特別研究報告書

遅延評価を行うオンライン最適化アルゴリズムのリグレット解析

指導教員　　山下信雄　准教授

京都大学工学部情報学科
数理工学コース
平成 21 年 4 月入学

鋒　幸洋

平成 26 年 1 月 31 日提出

摘要

統計や機械学習など，様々な分野で扱われる大規模な最適化問題を解く手法として，オンライン最適化が注目を集めている．オンライン最適化は，目的関数が多くの関数の和で表されているとき，その一部の関数の情報のみを用いて点列を生成する手法である．そのため，少量のメモリで実装することができ，扱うモデルにおいてデータが逐次増加するような場合にも対応できる．

本報告書では，オンライン最適化問題に現れる二つのスパース性に着目する．一つは目的関数を構成する関数のスパース性であり，もう一つは生成する解のスパース性である．スパースな解は，L1 正則化項などを加えた問題をオンライン近接勾配法を用いて解くことで求められる．一方，関数がスパース性を持つとき，その勾配の多くの成分は 0 となる．この性質を利用して，L1 正則化項の評価を後でまとめて行う，つまり遅延評価を行うオンライン近接勾配法が提案されている．しかしながら，この手法に対してはリグレットと呼ばれるオンライン最適化の収束評価がなされていない．また，L1 正則化の評価をまとめて行うとき以外の反復において，生成される点列がスパースな解でないという欠点がある．

本報告書では，まず遅延評価を行うオンライン近接勾配法のリグレット解析を行う．次に，遅延評価を行う近接勾配法に対して，生成する点列がスパースとなるような工夫を提案する．さらに，数値実験を行うことで，提案手法によってスパースな解を効率よく生成できることを確認した．

目次

1 序論

近年の情報技術の発展に伴い，様々な分野において大量のデータを扱う問題が増えてきている [2, 9]．このような問題に対する解決策として，統計や機械学習が期待されている．しかしながら，統計や機械学習の分野では，扱うデータの数に比例した大量の変数や大きな次元の変数を伴う最適化問題を解く必要がある．また，応用問題によっては，一定時間内にある程度の精度の解を得ることが求められる．例えば，ビックデータを扱う機械学習では，次々に与えられる大量のデータを迅速に処理し，与えられたデータの規則性を浮き彫りにすることが求められている．

このような大規模な最適化問題に対するアプローチとして，バッチ最適化とオンライン最適化の二つの手法がある．バッチ最適化では，得られたすべてのデータを含んだ目的関数 (損失関数の和) を構築し，その勾配などの情報を用いて最適解を求める．一般的なバッチ最適化のアルゴリズムとしては，内点法，最急降下法，ニュートン法が挙げられる [10, 11]．一方，オンライン最適化では，一回の反復で一つのデータから損失関数を構築し，その情報に基づいて最適解を求める [7, 8]．

オンライン最適化は与えられる損失関数が逐次的に変化するような問題に対して有効である．そのため，株式の配分問題や迷惑メールのフィルタリングなどに用いられている．また，機械学習や統計の分野においては，一定時間内にある程度の精度の解が必要とされる場面が多く，そのような状況においてオンライン最適化アルゴリズムは十分実用的であることが知られている．したがって，本報告書では，前述のビッグデータを伴う問題に対し，オンライン最適化によるアプローチを用いて最適解を求めることを考える．

本報告書で取り扱うオンライン最適化問題は以下のように表される．

$$\min \quad \sum_{t=1}^{T} F^t(x) \equiv \sum_{t=1}^{T} \{f^t(x) + r(x)\} \tag{1}$$

ここで，関数 $f^t : \mathbb{R}^n \to \mathbb{R}(t = 1, \ldots, T)$ を微分可能な凸関数，関数 $r : \mathbb{R}^n \to \mathbb{R}$ を連続な凸関数とする．(a^t, b^t) を与えられたデータのペアと考えたとき，関数 f^t の例としてヒンジ損失関数 $f^t(x) = [-b^t\langle a^t, x\rangle + 1]_+$ やロジスティック損失関数 $f^t(x) = \log(1 + \exp(-b^t\langle a^t, x\rangle))$ が挙げられる．関数 r としては，L1 正則化項 $r(x) = \lambda\|x\|_1$ や L2 正則化項 $r(x) = \frac{\lambda}{2}\|x\|_2^2$，標示関数 $r(x) = \begin{cases} 0 & (x \in \Omega) \\ +\infty & (x \notin \Omega) \end{cases}$ などが考えられる．ただし，関数 r は微分可能でなくても構わない．これらの関数を用いて定義した最適化問題 (1) は凸最適化問題となり，前述の統計や機械学習で頻繁に扱われる問題となっている [2]．このような問題に対する解法としては，劣勾配法や近接勾配法などのバッチ最適化のアルゴリズムにもとづいたオンライン最

適化手法が提案されている [6].

通常の最適化問題と違い，オンライン最適化では目的関数が未知の状態で解を定め，その解に基づいて損失が評価される．そのため，毎回の反復で与えられる損失関数すべてを考慮して最適解を求めることは非常に難しい．そこで，オンライン最適化アルゴリズムの評価のためにリグレットという概念を用いる．リグレットとは，オンライン最適化アルゴリズムから得られる合計コストとバッチ最適化によって得られる合計コストの最適値の差である．オンライン最適化アルゴリズムにおいて，リグレットが小さな値で抑えられているならば，そのアルゴリズムからは精度の高い解が得られると考えることができる．

最も一般的なオンライン最適化アルゴリズムとして知られる劣勾配法は，オンライン学習の文脈では確率劣勾配法，あるいは確率最急降下法とも呼ばれる．このアルゴリズムでは，損失関数が凸であるときにリグレットは $\mathcal{O}(\sqrt{T})$ 以下となる．また，損失関数が強凸であるとき，そのリグレットが $\mathcal{O}(\log T)$ 以下となることが示されている [6]．しかしながら，このアルゴリズムは L1 正則化項などの微分不可能な関数を含む問題に対しては非効率的であり，スパースな解を得ることも難しい．近接勾配法とは劣勾配法と近接点法を組み合わせた手法であり，目的関数に微分不可能な項を含む問題にも対応できる手法である．例として，問題 $\min \ \sum_{t=1}^{T} F^t(x) \equiv \sum_{t=1}^{T} \{f^t(x) + r(x)\}$ への適用を考える．このとき，近接勾配法のアルゴリズムでは，関数 r が特別な構造を持つ場合には，その構造を利用して関数 r に近接点法を，微分可能な関数 f^t に対しては劣勾配法を用いて最小化を行う．バッチ最適化の場合，近接勾配法は以下のアルゴリズムで表される．

$$x^{k+1} = \mathrm{argmin}_x \left\{ \langle \sum_{t=1}^{T} \nabla f^t(x^k), x \rangle + \frac{1}{\eta_k} B_\psi(x, x^k) + r(x) \right\} \tag{2}$$

ここで，$B_\psi(x, y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ は Bregman 関数であり，x-y 間の距離のようなものを表す性質を持っている．また，η_k は k 回目の反復におけるステップ幅である．これをオンライン最適化に拡張したのが次のアルゴリズムである [3, 4].

$$x^{k+1} = \mathrm{argmin}_x \{ \langle \nabla f^k(x^k), x \rangle + \frac{1}{\eta_k} B_\varphi(x, x^k) + r(x) \} \tag{3}$$

