情報論的学習理論テクニカルレポート 2009
Technical Report on Information-Based Induction Sciences 2009 (IBIS2009)

重み付きカーネルマシンの多次元パス追跡法に関する一考察

A Study on Multi-dimensional Path Following for Weighted Kernel Machines

烏山 昌幸* 原田 尚幸† 竹内 一郎‡
Masayuki Karasuyama Naoyuki Harada Ichiro Takeuchi

Abstract: We propose multi-dimensional path following for weighted kernel machines. In some situations of data modeling, each data point has a weight which represents the importance or confidence of the data point. We can easily implement such weighting schema in kernel machines by introducing multiple regularization parameters. In this paper, we derive the piece-wise linear path of these parameters which extends the idea of well known regularization path. Conventional algorithm only deals with the change of single regularization parameter. On the other hand, our approach can handle the change of multiple parameters simultaneously. Experimental results show that proposed algorithm can efficiently update optimal parameters. Our approach is especially beneficial for adaptive learning or online learning of weighted model.

Keywords: weighted kernel machines, support vector machines, path following.

1 まえがき

パターン認識において訓練データからモデルを推定する場合に，各々のデータ点ごと（あるいは，いつかのデータ点のグループごと）に異なる重みを与えたい場合がある．重みはデータの分散を考慮して適応的に定められたり [1]，データの性質に関する事前知識などから決められる．後者の具体例としては，経済データにおいて最近のデータ点ほど重要視されるようにしたり [2, 3]，スパムメールのフィルタリングにおいてメールのカテゴリごとに異なる重みを与えたり [4] といった事例がある．

n 個の訓練データ点を $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ とする．ただし，$\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ であり，$y_i \in \{1, -1\}$ なら 2 値分類問題，$y_i \in \mathbb{R}$ なら回帰問題として考えることができる．サポートベクトルマシン (SVM)[5] に代表されるカーネル法の多くでは特徴写像 $\Phi : \mathcal{X} \to \mathcal{F}$ による線形モデル $f(\boldsymbol{x}) = \boldsymbol{w}^T\Phi(\boldsymbol{x}) + b$ の推定を以下のような最適化問題に帰着させる:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^n L(y_i, f(\boldsymbol{x}_i)).$$

ただし，$L(y, f(\boldsymbol{x}))$ は訓練誤差を測る損失関数，$C > 0$ は正則化パラメータとする．ここで，n 個の正則化パラメータ $C_i > 0, i = 1, \cdots, n$, を考えることでデータ点ごとに個別の重みを課すことができる:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^n C_i L(y_i, f(\boldsymbol{x}_i)).$$

例えばデータ点 $i = 1, \cdots, n$, が時系列順に並んでいるとして，最近のデータをより重視させたいのであれば $C_i < C_{i+1}$ となるように設定すればよい．

これらの最適化問題を解く際には C_i はあらかじめ固定された値に設定しておくことが多い．しかし，C_i を様々な方法で変化させ，それぞれの設定に対して最適な $f(\boldsymbol{x})$ を得たい場合がある．例えば，

- どのデータ点をどの程度重視するかを適応的に決めたい場合

*名古屋工業大学大学院工学研究科, 466-8555 名古屋市昭和区御器所町, tel. 052-735-7524, e-mail krsym@ics.nitech.ac.jp, Nagoya Institute of Technology, Gokisocho, Showa, Nagoya, 466-8555 Japan
†名古屋工業大学大学院工学研究科, 466-8555 名古屋市昭和区御器所町, tel. 052-735-7524, e-mail harada@goat.ics.nitech.ac.jp, Nagoya Institute of Technology, Gokisocho, Showa, Nagoya, 466-8555 Japan
‡名古屋工業大学大学院工学研究科, 466-8555 名古屋市昭和区御器所町, tel. 052-735-7524, e-mail takeuchi.ichiro@nitech.ac.jp, Nagoya Institute of Technology, Gokisocho, Showa, Nagoya, 466-8555 Japan

- 新しくデータ点が追加され，データセット全体の重要度の振り分けを再調整したい場合（古いデータ点の重要度を下げる等）

といった状況がある．

正則化の度合いを変化させたい場合に，単一の正則化パラメータ C を用いた SVM においては正則化パス [6] と呼ばれる方法で C の変化に対する最適な $f(\boldsymbol{x})$ の変化を求めることができる．これは最適化においてパス追跡 (Path Following) やパラメトリック計画法 (Parametic Programming) と呼ばれる手法 [7] を利用しており，正則化係数の連続的な変化に対する $f(\boldsymbol{x})$ の変化を解析的に計算することができる．そのため最適な C を探すモデル選択時などにおいて有効であった．一方で個別の正則化パラメータを導入した場合，上で示したような状況では n 個の C_i が同時に変化することを考えたい．そこで本稿では，多変数に対するパス追跡 [8] を導入することで各データ点ごとの重みを動かした時のモデルの変化を効率的に追跡する方法を提案する．

