L1正則化を導入した Online学習手法

東京大学　中川研究室
修士一年 大岩秀和

目次

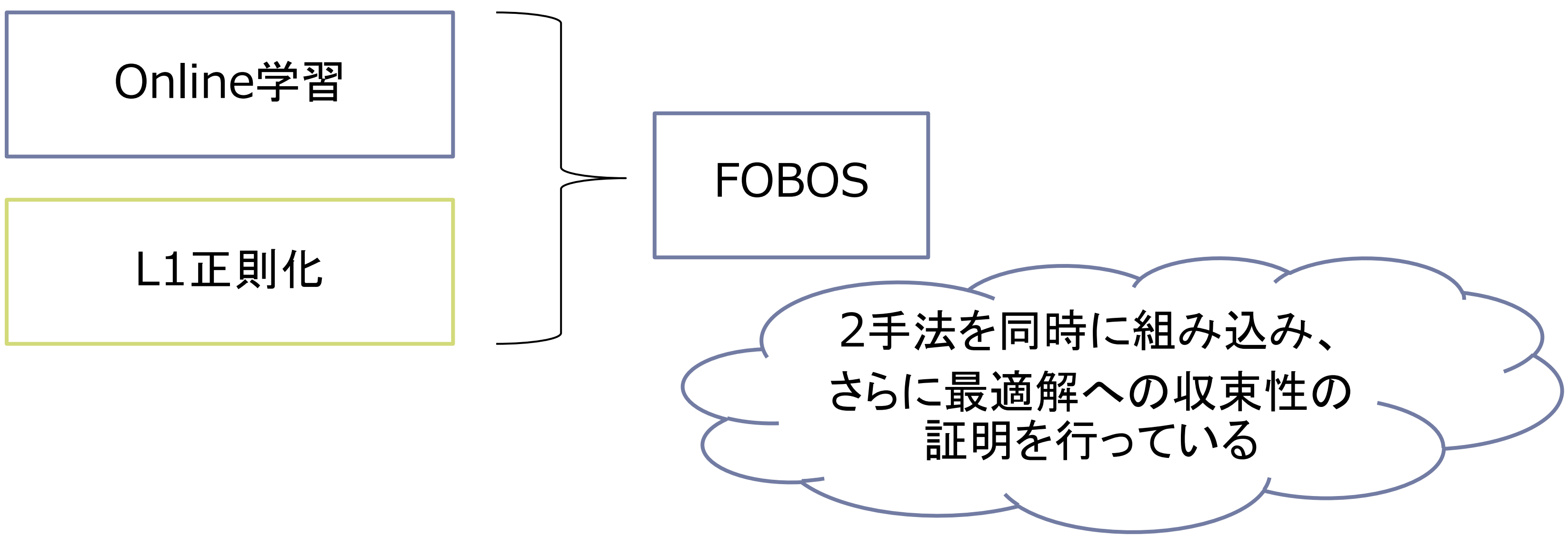
目次

- はじめに
  - 紹介論文概要
  - 問題設定：教師あり学習
- Online学習／L1正則化
  - Online学習
  - L1正則化
- Forward Backward Splitting(FOBOS)
  - FOBOS Algorithm
  - Regret分析
- 実験
- まとめ

紹介論文概要

- Efficient Online and Batch Learning using Forward Backward Splitting

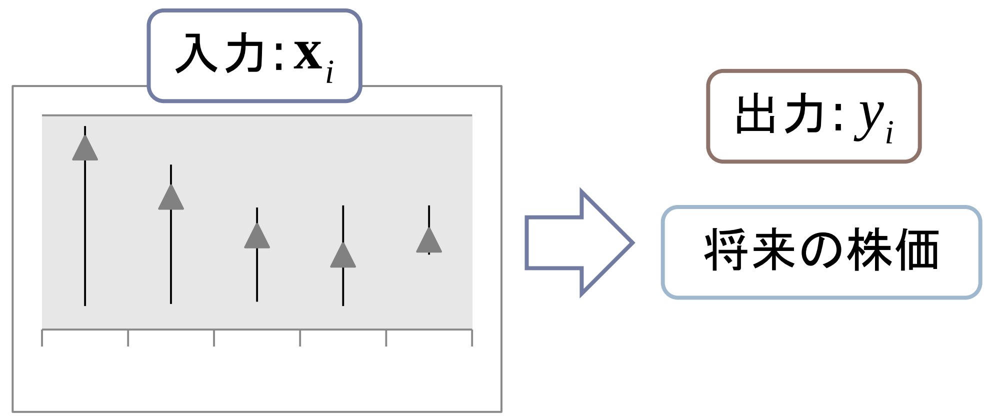
(J. Duchi, Y. Singer) [Journal of Machine Learning Research, 2009]



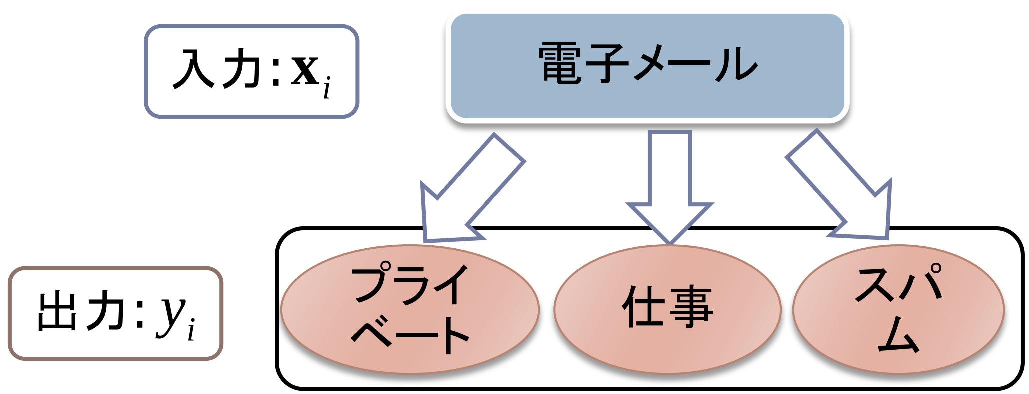
- Online学習を行いながら、同時に特徴選択を行う手法
  - L1正則化に限らず様々な正則化項を一般的に取り扱った議論をしているが、本発表ではL1正則化に話を限定して紹介

問題設定

- 教師あり学習
  - 訓練集合と呼ばれる大きな集合$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots\}$を用いて、入力 $\mathbf{x}_i$ から出力y_i を正確に予測する学習器を作成する技術



- 代表的な応用例
  - 回帰問題
    - 株価推定
  - 分類問題
    - メールフィルタリング
    - 画像認識
    - 評価分類



応用例を題材として、問題設定を解説

- 評価分類(Sentiment Classification)
  - ある商品のレビュー文章から、その文章が商品に対してpositiveな評価を与えているかnegativeな評価を与えているかを判定する

8人中、7人の方が、「このレビューが参考になった」と投票しています。

パターン認識の教科書, 2009/2/20

By freefall - レビューをすべて見る

レビュー対象商品: Pattern Recognition and Machine Learning (Information Science and Statistics) (ハードカバー)

