Multi-parametric Programming in Machine Learning

烏山 昌幸

名古屋工業大学，竹内研究室 D3

2010 年 11 月 22 日

はじめに

- パラメトリック計画法
  - KKT 条件を監視して最適解の変化を解析
- 機械学習におけるパラメトリック計画法
  - Regularization path, solution path, path following, LARS-LASSO, incremental decremental algorithm
  - 基本的に原理は全部同じ
- 一度に複数のパラメータを動かす
  Multi-parametric なアプローチの例はほぼなし
- Multi-parametric アプローチで何ができるか？

Contents

Contents

Incremental Decremental Learning of SVM

- データ点の出入りがあった時に SVM の解を更新
  (Cauwenberghs and Poggio, NIPS 01)
- 区分線形性を利用したアルゴリズム
- 分類以外のタスクへの拡張
  - SV Regression
    (Martin, Techrep 02; Ma and Theiler, Neural Comput. 03)
  - One Class SVM
    (Laskov et al, JMLR 06)
- 全て一度に 1 つのデータ点のみ追加/削除
- 複数データ点の追加/削除を一気に扱う拡張
  (Karasuyama and Takeuchi, NIPS 09)

問題設定

- 訓練データ $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, ただし $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$, $y_i \in \{-1, +1\}$
- 判別関数:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \Phi(\boldsymbol{x}) + b,$$

  ただし, $\boldsymbol{w} \in \mathcal{F}$, $\Phi : \mathcal{X} \to \mathcal{F}$
- カーネル関数:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x}_j)$$

SVM

- 主問題

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi\}_{i=1}^n} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i\{\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b\} \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, n. \end{aligned}$$

C は正則化パラメータ

- 双対問題

$$\begin{aligned} \max_{\{\alpha_i\}_{i=1}^n} \quad & -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j Q_{ij} + \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n y_i\alpha_i = 0, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n, \end{aligned}$$

ただし，$Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

(Single) Incremental Decremental Algorithm (1)

(Cauwenberghs and Poggio, NIPS 01)

- あるデータ点 $(\boldsymbol{x}_c, y_c)$ を追加 or 削除
  - $\{\alpha_i\}_{i=1}^n$ と $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ それぞれ対応関係
  - $\alpha_i = 0$ となるようなデータ点 $(\boldsymbol{x}_i, y_i)$ は
    目的関数に影響を及ぼさない（なくしても解が変わらない）

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_c \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} * \\ \vdots \\ * \\ \vdots \\ * \end{bmatrix} \Rightarrow \begin{bmatrix} *' \\ \vdots \\ 0 \\ \vdots \\ *' \end{bmatrix}, \qquad \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_c \end{bmatrix} = \begin{bmatrix} * \\ \vdots \\ * \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} *' \\ \vdots \\ *' \\ *' \end{bmatrix}$$

削除: α_c を減らして 0 に　　追加: α_c を 0 から増やす

  - 最適性監視しながらパラメータ更新

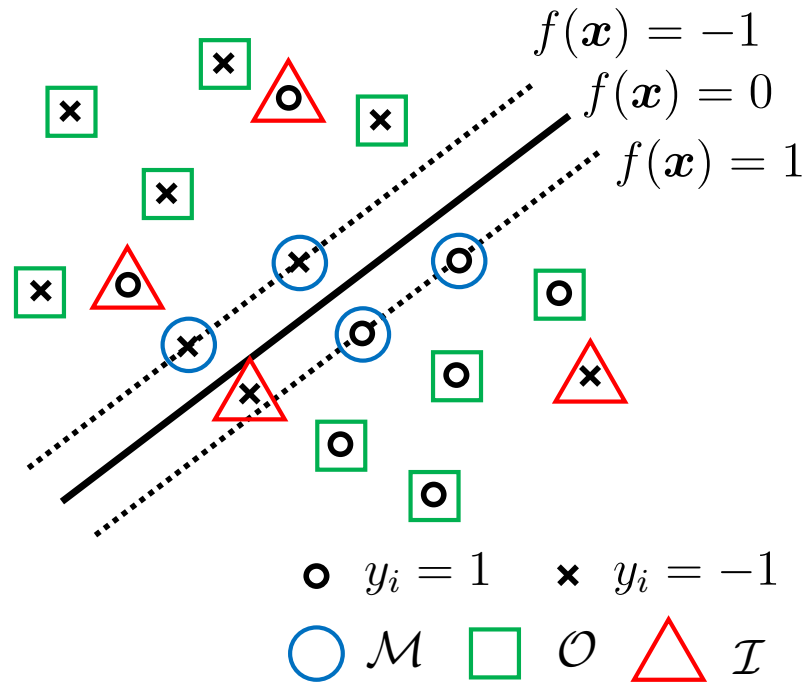
(Single) Incremental Decremental Algorithm (2)

- 最適性条件 (等式の条件 と 不等式の条件)

$$
\begin{aligned}
&\boldsymbol{y}^\top \boldsymbol{\alpha} = 0, && && \text{等式制約}\\
&y_i f(\boldsymbol{x}_i) = 1, && 0 \le \alpha_i \le C, && \text{for } i \in \mathcal{M}, && \text{マージン上 (下図○)}\\
&y_i f(\boldsymbol{x}_i) \ge 1, && \alpha_i = 0, && \text{for } i \in \mathcal{O}, && \text{マージン外側 (下図□)}\\
&y_i f(\boldsymbol{x}_i) \le 1, && \alpha_i = C, && \text{for } i \in \mathcal{I}. && \text{マージン内側 (下図△)}
\end{aligned}
$$

- 等式の条件 をまとめた連立方程式:

$$
\begin{aligned}
\begin{bmatrix} 0 \\ \boldsymbol{y}_{\mathcal{M}} \end{bmatrix} b + \begin{bmatrix} \boldsymbol{y}^\top \\ \boldsymbol{Q}_{\mathcal{M},:} \end{bmatrix} \boldsymbol{\alpha} &= \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \\
\boldsymbol{\alpha}_{\mathcal{O}} &= \mathbf{0}, \\
\boldsymbol{\alpha}_{\mathcal{I}} &= C\mathbf{1}
\end{aligned}
$$



- 正しい $\mathcal{M}, \mathcal{O}, \mathcal{I}$ が分かっていれば,
  連立方程式解いて解が計算できる

(Single) Incremental Decremental Algorithm (3)

- α_c だけ取り出してみる

$$\begin{bmatrix} 0 \\ \boldsymbol{y}_{\mathcal{M}} \end{bmatrix} b + \begin{bmatrix} \boldsymbol{y}^\top \\ \boldsymbol{Q}_{\mathcal{M},:} \end{bmatrix} \boldsymbol{\alpha} + \begin{bmatrix} y_c \\ \boldsymbol{Q}_{\mathcal{M},c} \end{bmatrix} \alpha_c = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix},$$
$$\boldsymbol{\alpha}_{\mathcal{O}} = \mathbf{0},$$
$$\boldsymbol{\alpha}_{\mathcal{I}} = C\mathbf{1}.$$

- α_c が $\Delta\alpha_c$ だけ変化した時の変化分が

$$\begin{bmatrix} 0 \\ \boldsymbol{y}_{\mathcal{M}} \end{bmatrix} \Delta b + \begin{bmatrix} \boldsymbol{y}^\top \\ \boldsymbol{Q}_{\mathcal{M},:} \end{bmatrix} \boldsymbol{\Delta\alpha} + \begin{bmatrix} y_c \\ \boldsymbol{Q}_{\mathcal{M},c} \end{bmatrix} \Delta\alpha_c = \begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix},$$
$$\boldsymbol{\Delta\alpha}_{\mathcal{O}} = \mathbf{0},$$
$$\boldsymbol{\Delta\alpha}_{\mathcal{I}} = \mathbf{0},$$

となっていれば等式の条件は保持される，不等式の条件は？

(Single) Incremental Decremental Algorithm (4)

- α_i の変化と $f(\boldsymbol{x}_i)$ の変化は $\Delta\alpha_c$ に対して線形
  - $\boldsymbol{\Delta\alpha}_{\mathcal{O}} = \boldsymbol{0}$, $\boldsymbol{\Delta\alpha}_{\mathcal{I}} = \boldsymbol{0}$ 代入して連立方程式解くと

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = \boldsymbol{a}\Delta\alpha_c, \text{ where } \boldsymbol{a} = -\boldsymbol{M}^{-1}\begin{bmatrix} y_c \\ \boldsymbol{Q}_{\mathcal{M},c} \end{bmatrix},$$

$$y_i\Delta f(\boldsymbol{x}_i) = \phi_i\Delta\alpha_c, \text{ where } \phi_i = \begin{bmatrix} 1 & \boldsymbol{Q}_{i,\mathcal{M}} \end{bmatrix}\boldsymbol{a},$$

ただし， $\boldsymbol{M} = \begin{bmatrix} 0 & \boldsymbol{y}_{\mathcal{M}}^\top \\ \boldsymbol{y}_{\mathcal{M}} & \boldsymbol{Q}_{\mathcal{M},\mathcal{M}} \end{bmatrix}$.

- $\Delta\alpha_c$ をどれだけ動かすと
  不等式の条件 が満たされなくなるか計算

$$\begin{aligned} \alpha_{m_i} + a_{i+1}\Delta\alpha_c &\in [0, C], & i &= 1, \ldots, |\mathcal{M}|, \\ y_i f(\boldsymbol{x}_i) + \phi_i\Delta\alpha_c &\geq 1, & i &\in \mathcal{O}, \\ y_i f(\boldsymbol{x}_i) + \phi_i\Delta\alpha_c &\leq 1, & i &\in \mathcal{I}, \end{aligned}$$

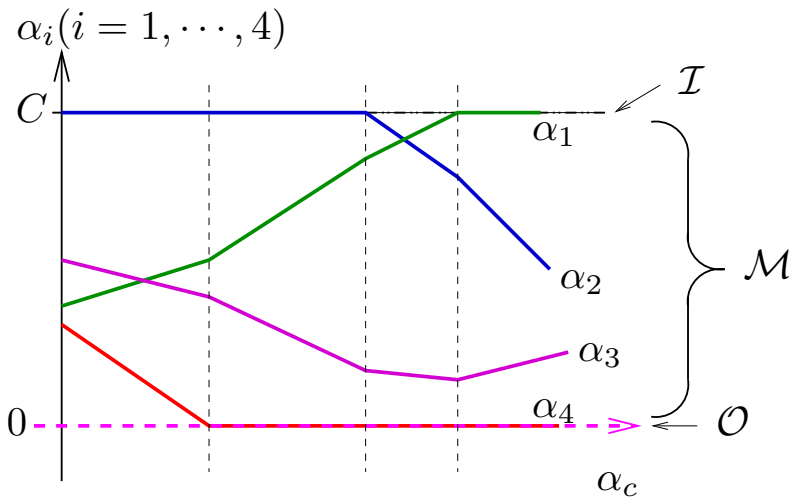
ただし， $\mathcal{M} = \{m_1, \ldots, m_{|\mathcal{M}|}\}$.

