



最急上昇法によるサポートベクトルマシンの学習

阿部, 重夫
廣川, 洋一
小澤, 誠一

(Citation)

システム制御情報学会論文誌, 15(11):607-614

(Issue Date)

2002-11-15

(Resource Type)

journal article

(Version)

Version of Record

(URL)

<https://hdl.handle.net/20.500.14094/90001022>



最急上昇法によるサポートベクトルマシンの学習*

阿部 重夫[†]・廣川 洋一[†]・小澤 誠一[†]

Training of Support Vector Machines by the Steepest Ascent Method*

Shigeo ABE[†], Youichi HIROKAWA[†] and Seiichi OZAWA[†]

Since the training of support vector machines needs to solve the dual problem with the number of variables equal to the number of training data, the training becomes slow when the number of training data is large. To speed up training the Sequential Minimal Optimization (SMO) technique has been proposed, in which two data are optimized simultaneously. In this paper, we propose to extend SMO so that more than two data are optimized simultaneously. Namely, we select a working set including variables, solve the equality constraint for one variable included in the working set, and substitute it into the objective function. Then we solve the subproblem related to the working set by calculating the inverse of the Hessian matrix. We evaluate our method for the five benchmark data sets and show the speed-up of training over SMO.

1. はじめに

サポートベクトルマシンでは、入力を高次元の特徴空間に写像することにより、線形分離性を上げ、その特徴空間における最適な分離超平面を二次計画問題の解として求める方法をとる [1–3]. 特徴空間が無次元になる場合もあるために原問題を双対問題に変換して解くことが一般に行われるが、教師データ数分の変数の双対問題を解く必要があるために、データの数が増えると学習時間が増大するという問題がある。このため、参考文献 [4] では、問題を二つに分割して、小さな問題を繰り返して解くことにより高速化をはかる方式を提案している。

二次計画問題の解法としては、共役勾配法、主双対内点法等の種々の方式が開発されており、汎用パッケージも入手可能であるが、問題の規模が大きい場合には上記の分割法と組み合わせて解くことが必要である [5].

またパーセプトロンの学習に開発されたアルゴリズムを特徴空間に拡張する研究もある [6–10]. カーネルアダプトロン [7,8] はアダプトロン [11] を特徴空間に拡張したものである。しかしながら、カーネルアダプトロンは等号制

約を考慮していないために、最適超平面のバイアス項の最適性が保証されていない。このため参考文献 [5] では、入力変数を追加する方法を提案している。Sequential Minimal Optimization (SMO) [12] では参考文献 [4] の分割法において一方の変数の個数を二つにして最適化する方法をとっている。すなわち、一度に二つのデータを最適化しており、共役勾配法により解を求めるよりも大幅に高速化ははかれるとしている。

SMO では、二個のデータを同時に最適化しているが、本論文ではこれを複数個の同時処理に拡張する。二次計画問題において複数個のデータを同時に最適化するとき、Hessian 行列を求めて、修正量を計算すればそのデータに対する最適解が求まるから、Newton 法 [13] の一種と見ることができ、解法は最急上昇法に近いことから、最急上昇法とよぶことにする。

本論文で提案する最急上昇法において、まず教師データから複数個のデータを選択して、SVM の双対問題における等号制約をこの中の一つのデータに対応する Lagrange 乗数に対して解いて、目的関数に代入することにより、等号制約を除く。ついで、選択したデータに対応する Lagrange 乗数を最適化する変化量を Hessian 行列の逆行列を計算することにより求める。このとき Hessian 行列が特異になるときは、特異にならない部分行列の逆行列を求めて、対応する変数を最適化する変化量を求める。

* 原稿受付 2001 年 12 月 18 日

[†] 神戸大学 大学院 自然科学研究科 Graduate School of Science and Technology, Kobe University; Rokkodai, Nada ward, Kobe city, Hyogo 657-8501, JAPAN

Key Words: dual problem, pattern recognition, quadratic programming, steepest ascent, support vector machines.

以下2.で[1,2]に基づき、二クラス分離問題を解くSVMの概要を述べ、3.で最急上昇法による学習法を述べる。さらに、4.でいくつかのベンチマークデータにより、最急上昇法とSMOの性能比較を行い、最急上昇法の高速度化の効果を検証する。また、分割法と併用した内点法との性能比較も行う。

2. ニクラスSVM

SVMでは、もともと二クラス分離問題を扱い、入力空間を高次元の特徴空間に写像し、特徴空間内で、分離超平面と最も近いデータとの距離（マージン）が最大になる超平面を最適化手法によって決定する。

ここで、入力 $\mathbf{x} \in R^m$ に対するラベルを $y \in \{-1, 1\}$ とし、 $y=1$ の集合をクラス I、 $y=-1$ の集合をクラス II とする。この入力の特徴空間上の超平面

$$D(\mathbf{x}) = \mathbf{w}^t \mathbf{g}(\mathbf{x}) + b$$

によって分離することを考える。ただし、 $\mathbf{w}$ は特徴空間上の重みベクトル、 $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \dots, g_i(\mathbf{x}), \dots)^t$ は特徴空間への写像関数、 b はスカラーである。