このアルゴリズムのリグレットも $\mathcal{O}(\sqrt{T})$ 以下で抑えられることが示されている．また，損失関数が微分可能で，制約条件に標示関数を持つとき，近接勾配法は Greedy Projection Method [8] と呼ばれる．

本報告書では，特に x^t の次元 n が大きく，勾配ベクトル $\nabla f^t(x^t)$ がスパースになるとき，つまり $\nabla f^t(x^t)$ の成分のほとんどが 0 になる場合を考える．例えば，テキストベースの応用問題では，次元 n が辞書などの大量の単語の集合に相当する．その一方で，一回の反復で与えられるテキストデータには比較的少数の単語しか含まれていない．ここで，(a^t, b^t) を t 回

目の反復で与えられるデータとする．ただし，a^t はテキスト中に含まれる単語の頻度を表すベクトル，b^t はテキストのクラスを表すとする．このとき，a^t のほとんどの成分は 0 となる．そのため，ロジスティック損失関数やヒンジ損失関数を f^t とおいたとき，$\nabla f^t(x^t)$ がスパースとなる場面が多い．そのような問題に対して，式 (3) の計算には $\mathcal{O}(n)$ の時間がかかる．ここで，$\mathcal{O}(n)$ の計算が必要となるのは，正則化項 $r(x)$ が存在するからである．もし $r(x)$ がなければ，$\frac{\partial f^t(x^t)}{\partial x_i} \neq 0$ でない x_i だけ更新すればよい．

そこで，[3] では，K 回ごとにまとめて正則化項を評価する遅延評価という手法が紹介されている [5]．この手法を用いることで，計算時間が著しく短くなることが知られている．しかしながら，この遅延評価を含むオンライン最適化アルゴリズムのリグレットの解析は与えられていない．また，この遅延評価を含むアルゴリズム [3] では，正則化項をまとめて評価するため，各反復においてスパースな解を得ることが難しいという欠点がある．

そこで本報告書では，遅延評価を行うオンライン最適化アルゴリズムのリグレットが $\mathcal{O}(\sqrt{T})$ 以下となることを理論的に示し，よりスパースな解を生成するアルゴリズムを提案する．

本報告書の構成を述べる．2 節では，オンライン最適化問題とそれを解くためのアルゴリズムについて述べる．3 節では，遅延評価を行うアルゴリズムのリグレットの解析を行う．4 節では，2 節のアルゴリズムを改良し，よりスパースな解が得られるようなアルゴリズムを提案する．さらに，5 節では提案したアルゴリズムの有用性を確かめるため，いくつかのテスト問題に対して数値実験を行う．最後に，6 節では結論を述べる．

本報告書において，$\|x\|_1$，$\|x\|_2$，$\|x\|_p$ は x の L1，L2，Lp ノルムを表す．微分可能な関数 $f : \mathbb{R}^n \to \mathbb{R}$ に対して，$\nabla_i f(x)$ は $\nabla f(x)$ の i 番目の要素を表す．$[\cdot]_+$ は max 関数，つまり $[a]_+ = \max\{a, 0\}$ である．

2 準備

本節では，オンライン最適化問題の評価方法であるリグレットと，一般的なオンライン最適化アルゴリズムを紹介し，さらにアルゴリズムを高速化するためのテクニックである遅延評価について説明する．

2.1 オンライン最適化問題

大規模な最適化問題に対して，与えられたすべての損失関数を用いて最適解を求める手法をバッチ最適化と呼ぶ．この手法では，精度の良い解が得られる一方で，損失関数の保存のために多くのメモリが必要とされる．一般的なバッチ最適化のアルゴリズムとしては，内点

法が挙げられる．内点法は目的関数のヘッセ行列が疎な場合に高精度な解を高速に求めることができる．しかし，統計や機械学習の分野では一般的にヘッセ行列が密になることが多く，内点法の適用が困難である．

一方，オンライン最適化では，逐次的に与えられるデータごとに損失関数を構築する．オンライン最適化のアルゴリズムでは，次に与えられる損失関数が未知の状態で過去に与えられた損失関数から”最適解”を決定する．さらに，その後に開示された損失関数を用いて損失の計算を行う．与えられる損失関数が凸関数であるとき，または実行可能領域 Q が凸集合であるとき，この問題はオンライン凸最適化問題と呼ばれる．オンライン最適化は少量のメモリしか必要としないのに加え，近年では高い精度を持ったアルゴリズムも開発されている．また，データが増えた際の対応も，オンライン最適化では逐次的に行うことができる．したがって，統計や金融，機械学習の分野では，オンライン最適化を用いて最適解を求めることが多い．

オンライン最適化では，バッチ最適化のようにすべての目的関数の情報を既知として解を決定することができない．そこで，リグレットと呼ばれる指標で解の評価を行う．ここで，あるオンラインアルゴリズム A における，ある時刻 t での解 x^t とする．反復 $t = 1, 2, ..., T$ で与えられる損失関数を $f^t : \mathbb{R}^n \to \mathbb{R}$ として，それまでの損失の合計 $\sum_{t=1}^{T} f^t(x^t)$ を評価する．この場合のリグレットは次の式で表される．

定義 2.1. あるオンラインアルゴリズム A 与えられ，それによって生成された点列 $\{x^t\}$ としたとき，以下の値 $R_{\mathrm{A}}(T)$ をオンラインアルゴリズム A によるリグレットと呼ぶ．

$$R_{\mathrm{A}}(T) = \sum_{t=1}^{T} f^t(x^t) - \min_{x^\star} \sum_{t=1}^{T} f^t(x^\star)$$

上式において，第一項はオンラインアルゴリズム A の解の点列 x^t によって得られる損失の合計を表している．第二項は，実行可能領域内のある一点で解を固定させた時の損失の合計の最小値を表している．つまり，$x^\star$ はバッチ最適化の最適解となり，第二項はその最適値とみなすことができる．

このリグレットを T で割った値，$\frac{R(T)}{T}$ が $T \to \infty$ において 0 に収束するとき，反復を重ねるにつれて，一回の反復当たりのリグレットの値が小さくなることを意味している．つまり次式が成り立つとき，最適値に収束しているとみなす．

$$\lim_{T\to\infty} \frac{R(T)}{T} = 0 \tag{4}$$

例えば，$R(T) = \mathcal{O}(\sqrt{T})$ であれば，式 (2.1) を満たす．本報告書では，オンライン最適化アルゴリズムの収束性の指標として式 (2.1) を用いる．

2.2 既存のオンライン最適化手法

本小節では，オンライン最適化問題を解くための一般的なアルゴリズムを紹介する．まず，オンライン最適化問題を解くためのアルゴリズムとして，最も一般的なものとして次の劣勾配法が挙げられる [6].

(劣勾配法)

初期点 $x^0 \in \mathbb{R}^n$，初期ステップ幅 $\eta_0 > 0$ とする．$t = 0, ..., T$ において次の反復を行う．

Step 1: 劣勾配 $l^t \in \partial f^t(x^t)$ を選ぶ.
Step 2: $x^{t+1} = \mathrm{argmin}_x\{\langle l^t, x\rangle + \frac{1}{2\eta_t}\|x - x^t\|^2\}$ とする.
Step 3: ステップ幅 η_{t+1} を更新する.