2 重み付き SVM のパス追跡

個別正則化パラメータのパス追跡として SVM による 2 値分類問題 ($y_i \in \{1, -1\}$) を例に解説する．同様の理論はサポートベクトル回帰（SVR）等の他のカーネルマシンにおいても適用が可能である．

2.1 SVM と KKT 条件

SVM では損失関数 L として以下のヒンジ損失関数を使う:

$$L(y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}.$$

個別正則化パラメータを使った場合，以下のような最適化問題に帰着できる:

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

このように訓練誤差に個別の重みを課した SVM は Weighted SVM[9] や Fuzzy SVM[10] と呼ばれることもある．ラグランジュの未定乗数 $\alpha_i, \rho_i \geq 0, i = 1, \cdots, n$, を導入するとラグランジュ関数は以下のようになる:

$$\begin{aligned} L \quad = \quad & \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^n C_i \xi_i \\ & - \sum_{i=1}^n \alpha_i \{y_i f(\boldsymbol{x}_i) - 1 + \xi_i\} - \sum_{i=1}^n \rho_i \xi_i. \quad (1) \end{aligned}$$

主変数 $\boldsymbol{w}, b, \xi_i$ に関して微分して 0 とおくと，

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{0} \quad &\Leftrightarrow \quad \boldsymbol{w} = \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{x}_i), \\ \frac{\partial L}{\partial b} = 0 \quad &\Leftrightarrow \quad \sum_{i=1}^n \alpha_i y_i = 0, \\ \frac{\partial L}{\partial \xi_i} = 0 \quad &\Leftrightarrow \quad \alpha_i = C_i - \rho_i,\ i = 1, \cdots, n, \end{aligned}$$

が得られる．これらをラグランジュ関数 (1) に代入することで以下の双対問題が得られる:

$$\begin{aligned} \max_{\{\alpha_i\}_{i=1}^n} \quad & -\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C_i, \end{aligned}$$

ただし，$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ はカーネル関数とする．$f(\boldsymbol{x})$ の双対表現は以下のようになる:

$$f(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b.$$

バイアス b は $y_i f(\boldsymbol{x}_i) - 1 = 0$ となるような i を使って求める．最適解において $y_i f(\boldsymbol{x}_i) - 1$ の値は KKT 条件の相補性条件

$$\begin{aligned} \alpha_i \{y_i f(\boldsymbol{x}_i) - 1 + \xi_i\} = 0, \qquad & i = 1, \cdots, n, \\ \xi_i (\alpha_i - C_i) = 0, \qquad & i = 1, \cdots, n, \end{aligned}$$

と主問題，双対問題の不等式制約を考慮することにより

$$\begin{aligned} y_i f(\boldsymbol{x}_i) - 1 > 0 \quad &\Rightarrow \quad \xi_i = 0, \qquad \alpha_i = 0, \\ y_i f(\boldsymbol{x}_i) - 1 = 0 \quad &\Rightarrow \quad \xi_i = 0, \quad 0 \leq \alpha_i \leq C_i, \\ y_i f(\boldsymbol{x}_i) - 1 < 0 \quad &\Rightarrow \quad \xi_i > 0, \qquad \alpha_i = C_i, \end{aligned}$$

といった関係を持つことがわかる．この関係を使って以下の集合を定義しておく:

$$\mathcal{O} = \{i : y_i f(\boldsymbol{x}_i) > 1, \alpha_i = 0\}, \quad (2a)$$

$$\mathcal{M} = \{i : y_i f(\boldsymbol{x}_i) = 1, 0 \leq \alpha_i \leq C_i\}, \quad (2b)$$

$$\mathcal{I} = \{i : y_i f(\boldsymbol{x}_i) < 1, \alpha_i = C_i\}. \quad (2c)$$

以降では，部分行列や部分ベクトルを表わすため，集合を下付きの添え字として用いることとする．例えば，ベクトル $\boldsymbol{v} = [v_1, \cdots, v_n]^\top$ の要素のうち集合 $\mathcal{I} = \{I_1, \cdots, I_{|\mathcal{I}|}\}$ に対応する部分ベクトル $[v_{I_1}, \cdots, v_{I_{|\mathcal{I}|}}]^\top$ は $\boldsymbol{v}_\mathcal{I}$ と表わす．同様に，部分行列については $n \times n$ 行列 $\boldsymbol{M}$ に対して $\boldsymbol{M}_{\mathcal{M},\mathcal{O}}$ とした場合，行が $\mathcal{M}$，列が $\mathcal{O}$ に対応する部分行列とする．

2.2 多次元パス追跡の導入

正則化パス [6] では単一の C を変化させた時のパラメータ $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^\top, b$ の変化を追跡するが，ここでは個別正則化パラメータのベクトル

$$\boldsymbol{c} = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix},$$

を $\boldsymbol{c}^{(\mathrm{old})}$ から $\boldsymbol{c}^{(\mathrm{new})}$ へと変化させたい場合を考える．アルゴリズムは [6] と同様に集合 $\mathcal{O}, \mathcal{M}, \mathcal{I}$ を監視して最適性を満たしながらパラメータを変化させる．