この超平面により、クラス I のデータが $D(\mathbf{x}) \geq 1$ となり、クラス II のデータが $D(\mathbf{x}) \leq -1$ となるように分離されるためにはつぎの条件を満たさなければならない。

$$y_i(\mathbf{w}^t \mathbf{g}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \dots, M \quad (1)$$

ここで、 ξ_i は、特徴空間上で線形分離でない場合も許すための非負のスラック変数であり、 M は教師データの個数を表す。すなわち、 $\xi_i > 1$ となるデータ $\mathbf{x}$ は $D(\mathbf{x}) = 0$ で分離できていないことになる。(1)式を制約条件として、マージンを最大化する超平面を求めるために、つぎの目的関数を最小化する。

$$\min Q_p(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad (2)$$

ここで C は教師データに対する認識率と汎化能力のトレードオフを与えるパラメータでマージンパラメータとよぶ。つまり、 C の値が大きければ大きいほど、(2)式の第二項の最小化が図られるために、教師データに対して高い認識率を要求することになる。

特徴空間が無限次元のときは数値的に解くのが難しいため、Lagrange 乗数 α_i を用いてつぎの等価な双対問題に変換する。

$$\begin{aligned} \max Q_d(\alpha) &= \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=0}^M \alpha_i \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad &\sum_{i=1}^M y_i \alpha_i = 0 \\ &C \geq \alpha_i \geq 0 \quad i = 1, \dots, M \end{aligned} \quad (3)$$

ただし、 $\alpha = (\alpha_1, \dots, \alpha_M)^t$ で、 $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\mathbf{x})^t \mathbf{g}(\mathbf{x}')$ は特徴空間上の内積を表す kernel 関数である。これにより高次元の特徴空間上の変数を直接扱う必要がなくなる。

この二次計画問題を解くことにより SVM を構成することができる。この二次計画問題の解のうち $\alpha_i \neq 0$ のデータのみが、識別関数に影響を与える。これらのデータをサポートベクトルという。

3. 最急上昇による学習法

3.1 分割法による学習

ワーキングセット分割法による学習では、変数の集合 $\{\alpha_i | i = 1, \dots, M\}$ をワーキングセット $\alpha_B = \{\alpha_i | i \in B\}$ と $\alpha_N = \{\alpha_i | i \in N\}$ とに分割する [4]。ここで、 B および N はインデックスの集合で、 $B \cup N = \{1, \dots, M\}$ 、 $B \cap N = \emptyset$ である。このとき α_N を固定して、以下の問題を解く。

$$\begin{aligned} \max Q_d(\alpha_B) &= \sum_{i \in B} \alpha_i - \frac{1}{2} \sum_{i,j \in B} \alpha_i \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - \sum_{\substack{i \in B \\ j \in N}} \alpha_i \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - \frac{1}{2} \sum_{i,j \in N} \alpha_i \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \sum_{i \in N} \alpha_i \end{aligned} \quad (4)$$

$$\text{subject to} \quad \sum_{i \in B} y_i \alpha_i = - \sum_{i \in N} y_i \alpha_i \quad (5)$$

$$0 \leq \alpha_i \leq C \quad \text{for } i \in B \quad (6)$$

部分問題の解が得られたら α_N の中で最適解の条件を満たさないものを選び出し、これらをワーキングセットにして、部分問題を解く。このようにして、最適解の条件を満たさないものがなくなるまで繰り返す。SMO ではワーキングセットのサイズを二個にして、二個の変数の最適化を繰り返すことにより最適解を求めている [12]。これに対して本論文では、ワーキングセットサイズを2個以上にして、最急上昇法で解く。以下にそのアルゴリズムを詳細に説明する。

3.2 学習アルゴリズム

3.2.1 修正量の計算

部分問題 (4)-(6) を α_B について解くために、まず (5) 式の等式を $\alpha_s \in \alpha_B$ で解くと次式が得られる。

$$\alpha_s = - \sum_{\substack{i=1 \\ i \neq s}}^M y_i y_s \alpha_i \quad (7)$$

(7) 式を (4) 式に代入することにより、等号制約を除くことができる。ここで $B' = \{\alpha_i | i \neq s, i \in B\}$ とすると、 $Q_d(\alpha_{B'})$ は二次であるから、 $\alpha_{B'}$ の微小変化 $\Delta \alpha_{B'}$ に対する $Q_d(\alpha_{B'})$ の微小変化 $\Delta Q_d(\alpha_{B'})$ は次式で与えられる。

$$\Delta Q_d(\alpha_{B'}) = \frac{\partial Q_d(\alpha_{B'})}{\partial \alpha_{B'}} \Delta \alpha_{B'} + \frac{1}{2} \Delta \alpha_{B'}^t \frac{\partial^2 Q_d(\alpha_{B'})}{\partial \alpha_{B'}^2} \Delta \alpha_{B'} \quad (8)$$