(Single) Incremental Decremental Algorithm (5)

- 不等式の条件 が満たさなくなったらどうするか

  (例) $\alpha_i > 0,\ i \in \mathcal{M}$ が $\alpha_i + \Delta\alpha_i = 0$ になったら,

  $\Rightarrow$ i を $\mathcal{O}$ へ移動



- アルゴリズムの流れ (追加の場合)

  1. $\alpha_c = 0$
  2. $\Delta\alpha_c$ を 不等式の条件 が満たされている範囲で最大化
     範囲内で α_c が最適性を満たせば終了
  3. $\boldsymbol{\alpha}$, b と $\mathcal{M}$, $\mathcal{O}$, $\mathcal{I}$ 更新.
     step 2 に戻る

問題点

- 1 点 (x_c, y_c) の追加/削除はできるが,
  複数点の追加/削除したい場合はどうするか？
- 1 点ずつ追加/削除
  - 2 点削除の例

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \Rightarrow \begin{bmatrix} *' \\ *' \\ *' \\ *' \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} *'' \\ *'' \\ *'' \\ 0 \\ 0 \end{bmatrix},$$

Multiple Incremental Decremental Algorithm (1)

(Karasuyama and Takeuchi, NIPS 09)

- 複数のデータ点を一気に追加 and/or 削除
  - 追加するデータ点の集合 $\mathcal{A} = \{a_1, \cdots, a_m\}$
  - 削除するデータ点の集合 $\mathcal{R} = \{r_1, \cdots, r_\ell\}$
- $\boldsymbol{\alpha}_{\mathcal{A}}$ を $\mathbf{0}$ から少しずつ増やす，$\boldsymbol{\alpha}_{\mathcal{R}}$ を $\mathbf{0}$ へと少しずつ減らす

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_{a_1} \\ \vdots \\ \alpha_{a_m} \\ \alpha_{r_1} \\ \vdots \\ \alpha_{r_\ell} \end{bmatrix} = \begin{bmatrix} * \\ \vdots \\ * \\ 0 \\ \vdots \\ 0 \\ * \\ \vdots \\ * \end{bmatrix} \Rightarrow \begin{bmatrix} *' \\ \vdots \\ *' \\ *' \\ \vdots \\ *' \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Multiple Incremental Decremental Algorithm (2)

- Single の場合と同じ式展開
- 等式の条件 を保持するには

$$\begin{bmatrix} 0 \\ \boldsymbol{y}_{\mathcal{M}} \end{bmatrix} \Delta b + \begin{bmatrix} \boldsymbol{y}^\top \\ \boldsymbol{Q}_{\mathcal{M},:} \end{bmatrix} \boldsymbol{\Delta\alpha} + \begin{bmatrix} \boldsymbol{y}_{\mathcal{A}}^\top & \boldsymbol{y}_{\mathcal{R}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{A}} & \boldsymbol{Q}_{\mathcal{M},\mathcal{R}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\alpha}_{\mathcal{A}} \\ \boldsymbol{\Delta\alpha}_{\mathcal{R}} \end{bmatrix} = \boldsymbol{0},$$
$$\boldsymbol{\Delta\alpha}_{\mathcal{O}} = \boldsymbol{0},$$
$$\boldsymbol{\Delta\alpha}_{\mathcal{I}} = \boldsymbol{0},$$

- 解くと $\boldsymbol{\Delta\alpha}_{\mathcal{A}}$, $\boldsymbol{\Delta\alpha}_{\mathcal{R}}$ とその他の $\boldsymbol{\alpha}$ の関係性わかる

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_{\mathcal{A}}^\top & \boldsymbol{y}_{\mathcal{R}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{A}} & \boldsymbol{Q}_{\mathcal{M},\mathcal{R}} \end{bmatrix} \underbrace{\begin{bmatrix} \boldsymbol{\Delta\alpha}_{\mathcal{A}} \\ \boldsymbol{\Delta\alpha}_{\mathcal{R}} \end{bmatrix}}_{(*)}$$

- (*) の部分が Single 時はスカラで
  増やす (追加) か減らす (削除) だけでパス追跡できた
- $\boldsymbol{\Delta\alpha}_{\mathcal{A}}$, $\boldsymbol{\Delta\alpha}_{\mathcal{R}}$ はベクトルだけどどうするか？

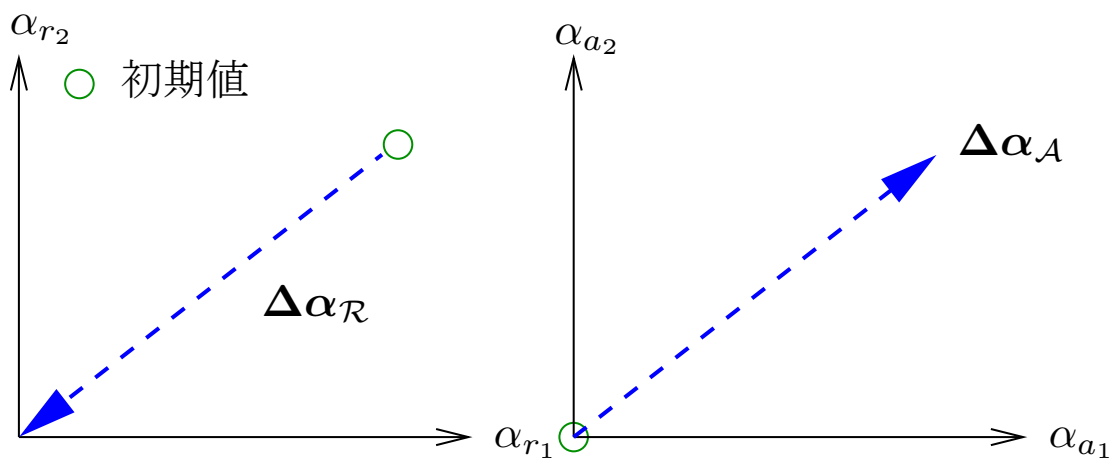
Multiple Incremental Decremental Algorithm (3)

- Multiple の場合は $\boldsymbol{\alpha}_{\mathcal{R}}$, $\boldsymbol{\alpha}_{\mathcal{A}}$ の方向が必要

$$\begin{bmatrix} \boldsymbol{\Delta\alpha}_{\mathcal{A}} \\ \boldsymbol{\Delta\alpha}_{\mathcal{R}} \end{bmatrix} = \theta \begin{bmatrix} C\mathbf{1} \\ -\boldsymbol{\alpha}_{\mathcal{R}}^{\mathrm{bgn}} \end{bmatrix}.$$

$\boldsymbol{\alpha}_{\mathcal{R}}^{\mathrm{bgn}}$ は $\boldsymbol{\alpha}_{\mathcal{R}}$ の削除前の値，$\theta \geq 0$ はステップ幅

  - $\boldsymbol{\alpha}_{\mathcal{R}}$ に関しては α の空間で最短 ($\boldsymbol{\alpha}_{\mathcal{R}} = \mathbf{0}$ に向かって一直線)
  - $\boldsymbol{\alpha}_{\mathcal{A}}$ は最終的な値事前に分からない
    最適解が $\boldsymbol{\alpha}_{\mathcal{A}} = C\mathbf{1}$ なら最短



$\Delta\boldsymbol{\alpha}_{\mathcal{A}} = C\mathbf{1}$ で良い理由

- $\Delta\alpha_i,\ i \in \mathcal{A}$ は $\Delta\alpha_i > 0$ であればよい
  - $i \in \mathcal{A}$ については常に

$$y_i f(\boldsymbol{x}_i) < 1.$$

    なぜなら

    - もし開始時点 $(\alpha_i = 0)$ で $y_i f(\boldsymbol{x}_i) \geq 1$ ならば $i \in \mathcal{O}$ として最適性を満たす
    - もしパス追跡途中 $(\alpha_i < C)$ で $y_i f(\boldsymbol{x}_i) = 1$ になったら $i \in \mathcal{M}$ にできる

  - パス追跡の最後 $(\alpha_i = C)$ まで

$$y_i f(\boldsymbol{x}_i) < 1$$

    なら $i \in \mathcal{I}$ として最適性満たす
- つまり増やしていけば，どこかで必ず最適性満たす

Multiple Incremental Decremental Algorithm (4)

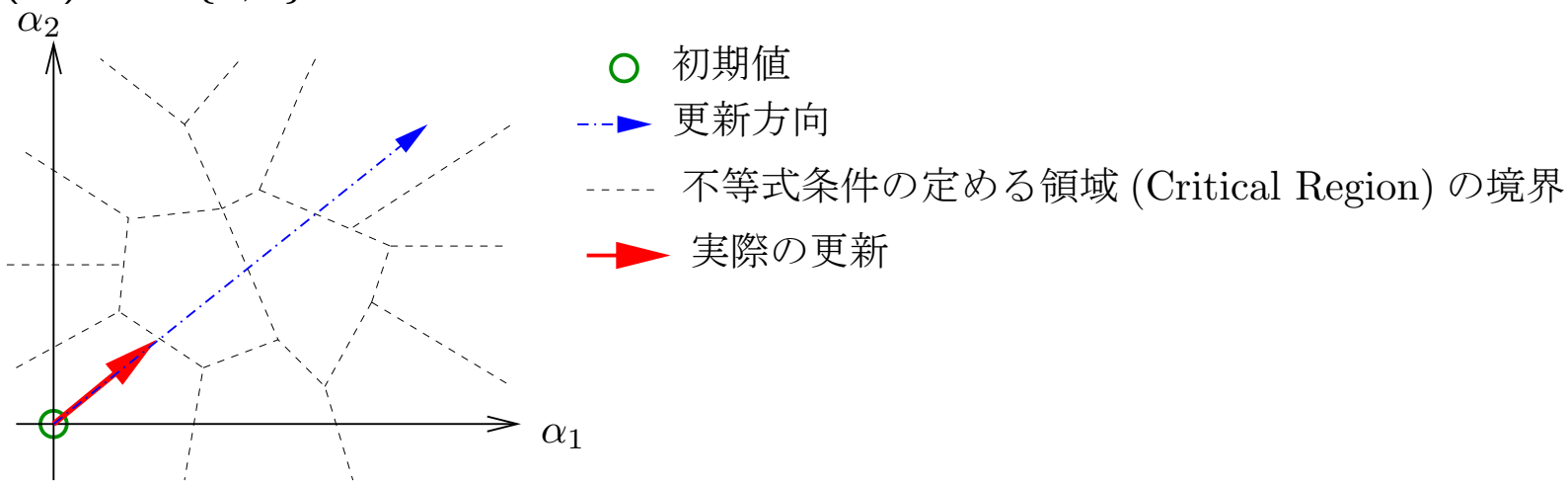
- θ に対して線形な変化

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = \boldsymbol{a}\theta, \text{ where } \boldsymbol{a} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_{\mathcal{A}}^{\top} & \boldsymbol{y}_{\mathcal{R}}^{\top} \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{A}} & \boldsymbol{Q}_{\mathcal{M},\mathcal{R}} \end{bmatrix} \begin{bmatrix} C\mathbf{1} \\ -\boldsymbol{\alpha}_{\mathcal{R}}^{\mathrm{bgn}} \end{bmatrix}$$

$$y_i \Delta f(\boldsymbol{x}_i) = \phi_i \theta, \text{ where } \phi_i = \begin{bmatrix} 1 & \boldsymbol{Q}_{i,\mathcal{M}} \end{bmatrix} \boldsymbol{a},$$