今，$\boldsymbol{c}$ を $\boldsymbol{\Delta c} = [\Delta C_1, \cdots, \Delta C_n]^\top$ だけ変化させた時のパラメータ $\boldsymbol{\alpha}, b$ の変化を考える．Δ を付けた変数は各変数の変化量を表わすものとする．$\alpha_i = C_i, i \in \mathcal{I}, \alpha_i = 0, i \in \mathcal{O}$ であることに注意すると $y_i f(\boldsymbol{x}_i)$ は以下のように書くことができる:

$$y_i f(\boldsymbol{x}_i) = \sum_{j \in \mathcal{M}} Q_{ij}\alpha_j + \sum_{j \in \mathcal{I}} Q_{ij} C_j + y_i b,$$

ただし，$Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ とした．$\boldsymbol{c}$ が $\boldsymbol{\Delta c}$ だけ変化したとしても，(2b) よりマージン上の点では $y_i \Delta f(\boldsymbol{x}_i) = 0$ でなければならない:

$$\sum_{j \in \mathcal{M}} Q_{ij}\Delta\alpha_j + \sum_{j \in \mathcal{I}} Q_{ij}\Delta C_j + y_i \Delta b = 0, \ i \in \mathcal{M}. \quad (3)$$

また双対問題の和制約を考慮すると，

$$\sum_{j \in \mathcal{M}} y_j \Delta\alpha_j + \sum_{j \in \mathcal{I}} y_j \Delta C_j = 0, \quad (4)$$

が得られる．式 (3),(4) の連立方程式を行列で表記すると以下のようにまとめられる:

$$\boldsymbol{M} \begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{y}_{\mathcal{I}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{I}} \end{bmatrix} \boldsymbol{\Delta c}_{\mathcal{I}} = \boldsymbol{0}, \quad (5)$$

ただし，

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

とした．(5) を解くと

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_{\mathcal{I}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{I}} \end{bmatrix} \boldsymbol{\Delta c}_{\mathcal{I}}, \quad (6)$$

となる．また，

$$\boldsymbol{\Delta\alpha}_{\mathcal{O}} = \boldsymbol{0}, \quad (7)$$

$$\boldsymbol{\Delta\alpha}_{\mathcal{I}} = \boldsymbol{\Delta c}_{\mathcal{I}}, \quad (8)$$

であるため $\boldsymbol{\Delta c}$ が決まれば全てのパラメータの変化が計算できることになる．ただし，$\mathcal{I}, \mathcal{M}, \mathcal{O}$ の定義 (2a)-(2c) より，以下の不等式条件が満たされていなければならない:

$$y_i\{f(\boldsymbol{x}_i) + \Delta f(\boldsymbol{x}_i)\} < 1, \quad i \in \mathcal{I}, \quad (9a)$$

$$0 \le \alpha_i + \Delta\alpha_i \le C_i + \Delta C_i, \quad i \in \mathcal{M}, \quad (9b)$$

$$y_i\{f(\boldsymbol{x}_i) + \Delta f(\boldsymbol{x}_i)\} > 1, \quad i \in \mathcal{O}. \quad (9c)$$

本稿ではこの不等式が定義する領域を *Critical Region*[8] と呼ぶこととする．Critical Region の中ではパラメータ $\boldsymbol{\alpha}, b$ の変化は式 (6)-(8) によって求めることができる．ここで $\boldsymbol{\Delta c}$ をステップ幅 $\eta \ge 0$ を使って

$$\boldsymbol{\Delta c} = \eta(\boldsymbol{c}^{(\mathrm{new})} - \boldsymbol{c}^{(\mathrm{old})}), \quad (10)$$

として定めるとする．これは，$\boldsymbol{c}$ を $\boldsymbol{c}^{(\mathrm{old})}$ から $\boldsymbol{c}^{(\mathrm{new})}$ へ直線的に動かすことに対応している．ただし，η を大きくしていくと，ある値で Critical Region の境界に達する．式 (10) のように η を定めたことで，境界に達するステップ幅は容易に計算できる．具体的には，式 (10) を式 (6) に代入して整理すると，

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = \eta\boldsymbol{\phi}, \quad (11)$$

と書くことができる．ただし，

$$\boldsymbol{\phi} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_{\mathcal{I}}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{I}} \end{bmatrix} (\boldsymbol{c}_{\mathcal{I}}^{(\mathrm{new})} - \boldsymbol{c}_{\mathcal{I}}^{(\mathrm{old})}), \quad (12)$$

とした．また，不等式中の $\Delta f(\boldsymbol{x}_i)$ は

$$\boldsymbol{y} \odot \boldsymbol{\Delta f} = \begin{bmatrix} \boldsymbol{y} & \boldsymbol{Q}_{:,\mathcal{M}} \end{bmatrix} \begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} + \boldsymbol{Q}_{:,\mathcal{I}} \boldsymbol{\Delta c}_{\mathcal{I}} \quad (13)$$

$$= \eta\, \boldsymbol{\psi}, \quad (14)$$

を使って計算できる．ただし，$\odot$ を行列の要素ごとの積，$\boldsymbol{\psi}$ を

$$\boldsymbol{\psi} = [\ \boldsymbol{y}\ \boldsymbol{Q}_{:,\mathcal{M}}\ ]\ \boldsymbol{\phi} + \boldsymbol{Q}_{:,\mathcal{I}}(\boldsymbol{c}_{\mathcal{I}}^{(\mathrm{new})} - \boldsymbol{c}_{\mathcal{I}}^{(\mathrm{old})}),$$