ここで

$$\frac{\partial \alpha_s}{\partial \alpha_i} = -y_s y_i \quad (9)$$

であるから、 $Q_d(\alpha_{B'})$ の α_i ($i \neq s, i \in B'$) に対する偏微分は次式で与えられる。

$$\begin{aligned} \frac{\partial Q_d(\alpha_{B'})}{\partial \alpha_i} &= 1 + \frac{\partial \alpha_s}{\partial \alpha_i} - \sum_{j=1}^M \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - \sum_{j=1}^M \alpha_j y_s y_j \frac{\partial \alpha_s}{\partial \alpha_i} H(\mathbf{x}_i, \mathbf{x}_j) \\ &= 1 - y_s y_i - \sum_{j=1}^M \alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \sum_{j=1}^M \alpha_j y_j y_i H(\mathbf{x}_s, \mathbf{x}_j) \end{aligned} \quad (10)$$

(10)式を用いると、 α_j ($j \neq s, j \in B'$) に対する $Q_d(\alpha_{B'})$ の二次の偏微分は次式で与えられる。

$$\frac{\partial^2 Q_d(\alpha_{B'})}{\partial \alpha_i^2} = -H(\mathbf{x}_i, \mathbf{x}_i) + 2H(\mathbf{x}_i, \mathbf{x}_s) - H(\mathbf{x}_s, \mathbf{x}_s) \quad (11)$$

$$\frac{\partial^2 Q_d(\alpha_{B'})}{\partial \alpha_i \partial \alpha_j} = y_i y_j (-H(\mathbf{x}_i, \mathbf{x}_j) + H(\mathbf{x}_i, \mathbf{x}_s) + H(\mathbf{x}_s, \mathbf{x}_j) - H(\mathbf{x}_s, \mathbf{x}_s)) \quad (12)$$

ここで $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\mathbf{x})^t \mathbf{g}(\mathbf{x}')$ であるから、

$$\frac{\partial^2 Q_d(\alpha_{B'})}{\partial \alpha_{B'}^2} = - \left(\cdots y_i (\mathbf{g}(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_s)) \cdots \right)^t \left(\cdots y_j (\mathbf{g}(\mathbf{x}_j) - \mathbf{g}(\mathbf{x}_s)) \cdots \right) \quad (13)$$

となる。したがってもし $\mathbf{x}_i = \mathbf{x}_s$ ($i, s \in B$) なら、 $\partial^2 Q_d(\alpha) / \partial \alpha_{B'}^2$ は特異になる。これより、 $\mathbf{x}_s$ と同じ $\mathbf{x}_i$ を選択するのを避ける必要がある。ここで $-\partial^2 Q_d(\alpha) / \partial \alpha_{B'}^2$ は転置行列と元の行列の積で表現されるために、半正定である。正定と仮定して、かつ上下限の制約がないとすると、 $\Delta Q_d(\alpha_{B'})$ は

$$\Delta \alpha_{B'} = - \left(\frac{\partial^2 Q_d(\alpha_{B'})}{\partial \alpha_{B'}^2} \right)^{-1} \frac{\partial Q_d(\alpha_{B'})}{\partial \alpha_{B'}} \quad (14)$$

で最大値をとる。

ここで、 α_i ($i = 1, \dots, M$) が (5) 式を満たすとすると (5), (14) 式より、 α_s の修正ベクトルとして次式を得る。

$$\Delta \alpha_s = - \sum_{i \in B'} y_s y_i \Delta \alpha_i \quad (15)$$

したがって、 α_B の修正ベクトル $\Delta \alpha_B$ は (14), (15) 式により与えられる。 α_i ($i \in B$) において、ここでもし

$$\alpha_i = 0, \quad \Delta \alpha_i < 0 \quad \text{あるいは} \quad (16)$$

$$\alpha_i = C, \quad \Delta \alpha_i > 0 \quad (17)$$

のときは、 α_B を修正すると、下限あるいは上限に違反するために、修正できない。したがって、このときはこれらの変数をワーキングセットから除外して、上記の手順を繰り返す。

もし (16), あるいは (17) 式がワーキングセットの変数に対して成立しないときは、変数の修正が行える。ここで $\Delta \alpha'_i$ を上下限の制約条件を満たすように $\Delta \alpha_i$ を修正したものとする。すなわち $\alpha_i + \Delta \alpha_i < 0$ のときは、 $\Delta \alpha'_i = -\alpha_i$ とし、 $\alpha_i + \Delta \alpha_i > C$ のときは $\Delta \alpha'_i = C - \alpha_i$ とする。これ以外では $\Delta \alpha'_i = \Delta \alpha_i$ とする。つぎに

$$r = \min_{i \in B} \frac{\Delta \alpha'_i}{\Delta \alpha_i} \quad (18)$$

を計算する。ただし $0 < r \leq 1$ である。

以上より、 α_B を次式により修正する。

$$\alpha_B^{\text{new}} = \alpha_B^{\text{old}} + r \Delta \alpha_B \quad (19)$$

以上の計算で、 α_B^{new} が等号制約を満たすことは明らかである。さらに、 $Q_d(\alpha_B^{\text{new}}) \geq Q_d(\alpha_B^{\text{old}})$ が満たされることも明らかである。