- 不等式の条件 が満たされている範囲で θ を最大化

(例) $\mathcal{A} = \{1, 2\}$

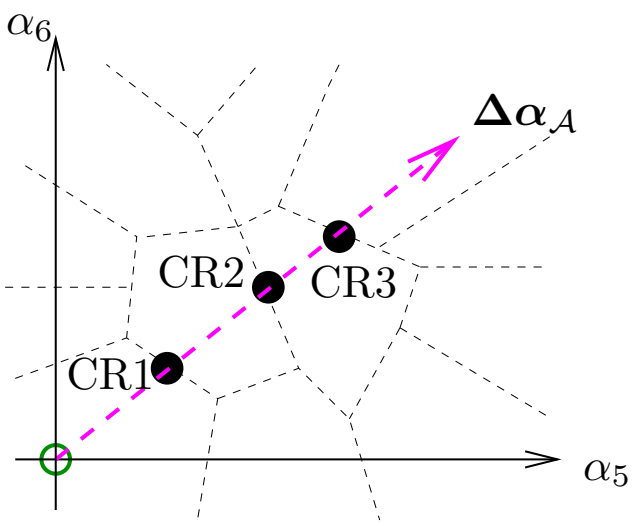


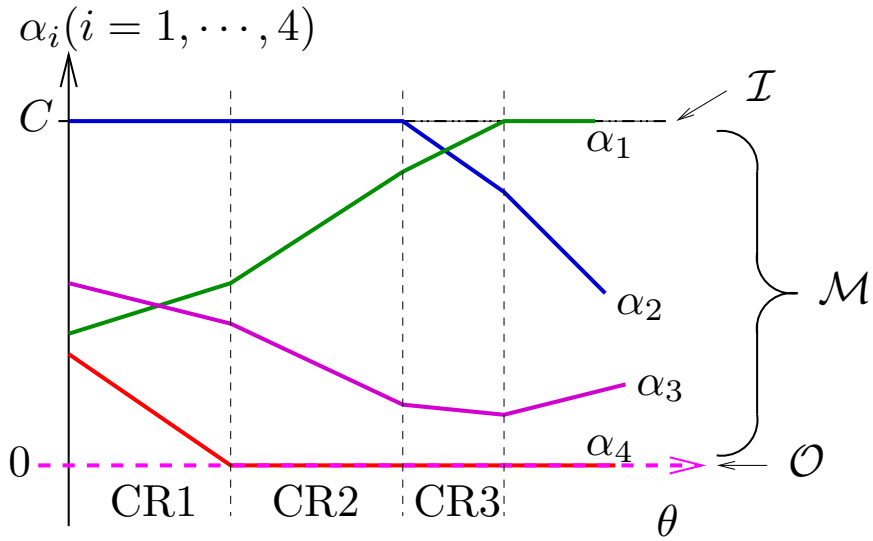
不等式の条件 は全て $\boldsymbol{\alpha}_{\mathcal{A}}, \boldsymbol{\alpha}_{\mathcal{R}}$ に関して一次

$\boldsymbol{\alpha}_{\mathcal{A}}, \boldsymbol{\alpha}_{\mathcal{R}}$ の空間では凸多面体 (Critical Region) として表現

アルゴリズムの流れ

(例) $n = 4,\ \mathcal{A} = \{5, 6\}$

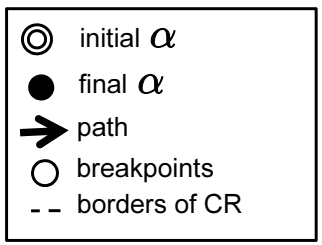


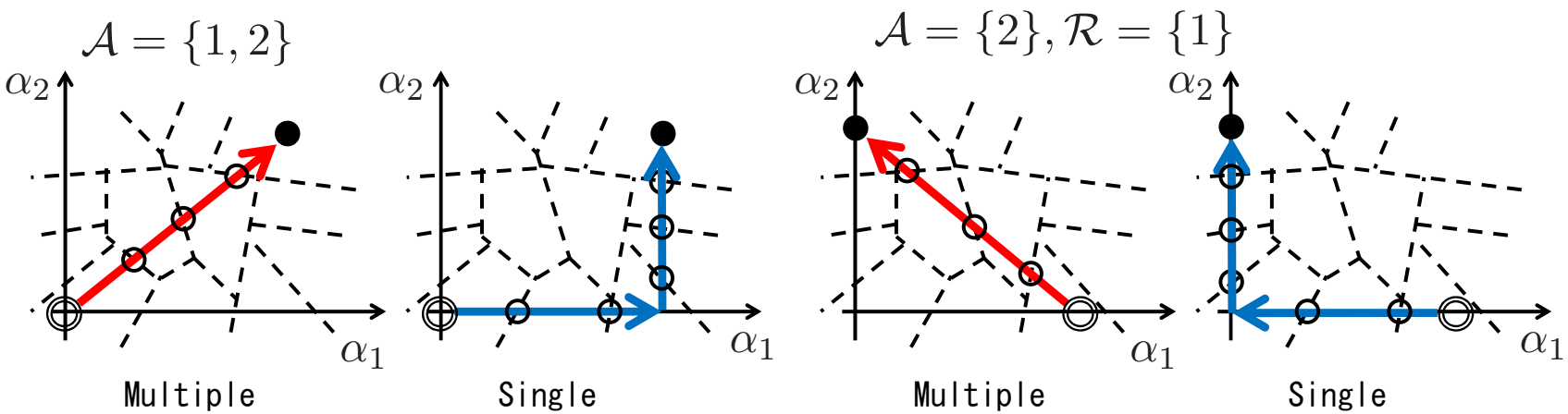


- CR1 から CR2 では $i = 4$ が $\mathcal{M}$ から $\mathcal{O}$ へ
  CR2 から CR3 では $i = 2$ が $\mathcal{I}$ から $\mathcal{M}$ へ
- 計算コスト
  - $\mathcal{M}$ のサイズ変わるので連立方程式が拡大 or 縮小
    Rank-one-update により $O(|\mathcal{M}|^2)$ で計算可能
  - 不等式のチェック $O(n)$
  - 全体のコスト $= O($ブレイクポイント数 $\times\ n|\mathcal{M}|)$

挙動の比較

- Single では複数データ点扱うには 1 点ずつ更新



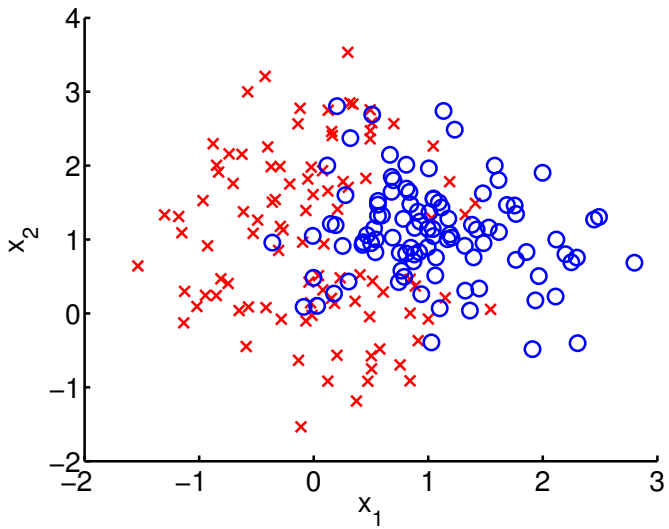


実験

- 人工データ
- オンライン学習
- 交差検証

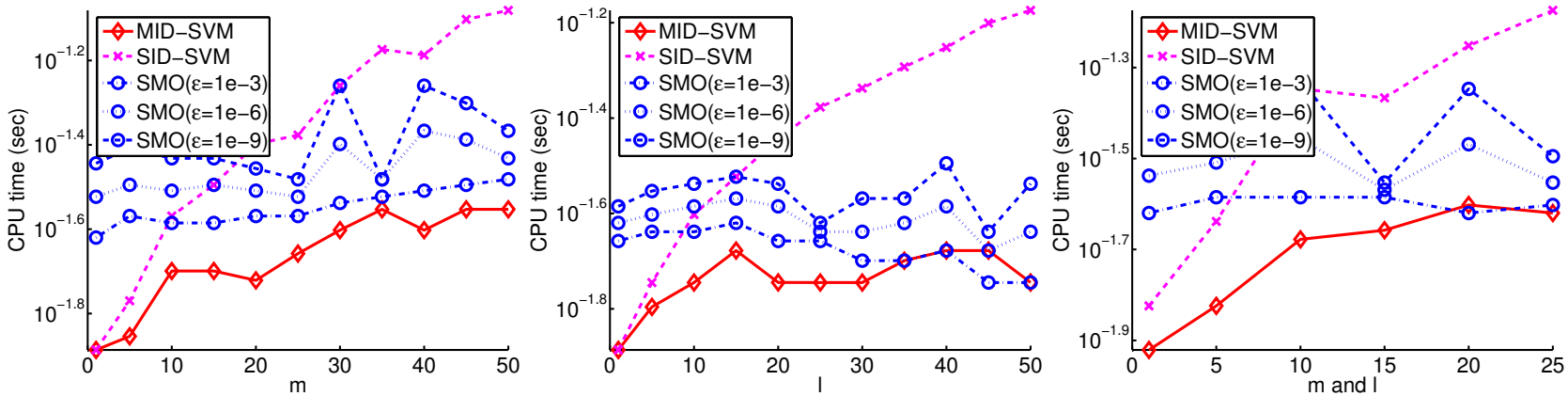
人工データ実験 (1)

- 実験設定
  - 正規分布を使って生成 $n = 500$, $\boldsymbol{x}_i \in \mathbb{R}^2$
  - RBF カーネル $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2)$, $\gamma = 1$
  - 正則化係数 $C = 10$
  - 比較には LIBSVM (SMO)
- MID = Multiple Incremental Decremental,
  SID = Single Incremental Decremental
- 追加・削除するパラメータは $\alpha_i = C$



人工データ実験 (2)