とした．式 (11),(14) を使うと不等式 (9) を満たしたままどこまで η を大きくできるかが評価できる．

式 (6),(13) を考慮すると，式 (9) の不等式は $\boldsymbol{\Delta c}$ に関して一次であることがわかる．そのため，各不等式は $\boldsymbol{c}$ の空間において半空間を定義しており，その共通部分として定義される Critical Region は凸多面体となる．各不等式について境界値に達する η の値（η をそれ以上大きくすると半空間から出てしまうような値）の集合を $\mathcal{H}$ と定義する（η をどれだけ大きくしても境界値に達し

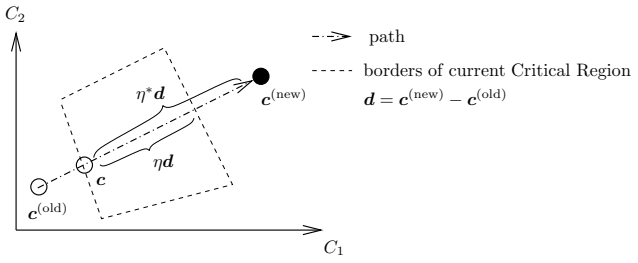


図 1: $n = 2$ での概念図．$\boldsymbol{c}$ は $\boldsymbol{c}^{(\mathrm{old})}$ から $\boldsymbol{c}^{(\mathrm{new})}$ へと動く途中であるとする．$\boldsymbol{c}$ から $\eta^*\boldsymbol{d}$ だけ進むと $\boldsymbol{c}^{(\mathrm{new})}$ に到達するが，$\eta\boldsymbol{d}$ だけ進んだ地点で不等式の定義する境界（破線）にぶつかるため一旦そこまで止まる．$\mathcal{O}, \mathcal{M}, \mathcal{I}$ を更新することで新たな Critical Region が構成されるので同様の手順で $\boldsymbol{c}^{(\mathrm{new})}$ に向けて進めるだけ進む．

ない不等式もあるが，こういった場合は $\eta = \infty$ と考える）．$\boldsymbol{c} + \boldsymbol{\Delta c}$ が $\boldsymbol{c}^{(\mathrm{new})}$ となるような η を η^* とすると，η^* が $\mathcal{H}$ のどの要素よりも小さければ η^* による更新でアルゴリズムは終了する ($\boldsymbol{c}$ が $\boldsymbol{c}^{(\mathrm{new})}$ となる)．つまりステップ幅 η は

$$\min(\mathcal{H} \cup \{\eta^*\}),$$

によって定められる（図 1 に 2 次元での概念図を示した)．η^* 以外の η によってパラメータが更新された場合は，達した境界に応じて $\mathcal{O}, \mathcal{M}, \mathcal{I}$ の要素を移動させなければならない．このように Critical Region が切り替わる点を *breakpoint* と呼ぶ．$\mathcal{O}, \mathcal{M}, \mathcal{I}$ に変化が起こるため breakpoint では連立方程式 (12) の再計算が必要になるが，逆行列もしくはコレスキー分解のランク 1 更新 [11, 12] を利用することで再計算は $O(|\mathcal{M}|^2)$ で可能になる．式 (6)-(8) よりパラメータ $\boldsymbol{\alpha}, b$ の変化は全て $\boldsymbol{\Delta c}$ に対して線形であることがわかるが，連立方程式の更新によりパラメータの変化は breakpoint で折れ曲がる（区分線形性）．アルゴリズムの流れを Algorithm 1 に示した:

Algorithm 1 Multi-dimentional Path Following for Weighted SVM

1: **given** optimal $\boldsymbol{\alpha}, b$ for $\boldsymbol{c}^{(\mathrm{old})}$
2: initialize $\mathcal{M}, \mathcal{O}, \mathcal{I}$
3: calculate $\boldsymbol{M}^{-1}$
4: **while** $\boldsymbol{c} \neq \boldsymbol{c}^{(\mathrm{new})}$ **do**
5: calculate $\boldsymbol{\phi}, \boldsymbol{\psi}$ using $\boldsymbol{M}^{-1}$
6: calculate $\mathcal{H}$ using $\boldsymbol{\phi}, \boldsymbol{\psi}$
7: $\eta \leftarrow \min(\mathcal{H} \cup \{\eta^*\})$
8: update $\boldsymbol{\alpha}, b, \boldsymbol{c}$ using step length η
9: update $\mathcal{M}, \mathcal{O}, \mathcal{I}$
10: update $\boldsymbol{M}^{-1}$
11: **end while**

