_i \mathbf{A_i} + \delta\mathbf{S} = 0, \\ (\mathbf{X} + \delta\mathbf{X})(\mathbf{S} + \delta\mathbf{S}) = \mu I. \end{aligned} \tag{3.3.3}$$

注意到, 最后一个方程关于 $\delta\mathbf{X}$ 和 $\delta\mathbf{S}$ 不是线性的, 因此, 不同的原始-对偶内点法的不同之处在于利用不同的方法将其改写为线性的. 例如, 张寅[34]提出, 可以将其更改为

$$\mathbf{H_P}(\delta\mathbf{XS} + \mathbf{X}\delta\mathbf{S}) = \mu\mathbf{I} - \mathbf{H_P}(\mathbf{XS}),$$

其中 $\mathbf{H_P}$ 是线性变换

$$\mathbf{H_P}(\mathbf{M}) := \frac{1}{2}\left[\mathbf{PMP^{-1}} + \mathbf{P}^{-\top}\mathbf{M}^{\top}\mathbf{P}^{\top}\right].$$

文献 [34] 还指出, 历史上的很多算法本质上是选取了不同的 $\mathbf{P}$, 如 $\mathbf{I}$[35], $\mathbf{S^{1/2}}$[36], $\mathbf{X^{-1/2}}$[37].

结合上面给出的迭代方向计算方法, 即可得到标准的原始对偶内点法的一般框架.

Algorithm 2 半定规划的原始对偶内点算法[23]

输入: $\mathbf{C}, \mathbf{A_j}, b_j, \epsilon$.

输出: 问题的 ϵ 近似解.

// 初始化

1: 计算问题的一个初始解 $\mathbf{X_0}, \mathbf{y_0}$, 进而计算 $\mathbf{S_0}$, 初始化 $\gamma_0, k = 0$.

// 迭代

2: **while** $\mathbf{S_k} \bullet \mathbf{X_k} \geq \epsilon$ **do**

3: 以某方式递减 γ_k 得到 γ_{k+1}.

4: 代入 $\mathbf{X_k}, \mathbf{S_k}, \mathbf{y_k}, \gamma_{\mathbf{k+1}}$, 求解3.3.3给出的方程, 得到下降方向 $\delta\mathbf{X}, \delta\mathbf{S}$.

5: 计算 $\mathbf{X_{k+1}} = \mathbf{X_k} + \delta\mathbf{X}, \mathbf{S_{k+1}} = \mathbf{S_k} + \delta\mathbf{S}$, 进而计算 $\mathbf{y_{k+1}}$.

6: $k = k + 1$.

7: **end while**

8: 输出 $\mathbf{X_k}, \mathbf{y_k}$.

下面, 我们讨论如何分析算法的收敛性. 当然, 算法的每一步并不是准确地沿着中心路径移动. 因此, 为了度量当前迭代点和中心路径的距离, 我们需要引入势函数

$$\psi(\mathbf{y}, \mathbf{X}) \triangleq -\log\det \mathbf{SX} + n\log \mathbf{S} \bullet \mathbf{X} - n\log n.$$

这样, 我们只需要证明势函数的值和权重 μ 同时随着迭代在多项式时间内达到指定精度即可. 文献 [23] 详细地描述了如何构造辅助函数来证明算法在 $O(\sqrt{n})$ 步内即可终止, 又因为迭代时只需要求解关于 $\delta\mathbf{X}, \delta\mathbf{S}$ 的线性方程组, 复杂度在 $O(mn^2)$ 左右, 因此算法的计算复杂度降低到 $O(mn^{2.5})$ 左右, 理论效率比椭球法提高了不少.

3.3.3 总结

与椭球法相比, 内点法的优点在于既有理论保证, 实践上效果也很好. 因此, 在应用中, 内点法的收敛速度往往非常快, 基本在常数步左右就能结束 (参考 [23]), 因此成为了半定规划问题的应用最广泛的算法, 直到现在仍在大量使用.

3.4 乘法加权更新算法

3.4.1 简述

2007 年, Arora 和 Kale[13]提出了半定规划问题的乘法加权更新算法 (Multiplicative Weight Update Algorithm). 严格来说, 乘法加权更新算法是一类预测论中算法的统称, Arora 和 Kale 使用了该思想, 但是他们对算法的命名是原始-对偶法. 为了与原始-对偶内点法做区别, 因此这里称为乘法加权更新法.

严格来说, 乘法加权更新算法并不是一个完整的算法. 它依赖于在迭代过程中求解一个神谕, 而作者并没有说明如何求解该神谕. 但是, 在近年兴起的量子计算中, 人们发现可以使用量子计算的 Gibbs 采样方法求解该神谕, 并提出了半定规划问题的一系列量子算法 [38-39].

3.4.2 算法原理

乘法加权更新算法的背景是预测论中的一个著名的问题, 即根据多位专家的建议进行动态决策, 并且最大化自己的收益. 这种范式最初是在 De Santis 等人的开创性论文中作为在线学习模型引入的. 在 20 世纪 80 年代末至 90 年代初, 人们开始对其进行深入研究. 现在, 预测论在信息论和优化等领域得到广泛应用. 读者可以阅读教材[40] 以进一步了解预测理论.

现在, 考虑一个矩阵博弈. 假设 $\mathbb{S}^{n-1}$ ($\mathbb{R}^n$ 中的单位球) 中的每个向量 $\mathbf{v}$ 都关联了一个专家. 在每一轮博弈中, 每个专家都会推荐一个行动. 我们的任务是使用某种随机策略选择某个专家 $\mathbf{v} \in \mathbb{S}^{n-1}$ 并遵循他建议的行动. 选择完成后, 博弈会给出损失矩阵 $\mathbf{M} \in \mathbb{R}^{n\times n}$, 专家 $\mathbf{v}$ 的损失为 $\mathbf{v}^\top \mathbf{M}\mathbf{v}$. 我们假设所有这些损失要么在 $[0,1]$ 范围内, 要么在 $[-1,0]$ 范围内. 等价地, 我们要求矩阵 $\mathbf{M}$ 满足约束 $\mathbf{0} \preceq \mathbf{M} \preceq \mathbf{I}$ 或 $-\mathbf{I} \preceq \mathbf{M} \preceq \mathbf{0}$.

现在, 我们给出该问题的数学模型. 记 $t = 1, 2, \ldots, T$ 为博弈的轮次. 在第 t 轮中, 我们在专家集 $\mathbb{S}^{n-1}$ 上选择分布 $\mathcal{D}^{(t)}$, 从中随机抽取专家 $\mathbf{v}$, 使用该专家建议的行动方案. 那么, 如果损失矩阵是 $\mathbf{M}^{(t)}$, 则选择分布 $\mathcal{D}^{(t)}$ 的期望损失为

$$\mathop{\mathbb{E}}_{\mathbf{v}\in\mathcal{D}^{(t)}} \left[\mathbf{v}^\top \mathbf{M}^{(t)} \mathbf{v}\right] = \mathop{\mathbb{E}}_{\mathbf{v}\in\mathcal{D}^{(t)}} \left[\mathbf{M}^{(t)} \bullet \mathbf{v}\mathbf{v}^\top\right] = \mathbf{M}^{(t)} \bullet \mathop{\mathbb{E}}_{\mathbf{v}\in\mathcal{D}^{(t)}} \left[\mathbf{v}\mathbf{v}^\top\right].$$

定义矩阵 $\mathbf{P}^{(t)} = \mathbb{E}_{\mathbf{v}\in\mathcal{D}^{(t)}} [\mathbf{v}\mathbf{v}^\top]$. 注意到 $\mathbf{P}^{(t)}$ 是半正定矩阵 $\mathbf{v}\mathbf{v}^\top$ 的凸组合, 因此是半正定的. 并且, 因为对于所有 $\mathbf{v}$, 我们有 $\mathrm{Tr}\,(\mathbf{v}\mathbf{v}^\top) = \|\mathbf{v}\|^2 = 1$, 我们有 $\mathrm{Tr}\left(\mathbf{P}^{(t)}\right) = 1$. 我们把满足这样性质 (半正定且迹为 1) 的矩阵称为密度矩阵.

由于我们只对算法的期望损失感兴趣, 所以我们需要的所有信息都已包含在矩阵 $\mathbf{P}^{(t)}$ 中. 可以证明密度矩阵 P 与分布 $\mathcal{D} = (\lambda_1(P), ..., \lambda_n(P))$ 一一对应. 因此, 在每一轮 t 中, 我们可以只要求算法选择一个密度矩阵 $\mathbf{P}^{(t)}$. 在该公式中, 我们观察到期望损失可以由 $\mathbf{M}^{(t)} \bullet \mathbf{P}^{(t)}$ 给出. 在 T 轮之后, 总的期望损失是 $\sum_{t=1}^{T} \mathbf{M}^{(t)} \bullet \mathbf{P}^{(t)}$. 对于该损失, 最好的专家即为最小化 $\sum_{t=1}^{T} \mathbf{v}^\top \mathbf{M}^{(t)} \mathbf{v}$ 的专家 $\mathbf{v}$. 因此, 最小损失由 $\lambda_n(\sum_{t=1}^{T} \mathbf{M}^{(t)})$ (最小特征值) 给出, 对应的专家为其任意特征向量.

但是, 我们在博弈开始并不知道该信息. 那么, 怎么做才能不依赖该信息, 同时让自己的损失接近最小呢? 算法的直觉是赋予每个专家一个权重, 并根据每个专家的表现动态更新它.

Algorithm 3 乘法加权更新算法[40]

输入: $\epsilon \le \frac{1}{2}$. 初始化权重矩阵 $\mathbf{W}^{(1)} = \mathbf{I}_n$.

1: **for** $t = 1, 2, \ldots, T$ **do**

2: 以概率分布 $\mathbf{P}^{(t)} = \frac{\mathbf{W}^{(t)}}{\mathrm{Tr}\left(\mathbf{W}^{(t)}\right)}$ 选择策略.

3: 博弈给出损失矩阵 $\mathbf{M}^{(t)}$.

4: 更新权重矩阵为

$$\mathbf{W}^{(t+1)} = \mathbf{W}^t \exp\left(-\epsilon \mathbf{M}^t\right). \tag{3.4.4}$$

5: **end for**

其中, 矩阵的幂次定义为

$$\exp(\mathbf{A}) = \sum_{i=0}^{\infty} \frac{\mathbf{A}^i}{i!}.$$

定理 3.4.1. *([41], 定理 10)*

乘法加权更新算法保证在 T 轮之后, 对于任何专家 $\mathbf{v}$, 我们有

$$(1-\epsilon)\sum_{t:\mathbf{M}^{(t)}\succeq\mathbf{0}}\mathbf{M}^{(t)}\bullet\mathbf{P}^{(t)}+(1+\epsilon)\sum_{t:\mathbf{M}^{(t)}\preceq\mathbf{0}}\mathbf{M}^{(t)}\bullet\mathbf{P}^{(t)}\leq\sum_{t=1}^{T}\mathbf{v}^{\top}\mathbf{M}^{(t)}\mathbf{v}+\frac{\ln n}{\epsilon}.$$

特别地, 取 $\mathbf{v}$ 为对应最优策略的专家, 则上面的算法以 $O(\ln n)$ 的速度逼近最优策略.

证明的直觉是构造和比较势函数 $\Phi^{(t)}=\mathrm{Tr}\left(\mathbf{W}^{(t)}\right)$ 的上下界. 完整版本的证明可以参考 [40] 的第 3 部分.

