DCアルゴリズムを用いたl_0-ノルム最小化

DC Optimization Approaches for l_0-norm Minimization

経営システム工学専攻　山本 信

1　はじめに

近年，最適化や情報学の分野でスパースモデリングが注目されている [1]．スパースモデリングとは，少ない情報から全体像を的確にあぶり出すモデリングのことであり，ビッグデータなどの大規模なデータであっても本質的には少数の説明変数しか存在しないというスパース性に着目している．このスパースモデリングにおいて，スパース性を測る指標としてl_0-ノルム（対象となるベクトルの非ゼロな要素の総数）が使われている．スパースモデリングの一例として，最もスパースな解を選ぶという最適化問題がある．これは非ゼロな要素の数をできるだけ減らす最適化問題である．特に目的関数にl_0-ノルムのみを持つ最適化問題はl_0-ノルム最小化問題と呼ばれ，本質的に組合わせ最適化であり，高次元のデータに対して，そのまま総当たり的に最適解を求めること（厳密解法）は実用時間で解くことが難しいとされている．このため，最適化手法ではl_0-ノルム最小化問題はNP困難な問題とされ，l_0-ノルムをl_1-ノルムに置き換えて解を求める方法が多く研究されている．

そこで本論文では，l_0-ノルム最小化問題について考え，それを先行研究 [4] で提示されている2つの凸関数の差（DC）で表される関数の最小化問題を反復的に解く枠組みを適用し，効率的な解法の提案を行う．これにより，l_0-ノルム最小化問題を厳密解法よりも時間をかけずに，l_1-ノルムで緩和した解よりも非ゼロ要素の数が少ない解を得られることを示す．

2　線形モデルとスパース性

始めに以下のような線形回帰モデルについて説明する．

$$y = w_1x_1 + w_2x_2 + \cdots + w_px_p + \epsilon \qquad (1)$$

y は被説明変数，x_j $(j = 1,2,\cdots,p)$ は説明変数，w_j $(j = 1,2,\cdots,p)$ は求めたい未知のパラメータで，ϵ は残差である．線形回帰では，このパラメータ w_j を求めることで，説明変数 x_j から被説明変数 y を予測することを考えている．いま，$(x_1,\cdots,x_p,y)$ の n 組の実現値 $(x_{1i},\cdots,x_{pi},y_i)$, $(i = 1,...,n)$ が観測されているとする．このとき，このデータセットを (1) 式に当てはめたときの誤差は残差と呼ばれ，以下のように定義される:

$$\boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$$

ただし，

$\boldsymbol{y} := (y_1\ y_2\ \cdots\ y_n)^\top$，$\boldsymbol{w} := (w_1\ \cdots\ w_p)^\top$，

$\boldsymbol{\epsilon} := (\epsilon_1\ \epsilon_2\ \cdots\ \epsilon_n)^\top$，

$$\boldsymbol{X} := \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

である．

また，未知のパラメータ $\boldsymbol{w}$ の持つ役割の一つとして，説明変数を採用するかしないかを決めるというものがある．たとえば，$w_j = 0$ とすれば，説明変数 x_j にどのような値が入っていても (1) の式に影響を及ぼすことがなくなる．未知のパラメータ $\boldsymbol{w}$ の要素のほとんどが0である（スパースな）状態ならば，重要な要素は未知のパラメータ $\boldsymbol{w}$ の非ゼロな要素となっている箇所と言える．つまり，最もスパースとなるようにパラメータ $\boldsymbol{w}$ を決めることが出来れば，大規模なデータの重要な要素を見つけ出すことが出来るようになる．

3　l_0-ノルム最小化問題

データの重要な要素を見つけ出すために，最もスパースとなるようにパラメータ $\boldsymbol{w}$ を決めることを考える．推測する $\boldsymbol{w}$ の非ゼロ要素を小さくすることを目的にする際に使われる表現として，l_0-ノルムがある．p 次元ベクトル $\boldsymbol{w}$ の l_0-ノルムの定義を以下に示す．

$$\|\boldsymbol{w}\|_0 = \sum_{i=1}^{p} \delta(w_i) \quad , \quad \delta(w_i) = \begin{cases} 1 & (w_i \neq 0) \\ 0 & (w_i = 0) \end{cases}$$

l_0-ノルムとは，上記式のように非ゼロ要素数を数えるものである．

通常，l_0-ノルムの最小化問題では，圧縮センシングなどのように，制約条件に線形制約を入れることが一般的であるが，今回は回帰などにも適用できるように 2 次制約を加える．