CPU 時間対数プロット



（a）m 個のデータ点を追加（b）ℓ 個のデータ点を削除（c）m 個の追加と ℓ 個の削除

- 全ての結果で MID は SID より早く，
  出入りするデータ数増えるほど差は広がる
- $m = 1, \ell = 0$ または $m = 0, \ell = 1$ の時は MID と SID 一致
- 追加・削除するデータ数が増えると SMO の方が高速
  (変化を追跡するより解きなおした方が速い)

ブレイクポイント数比解析

- 追加のみ $|\mathcal{A}| = m, |\mathcal{R}| = 0$ の場合を例に Mmultiple と Single の breakpoint 数の比を考える

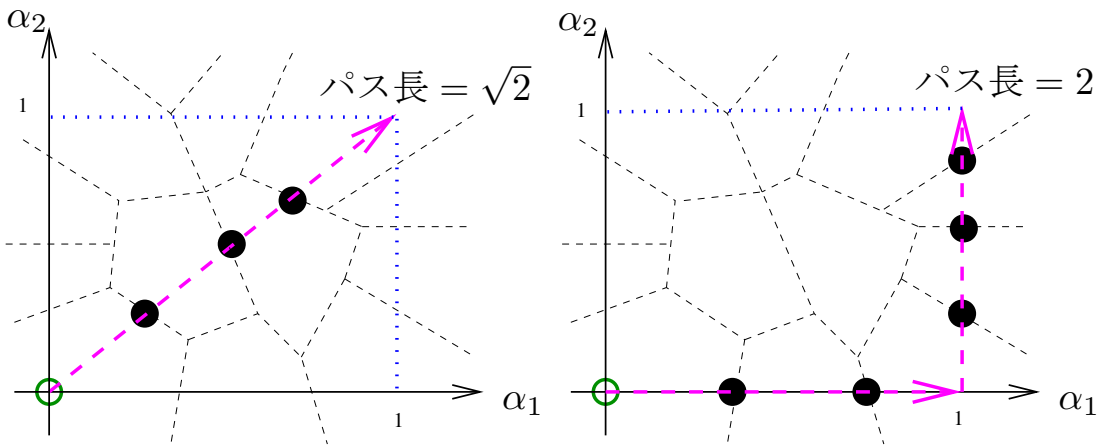
  仮定

  1. breakpoint の数はパス長に比例
  2. MID が最短パスを通る

  の元でなら $\|\boldsymbol{\alpha}_{\mathcal{A}}\|_2 : \|\boldsymbol{\alpha}_{\mathcal{A}}\|_1$

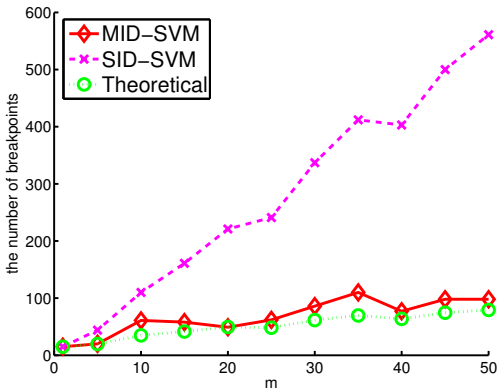
- さらに $\alpha_i = C, i \in \mathcal{A}$ の場合は $\sqrt{m} : m$

  （例）$m = 2, C = 1$

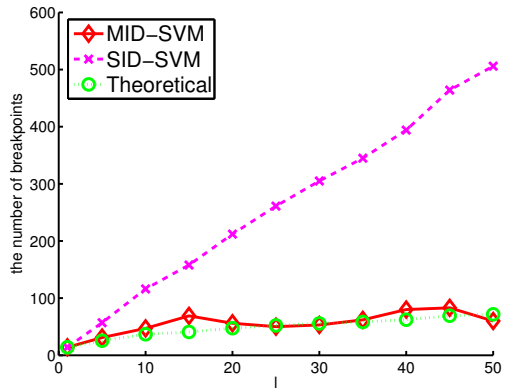


CPU 時間とブレイクポイント数

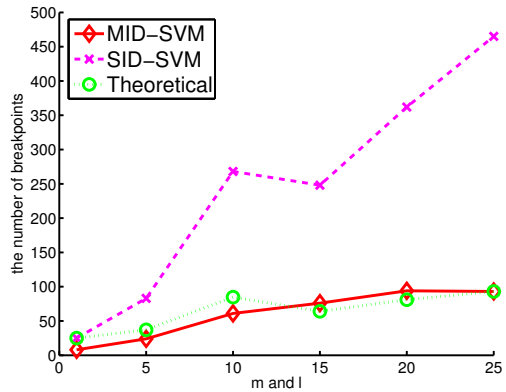
breakpoint の数



（a）m 個のデータ点を追加

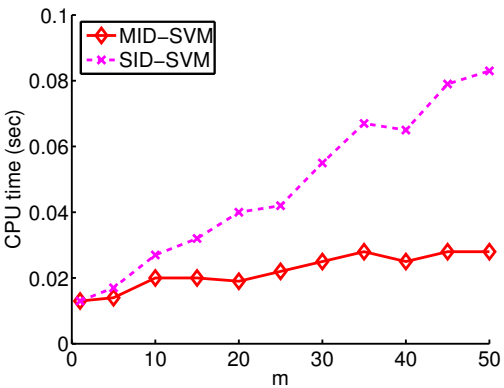


（b）ℓ 個のデータ点を削除

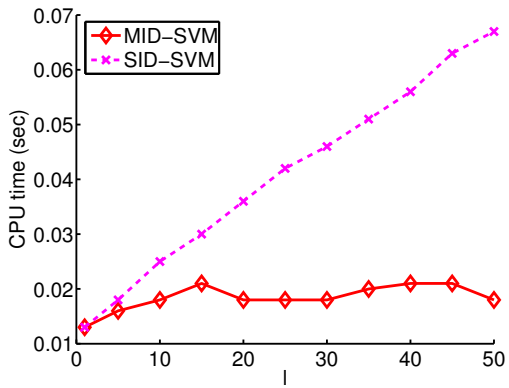


（c）m 個の追加と ℓ 個の削除

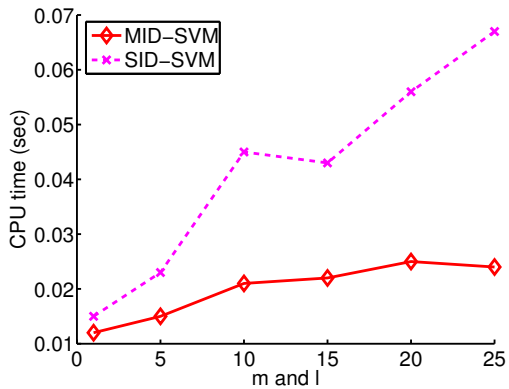
CPU 時間 (sec)



（a）m 個のデータ点を追加



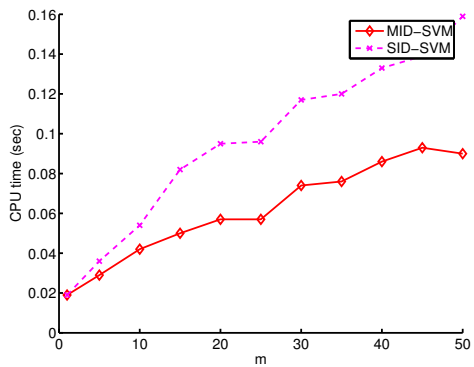
（b）ℓ 個のデータ点を削除



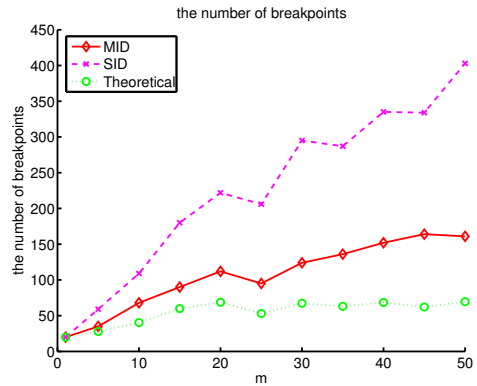
（c）m 個の追加と ℓ 個の削除

仮定が成り立たない場合

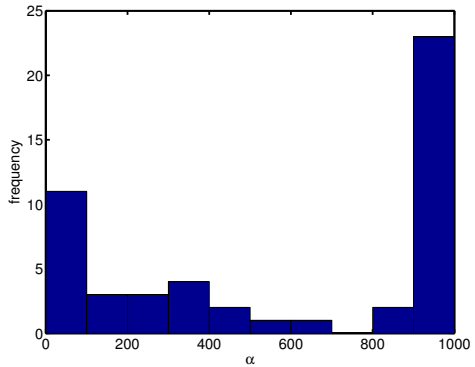
- 追加するデータ点 $\alpha_i > 0, i \in \mathcal{A}$ ($\alpha_i = C$ とは限らない)
  - $\gamma = 5, C = 1000$



(a) CPU 時間 (sec)



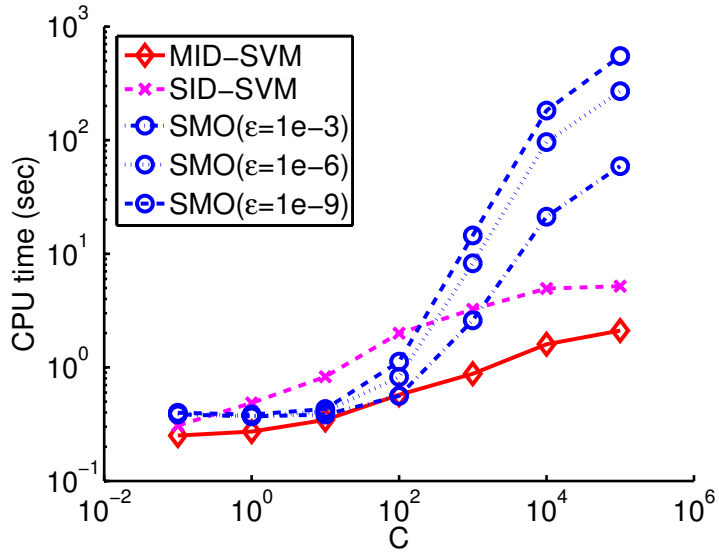
(b)breakpoint の数



(c)$\alpha_i, i \in \mathcal{A}$ の値のヒストグラム ($m = 50$)