下面, 我们讨论如何基于乘法加权更新算法得到半定规划问题的一个算法. 这里, 我们讨论更一般的含不等式的半定规划问题

$$\begin{aligned}\max\quad & \mathbf{C}\bullet\mathbf{X}\\ \forall j\in[m],\quad & \mathbf{A_j}\bullet\mathbf{X}\leq b_j\\ & \mathbf{X}\succeq 0,\end{aligned}\tag{3.4.5}$$

其中, 为了保证问题有界, $\mathbf{A}_1=\mathbf{I}, b_1=R$, 即限制矩阵 $\mathbf{X}$ 的迹小于等于 R. 其对偶问题为

$$\begin{aligned}\min\quad & \mathbf{b}\cdot\mathbf{y}\\ & \sum_{j=1}^{m}\mathbf{A}_j y_j\succeq\mathbf{C}\\ & \mathbf{y}\geq\mathbf{0},\end{aligned}\tag{3.4.6}$$

其中 $\mathbf{y}=(y_1,\ldots,y_m)^{\top}$.

为了求解, 我们把原始、对偶优化问题转化为判定问题

$$\begin{array}{ll}\mathbf{C}\bullet\mathbf{X}\geq\alpha & \mathbf{b}\cdot\mathbf{y}\leq\alpha\\ \forall j\in[m]:\mathbf{A}_j\bullet\mathbf{X}\leq b_j & \sum_{j=1}^{m}\mathbf{A}_j y_j\succeq\mathbf{C}\\ \mathbf{X}\succeq\mathbf{0}, & \mathbf{y}\geq\mathbf{0}.\end{array}\tag{3.4.7}$$

这两个问题之间的强对偶性表明, 对于任意 α, 这两个问题中都恰有一个有解. 我们只需要对任意 α 判定其中哪个问题有解即可. 这样, 根据问题的有界性, 我们二分地选取 α, 即可计算问题的近似解.

将 $\mathbf{X}^{(1)}$ 初始化为迹为 R 的平凡正定对称矩阵 $\frac{R}{n}\mathbf{I}$ (可能不在可行域内). 随后, 迭代地生成原始解 $\mathbf{X}^{(2)},\mathbf{X}^{(3)},\ldots$ 每一步, 算法都试图改进 $\mathbf{X}^{(t)}$ 得到 $\mathbf{X}^{(t+1)}$. 更新由称为神谕 (ORACLE) 的辅助算法协助, 如下所示.

ORACLE 尝试证明当前 $\mathbf{X}^{(t)}$ 的有效性. 它在多胞体 $\mathcal{D}_\alpha=\{\mathbf{y}:\mathbf{y}\geq\mathbf{0},\mathbf{b}\cdot\mathbf{y}\leq\alpha\}$ 中搜索向量 $\mathbf{y}$, 满足

$$\sum_{j=1}^{m}\left(\mathbf{A}_j \bullet \mathbf{X}^{(t)}\right) y_j - \left(\mathbf{C} \bullet \mathbf{X}^{(t)}\right) \geq 0. \tag{3.4.8}$$

- 如果 ORACLE 成功找到这样的 $\mathbf{y}$, 那么要么 $\mathbf{X}^{(t)}$ 是原始不可行的, 要么 $\mathbf{C} \bullet \mathbf{X}^{(t)} \leq \alpha$. 否则有

$$\sum_{j=1}^{m}\left(\mathbf{A}_j \bullet \mathbf{X}^{(t)}\right) y_j - \left(\mathbf{C} \bullet \mathbf{X}^{(t)}\right) \leq \sum_{j=1}^{m} b_j y_j - \left(\mathbf{C} \bullet \mathbf{X}^{(t)}\right) < \alpha - \alpha = 0,$$

  矛盾!
- 另一方面, 如果不存在这样的 $\mathbf{y}$, 那么必然存在 ϕ^* 给出了一个原始可行解 $\mathbf{X}^* = \mathbf{X}/\phi^*$, 这样的 ϕ^* 可以用线性规划求出. ([41], 引理 3)

最后, 我们要用乘法加权更新算法构造某个程序. 在每一轮中, 如果 ORACLE 反馈"否", 那么我们立即知道原始问题是可行的; 否则, 我们可以使用 ORACLE 生成的 $\mathbf{y}^{(t)}$ 的值来更新矩阵 $\mathbf{X}^{(t)}$ 并最终逼近对偶的解. 算法的具体步骤如下.

定义 3.4.1. 对于参数 $0 \leq \ell \leq \rho$, 一个 (ℓ, ρ)-有界的 ORACLE, 是一个算法, 其搜索一个满足3.4.8的 $\mathbf{y} \in \mathcal{D}_\alpha$, 使得要么 $\sum_j \mathbf{A}_j y_j - \mathbf{C} \in [-\ell, \rho]$, 要么 $\sum_j \mathbf{A}_j y_j - \mathbf{C} \in [-\rho, \ell]$. ρ 称为该 ORACLE 的宽度.

Algorithm 4 半定规划的乘法加权更新算法[41]

输入: 原始问题和原始问题的一个 (ℓ, ρ)-有界的 ORACLE, 其中 $\ell \geq \frac{\delta\alpha}{4R}$.

1: **for** t=1, 2, …, T, $T = \frac{8\ell\rho R^2 \ln(n)}{\delta^2\alpha^2}$ **do**

2: 以参数 $\epsilon = \frac{\delta\alpha}{2\ell R}$ 运行算法3, 得到输出的密度矩阵 $\mathbf{P}^{(t)}$.

3: 如果 ORACLE 失败了, 则停止运行, 输出 $\mathbf{X}^{(t)} = R\mathbf{P}^{(t)}$.

4: 否则, 令 $\mathbf{y}^{(t)}$ 为 ORACLE 生成的向量.

5: 向算法3提供矩阵

$$\mathbf{M}^{(t)} = \left[\frac{1}{\ell+\rho}\sum_{j=1}^{m}\mathbf{A}_j y_j^{(t)} - \mathbf{C} + \ell^{(t)}\mathbf{I}\right],$$

其中

$$\ell^{(t)} = \begin{cases} \ell & \text{if} \quad \sum_{j=1}^{m}\mathbf{A}_j y_j^{(t)} - \mathbf{C} \in [-\ell, \rho], \\ -\ell & \text{if} \quad \sum_{j=1}^{m}\mathbf{A}_j y_j^{(t)} - \mathbf{C} \in [-\rho, \ell]. \end{cases}$$

6: **end for**

使用定理3.4.1, 我们可以得到该算法运行时间的理论保障.

定理 3.4.2. *([41], 定理 13)*

假设算法4在 T 次迭代后停止 (否则原始问题有解), 令 $\overline{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}^{(t)}$, 则 $\mathbf{y}^* = \overline{\mathbf{y}} + \frac{\delta\alpha}{R}e_1$ 是一个对偶问题的解, 而且目标函数值至多为 $(1+\delta)\alpha$.

3.4.3 总结

乘法加权算法最大的优点在于迭代速度很快. 根据上面的定理, 只需要 $O(\ln n)$ 次迭代, 我们就能得到原始-对偶问题的一个分离近似解. 但是, 算法的过程中还需要求解 ORACLE. 在经典算法下, 没有什么求解 ORACLE 的比较好的手段. 但是, 在量子算法中, 利用约束的信息, Gibbs 采样方法可以给出 ORACLE 的一个高效求解器. 在该求解器的辅助下, 可以给出复杂度在 $O(m^{1/2}n^{5/2}/\epsilon^8)$ 的量子算法[38].

3.5 sketchyCGAL 算法

3.5.1 简述

尽管之前介绍的众多算法已经大大降低了半定规划问题的复杂度, 但是在大规模问题下, 算法仍然不如很多启发式方法. 这主要是因为之前的算法必须涉及矩阵运算和存储, 算法的内存至少为 $O(n^2)$, 这对大规模问题已经是不可忽视的开销了. 为了进一步打破该约束, 2021 年, Yurtsever, Tropp, Fercoq, Udell 和 Ceveher[25]提出了近似条件梯度增广拉格朗日 (sketchyCGAL) 算法. 该算法的基础是基于锥优化 (conic optimzation) 方法的条件梯度增广拉格朗日算法[17]. 作者利用随机性和低秩近似的手段, 把迭代法的时间、空间复杂度进一步降低了. 以精度降低为代价, 比起以往的方法, 该方法可以求解更大规模的半定规划问题.

在介绍该算法之前, 需要补充一些基础知识.

3.5.2 基础知识

3.5.2.1 增广拉格朗日法 (Augmented Lagrangian)

对于优化问题: $\min f(\mathbf{x})$ s.t. $\mathbf{c}(\mathbf{x}) = \mathbf{0}$, 定义其增广拉格朗日函数

$$\phi(\mathbf{x}, \lambda, \sigma) = f(\mathbf{x}) - \lambda^\top \mathbf{c}(\mathbf{x}) + \frac{1}{2}\sigma \sum_i c_i(\mathbf{x})^2,$$

即拉格朗日函数与加权平方罚函数的和.

增广拉格朗日函数最重要的性质是, 存在 σ^*, 当 $\sigma > \sigma^*$ 时, x^* 是函数的严格局部极小点推出 x^* 原问题的严格局部最优解. 反之, 若 x^*, λ^* 满足二阶充分条件, 且是原问题的严格局部最优解, 则是函数的严格局部极小点. 证明参考 [42], 第 17.3 节. 这表明, 若迭代法保证 σ 趋于正无穷, 则迭代法在有限次内收敛.

Algorithm 5 增广拉格朗日函数法 ([42], 第 17.3 节)

输入: 给定初始点 $\mathbf{x_0}, \lambda_1, \sigma_1 > 0, \rho > 1, \epsilon > 0, \epsilon_1 > 0, k = 1$.

1: 以 $\mathbf{x_{k-1}}$ 为初始点, 求解

$$\min_{\mathbf{x}} \Phi\left(\mathbf{x}, \lambda_{\mathbf{k}}, \sigma_k\right)$$

得 $\mathbf{x_k}$, 它满足

$$\left\|\nabla_{\mathbf{x}} \Phi\left(\mathbf{x_k}, \lambda_{\mathbf{k}}, \sigma_k\right)\right\| < \epsilon_1.$$

2: $\lambda_{\mathbf{k+1}} = \lambda_{\mathbf{k}} - \sigma_k \mathbf{c}\left(\mathbf{x_k}\right)$.

3: 若 $\left\|\mathbf{c}\left(\mathbf{x_k}\right)\right\| \leqslant \epsilon$, 则迭代停止, 得近似解 $\mathbf{x_k}, \lambda_{\mathbf{k+1}}$;
否则, 取 $\sigma_{k+1} = \rho\sigma_k$, $k = k + 1$, 转步 1.

3.5.2.2　条件梯度法 (Conditional Gradient Method)

考虑优化问题

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } \mathbf{x} \in C,$$

其中 C 为全空间 E 上的凸集, f 光滑.

该问题最常见的方法是一阶的投影梯度法. 设计投影算子 $\mathbf{P_C} : E \to C$, 迭代过程为 $\mathbf{x^k} = \mathbf{P_C}(\mathbf{x^{k-1}} - t_k \nabla f(\mathbf{x^{k-1}}))$, 它是二阶方法近端梯度法的特殊情况.

另一种方法则是 1953 年由 Frank-Wolfe 发明的条件梯度法. 首先定义 f 做局部线性展开

$$\hat{f}^{lin}(\mathbf{s}) = f(\mathbf{x^{k-1}}) + \nabla f(\mathbf{x^{k-1}})^\top (\mathbf{s} - \mathbf{x^{k-1}}).$$

随后迭代求解. 取 $\mathbf{s^{k-1}} = \arg\min_{\mathbf{s}\in C} \hat{f}^{lin}(\mathbf{s})$, 然后更新 $\mathbf{x^k}$ 为 $\mathbf{x^k} = (1 - \gamma^k)\mathbf{x^{k-1}} + \gamma^k \mathbf{s^{k-1}}$. 取 $\gamma_k = \frac{2}{k+1}$, 则该算法收敛.