一般的に未知のパラメータ $\boldsymbol{w}$ の採用される変数の数が多いほど，残差平方和は小さくなる．すなわち，パラメータ $\boldsymbol{w}$ の採用される変数の数と残差平方和は，トレードオフの関係にある．そこで，残差平方和をある定数 C 以下に抑えた中で（一定水準の中で），パラメータ $\boldsymbol{w}$ として最もスパースな解を得る以下のような l_0-ノルム最小化問題を考える．

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \|\boldsymbol{w}\|_0 \\ \text{subject to} & \|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})\|_2^2 \leq C \end{array} \tag{2}$$

この問題 (2) を解くことで，残差平方和は一定の水準を超えないようにしつつ，スパース性のある解を得ることができているようにパラメータ $\boldsymbol{w}$ を求める．

3.1 厳密解法（MIQCP）

l_0-ノルム最小化問題 (2) を 2 次制約付き混合整数計画問題（Mixed Integer Quadratically Constrained Program , MIQCP）として定式化する．まず，j 番目の説明変数を選択する/しないを表す 0-1 決定変数

$$z_j \in \{0,1\} \quad (j=1,2,\cdots,p) \tag{3}$$

を導入する．そして，$z_j=0$ の場合には以下の制約条件によって説明変数を回帰式から削除する．

$$z_j=0 \;\Rightarrow\; w_j=0 \quad (j=1,2,\ldots,p) \tag{4}$$

このような論理条件は分枝限定法の分枝操作の中で扱うことができ，例えば整数計画ソルバー CPLEX の `indicator` 関数を使えばこの機能が利用できる．またよく知られている big-M 法でも，このような論理条件は表現できる．

選択された説明変数の数が $\sum_{j=1}^{p} z_j$ と等しいことに注意すると，l_0-ノルム最小化問題 (2) は以下のように定式化できる．

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{z}}{\text{minimize}} & \displaystyle\sum_{j=1}^{p} z_j \\ \text{subject to} & \displaystyle\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p} w_j x_{ij}\right)^2 \leq C \\ & (3),(4) \end{array}$$

この定式化を解く厳密解法（MIQCP）は，実用時間内に問題を解くことが出来れば解は大域的最適化を得るというメリットがある．しかし，問題の規模（データのサイズ）が大きくなると，実用時間で問題を解ききることがかなり難しくなってしまうというデメリットがある．

3.2 Lasso 手法による緩和

l_0-ノルム最小化問題 (2) を厳密に解かず，l_0-ノルムを l_1-ノルムで置き換えて，元の問題 (2) よりも早く解く，かつ，スパースな解を得る方法がある．以下に，l_1-ノルムにより置き換えた問題の定式化を示す．

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \|\boldsymbol{w}\|_1 \\ \text{subject to} & \|(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})\|_2^2 \leq C \end{array} \tag{5}$$

(5) の目的関数に残差平方和を加えた最適化問題は，l_1 正則化または Lasso（Least Absolute Shrinkage and Selection Operator）と呼ばれ，Tibshirani[5] によって提案された．

l_1 緩和手法は，既存の手法で早く求解できるというメリットがある．一方で，厳密に l_0-ノルムを最小化しているわけではなく，l_1-ノルムを最小化しているので，得られた解の l_0-ノルムの値が小さくはあまり期待できないというデメリットがある．

3.3 Adaptive Lasso 手法による緩和

ここでは，l_1-ノルムに置き換える手法の解の精度を上げるための改良手法 Adaptive Lasso[6] について述べる．l_1-ノルムに置き換える手法では，単純に l_1-ノルムを最小化するが，Adaptive Lasso では l_1-ノルムに重みを付けた目的関数を最小化する．このようにすることにより，効果のある変数に対して，効果があまりない変数より少ない罰則を課すことが出来る．l_1-ノルムに重みの強さを変数ご

とに変えることで，重要な変数が選ばれやすくなることが期待される．以下に Adaptive Lasso の定式化を行う．

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \sum_{j=1}^{p} \lambda_j |w_j| \\ \text{subject to} \quad & \|(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})\|_2^2 \leq C \end{aligned} \tag{6}$$