素晴らしい本です。

パターン認識の教科書として、非常に優れていると思います。
パターン認識の原理や特徴、既存の有用な手法などが分かりやすく書かれています。
これらは統計の知識を駆使していますが、その基本の部分から書かれているので
独習する事も可能です。
また、フルカラーなので、グラフや図が非常に綺麗で見やすいです。

パターン認識を研究する初・中級者向けの本と言えると思います。

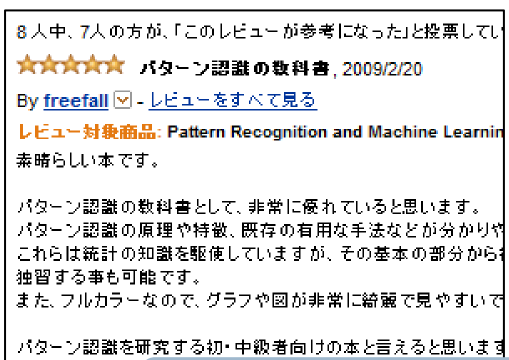
このレビューはpositiveかnegativeか？

問題設定：訓練集合

- レビュー文章から正解を正確に予測したい

文章と正解データがペアで与えられる

文章形式のままでは、学習を行いにくい


8人中、7人の方が、「このレビューが参考になった」と投票してい
★★★★★ パターン認識の教科書, 2009/2/20
By freefall - レビューをすべて見る
レビュー対象商品: Pattern Recognition and Machine Learnin
素晴らしい本です。

パターン認識の教科書として、非常に優れていると思います。
パターン認識の原理や特徴、既存の有用な手法などが分かりや
これらは統計の知識を駆使していますが、その基本の部分から
独習する事も可能です。
また、フルカラーなので、グラフや図が非常に綺麗で見やすいで

パターン認識を研究する初・中級者向けの本と言えると思いま

文章から入力ベクトルを生成

正解
$$y = \begin{cases} 1 & (\mathit{positive}) \\ -1 & (\mathit{negative}) \end{cases}$$

正解データは、-1/1で取り扱う

入力
$$\mathbf{x} = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \\ \vdots \\ x^{(n)} \end{pmatrix}$$

嫌い
好き
退屈
認識

入力ベクトルの各要素は、対応する単語の出現頻度

入力 $\mathbf{x}$から正解yを正確に予測したい

問題設定：予測

- 入力$\mathbf{x}$から出力yを予測するための道具として、重みベクトル$\mathbf{w}$を導入する

重みベクトル $\mathbf{w} = \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \\ \vdots \\ w^{(n)} \end{pmatrix}$

重みベクトルと入力ベクトルの内積の値を用いて予測を行う

$sign(\mathbf{x} \cdot \mathbf{w}) > 0$ ⇨ positive
$sign(\mathbf{x} \cdot \mathbf{w}) < 0$ ⇨ negative
と予測

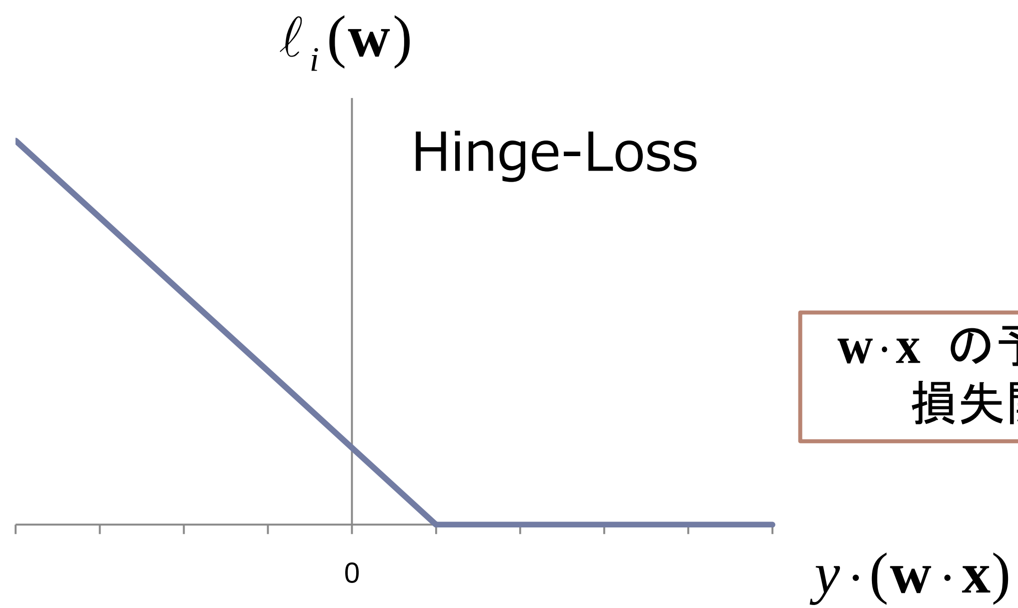
$y(\mathbf{x} \cdot \mathbf{w}) > 0$ ⇨ 予測は正しい
$y(\mathbf{x} \cdot \mathbf{w}) \leq 0$ ⇨ 予測は誤り

適切な$(\mathbf{x} \cdot \mathbf{w})$の符号を返す重みベクトル$\mathbf{w}$を学習することが目標

問題設定：損失関数

- 損失関数を、予測の正確さを図る指標として導入
  - 今回の例では損失関数の一例として、Hinge-Lossを用いる

損失関数 $\ell_i(\mathbf{w}) = [1 - y \cdot (\mathbf{w} \cdot \mathbf{x}_i)]_+ = \begin{cases} 0 & 1 - y \cdot (\mathbf{w} \cdot \mathbf{x}_i) < 0 \\ 1 - y \cdot (\mathbf{w} \cdot \mathbf{x}_i) & 1 - y \cdot (\mathbf{w} \cdot \mathbf{x}_i) \geq 0 \end{cases}$