劣勾配法はオンライン学習の文脈では確率勾配降下法とも呼ばれ，サポートベクターマシンや主成分分析などの分野で多く適用されている．このアルゴリズムにおいて，損失関数が凸であるとき，前小節で紹介したリグレットが $\mathcal{O}(\sqrt{T})$ 以下となることが示されている．さらに，損失関数が強凸であるとき，そのリグレットが $\mathcal{O}(\log T)$ 以下となることが示されている [6].

次に紹介するのは，Regularized Dual Averaging Method [7] である．このアルゴリズムによる更新は次のようになる.

(Regularized Dual Averaging Method)

初期点 $x^0 \in \mathbb{R}^n$，初期ステップ幅 $\eta_0 > 0$ とする．$t = 0, ...$ において次の反復を行う．

Step 1: 劣勾配 $l^t \in \partial f^t(x)$ を決定する.
Step 2: $\bar{l}^t = \frac{1}{t}\sum_{\tau=1}^{t} l_\tau$ とする.
Step 3: $x^{t+1} = \mathrm{argmin}_x\{\bar{l}_t + r(x) + \frac{1}{2\eta_t}\|x\|^2\}$ を計算する.
Step 4: η_{t+1} を計算する.

$r(x)$ が L1 正則化項であるとき，このアルゴリズムで途中に生成される解 x^t が，劣勾配法で得られる解よりもスパースになることが知られている．また，リグレットが $\mathcal{O}(\sqrt{T})$ 以下となることが証明されている.

前述した劣勾配法では最適化問題 (1) のような正則化項付きの目的関数を扱う問題に対して有効ではない．そこで正則化項付きのオンライン最適化問題に対するアルゴリズムとして

近接勾配法を紹介する．この節の以下では，次の問題を考える．

$$\min f(x) + \sum_{i=1}^{n} r_i(x_i) \tag{5}$$

ここで関数 $f : \mathbb{R}^n \to \mathbb{R}$ は微分可能な凸関数である．近接勾配法は，近節点法と劣勾配法を組み合わせたアルゴリズムであり，問題 (5) を解く際に，関数 $r_i(x_i)$ などの微分不可能な項に対しては近接点法を用い，微分可能な項に対して劣勾配法を用いることで解を求める．バッチ型の近接勾配法は以下の式で表される．

$$x^{t+1} = \mathrm{argmin}_x \{ f(x^t) + \langle x - x^t, \nabla f(x^t) \rangle + \frac{1}{\eta_t} B_\psi(x, x^t) + \sum_{i=1}^{n} r_i(x_i) \}$$

ここで，η_t はステップ幅を表す正の実数，関数 $B_\psi : \mathbb{R}^n \to \mathbb{R}^n$ は Bregman 関数であり，微分可能な強凸関数 $\psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ を用いて，$B_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla\psi(y), x - y \rangle$ と表される．本報告書では $B_\psi(x, y) = \frac{1}{2}\|x - y\|^2$ とする．

最適化問題 (1) において，$r(x)$ が L1，L2 正則化項であるとき，正則化項は $\sum_{i=1}^{n} r_i(x_i)$ と分解可能であるので，上式は以下の n 個の部分問題に分解できる．したがって次式のようなアルゴリズムが得られる．

$$x_i^{t+1} = \mathrm{argmin}_{x_i} \left\{ \langle x_i, \nabla_i f(x_t) \rangle + \frac{1}{2\eta_t}(x_i - x_i^t)^2 + r_i(x_i) \right\} \quad (i = 1, ..., n) \tag{6}$$

ここで，λ は正の実数とする．L1 正則化項 $r_i(x_i) = \lambda|x_i|$ を用いる場合，式 (6) は以下のように書くことができる．

$$x_i^{t+1} = \mathrm{sgn}(x_i^t - \eta_t \nabla_i f^t(x^t)) \max\{|x_i^t - \eta_t \nabla_i f^t(x^t)| - \lambda\eta_t, 0\} \quad (i = 1, ..., n) \tag{7}$$

ここで，関数 sgn は符号関数であり，以下のように表される．

$$\mathrm{sgn}(a) = \begin{cases} +1 & (a > 0) \\ 0 & (a = 0) \\ -1 & (a < 0) \end{cases}$$

また，L2 正則化項 $r_i(x_i) = \frac{\lambda}{2}x_i^2$ を用いる場合，式 (6) は以下のように書くことができる．

$$x_i^{t+1} = \frac{1}{1 + \lambda\eta_t} x_i^t - \frac{\eta_t}{1 + \lambda\eta_t} \nabla_i f^t(x^t) \quad (i = 1, ..., n) \tag{8}$$

ここで，$f(x) = \sum_{t=1}^{T} f^t(x)$ とすると，近接勾配法をオンライン最適化問題に拡張したアルゴリズムは次の式で表される．

$$x_i^{t+1} = \mathrm{argmin}_{x_i} \left\{ \langle x_i, \nabla_i f^t(x_t) \rangle + \frac{1}{2\eta_t}(x_i - x_i^t)^2 + r_i(x_i) \right\} \quad (i = 1, ..., n) \tag{9}$$

本報告書では，このアルゴリズムをオンライン近接勾配法と呼ぶ．このアルゴリズムは，近接勾配法の拡張である Foward Backward Splitting(FOBOS) [3] を一般化したアルゴリズムであり，そのリグレットは $\mathcal{O}(\sqrt{T})$ 以下となることが証明されている．式 (6) と同様，式 (9) は，正則化項が L1 ノルム，L2 ノルムのときに，式 (7)，(8) のように n 個の式に分解して書くことができる．また，すべての損失関数が微分可能で，かつ制約に標示関数 $g(x) = \begin{cases} 0 & (x \in \Omega) \\ +\infty & (x \notin \Omega) \end{cases}$ を持つとき，オンライン近接勾配法は Greedy Projection Method と呼ばれ，そのリグレットが $\mathcal{O}(T)$ 以下となることが示されている [8].

しかしながら，$\nabla f^t(w^t)$ がスパースなベクトルであるとき，毎回の反復で式 (7)，(8) の更新を行うのは非効率である．そこで，そのようなスパース性を利用して効率よく書いを生成するアルゴリズムを 4 節で紹介する．

2.3 アルゴリズムの遅延評価

オンライン学習において，重みベクトル x^t やその勾配 $\nabla f^t(x^t)$ が高次元であることが多い．その一方，勾配 $\nabla f^t(x^t)$ の非ゼロ要素は比較的少ないことが多い．特に，テキストベースの応用分野では以上のような場面は一般的である．ここで，5 節の数値実験でも扱うテキスト分類の事例を考える．この場合，x^t の次元 n は辞書などの単語の全体集合の要素数に相当し，高次元となっている．一方，オンライン最適化を用いてこの問題を解く場合，一回の反復で取り扱うテキストデータには少量の単語しか含まれていない．そのため，勾配ベクトルの非ゼロ要素はその次元と比べて非常に少なく，毎回の反復で x^t のすべての要素を更新するのは明らかに非効率である．そこで，最適化問題 (1) に対しても，正則化項のみを後からまとめて評価するアルゴリズムを考える．なお，機械学習や統計の分野では近接勾配法が多く用いられるため，次のアルゴリズムでは解の更新に近接勾配法を用いている．

アルゴリズム 1 (遅延評価を行うオンライン最適化アルゴリズム)

