第 1 章

イントロダクション

1.1 数理最適化

数理最適化問題 (または単に最適化問題) は以下の形で表される。

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq b_i, \quad i = 1, \cdots, m \end{aligned} \tag{1.1}$$

ここで、ベクトル $x = (x_1, \cdots, x_n)$ は問題の最適化変数、$f_0 : \mathbb{R}^n \to \mathbb{R}$ は問題の目的関数、$f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \cdots, m$ は (不等形の) 制約関数、定数 $b_1, \cdots, b_m$ は制約関数の限界 (または境界) である。この問題の解、または最適値はベクトル x^* で表され、制約を満たす全てのベクトルの中で最小の目的値となる。式で示すと、$\forall z$ with $f_1(z) \leq b_1, \cdots, f_m(z) \leq b_m$ の時、$f_0(z) \geq f_0(x^*)$ を満たすベクトル x^* が問題の解となる。一般的には、目的関数と制約関数が特定の形式で特徴付けることが出来る最適化問題のクラスや族について考える。重要な例として**??**節と**??**節で示す最小二乗法や線形計画問題がある。例えば線形計画問題では、もし目的関数と制約関数 $f_0, \cdots, f_m$ が線形である場合、以下を満す。

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y) \tag{1.2}$$

ここで、$\forall x, y \in \mathbb{R}^n, \forall \alpha, \beta \in \mathbb{R}$ である。問題が線形ではない場合は非線形計画問題と呼ばれる (**??**で少し触れる)。本書で扱うのは凸最適化問題と呼ばれる問題であり、目的関数と制約関数が凸面である問題である。凸面であるということは以下を満たすことを言う。

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{1.3}$$

ここで、$\forall x, y \in \mathbb{R}^n, \forall \alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$ である。(1.2) と (1.3) を比較すると、凸面は線形性の一般化であることが分かる。

1.1.1 応用

最適化問題はある候補選択の集合から最良の選択となるようなベクトル x^* を選ぶ問題の抽象化と考えることが出来る。変数 x は選択した候補、制約関数 $f_i(x) \leq b_i$ は選択肢を制限するような不変的な要件や定義、目的関数 $f_0(x)$ は x を選択するコストである。$-f_0(x)$ として x を選択する効果や有効性と考えることもできる。この解は全ての制約を満たす選択の中で最小のコスト (または最大の有用性) を持つ選択に対応している。

例 1. ポートフォリオを例にとると、n 個の資産の集合に資本を分配して投資する際の最良の方法を探すのが最適化の目的となる。式 (1.1) で x_i は i 番目の資産への投資を表し、ベクトル $x \in \mathbb{R}^n$ は資産の集合全体への投資割り当てを示している。ここでの制約は予算の限度、投資額が非負であること、ポートフォリオ全体の期待収益額に対する最小の許容値 *(*つまり儲けは最小でどれだけいるか*)* である。目的 *(*またはコスト*)* 関数はポートフォリオ全体のリターンに対するリスク、分散の尺度が該当する。この問題の場合は、全ての不変的な制約を満す割り当てから最小のリスクとなる割り当てを選ぶ問題に対応する。

他にもデバイスサイズ (デバイスの幅、長さの最適化) やデータフィッティング (得られたデータに最も良く一致するモデルを選択する) などに応用可能であり、意思決定 (システム設計、解析、運用) を含む非常に多くの実世界の問題はこの最適化問題や多目的最適化問題の形に当てはめることができる。これらの応用の大部分で数理最適化はプロセスの監督、結果のチェック、必要に応じて問題の修正を行う人間の意思決定、システム設計、システム運用の補助を行う。意思決定を行う人間は、数理最適化によって示唆された行動を実際に実行する。

最近では数理最適化は組込みでの応用が急増している。このような組込み最適化では最適化を用いてリアルタイムでの選択と関連する操作を自動的に実行するため、ほとんど人間の手や監視は介さないで実行される。ただ、組込み最適化には非常に高い信頼性を維持しつう予測可能な時間で問題を解決する手法が必要という課題もある。

1.1.2 最適化問題の解法

最適化問題のクラスに対する解法は、クラスから特定の問題、つまり問題のインスタンスが与えられた時に与えられた精度で問題を解くアルゴリズムである。1940 年代後半以降、様々なクラスの最適化問題を解くアルゴリズムの開発、それらの特性解析、そして良いソフトウェア実装に多大な努力が払われてきた。これらのアルゴリズムの有用性は大きく変化し、特定の形の目的、制約関数や制約と変数の数、スパース性み代表される特徴的な構造のような要因に依存している。ここでいうスパース性は、問題の制約関数が少数の変数にのみ依存するような問題を言う。