$\mathbf{w} \cdot \mathbf{x}$ の予測が失敗しているほど、損失関数の値が大きくなる

問題設定の一般化

- 評価分類器の作成は、以下のように定義できる

学習器を作成する

全データの損失の合計 $\sum_i \ell_i(\mathbf{w})$ が
最小となる重みベクトル $\mathbf{w}$ を求める

Batch学習

- 入力データと損失関数は以下のように一般化出来る
  - 1つのデータ: 重みベクトルを受け取り損失を返す損失関数

$$\ell_i(\cdot) : \mathbf{w} \in \Re^n \rightarrow \Re_+$$

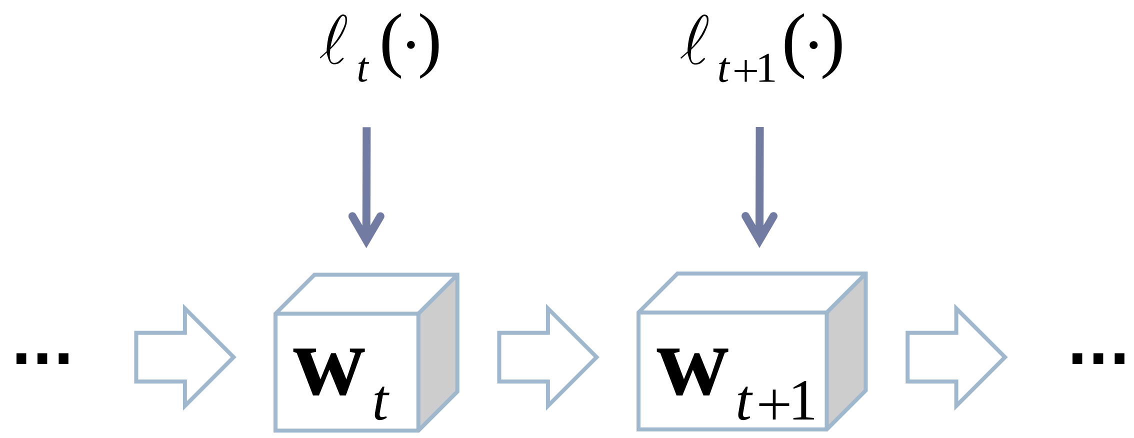
目次

- はじめに
  - 紹介論文概要
  - 問題設定:教師あり学習
- Online学習／L1正則化
  - Online学習
  - L1正則化
- Forward Backward Splitting(FOBOS)
  - FOBOS Algorithm
  - Regret分析
- 実験
- まとめ

Online学習

- Online学習
  - データ一つに対して、重みベクトル $\mathbf{w}$ を逐次的に更新する手法

1. 損失関数 $\ell_t(\cdot)$ を受け取る
2. 重みベクトル $\mathbf{w}_t$ と $\ell_t(\cdot)$ を用いて、重みベクトルを $\mathbf{w}_{t+1}$ に更新
3. 次の損失関数 $\ell_{t+1}(\cdot)$ が存在すれば、上記の操作を繰り返す



Online学習アルゴリズムの目標

Batch学習の目標

$$\min_{\mathbf{w}} \sum_t \ell_t(\mathbf{w})$$

全データの損失の合計$\sum_i \ell_i(\mathbf{w})$が最小となる重みベクトル$\mathbf{w}$を求める

Online学習の目標

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\dots} \sum_t \ell_t(\mathbf{w}_t)$$

全データの損失の合計$\sum_t \ell_t(\mathbf{w}_t)$が最小となる重みベクトル$\mathbf{w}_t$を求める

Online学習アルゴリズムの目標

Online学習の目標

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\ldots} \sum_t \ell_t(\mathbf{w}_t)$$

⇒

Online学習の目標

$$\forall t \quad \min_{\mathbf{w}_t} \ell_t(\mathbf{w}_t)$$

次にやってくる損失関数 $\ell_t(\cdot)$ の値を最小化するように重みベクトル $\mathbf{w}_t$ を更新する問題

ただし、どのようなアルゴリズムを用いても、損失関数 $\ell_t(\cdot)$ を意地悪に設定してやれば、$\min_{\mathbf{w}_1,\mathbf{w}_2,\ldots} \sum_t \ell_t(\mathbf{w}_t)$ のworst caseの値は無限に大きくなる

Online学習アルゴリズムの評価

- *Regret* という概念を導入
  - 元々は、ゲーム理論等で使用されていた枠組み
  - 学習をする過程で蓄積した累積損失と、データを全て見た後で重みベクトルを定めた時の最小合計損失との差

$$Regret(T) = \sum_{t=1}^{T} \left\{ \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \right\} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{t=1}^{T} \ell_t(\mathbf{w})$$

  - *Regret* の上限が $o(T)$ならば、更新を重ねるごとに1データ当たりの *Regret* は0に収束する→最適解に収束する

$$Regret(T) = o(T) \Leftrightarrow \lim_{T \to \infty} \frac{\sum_{t=1}^{T} \left\{ \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \right\}}{T} = 0$$

  - *Regret* の上限を小さなオーダーで抑える事が出来るアルゴリズムは、重みベクトルを高速に最適解へ収束させられる

Online学習の利点

- 大量のメモリを必要としない
  - 数TB程度の大規模なデータから学習を行う場合等に有効
  - 大規模データでは、直接 $\arg\min_{\mathbf{w}} \sum_t \ell_t(\mathbf{w})$ を求めることは難しい
- 新たなデータを入手した時に、再学習が容易
  - batch学習では、新しいデータを入手するたびに、過去のデータも含めた $\arg\min_{\mathbf{w}} \sum_t \ell_t(\mathbf{w})$ を計算しなければならない
  - Online学習では新しいデータのみを使って更新を行えば良い

Greedy Projection

- Online学習での典型的な学習手法
- パラメータ更新式

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t) \quad \eta_t : \text{更新幅(スカラー)}$$

損失関数が最小となる方向へ重みベクトル $\mathbf{w}$ を更新する

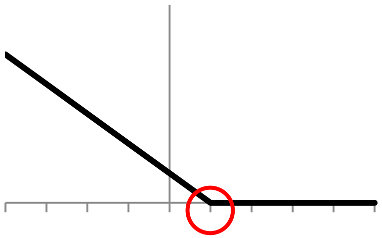
- $Regret$ が $o(T)$ となる条件
  - $\ell(\cdot)$ が凸(convex)
  - $\ell(\cdot)$ が微分可能

Convex

$$\forall \mathbf{w}, \mathbf{v} \quad \forall \lambda \in [0,1]$$

$$\lambda \ell(\mathbf{v}) + (1-\lambda)\ell(\mathbf{w}) \geq \ell(\lambda \mathbf{v} + (1-\lambda)\mathbf{w})$$

- Hinge-Loss等の関数では、微分不可能な点が存在
  - 劣勾配法(Subgradient Method)を用いる



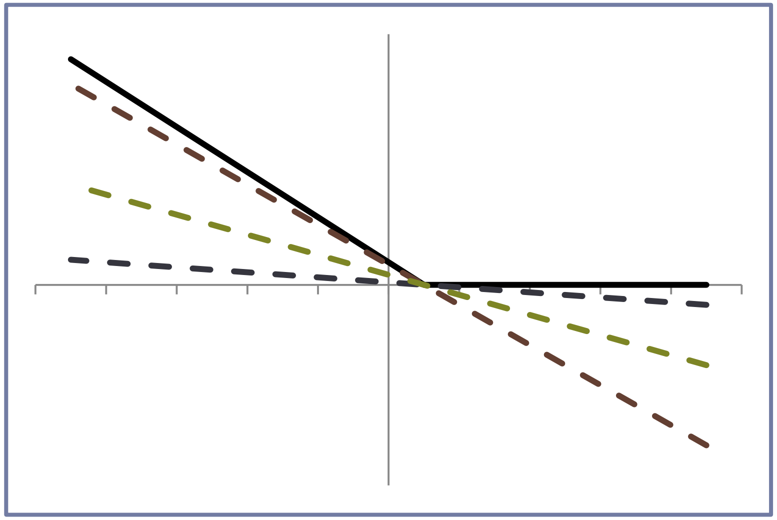
Hinge-Loss

劣勾配法(Subgradient Method)

- 微分不可能な点では、劣勾配 (Subgradient) を用いる

$$\text{劣勾配}: \partial\ell(\mathbf{w}) = \{\mathbf{g} \mid \forall\mathbf{v}: \ell(\mathbf{v}) \geq \ell(\mathbf{w}) + \langle\mathbf{g}, \mathbf{v}-\mathbf{w}\rangle\}$$