3 計算機実験

ここでは，忘却係数モデル [13, 14] のオンライン学習を使って提案法の有効性を検証する．忘却係数モデルでは，ある C を 1 つ決めてから各正則化パラメータを減衰率 $\lambda \in (0, 1)$ を使って定義する:

$$C_i = C\lambda^{n-i}, \qquad i = 1, \cdots, n.$$

これはデータ点 $i = 1, \cdots, n$ が時系列に並んでいる場合などに，最近のデータほど重視させる方法の 1 つである．学習済みの忘却係数モデルに対して新たなデータ点 $(\boldsymbol{x}_{n+1}, y_{n+1})$ が追加される時，以下のような更新を行うことにする:

- 最も古い学習データ点 $(\boldsymbol{x}_1, y_1)$ を学習データから除く ($C_1 = 0$ とする)．
- 新しいデータ点 $(\boldsymbol{x}_{n+1}, y_{n+1})$ を学習データに加え，$C_{n+1} = C$ とする．
- $i = 2, 3, \cdots, n$ に対して $C_i \leftarrow \lambda C_i$ となるように重みを減少させる．

以上を考慮すると多次元パス追跡での更新方向は以下のようになる:

$$\boldsymbol{c}^{(\mathrm{new})} - \boldsymbol{c}^{(\mathrm{old})} = \begin{bmatrix} 0 \\ C\lambda^{n-1} \\ \vdots \\ C\lambda^2 \\ C\lambda \\ C \end{bmatrix} - \begin{bmatrix} C\lambda^{n-1} \\ C\lambda^{n-2} \\ \vdots \\ C\lambda \\ C \\ 0 \end{bmatrix}.$$

今回は全ての実験で $\lambda = 0.99$ とした．

また，以下の 2 つの手法について計算時間の比較を行う:

逐次１次元パス追跡 提案法では全ての正則化パラメータを同時に動かすことができるが，比較として各 C_i を 1 つずつ動かしてモデルの更新を行う．ただし，ある C_i を減少させたい場合に，減少後の値 λC_i より α_i が小さいのであれば $\boldsymbol{\alpha}, b$ を更新する必要はない（更新しなくても最適性条件が満たされているため)．

SMO SVM の 2 次計画問題の解法はパラメータ α_i を 2 つずつ最適化する Sequential Minimal Optimization (SMO) [15] が広く使われている．今回は最適化する 2 点を maximum violating pair [16, 17] により選択し，KKT 条件を $\varepsilon = 10^{-6}$ 以下の精度で

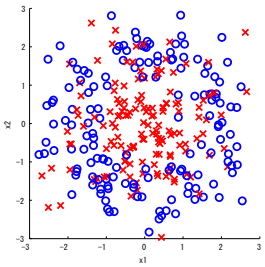


図 2: 2 つの正規分布を用いて人工データを生成した．原点を平均として時間 t に対し不変な分布と，原点を中心とする半径 2 の円周上を時間 t とともに平均ベクトル $\boldsymbol{\mu}$ が動くような分布とした．

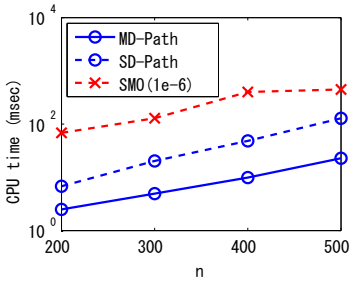


図 3: 人工データによる CPU 時間比較 (MD-Path:多次元パス追跡，SD-Path:逐次 1 次元パス追跡)

満たしたら停止することとした．また，$\boldsymbol{\alpha}$ の初期値としてデータ点が追加される前の解を利用する alpha seeding [18] を行った．

どの手法においても初期解 $\boldsymbol{\alpha}, b$ とカーネル関数 $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ はあらかじめ計算しておく．カーネルには以下の RBF カーネルを用いた:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}).$$

また，パス追跡においては初期逆行列 $\boldsymbol{M}^{-1}$ も用意された状況を想定して計算コストを選択した．

3.1 人工データ

人工データとして時刻 $t = 1, \cdots, n,$ に伴い平均が変化する正規分布から生成されたデータセットを用いる (図 2):

$$\begin{aligned} p(\boldsymbol{x}_t|y_t = +1) &= \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \\ p(\boldsymbol{x}_t|y_t = -1) &= \mathcal{N}(\boldsymbol{\mu}(t), 0.25\boldsymbol{I}), \\ \boldsymbol{\mu}(t) &= 2 \begin{bmatrix} \cos(2\pi t/n) \\ \sin(2\pi t/n) \end{bmatrix}. \end{aligned}$$

ただし，$\boldsymbol{I} \in \mathbb{R}^{2\times 2}$ は単位行列である．$n = 200, 300, 400, 500$ として，初期の学習を $t = 1, \cdots, n/2$ までのデータ点で行う．残りの $n/2$ 個を 1 つずつ追加してモデルを更新していき（この時，同時に最古のデータ点が削除されるため訓練データ数は常に $n/2$），1 つの追加あたりにかかった平均計算時間の比較を行う．$\boldsymbol{x}$ はそれぞれの次元を平均 0，分散 1 に正規化し，$C = 100$，RBF カーネルのパラメータ $\sigma^2 = 1$ とした．