该方法有一个很特殊的性质. 若 $C = \{\mathbf{s} : ||\mathbf{s}|| \leq t\}$, $||\cdot||$ 为某个范数, 则

$$\arg\min_{\mathbf{s}\in C} \hat{f}^{lin}(\mathbf{s}) = \arg\min_{||\mathbf{s}||\leq t} \nabla f(\mathbf{x^{k-1}})^\top \mathbf{s} = -t \arg\max_{||\mathbf{s}||\leq 1} \nabla f(\mathbf{x^{k-1}})^\top \mathbf{s} = -t||\nabla f(\mathbf{x^{k-1}})||_*,$$

其中, $||\cdot||_*$ 是范数 $||\cdot||$ 的对偶范数.

特别地, 对于矩阵而言, 其迹范数 $||A||_{tr} = \sum_i \lambda_i(A)$ 的对偶范数是算子范数 $||A||_{op} = \lambda_{max}(A)$. 因此, 算法的更新只需要求矩阵的最大特征值.

3.5.2.3　半正定矩阵的低秩近似

在机器学习中, 一个重要的问题是如何导出一个大规模矩阵的高效近似. 该矩阵可能是支持向量机、主成分分析的核矩阵等等. 在大规模矩阵中, 运算和存储都是非常困难的. 一个引人注目的解决方案是 2000 年由 Williams 和 Seeger 引入的 Nyström 方法.

Nyström 方法是 1928 年由 Nyström 提出的对半正定矩阵导出低秩近似的方法. 对

于一个半正定矩阵

$$\mathbf{D} = \begin{bmatrix} \mathbf{A} & \mathbf{S} \\ \mathbf{S^T} & \mathbf{Q} \end{bmatrix},$$

其中 $\mathbf{A}$ 的秩为 r, $k < r$, $\mathbf{A} = \mathbf{U\Sigma U}^\top$. $\mathbf{A}_k^+ = \sum_{i=1}^k \sigma_i^{-1}\mathbf{U}_i\mathbf{U}_i^\top$, 记 $\mathbf{R} = [\mathbf{A}\ \mathbf{S}]$, 则 $\hat{\mathbf{D}}_k = \mathbf{R}^\top\mathbf{A}_k^+\mathbf{R}$ 是 $\mathbf{D}$ 的一个秩 k 近似.

使用 Nyström 方法的思想, 文献 [8] 提出了半正定矩阵的低阶近似方法. 考虑 $c > k$, 在 $\mathbf{G}$ 中以概率 $p_i, i = 1, \cdots, n$ 独立同分布地选择 c 个列, 如果选中了第 i 个列, 则将其除以 cp_i. 我们把这 c 个列合并得到 $n \times c$ 矩阵 $\mathbf{C}$. 同时, 取出这 c 个行和列在 $\mathbf{G}$ 中对应的元素, 得到一个 $c \times c$ 的子矩阵, 并且对在第 i 个行, 第 j 个列的元素除以 $c\sqrt{p_ip_j}$, 得到的矩阵记为 $\mathbf{W}$. 取 $\mathbf{W}_k$ 为 $\mathbf{W}$ 的 k 维最佳低秩近似 (参考第2.2.4节), 则 $\hat{\mathbf{G}}_k = \mathbf{C}\mathbf{W}_k^+\mathbf{C}^\top$ 为矩阵 $\mathbf{G}$ 的一个半正定的秩 k 近似.

与利用复杂度为 $O(mn(m+n))$ 的奇异值分解直接生成矩阵的秩 k 近似相比, 这种方法在计算和存储上都非常容易. 为了计算该矩阵, 我们只需要生成 c 个随机数, 并且在放缩运算中计算 $nr + r^2$ 次. 要存储该矩阵, 只需要存储矩阵 $\mathbf{C}$ 和 $\mathbf{W}_k^+$, 消耗的内存也减少到 $nr + r^2$. 特别地, 当 r 取常数时, 计算和存储消耗都缩减到 $O(n)$. 同时, 文献 [8] 证明, 这种方法导出的近似 $\hat{\mathbf{G}}_k$ 以高概率接近矩阵最佳的秩 k 近似 $\mathbf{G}_k$.

定理 3.5.1. *([8], 定理 3)*

对于 $n \times n$ 维半正定矩阵 $\mathbf{G}$, 取 $p_i = G_{ii}^2/\sum_{i=1}^n G_{ii}^2$, 令 $\epsilon > 0$, $\eta = 1 + \sqrt{8\log(1/\delta)}$, $c \geq 64k\eta^2/\varepsilon^4$, 则

$$\mathsf{E}\left[\left\|\mathbf{G} - \hat{\mathbf{G}}_k\right\|_F\right] \leq \|\mathbf{G} - \mathbf{G}_k\|_F + \epsilon\sum_{i=1}^n G_{ii}^2,$$

同时, 如果 $c \geq 4/\varepsilon^2$, 则

$$\mathsf{E}\left[\left\|\mathbf{G} - \hat{\mathbf{G}}_k\right\|_2\right] \leq \|\mathbf{G} - \mathbf{G}_k\|_2 + \varepsilon\sum_{i=1}^n G_{ii}^2.$$

当然, 这样导出的低秩近似是静态的, 不足以满足目前迭代的需要. 我们需要导出一个可以动态更新的低秩近似. 文献 [43] 修改了上面介绍的随机选取若干列做近似的思路, 提出了一个可以满足动态更新要求的低秩近似算法.

假设现在的更新方式为 $\mathbf{A}_{t+1} = \lambda\mathbf{A}_t + (1-\lambda)\alpha_t\alpha_t^\top$. 选择一个随机正交测试矩阵 $\mathbf{\Omega} \in \mathbb{R}^{n\times k}$, 定义 $\mathbf{A}$ 的近似 (sketch) 为 $\mathbf{Y} = \mathbf{A\Omega} \in \mathbb{R}^{n\times k}$. 定义 $\hat{\mathbf{A}} = \mathbf{Y}([\mathbf{\Omega}^\top\mathbf{Y}]_r)^\dagger\mathbf{Y}^\top$, 即为所求 r 阶近似. 此时, $\mathbf{Y}$ 和测试矩阵 $\mathbf{\Omega}$ 的大小为 nk, 在迭代过程中, 算法的空间复杂度仍为 $O(nk + k^2)$.

在每一步更新时, 只需更新近似 $\mathbf{Y}$, 即 $\mathbf{Y}_{t+1} = \lambda\mathbf{Y}_t + (1-\lambda)\alpha_\mathbf{t}(\alpha_\mathbf{t}^\top\mathbf{\Omega})$, 计算次数为两次矩阵向量乘和一次矩阵求和, 复杂度为 $O(nk)$. 最后, 还原矩阵 $\mathbf{A}$ 的近似时, 使用奇异

值分解计算 k 阶矩阵的 r 阶近似的逆的时间复杂度为 $O(k^3)$, 计算矩阵乘法的复杂度为 $O(nk^2)$.

总之, 该方法的时间和空间复杂度基本不变. 同时, 我们还能证明类似3.5.1的结果.

定理 3.5.2. *([43], 定理 4.1)*

$$\mathsf{E}\left\|\mathbf{A}-\hat{\mathbf{A}}_r\right\|_1 \leq \left(1+\frac{r}{k-r-1}\right)\cdot\|\mathbf{A}-\mathbf{A}_r\|_1$$
$$\mathsf{E}\left\|\mathbf{A}-\hat{\mathbf{A}}_r\right\|_\infty \leq \|\mathbf{A}-\mathbf{A}_r\|_\infty+\frac{r}{k-r-1}\cdot\|\mathbf{A}-\mathbf{A}_r\|_1$$

经过这些准备, 我们可以给出完整的近似条件梯度增广拉格朗日算法.

3.5.3 算法原理

近似条件梯度增广拉格朗日算法求解的是标准形式的半定规划问题 (2.3.1), 但是要求其中一个等式约束为 $\mathbf{I}\bullet\mathbf{X}=\alpha$, 即 $\mathrm{Tr}\,\mathbf{X}=\alpha$. 定义 $\Delta_n=\{\mathbf{X}:\mathrm{Tr}\,\mathbf{X}=1,\mathbf{X}\in\mathbb{S}_+^n\}$. 那么, 该约束与对称半正定的要求将 $\mathbf{X}$ 限制在凸集 $\alpha\Delta_n$ 中, 我们就能使用条件梯度法求解这个问题.

近似条件梯度增广拉格朗日算法是基于经典的条件梯度增广拉格朗日算法的近似加速. 因此, 我们先介绍经典的条件梯度增广拉格朗日算法, 随后介绍如何使用随机和近似算法对其进行加速.

3.5.3.1 条件梯度增广拉格朗日算法

方便起见, 我们定义算子 $\mathcal{A}\mathbf{X}=[\mathbf{A_1}\mathbf{X},\cdots,\mathbf{A_m}\mathbf{X}]^\top$ 和其伴随算子 $\mathcal{A}^*\mathbf{z}=\sum_{i=1}^m z_i\mathbf{A}_i$.

原始问题的增广拉格朗日函数为

$$\mathbf{L}_\beta(\mathbf{X},\mathbf{y})=\mathbf{C}\bullet\mathbf{X}+\mathbf{y}^\top(\mathcal{A}\mathbf{X}-\mathbf{b})+\frac{\beta}{2}||\mathcal{A}\mathbf{X}-\mathbf{b}||^2,$$

其中 $\mathbf{X}\in\alpha\Delta_n, y\in R^d$. 该函数关于 $\mathbf{X}$ 的偏导数为

$$\partial_{\mathbf{X}}\mathbf{L}_\beta=\mathbf{C}+\mathcal{A}^*\mathbf{y}+\beta\mathcal{A}^*(\mathcal{A}\mathbf{X}-\mathbf{b}).$$

使用原始的增广拉格朗日方法5, 问题的过程为

1. **初始化** 选取 $\mathbf{X}_1\in\alpha\mathbf{\Delta}_n, \mathbf{y}_1\in\mathbb{R}^d$ 和 β_0, 随后循环执行下面的过程.
2. **原始步骤** $\mathbf{X}_{t+1}\in\arg\min\left\{L_\beta\left(\mathbf{X},\mathbf{y}_t\right):\mathbf{X}\in\alpha\Delta_n\right\}$.
3. **对偶步骤** $\mathbf{y}_{t+1}=\mathbf{y}_t+\beta(\mathcal{A}\mathbf{X}-\mathbf{b})$.
4. $\beta=\beta_0\sqrt{t+1}$.

一般而言, 该问题的主要困难在于求解 $\arg\min\left\{L_\beta\left(\mathbf{X},\mathbf{y}_t\right):\mathbf{X}\in\alpha\mathbf{\Delta}_n\right\}$. 文献 [25] 注意到 $\alpha\Delta_n$ 是一个凸集, 因此使用条件梯度法近似求解该步骤.