ただし，$\lambda_j > 0$ は適当な定数であり，OLS の求解をして得られた解に基づいて決める．Adaptive Lasso 手法は，l_1-ノルムに置き換える手法と同様に早く求解できるというメリットがある．一方で，厳密に l_0-ノルムを最小化しているわけではなく，l_1-ノルム最小化しているので，得られた解の l_0-ノルムの値が小さいさはあまり期待できないというデメリットがある．ただし，Lasso と比較すると重要な変数が選ばれやすくなることが期待される．

4 提案手法

l_0-ノルムは，不連続であり，非常に取り扱いにくい関数である．そこで，l_0-ノルムを 2 つの凸関数の差（DC）[4] として表現し，それに対応する DC アルゴリズムが適用できるようにする．

その中で，最大-K ノルムが用いられている．最大-K ノルムとは，整数 $K \in \{1, 2, \ldots, p\}$ について，ベクトル $\boldsymbol{w} \in \mathbb{R}^p$ が，$|||\boldsymbol{w}|||_K$ のように表される．まず，ベクトル $\boldsymbol{w}$ の要素の絶対値を次のように大きい順に並べ替える．$|w_{(1)}| \geq |w_{(2)}| \geq \cdots \geq |w_{(K)}| \cdots \geq |w_{(p)}|$ ただし，$w_{(1)}$ とはベクトル $\boldsymbol{w}$ の要素の中での絶対値が 1 番大きいものを，要素の下付き丸括弧で表している．そして，$|||\boldsymbol{w}|||_K$ とは，上位 K（任意の整数）個までの要素の絶対値の和である．定義を以下に表す．

$$|||\boldsymbol{w}|||_K := |\boldsymbol{w}_{(1)}| + |\boldsymbol{w}_{(2)}| + \ldots + |\boldsymbol{w}_{(K)}|$$

この最大-K ノルムを用いると，ベクトルの l_0-ノルム $\|\boldsymbol{w}\|_0$ は以下のように等価に置き換えることができる [3]．

$$\begin{aligned} \|\boldsymbol{w}_0\| = \quad & \min_{k} \quad k \\ & \text{s.t.} \quad |||\boldsymbol{w}|||_h - |||\boldsymbol{w}|||_k = 0 \end{aligned} \tag{7}$$

今回は，この上記式 (7) が成り立つことを l_0-ノルム最小化問題へ応用する．

4.1 l_0-ノルム最小化問題の DC 表現

問題 (2) の l_0-ノルム最小化問題を，先程の式 (7) を用いて，DC 表現すると以下のようになる．

$$\begin{aligned} \min_{k} \quad & k \\ \text{s.t.} \quad & \phi_k = 0 \end{aligned} \tag{8}$$

ただし，

$$\begin{aligned} \phi_k := \underset{\boldsymbol{w},k}{\text{minimize}} \quad & \{k : |||\boldsymbol{w}|||_h - |||\boldsymbol{w}|||_k = 0\} \\ \text{subject to} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\| \leq C \end{aligned} \tag{9}$$

この定式化は，制約条件である $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\| \leq C$ を満たし，$|||\boldsymbol{w}|||_h - |||\boldsymbol{w}|||_k = 0$ となるような最小の k を見つける問題ということを意味している．

また，問題 (9) の ϕ_k について以下のことが言える．

- ϕ_k は k について単調非増加関数
- $1 \leq k^{\exists} \leq p$ のとき，$\phi_k = 0$ となる

問題 (8-9) は，k の値の一つずつ固定していき（具体例を出すと k を p から $p-1, p-2, \cdots$ のように徐々に小さくしていくなどして），その都度，DC アルゴリズムを適用する必要がある．つまり，何度も何度も DC アルゴリズムを適用しなければならないという点において，計算量が多くなってしまう．そこで，今回は k の値をより効率良く固定していくために二分探索法を用いた．二分探索法を用いることで，p 個あるデータを一回の検索で $(p/2)$ 個無視できるようになるため，すべてを調べるよりも計算時間がかからずに済むようになる．