図 3 は計算時間の測定結果である．多次元パス追跡が全ての n で他の手法と比べて高速であるのが確認できる．また，どの手法も対数プロット上で n に対してほぼ線形に時間が増加している．図 3 を表にしたものが表 1 であり，表 2 には多次元と逐次 1 次元パス追跡において breakpoint の数を記録した結果を示した．多次元パス追跡の breakpoint 数が全体的にやや少ないが，CPU 時間の差に対して breakpoint 数の差は比較的に小さい．

表 1: 人工データの平均実行時間 (msec)

method \ n	200	300	400	500
多次元	2.50	4.93	9.90	22.80
逐次 1 次元	6.80	20.27	48.30	128.24
SMO(1e-6)	68.50	128.73	400.75	447.64

表 2: 人工データの平均 breakpoint 数

method \ n	200	300	400	500
多次元	13.10	13.35	15.74	22.37
逐次 1 次元	16.02	19.07	22.36	38.20

表 3: 人工データの平均サーチ回数

method \ n	200	300	400	500
多次元	14.10	14.35	16.74	23.37
逐次 1 次元	39.76	60.13	81.81	132.04

表 4: 人工データの CPU 時間比とサーチ回数比

ratio \ n	200	300	400	500
計算時間比（一/多）	2.72	4.11	4.88	5.62
サーチ回数比（一/多）	2.82	4.19	4.89	5.65

ある方向を決めて一方向にパス追跡をする場合，各 breakpoint での計算量が同じであるとすると，全体の計算コストは breakpoint の数にほぼ比例する．ただし，逐次 1 次元パス追跡は途中で方向を変える（動かす C_i を変える）ことになるため，breakpoint だけでなく探索幅 η のサーチ回数を考慮する必要がある．以降では，探索幅を得るための計算とパラメータの更新を合わせて探索幅サーチと呼ぶこととする (Algorithm 1 の 5 行目から 8 行目に相当)．breakpoint では探索幅のサーチと連立方程式の更新の 2 つが主な計算コストとなるが，目的地に到達する最後の更新では探索幅サーチのみを行えば

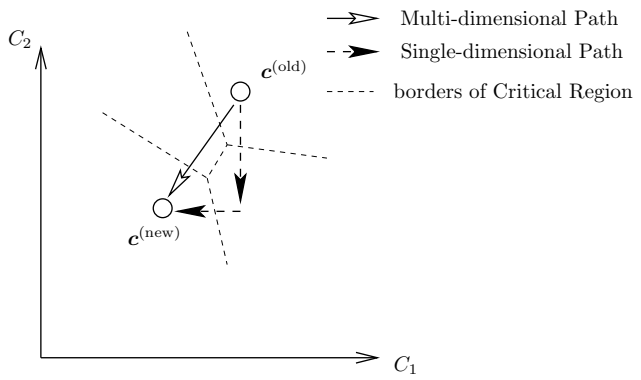


図 4: breakpoint 数とサーチ回数．図のパスにおいては多次元パス追跡では 2 回の breakpoint，3 回の探索幅サーチが発生するのに対して，逐次 1 次元パス追跡では 2 回の breakpoint，4 回のサーチが発生する．

連立方程式を更新する必要はない．この状況は多次元パス追跡では $\boldsymbol{c}^{(\mathrm{new})}$ に到達する最後の更新にだけ起こるが，逐次 1 次元パス追跡では 1 つの C_i に関するパス追跡が終わるたびに発生する（図 4 に例を示した）．多次元パス追跡，逐次 1 次元パス追跡についてそれぞれ，探索幅サーチの回数を $\mathcal{S}_m, \mathcal{S}_s$，breakpoint 数を $\mathcal{B}_m, \mathcal{B}_s$ として，逐次 1 次元パス追跡が p 回の 1 変数パス追跡を繰り返すとすると，

$$\mathcal{S}_m = \mathcal{B}_m + 1, \tag{15}$$

$$\mathcal{S}_s = \mathcal{B}_s + p, \tag{16}$$

といった関係がある．今回の実験では p は最大で $n/2+1$ である．実際にサーチ回数を記録した結果を表 3 に示す．breakpoint 数と比べて多次元と逐次 1 次元パス追跡で大きな差が出ているのが確認できる．表 4 は多次元と逐次 1 次元パス追跡で計算時間の比とサーチ回数の比を比較したものである．全ての n で 2 つの比に非常に近い値が確認されている．ここで，一回の探索幅サーチにかかる時間を T_S，連立方程式の更新にかかる時間を T_L とし，各繰り返しにおいて T_S, T_L は常に一定であると仮定すると初期化を除いた多次元パス追跡と逐次 1 次元パス追跡の計算時間の比はおおよそ

