動的計画法

稲毛惇人

2017 年 7 月 7 日

1　はじめに

変数が離散値を取る関数の最大，最小化は一般には全ての組み合せを調べなければならない．しかし，関数が 2 変数関数の和に表せる場合は，“動的計画法” によって最適解を効率的に計算できる．また，このような問題は “最適経路問題” とみなせ，“最適性の原理” が成立する．これをパタン認識，音声認識，画像認識などの人工知能の多くの問題で基礎となる “ストリングマッチング” に応用する方法や，制約条件がある場合の処理法を紹介する．

ここで，“決定の全系列にわたって最適化を行うためには，初期の状態と最初の決定がどんなものであっても，残りの決定は最初の決定から生じた状態に関して最適な政策を構成していなければならない” ということを，“最適性の原理” と呼ぶ．別の表現をすれば，“最適経路中の部分経路もまた最適経路になっている” ということを意味する．

2　多段階決定問題

ある各変数 x_i が有限である n 個の離散値 $\{a_1, ..., x_n\}$ をとる場合に，n 変数関数 $f(x_1, ..., x_n)$ を最大化する問題を考える．

$$J = f(x_1, ..., x_n) \to \max \tag{1}$$

このとき，各変数がとる値の組み合わせは N^n 通りあり，これらすべて調べなければならない．しかし，N, n が大きくなると，関数を評価する回数が膨大になり，計算機でも効率的に計算することが困難になる．

一方，関数 f の形によってはそのような全数検査が避けられる場合がある．代表的なの

は，f が次のような対ごとの 2 変数関数の和の場合である．

$$J = f_1(x_1) + h_1(x_1, x_2) + h_2(x_2, x_3) + \cdots + h_{n-1}(x_{n-1}, x_n) \to \max \qquad (2)$$

変数 x_i の値を選ぶことを何らかの “決定” とみなせば，これは段階的に次々と決定を下す問題とみなせる．まず初期に x_1 の値を選ぶことに対する利益が関数 f_1 で与えられる．初期に選んだ x_1 に対して次に x_2 を選ぶことの利益が $h_1(x_1, x_2)$ である．同様に，第 i 段階で選んだ x_i 似た逸して第 $i+1$ 段階で x_{i+1} を選ぶ利益が $h_x(x_i, x_{i+1})$ である．そして，式 2 は “利益の総和が最大になるように各段階で決定を下せ” という問題であると解釈できる．

2.1 グリーディ（貪欲）法

上記のような多段決定問題に対して次のような “解法 (?)” が考えられる．

解法 (?)

1. $f_1(x_1)$ が最大になる x_1 を選ぶ．
2. その x_1 に対して $h_1(x_1, x_2)$ が最大になる x_2 を選ぶ．
3. その x_2 に対して $h_2(x_2, x_3)$ が最大になる x_3 を選ぶ．

   ⋮

4. その x_{n-1} に対して $h_{n-1}(x_{n-1}, x_n)$ が最大になる x_n を選ぶ．

一見これで良さそうであるが，これでは近視眼的すぎて最適解が得られる保証がない．例えば，初期の段階で損をしても後の段階で大きな利益が得られるのなら，そのほうが最終的に利益が大きいからである．

ここに示した “解法 (?)” は探索の各時点で常に目的関数を最大にするように変数を選択するものであり，“グリーディ（貪欲）法” と呼ばれている．しかし，これによって最適解が得られるのは，線形計画などの，問題に特殊な性質があるのみである．ほとんどのゲームでは，途中で損をするような選択をしたほうが最終的な利益が大きいことが多い（そうでないと “読み” が不要になり，ゲームとしてつまらない）．例として囲碁では途中で石を捨て，将棋では大駒を切って最終的に勝てることがよくある．

2.2 動的計画法

このような問題を効率的に解く方法が**動的計画法（ダイナミックプログラミング）**である．動的計画法の原理は単純である．式 2 でまず x_1 に着目する．これは $f_1(x_1)$ と