Hinge-loss



$\ell(\cdot)$が凸であれば、劣勾配が存在

劣勾配の集合の要素を点線で示している

- パラメータ更新式

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t^{\ell} \qquad \mathbf{g}_t^{\ell} \in \partial\ell(\mathbf{w}_t)$$

目次

- はじめに
  - 紹介論文概要
  - 問題設定:教師あり学習
- Online学習／L1正則化
  - Online学習
  - L1正則化
- Forward Backward Splitting(FOBOS)
  - FOBOS Algorithm
  - Regret分析
- 実験
- まとめ

正則化

- 過学習 (Over-Fitting)
  - 訓練集合の損失関数を最小化する最適なパラメータを求めると、訓練集合に過剰適合する問題が発生することがある
- 正則化
  - パラメータが複雑になればなるほど値が大きくなるペナルティ項を、最適化問題に導入

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \sum_t \ell_t(\mathbf{w}) + r(\mathbf{w}) \right\}$$

  - 過学習を抑止する
  - L2正則化とL1正則化が代表的

L1正則化（Lasso）

- 自然言語等を扱う場合、パラメータ次元数が膨大になる
  - 計算量が膨大になるため、全パラメータを同時には扱いにくい

$$\mathbf{x} = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \\ \vdots \\ x^{(n)} \end{pmatrix} \begin{array}{l} \leftarrow \text{嫌い} \\ \leftarrow \text{好き} \\ \leftarrow \text{退屈} \\ \leftarrow \text{認識} \\ \end{array} \qquad \mathbf{w} = \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \\ \vdots \\ w^{(n)} \end{pmatrix}$$

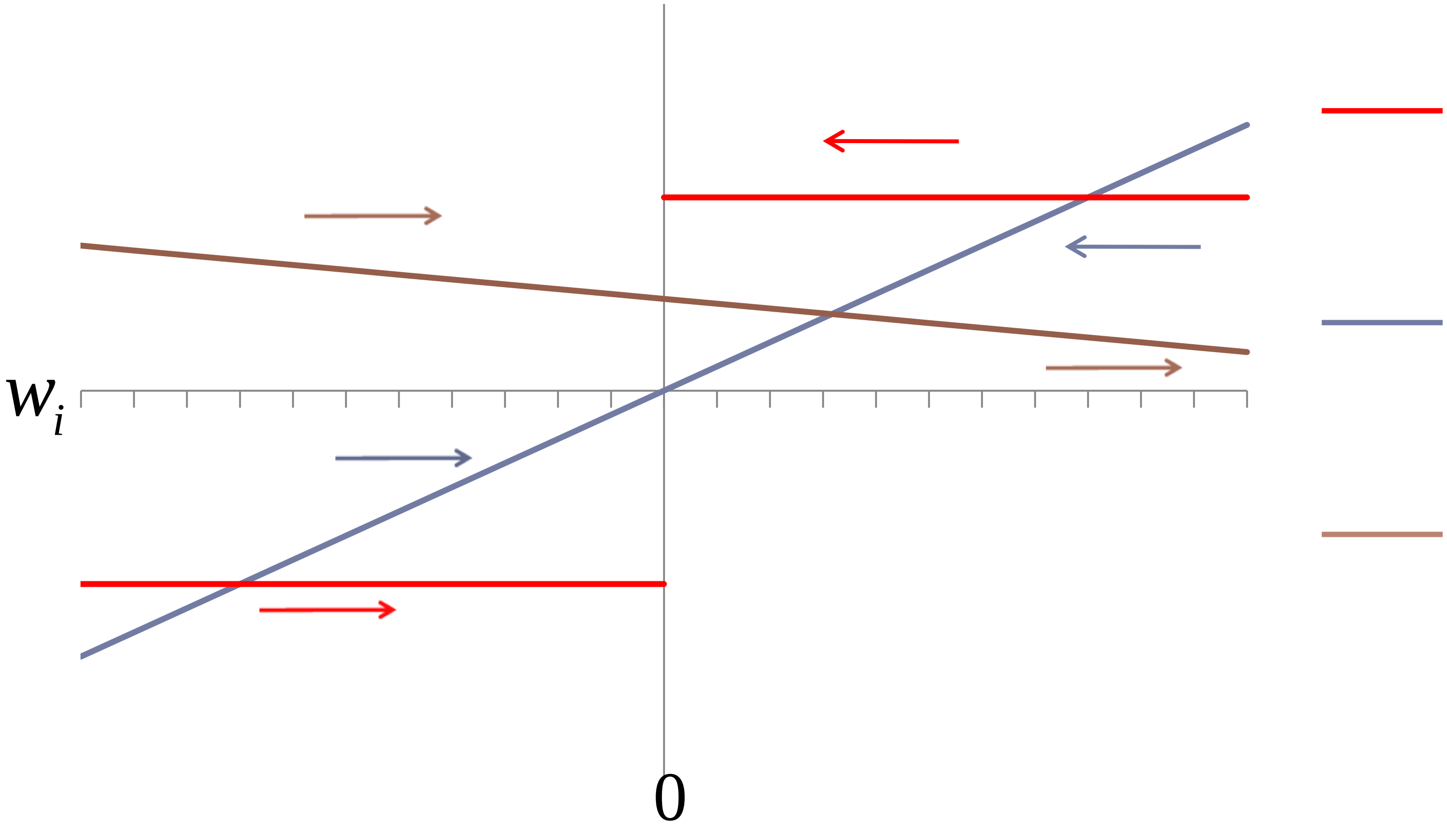
nが非常に大きい場合、
計算量も膨大になる

- L1正則化を導入する
  - パラメータを決める基準に L_1 normを追加する

$$\sum_t \ell_t(\mathbf{w}) + \lambda \|\mathbf{w}\|_1 \qquad \lambda\text{:正則化の比率}$$

L1正則化(Lasso)

$$\frac{\partial}{\partial w^{(i)}}\left(\sum_t \ell_t(\mathbf{w}) + \lambda \|\mathbf{w}\|_1\right) = 0$$



$$\frac{\partial \|\mathbf{w}\|_1}{\partial w^{(i)}}$$

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial w^{(i)}}$$

$$-\frac{\partial \sum_t \ell_t(\mathbf{w})}{\partial w^{(i)}}$$

$\frac{\partial \sum_t \ell_t(\mathbf{w})}{\partial w^{(i)}} \approx 0$の時、$\exists w^{(i)} \approx 0$ $\left|\frac{\partial \sum_t \ell_t(\mathbf{w})}{\partial w^{(i)}}\right| > \left|\frac{\partial \|\mathbf{w}\|_2^2}{\partial w^{(i)}}\right|$ かつ $\left|\frac{\partial \sum_t \ell_t(\mathbf{w})}{\partial w^{(i)}}\right| < \left|\frac{\partial \|\mathbf{w}\|_1}{\partial w^{(i)}}\right|$