3.2.2 $\partial^2 Q_d(\alpha_{B'}) / \partial \alpha_{B'}^2$ が特異のときの処理

もし $\partial^2 Q_d(\alpha_{B'}) / \partial \alpha_{B'}^2$ が特異のときは特異値分解を用いてもよいが、時間がかかるために、特異になる変数をワーキングセットから削除する方法をとる。このときに、削減されたワーキングセットの変数の数が1より大きいときは、3.2.1の修正量の計算を縮小されたワーキングセットに対して行う。ここで (14) 式の逆行列の計算を対称Cholesky分解で行うことを考える。 $\partial^2 Q_d(\alpha_{B'}) / \partial \alpha_{B'}^2$ の i 番目の列を分解しているときに、下三角行列の i 番目の対角要素があらかじめ指定した値よりも小さいときは、 $\partial^2 Q_d(\alpha_{B'}) / \partial \alpha_{B'}^2$ の i 番目の行および列を捨てて、 $(i+1)$ 番目の列に移る。この処理を簡単にするために、以下のシミュレーションでは、三角行列分解をこの時点で中止して、 i 番目から $(|B'|)$ 番目までの対角要素に対応する変数を捨てることにする。

3.2.3 収束計算

V をサポートベクトルの候補のインデックス集合とする。学習の最初では、 V はすべてのインデックスを含む。すなわち、 $V = \{1, \dots, M\}$ である。ワーキングセットの最大の大きさ $|B|_{\max}$ が与えられているとする。このときワーキングセットの大きさを $|V|$ の値の変化にしたがって以下のように変える。

$$|B| = \begin{cases} |B|_{\max} & \text{for } |V| \geq |B|_{\max} \\ \max\{2, |V|\} & \text{for } |V| < |B|_{\max} \end{cases} \quad (20)$$

すべての変数 α_i ($i \in V$) を一度修正することを一回の学習の繰り返しということにする。おのおのの学習の繰り返しの最初に、一つの変数 α_i ($i \in V$) を選択して、

$B = \{i\}$, $V = V - \{i\}$ とする。ついで, $(|B|-1)$ 個の変数 $\alpha_i (i \in V)$ をランダムに選び, V から対応する変数のインデックスを削除し, B に変数のインデックスを追加する。ただし, $|B|$ は (20) 式で与えられる。つぎに, 変数ベクトル α_B^{new} を計算する。この手順を V が空集合になるまで繰り返せば, ワーキングセットのすべての変数を処理したことになる。すなわち, 一回の学習の繰り返しを行ったことになる。

α_B^{new} を計算するごとに,

$$|r\Delta\alpha_i| < \varepsilon_{\text{var}} \quad (21)$$

がすべての $i (i \in B)$ に対して成立するか調べる。ただし, ε_{var} は変数に対する収束判定の条件である。学習の繰り返しの最後に, 上下限内の変数, すなわち $0 < \alpha_i < C$ を満たす $\alpha_i (i \in B)$ に対するバイアス項 b_i を以下の式で計算する。

$$b_i = y_i - \sum_{j \in B} \alpha_j H(\mathbf{x}_j, \mathbf{x}_i) \quad (22)$$

またそれらの平均 b_{ave} を

$$b_{\text{ave}} = \frac{1}{N_b} \sum_{\substack{i \in B \\ 0 < \alpha_i < C}} b_i \quad (23)$$

で計算する。ただし, N_b は $0 < \alpha_i < C$ を満たす $\alpha_i (i \in V)$ の数である。

学習を高速化するために, 連続した N_{ite} 回の繰り返し計算において, サポートベクトルの候補内で計算を行う [12]。ここで, N_{ite} は正の整数である。すなわち, 毎繰り返し計算の最後に, $\alpha_i = 0$ となるインデックス i を V から削除して学習を進める。

もし変数が削除されずに, 繰り返し計算途中ですべての α_B に対して (21) 式が成立し, また $i (i \in V, 0 < \alpha_i < C)$ について

$$\frac{|b_{\text{ave}} - b_i|}{|b_{\text{ave}}|} \leq \varepsilon_b \quad (24)$$

が成立するとする。ただし ε_b は小さな正のパラメータである。このとき

$$y_j \left(\sum_{i \in V} \alpha_i y_i H(\mathbf{x}_i, \mathbf{x}_j) + b_{\text{ave}} \right) < 1 \quad (25)$$