$h_1(x_1, x_2)$ にしか関係しない．したがって，$f_1(x_1) + h_1(x_1, x_2)$ を最大にするように x_1 を選べばよい．しかし，そのためには x_2 の値がわかっていなければならない．そこで，x_2 の**可能なすべての値**に対して適切な x_1 をそれぞれ計算する．これを x_2 の**関数とみなして** $\hat{x}_1(x_2)$ と書く．その $f_1(x_1) + h_1(x_1, x_2)$ の最大値も x_2 に依存するので，それを x_2 の**関数とみなして**

$$f_2(x_2) = \max_{x_1} [f_1(x_1) + h_1(x_1, x_2)] \tag{3}$$

と定義する．これを用いると，式 2 は次のように書ける．

$$J = f_2(x_2) + h_2(x_2, x_3) + \cdots + h_{n-1}(x_{n-1}, x_n) \to \max \tag{4}$$

これは式 2 と**同じ形であり，かつ変数の数がひとつ減っている**．同様に進み，最終的に関数 $f_n(x_n)$ を

$$f_n(x_n) = \max_{x_{n-1}} [f_{n-1}(x_{n-1}) + h_{n-1}(x_{n-1}, x_n)] \tag{5}$$

と定義し，最大値を与える x_{n-1} を $\hat{x}_{n-1}(x)$ と置く．すると式 4 は次のように書ける．

$$J = f_n(x_n) \to \max \tag{6}$$

この 1 変数関数 $f_n(x_n)$ が最大になる x_n の値を x_n^* とする．これに対する最適な x_{n-1} の値は $x_{n-1}^* = \hat{x}_{n-1}(x_n^*)$ である．以下，**逆にたどって** x_{n-2} の最適な値 $x_{n-2}^* = \hat{x}_{n-2}(x_{n-1}^*), \cdots, x_1$ の最適な値 $x_1^* = \hat{x}_1(x_2^*)$ が決定できる．上記の手順を以下に示す．

動的計画法のアルゴリズム

procedure $DP(f_1(x), \{h_i(x, y)_{i=1}^n\})$

1. $i = 1, \cdots, n-1$ に対して，関数

   $$f_{i+1}(x_{i+1}) = \max_{x_i} [f_i(x_i) + h_i(x_i, x_{i+1})]$$

   およびその最大値を与える x_i を関数 $\hat{x}_i(x_{i+1})$ として**数表の形**で定義する．
2. $f_n(x_n)$ を最大にする x_n の値 x_n^* を探索し，その最大値を $J^* = f_n(x_n^*)$ とする．
3. $i = n-1, \cdots, 1$ に対して $x_i^* = \hat{x}_i(x_{i+1}^*)$ を計算する．
4. $(x_1^*, x_2^*, \cdots, x_n^*)$ および J^* を返す．

3　ストリングマッチング

パタン認識，音声認識，画像認識などの人工知能の多くの問題で重要なのがデータ文字列をデータベース中の文字列にマッチングさせる問題である．次の例を考える．

ストリングマッチングの 3 変化

1. 文字が別の文字に置換される．（置き換え）
2. 文字が脱落する．（脱落）
3. 文字が挿入される．（挿入）

上記の 3 種類の変化が組み合わさってある入力文字列が入力されたとき，出力文字列が出力されるとする．二つの文字列が与えられたとき，1,2,3 がどのように生じたかをマッチングによって表す問題を**ストリングマッチング**という．どのような変化が起きたかは入力文字列の各文字を出力文字列の各文字に対応させる**マッチング**によって記述できる．これを一般化して，入力文字列 $a_1, \cdots, a_n$ を出力文字列 $b_1, \cdots, b_m$ に対応させるマッチングを考える．このとき，入力文字 a_i を出力文字 x_i に対応させるとする．x_i は $b_1, \cdots, b_m$ のどれかであるが，重複していたり，対応する b_i が存在しなくてもよい．