参考3.5.2.2给出的条件梯度法的一般过程, 我们要在更新中求解线性化的 $\mathbf{L}_\beta$ 的最小值, 即求解

$$\arg\min_{\mathbf{X}'\in\alpha\Delta_n} \partial_{\mathbf{X}}\mathbf{L}_\beta(\mathbf{X}^{k-1})\bullet\mathbf{X}'.$$

将 $\partial_{\mathbf{X}}\mathbf{L}_\beta(\mathbf{X}^{k-1})$ 简记为 $\mathbf{D}_t$, 则相当于求解

$$\arg\min_{\mathbf{H}\in\alpha\Delta_n}\{\mathbf{D}_t\bullet\mathbf{H}\}=\alpha\arg\min_{\mathbf{H}\in\Delta_n}\{\mathbf{D}_t\bullet\mathbf{H}\}.$$

注意到, $\mathbf{D}_t$ 和 $\mathbf{H}$ 都是实对称矩阵, 因此, 根据矩阵内积 (迹运算) 的交换性, 不妨设 $\mathbf{D}_t$, $\mathbf{H}_t$ 都是对角矩阵 (通过交换可以消去正交对角化带来的正交矩阵). 根据 $\mathbf{H}\in\Delta_n$, 有 $\mathrm{Tr}\,\mathbf{H}=1, \lambda_i(H)\geq 0, \forall i$, 从而 $\mathbf{D}_t\bullet\mathbf{H}$ 相当于对 $\mathbf{D}_t$ 的特征值的非负加权和, 因此最小值即为 $\mathbf{D}_t$ 的最小特征值 $\lambda_n(\mathbf{D}_t)$. 同时, $\arg\min_{\mathbf{H}\in\Delta_n}\{\mathbf{D}_t\bullet\mathbf{H}\}$ 即为 $\mathrm{conv}(\mathbf{v}\mathbf{v}^\top:\mathbf{D}_t\mathbf{v}=\lambda_{min}\mathbf{v}, ||\mathbf{v}||=1)$.

因此, 根据上面的讨论, 我们只需要把条件梯度法结合到算法中, 即把第 2 步修改为 $\mathbf{X}_{t+1}=(1-\eta_t)\mathbf{X}_t+\eta_t\mathbf{v}_t\mathbf{v}_t^\top$, 同时每次更新 $\eta_t=2/(t+1)$. 其中, $\mathbf{v}_t$ 是 $\partial_{\mathbf{X}}\mathbf{L}_\beta(\mathbf{X}^{k-1})$ 的最小特征值的任意特征向量. 同时, 为了保证满足若干理论性质, 作者把第 3 步中的 β_t 改为了满足若干性质的 γ_t, 并且要求在 $||\mathbf{y}_t||$ 过大时不再执行对偶步骤.

经过这些处理, 我们就得到了经典的条件梯度增广拉格朗日方法.

定理 3.5.3. *([25], 事实 3.1)*

使用条件梯度增广拉格朗日方法得到的序列 $\{\mathbf{X}_t:t=1,2,3,...\}\in\alpha\Delta_n$ *满足*

$$||\mathcal{A}\mathbf{X}_t-\mathbf{b}||\leq\frac{Const}{\sqrt{t}}, |\mathbf{C}\bullet\mathbf{X}_t-\mathbf{C}\bullet\mathbf{X}_*||\leq\frac{Const}{\sqrt{t}}.$$

因此, 为了生成一个 ϵ-近似最优解, 我们只需要执行 $O(\epsilon^{-2})$ 次迭代. 下面, 我们讨论如何使用近似和随机方法来进一步做加速.

3.5.3.2 近似加速

近似加速主要是用随机 Lanczos 方法对求最小特征值和特征向量做加速和用 Nyström 方法对迭代过程加速.

首先, 文章中, 作者采用了比较经典的随机 Lanczos 算法[44], 其可以保证在每一轮仅使用 $O(n)$ 次计算的情况下, $O(\log n)$ 次迭代内以高概率收敛到一个比较好的近似解 ([25], 事实 4.1).

同时, 注意到, 我们上面提出的更新过程恰好满足使用3.5.2.3中提出的对动态更新的半正定矩阵给出近似的条件. 因此, 我们可以直接使用上面的方法, 利用一个大小为 $n\times k$ 的测试矩阵 $\mathbf{\Omega}$ 执行更新. 这样, 我们只需要更新目标矩阵 $\mathbf{X}$ 的近似 $\mathbf{S}=\mathbf{X}\mathbf{\Omega}$. 算法步骤 3 的更新 $\mathbf{X}_{t+1}=(1-\eta_t)\mathbf{X}_t+\eta_t\mathbf{v}_t\mathbf{v}_t^\top$ 可以通过计算 $\mathbf{S}=(1-\eta_t)\mathbf{S}+\eta_t\mathbf{v}_t(\mathbf{v}_t^\top\mathbf{\Omega})$ 执行. 内存消耗缩减为 $2kn$, 计算一次更新则只需要执行 $O(kn)$ 次代数运算.

在算法结束时, 我们还需要利用求出的近似还原出原始矩阵. 利用 Nyström 方法, 我们还原为 $\hat{\mathbf{X}} = \mathbf{S}(\mathbf{\Omega}^\top\mathbf{S})^\dagger\mathbf{S}^\top$ 即可.

把以上的方法组合起来, 并且令 $\mathbf{z}_t = \mathcal{A}\mathbf{X}$ 同步迭代, 即可得到以下算法.

Algorithm 6 近似条件梯度增广拉格朗日算法[25]

输入: 缩放后的原始问题 (要求 $||C||_F = ||\mathbf{A}|| = \alpha = 1, ||\mathbf{A}_1||_F = \cdots = ||\mathbf{A}_m||_F$.)

1: 初始化 $\beta_0 = 1, K = +\infty, \mathbf{z}_0 = \mathbf{0}_d, \mathbf{y}_0 = \mathbf{0}_d$.
2: 生成 Nyström 随机测试矩阵 $\mathbf{\Omega}$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: $\quad\beta_t = \beta_0\sqrt{t+1}, \eta_t = 2/(t+1)$.
5: $\quad$使用随机 Lanczos 方法计算矩阵 $\mathbf{C} + \mathcal{A}^*(\mathbf{y} + \beta(\mathbf{z}_{t-1} - \mathbf{b}))$ 的近似最小特征值和特征向量 $[\xi, \mathbf{v}_t]$.
6: $\quad\mathbf{z}_t \leftarrow (1-\eta_t)\mathbf{z}_{t-1} + \eta_t\mathcal{A}\left(\alpha\mathbf{v}_t\mathbf{v}_t^\top\right)$.
7: $\quad\mathbf{y}_t = \mathbf{y}_t + \gamma_t(\mathbf{z}_t - \mathbf{b})$.
8: $\quad$使用向量 $\sqrt{\alpha}\mathbf{v}_t$ 和参数 η_t 类似 $\mathbf{z}_t$ 更新 $\mathbf{X}_t$ 的近似 $\mathbf{Z}_t$.
9: **end for**
10: 利用 $\mathbf{Z}_T$ 重建近似解 $\hat{\mathbf{X}}$.

定理 3.5.4. *([25], 定理 6.3)*

在原问题满足强对偶性时, 通过适当选取近似的大小 R, 该算法以高概率在 $O(\epsilon^{-2})$ 次迭代内生成原问题的一个 ϵ-近似解的 ξ-最优 r 阶近似, 使用的内存为 $O(m + Rn)$, 运算次数为 $O(\epsilon^{-2}(m + Rn) + \epsilon^{-2.5}\log(n/\epsilon))$.

而且, 2000 年左右的一些关于半定规划问题的低秩解的研究进一步证明, 稀疏的半定规划问题本身甚至就保证了低秩解的存在性和唯一性, 从而向我们暗示, 该算法在低秩问题上可能会有更好的表现.

定理 3.5.5. *([45-46])*

半定规划问题在实数域上必定有秩小于等于 $\sqrt{2(m+1)}$ 的解, 在复数域上必定有秩小于等于 $\sqrt{(m+1)}$ 的解.

除了一个 $\mathbf{C}, \mathbf{A_1}, ..., \mathbf{A_m}$ 的零测集外, 半定规划问题在实数域上必定有唯一秩小于等于 $\sqrt{2(m+1)}$ 的解.

第四章　半定规划问题在组合优化中的应用

本章中, 我们主要介绍半定规划问题在的若干应用. 我们主要介绍在组合优化中的应用, 并且在最后一节概括一下半定规划在其它问题中的应用. 由于半定规划问题的高效算法已经很成熟, 所以在实践中, 一种应用是直接把组合优化问题建模为半定规划问题, 而更加广泛的应用则是利用半定规划问题近似求解困难的组合优化问题 (常为NP-困难的).

4.1　直接把组合优化问题建模为半定规划问题

本节的内容主要参考 [47-48].

一般而言, 组合优化问题的直接半定规划建模往往与图距离问题紧密关联. 假如我们把 n 个顶点的图 G 上的顶点 $v_1, \cdots, v_n$ 映射为 n 维向量 $\mathbf{v}_1, \cdots, \mathbf{v}_n$, 并且对内积做一些要求, 则自然地, 我们可以取矩阵 $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$, 及 $\mathbf{X} = \mathbf{V}^\top\mathbf{V}$. 注意到, 矩阵 $\mathbf{V}$ 和半正定矩阵 $\mathbf{X}$ 的这种对应是唯一的 (利用正交分解) , 从而, 原问题中对内积的约束可以转化为对半正定矩阵 $\mathbf{X}$ 的约束, 得到一个半定规划问题.

4.1.1　图的最小单位距离表示问题

假设我们要求图 G 的距离表示满足约束 $|\mathbf{v_i} - \mathbf{v_j}| = 1, \forall (i,j) \in E$, 图的最小单位距离表示问题即为求这种距离表示中向量的最大模长的最小可能值.

根据上面的方法, $|\mathbf{v_i} - \mathbf{v_j}| = 1$ 可以写做 $\mathbf{X}_{ii} + \mathbf{X}_{jj} - 2\mathbf{X}_{ij} = 1$, 而 $\mathbf{v}_i$ 的模长就是 $\mathbf{X}_{ii}$. 因此, 我们可以把这个问题写成一个2.3.3形式的半定规划问题

$$\begin{aligned} \min \quad & w \\ \text{s.t.} \quad & \mathbf{A} \succeq 0 \\ \forall i, \quad & \mathbf{A}_{ii} \le w \\ \forall (i,j) \in E, \quad & \mathbf{A}_{ii} - 2\mathbf{A}_{ij} + \mathbf{A}_{jj} = 1. \end{aligned}$$

该半定规划问题的解即为最大模长最小值的平方.

4.1.2　图的 Lovász theta 函数

除了如上面的例子求解一些简单的优化问题以外, 有时候, 我们定义的优化问题的解还可能具有组合意义.

1973 年, 为了求解香农熵问题, Lovász 在图上用半定规划问题定义了 Lovász theta

函数 $\vartheta(G)$, 后来发现, ϑ 函数还具有很多独特的性质.

在图 G 上, 定义顶点集合 $S \subset V(G)$ 为一个稳定集 (或称独立集), 若其中任意两顶点不相邻. 最大稳定集问题即求解图中有稳定集的最大可能顶点数量, 是一个 NP 完全问题[49]. 图 G 的最大稳定集问题的解记为 $\alpha(G)$.

ϑ 函数定义的初衷是希望能够给出最大稳定集问题的一个上界, 从而证明 Petersen 图的香农熵为 $\sqrt{5}$. 具体而言, 如果我们能够把图 G 上的顶点 $v_1, \cdots, v_n$ 映射为 n 维向量 $\mathbf{v}_1, \cdots, \mathbf{v}_n$, 并且保证如果任意两个顶点不相邻, 则其对应的向量正交, 则对任意单位向量 $\mathbf{c}$ 和任意稳定集 S, 由于 S 的顶点对应的向量相互正交, 因此根据 Parseval 不等式, 我们有 $1 = ||\mathbf{c}||^2 \geq \sum_{i \in S}(\mathbf{c} \cdot \mathbf{v_i})^2$. 假设 $\lambda = \min_{i \in S}(\mathbf{c} \cdot \mathbf{v_i})^2$, 则有 $1 \geq \lambda |S|$, 从而最大稳定集有上界 $1/\lambda$.