を満たすサポートベクトルの新しい候補 $\mathbf{x}_i (i \notin V)$ があるかどうかを調べる。もしないときは学習を終了する。あるときは, インデックスを V に加えて学習を進める。 N_{ite} 回連続してサポートベクトルを削除したつぎの繰り返し計算で, すなわち $(N_{\text{ite}}+1)$ 回目ごとの繰り返し計算の最後に (25) 式を調べサポートベクトルの候補があれば追加する。なおこの処理は必要なサポートベクトルが永久に除外されるのを防ぐために行う。

学習をさらに高速化するために, 目的関数 $Q_d(\alpha)$ の

相対的変化が小さくなったときに計算を終了する。すなわち, つぎの終了条件を設ける。

$$\frac{Q_d^{(n+1)}(\alpha) - Q_d^{(n)}(\alpha)}{Q_d^{(n)}(\alpha)} \leq \varepsilon_q \quad (26)$$

ここで ε_q は小さな正の数である。もしこの条件が繰り返し計算の初期の段階で適用されると, 最適解に収束する前に計算が終了する可能性がある。このために, 上記の終了条件を $N_q (> 1)$ 回の繰り返し計算の後から適用するようにする。

3.2.4 キャッシュメモリ

$H(\mathbf{x}_i, \mathbf{x}_j)$ の計算は入力次元が大きくなると計算量が増大する。このために, いったん計算した $H(\mathbf{x}_i, \mathbf{x}_j)$ を記憶することにより, 計算回数を削減するために, $H(\mathbf{x}_i, \mathbf{x}_j)$ に対する $N_M \times N_M$ 次の行列 H を用意する [12]。 α_i と α_j がサポートベクトルの候補 ($i, j \in V$) のときに, $H(\mathbf{x}_i, \mathbf{x}_j)$ を計算した際に, コンピュータのキャッシュメモリと同様に H に格納する。すなわち, もし $i=j$ で $H(\mathbf{x}_i, \mathbf{x}_i)$ が H に格納されておらず, k 番目の行と列が空いているときは, i を k 番目の行と列に割り当てて, $H(\mathbf{x}_i, \mathbf{x}_i)$ を H_{kk} に格納する。もし, $i \neq j$ で i と j が割り当てられていないときは, i と j を空いている k_1 行, k_2 列に割り当てて, $H(\mathbf{x}_i, \mathbf{x}_j)$ を $H_{k_1 k_2}$ に格納する。もし i と j のどちらかがすでに割り当てられているときは, 割り当てられていない方を割り当てて, $H(\mathbf{x}_i, \mathbf{x}_j)$ を H に格納する。 α_i が V から削除されて, i が H の行 (列) に割り当てられているときは, これを削除する。

H を設定する効果を調べるために, H へのアクセスに対するヒット率を次のように定義する。

$$\frac{H \text{ に含まれていた回数}}{\text{全体のアクセス回数}} \times 100(\%) \quad (27)$$

4. 性能評価

Table 1 に示す五種類のデータを用いて学習時間の評価を行った。表において甲状腺データは 21 入力のうち 16 入力に離散値を取りクラス間にオーバーラップのあるデータである。また, 白血球識別のための血球データは白血球の成長段階でクラスが定義されているために, クラス間のオーバーラップがあるデータである。これに対して, 三種類のひらがなデータは, ナンバープレートのひらがなデータを収集したものであり, クラス間のオーバーラップが少ないデータである。

シミュレーションにおいてパラメータを以下のように設定した。 $C = 5000$, $\varepsilon_b = 10^{-2}$, $\varepsilon_{\text{var}} = 10^{-6}$, $\varepsilon_q = 10^{-7}$, $N_{\text{ite}} = 10$, $N_q = 50$, $N_M = 400$ とした。ただし, 甲状腺データおよび血球データでは, $N_q = 50$ では最適値に収束する前に学習が終了するために以下のように大きい値に設定した。すなわち, 甲状腺データの $d = 4$ 以外, および血球データで $N_q = 100$ とし, 甲状腺データで $d = 4$ のときは $N_q = 600$ とした。

Table 1 Feature of benchmark data

Data	Inputs	Classes	Train.	Test
Thyroid	21	3	3772	3428
Blood cell	13	12	3097	3100
Hiragana-50	50	39	4610	4610
Hiragana-105	105	38	8375	8356
Hiragana-13	13	38	8375	8356

SVM を構成する kernel 関数としてはつぎの関数を用いた。

- 内積（線形分離）： $H(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t \mathbf{x}'$
- 多項式： $H(\mathbf{x}, \mathbf{x}') = [\mathbf{x}^t \mathbf{x}' + 1]^d$
- RBF： $H(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$

また、ひらがなデータの学習に際しては、データの次元数が大きいため、多項式カーネルにおいて最適化問題の解が小さくなり過ぎ、学習が途中で止まるという問題が起るため、これらのデータでは、教師データの kernel の最大値で kernel 関数を割った

$$H_{new}(\mathbf{x}, \mathbf{x}') = \frac{H(\mathbf{x}, \mathbf{x}')}{\max H(\mathbf{x}_t, \mathbf{x}_t')}$$