L1正則化の特徴

- L1正則化では、勾配が一定
  - 損失関数にとって重要でないパラメータは0に追いやられる

- L1正則化では、多くのパラメータを0にすることが出来る
  - このような操作:Sparse化
  - 多くのパラメータが0になった解:Sparseな解

$$\mathbf{x} = \begin{array}{cl} \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \\ \vdots \\ x^{(n)} \end{pmatrix} & \begin{array}{l} \leftarrow \text{嫌い} \\ \leftarrow \text{好き} \\ \leftarrow \text{退屈} \\ \leftarrow \text{認識} \\ \\ \end{array} \end{array} \qquad \mathbf{w} = \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \\ \vdots \\ w^{(n)} \end{pmatrix}$$

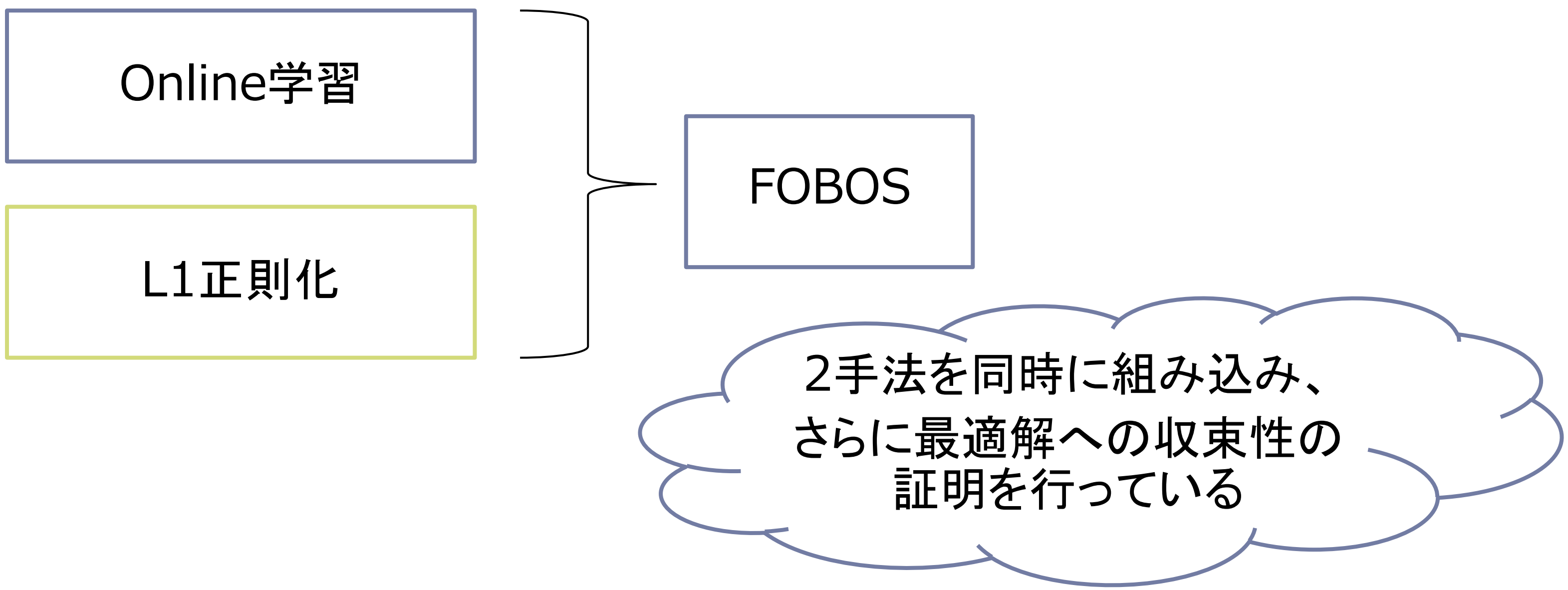
認識に対応する
パラメータは0に

目次

紹介論文（再掲）

- Efficient Online and Batch Learning using Forward Backward Splitting

(J. Duchi, Y. Singer) [Journal of Machine Learning Research, 2009]



- L1正則化に限らず様々な正則化項を一般的に取り扱った分析を行っているが、本発表ではL1正則化に話を限定する

正則化項付きOnline学習

$\mathbf{w}$が最も小さくなる方向へ重みベクトル$\mathbf{w}$を動かす

- 重みベクトルの更新基準　$\ell_t(\mathbf{w}) + r(\mathbf{w})$
  - 条件：それぞれの関数は凸かつ下に有界

$\ell_t(\mathbf{w})$：損失関数項　（重みベクトル$\mathbf{w}$が不適である度合）

例：
最小二乗損失　$\ell_t(\mathbf{w}) = (\mathbf{x}_t \cdot \mathbf{w} - y)^2$

Hinge-Loss　$\ell_t(\mathbf{w}) = [\mathbf{x}_t \cdot \mathbf{w} - y]_+$

$r(\mathbf{w})$：正則化項（重みベクトル$\mathbf{w}$の複雑さの度合）

例：
L2正則化　$r(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2$

L1正則化　$r(\mathbf{w}) = \lambda \|\mathbf{w}\|_1$

今回は、L1正則化に話を絞って紹介

劣勾配法で解くと…

- パラメータ更新式

$\mathbf{w}_t$ を0においやる力が働く

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t^{\ell} - \eta'_t \mathbf{g}_t^{r}$$

η_t, η'_t : ステップ幅

$\mathbf{g}_t^{\ell} \in \partial \ell(\mathbf{w}_t)$: ℓの劣勾配中の任意のベクトル

$\mathbf{g}_t^{r} \in \partial r(\mathbf{w}_t)$: rの劣勾配中の任意のベクトル

- L1正則化を用いても、パラメータはSparseになりにくい
  - $\eta_t g_t^{\ell,(j)} + \eta'_t g_t^{r,(j)} = w_t^{(j)}$ が成立することは稀
  - L1正則化項を導入しても、Sparseな解は得られにくい

Forword Backward Splitting(FOBOS)

- 提案手法
  - 重みベクトル $\mathbf{w}_t$ の更新を2ステップに分ける

$\mathbf{w}_t$ ⇨ ステップ1 損失最小化 ⇨ $\mathbf{w}_{t+1/2}$ ⇨ ステップ2 正則化 ⇨ $\mathbf{w}_{t+1}$

■ ステップ1

損失関数 $\ell_t(\mathbf{w}_t)$ が最も小さくなる方向へ重みベクトルを更新

正則化項 $\|\mathbf{w}\|_1$ は考えない

$$\mathbf{w}_{t+1/2} = \mathbf{w}_t - \eta_t \mathbf{g}_t^{\ell}$$

⇦ 劣勾配法

η_t :ステップ幅

$\mathbf{g}_t^{\ell} \in \partial\ell(\mathbf{w}_t)$: ℓの劣勾配中の任意のベクトル

Forword Backward Splitting(FOBOS)

■ ステップ2
ステップ1で更新した重みパラメータをできるだけ動かさず、
L1正則化を行う

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \underbrace{\frac{1}{2}\left\|\mathbf{w} - \mathbf{w}_{t+1/2}\right\|^2}_{\text{パラメータをできるだけ動かさない}} + \underbrace{\eta_{t+1/2}\lambda\|\mathbf{w}\|_1}_{\text{正則化}} \right\}$$