4.2 DC アルゴリズム

二分探索法により固定した k で問題 (9) を解くことを考える．問題 (9) は，$|||\boldsymbol{w}|||_h - |||\boldsymbol{w}|||_k$ という凸関数 − 凸関数，つまり DC として表現されており，DC アルゴリズムが適用できる [3]．

$|||\boldsymbol{w}|||_h - |||\boldsymbol{w}|||_k$ という凸関数 − 凸関数の凹関数（− 凸関数の部分）である $|||\boldsymbol{w}|||_k$ を初期点 $\boldsymbol{w}^{(0)}$ について劣勾配 $s_{\boldsymbol{w}}$ を求めることで線形近似し，出来上がった以下のような凸計画問題を最小化することを考える．

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & |||\boldsymbol{w}|||_h - \boldsymbol{w}^\top s_{\boldsymbol{w}} \\ \text{subject to} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \leq C \end{aligned} \tag{10}$$

この凸計画問題 (10) を DC アルゴリズムの中で何度も解いていくことになる．ただし，h は $1 \leq K < h \leq p$ を満たす任意の整数である．今回の DC アルゴリズムを以下に示す．

Algorithm 1 DC Algorithm

Require: $\boldsymbol{w}^{(0)}, \epsilon > 0$
  $t = 1$
  repeat
    $\boldsymbol{s}_w^{(t-1)} \in \partial \|\!|\boldsymbol{w}^{(t-1)}|\!\|_k$ を求める
    凸計画問題 (10) を解く
    $t \leftarrow t + 1$
  until $|f^{(t-1)} - f^{(t)}| < \epsilon$ holds

DC アルゴリズムは停留点を得る．つまり，解に大域的最適解という保証がない．今回は問題 (9) に DC アルゴリズムを適用しているため，元の問題 (2) の大域的最適解が出てくることはほとんどない．

また DC アルゴリズムの初期点 $\boldsymbol{w}^{(0)}$ として，今回は以下の 3 つを用意した．

1. OLS（最小二乗法）により得られた解
2. Lasso(5) により得られた解
3. Adaptive Lasso(6) により得られた解

5 数値実験

UCI Machine Learning Repository[2] からダウンロードしたデータセットを使用して計算実験を行った．なお，計算実験は CPU: Intel Core i7 3.6GHz, RAM: 32GB, OS:Windows 8.1 Pro の PC を使用し，アルゴリズムの実装には MATLAB R2015b を用いた．MIQCP の求解には CPLEX を使用した．

また，残差平方和をある定数 C で抑える際に，C の値の決め方として以下の二つの例が考えられる．一つは，$C = L \cdot \|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}\|_2^2$ とする場合である．これは，残差平方和が最も小さいときの値 $\|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}\|_2^2$ から，$L(> 1)$ というパラメータで最小の残差平方和を割増しするという意味になる．今回の実験では，この C の値を用いた．

結果を要約すると，

- 説明変数が 30 程度よりも少ないデータセットに関しては，厳密解法の方が提案手法よりも早く，大域的最適解を得ることが出来たので，提案手法の優位性はないことがわかった．
- 与えられたデータセットの説明変数の数が 70 個より大きくなればなるほど提案手法の方が厳密解法と比較して，非ゼロ要素数が少なく，計算時間もあまりかからず（10(s) 以内）解を求めることができた．

計算結果の詳細については，修士論文の本体を参照されたい．

参考文献

[1] 山内 結子, 特集　スパースモデリングの発展 – 原理から応用まで – , 電子情報通信学会誌, Vol.99, No.5 (2016)

[2] K. Bache and M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, URL `https://archive.ics.uci.edu/ml/datasets.html` (2017)

[3] T. Pham Dinh and H. A. Le Thi, “Convex analysis approach to d.c. programming:theory, algorithms and applications,”Acta Mathematica Vietnamica, pp.289 - 355 (1997).

[4] J.Gotoh, A.Takeda, K.Tono “DC formulations and algorithms for sparse optimization problems,” *Mathematical Programming*, pp.1 - 36 (2017).

[5] R.Tibshirani “Regression shrinkage and selection via the LASSO,” *J.R.statist. Soc. Ser, B*, Vol.58, No.1, pp.267 - 288 (1996).

[6] H. Zou “The adaptive lasso and its oracle properties ,” *J.Am.statist. Assoc 101*, pp.1418 - 1429 (2006).