$$\mathcal{B}_m(T_S + T_L) + T_S : \mathcal{B}_s(T_S + T_L) + pT_S, \tag{17}$$

となるが，式 (17) がサーチ回数の比 $\mathcal{S}_m : \mathcal{S}_s$ に等しいとして整理すると

$$\frac{T_S}{T_S + T_L} = 1,$$

となるため，仮定のもとでは計算コスト比がサーチ回数比に近い場合は $T_L \ll T_S$ であると考えられる．

3.2 時系列実データによる実験

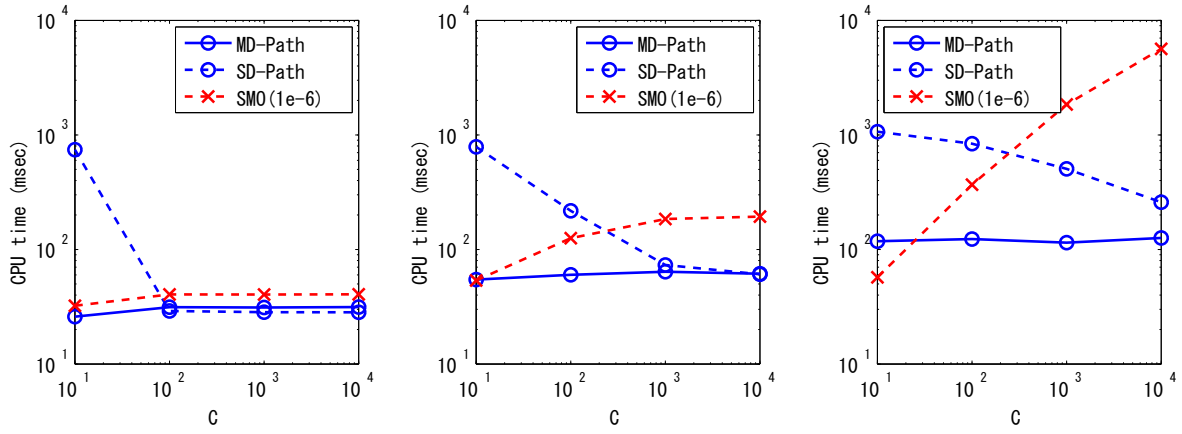
時系列実データによるオンライン学習での比較を行う．Fisher river データ[1]を用いて過去 7 日間の気温，降水量，川の水量 ($\boldsymbol{x} \in \mathbb{R}^{21}$) から翌日の水量の増減 ($y \in \{1, -1\}$) を予測する．$\boldsymbol{x}$ の各次元は平均 0 分散 1 に正規化し，初期データ数 $n = 365$ に対して計 365 個の新たなデータ点を 1 つずつ追加する．正則化パラメータとカーネルパラメータを $C \in \{10^1, 10^2, 10^3, 10^4\}, \sigma^2 \in \{10^{-1}, 10^0, 10^1\}$ の中でそれぞれ値を設定して結果を比較する．

図 5 は各手法で 1 回の追加にかかった平均時間の対数プロットである．多次元と逐次 1 次元パス追跡を比較すると，C が小さく，σ が大きい時に多次元パス追跡の方が速い．C が大きくなるとある程度までは逐次 1 次元パスの計算時間が減り，両者の違いは小さくなる．SMO との関係については C が大きいほどパス追跡が有利であった，また σ は大きいほど SMO はパス追跡と比較して遅い．SMO と C の関係については C が大きい時に計算コストが大きくなることが知られている [17].

表 5-7 には breakpoint 数を示した．小さい σ では breakpoint 数にほとんど差はないが，σ の大きい表 7 では多次元と逐次 1 次元パス追跡に差が見られた．breakpoint が $\boldsymbol{c}$ のパス上に一様に分布していると仮定すると，p が大きい場合は多次元空間上で $\boldsymbol{c}^{(\mathrm{new})}$ へ直線的に動く多次元パス追跡は，1 変数ずつ動かす逐次 1 次元パス追跡に比べて breakpoint は減少する．表 8-10 の探索幅サーチの回数を見ると，多次元と逐次 1 次元パス追跡の breakpoint 数に差が大きい個所では逐次一次元のサーチ回数が breakpoint に比べて大きい (p が大きい) 傾向が見られる．ただし，breakpoint の絶対数が少ない表 5 ではサーチ回数が多くとも breakpoint 数に差はあまりない．

表 8-10 の探索幅サーチ回数については，全ての表において C が小さい時に多次元と逐次 1 次元パス追跡に大きな差が確認できる．ただし，逐次 1 次元パス追跡は C が大きくなるとサーチ回数が急激に減少し，多次元パス追跡と同程度のサーチ回数となっている．これは逐次 1 次元パス追跡において実際に 1 変数パス追跡が行われた回数 p が減少したためである．SVM において，ある程度以上 C を大きくすると上限 C_i に達する（もしくは近い値を持つ）ようなパラメータ α_i は少なくなる．この場合には，減少させる C_i について減少後の値 λC_i が α_i より大きいことが多くなり C_i に関するパス追跡が不要となる．表 11-13 は多次元と逐次 1 次元パス追跡で計算時間の比とサーチ回数の比である．ほとんどの設定で計算時間の比とサーチ回数の比は近い値を持っている．