$\eta_{t+1/2}$:ステップ幅

パラメータ更新を2ステップに分けることで、
L1正則化を導入した上で、
Regret上限が$o(T)$のアルゴリズムが導出される

FOBOS更新式の導出

- ステップ2をパラメータ各要素の式に分解

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2} \left\| \mathbf{w} - \mathbf{w}_{t+1/2} \right\|^2 + \eta_{t+1/2} \lambda \left\| \mathbf{w} \right\|_1 \right\} \Leftrightarrow$$

$$w_{t+1}^{(j)} = \arg\min_{w^{(j)}} \left\{ \frac{1}{2} \left(w^{(j)} - w_{t+1/2}^{(j)} \right)^2 + \eta_{t+1/2} \lambda \left| w^{(j)} \right| \right\}$$

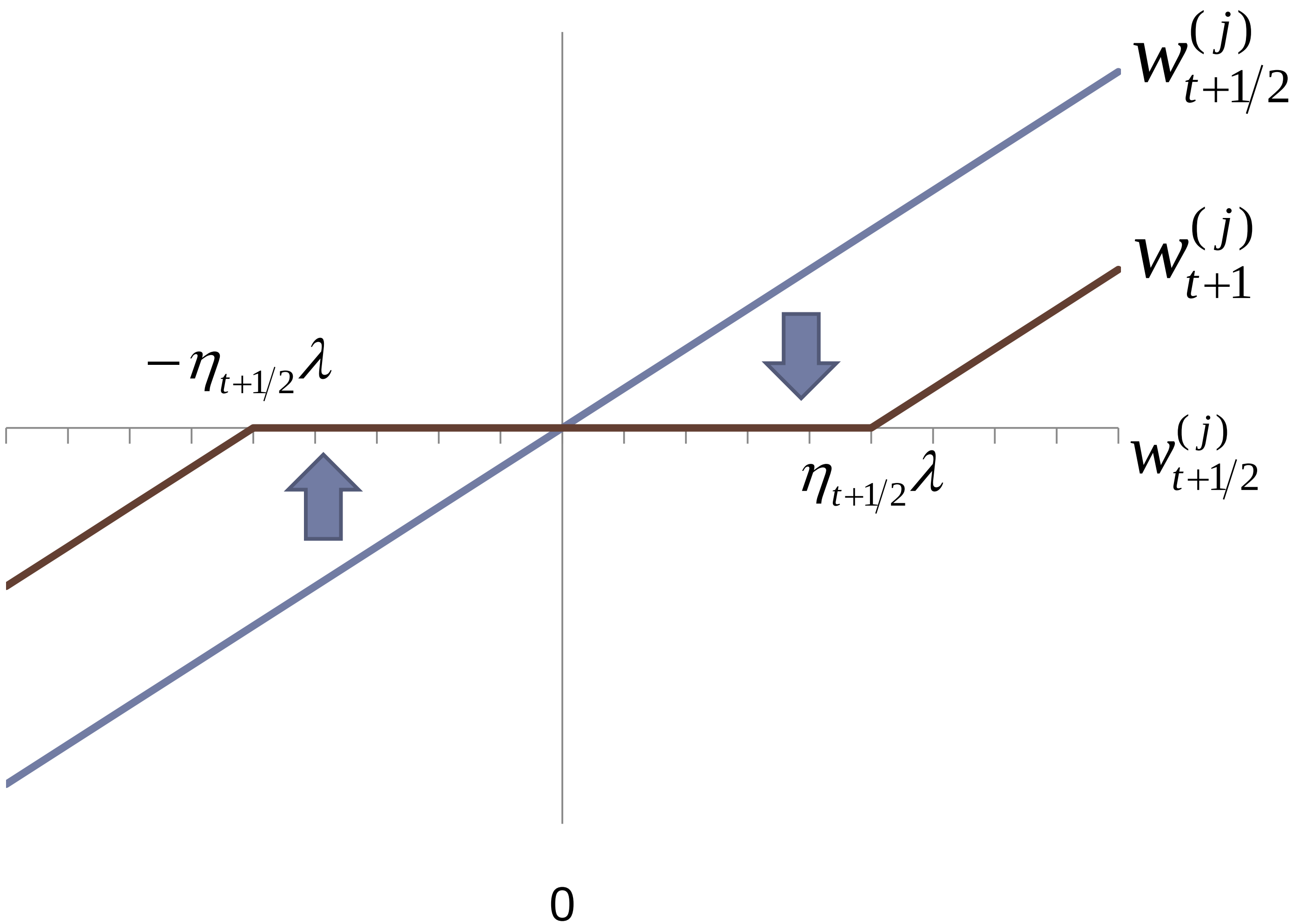
- 閉じた更新式を導出可能

$$\begin{aligned} w_{t+1}^{(j)} &= sign(w_{t+1/2}^{(j)}) \left[\left| w_{t+1/2}^{(j)} \right| - \eta_{t+1/2} \lambda \right]_+ \\ &= \underline{sign(w_t^{(j)} - \eta_t g_t^{\ell,(j)}) \left[\left| w_t^{(j)} - \eta_t g_t^{\ell,(j)} \right| - \eta_{t+1/2} \lambda \right]_+} \end{aligned}$$

$w_t^{(j)}$から$w_{t+1}^{(j)}$を直接導くことが出来る

  - $\left| w_{t+1/2}^{(j)} \right| < \eta_{t+1/2} \lambda \rightarrow 0$になる

FOBOSによるL1正則化



FOBOSの貢献

- Subgradient Method等の既存手法と比べても、実質的に計算量を増やすことなくパラメータ更新が可能
  - 閉じた式でパラメータ更新が可能

$$\boxed{w_{t+1}^{(j)} = sign\left(w_t^{(j)} - \eta_t g_t^{\ell,(j)}\right)\left[\left|w_t^{(j)} - \eta_t g_t^{\ell,(j)}\right| - \eta_{t+1/2}\lambda\right]_+}$$

- Subgradient Methodの$Regret$の性質が、FOBOSでも同じく成立する
  - Subgradient Methodと同じ$Regret$を得ることが出来る

目次

- はじめに
  - 紹介論文概要
  - 問題設定：教師あり学習
- Online学習／L1正則化
  - Online学習
  - L1正則化
- Forward Backward Splitting(FOBOS)
  - FOBOS Algorithm
  - *Regret* 分析
- 実験
- まとめ

Online学習アルゴリズムの評価（再掲）

- *Regret* という概念を導入
  - 元々は、ゲーム理論等で使用されていた枠組み
  - 学習をする過程で蓄積した累積損失と、データを全て見た後で重みベクトルを定めた時の最小合計損失との差

$$Regret(T) = \sum_{t=1}^{T} \left\{ \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \right\} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{t=1}^{T} \ell_t(\mathbf{w})$$

  - *Regret* の上限が $o(T)$ならば、更新を重ねるごとに1データ当たりの *Regret* は0に収束する→最適解に収束する

$$Regret(T) = o(T) \Leftrightarrow \lim_{T \to \infty} \frac{\sum_{t=1}^{T} \left\{ \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \right\}}{T} = 0$$

  - *Regret* の上限を小さなオーダーで抑える事が出来るアルゴリズムは、重みベクトルを高速に最適解へ収束させられる

Regret 分析(1/2)