由于这个上界对任意的满足上面条件的 $\mathbf{c}, \mathbf{v_i}$ 都成立, Lovász 提出, 记 $\vartheta(G) =$

$$\begin{aligned} \min_{\mathbf{c}, \mathbf{v}_i} 1/\lambda = \min_{\mathbf{c}, \mathbf{v}_i} \max_i (1/(\mathbf{c}^\top \mathbf{v_i}))^2 \\ \text{s.t.} \quad ||\mathbf{c}|| = 1, ||\mathbf{v}_i|| = 1, \mathbf{v_i}^\top \mathbf{v_j} = 0, \end{aligned}$$

则 $\vartheta(G) \geq \alpha(G)$. 并且, 由于上面的问题由于约束只与内积相关, 因此可以将其写成一个半定规划问题. 记 $\mathbf{V} = [v_1, ..., v_n]$, $\mathbf{X} = \mathbf{V}^\top \mathbf{V}$, 则 $1/(1 - \vartheta(G)) =$

$$\begin{aligned} \max\, & t \\ \text{s.t.} \forall i \in [n] \quad & \mathbf{X}_{ii} = 1 \\ \forall (i, j) \notin E, \quad & \mathbf{X}_{ij} = t. \end{aligned}$$

此外, Lovász 及其后续的工作还证明了 $\vartheta(G)$ 的若干性质. 例如, 三明治定理[50]证明 $\alpha(G) \leq \vartheta(G) \leq \chi(\overline{G})$.($\chi$ 为图的色数), 随机图上的 $\vartheta(G)$ 以概率 $1 - o(1)$ 取到 $[\sqrt{n}, 2\sqrt{n}]$[51].

4.2 导出组合优化问题的半定规划近似

本节的内容参考[52]第 12 章.

事实上, 半定规划问题更加广泛的应用是给出组合优化问题的松弛. 很多组合优化问题的自然形式都是整数规划问题. 而在建模整数约束时, 半定规划问题天然具有优势. 例如, 整数约束 $x \in \{-1, 1\}$ 可以写成二次形式 $x^2 = x$, 从而与半定规划联系起来. 这种联系一般是通过把向量嵌入高维空间建立的. 我们下面通过著名的最大割问题的例子来解释如何获取这种联系.

最大割问题是一个常见的 NP 完全[53]组合优化问题, 其目标为求解带权图 G 的一个顶点分割 S 和 $G \backslash S$, 使得两个部分之间的边数最大. 不妨设图 G 的顶点数量为 n, 则

我们用变量 $x_i \in \{-1, 1\}$ 表示顶点 i 在 S 还是 $G\backslash S$ 中, 则 $1 - x_i x_j$ 就是边 (i, j) 是否连接在两个部分之间的表征. 从而, 两个部分之间的边数可以表示为

$$\frac{1}{2}\sum_{i<j} w_{ij}\left(1 - x_i x_j\right),$$

其中, w_{ij} 为边 (i, j) 的权重.

因此, 最大割问题可以表示为整数规划问题

$$\begin{array}{ll} \max & \frac{1}{2}\sum_{i<j} w_{ij}\left(1 - x_i x_j\right) \\ \text{s.t. } \forall i \in [n] & x_i \in \{-1, 1\}. \end{array} \tag{4.2.1}$$

但是, 这样的等价转换并不能改变最大割问题是 NP-困难的本质, 我们必须对整数规划问题做适当的松弛. 为此, 1991 年, Lovász 和 Schrijver[54]首先提出了整数规划问题的投影法. 这个方法通过把变量松弛为高维向量或把约束松弛为半定约束, 最终导出原问题的松弛形式. 后续的大量工作则进一步发展了这两种松弛方法, 一方面对更加广泛的整数规划问题提出松弛, 另一方面讨论各种松弛的强弱关系和等价性. 我们在4.2.1中介绍这些工作.

此外, 如果要求出原问题的近似解, 那么还需要给出把松弛解舍入原问题解的技巧. 这方面最重要的工作是 Goemans, Williamson 在 1995 年提出的随机分离超平面方法[6]. 他们基于该方法提出了最大割问题的 GW 算法, 到目前为止仍然是最大割问题的理论保障最好的算法. 我们在4.2.2中介绍该工作.

4.2.1　组合优化问题的松弛

组合优化问题的半定规划松弛一般是通过整数规划的问题形式导出的. 不同性质的组合优化问题导出的整数规划问题的难度有显著的不同, 主要体现在除了整数约束之外是否有额外的约束上. 对于不同类型的额外约束, 人们提出了不同的方法来解决.

4.2.1.1　无额外约束

最简单的整数规划问题除了变量是整数外没有额外约束. 这类问题的典型代表即最大割问题4.2.1. 对于这类问题, 我们可以从三种不同的视角导出其半定规划松弛 (后面更加复杂的问题也会涉及不同视角).

- **向量化视角**[6,54] 这种视角的主要思想是把整数变量松弛为高维向量.
    1. 把整数规划和目标函数写成关于变量的二次形式, 从而问题转化为二次规划问题. 例如, $y_i \in \{-1, 1\}$ 可以写成 $y_i^2 = 1$, $y_i \in \{0, 1\}$ 可以写成 $(y_i - \frac{1}{2})^2 = \frac{1}{4}$. 在目标函数中, 对于线性出现的 y_i, 也可以类似地利用二次约束写成 y_i^2 的形式.

2. 假设变量总数为 n, 我们把每个变量松弛为 n 维向量, 再仿照上面建模最小单位距离问题的方法把向量内积的约束变成关于半正定矩阵的约束即可. 假设松弛后的变量为 $\mathbf{v_1}, \cdots, \mathbf{v_n}$, 记 $\mathbf{V} = [\mathbf{v_1}, \cdots, \mathbf{v_n}]$, $\mathbf{X} = \mathbf{V}^\top\mathbf{V}$, 则关于 $\mathbf{v_i}$ 内积的约束就转化对半正定矩阵 $\mathbf{X}$ 的约束.

- **约束松弛视角**[55] 这种视角主要是先把问题等价转化为带非半定约束的半定规划问题, 再适当松弛约束, 得到松弛的半定规划问题. 与比较定势的向量化相比, 这个方法更加灵活, 可以导出多种松紧不同的约束以供选择.
  1. 把目标函数写成关于变量形如 $\mathbf{x}^\top\mathbf{Q}\mathbf{x}$ 的形式, 注意到其值等于 $\mathbf{Q} \bullet (\mathbf{x}\mathbf{x}^\top)$.
  2. 把半正定矩阵 $xx^\top$ 视为矩阵 $\mathbf{X}$, 利用关于 x_i 的整数约束推出 (不是等价转化)$\mathbf{X}$ 的一系列半定规划形式的约束, 从而得到原问题的半定规划松弛.
- **拉格朗日对偶视角**[56] 这种视角直接把问题当成优化问题, 用拉格朗日对偶方法求解2.3.2.
  1. 类似向量化视角的步骤 1, 把整数规划约束和目标函数写成关于变量的二次形式.
  2. 对该问题求拉格朗日对偶, 得到半定规划问题.

下面, 我们以最大割问题4.2.1为例解释如何使用这三种视角推导问题的半定规划松弛.

在向量化视角下, 目标函数已经是二次的. 只需把 n 个整数变量松弛为 $\mathbf{v_1}, \cdots, \mathbf{v_n}$, 记 $\mathbf{V} = [\mathbf{v_1}, \cdots, \mathbf{v_n}]$, $\mathbf{X} = \mathbf{V}^\top\mathbf{V}$, 问题的松弛即为

$$\begin{aligned} \max \quad & \tfrac{1}{4}\textstyle\sum_{i,j} w_{ij} - \tfrac{1}{4}\mathbf{X} \bullet \mathbf{W} \\ \text{s.t.} \quad & \mathbf{X} \succeq 0, \operatorname{diag}(X) = 1_n. \end{aligned} \tag{4.2.2}$$

而在约束松弛视角下, 我们希望根据 $x_i \in \{-1, 1\}$ 导出关于半正定矩阵 $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ 的若干性质, 而且由于目标是提出半定规划问题, 我们希望其尽量形如 $\mathbf{A_i} \bullet \mathbf{X} = (\leq) b_i$. 常见的性质包括

1. $\operatorname{rank}(\mathbf{X}) = 1$.
2. $\mathbf{X}_{ij} \in \{-1, 1\}$.
3. $\operatorname{diag}(\mathbf{X}) = \mathbf{1}_n$.
4. $\mathbf{X}^2 = n\mathbf{X}$[55].
5. $\mathbf{X}_{ij}\mathbf{X}_{jk} = \mathbf{X}_{ik}$[55].

这些性质之间有很强的关系. 例如, 性质 1+3 和原约束是等价的. 注意到, 半正定矩阵 $\mathbf{X}$ 秩为 1 当且仅当其可以分解为形如 $\mathbf{x}\mathbf{x}^\top$ 的形式, 而性质 3 又给出 $x_i^2 = 1$, 从而和原问题等价. 若只选取性质 3, 即可得到视角 1 给出的松弛4.2.2. 文献 [55] 则利用性质 4, 5 给出了更强的半定规划松弛, 但代价是把问题的维度进一步提升到 n^2 维上, 并不实用. 因

此, 实际设计松弛问题时, 需要在松弛问题的困难程度和近似度之间做好平衡.

在拉格朗日对偶视角下, 我们首先把4.2.1的目标函数转化为二次形式 $\min \mathbf{x}^\top \mathbf{W}\mathbf{x}$, 然后把整数约束 $x_i \in \{-1, 1\}$ 转化为 $x_i^2 = 1$, 然后直接对这些约束引入拉格朗日乘子 μ_i, 拉格朗日函数为

$$\begin{aligned} L(\mu) &= \min_{\mathbf{x}} \mathbf{x}^\top \mathbf{W}\mathbf{x} - \sum_i \mu_i x_i^2 + \sum_i \mu_i \\ &= \min_{\mathbf{x}} \mathbf{x}^\top (\mathbf{W} - \mathrm{Diag}(\mu))\mathbf{x} + \sum_i \mu_i. \end{aligned}$$

注意到,

$$\min_{\mathbf{x}} \mathbf{x}^\top (\mathbf{W} - \mathrm{Diag}(\mu))\mathbf{x} = \begin{cases} 0 & \text{若} \mathbf{W} - \mathrm{Diag}(\mu) \succeq 0, \\ -\infty & \text{否则}. \end{cases}$$

因此, 其拉格朗日对偶为

$$\max \sum_i \mu_i \text{ s.t. } \mathbf{W} \succeq \mathrm{Diag}(\mu).$$

这是一个对偶形式的半定规划问题, 且其对偶即为4.2.1.

4.2.1.2　含等式约束

很多组合优化问题关联的整数规划问题还额外包含若干等式约束, 如图分割问题 (Graph Partitioning), 在最大割问题的基础上还额外要求分成的部分顶点数量相等. 由于变量之间的等式约束无法转化为关于松弛向量的约束, 这类问题不能用向量松弛求解. 文献 [56] 从拉格朗日对偶视角出发, 对这类问题提出了一个通用的松弛方法.