例として，以下の入力文字列と出力文字列との対応を考える．

$$(S, a, b, a, b, a, c, d) \to (S, x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (S, a, c, a, a, b, b, d) \quad (7)$$

まず，入力文字列と出力文字列の冒頭に開始文字列 S が付加され，$a_0 = S, b_0 = S$ とする．そして，$a_1 \to x_1, \cdots, a_n \to x_n$ となった結果，$b_1, \cdots, b_m$ が得られたとする．

- 開始文字列 S は S に対応し，次の $a_1 = a$ が $x_1 = a$ に同一文字で対応する．
- 次の $a_2 = b$ が $x_2 = c$ に異なる文字（置換）で対応する．
- 次の $a_3 = a$ が $x_4 = a$ に一つ飛んで（同じ文字 a が挿入され）対応する．
- 次の $a_4 = a$ が $x_5 = a$ に同一文字で対応する．
- $x_4 = b$ と $x_5 = b$ が出力文字列の同じ位置にあるので a_5 は脱落で対応する．
- $x_5 = b$ と $x_6 = b$ が出力文字列の同じ位置にあるので a_6 は脱落で対応する．
- 次の $a_7 = d$ が $x_7 = d$ に同一文字で対応する．

問題は $x_1, \cdots, x_n$ を適切に定めることである．これを解く代表的な方法は，文字の置換や脱落や挿入が生じる確率を求め，出力文字に対する入力文字列の確からしさの**評価関数**を導入して，最適化問題に帰着することである．

これに対して，例えば次のような評価関数を定義する．

$$p_i(x_i) = \begin{cases} 0.8 & x_i \neq x_{i-1} \text{ かつ } x_i \text{ が } a_i \text{ と同一文字のとき} \\ 0.2 & x_i \neq x_{i-1} \text{ かつ } x_i \text{ が } a_i \text{ と異なる文字のとき} \\ 0 & \text{それ以外} \end{cases} \tag{8}$$

$$q(x_i, x_{i+1}) = \begin{cases} 0.1 & \text{ある } j \text{ に対して } x_i = b_j, x_{i+1} = bj \text{（脱落）のとき} \\ 0.79 & \text{ある } j \text{ に対して } x_i = b_j, x_{i+1} = bi+1 \text{（脱落・挿入なし）のとき} \\ 0.1 & \text{ある } j \text{ に対して } x_i = b_j, x_{i+1} = bi+2 \text{（1 個挿入）のとき} \\ 0.01 & \text{ある } j \text{ に対して } x_i = b_j, x_{i+1} = bi+3 \text{（2 個挿入）のとき} \\ 0 & \text{それ以外} \end{cases} \tag{9}$$

このとき，問題は次の関数 J の最大化問題となる．

$$J = p_1(x_1) + p_2(x_2) + q(x_1, x_2) + p_3(x_3) + q(x_2, x_3) + \cdots + p_n(x_n)q(x_{n-1}, x_n) \to \max \tag{10}$$

以上より，式 10 の評価は次のようになる．

$$J = 0.8+0.2+0.79+0.8+0.01+0.8+0.79+0+0.1+0+0.1+0.8+0.79 = 6.07 \tag{11}$$

また，式 10 において

$$f_1(x_1) = p_1(x_1) \tag{12}$$

$$h_i(x_i, x_{i+1}) = p_{i+1} + q(x_i, x_{i+1}), i = 1, \cdots, n-1 \tag{13}$$

と置けば，式 10 は式 2 のように書くことができる．つまり，この解は動的計画法によって計算できる．

ストリングマッチングが動的計画法に帰着するということは，これが最適経路問題としても表せることを意味している．このとき，関数 $p_i(x_i)$ は**対応する文字が類似しているほど大きい**評価を与える役割をし，関数 $q(x_i, x_{i+1})$ は**経路の傾きが 1 に近いほど大きい**役割を果たしている．