を kernel 関数として用いた。ここで $\mathbf{x}_t$ は教師データを表す。

なお、以下の表で示す計算時間は SUN UltraSPARC-IIi (335MHz) ワークステーションを用いて測定した。

4.1 SMO との比較

甲状腺データ、血球データのようにクラス間のオーバーラップが激しい場合には、最適な超平面を決定するのが難しくなる。このような場合は、複数の教師データの関係を考慮しながら最適な超平面を決定する必要があるが、SMO では一度に二つのデータしか最適化しないために反復回数が増える。これに対して複数個のデータを同時処理する SAM では反復回数を減らすことができ、高速化できると期待できる。しかしながらワーキングセットサイズを大きくしすぎると逆行列の計算量が増えることと、行列が特異になる可能性が高くなることより、一回の繰り返し計算の計算量が増加する。Table 1 のデータでは甲状腺、血球のデータでもワーキングセットのサイズを 100 以上にするとオーバーヘッドにより計算時間が増大するために、以下の評価ではワーキングセットサイズを 100 以下に設定して SAM に対する高速化を評価した。なおワーキングセットサイズを変えると収束が異なってくるために、以下の評価では異なるワーキングセットサイズで最終の認識率が異なる場合が生じている。

Table 2 に甲状腺データに対する結果を示す。表において Parm は多項式の次数あるいは RBF のパラメータ、Size はワーキングセットのサイズ $|B|_{\max}$ 、Rates はテストデータ（教師データ）に対する認識率、Epochs は収束に要した繰り返し回数、Time は学習時間を示す。学

Table 2 Performance for thyroid data

Parm	Size	Rates [%]	Epochs	Time [s]
$d=1$	2	97.43 (98.14)	4927	1531
	10	97.37 (98.70)	1803	931
	20	97.58 (98.49)	1870	1139
$d=4$	2	95.04 (97.56)	3601	2033
	20	96.27 (98.91)	1839	636
	40	96.56 (99.97)	663	205
	60	96.62 (100)	57	109
$\gamma=10$	2	95.13 (100)	2720	4044
	20	95.16 (100)	841	1356
	40	95.13 (100)	408	717
	60	95.16 (100)	299	553
	80	95.16 (100)	237	424

Table 3 Performance for blood cell data

Parm	Size	Rates [%]	Epochs	Time [s]
$d=1$	2	91.23 (96.84)	1008	463
	4	91.29 (96.77)	480	339
	13	91.32 (96.84)	176	150
$d=4$	2	92.00 (99.00)	595	338
	19	92.32 (100)	69	65
	60	92.35 (100)	15	31
$\gamma=10$	2	91.74 (100)	128	238
	20	91.74 (100)	68	163
	40	91.74 (100)	105	165

習時間には、教師データ、テストデータの認識に要した計算時間は含めていない。Size が 2 のときが SMO に対応するが、参考文献 [12] の方法をそのままインプリメントしたわけではないので、参考文献 [12] の SMO の処理性能とは必ずしも一致はしない。したがって異なるワーキングセットサイズとの比較において意味があると考えていただきたい。

表において $d=4$ 、 $\gamma=10$ では教師データでの認識率が 100% になっており、この場合は、ワーキングセットサイズが 2 に対してサイズを大きくすることにより、学習が最大で 19 倍および 9.5 倍に高速化されている。しかしながら、 $d=1$ のときは教師データの認識率が 100% になっておらず、1.6 倍の高速化にとどまっている。

Table 3 に血球データに対する結果を示す。表において $d=4$ 、 $\gamma=10$ では教師データでの認識率が 100% になっているが、甲状腺データと異なり、 $d=1, 4$ のときに SMO に対して最大で 3.1 倍および 11 倍学習が高速化されており、 $\gamma=10$ のときは最大で 1.4 倍の高速化にとどまっている。

Table 4 に 50 入力のひらがなデータに対する結果を示す。表において $d=2$ 、 $\gamma=0.1$ で教師データでの認識率が 100% になっており、SMO に対して最大で 1.7 倍およ

Table 4 Performance for hiragana-50 data

Parm	Size	Rates [%]	Epochs	Time [s]
$d=2$	2	98.98 (100)	108	202
	5	98.96 (100)	59	127
	10	98.94 (100)	44	118
$\gamma=0.1$	2	98.98 (100)	100	321
	5	99.02 (100)	57	201
	10	99.02 (100)	51	194

Table 5 Performance for hiragana-105 data

Parm	Size	Rates [%]	Epochs	Time [s]
$d=2$	2	100 (100)	112	671
	5	100 (100)	61	423
	10	100 (100)	52	394
$\gamma=0.1$	2	100 (100)	96	2262
	5	100 (100)	58	1534
	10	100 (100)	53	1479

Table 6 Performance for hiragana-13 data

Parm	Size	Rates [%]	Epochs	Time [s]
$d=2$	2	99.63 (100)	136	139
	5	99.63 (100)	61	92
	10	99.62 (100)	50	93
$\gamma=1$	2	99.76 (100)	129	325
	5	99.76 (100)	65	209
	10	99.76 (100)	54	190