[1] StatLib `http://lib.stat.cmu.edu/index.php` から入手可能



(a) $\sigma^2 = 0.1$ (b) $\sigma^2 = 1$ (c) $\sigma^2 = 10$

図 5: Fisher river データによる CPU 時間の比較 (MD-Path:多次元パス追跡，SD-Path:逐次 1 次元パス追跡)

表 5: σ^2=0.1 の平均 breakpoint 数

method \ C	10^1	10^2	10^3	10^4
多次元	3.20	3.07	3.07	3.08
逐次 1 次元	3.24	2.02	2.01	2.01

表 6: σ^2=1 の平均 breakpoint 数

method \ C	10^1	10^2	10^3	10^4
多次元	10.06	8.76	9.83	9.52
逐次 1 次元	16.82	11.85	9.21	8.77

表 7: σ^2=10 の平均 breakpoint 数

method \ C	10^1	10^2	10^3	10^4
多次元	32.43	31.13	27.05	29.08
逐次 1 次元	66.59	61.85	46.51	36.44

表 8: σ^2=0.1 の平均サーチ回数

method \ C	10^1	10^2	10^3	10^4
多次元	4.20	4.07	4.07	4.08
逐次 1 次元	135.75	4.03	4.01	4.01

表 9: σ^2=1 の平均サーチ回数

method \ C	10^1	10^2	10^3	10^4
多次元	11.06	9.76	10.83	10.52
逐次 1 次元	171.88	40.15	13.03	10.56

表 10: σ^2=10 の平均サーチ回数

method \ C	10^1	10^2	10^3	10^4
多次元	33.43	32.13	28.05	30.08
逐次 1 次元	302.67	223.67	131.22	64.02

4 まとめ

本稿では，重み付きカーネルマシンの多次元パス追跡法を提案した．具体例として，SVM の個別正則化パラメータに関するパラメータ $\boldsymbol{\alpha}, b$ の区分線形性を導出し，パス追跡によるモデルの連続的な変化の追跡が可能であることを示した．計算機実験では多次元のパス追跡が 1 次元パス追跡の繰り返しや従来の SVM Solver より効率的であることを確認した．

参考文献

[1] R. J. Carroll and D. Ruppert, *Transformation and weighting in regression*. London, UK, UK: Chapman & Hall, Ltd., 1988.

[2] A. Refenes, Y. Bentz, D. Bunn, A. Burgess, and A. Zapranis, “Financial time series modelling with discounted least squares backpropagation,” *Neurocomputing*, vol. 14, no. 2, pp. 123–138, 1997.

[3] L. Cao and F. Tay, “Support vector machine with adaptive parameters in financial time series forecasting,” *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506–1518, 2003.

[4] A. Kolcz and J. Alspector, “SVM-based filtering of e-mail spam with content-specific misclassification costs,” in *In Proceedings of the TextDM ’01 Workshop on Text Mining - held at the 2001 IEEE International Conference on Data Mining*, 2001.

[5] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

表 11: σ^2=0.1 の計算回数比とサーチ回数比

ratio \ C	10^1	10^2	10^3	10^4
計算時間比（一/多）	32.32	0.99	0.99	0.98
サーチ回数比（一/多）	28.69	0.92	0.91	0.90

表 12: σ^2=1 の計算回数比とサーチ回数比

ratio \ C	10^1	10^2	10^3	10^4
計算時間比（一/多）	15.54	4.11	1.20	1.00
サーチ回数比（一/多）	14.42	3.62	1.14	0.99

表 13: σ^2=10 の計算回数比とサーチ回数比

ratio \ C	10^1	10^2	10^3	10^4
計算時間比（一/多）	9.05	6.96	4.68	2.13
サーチ回数比（一/多）	9.03	6.80	4.40	2.05

[6] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *Journal of Machine Learning Research*, vol. 5, pp. 1391–1415, 2004.

[7] T. Gal, *Postoptimal Analysis, Parametric Programming, and Related Topics*. Walter de Gruyter, 1995.

[8] E. N. Pistikopoulos, M. C. Georgiadis, and V. Dua, *Process Systems Engineering: Volume 1: Multi-Parametric Programming*. WILEY-VCH, 2007.

[9] X. Yang, Q. Song, and Y. Wang, "A weighted support vector machine for data classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 5, pp. 961–976, 2007.

[10] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.

[11] J. R. Schott, *Matrix Analysis For Statistics*. Wiley-Interscience, 2005.

[12] G. H. Golub and C. F. Van Loan, *Matrix computations*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[13] F. Liu, T. Zhang, and R. Zhang, "Modified kernel RLS-SVM based multiuser detection over multipath channels," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E86-A, no. 8, pp. 1979–1984, 2003.

[14] H. Funaya, Y. Nomura, and K. Ikeda, "A support vector machine with forgetting factor and its statistical properties," in *Proc. Intl. Conf on Neural Information Processing (ICONIP 08)*.

[15] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods — Support Vector Learning* (B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds.), (Cambridge, MA), pp. 185–208, MIT Press, 1999.

[16] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[17] L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, eds.), pp. 301–320, Cambridge, MA.: MIT Press, 2007.

[18] D. DeCoste and K. Wagstaff, "Alpha seeding for support vector machines," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 345–359, 2000.