最適化手法入門
6章後半

金栄世

2022年5月26日

6　近似解法と発見的手法

6.3　劣モジュラ最大化問題

6.3.1　劣モジュラ関数

集合関数とは, ある集合 V の全ての部分集合それぞれに対して一つの実数値を割り当てる関係のことをいう. このとき, V をその集合関数の**台集合**いい, V の部分集合の集合を V の**べき集合**という. f が V を台集合とする集合関数であるとき, $f : 2^V \to \mathbb{R}$ と書く.

集合関数の例

例として, $V =$ {' ご飯', ' 焼肉'} という集合について, どれくらい嬉しいかという値を集合関数 f とする.
この集合 V のべき集合は, { {}, {' ご飯'}, {' 焼肉'}, {' ご飯', ' 焼肉'} } である.
それぞれの部分集合が持つ「嬉しさ」は例えば, 下のようになる.
$f(\{\}) = 0$, $f(\{'ご飯'\}) = 1$, $f(\{'焼肉'\}) = 5$, $f(\{'ご飯','焼肉'\}) = 10$

劣モジュラ関数とは限界効用逓減の法則に従う集合関数である. 限界効用とは, 財 (モノ、およびサービス) を 1 単位追加して消費することによる効用 (満足度) の増加分のことであり, 限界効用逓減の法則とは, 財の量が増加するにつれてその追加分から得られる効用は次第に減少するという法則である. これを式で表すと,

$$h(s + \Delta x) - h(s) \geq h(t + \Delta x) - h(t) \ \ (s < t) \tag{1}$$

のようになる. 限界効用逓減の法則に従う関数の例を下の Fig. 1 に示す.

つまり劣モジュラ関数は, 元を一つ追加することによってもたらされる関数値の増加量が, 追加前の集合が大きくなるにつれて減少するような関数のことをいう. 正確な定義としては, 集合関数 $f : 2^V \to \mathbb{R}$ が劣モジュラ関数であるとは, 任意の $S \subset T (\subset V)$ と任意の $j \in (V - T)$ に対して以