Step 0: 自然数 K をとる．初期点 $x^0 \in \mathbb{R}^n$，$t = 0$，$\tau = 0$ とする．

Step 1-1: 関数 f^t に対してのみ，式 (9) の更新を適用する．すなわち，次の更新を行う．

$$x_i^{t+1} = \mathrm{argmin}_{x_i}\{\langle x_i, \nabla_i f^t(x_t)\rangle + \frac{1}{2\eta_t}(x_i - x_i^t)^2\} \quad (i = 1, ..., n)$$

Step 1-2: $t = T$ であれば終了．$t \mod K \neq 0$ であれば，$t = t + 1$ として **Step 1-1** へ戻る．そうでなければ **Step 2-0** へ進む．

Step 2-0: $y^{\tau,0} = x^t$，$j = 0$ とする．

Step 2-1: 次の更新 (正則化項 $r(x)$ の遅延評価) を行う．

$$y^{\tau,j+1} = \mathrm{argmin}_y\{r(y) + \frac{1}{2\eta_{\tau K+j}}\|y - y^{\tau,j}\|^2\}$$

Step 2-2: $j = j+1$，$j < K$ なら **Step 2-1** へ戻る．そうでなければ，$x^t = y^{\tau,K}$，
$\tau = \tau + 1$ として **Step 1-1** に進む．

このアルゴリズムの Step1-1 で，ある要素 i において $\nabla_i f^t(x^t) = 0$ であれば，$x_i^{t+1} = x_i^t$ となる．つまり，$i \in \{i|\nabla_i f^t(x^t) \neq 0\}$ となる x_i^t のみが更新される．続いて，Step2-1 においては，それまで考慮されなかった正則化項 $r(x)$ に関してまとめて更新を行っている．Step2-1 のように計算の一部を後回しにする計算手法を遅延評価と呼ぶ．この遅延評価の各計算には $\mathcal{O}(n)$ の時間を必要とするが，反復 $j = 0, ..., K-1$ のそれぞれで $\mathcal{O}(n)$ の計算を行っていては時間の短縮にならない．しかし，ここで幸いなことに，Step2-0 から Step2-2 にかけての計算は，以下の重要な命題のおかげで $\mathcal{O}(n)$ の計算で実行できることが知られている．

命題 2.1. [3, proposition11.] オンライン最適化アルゴリズムによって生成される点列を y^j とする．$(j = 1, ..., K)$ 通常通りの更新は以下の式で表される．

$$y^j = \mathrm{argmin}_y \frac{1}{2}\|y - y^{j-1}\|^2 + \eta_j\|y\|_q$$

$y^\star$ を以下の最適化問題の最適解とすると，

$$y^\star = \mathrm{argmin}_y \frac{1}{2}\|y - y^0\|^2 + (\sum_{j=1}^{K}\eta_j)\|y\|_q$$

この時，$q \in 1, 2, \infty$ において，y^K と $y^\star$ は等しくなる．

命題 2.1 では，正則化項が L1，L2，L∞ ノルムであるとき，Step2-0 から Step2-2 にかけての反復を K 回行って得られた解 $y^{\tau,K}$ と，ステップ幅をまとめて反復を 1 回行って得られる解が等しくなることが示されている．つまり，最適化問題 (1) の正則化関数 $r(x)$ として，L1，L2，L∞ ノルムを用いるとき，この命題を用いることで，アルゴリズム 1 の Step2-0 から Step2-2 における K 回の反復が，1 回の反復でまとめて計算できることが示されている．しかしながら，この遅延評価を用いたオンライン最適化アルゴリズムのリグレット解析は与えられていないため，本報告書の次節においてその解析を行う．また，正則化項 $r(x)$ の計算

が後回しになるため，x^t の要素のほとんどが 0 にならないという欠点がある．そこで，本報告書の 4 節において，スパースな解を生成するアルゴリズムを提案する．

3　遅延評価を行うアルゴリズムのリグレット解析

アルゴリズム 1 では，K 回に 1 度の反復で，すべての正則化項 $r(x)$ の評価が行われる．しかし，一部の $r(x)$ がアルゴリズムの Step1 で計算される場合もあってよいはずである．そこで以下では，どちらの場合も含むような一般的な遅延評価を扱うようなアルゴリズムを考える．損失関数を $g^t : \mathbb{R}^n \to \mathbb{R}$，遅延評価される損失関数を $h^t : \mathbb{R}^n \to \mathbb{R}$ とする．このとき，以下の関係が成立する．

$$h^t(x^t) = f^t(x^t) + r(x^t) - g^t(x^t)$$

これを用いて，アルゴリズム 1 は以下のように書き直すことができる

アルゴリズム 2 (遅延評価を行う一般的なオンライン最適化アルゴリズム)

Step 0: 自然数 K をとる．初期点 $x^0 \in \mathbb{R}^n$，$t = 0$，$\tau = 0$ とする．

Step 1: $f^t(x) + r(x)$ を関数 g^t と h^t に分割する．さらに，g^t を微分可能な関数 $g_1^t(x)$ と関数 $g_2^t(x) = g^t(x) - g_1^t(x)$ に分割する．

Step 1-1: $i = 1, ..., n$ に対して次の更新を行う．

$$x_i^{t+1} = \mathrm{argmin}_x \{g_2^t(x) + \langle x, \nabla g_1^t(x^t)\rangle + \frac{1}{2\eta_t}(x - x^t)^2\}$$

Step 1-2: $t = T$ であれば終了．$t \mod K \neq 0$ であれば，$t = t + 1$ として **Step 1-1** へ戻る．そうでなければ **Step 2** へ進む．

Step 2: $y^{\tau,0} = x^t$，$j = 0$ とする．$h^{\tau K+j+1}$ を微分可能な関数 $h_1^{\tau K+j+1}(x)$ と $h_2^{\tau K+j+1}(x) = h^{\tau K+j+1}(x) - h_1^{\tau K+j+1}(x)$ に分割する．

Step 2-1: $i = 1, ..., n$ に対して次の更新を行う．

$$y^{\tau,j+1} = \mathrm{argmin}_y \{h_2^{\tau K+j+1}(y) + \langle x, \nabla h_1^{\tau K+j+1}(y^{\tau,j})\rangle + \frac{1}{2\eta_{\tau K+j+1}}(y - y^{\tau,j})^2\}$$

Step 2-2: $j = j + 1$ とする．$j < K$ なら **Step 2-1** へ戻る．そうでなければ，$x^t = y^{\tau,K}$，$\tau = \tau + 1$ として **Step 1** に進む．

ここで，$g^t(x) = f^t(x)$，$h^t(x) = r(x)$ とすれば，アルゴリズム 1 となる．一方で，$r(x) = \sum_{i=1}^n r_i(x_i)$ と書けるとき，$I_t = \{i | \nabla_i f^t(x) \neq 0\}$ とおくことで，$g^t(x) = f^t(x) + \sum_{i \in I_t} \lambda |x_i|$，$h^t(x) = \sum_{i \notin I_t} \lambda |x_i|$ とすることもできる．ここで，$r_i(x_i) = \lambda |x_i|$ のとき，Step1 の計算は $\mathcal{O}(n)$ となり，この場合でも各反復の計算のオーダーは変わらない．