1. 把整数约束更换为二次约束, 如 $x_i \in \{0, 1\}$ 重写为 $x_i^2 = x_i$. 把等式约束更换为关于其范数的平方二次约束, 如 $\mathbf{A}\mathbf{x} = \mathbf{b}$ 更换为 $||\mathbf{A}\mathbf{x} - \mathbf{b}||^2 = 0$. 这样, 问题的约束全部转化为二次约束.
2. 写出二次规划问题的拉格朗日函数.
3. 对拉格朗日函数做齐次化.
4. 使用隐藏半定约束获得对偶半定规划问题.
5. 对对偶半定规划问题再做一次对偶, 得到原问题的半定规划松弛.

我们以图分割问题为例解释这种方法. 设 G 为带权 $\mathbf{A}$ 的无向图. 给定 $m_1 \geq m_2 \geq \cdots \geq m_k \geq 1$. 图分割问题即把顶点集分为 k 份, 每份的顶点数量分别为 $m_1, \cdots, m_k$, 并且最小化这些部分之间的权重和.

记 $\mathbf{m} = (m_1, \cdots, m_k)^\top$, 我们选取 $n \times k$ 维矩阵 $\mathbf{X}$ 为

$$\mathbf{X}_{ij} = \begin{cases} 1 & \text{若} i \in S_j, \\ 0 & \text{否则}. \end{cases}$$

显然, 任意的 $n \times k$ 维矩阵 $\mathbf{X}$ 对应一个分割当且仅当其同时满足三个约束, $\mathbf{X}_{ij} \in \{0,1\}$, $\mathbf{Xe_k} = \mathbf{e_n}$, $\mathbf{X}^\top \mathbf{e_n} = \mathbf{m}$. 注意到, $(\mathbf{X}^\top \mathbf{AX})_{ii} = \sum_{j,k} \mathbf{X}_{ji}\mathbf{A}_{jk}\mathbf{X}_{ki}$, 这恰好计算了同属于图的第 i 个分割内部的边权重. 由于图内部所有边的权重和为定值, 因此最小化顶点集之间的权重和就相当于最大化内部的边权重. 这样, 我们得到图分割题的整数规划表示

$$\begin{aligned} \max \quad & \frac{1}{2}\operatorname{Tr}\mathbf{X}^\top\mathbf{AX} \\ \text{s.t.} \quad & \mathbf{Xe_k} = \mathbf{e_n} \\ & \mathbf{X}^\top\mathbf{e_n} = \mathbf{m} \\ & \mathbf{X}_{ij} \in \{0,1\}. \end{aligned}$$

下面我们导出其半定规划松弛. 根据步骤 1, 我们首先把约束转化为二次形式

$$\begin{aligned} \max \quad & \frac{1}{2}\operatorname{Tr}\mathbf{X}^\top\mathbf{AX} \\ \text{s.t.} \quad & \|\mathbf{Xe_k} - \mathbf{e_n}\|^2 + \left\|\mathbf{X}^\top\mathbf{e_n} - \mathbf{m}\right\|^2 = 0 \\ \forall i,j \quad & \mathbf{X}_{ij}^2 - \mathbf{X}_{ij} = 0. \end{aligned}$$

根据步骤 2, 对约束引入对偶变量 $\alpha, \mathbf{W}$, 我们写出拉格朗日函数

$$\begin{aligned} B_{GP} = \min_{\alpha,W}\max_{\mathbf{X}} \operatorname{Tr}\Big[&\frac{1}{2}\mathbf{X}^\top\mathbf{AX} + \alpha\left(\mathbf{e_k}\mathbf{e_k}^\top\mathbf{X}^\top\mathbf{X} + \mathbf{X}^\top\mathbf{e_n}\mathbf{e_n}^\top\mathbf{X}\right) + \mathbf{W}^\top\mathbf{X}\circ\mathbf{X} \\ &-2\alpha\left(\mathbf{e_k}\mathbf{e_n}^\top\mathbf{X} + \mathbf{m}\mathbf{e_n}^\top\mathbf{X}\right) - \mathbf{W}^\top\mathbf{X}\Big] + \alpha\left(n + \sum_i m_i^2\right), \end{aligned}$$

其中, $\mathbf{X}\circ\mathbf{X}$ 是指 $\mathbf{X}$ 的对应元素直接相乘得到的矩阵.

根据步骤 3, 为了消去拉格朗日函数中引入的关于 $\mathbf{X}$ 的一次项, 我们引入新的约束 $x^2 = 1$, 用 x 作为系数, 对拉格朗日函数做齐次化. 注意到, 对二次函数而言一次项的系数是 +1 或-1 不影响极值, 因为任意解 $\mathbf{X}$, $-\mathbf{X}$ 具有对称性. 因此,

$$\begin{aligned} B_{GP} = \min_{\alpha,W}\max_{\mathbf{X}} \operatorname{Tr}\Big[&\frac{1}{2}\mathbf{X}^\top\mathbf{AX} + \alpha\left(\mathbf{e_k}\mathbf{e_k}^\top\mathbf{X}^\top\mathbf{X} + \mathbf{X}^\top\mathbf{e_n}\mathbf{e_n}^\top\mathbf{X}\right) + \mathbf{W}^\top\mathbf{X}\circ\mathbf{X} + \delta x^2 \\ &x(-2\alpha\left(\mathbf{e_k}\mathbf{e_n}^\top\mathbf{X} + \mathbf{m}\mathbf{e_n}^\top\mathbf{X}\right)) - \mathbf{W}^\top\mathbf{X}\Big] + \alpha\left(n + \sum_i m_i^2\right) - \delta. \end{aligned} \tag{4.2.3}$$

根据步骤 4, 我们利用二次问题的性质导出隐含二次规划问题. 由于现在 Tr 算子内部的内容是关于变量 $\mathbf{X}, x$ 的二次函数, 问题实际上形如 $\max_{\mathbf{y}} \mathbf{y}^\top\mathbf{Qy}$. 该二次问题的解为

$$\begin{cases} 0 & \text{若 } \mathbf{Q} \preceq 0, \mathbf{y} = \mathbf{0} \text{ 时取得,} \\ +\infty & \text{否则.} \end{cases}$$

根据这个条件, 我们用 $\mathbf{X}$ 的向量展开把4.2.3中 Tr 的内部重整为上面 $\mathbf{y}^\top\mathbf{Qy}$ 的形式, 即可

导出原问题隐含的半定规划问题

$$\begin{aligned}
\min \quad & \alpha\left(n+\textstyle\sum_i m_i^2\right)-\delta \\
\text{s.t.} \quad & \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{I}\otimes\mathbf{A} \end{bmatrix} + \begin{bmatrix} \delta & -\frac{1}{2}(\mathrm{vec}(W))^\top \\ -\frac{1}{2}(\mathrm{vec}(\mathbf{W})) & \mathrm{diag}(\mathrm{vec}(\mathbf{W})) \end{bmatrix} + \\
& \alpha\begin{bmatrix} 0 & -(\mathbf{e}+\mathbf{v})^\top \\ -(\mathbf{e}+\mathbf{v}) & \left(\mathbf{e_k}\mathbf{e_k}^\top\mathbf{I}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{e_n}\mathbf{e_n}^\top\right) \end{bmatrix} \le 0,
\end{aligned}$$

其中, 矩阵算子 vec 即把给定的矩阵按照列展开后得到的向量, $\otimes$ 为矩阵的 Kronecker 积, $\mathbf{v}=\mathrm{vec}\,\mathbf{e_n}\mathbf{m}^\top$.

最后, 按照步骤 5, 再求一次对偶即可得到原问题的半定规划松弛. 取 $\mathbf{Y}$ 为对偶变量, 得对偶半定规划问题

$$\begin{aligned}
\max \quad & \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{I}\otimes\mathbf{A} \end{bmatrix} \bullet \mathbf{Y} \\
\text{s.t.} \quad & \mathrm{diag}(\mathbf{Y}) = \left(1, \mathbf{Y}_{0,1:n}\right)^\top \\
& \mathbf{Y}\bullet\begin{bmatrix} 0 & -(\mathbf{e}+\mathbf{v})^\top \\ -(\mathbf{e}+\mathbf{v}) & \left(\mathbf{e_k}\mathbf{e_k}^\top\mathbf{I}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{e_n}\mathbf{e_n}^\top\right) \end{bmatrix} = 0 \\
& \mathbf{Y}\succeq 0.
\end{aligned}$$

4.2.1.3　含不等式约束

半定规划的方法很难推广到含一般不等式约束的组合优化问题上. 现有的工作一般都是对于不等式约束仅包含关于变量的二次项的问题做讨论, 如 [57] 等. 因此, 对于非二次约束问题, 一般需要先把约束转化或松弛为二次约束. 从约束松弛视角出发, 人们提出了多种把线性约束松弛为二次约束的方法, 下面列举其中三种最常见的方法.

- **对角提升** 利用整数规划的条件直接给出二次约束. 例如, 问题给定 $x_i\in\{0,1\}$, 则关于 x_i 的线性约束, 如 $\mathbf{a}^\top\mathbf{x}\le b$, 可以直接转化为 $\mathbf{a}^\top[\mathbf{x}]^2\le b$ ($[\mathbf{x}]^2\triangleq\mathbf{x}\circ\mathbf{x}=[x_1^2,\cdots,x_n^2]^\top$). 这样, 对 $\mathbf{x}$ 的约束转化为对矩阵 $\mathbf{X}=\mathbf{x}\mathbf{x}^\top$ 的约束 $\mathrm{Diag}(\mathbf{a})\bullet\mathbf{X}\le b$.
- **平方提升** 把线性约束 $0\le\mathbf{a}^\top\mathbf{x}\le b$(一般组合优化问题的不等式会隐含一个非负下界) 松弛为 $|\mathbf{a}^\top\mathbf{x}|\le\mathbf{b}$, 从而有 $(\mathbf{a}\mathbf{a}^\top)\bullet(\mathbf{x}\mathbf{x}^\top)\le b^2$. 与上面相同, 即为关于 $\mathbf{X}$ 的约束 $(\mathbf{a}\mathbf{a}^\top)\bullet\mathbf{X}\le b^2$.
- **Lovász-Schrijver 提升** 文献 [54] 提出, 对任意 $j\in[n]$, 直接对约束 $0\le\mathbf{a}^\top\mathbf{x}\le b$ 两边都分别乘以 x_j 和 $1-x_j$, 得到关于 $\mathbf{X}$ 的 $2n$ 个不等式约束. 这种松弛无疑是相当紧的, 但是计算代价也很大.

这里, 我们需要阐述一个可能的误区. 由于整数约束必定有 $x_i^2=x_i$, 我们上面对 $\mathbf{x}$ 的二次转化单独看是等价的. 但是, 我们的整体目标是对 $\mathbf{x}$ 求解半定规划问题. 因此, 我

们之后将变量 $\mathbf{x}\mathbf{x}^\top$ 更改为 $\mathbf{X}$ 时相当于又做了一步松弛. 因此, 尽管关于 $\mathbf{x}$ 的二次转化是相同的, 但是其视为关于 $\mathbf{X}$ 的约束则有松有紧.

事实上, 文献 [57] 证明了这步松弛的相对误差有界.