び1.6倍学習が高速化されている。

Table 5に105入力ひらがなデータに対する結果を示す。表において $d=2$, $\gamma=0.1$ で教師データでの認識率が100%になっており、SMOに対して最大で1.7倍および1.5倍学習が高速化されている。

Table 6に13入力ひらがなデータに対する結果を示す。表において $d=2$, $\gamma=0.1$ で教師データでの認識率が100%になっており、SMOに対して最大で1.5倍および1.7倍学習が高速化されている。

以上のシミュレーション結果よりクラス間のオーバーラップのある甲状腺、血球データでは最適なワーキングセットサイズは20-80で、SMOに対して1.4倍から19倍の高速化が得られた。これに対してクラス間のオーバーラップがほとんどないひらがなデータでは最適なワーキングセットサイズは10で、学習は1.5倍から1.7倍に高速化された。このように、クラス間のオーバーラップがある場合は、SAMによる高速化の効果が高いことがシミュレーションによっても確かめられた。

4.2 キャッシュメモリの効果

入力次元が大きくなると、カーネル関数の値を計算するのに時間がかかるためにいったん計算したカーネル関

数の値をキャッシュメモリに格納して、再度必要になったときに使用することにより、カーネル関数の値を計算するオーバーヘッドを減らすことができる。

Table 7に 400×400 次元のキャッシュメモリ H を設けたときの学習の加速化率を示す。表において各データセットでヒット率の最小値と最大値を示すときの学習の加速化率を示している。甲状腺データ、血球データでは1.3倍から1.7倍の高速化でそれほどの効果はないが、入力数の多い50入力および105入力のひらがなデータでは2倍から4.3倍の高速化が達成されている。これに対して13入力のひらがなデータで $d=2$ のときは学習が逆に減速している。

Table 7 Speedup of training by the cache memory

Condition	Size	Hit [%]	Speedup
THY ($d=4$)	60	49	1.3
THY ($\gamma=10$)	80	71	1.7
BLD ($d=1$)	13	89	1.7
BLD ($\gamma=10$)	60	76	1.4
H-50 ($d=2$)	2	70	2.0
H-50 ($\gamma=0.1$)	20	84	2.8
H-105 ($d=2$)	2	59	2.0
H-105 ($\gamma=0.1$)	10	86	4.3
H-13 ($d=2$)	2	48	0.9
H-13 ($\gamma=1$)	10	68	1.5

4.3 LOQO との比較

Table 8に主双対内点法 (LOQO) [14] に分割法を併用したロンドン大学のソフトウェア¹を用いて学習時間の比較を行った結果を示す。LOQOでは血球データでワーキングセットサイズを変えたときに50で学習時間が最短になったので、ワーキングセットサイズを50にして分割法を用いて計算した。表において学習時間の欄の括弧内の数字はワーキングセットサイズが2 (SMO) のときの学習時間を基準にした高速化率を示す。SAMは最急上昇法を示し、Tables 2-6より最も計算時間が短くなった結果を用いた。LOQOのインプリメンテーションの詳細が分からず、またLOQOのワーキングセットサイズをさらにデータセットごとに最適化することにより、計算時間が短縮される可能性があるため厳密な時間の比較は意味を持たない。

ひらがなデータではLOQOとSAMは同程度の学習時間になっているが、甲状腺データおよび血球データで $d=1$ のときはLOQOの方がSAMよりも高速である。これらのデータはクラス間の重なりが多いが、これが学習時間の差となって現れたと思われる。参考文献[12]では、クラス間の重なりがないときはSMOが共役勾配法よりも大幅に高速化されたとあるが、これはワーキングセットサイズが500と大きかったためではないかと思わ

¹<http://svm.cs.rhbc.ac.uk>

Table 8 Performance comparison of training methods

Condition	LOQO [s]	SMO [s]	SAM [s]
THY ($d=1$)	38 (40)	1531 (1)	931 (1.6)
THY ($d=4$)	14 (145)	2032 (1)	109 (19)
THY ($\gamma=10$)	108 (37)	4044 (1)	424 (9.5)
BLD ($d=1$)	20 (23)	463 (1)	150 (3.1)
BLD ($d=4$)	19 (18)	338 (1)	31 (11)
BLD ($\gamma=10$)	185 (1.3)	238 (1)	165 (1.4)
H-50 ($d=2$)	143 (1.4)	202 (1)	117 (1.7)
H-50 ($\gamma=0.1$)	288 (1.1)	321 (1)	194 (1.6)
H-105 ($d=2$)	318 (2.1)	671 (1)	394 (1.7)
H-105 ($\gamma=0.1$)	1889 (1.2)	2262 (1)	1479 (1.5)
H-13 ($d=2$)	99 (1.4)	136 (1)	93 (1.5)
H-13 ($\gamma=1$)	181 (1.8)	325 (1)	190 (1.7)