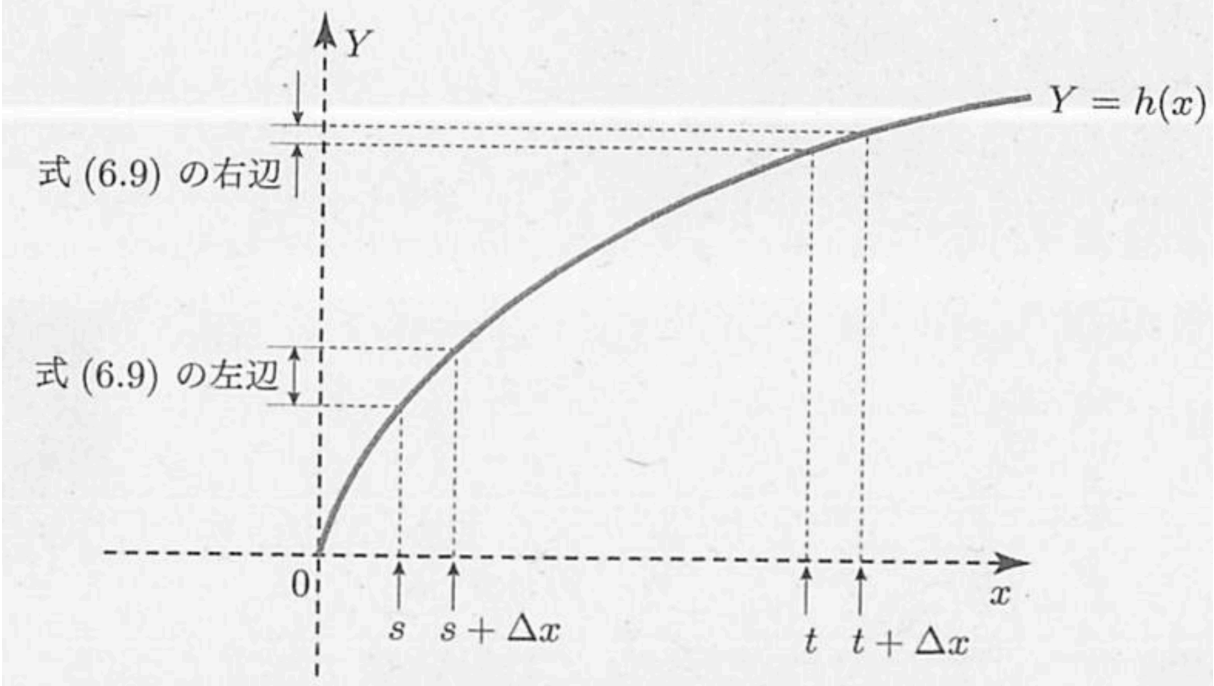

図 6.8　単調増加な凹関数と限界効用逓減の法則

Fig. 1: example of diminishing marginal utility

下の条件が成り立つことをいう. (下の二式は等価である)

$$f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T) \tag{2}$$

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \tag{3}$$

劣モジュラ関数 $f : 2^V \to \mathbb{R}$ が任意の $S \subset T (\subseteq V)$ に対して条件 $f(S) \geq f(T)$ を満たすとき, f は**単調**であるという. また, 劣モジュラ関数 $f : 2^V \to \mathbb{R}$ が任意の $S \subset T (\subseteq V)$ に対して条件 $f(S) = f(V - S)$ を満たすとき, f は**対称**であるという. 劣モジュラ関数の例として, 教科書で図形の集合とグラフカット容量が紹介されている.

6.3.2　劣モジュラ最大化に対する貪欲算法

劣モジュラ関数最適化問題は以下のように表現される.

$$\text{Maximize} \quad f(S) \tag{4}$$

$$\text{subject to} \quad |S| \leq m, \tag{5}$$

$$S \subseteq V \tag{6}$$

$|S|$ は集合 S の元の数を表し, m は正の定数を表す. 単調な劣モジュラ関数では S に含まれる元の数が多いほど $f(S)$ の値は大きいので, 元に関する制約がなければ $S = V$ になってしまうので, 式 (4) の制約が設けられている. 以上のように定式化された問題は, 基数制約付き単調劣モジュラ関数最大化問題, 単に劣モジュラ最大化問題と呼ばれる.

劣モジュラ最大化問題は NP 困難であることが知られていて, 効率の良いアルゴリズムはそうそう存在しないと考えられているが, 以下のような単純な貪欲算法が 0.63-近似解法であることが知られている. このアルゴリズムでは, 集合に含まれていない要素のうち, 追加した場合の効果が最も高いものを集合に追加して更新していく.

Algorithm 1 劣モジュラ最大化問題の貪欲算法

1: $S_0 \leftarrow \emptyset$
2: **for** $j = 1 \ldots m$ **do**
3: $\quad j^* \leftarrow \mathrm{argmax}\{f(S_{k-1} \cup \{j\}) | j \in V\}$
4: $\quad S_k \leftarrow S_{k-1} \cup \{j^*\}$
5: $\quad V \leftarrow V - \{j^*\}$
6: **end for**
7: S_mを近似解として出力する

以下では劣モジュラ最大化問題の目的関数 f は $f(\emptyset) = 0$ を満たすとし, 問題の最適解を $\bar{S}$ と表現する. 劣モジュラ最大化問題においては以下のような定理が成り立つ. (証明は教科書を参照)

定理 6.2

アルゴリズム 1 が生成する $S_0, S_1, \ldots, S_m$ に対して,

$$f(\bar{S}) - f(S_k) \leq (1 - \frac{1}{m})(f(\bar{S}) - f(S_{k-1})), \qquad k = 1, \ldots, m \tag{7}$$

が成り立つ.