ストリングマッチングの動的計画法による解法は，パタン認識，音声認識，画像認識などの人工知能の問題のみならず，最近では医学，生物学，遺伝子学に関連して遺伝子（DNA）の解析にも用いられる．

4　制約のある多段階決定問題

実際の場で現れる多段階決定問題では，目的関数を最大にする際に，変数間に制約条件があることがある．そのような問題には動的計画法が適用できないように思えるが，多く

の場合に適当小名変形によって制約条件が除去され，問題が式 2 の形に帰着する．このような工夫によって，一見動的計画法が適用できないように見える問題でも動的計画法が適用できる代表的な例を考える．

次の制約のある最大化問題の解を考える．

$$J = g_1(x_1) + g_2(x_2) + \cdots + g_n(x_n) \to \max \tag{14}$$

$$x_1 + x_2 + \cdots + x_n = M \tag{15}$$

$$x_1 \geq 0, x_2 \geq 0, \cdots, x_n \geq 0 \tag{16}$$

制約条件を取り除くために，新しい変数 $y_1, \cdots, y_n$ を次の**部分和**によって定義する．

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_1 + x_2 \\ y_3 &= x_1 + x_2 + x_3 \\ &\vdots \\ y_n &= x_1 + x_2 + \cdots + x_n \end{aligned} \tag{17}$$

もとの変数 $x_1, \cdots, x_n$ は次のように，部分和の差によって表せる．これを代入すると、式 14〜17 は次のように書ける．

$$J = g_1(y_1) + g_2(y_2 - y_1) + \cdots + g_n(y_n - y_{n-1}) \to \max \tag{18}$$

$$y_1 \geq 0, y_2 \geq y_1, \cdots, y_n \geq y_{n-1}, y_n = M \tag{19}$$

式 19 の不等式制約は，関数 $f_1(y_1), h_1(y_i, y_{i+1})$ を次のように定義すれば除去できる．

$$f_1(y_1) = \begin{cases} g_1(y_1) & y_1 \geq 0 \text{ のとき} \\ \perp & \text{それ以外} \end{cases} \tag{20}$$

$$h_i(y_i, y_{i+1}) = \begin{cases} g_{i+1}(y_{i+1} - y_i) & y_{i+1} \geq y_i \text{ のとき} \\ \perp & \text{それ以外} \end{cases} \quad i = 1, \cdots, n-2 \tag{21}$$

$$h_{n-1}(y_{n-1}, y_n) = \begin{cases} g_n(M - y_{n-1}) & y_n = M \text{ のとき} \\ \perp & \text{それ以外} \end{cases} \tag{22}$$

ここで，$\perp$ は関数 f_1, h_i に対する各変数 $y_1, (y_i, y_{i+1})$ が与えられたとき，“定義されない値” を示す．このように $\perp$ を関数の取り得る “値” の 1 つとみなすことによって動的計画法の応用範囲が広がる．また，すべての値に対して $\perp$ をとる関数を “空関数” と呼ぶ．式

2 に関して，$f_1(x_1)$ が与えられていない場合は f_1 を空関数として常に式 2 の形に表すものとする．これらを用いると，問題は制約のない最大化

$$J = f_1(y_1) + h_1(y_1, y_2) + h_2(y_2, y_3) + \cdots + h_{n-1}(y_{n-1}, x_y) \to \max \tag{23}$$

となる．$f_i(x_i)$ や $h_i(y_i, y_{i+1})$ が定義されない（$\perp$ をとる）$y_1, \cdots, y_n$ の値に対しては J の値が定義されないので，その大小も比較されない．それ以外は動的計画法の手順がそのまま適用できる．これを解き最適解 $y_1^*, \cdots, y_n^*$ が求まれば，もとの変数の最適解 $x_1^*, \cdots, x_n^*$ は式 17 を変形すれば定まる．

参考文献

[1] 金谷健一, “これなら分かる最適化数学”, 共立出版株式会社, 2012.