定理 4.2.1. *([57], 定理 3.3)*

假设 $m^*(\mathbf{A}) = \max \sum_{ij} \mathbf{A}_{ij} x_i x_j \ s.t.\ \mathbf{B}[\mathbf{x}]^2 \le \mathbf{c}$, $m_*(\mathbf{A}) = \max \sum_{ij} \mathbf{A}_{ij} x_i x_j \ s.t.\ \mathbf{B}[\mathbf{x}]^2 \le \mathbf{c}$, 其将 xx^T 松弛为 $\mathbf{X}$ 后的问题的解记为 $s^*(\mathbf{A}) = \max_{\mathbf{X}\ge 0, \mathbf{B}d(\mathbf{X})\le \mathbf{c}} \sum_{ij} \mathbf{A}_{ij}\mathbf{X}_{ij}$, $s_*(\mathbf{A}) = \max_{\mathbf{X}\ge 0, \mathbf{B}d(\mathbf{X})\le \mathbf{c}} \sum_{ij} \mathbf{A}_{ij}\mathbf{X}_{ij}$, 则

$$s_*(\mathbf{A}) \le m_*(\mathbf{A}) \le s_{1-\alpha^*}(\mathbf{A}) \le s_{\alpha^*}(\mathbf{A}) \le m^*(\mathbf{A}) \le s^*(\mathbf{A}),$$

其中 $s_\alpha = \alpha s^* + (1-\alpha) s_*$.

此外, 这步松弛也导致不同的松弛方法的松紧程度不同. 文献 [58] 证明, 平方提升的松弛比对角提升更紧, 而 Lovász-Schrijver 提升则比平方提升更紧.

我们以最大稳定集问题说明如何利用这些方法导出半定规划问题的松弛. 回忆上一节的定义, 图 G 中的顶点集 S 称为稳定的, 若其中任意两个顶点互不相邻. 我们希望求出顶点数量最大的稳定集. 用 $x_i \in \{0,1\}$ 表示顶点是否在 S 中, 则最大稳定集问题的整数规划形式为

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} x_i \\ \text{s.t.} \forall (i,j) \in E \quad & x_i + x_j \le 1 \\ \forall i \in [n] \quad & x_i \in \{0,1\}. \end{aligned}$$

注意到, 整数约束 $x_i \in \{0,1\}$ 等价于二次约束 $x_i^2 = x_i$. 首先把目标函数转化为二次形式 $\sum_{i=1}^n x_i^2$. 若定义 $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$, 则目标函数改写为 $\mathbf{I} \bullet \mathbf{X}$. 然后, 利用以下三种方式转化线性不等式约束.

1. **对角提升** 对任意 $(i,j) \in E$, $x_i^2 + x_j^2 \le 1$, 即 $\mathrm{Diag}(\mathbf{e}_i + \mathbf{e}_j) \bullet \mathbf{X} \le 1$.
2. **平方提升** 把约束两边平方, 得到 $x_i^2 + 2x_i x_j + x_j^2 \le 1$, 即 $(\mathbf{E}_{ii} + \mathbf{E}_{jj} + 2\mathbf{E}_{ij}) \bullet \mathbf{X} \le 1$.
3. **Schirijver 提升** 把每个约束扩展为 $(x_i + x_j)x_k \le 1$ 和 $(x_i + x_j)(1 - x_k) \le 1$, 即 $(\mathbf{E}_{ik} + \mathbf{E}_{jk}) \bullet \mathbf{X} \le 1$, 且 $(\mathbf{E}_{ii} + \mathbf{E}_{jj} - \mathbf{E}_{ik} - \mathbf{E}_{kj}) \bullet \mathbf{X} \le 1$.

最后, 利用第4.2.1.1节的约束松弛视角, 把整数约束也转化为关于 $\mathbf{X}$ 的约束即可. 例如, 转化为 $\mathrm{diag}(\mathbf{X}) = \mathbf{1}_n$.

4.2.2 松弛解的舍入

上一节提出了众多方法来给出松弛问题, 但是松弛问题终归只是给出了原问题的一个上/下界. 如果我们想获得原问题的一个解, 那么还需要给出把松弛解归约为原问题

解的方法. 目前唯一的松弛解舍入方法是由 [6] 提出的随机分离超平面方法.

这个方法本身是基于向量化视角4.2.1.1提出的. 注意到, 我们在松弛时是把整数变量 $x_i \in \pm 1$ 松弛为 n 维向量 $\mathbf{v}_i$, 现在决定将 $\mathbf{v}_i$ 还原为 +1 还是 −1, 实质上就是对向量集 $\mathbf{v}_i$ 做一个二分类问题. 最简单的分类就是直接用超平面分成两半 (支持向量机). 文献 [6] 提出, 在舍入时直接取随机单位向量 $\mathbf{a}$, 计算 $\hat{x}_i = \mathrm{sign}(\mathbf{a}^\top \mathbf{v}_i)$ 即可. 其中, sign 即符号函数, 将非负数映射为 +1, 负数映射为-1.

该方法最大的优点在于其舍入出的原始解有很好的理论保障. 具体而言, 可以证明, 舍入解在期望上至少达到原问题最优解的 0.878 倍. 我们下面详述该证明.

考虑权重非负的最大割问题4.2.1的松弛半定规划问题4.2.2. 假设松弛解为 $\hat{\mathbf{X}}$, 我们可以利用正交分解将其还原为 $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_\mathbf{1}, ..., \hat{\mathbf{v}}_\mathbf{n}]$. 根据约束 $\mathrm{diag}(\mathbf{X}) = \mathbf{1}_n$, 即 $|\mathbf{v}_i| = 1$, 我们得到单位向量 $\hat{\mathbf{v}}_\mathbf{1}, ..., \hat{\mathbf{v}}_\mathbf{n}$. 现在, 随机选取单位向量 $\mathbf{a}$, 并且令 $\hat{x}_i = \mathrm{sign}(\mathbf{a}^\top \hat{\mathbf{v}}_\mathbf{i})$, 则其为原始问题题4.2.1的一组解, 且其给出的目标函数值为 $\frac{1}{4}\sum_{i,j} w_{ij}(1 - \hat{x}_i\hat{x}_j)$.

这里最关键的是注意到, $1-\hat{x}_i\hat{x}_j = 2\Pr(\hat{x}_i \neq \hat{x}_j)$. 而事件 $\hat{x}_i \neq \hat{x}_j$ 相当于随机向量 $\mathbf{a}$ 和 $\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j$ 的夹角一个大于 90 度, 一个小于 90 度. 但是, 由于 $\mathbf{a}$ 是随机单位向量, $\Pr(\hat{x}_i \neq \hat{x}_j)$ 的期望值仅与 $\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j$ 的夹角有关, 即 $\frac{\arccos(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j)}{\pi}$. 取 $\alpha = \min_{0\le\theta\le\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} \approx 0.878$, 则 $\frac{\arccos(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j)}{\pi} \geq \alpha\frac{1}{2}(1 - \hat{\mathbf{v}}_i^\top \hat{\mathbf{v}}_j)$. 综上, 有

$$\mathsf{E}_\mathbf{a} w_{ij}\left[\frac{1}{4}\sum_{i,j}(1 - \hat{x}_i\hat{x}_j)\right] \geq \alpha\frac{1}{4}\sum_{i,j} w_{ij}(1 - \hat{\mathbf{v}}_i^\top \hat{\mathbf{v}}_j).$$

不等式右端即为 0.878 倍的松弛问题的目标函数值. 由于松弛问题的值大于等于原问题, 因此, GW 算法舍入的解 $\hat{x}_1, \cdots, \hat{x}_n$ 在期望上大于等于最佳值的 0.878 倍.

事实上, 有证据[59]表明 0.878 可能是最大割问题的最好理论期望上界. 同时, 这个舍入技巧也被广泛用在其他基于半定规划问题的组合优化问题中, 例如文献 [60] 利用这个技巧和一些统计方法给出了类似组合优化问题的下界, 如带源和汇的最大割问题 (要求某两个顶点必须不在同一个部分中) 等.

不过, 半定规划问题正在失去组合优化问题的求解方法的主流地位. 这主要是因为 20 世纪后组合优化问题的各种启发式算法开始蓬勃发展, 尤其是大量基于物理方法的调整法、低秩方法[61]、连续化迭代法等[62]方法. 这些方法虽然没有半定规划松弛的理论保障, 但是实际效果不论是近似度还是运算效率上都要好得多. 但是, 随着近年来半定规划的低秩近似算法的提出3.5和一系列设计更高效半定规划松弛的方法的提出 (如发掘问题的稀疏性和对称性[63]), 半定规划有望重新在运算效率上追赶这些启发式算法.

4.3 半定规划问题的其他应用

当然, 半定规划的应用是非常广泛的, 组合优化问题只是它最重要的应用之一. 下面, 我们列举半定规划问题在其它领域的应用, 并给出相关综述和重要论文供读者参考.

- **通信**、**控制** 论文 [64-66], 综述 [52] 第十四、十五章.
- **随机优化**、**统计** 论文 [67], 综述 [52] 第十六章, 十七章.
- **矩阵填充问题** 论文 [68-69], 综述 [52] 第十八章.
- **机器学习** 论文 [70], 综述 [63].

第五章　结论与展望

本文介绍了半定规划问题的基本形式与概念, 总结了半定规划问题的历史算法的发展, 并选取其中具有代表性的四个算法做了介绍. 随后, 本文系统介绍了导出组合优化问题的半定规划近似的技巧以及近似解的舍入技巧.

虽然半定规划问题在效率上逐渐落后于启发式算法, 但是近年来机器学习反哺的对近似算法的研究让半定规划问题正在重新焕发生机. 笔者认为, 目前半定规划领域的发展主要集中在三个方向.

1. 拓展半定规划问题在实用领域的应用.
2. 利用问题特性, 对应用问题导出更加有针对性的半定规划近似, 如利用稀疏性 (3.5)、对称性 ([63]) 等.
3. 仿照近似条件梯度增广拉格朗日算法, 导出半定规划问题更加高效的算法.

参考文献

[1] KHACHIYAN A. A Polynomial Algorithm in Linear Programming[J]. Soviet Math. Dokl., 1979, 20: 191-194.

[2] GRÖTSCHEL M, LOVÁSZ L, SCHRIJVER A. The Ellipsoid Method and Its Consequences in Combinatorial Optimization[J]. Combinatorica, 1981, 1: 169-197.

[3] ALIZADEH F. Optimization over the Positive-Definite Cone: Interior Point Methods and Combinatorial Applications[J]. Advances in optimization and parallel computing, 1992.

[4] KAMATH A, KARMARKAR N. A Continuous Approach to Compute Upper Bounds in Quadratic Maximization Problems with Integer Constraints[G]//Recent Advances in Global Optimization. 1992: 125-140.

[5] KAMATH A P, KARMARKAR N K. An O (nL) Iteration Algorithm for Computing Bounds in Quadratic Optimization Problems[G]//Complexity in Numerical Optimization. World Scientific, 1993: 254-268.

[6] GOEMANS M X, WILLIAMSON D P. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming[J]. Journal of the ACM, 1995, 42(6): 1115-1145 [2023-03-19]. DOI: 10.1145/227683.227684.

[7] CHU M T, FUNDERLIC R E, PLEMMONS R J. Structured Low Rank Approximation[J]. Linear algebra and its applications, 2003, 366: 157-172.

[8] DRINEAS P, MAHONEY M W, CRISTIANINI N. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning.[J]. journal of machine learning research, 2005, 6(12).

[9] 修乃华, 罗自炎. 半定规划[M]. 北京交通大学出版社, 2014.

[10] 文再文. 对偶理论, 凸优化最优性条件[Z]. 2021.

[11] BOT R I, GRAD S M, WANKA G. Duality in Vector Optimization[M]. Springer Science & Business Media, 2009.

[12] BOYD S P, VANDENBERGHE L. Convex Optimization[M]. Cambridge university press, 2004.

[13] ARORA S, KALE S. A Combinatorial, Primal-Dual Approach to Semidefinite Programs[C]// Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing. 2007: 227-236.