Regret の上限を0に近づけたい

- *Regret*

$$R_{\ell+r}(T)=\sum_{t=1}^{T}\left[\ell_t(\mathbf{w}_t)+r(\mathbf{w}_t)-\left(\ell_t(\mathbf{w}^*)+r(\mathbf{w}^*)\right)\right]$$

$$\mathbf{w}^*=\arg\min_{\mathbf{w}}\sum_{t=1}^{T}\left[\ell_t(\mathbf{w})+r(\mathbf{w})\right]$$

- FOBOSにおける*Regret* の上限

$\forall \mathbf{w}_t \quad \left\|\mathbf{w}_t-\mathbf{w}^*\right\|\le D \quad \eta_{t+1}\le\eta_{t+1/2}\le\eta_t$

$\partial\ell_t,\partial r\le G \qquad \eta_t\le 2\eta_{t+1}$ の条件下では、$O\left(\sqrt{T}\right)$ が保証される

$$\forall c\ge 0 \quad \eta_t=c/\sqrt{t}\Rightarrow$$

$$R_{\ell+r}(T)\le 2GD+\left(\frac{D^2}{2c}+8G^2c\right)\sqrt{T}$$

Regret 分析(2/2)

$r(\mathbf{w}) = \|\mathbf{w}\|_1$は、Strongly Convexではない

- $\ell_t(\cdot) + r(\cdot)$ がStrongly Convexの場合
  - つまり、$\ell_t(\cdot)$または $r(\cdot)$ がStrongly Convexの場合

Strongly Convex

$$\exists H > 0 \quad \forall \mathbf{w}, \mathbf{w}_t \quad f(\mathbf{w}) \geq f(\mathbf{w}_t) + \langle \nabla f(\mathbf{w}_t), \mathbf{w} - \mathbf{w}_t \rangle + \frac{H}{2} \|\mathbf{w} - \mathbf{w}_t\|^2$$

- *Regret* は $O(\log T)$ で上から押さえる事ができる

$$\eta_t = \frac{1}{Ht} \Rightarrow$$

$$R_{\ell+r}(T) \leq 2GD + HD^2 + \frac{4G^2}{H}\left(1 + \log T\right) = O\left(\frac{G^2}{H} \log T\right)$$

目次

- はじめに
  - 紹介論文概要
  - 問題設定:教師あり学習
- Online学習／L1正則化
  - Online学習
  - L1正則化
- Forward Backward Splitting(FOBOS)
  - FOBOS Algorithm
  - Regret分析
- 実験
- まとめ

実験概要

- Amazon.comのデータセットで評価分類 [J. Blitzer+ 2007]
  - 原論文の実験とは異なり、追実験を行った
  - 損失関数はHinge-Loss, $\eta_t = 1/\sqrt{t}, \ \eta_{t+\frac{1}{2}} = \eta_t, \ \lambda = 1/200$
  - 10回の交差検定、20回反復計算
  - FOBOSとSubgradient Methodで精度とSparseさを比較
  - データは、(レビュー文章, Positive/Negative)の組の集合

	データ数	特徴次元数	代表的な単語
books	4465	332441	book, read, like, story, good, author, pages
dvd	3586	282901	movie, film, see, best, original, character
electronics	5681	235798	sound, product, work, quality, buy, iPod, headphones
kitchen	5945	205666	use, pan, coffee, product, machine, little

実験結果

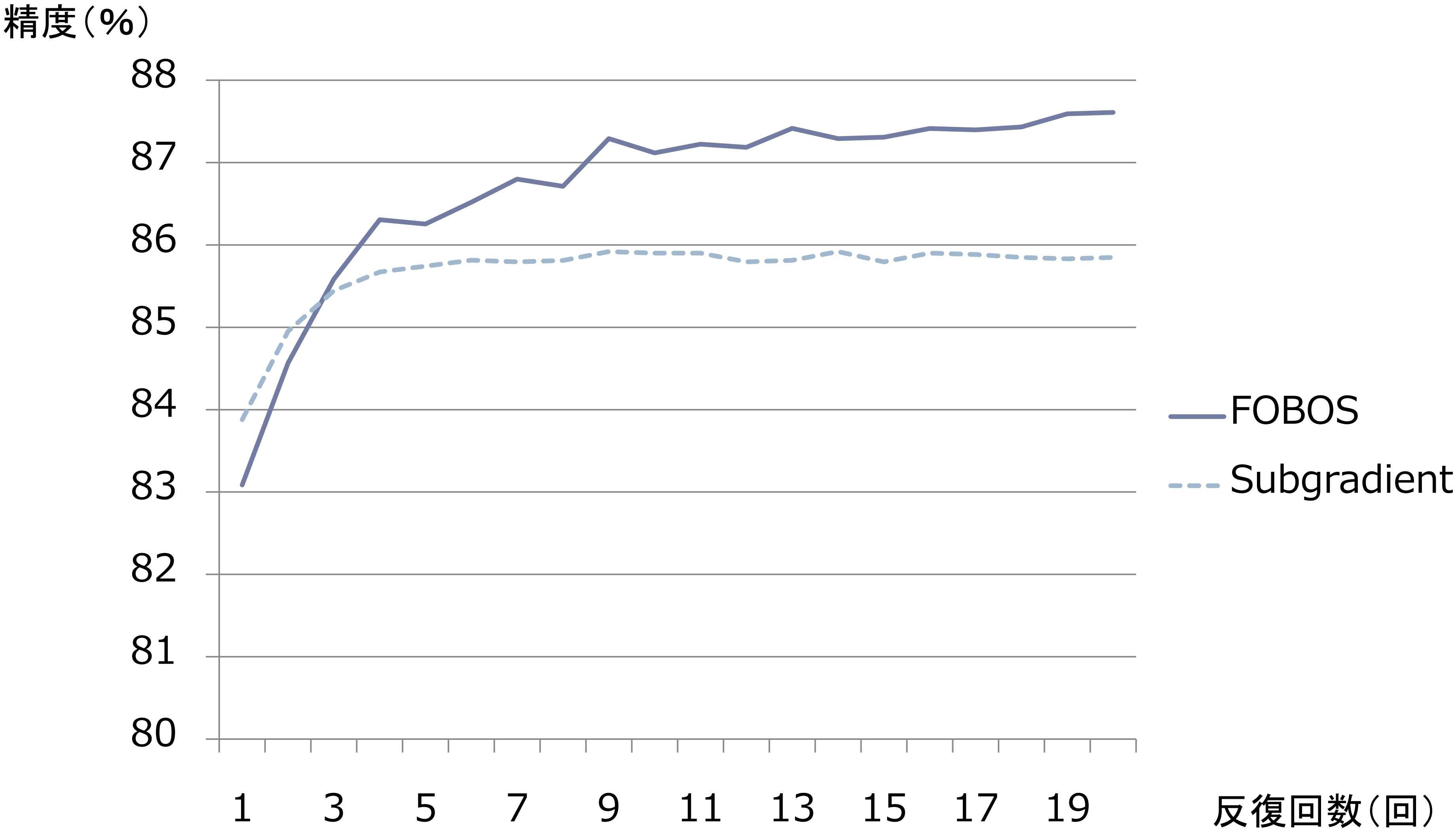
- 20回反復計算を10回の交差検定した結果の平均値

	FOBOS (L1)	Subgradient Method
books	82.84 (**92.87**)	**83.66** (50.39)
dvd	**82.12** (**92.24**)	81.37 (50.42)
electronics	**87.61** (**92.44**)	85.85 (54.43)
kitchen	**88.44** (**92.64**)	87.91 (56.53)