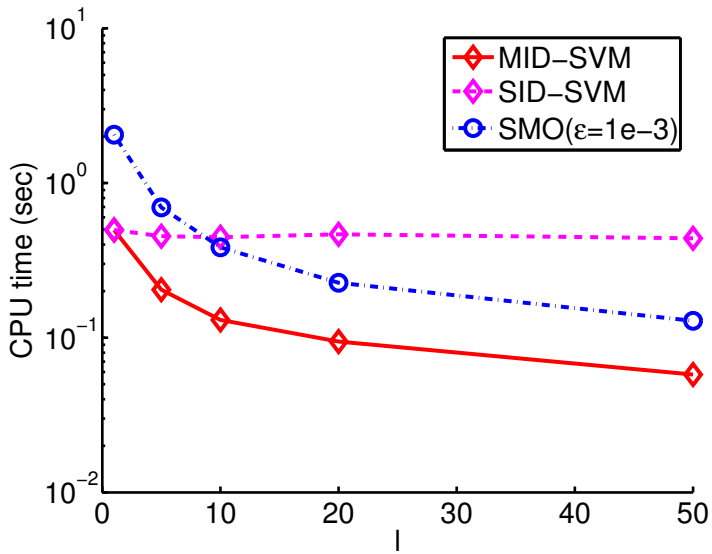
応用例 (1): 時系列オンライン学習

- 新しいデータ点の追加とともに古いデータ点を削除
  - Fisher river データ $n = 1423$, $\boldsymbol{x}_i \in \mathbb{R}^{21}$
  - 川の水量の増加/減少を過去の気象情報から予測
  - 30 日分の新規データの追加と削除
- 追加/削除する α_i に特に条件なし



応用例 (2): 交差検証

- 交差検証誤差の計算高速化
  1. 全てのデータ点使った解を計算
  2. ℓ 個抜き交差検証の場合,
     ℓ 個のデータ点を抜いた解をそれぞれ計算



横軸 ℓ, diabetes data set ($n = 500$, $C = 1$)

- Single だと常に LOO と同じくらい時間かかる

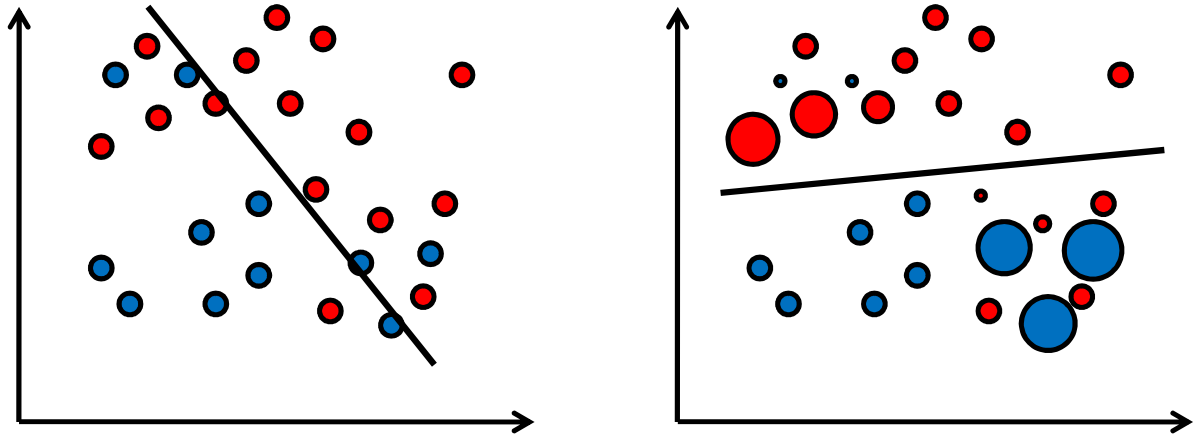
Incremental Decremental SVM まとめ

- Incremental decremental SVM
  (Cauwenberghs and Poggio, NIPS 01)
  の拡張として Multipel Incremental decremental SVM を提案
- 複数データ点の出入りを扱うならこの形が自然
- 仮定の元で解析的にも高速と言える

Contents

Instance-weighted learning

- 各学習データ点が “重み” を持つ



(左) 重み無し (右) 重み付き

  - 重み付き学習事例:
    Non-stationary data analysis, 分散不均一モデリング, 共変量シフト, ランキング学習, 半教師付き学習, etc.
  - 重みが変化: モデル選択, 適応的重み学習, オンライン学習

- “重み” の変化に対するパスを定式化し様々な応用を示す

  (Karasuyama, Harada, Sugiyama and Takeuchi, arXiv 10)

Weighted SVM

- C_i をデータ点個々の重みとして表現

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi\}_{i=1}^n} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{i=1}^n C_i \xi_i \\ \text{s.t.} \quad & y_i\{\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b\} \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, n. \end{aligned}$$

- 双対問題

$$\begin{aligned} \max_{\{\alpha_i\}_{i=1}^n} \quad & -\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j Q_{ij} + \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C_i,\ i = 1, \ldots, n, \end{aligned}$$

最適解の解析的表現

- 重みベクトル $\boldsymbol{c} = [C_1, \ldots, C_n]^\top$ を
  $\boldsymbol{c}^{(\mathrm{old})}$ から $\boldsymbol{c}^{(\mathrm{new})}$ に動かしたい

$$\boldsymbol{c}(\theta) = \boldsymbol{c}^{(\mathrm{old})} + \theta \left(\boldsymbol{c}^{(\mathrm{new})} - \boldsymbol{c}^{(\mathrm{old})} \right), \ \theta \in [0, 1],$$

  θ を 0 から 1 にパス追跡

- 重みベクトル $\boldsymbol{c}$ と最適解の関係

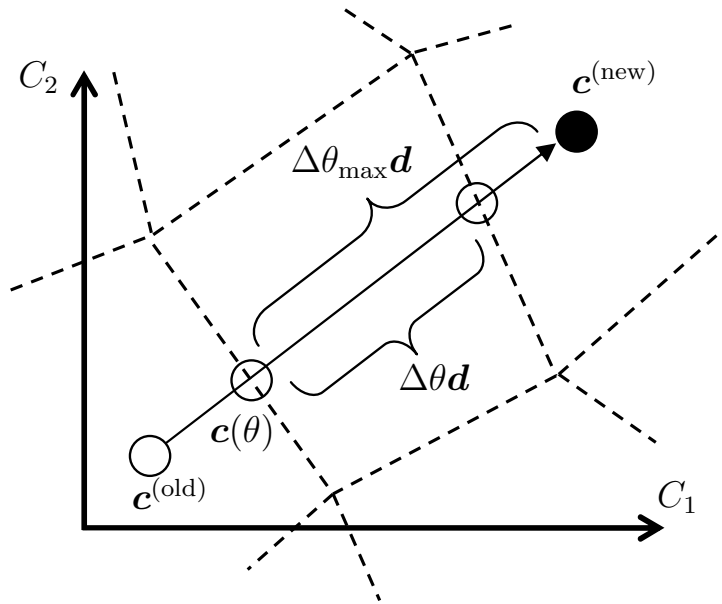
$$\begin{bmatrix} b \\ \boldsymbol{\alpha}_{\mathcal{M}} \end{bmatrix} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_{\mathcal{I}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{I}} \end{bmatrix} \boldsymbol{c}_{\mathcal{I}} + \boldsymbol{M}^{-1} \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix},$$

$$\boldsymbol{\alpha}_{\mathcal{O}} = \mathbf{0},$$

$$\boldsymbol{\alpha}_{\mathcal{I}} = \boldsymbol{c}_{\mathcal{I}}.$$

アルゴリズムの流れ

- $\theta = 0$ から $\theta = 1$ になるまで以下の繰り返し
  1. 現在の θ から進める最大の $\Delta\theta$ を計算
  2. 解と $\mathcal{M}, \mathcal{O}, \mathcal{I}$ を更新



$\boldsymbol{c}$ の空間での動き.

$\boldsymbol{d}$ は進行方向のベクトル: $\boldsymbol{d} = \boldsymbol{c}^{(\mathrm{new})} - \boldsymbol{c}^{(\mathrm{old})}$.

特徴

- $\boldsymbol{c}^{(\mathrm{old})}$, $\boldsymbol{c}^{(\mathrm{new})}$ が与えられれば
  その間の解を調べつくすことが可能
- 解 $\boldsymbol{\alpha}$, b 以外にもモデル選択基準等を同時に追跡可能

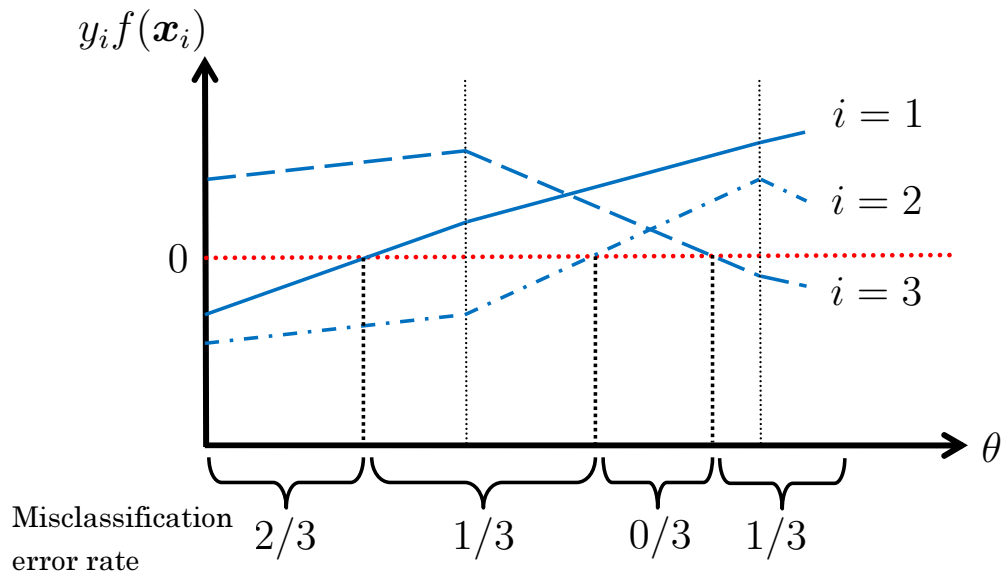
(例) 検証誤差 (0-1 loss) 追跡

- 検証用のデータ点集合 $\mathcal{V}$ への誤差

$$\sum_{i \in \mathcal{V}} I(y_i f(\boldsymbol{x}_i) \leq 0),$$

  I: indicator function

- $y_i f(\boldsymbol{x}_i)$, $i \in \mathcal{V}$ も θ に関して区分線形に変化
- 検証誤差は θ に関して区分定数



実験

- 重みパス追跡の応用例
  1. 経済データによる重み付き時系列オンライン学習
  2. 共変量シフト下での適応重要度重みによる学習
  3. 不均一分散モデリング
  4. ランキング学習
  5. Transductive learning
- 共通の実験設定
  - RBF カーネル $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{p}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$, p は $\boldsymbol{x}$ の次元数
  - 計算時間の比較は SMO による $\boldsymbol{c}^{(\mathrm{old})}$ から $\boldsymbol{c}^{(\mathrm{new})}$ へ更新 (hot start)