[14] HUANG B, JIANG S, SONG Z, et al. A Faster Quantum Algorithm for Semidefinite Programming via Robust IPM Framework[J]. arXiv preprint arXiv:2207.11154, 2022. arXiv: 2207.11154.

[15] WEN Z, GOLDFARB D, YIN W. Alternating Direction Augmented Lagrangian Methods for Semidefinite Programming[J]. Mathematical Programming Computation, 2010, 2: 203-230. DOI: 10.1007/s12532-010-0017-1.

[16] YANG L, SUN D, TOH K C. SDPNAL+: A Majorized Semismooth Newton-CG Augmented Lagrangian Method for Semidefinite Programming with Nonnegative Constraints[J]. Mathematical Programming Computation, 2015, 7(3): 331-366.

[17] YURTSEVER A, FERCOQ O, CEVHER V. A Conditional-Gradient-Based Augmented Lagrangian Framework[C]//International Conference on Machine Learning. PMLR, 2019: 7272-7281.

[18] JIANG H, LEE Y T, SONG Z, et al. An Improved Cutting Plane Method for Convex Optimization, Convex-Concave Games, and Its Applications[C]//Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. 2020: 944-953.

[19] VAIDYA P M. A New Algorithm for Minimizing Convex Functions over Convex Sets[C]//30th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, 1989: 338-343.

[20] KRISHNAN K, MITCHELL J E. Properties of a Cutting Plane Method for Semidefinite Programming[J]. submitted for publication, 2003.

[21] LEE Y T, SIDFORD A, WONG S C W. A Faster Cutting Plane Method and Its Implications for Combinatorial and Convex Optimization[C]//2015 IEEE 56th Annual Symposium on Foundations of Computer Science. IEEE, 2015: 1049-1065.

[22] JIANG H, KATHURIA T, LEE Y T, et al. A Faster Interior Point Method for Semidefinite Programming[Z]. 2020. arXiv: 2009.10217 [cs, math] [2023-05-23].

[23] VANDENBERGHE L, BOYD S. Semidefinite Programming[Z]. 1996.

[24] HELMBERG C, RENDL F. A Spectral Bundle Method for Semidefinite Programming[J]. SIAM Journal on Optimization, 2000, 10(3): 673-696.

[25] YURTSEVER A, TROPP J A, FERCOQ O, et al. Scalable Semidefinite Programming[J]. SIAM Journal on Mathematics of Data Science, 2021, 3(1): 171-200 [2023-04-13]. DOI: 10.1137/19M1305045.

[26] FERNANDO M D O F. Lecture Notes on Semidefinite Programming, the Ellipsoid Method[Z]. 2014 [2023-04-28].

[27] NESTEROV Y, NEMIROVSKII A. Interior-Point Polynomial Algorithms in Convex Programming [M]. SIAM, 1994.

[28] 田文娟. 半定规划的原对偶内点算法[D]. 西安电子科技大学, 2014.

[29] De KLERK E. Interior Point Methods for Semidefinite Programming[M]. 1997.

[30] NESTEROV Y. Polynomial-Time Iterative Methods in Linear and Quadratic Programming[J]. Voprosy kibernetiki, Moscow, 1988: 102-125.

[31] FAYBUSOVICH L. On a Matrix Generalization of Affine-Scaling Vector Fields[J]. SIAM Journal on Matrix Analysis and Applications, 1995, 16(3): 886-897.

[32] FAYBUSOVICH L. Semidefinite Programming: A Path-Following Algorithm for a Linear–Quadratic Functional[J]. SIAM Journal on Optimization, 1996, 6(4): 1007-1024.

[33] ANSTREICHER K M. On Long Step Path Following and SUMT for Linear and Quadratic Programming[J]. SIAM Journal on Optimization, 1996, 6(1): 33-46.

[34] ZHANG Y. On Extending Some Primal–Dual Interior-Point Algorithms From Linear Programming to Semidefinite Programming[J]. SIAM Journal on Optimization, 1998, 8(2): 365-386.

[35] ALIZADEH F, HAEBERLY J P A, OVERTON M L. Primal-Dual Interior-Point Methods for Semidefinite Programming: Convergence Rates, Stability and Numerical Results[J]. SIAM Journal on Optimization, 1998, 8(3): 746-768.

[36] HELMBERG C, RENDL F, VANDERBEI R J, et al. An Interior-Point Method for Semidefinite Programming[J]. SIAM Journal on optimization, 1996, 6(2): 342-361.

[37] KOJIMA M, SHINDOH S, HARA S. Interior-Point Methods for the Monotone Semidefinite Linear Complementarity Problem in Symmetric Matrices[J]. SIAM Journal on Optimization, 1997, 7(1): 86-125.

[38] BRANDÃO F G S L, KALEV A, LI T, et al. Quantum SDP Solvers: Large Speed-Ups, Optimality, and Applications to Quantum Learning[J]. 2019: 14 pages [2023-03-07]. DOI: 10.4230/LIPICS.ICALP.2019.27.

[39] VAN APELDOORN J, GILYÉN A, GRIBLING S, et al. Quantum SDP-Solvers: Better Upper and Lower Bounds[J]. Quantum, 2020, 4: 230. arXiv: 1705.01843 [quant-ph] [2023-03-07]. DOI: 10.22331/q-2020-02-14-230.

[40] CESA-BIANCHI N, LUGOSI G. Prediction, Learning, and Games[M]. Cambridge university press, 2006.

[41] KALE S. Efficient Algorithms Using The Multiplicative Weights Update Method[J]. 2007.

[42] NOCEDAL J, WRIGHT S J. Numerical Optimization[M]. Springer, 1999.

[43] TROPP J A, YURTSEVER A, UDELL M, et al. Fixed-Rank Approximation of a Positive-Semidefinite Matrix from Streaming Data[J]. Advances in Neural Information Processing Systems, 2017, 30.

[44] KUCZYŃSKI J, WOŹNIAKOWSKI H. Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start[J]. SIAM journal on matrix analysis and applications, 1992, 13(4): 1094-1122.

[45] PATAKI G. On the Rank of Extreme Matrices in Semidefinite Programs and the Multiplicity of Optimal Eigenvalues[J]. Mathematics of operations research, 1998, 23(2): 339-358.

[46] ALIZADEH F, HAEBERLY J P A, OVERTON M L. Complementarity and Nondegeneracy in Semidefinite Programming[J]. Mathematical programming, 1997, 77(1): 111-128.

[47] ANUPAM G, RYAN O. Advanced Algorithms: Linear and Semidefinite Programming, Lecture 11 [Z]. 2011.

[48] LOVÁSZ L. Semidefinite Programs and Combinatorial Optimization[G]//REED B A, SALES C L. Recent Advances in Algorithms and Combinatorics. New York, NY: Springer New York, 2003: 137-194 [2023-04-25]. DOI: 10.1007/0-387-22444-0_6.

[49] GAREY M R, JOHNSON D S. “strong”np-Completeness Results: Motivation, Examples, and Implications[J]. Journal of the ACM (JACM), 1978, 25(3): 499-508.

[50] KNUTH D E. The Sandwich Theorem[J]. The Electronic Journal of Combinatorics, 1994, 1(1): A1.

[51] JUHÁSZ F. The Asymptotic Behaviour of Lovász’theta Function for Random Graphs[J]. Combinatorica, 1982, 2(2): 153-155.

[52] WOLKOWICZ H, SAIGAL R, VANDENBERGHE L, et al. Handbook of Semidefinite Programming: vol. 27[M]. Boston, MA: Springer US, 2000 [2023-03-22]. DOI: 10.1007/978-1-4615-4381-7.

[53] GAREY M R, JOHNSON D S, STOCKMEYER L. Some Simplified NP-complete Problems[C]//

Proceedings of the Sixth Annual ACM Symposium on Theory of Computing. 1974: 47-63.

[54] LOVÁSZ L, SCHRIJVER A. Cones of Matrices and Set-Functions and 0–1 Optimization[J]. SIAM journal on optimization, 1991, 1(2): 166-190.

[55] ANJOS M F, WOLKOWICZ H. Strengthened Semidefinite Relaxations via a Second Lifting for the Max-Cut Problem[J]. Discrete Applied Mathematics, 2002.

[56] POLJAK S, RENDL F, WOLKOWICZ H. A Recipe for Semidefinite Relaxation for (0,1)-Quadratic Programming: In Memory of Svata Poljak[J]. Journal of Global Optimization, 1995, 7(1): 51-73 [2023-03-27]. DOI: 10.1007/BF01100205.

[57] NESTEROV Y. Semidefinite Relaxation and Nonconvex Quadratic Optimization[J]. Optimization Methods and Software, 1998, 9(1-3): 141-160 [2023-03-19]. DOI: 10.1080/10556789808805690.

[58] HELMBERG C, RENDL F, WEISMANTEL R. Quadratic Knapsack Relaxations Using Cutting Planes and Semidefinite Programming[C]//Integer Programming and Combinatorial Optimization: 5th International IPCO Conference Vancouver, British Columbia, Canada, June 3–5, 1996 Proceedings 5. Springer, 1996: 175-189.

[59] KHOT S, KINDLER G, MOSSEL E, et al. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?[J]. SIAM Journal on Computing, 2007, 37(1): 319-357 [2023-03-31]. DOI: 10.1137/S0097539705447372.

[60] BERTSIMAS D, YE Y. Semidefinite Relaxations, Multivariate Normal Distributions, and Order Statistics[J]. Handbook of Combinatorial Optimization: Volume1–3, 1998: 1473-1491.

[61] BURER S, MONTEIRO R D C, ZHANG Y. Rank-Two Relaxation Heuristics for MAX-CUT and Other Binary Quadratic Programs[J]. SIAM Journal on Optimization, 2002, 12(2): 503-521 [2023-03-31]. DOI: 10.1137/S1052623400382467.

[62] SHAO S, ZHANG D, ZHANG W. A Simple Iterative Algorithm for Maxcut[Z]. 2023. arXiv: 1803.06496 [cs, math] [2023-03-31].

[63] MAJUMDAR A, HALL G, AHMADI A A. A Survey of Recent Scalability Improvements for Semidefinite Programming with Applications in Machine Learning, Control, and Robotics[Z]. 2019. arXiv: 1908.05209 [cs, eess, math] [2023-04-23].

[64] APKARIAN P, ADAMS R J. Advanced Gain-Scheduling Techniques for Uncertain Systems[G]// Advances in Linear Matrix Inequality Methods in Control. SIAM, 2000: 209-228.

[65] EL GHAOUI L, LEBRET H. Robust Solutions to Least-Squares Problems with Uncertain Data[J]. SIAM Journal on matrix analysis and applications, 1997, 18(4): 1035-1064.

[66] LAVAEI J, LOW S H. Zero Duality Gap in Optimal Power Flow Problem[J]. IEEE Transactions on Power Systems, 2011, 27(1): 92-107.

[67] LASSERRE J B. Global Optimization with Polynomials and the Problem of Moments[J]. SIAM Journal on optimization, 2001, 11(3): 796-817.

[68] MAZUMDER R, HASTIE T, TIBSHIRANI R. Spectral Regularization Algorithms for Learning Large Incomplete Matrices[J]. The Journal of Machine Learning Research, 2010, 11: 2287-2322.

[69] CANDES E J, PLAN Y. Matrix Completion with Noise[J]. Proceedings of the IEEE, 2010, 98(6): 925-936.

[70] LANCKRIET G R, CRISTIANINI N, BARTLETT P, et al. Learning the Kernel Matrix with Semidefinite Programming[J]. Journal of Machine learning research, 2004, 5(Jan): 27-72.