ここで, 定理 6.2 と $f(\emptyset) = 0$ より, 次の不等式が得られる.

$$\begin{aligned} f(\bar{S}) - f(S_m) &\leq (1 - \frac{1}{m})(f(\bar{S}) - f(S_{m-1})) && (8)\\ &\leq (1 - \frac{1}{m})^2(f(\bar{S}) - f(S_{m-2})) && (9)\\ &\leq \ldots \leq (1 - \frac{1}{m})^m f(\bar{S}) && (10) \end{aligned}$$

従って, 不等式

$$\frac{f(S_m)}{f()\bar{S}} \geq 1 - (1 - \frac{1}{m})^m = 1 - \frac{1}{(1 + \frac{1}{m-1})^m} \geq 1 - \frac{1}{e} \geq 0.63 \tag{11}$$

が成り立つので, アルゴリズム 1 は 0.63-近似解法であると言える.

6.3.3 応用: 文書要約

劣モジュラ関数の応用として, **文書要約**がある. 文書要約は, 一つの長い文書が与えられてその要約を作るという**単文書要約**と, ある一つのテーマに関する複数の文書が与えられてその要約を作るという**複数文書要約**に大別される.

文書要約の手法の一つとして, **文選択法**がある. これは文書を一つ一つの文に分解し, その中からいくつか重要な文を抽出することで要約を作るという方法である. つまり, 文書を文の集合とみなし, その部分集合として要約を生成する手法である. これより, 文選択法における要約とは, 制限された要素数で最も内容を充実させるような部分集合を求める問題, 劣モジュラ最大化問題として定式化できる.

以下では, 要約を作る際に元の文書の内容がなるべく多く伝わるように文を選択するという具体例について考える. まず文書に含まれる文の集合を V とおく. 次に, 今選ばれている文の集合を S(Fig. 2 の (b) の太いところ) とおく. また, S に少し文が追加された集合を T(Fig. 2 の (c) の太いところ) とおく. 選ばれている文の集合がどれだけ内容を充実させているかという値を集合関数 $f(S), f(T)$ で表すと, $S \subset T$ より $f(S) \leq f(T)$ が成り立つのが一般的である. ここで, 最適化が進むとして元の文書集合 V のうちで集合 V に含まれていない文 j を追加する場合を考える. この時, 一般には $f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T)$ となる. このように, 要約の評価尺度 f に対して単調な劣モジュラ関数が現れることになる. 一般に要約に用いられる文の数には制限があるので, このことから文書要約は劣モジュラ最大化問題となる. (評価尺度に関しては最適化ではなく自然言語処理の話になるので, 小池くんか小坂くんに聞いてください)

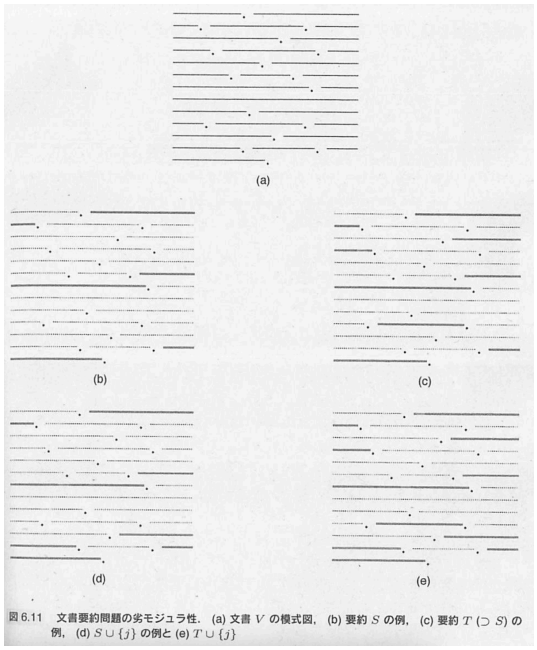

図 6.11 文書要約問題の劣モジュラ性. (a) 文書 V の模式図, (b) 要約 S の例, (c) 要約 T ($\supset S$) の例, (d) $S \cup \{j\}$ の例と (e) $T \cup \{j\}$