本節では，アルゴリズム 2 のリグレットが $\mathcal{O}(\sqrt{T})$ となることを示す．証明のために，以下の仮定を設ける．

仮定 3.1. すべての $t = 0, 1, ..., T$ において，$\|\nabla g_1^t(x^t)\| \leq G$ かつ $\|\nabla h_1^t(x^t)\| \leq G$ を満たす任意の実数 G が存在する．また，$\max_{\delta \in \partial g_2^t(x^{t+1})} \|\delta\| \leq N$ かつ $\max_{\epsilon \in \partial h_2^t(y^{\tau,j+1})} \|\epsilon\| \leq N$ を満たす任意の実数 N が存在する．

仮定 3.2. $t = 1, ..., T$ において，関数 h^t がリプシッツ連続とする．

仮定 3.2 は通常オンラインアルゴリズムのリグレット解析では仮定されているものである．一方で，仮定 3.1 は遅延評価を用いたオンラインアルゴリズムのリグレット解析のために新たに加えたものである．$h^t(x) = \sum_{i=1}^n \lambda |x_i|$ のときはこの仮定が成り立つ．

定理 3.1. 仮定 3.1，3.2 が成り立ち，かつアルゴリズム 4 の x^t の更新に近接勾配法を用いるとする．$t = 1, ..., T$ において，ステップ幅の更新を $\eta_t = \frac{\eta_0}{\sqrt{t}}$ とおくと，次の式が得られる．

$$\sum_{t=0}^{T} f^t(x^t) + r(x^t) - f^t(x^\star) - r(x^\star) \leq \mathcal{O}(\sqrt{T}) \tag{10}$$

Proof. アルゴリズムの評価のために，$T = MK$ となる自然数 M が存在するとし，時刻 T におけるリグレットを考える．そのような M が存在しないときも $M(K-1) \leq T \leq MK$ を満たすような M が存在するため，一般性は失わない．ここで，アルゴリズムによる合計損失を求めるために，$j = 1, ..., K$，$\tau = 0, ..., M$ において，次の関数 P を考える．

$$P^{2\tau K + j}(z) = g^{\tau K + j}(z)$$

$$P^{2\tau K + j + K}(z) = h^{\tau K + j}(z)$$

また，このとき z を次のようにおく．

$$z^{2\tau K + j} = x^{\tau K + j}$$

$$z^{2\tau K + j + K} = y^{\tau, j}$$

また，$\tau = 0$，$j = 0$ のとき，$P^0(x^0) = g^0(x^0)$ とおく．このとき，アルゴリズム 2 は問題 $\min \sum_{t=0}^{2T} P^t(z)$ に対してオンライン近接勾配法適用したものとみなすことができ，そのリ

グレットは次の式で表される．

$$\sum_{t=0}^{2T}(P^t(z^t)-P^t(z^\star))\leq\mathcal{O}(\sqrt{T}) \tag{11}$$

式 (11) を関数 g^t と関数 h^t を用いて書き直す，次のアルゴリズム 2 のリグレット $R_{\mathrm{al2}}(T)$ が得られる．

$$R_{\mathrm{al2}}(T)=\sum_{\tau=0}^{M}\sum_{j=1}^{K}\{g^{\tau K+j}(x^{\tau K+j})+h^{\tau K+j}(y^{\tau,j})-g^{\tau K+j}(x^\star)-h^{\tau K+j}(x^\star)\}+\{g^0(x^0)-g^0(x^\star)\}$$

ここで，実際に評価したいリグレットは

$$R(T)=\sum_{\tau=0}^{M}\sum_{j=1}^{K}\{g^{\tau K+j}(x^{\tau K+j})+h^{\tau K+j}(x^{\tau K+j})-g^{\tau K+j}(x^\star)-h^{\tau K+j}(x^\star)\}+\{g^0(x^0)-g^0(x^\star)\}$$

このリグレット $R(T)$ は $\mathcal{O}(\sqrt{T})$ 以下となる．$R(T)$ と $R_{\mathrm{al2}}(T)$ との差は以下の式で表される．

$$|R_{\mathrm{al2}}(T)-R(T)|=\sum_{\tau=0}^{M}\sum_{j=1}^{K}|h^{\tau K+j}(y^{\tau,j})-h^{\tau K+j}(x^{\tau K+j})| \tag{12}$$

これより，式 (12) の右辺の評価を行う．

仮定 3.2 より，定数 L_t を関数 h^t におけるリプシッツ定数とすると式 (12) より，

$$\sum_{\tau=0}^{M}\sum_{\mathrm{j}=1}^{K}\{|h^{\tau K+j}(x^{\tau K+j})-h^{\tau K+j}(y^{\tau,j})|\}\leq\sum_{\tau=0}^{M}\sum_{j=1}^{K}L_{\tau K+j}\|x^{\tau K+j}-y^{\tau,j}\|$$

さらに，$\max_{t\in\{1,2,\ldots,T\}}L_t=\bar{L}$ とおくと，

$$\sum_{\tau=0}^{M}\sum_{\mathrm{j}=1}^{K}\{|h^{\tau K+j}(x^{\tau K+j})-h^{\tau K+j}(y^{\tau,j})|\}\leq\bar{L}\sum_{\tau=0}^{M}\sum_{j=1}^{K}\|x^{\tau K+j}-y^{\tau,j}\| \tag{13}$$

また，$\|x^{\tau K+j}-y^{\tau,j}\|$ の評価を考えて，

$$\|x^{\tau K+j}-y^{\tau,j}\|\leq\sum_{i=j}^{K-1}\|x^{\tau K+i}-x^{\tau K+i+1}\|+\sum_{i=0}^{j-1}\|y^{\tau,i}-y^{\tau,i+1}\| \tag{14}$$

となる．ここで，x^t の更新に近接勾配法を用いていることに注意する．

$$x_i^{t+1}=\mathrm{argmin}_x\{g_2^t(x)+\langle x,\nabla g_1^t(x^t)\rangle+\frac{1}{2\eta_t}(x-x^t)^2\}$$

この部分問題は無制約の凸計画問題と見ることができる．そのため，x^{t+1} は最適性の一次の必要条件，つまり次式を満たす．

$$\nabla g_1^t(x^t) + \delta^t + \frac{1}{\eta_t}(x^{t+1} - x^t) = 0$$

ただし，$\delta \in \partial g_2^t(x^{t+1})$ である．これより，

$$\begin{aligned} \|x^{t+1} - x^t\| &= \| - \eta_t \nabla g_1^t(x^t) - \eta_t \delta^t \| \\ &\leq \eta_t(\|\nabla g_1^t(x^t)\| + \|\delta^t\|) \\ &\leq \eta_t(G + N) \end{aligned} \tag{15}$$

同様にして，

$$\begin{aligned} \|y^{\tau,i} - y^{\tau,i+1}\| &\leq \eta_{\tau K+i}(\|\nabla h_1^{\tau K+i}(y^{\tau,i})\| + \|\epsilon^{\tau K+i}\|) \\ &\leq \eta_{\tau K+i}(G + N) \end{aligned} \tag{16}$$