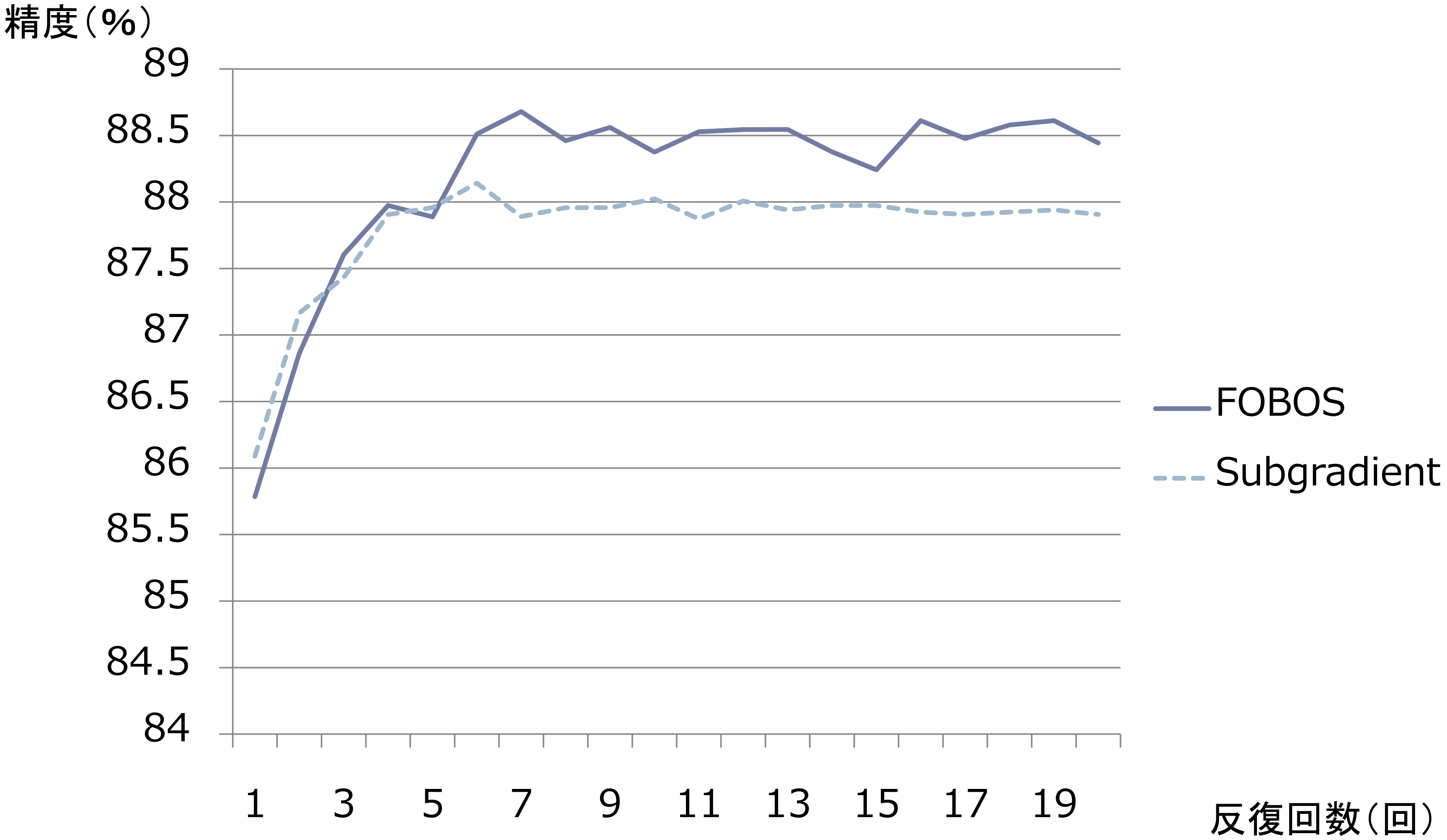
精度 (重みベクトル中の0要素の割合)

精度を落とすことなく、
Sparseな解を得られている

実験結果詳細(electronics)



実験結果詳細(kitchen)

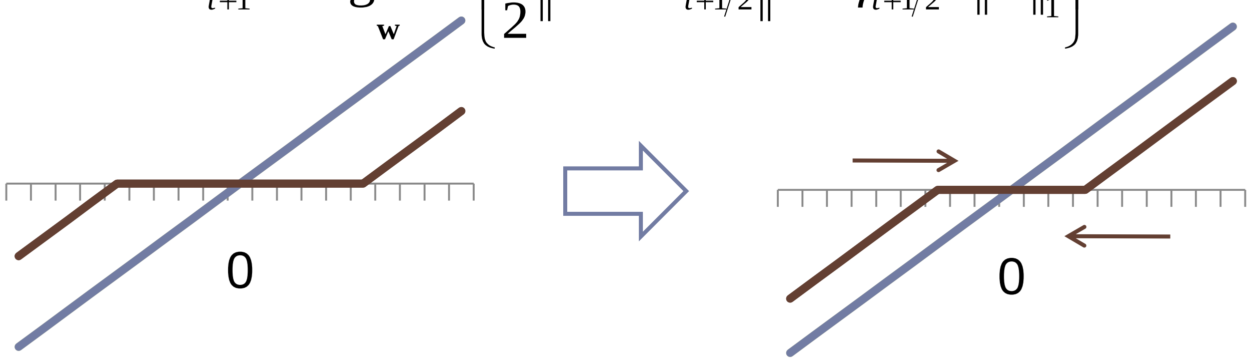


実験結果まとめ

- FOBOSはSubgradient Method等と比べて、精度を上回ることがある
- FOBOSでは、過学習を抑制する効果を持つ
  - Subgradient Methodでは、数回反復を行うとパラメータ更新が殆ど行われなくなる（訓練集合に過剰適合してしまう）
  - 一方FOBOSでは、反復試行を繰り返すことで正則化の力が弱まるため、稀だが有用なパラメータが残る

$$\boxed{\eta_{t+1/2} \to 0}$$

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2} \left\| \mathbf{w} - \mathbf{w}_{t+1/2} \right\|^2 + \eta_{t+1/2} \lambda \left\| \mathbf{w} \right\|_1 \right\}$$



発表のまとめ

- Online学習
  - データを一つ読み込むたびに、逐次的に学習
  - *Regret* という概念で、アルゴリズムを評価
- L1正則化
  - 重みベクトルをSparseにして、特徴選択を行う

- Forward Backward Splitting(FOBOS)
  - Online学習とL1正則化(Sparse化)を同時に実現
  - *Regret* の上限が $O(\sqrt{T})$ となることを証明
  - 評価実験でも、既存手法を上回る精度であることを確認