経済データによる重み付き時系列オンライン学習

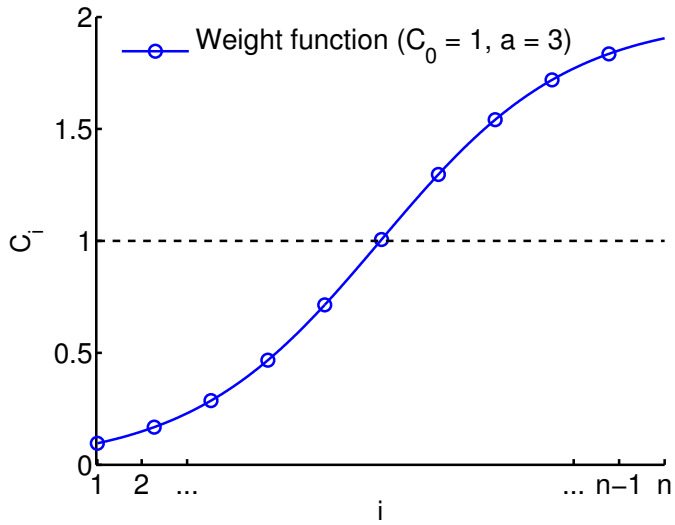
- 重み付き SVM による株価予想

  (Cao and Tay, IEEE Trans. on NN 03)

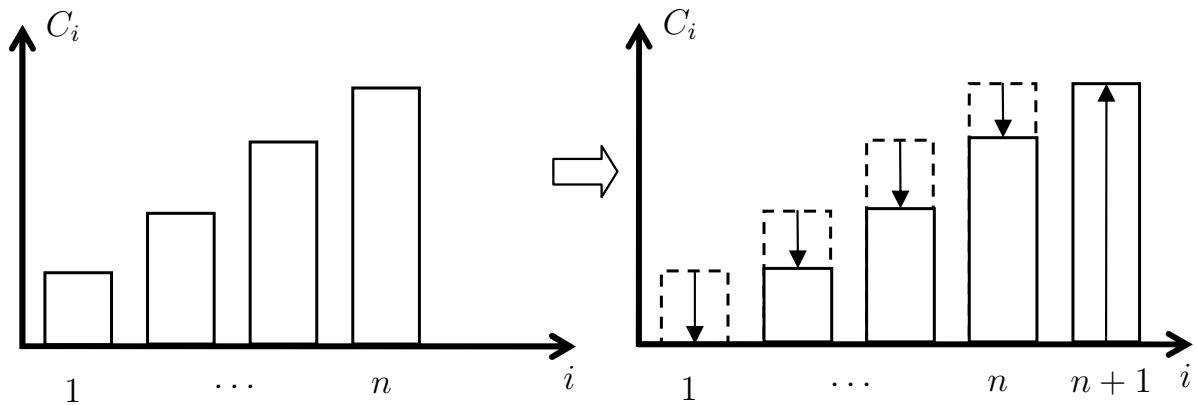
  - 最近のデータ点ほど重視するような重みづけ

$$C_i = C_0 \frac{2}{1 + \exp(a - 2a \times \frac{i}{n})}, \ i = 1, \ldots, n.$$

  (i 大きい方が最近とする)

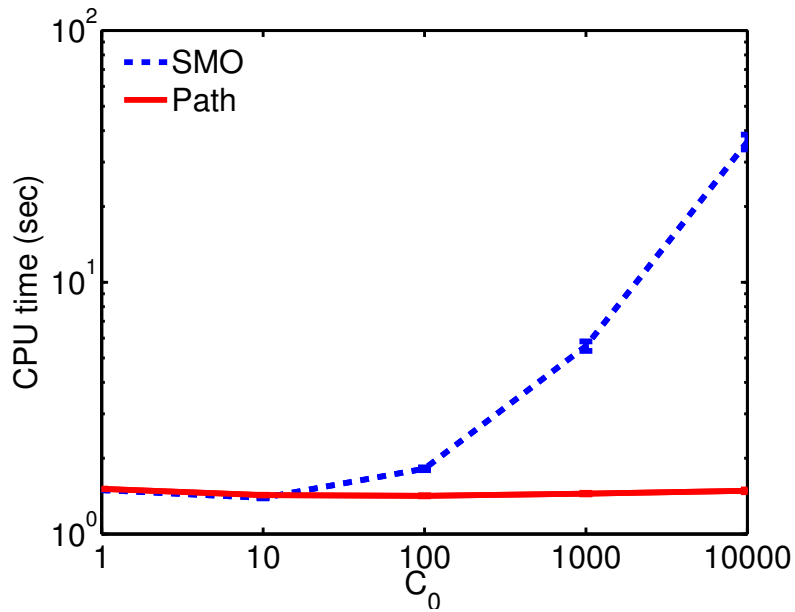


- 新しいデータ点が来たら重みを更新



実験

- 5 日後の NASDAQ 総合指数の増減を過去の値から予測
- $n = 2515$, $p = 5$,
- 5 個のデータ点が新規に得られる (最古の 5 個は同時に消去)



共変量シフト下での適応重要度重みによる学習

- 訓練とテストで異なる入力分布: $p_{train}(\boldsymbol{x})$, $p_{test}(\boldsymbol{x})$
- 密度比による重み付き SVM

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_i\}_{i=1}^n} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C_0 \sum_{i=1}^n \frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)} \xi_i \\ \text{s.t.} \quad & y_i\{\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b\} \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, n. \end{aligned}$$

- 平坦下した重要度のほうがバリアンス低 (Shimodaira, JSPI 01)

  $\Rightarrow$ モデル選択の必要性

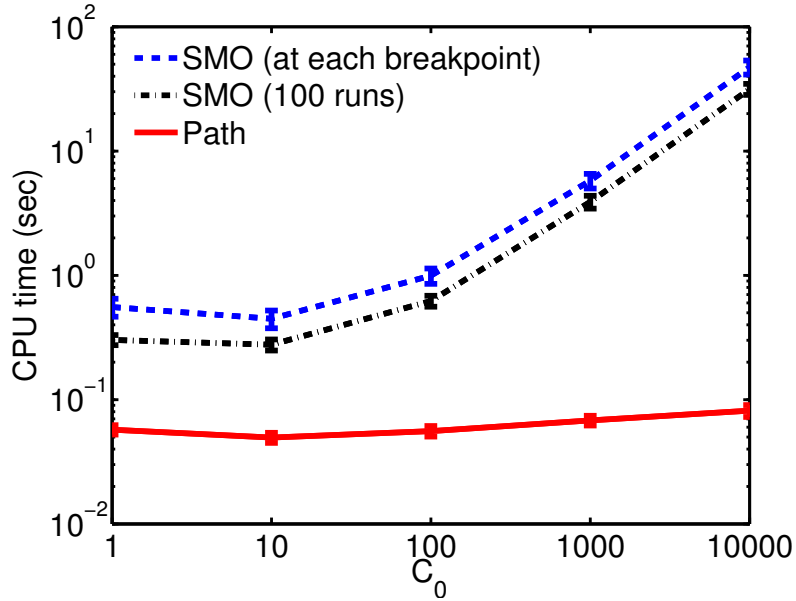
$$\begin{aligned} C_i^{(\mathrm{old})} &= C_0, \\ C_i^{(\mathrm{new})} &= \frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)} C_0 \end{aligned}$$

  の間をパス追跡してモデル選択

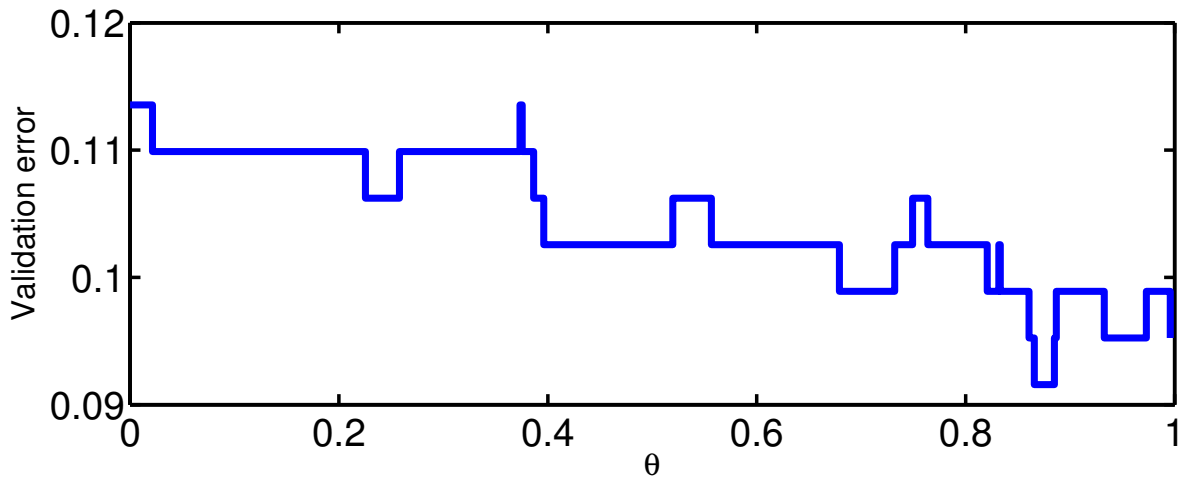
  (密度比は推定済みとする)

BCI データによる実験

- ブレイン・コンピュータインターフェース
  - 脳波から “左” と “右” のコマンドを判別
  - 訓練とテストで異なる心理状態
- 実験設定
  - $n = 500 \sim 1000,\ p = 4$
  - SMO は各ブレイクポイントでの実行と，一様に 100 点とった実行
  - 0-1 損失で検証誤差追跡



CPU 時間



検証誤差の推移

不均一分散モデリング

- 不均一分散下での回帰問題

$$y_i = f(\boldsymbol{x}_i) + \sigma(\boldsymbol{x}_i) \cdot \epsilon_i,\ E(\epsilon_i) = 0,\ i = 1, \ldots, n,\ \{\epsilon_i\}_{i=1}^n \text{i.i.d.},$$

誤差項の分散 $\sigma : \mathcal{X} \to \mathbb{R}^+$ が $\boldsymbol{x}$ に依存

  - 重み付き SV Regression (SVR)

$$\begin{aligned} \min_{\boldsymbol{w}, b, \{\xi_i, \xi_i^*\}_{i=1}^n} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^n C_i(\xi_i + \xi_i^*), \\ \text{s.t.} \quad & y_i - f(\boldsymbol{x}_i) \leq \varepsilon + \xi_i, \\ & f(\boldsymbol{x}_i) - y_i \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0,\ i = 1, \ldots, n, \end{aligned}$$