ここで，ϵ^t は，$\epsilon^t \in \partial h_2^t(y^{\tau,j})$ を満たす劣勾配である．したがって， 式 (13)-(16) の結果から，

$$\begin{aligned} \sum_{\tau=0}^{M} \sum_{\mathrm{j}=1}^{K} \{|h^{\tau K+j}(x^{\tau K+j}) - h^{\tau K+j}(y^{\tau,j})|\} &\leq \bar{L} \sum_{\tau=0}^{M} \sum_{j=1}^{K} \|x^{\tau K+j} - y^{\tau,j}\| \\ &\leq \bar{L} \sum_{\tau=0}^{M} \sum_{j=1}^{K} \left\{ \sum_{i=j}^{K-1} \|x^{\tau K+i} - x^{\tau K+i+1}\| + \sum_{i=0}^{j-1} \|y^{\tau,i} - y^{\tau,i+1}\| \right\} \\ &\leq \bar{L} \sum_{\tau=0}^{M} \sum_{j=1}^{K} \left\{ \sum_{i=j}^{K-1} \eta_{\tau K+i}(G + N) + \sum_{i=0}^{j-1} \eta_{\tau K+i}(G + N) \right\} \\ &\leq \bar{L}(G + N) \sum_{\tau=0}^{M} \sum_{j=1}^{K} \sum_{i=0}^{K-1} \eta_{\tau K+i} \\ &= \bar{L}K(G + N) \sum_{\tau=0}^{M} \sum_{i=0}^{K-1} \eta_{\tau K+i} \end{aligned}$$

ここで，ステップ幅を $\eta_t = \frac{\eta_0}{\sqrt{t}}$ とおくと，

$$\begin{aligned} R_{\mathrm{al2}}(T) &= R(T) + \bar{L}K(G+N)\sum_{\tau=0}^{M}\sum_{i=1}^{K-1}\eta_{\tau K+i} \\ &\leq R(T) + \bar{L}K(G+N)\sum_{t=0}^{MK}\eta_t \\ &\leq R(T) + \bar{L}K(G+N)\{\int_0^{MK}\eta_t + \eta_0\} \\ &= R(T) + 2\eta_0\bar{L}K(G+N)\{2\sqrt{MK}+1\} \end{aligned}$$

$T = MK$，かつ $R(T)$ が $\mathcal{O}(\sqrt{T})$ であるから，提案アルゴリズムのリグレットが $\mathcal{O}(\sqrt{T})$ で抑えられることが示された． □

4 スパース性を保つための遅延評価の工夫

前節で述べたように，アルゴリズム 1，2 はオンライン最適化問題 (1) の解を得るための有効なアルゴリズムである．しかしながら，これらのアルゴリズムではスパースな解を得ることが難しく，機械学習の分野では扱いづらいという欠点がある．そこで，本節では遅延評価の特性は維持しつつ，正則化項の計算をたえず組み込むことによって，スパースな解を求めるようなアルゴリズムを提案する．

本節では，次式のように最適化問題 (1) に対して L1 正則化項を用いた問題を考える．

$$\min \quad \sum_{t=1}^{t} f^t(x^t) + \sum_{i=1}^{n}\lambda|x_i^t|$$

遅延評価を行う際に添字集合 $I_t = \{i|\nabla_i f^t(x^t) \neq 0\}$ を用いて，要素を更新するかどうかの判定を行う．同時に，L1 正則化項の分解可能性を利用して，1 反復当たりの合計損失を以下の式のように分割する．

$$f^t(x^t) + \sum_{i=1}^{n}\lambda|x_i^t| = f^t(x^t) + \sum_{i\in I_t}\lambda|x_i^t| + \sum_{i\notin I_t}\lambda|x_i^t|$$

これを利用して，遅延評価を行い，かつスパースな解を生成するアルゴリズムは次のようになる．

アルゴリズム 3 (遅延評価を行うアルゴリズムでスパースな解を生成する工夫)

Step 0: 十分大きな自然数 M，K をとる．初期点 $x^1 = \mathbf{0}$，$\Gamma^1 = 0$，$z^1 = \mathbf{0}$，$t = 1$ とする．

Step 1: $I_t = \{i | \nabla_i f^t(x^t) \neq 0\}$ とする．

Step 2: $t = MK$ ならば終了．そうでないならば，

$$\hat{x_i}^{t+1} = \begin{cases} \mathrm{sgn}(x_i^t - \eta_t \nabla_i f^t(x^t)) \max\{|x_i^t - \eta_t \nabla_i f^t(x^t)| - \lambda\eta_t, 0\} & (i \in I_t) \\ x_i^t & (i \notin I_t) \end{cases}$$

$$x_i^{t+1} = \begin{cases} \mathrm{sgn}(\hat{x_i}^t) \max\{|\hat{x_i}^t| - \lambda(\Gamma^t - z_i^t), 0\} & (i \in I_t) \\ \hat{x_i}^t & (i \notin I_t) \end{cases}$$

$$\Gamma^{t+1} = \Gamma^t + \eta_t$$
$$z_i^{t+1} = \begin{cases} \Gamma^{t+1} & (i \in I_t) \\ z_i^t & (i \notin I_t) \end{cases}$$

Step 3: $t \mod K = 0$ のとき

$$x_i^{t+1} = \mathrm{sgn}(x_i^t)(|x_i^t| - \lambda(\Gamma^t - z_i^t))$$
$$\Gamma^{t+1} = \Gamma^t$$
$$z^{t+1} = \Gamma^{t+1} \quad (i = 1, ..., n)$$

とする．

$t := t + 1$ として, **Step 1** に戻る．

Γ^t，z^t はともにステップ幅を蓄積する変数であり，二つの変数の差は，x_i が K 回の反復で更新されなかった時のステップ幅の合計となる．アルゴリズム 1 では，正則化項 $r(x)$ の計算を K 回の反復で 1 度行うが，アルゴリズム 3 では，正則化項 $\sum_{i \notin I_t} r_i(x_i)$ の計算を，次に $i \in I_t$ となるような反復で計算している．Step3 は x_i^t のすべての要素を必ず K 回以内に遅延評価するために設けたが，アルゴリズムの動作上では必ずしも必要ではないことに注意する．

5 数値実験

本節では，前節で紹介したアルゴリズムの有用性を示すため，Multi-Domain Dataset[1] を用いた学習問題に対して，既存手法である L1 正則化項に対するアルゴリズム 1 とアルゴリズム 3 を適用した数値実験結果を示す．実験は CPU が Intel(R) Core(TM)2Duo 3.00GHz，

メモリが 3.9GB の計算機上で行い，プログラムの実装には MATLAB7.6.0(R2008a) を用いた．

Multi-Domain Dataset [1] には Amazon.com で扱われている商品のレビューデータが含まれている．今回の数値実験では，Amazon.com の本カテゴリのレビューデータを用いた評価分類を行う．本カテゴリには $K = 4465$ 個のサンプルデータが存在し，各サンプルはそれぞれ入力データ $a^k \in \mathbb{R}^n$ と出力データ $b^k \in \{-1, 1\}$ が含まれている．入力データ a^k には k 番目のレビューデータに含まれる特徴の回数が含まれている．ここでの特徴は，レビュー文章内で商品の説明のために用いられた単語を指す．a^k の次元 n は，本カテゴリのレビューに現れる特徴の総数を指し，今回の実験では $n = 332440$ である．a^k には最大で 5412 個，最小で 11 個，平均すると 228 個の非ゼロ要素しか含まれていない．出力データ b^k の値は，レビューがポジティブな時 (レート☆ 4 または☆ 5) に $+1$，ネガティブな時 (レート☆ 1 または☆ 2) に-1 となる．実験では，各反復で入力データと出力データのペア (a^t, b^t) が与えられ，時間の経過に伴って学習を行い，損失の合計の変化を計測する．今回の実験では，目的関数としてロジスティック損失関数 $f^t(x) = \log(1 + \exp(-b^t\langle a^t, x\rangle))$ に L1 正則化項を加えた関数を用いた．この関数 f^t を用いて，時刻 T までの損失の合計は次の式で表される．

$$\min \sum_{t=1}^{T} \{\log(1 + \exp(-\tilde{b^t}\langle \tilde{a^t}, x^t\rangle)) + \lambda |x^t|\}$$