Fig. 2: example of summarizing document

6.4　メタ戦略

6.4.1　メタ戦略の基本的な考え方

メタ戦略とは, 発見的解法を様々な初期点から実行したり局所最適解を発見した後も探索を続けるように工夫をするなど, 個別の最適化問題の性質に依存しない一般的な枠組みを通じて, より質の良い解が得られることを目指す戦略のことを言う. メタ戦略という言葉は具体的な解法を示すわけではなく, 解法を設計するための様々なアイデアの集合体を指している. メタ戦略に基づく解法では, 通常, 単純な発見的解法よりも大きな計算コストを費やすことで, より質の良い解を得ようとする. メタ戦略に基づく解法では求める解の精度 (最適性や近似比) が保証されていないという点で, 発見的解法に分類される. メタ戦略の多くは, 大きく**集中化**と**多様化**という二つの方策を, バランスをとりながら組み合わせている.

集中化とは, 優れた解の近くにはやはり優れた解がある可能性が高いだろうと考えて, これまでに得られた解のうちで良い解の近くを集中的に探索する方策である. 局所探索法は, 集中かを実行する典型的な手法であり, 多くのメタ戦略で採用されている. 一方で集中かの考え方のみで解法を設計すると, ごく一部にある領域にある回のみを探索することになり, より優れた解がそこから離れた領域にあったとしてもそれを見つけることはできない.

多様化とは, これまでに探索されていない領域にはより優れた解が存在し得ると考えて, これまでに得られた優れた解とは異なる性質を持つ解の探索を試みる方策である. 多様化を実現する方法には, 複数の初期解を用いる方法や, 目的関数値が悪くなる解へも移動してみる方法, これまでに得られた複数の解を組み合わせることで新しい解を得る方法などがある.

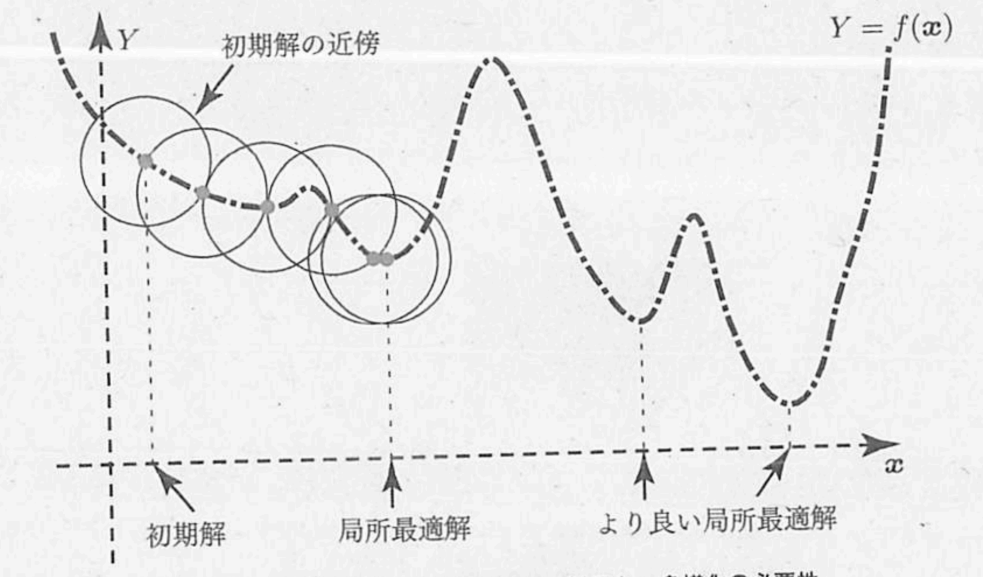

図 6.12　局所探索法による解の探索の様子と，多様化の必要性

Fig. 3: example of meta

6.4.2 多スタート局所探索法

多スタート局所探索法とは, 様々な初期解で局所探索を実行し, それぞれの初期解から得られた局所最適解のうちで最良のものを最終的に解として出力するという解法である. 多スタート局所探索法は局所探索法の実装があれば簡単に実現できるものであるので, 単純な局所探索法では良い解が得られない場合にはまず試してみるべきメタ戦略であるといえる.