れる。LOQOの場合ワーキングサイズを500とすると学習速度は大幅に低下する。たとえば血球データで $d=4$ とすると学習に936秒かかり、SMOに比べても遅くなる。このように従来手法に分割法を適用するときにも最適なワーキングセットサイズを設定することが重要であることが分かる。

SAMの最適なワーキングセットサイズは問題に依存するが、大まかな指標としてはクラス間のオーバーラップが少ない(分類しやすい)と予想されるときは小さい値(10程度)に、クラス間のオーバーラップが大きい(分類しにくい)と予想されるときは大きい値(100以内)に設定すればよい。さらに、最適な値に設定するためには、学習の途中で学習が難しいかどうかの判定をして $|B|_{\max}$ の値を調整することも考えられるが、これに関しては今後の課題としたい。

5. おわりに

本論文では、サポートベクトルマシン学習を高速化するために、変数の中から二個のデータを選択して最適化するSMO法を拡張して二個以上のデータを同時に最適化する最急上昇法を提案した。

この方式をいくつかのデータで評価した結果、三個以上同時処理することにより、二個同時処理するよりも高速化されること、またクラス間の重なりがある場合は高速化率が高いことを示した。分割法と併用した双対内点法との比較では、同時最適化の個数を増やすことにより、クラス間の重なりが少ないときは同程度の計算時間であることを示した。

謝 辞

血球データを提供していただいた川崎医科大学松田信義教授、数字およびひらがなデータを提供していただいた(株)日立製作所日立研究所小林芳樹技術主幹、およびサイロイドデータ(ics.uci.edu: pub/machine-

learning-databases)を提供いただいたカリフォルニア大学アーバイン校P. M. Murphy氏およびD. W. Aha氏に感謝の意を表する。

参 考 文 献

- [1] V. Cherkassky and F. Mulier: *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons (1998)
- [2] V. N. Vapnik: *Statistical Learning Theory*, John Wiley & Sons (1998)
- [3] S. Abe: *Pattern Classification: Neuro-fuzzy Methods and Their Comparison*, Springer-Verlag (2001)
- [4] E. Osuna, R. Freund and F. Girosi: An improved training algorithm for support vector machines; *Proc. 1997 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing VII*, pp. 276–285 (1997)
- [5] N. Cristianini and J. Shawe-Taylor: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press (2000)
- [6] Y. Freund and R. E. Schapire: Large margin classification using the perceptron algorithm; *Machine Learning*, Vol. 37, No. 3, pp. 277–296 (1999)
- [7] T. -T. Frieß, N. Cristianini and C. Campbell: The kernel-adatron algorithm: A fast and simple learning procedure for support vector machines; *Proc. Fifteenth International Conference on Machine Learning*, pp. 188–196 (1998)
- [8] C. Campbell, T. -T. Frieß and N. Cristianini: Maximal margin classification using the KA algorithm; *Proc. 1st International Symposium on Intelligent Data Engineering and Learning*, pp. 355–362 (1998)
- [9] I. Guyon and D. G. Stork: Linear discriminant and support vector classifiers; *Advances in Large Margin Classifiers* (A. J. Smola, P. L. Bartlett, B. Schölkopf and D. Schuurmans, Eds.), pp. 147–169, The MIT Press (2000)
- [10] J. Xu, X. Zhang and Y. Li: Large margin kernel pocket algorithm; *Proc. International Joint Conference on Neural Networks*, Vol. 2, pp. 1480–1485 (2001)
- [11] J. K. Anlauf and M. Biehl: The Adatron—An adaptive perceptron algorithm; *Europhysics Letters*, Vol. 10, pp. 687–692 (1989)
- [12] J. C. Platt: Fast training of support vector machines using sequential minimal optimization; *Advances in Kernel Methods: Support Vector Learning*, (B. Schölkopf, C. J. C. Burges and A. J. Smola, Eds.), pp. 185–208, The MIT Press (1999)
- [13] D. P. Bertsekas: *Nonlinear Programming*, Athena Scientific, Second edition (1999)
- [14] R. J. Vanderbei: LOQO: An interior point code for quadratic programming, Princeton University SOR-

94-15 (1998)

著者略歴

あ べ しげ お
阿 部 重 夫 (正会員)

論文誌 Vol. 15, No. 1, p.33 参照

ひろ かわ よう いち
廣 川 洋 一

2001 年神戸大学工学部電気電子工学科卒業。同年神戸大学大学院自然科学研究科電気電子工学専攻入学。サポートベクトルマシンの研究に従事。

お ざわ せい いち
小 澤 誠 一 (正会員)

1989 年神戸大学大学院工学研究科計測工学専攻修士課程修了。同年奈良工業高等専門学校情報工学科助手，1993 年大阪教育大学教育学部教養学科情報科学専攻助手，同講師を経て，2000 年 1 月より神戸大学大学院自然科学研究科システム機能科学専攻助教授となり，現在に至る。ニューラルネットによる知的情報処理の研究に従事。博士（工学）。電子情報通信学会，計測自動制御学会，日本神経回路学会などの会員。