上式では，$h(t) = (t \mod K) + 1$ として，$\tilde{a^t} = a^{h(t)}$，$\tilde{b^t} = b^{h(t)}$ とおくことで，アルゴリズム中でサンプル数以上の反復を行うことに対応した．また，$\lambda > 0$ は L1 正則化項における重みを表すパラメータになっている．今回の実験では $\lambda = 0.01$ に設定した．

アルゴリズムのステップ幅は $\eta_t = \frac{\bar{\eta}}{\sqrt{t}}$，$x^t$ はアルゴリズム 3，4 から生成される点列で，初期値 $x^1 = 0$ と設定した．このとき，$\bar{\eta}$ は初期ステップ幅であり，正の定数である．

アルゴリズム 3,4 の評価のために関数 $\hat{\mathcal{R}}(T)$ を次の式で定める．

$$\hat{\mathcal{R}}(T) = \sum_{t=1}^{T} \{\log(1 + \exp(-\tilde{b^t}\langle \tilde{a^t}, x^t\rangle)) + \lambda |x^t|\}$$

本報告書では，2 節，4 節で紹介したアルゴリズム 1 とアルゴリズム 3 に対して，Fsum $= \frac{\hat{\mathcal{R}}(T)}{T}$ を調べる実験と，一定時間経過後の非ゼロ要素の割合を調べる実験の 2 種類を行った．数値実験の終了条件として，プログラムの実行時間がある一定時間を超える場合に停止する条件と，プログラムが一定数以上の反復を行った場合に停止する条件の 2 種類を用意した．

実装にあたって，アルゴリズム 1 において遅延評価が行われる反復を外部反復，K 個のデータの情報から x_i^t を逐次的に更新する反復を内部反復として扱った．また，今回は $K = 4465$ とおくことで，すべてのサンプルデータの情報を用いて x^t を更新したうえで遅延評価を行うプログラムとした．

5.1　ステップ幅のパラメータ $\bar{\eta}$ に関する実験

まず初めに，アルゴリズム 3 のステップ幅の初期値を定める．今回の実験では，ステップ幅の初期値 $\bar{\eta}$ を $\bar{\eta} = \{0.1, 0.05, 0.01, 0.001\}$ から選び，それぞれのステップ幅での Fsum の値の比較を行った．まず，プログラムの終了条件をを 500 秒として実験を行い，その結果として図 1 が得られた．

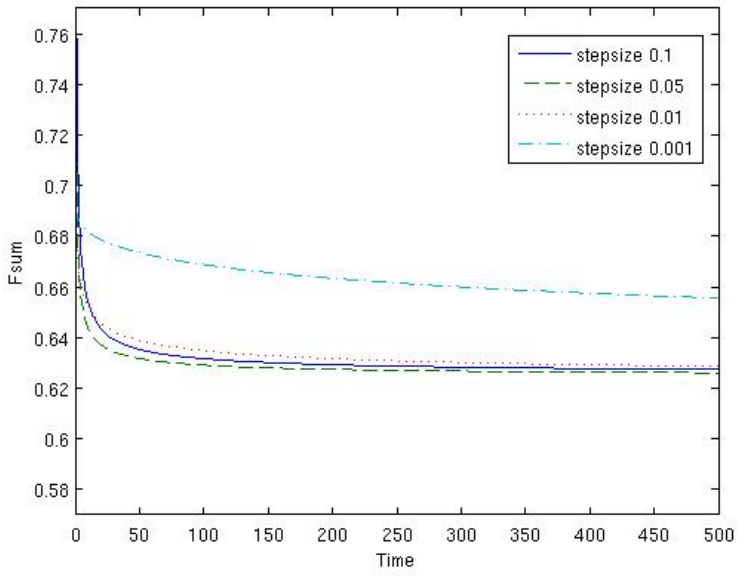

図 1　500 秒でのステップ幅の比較

次に，プログラムの終了条件を外部反復 1000 回として実験を行い，その結果として図 2 が得られた．

この二つの実験の結果から，ステップ幅の初期値が $\bar{\eta} = 0.05$ のとき，Fsum の値が最も小さくなることがわかる．そこで，以降はアルゴリズム 3 の初期ステップ幅を $\bar{\eta} = 0.05$ としたうえで実験を行うこととする．また，同様の実験をアルゴリズム 1 にも行ったところ，ステップ幅の初期値が $\bar{\eta} = 0.01$ のとき，Fsum の値が最も小さくなることがわかった．そこで，以降はアルゴリズム 1 の初期ステップ幅を $\bar{\eta} = 0.01$ としたうえで実験を行うこととする．

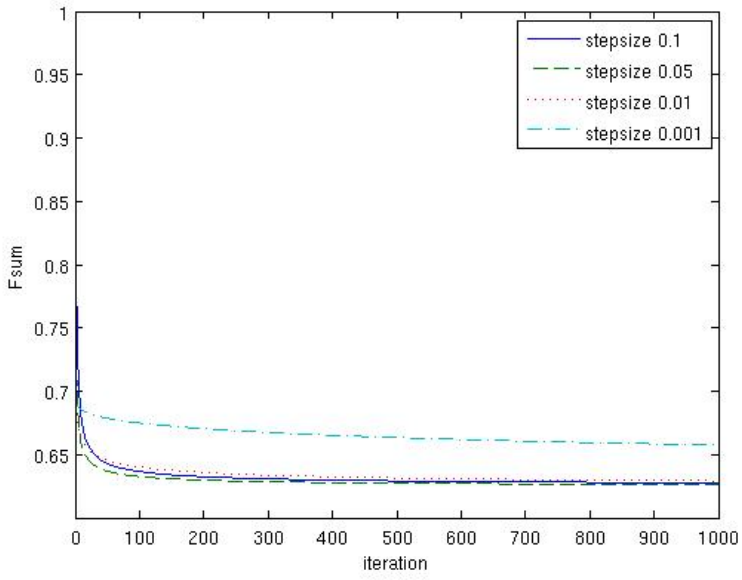

図 2　1000 反復でのステップ幅の比較

5.2　既存手法との比較

始めに，終了条件を実行時間 500 秒として，二つのプログラムの実行結果を比較した．その結果として図 3 が得られた．次に，終了条件を外部反復 1000 回として，二つのプログラムの実行結果を比較した．その結果として図 4 が得られた．

図 3，4 から，アルゴリズム 3 の方がアルゴリズム 1 よりも早く収束することが読み取れる．これは，アルゴリズム 3 では $\sum_{i \notin I_t} r_i(x_i)$ の計算を少なくとも K 回に一度は行っているからである．また，図 3，4 の間でグラフの概形に大きな変化がないことから，二つのアルゴリズムの間に大きな計算量の差は見られないことも分かる．最後に，非ゼロ要素の割合を調べるための実験を行う．終了条件を外部反復 10 回分として，二つのプログラムの実行結果を比較した．その結果が図 5 である．図 5 の横軸は内部反復の回数を指している．二つのアルゴリズムでは，$t \mod K = 0$ を満たす反復 t において $i \notin I_t$ に対応する要素の正則化が行われ，x^t がスパース化されていることがわかる．また，アルゴリズム 3 では内部反復中もスパースな解を生成するのに対し，アルゴリズム 1 では $i \notin I_t$ に対応する要素の正則化が内部反復，つまり $t \mod K \neq 0$ を満たす反復中に行われないため，非ゼロ要素の割合が増える．非ゼロ要素の割合の平均値は，アルゴリズム 3 では 2.43 %，アルゴリズム 1 では 57.07