6.4.3 模擬焼きなまし法

様々な初期解を用意するほかに, 局所探索法に多様化の方策を組み合わせるもう一つの代表的な方策として, 今得られている解よりも目的関数値の悪い解 (改悪解) に移動することを許すというものがあり, この考え方に基づくメタ戦略の一つとして, 模擬焼きなまし法がある. 模擬焼きなまし法 (Simulated Annealing) は, 金属の焼きなまし過程という物理現象にアイディアを得て, **温度**というパラメータ $t > 0$ を用いて改悪解への移動を制御する仕組みを作るという手法である. 模擬焼きなまし法を用いて最適解を求める手順は, 次のようになっている. まず今得られている解 x の近傍 $N(x)$ の中から他の解 y を一つ選ぶ. 以上の手順をまとめると, アルゴリズム 2 のように記述できる.

Algorithm 2 模擬焼きなまし法

Require: 初期解 x, 初期温度 $t > 0$, 温度調整のパラメータ γ $(0 < \gamma < 1)$

for $k = 1, 2 \ldots$ **do**
  ランダムに $y \in N(x)$ を選ぶ
  $\Delta \leftarrow f(y) - f(x)$
  if $\Delta \leq 0$ **then**
    $X \leftarrow y/x$
  else
    確率 $e^{-\Delta/t}$ で $x \leftarrow y$ とする
  end if
  $t \leftarrow \gamma t$
end for

アルゴリズム 2 を見ると, 改悪会へ移動する確率 $e^{-\Delta/t}$ は, 改悪量 Δ が小さいほど 1 に近く, 大きいほど 0 に近いことがわかる. つまり, 今得られている解と目的関数値が近いほど, 高い確率で改悪解へと移動する. 一方で温度 t に関しては, 値が大きいほど改悪解へ移動する確率が大きくなり, 小さいと局所探索法と同様の振る舞いをすることになる. つまり, 温度 t はループを繰り返すにつれて小さくなっていくので, 値が大きい初期段階では改悪解への移動をより多く許して解の探索範囲が広げられ, 探索の終盤では集中化が行われるようになる. このように温度 t が解法の振る舞いを制御しているので, 模擬焼きなまし法で得られる解の質は初期温度の値や温度の減らし方に大きく影響される. アルゴリズム 2 では単純な減らし方としたが, 実際には様々な工夫が提案されている.

6.4.4 タブー探索法

タブー探索法は過去に訪れた解を記憶して, 再度訪れないようにすることで多様化を目指す手法である. タブー探索法で, 近い過去に訪れた解の記憶は**タブーリスト**と呼ばれる. タブーリストとして保持する解の個数を定数とし, 解 x の近傍を $N(x)$ と表すと, タブー探索法はアルゴリズム 3 のように表現できる. 実際には, タブーリストの長さを探索状況に応じて適応的に変化させる方法や, 解の更新方法をタブーリストに記憶する方法など, 様々な工夫が提案されている.

Algorithm 3 タブー探索法

Require: 初期解 x, タブーリスト $T = \emptyset$ とその長さの上限値 l

1: **for** $k = 1, 2 \ldots$ **do**
2: $\quad |T| = l$ ならば, T から最も古い解を除く
3: $\quad T \leftarrow T \cup \{x\}$
4: $\quad$ 集合 $N(x) - T$ に含まれる解のうち, 最良のものを x とする
5: **end for**

参考文献

[1] 集合関数について (https://manabitimes.jp/math/1114)

[2] 焼きなまし工程について (https://www.keyence.co.jp/ss/products/recorder/heat/basics/type.jsp)

[3] 模擬焼きなまし法について (http://isw3.naist.jp/IS/Bio-Info-Unit/gogroup/masata-o/PDF/graduation_thesis.pdf)

[4] タブー探索について (http://www.orsj.or.jp/archive2/or58-12/or58_12_703.pdf)