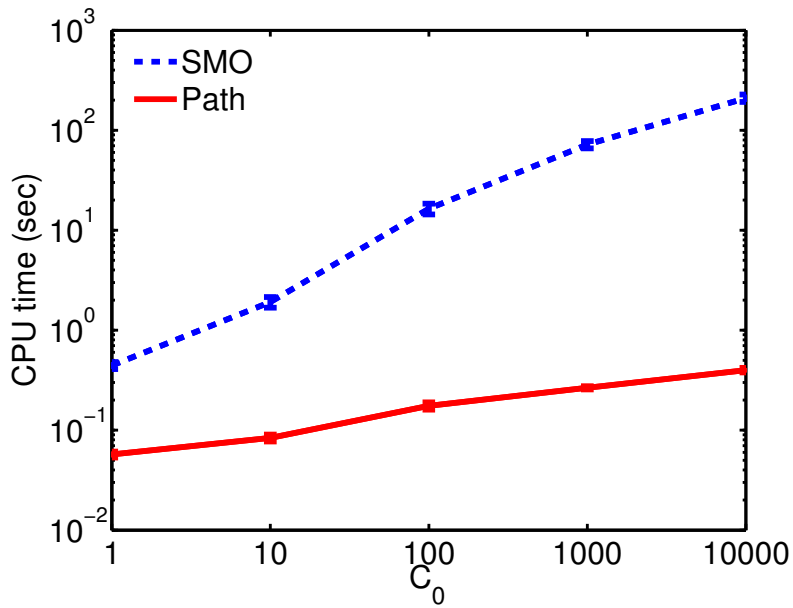
  - 分散を推定して各データ点に重み付け

$$C_i = \frac{\hat{\sigma}}{\hat{\sigma}_i} C_0$$

$\hat{\sigma}$ は共通の分散の推定値，$\hat{\sigma}_i$ は y_i の分散の推定値

実験

- アルゴリズム
  1. $C_i = C_0, \forall i$ で学習
  2. $\hat{\sigma}_i = |y_i - f(\boldsymbol{x}_i)|$, $\hat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \hat{\sigma}_i^2}$
     として $C_i = \frac{\hat{\sigma}}{\hat{\sigma}_i} C_0$ で重み更新
     重みの更新が小さくなるまで繰り返し
- Boston housing data set
  $n = 404$, $p = 13$



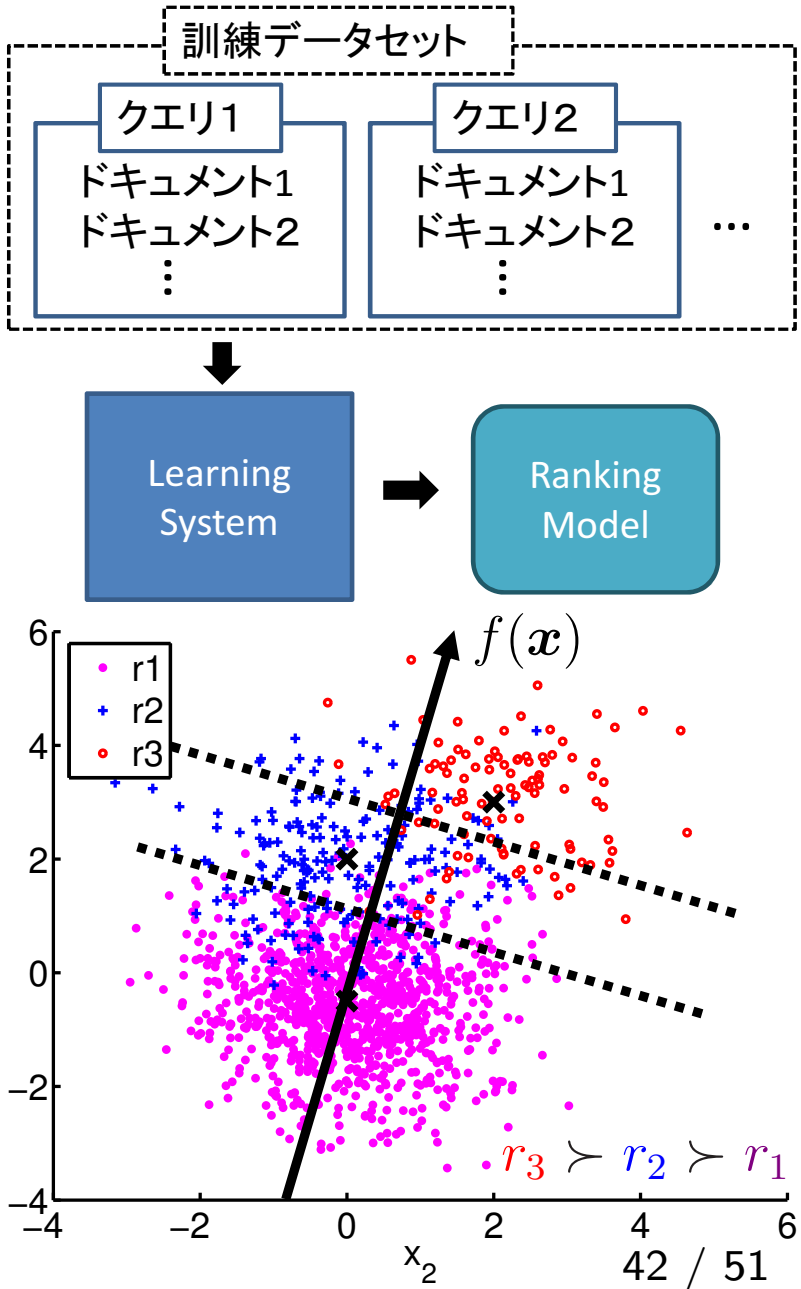
ランキング学習

- クエリに対して関連度の高いデータ点見つけたい
- 訓練データ $\boldsymbol{x}_i \in \mathcal{X}$, $y_i \in \{r_1, \ldots, r_q\}$
  - r_i は関連度を表わす
  - 関連度の高さを $\succ$ で表現するとして

$$y_i \succ y_j \Leftrightarrow f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$$

となるように f を推定

- 重み付けの例:
  関連度や順位の高いデータ点を重視
  (Cao et al, SIGIR 06; Xu et al, ECML 06)



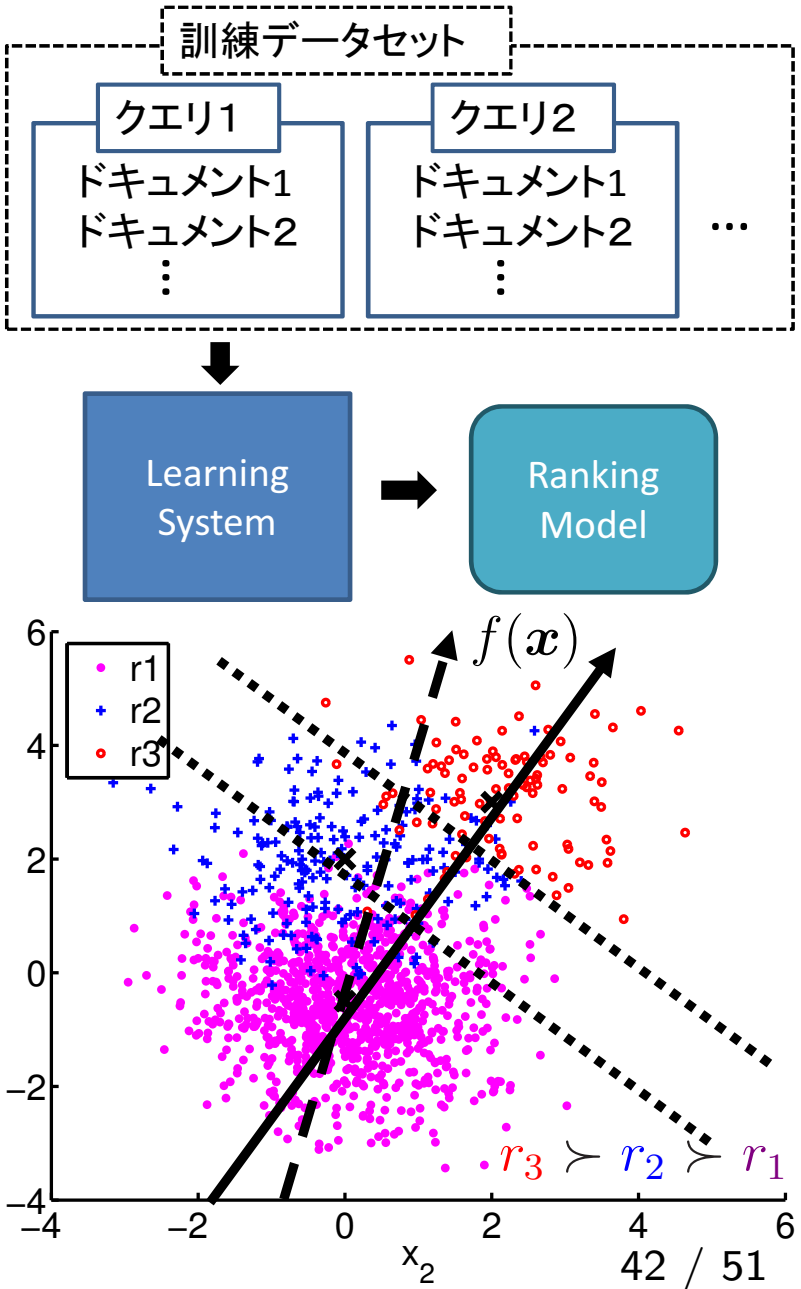
ランキング学習

- クエリに対して関連度の高いデータ点見つけたい
- 訓練データ $\boldsymbol{x}_i \in \mathcal{X}$, $y_i \in \{r_1, \ldots, r_q\}$
  - r_i は関連度を表わす
  - 関連度の高さを $\succ$ で表現するとして

$$y_i \succ y_j \Leftrightarrow f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$$

となるように f を推定

- 重み付けの例:
  関連度や順位の高いデータ点を重視
  (Cao et al, SIGIR 06; Xu et al, ECML 06)



ランキング SVM

- pair-wise approach

  $\mathcal{P} = \{(i,j) \mid y_i \succ y_j\}$ に属する (i,j) に対して

$$\begin{aligned} y_i \succ y_j \quad &\Leftrightarrow \quad f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \\ &\Leftrightarrow \quad \boldsymbol{w}^\top \{\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\} > 0 \end{aligned}$$

- ランキング SVM の主問題

$$\begin{aligned} \min_{\boldsymbol{w}, \{\xi_{ij}\}_{(i,j)\in\mathcal{P}}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_{(i,j)\in\mathcal{P}} \xi_{ij} \\ \text{s.t.} \quad & \boldsymbol{w}^\top \{\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\} \geq 1 - \xi_{ij}, \ (i,j) \in \mathcal{P}. \end{aligned}$$

ランキング SVM

- pair-wise approach
  $\mathcal{P} = \{(i,j) \mid y_i \succ y_j\}$ に属する (i,j) に対して

$$\begin{aligned} y_i \succ y_j \quad &\Leftrightarrow \quad f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \\ &\Leftrightarrow \quad \boldsymbol{w}^\top \{\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\} > 0 \end{aligned}$$