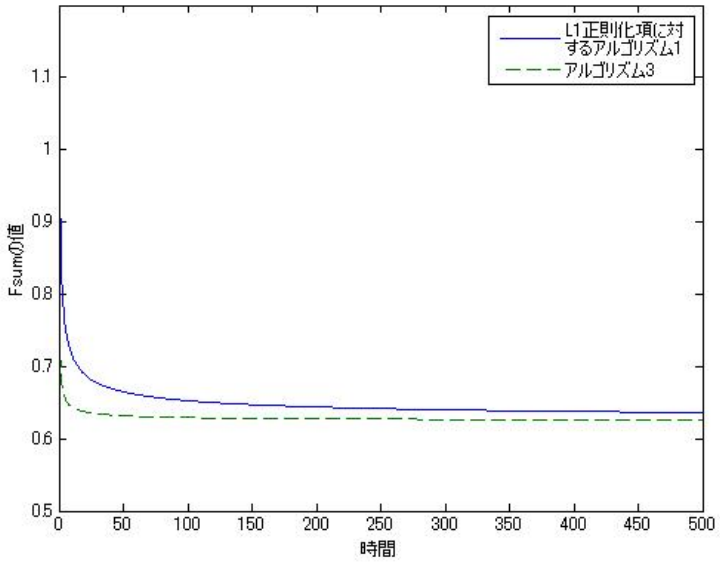

図 3　時間当たりの関数値の比較

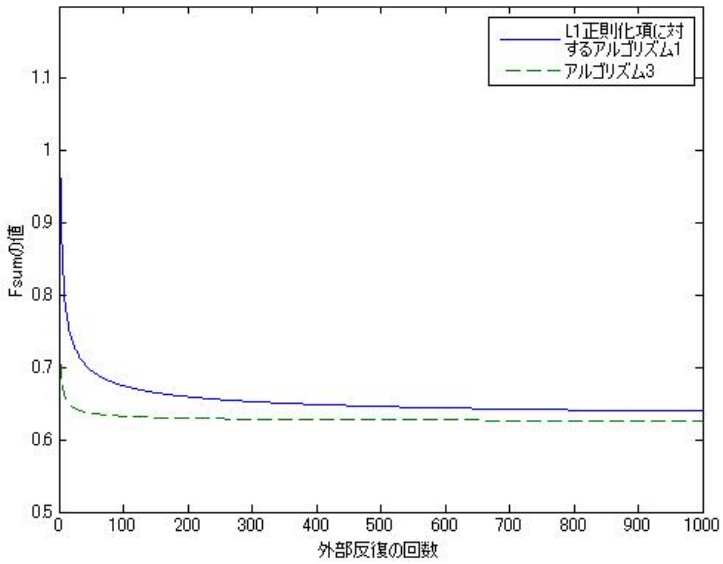

図 4　ステップ当たりの関数値の比較

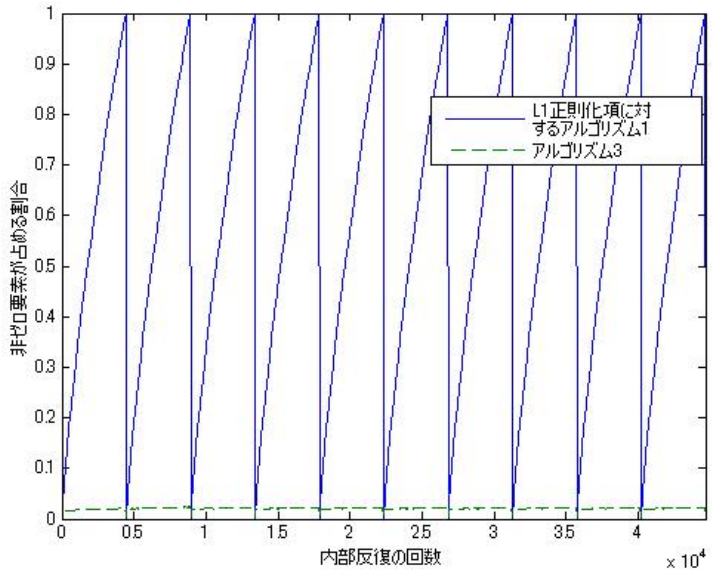

図 5　ステップごとの非ゼロ要素の割合

%となった．ここで，x^t の次元は 332440 であるため，アルゴリズム 4 を用いた学習後にテストを行う際には，大量の特徴の中から 8000 個の特徴だけに着目すればよいことがわかる．

6　結論と今後の課題

本報告書では，大規模な最適化問題に対するオンライン最適化アルゴリズムに遅延評価を適用し，そのリグレット解析を行った．また，遅延評価を行うアルゴリズムにおいて，よりスパースな解を生成する工夫を提案し，数値実験によってその有用性を示した．

今後の課題としては，L1 ノルム以外の正則化項に対する遅延評価の適用など，より一般的な場合に対する手法の開発が挙げられる．また，4 節で紹介したアルゴリズム 3 のリグレット解析がなされていないため，3 節と同様の解析を行うことも重要である．

謝辞

日頃から御教授下さり，本報告書の作成にあたっては細部に至るまで様々な御指摘と適切な御指導を賜った山下信雄准教授に深く感謝の意を表します．また，日頃からお世話になっている福田秀美助教，ならびに最適化数理研究室の皆様に厚く御礼申し上げます．

参考文献

[1] Blitzer, J., Dredze, M., Pereira, F:*Domain Adaptation for Sentiment Classification*, In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440-447, 2007.

[2] C.M. ビショップ：パターン認識と機械学習 上, シュプリンガー・ジャパン, 2007

[3] Duchi, J., Singer, Y.: *Efficient online and batch learning using forward backward splitting*, The journal of machine learning research volume 10 (2009), pp. 2899-2934.

[4] Duchi, J., Shalev-Shwartz, S., Singer, Y., Tewari, A.: *Composite objective mirror descent*, Conference on learning theory(COLT-2010), pp. 14-26.

[5] Langford, J., Li, L., Zhang, T.: *Sparse online learning via truncated gradient*, The journal of machine learning research volume 10 (2009), pp. 777-801.

[6] Shalev-Shwartz, S.: *Online learning and online convex optimization*, Foundations and trends in machine learning volume 4, no 2 (2011), pp. 107-194.

[7] Xiao, L.: *Dual averaging methods for regularized stochastic learning and online optimization*, The journal of machine learning research volume 11 (2010), pp. 2543-2596.

[8] Zinkevich, M.: *Online convex programming and generalized infinitesimal gradient ascent*, Proceedings of the twentieth international conference on machine learning (ICML-2003) pp. 928-936.

[9] 城田真琴：ビッグデータの衝撃 巨大なデータが戦略を決める, 東洋経済新報社, 2012.

[10] 福嶋雅夫：数理計画入門, 朝倉書店, 1996.

[11] 矢部博：工学基礎 最適化とその応用, 数理工学社, 2006.