たとえ目的関数や制約関数が十分滑らかであっても (多項式のように)、一般的な最適化問題は非常に解くのが困難である。そのため一般的な問題へのアプローチは計算時間が非常に長くなったり真の解が見つからない可能性などの妥協がある。しかし一般的に難しいとされる最適化問題の中でも、ある種の問題では数百〜数千の変数がある場合でも効率的に解くことが出来るアルゴリズムが存在する。その効率的に解けるアルゴリズムの中でも 2 つの重要で良く知られている例が 1.2節で述べる最小二乗法と線形計画問題である。またあまり知られていないが、本書で扱う凸最適化問題も同様に効率的に解くことが出来るアルゴリズムが存在する。

1.2 最小二乗法と線形計画問題

1.2.1 最小二乗法

最小二乗法は制約関数が無く $(m = 0)$、目的関数が $a_i^T x - b_i$ の形式の項の二乗和で表される最適化問題である。

$$\text{minimize} \quad f_0(x) = \|Ax - b\|_2^2 = \sum_{i=1}^{k} (a_i^T x - b_i)^2 \tag{1.4}$$

ここで、行列 $A \in \mathbb{R}^{k \times n}$ with $k \geq n$, A の行を表す a_i^T、ベクトル $x \in \mathbb{R}^n$ は最適化変数である。

最小二乗法の解

最小二乗法の解は線形方程式の集合を解くことに帰着する。

$$(A^T A)x = A^T b \tag{1.5}$$

ここから x について解くと、$x = (A^T A)^{-1} A^T b$ となる (正規方程式)。この問題は高精度かつ非常に高い信頼性で解くことが出来るアルゴリズムが存在し、既知の定数 $n^2 k$ に比例する時間で解くことができる。これは現在のデスクトップコンピュータを用いても 100 個の変数と 1000 個の項であれば数秒で解けてしまう。

係数行列 A が特殊な構造を持つ場合、より大きな問題を解くことが可能となる。例えば行列 A がスパース性を持つ場合、上で示した計算量 $n^2 k$ よりもはるかに速く解くことができる。

さらに大きな問題を解く場合やリアルタイム性が要求される厳しい場面では問題を解くのは困難であるが、知られている解法は非常に高効率かつ高信頼性であり、詳細を知らない人でも確実に利用できる。

最小二乗法の利用

最小二乗法は回帰分析、最適制御、多くのパラメータ推定とデータフィッティングの基礎となる。フィッティングの例で言えば、ガウスの誤差関数によって欠損した線形測定データが与えられた場合にベクトル x の最尤測定として様々な統計的解釈を与えてくれる。最小二乗法形式の最適化問題を認識するのは簡単であり、目的関数が二次形式かつ関連する二次形式が半正定値であることを確認すればよい (多分最小値を取る点で傾きが 0 になるため、他の点では二乗しているので傾きは正になる (?))。最小二乗法の基本的な形式は非常に単純である一方、いくつかの標準的なテクニックを用いて応用の幅を広げているのもある。

その 1 つとして重み付き最小二乗法がある。これは式にある基準の重みを付け、

$$\sum_{i=1}^{k} w_i (a_i^T x - b_i)^2 \tag{1.6}$$

を最小化する。ここで、$w1, \cdots, w_k$ は重みである。この重みは項 $a_i^T x - b_i$ のサイズに対する様々な懸念 (関心) を反映するように、または単に解に影響を与えるために設定される。統計的には不均一な分散を持つ誤差によって損われた線形測定データが与えられた場合のベクトル x を推定する際に生じる。

もう 1 つのテクニックとして正則化がある。これはコスト関数に追加の項を置くことでモデルに対して罰則を設ける方法である。簡単なケースでは、変数の平方和にある倍数をかけたものが追加される。

$$\sum_{i=1}^{k} (a_i^T x - b_i)^2 + \rho \sum_{i=1}^{n} x_i^2 \tag{1.7}$$

ここで、$\rho > 0$ である。追加の項は変数 x が大きい場合に罰則が加わり、結果として最初の項の和のみを解としない場合よりもより良い解となる。追加項に付くパラメータ ρ は元の目的関数を小さくしつつ、追加項を大きくしすぎないようにする適切なトレードオフを与えるようにユーザによって選ばれる。統計的には、事前分布をベクトル x に与えることで正則化が生じるようになる。