- 重み付きランキング SVM の主問題

$$\begin{aligned} \min_{\boldsymbol{w}, \{\xi_{ij}\}_{(i,j)\in\mathcal{P}}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{(i,j)\in\mathcal{P}} C_{ij}\xi_{ij} \\ \text{s.t.} \quad & \boldsymbol{w}^\top \{\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\} \geq 1 - \xi_{ij}, \ (i,j) \in \mathcal{P}. \end{aligned}$$

実験

- OHSUMED data from Microsoft Research

  $n = 16140$(全体), $p = 45$
- 関連度の高いデータを重視

$$
\begin{aligned}
C_{ij}^{(\mathrm{old})} &= C_0,\ (i,j) \in \mathcal{P}, \\
C_{ij}^{(\mathrm{new})} &= (2^{y_i} - 2^{y_j}) C_0,\ (i,j) \in \mathcal{P},
\end{aligned}
$$

  $C_{ij}^{(\mathrm{old})}$ と $C_{ij}^{(\mathrm{new})}$ の間で最適な重みはどこか？
- NDCG 最大化による最適重み選択

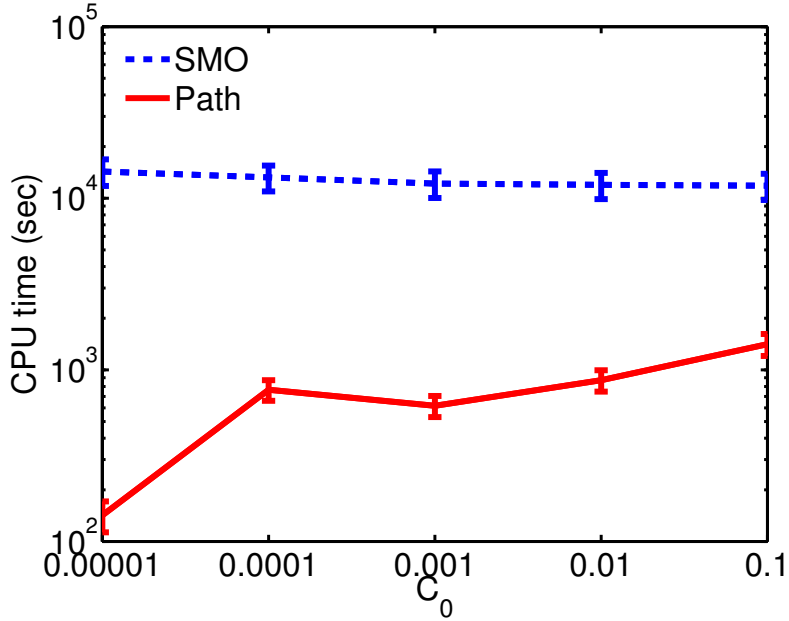
$$
\mathrm{NDCG@}k = Z \sum_{j=1}^{k} \begin{cases} 2^{y_{q(j)}} - 1, & j = 1, \\ \frac{2^{y_{q(j)}} - 1}{\log(j)}, & j > 1, \end{cases}
$$

  上位 k 位のランキングが正しいほど大きな値

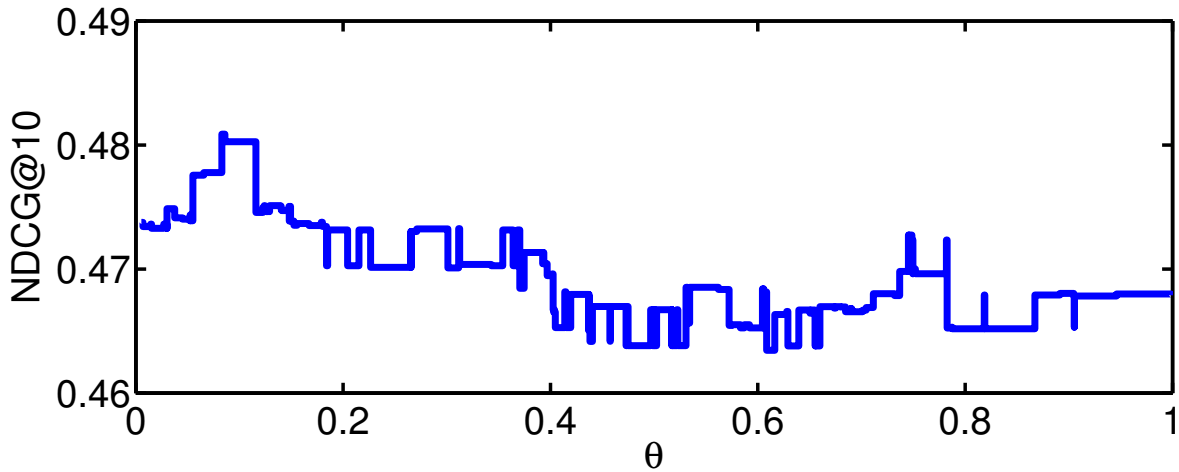
  ($q(j)$: j 番目にランク付けされたデータ, Z: 正規化定数)

実験

- SMO は 10 回実行
- NDCG の厳密な追跡が可能
  - ランキングは $f(\boldsymbol{x}_i)$ のソートによって決まる
  - 順位の入れ替わりの監視が可能



CPU 時間



検証データの NDCG 推移

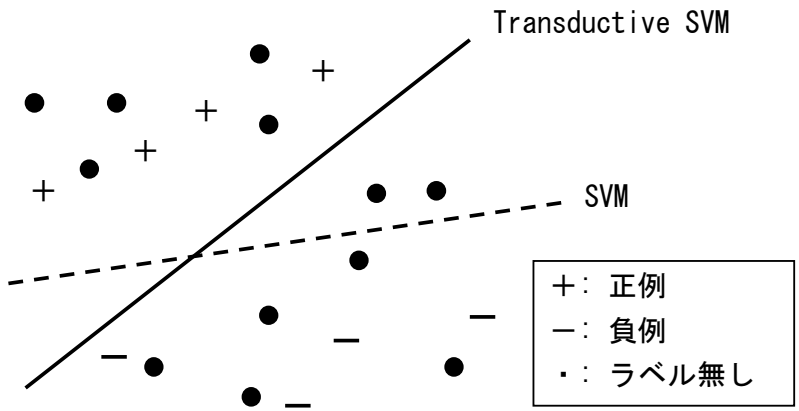
Transductive learning

- Given:

  ラベル付きデータ n 個 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$,

  ラベル無しデータ k 個 $\{(\boldsymbol{x}_i^*, y_i^*)\}_{i=1}^k$

  - ラベル無しデータのラベル得たい
  - Transductive SVM (Joachims, ICML 99)
    マージン最大化するラベルの与え方求める



$$\min_{\substack{\{y_i^*, \xi_i^*\}_{i=1}^k, \\ \boldsymbol{w}, b, \{\xi_i\}_{i=1}^n}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^n \xi_i + C^*\sum_{j=1}^k \xi_j^*$$

$$\text{s.t.} \quad \begin{aligned} & y_i\{\boldsymbol{w}^\top \boldsymbol{\Phi}(\boldsymbol{x}_i) + b\} \geq 1 - \xi_i, && i = 1, \ldots, n, \\ & y_j^*\{\boldsymbol{w}^\top \boldsymbol{\Phi}(\boldsymbol{x}_j) + b\} \geq 1 - \xi_j^*, && j = 1, \ldots, k, \\ & \xi_i \geq 0, && i = 1, \ldots, n, \\ & \xi_j^* \geq 0, && j = 1, \ldots, k, \end{aligned}$$

TSVM アルゴリズム

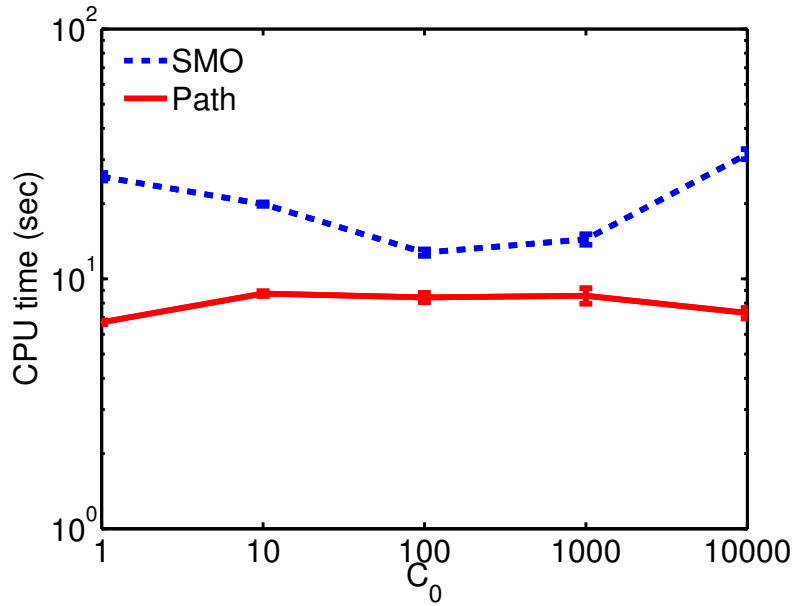
- (Joachims, ICML 99) より
  1. ラベル付きデータのみで学習
     $f(\boldsymbol{x}_i^*)$ の大きい方から k^+ 個を $y_i^* = +1$ とする
  2. SVM を学習.

     $$y_m^* * y_\ell^* < 0,\ \xi_m^* > 0,\ \xi_\ell^* > 0,\ \xi_m^* + \xi_\ell^* > 2$$

     であるような m と ℓ のラベルを入れ替える
     (そのような m と ℓ が存在する間 step2 を繰り返し)
  3. C^* を増やす
     C^* が事前に設定した値より大きくなったら終了,
     そうでなければ step 2 へ
- 赤字部分がパス追跡で行える
  - ラベルの入れ替えは一旦 y_i のデータを削除した後,
    $-y_i$ のデータ点を追加
    (Incremental Decremental だが C_i を動かすことでも可能)

実験

- Spam data set $n = 4601$, $p = 57$
- 10%のデータ点をランダムに選んでラベル付きデータとし，残りはラベル無しとして扱う



その他の話題

- 関数マージンによる重み付け

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_i\}_{i=1}^n} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^n \xi_i, \\ \text{s.t.} \quad & y_i f(\boldsymbol{x}_i) \geq \delta_i - \xi_i,\ \xi_i \geq 0,\ i = 1, \ldots, n, \end{aligned}$$

- Weighted Lasso

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \sum_{j=1}^p \boldsymbol{x}_j \beta_j\|_2^2 + \lambda \sum_{j=1}^p w_j |\beta_j|,$$

Instance-weighted SVM solution-path まとめ

- Multi-parametric アプローチによる重み付き SVM のパス
- 重み付きの形で書ける問題の多くに適用可能

まとめ

- Multi-parametric アプローチによる機械学習の様々なタスク
  - このアプローチはこれまであまり利用されてこなかった
- 複数のパラメータを同時に動かしたい，
  また，その挙動を厳密・詳細に知りたい場合に特に有効