2007－AVM－58 (9)
2007／9／26

サポートベクトルマシンとその効率的学習アルゴリズム

高橋 規一
九州大学大学院システム情報科学研究院情報工学部門
〒819-0395 福岡市西区元岡 744
norikazu@csce.kyushu-u.ac.jp

パターン認識や機械学習の分野で注目を集めているサポートベクトルマシン (SVM) の基本原理とその効率的学習アルゴリズムである分割法について解説する．前半ではマージン最大化，ソフトマージン，カーネルトリックといったSVM を特徴づける種々の手法について説明し，後半では分割法の基本アイデア，代表的なアルゴリズム，大域収束性に関する最近の結果を紹介する．

Support Vector Machine and its Efficient Learning Algorithms

Norikazu Takahashi
Department of Computer Science and Communication Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395 Japan
norikazu@csce.kyushu-u.ac.jp

Support vector machine (SVM) has been attracting considerable attention in the field of pattern recognition, machine learning, and so on. The purpose of this report is to review fundamental principles of the SVM and the decomposition method which is widely used for the training of SVMs. In the first part, some key ideas characterizing SVMs such as margin maximization, soft margin, and kernel trick are explained. In the second part, the basic idea behind the decomposition method, some representative algorithms, and recent results on its global convergence are introduced.

1 はじめに

パターン認識，機械学習，ニューラルネットワーク，データマイニングなどの分野において，1990 年代後半からサポートベクトルマシン[*1](Support Vector Machine, SVM) とよばれるパターン識別器が注目を集め [1]–[6]，現在では定番手法の一つとして定着している．SVM は，最も単純な線形識別器を基に，マージン最大化やカーネルトリックといった手法を巧妙に組み込んで構築されたパターン識別器であり，汎用のパターン識別器としてこれまで広く用いられてきた多層パーセプトロンのいくつかの欠点を解消するなど，数々の優れた特徴を有している．実際，文字認識やテキスト分類といった様々な分野に応用され，従来手法よりも優れていることが報告されている．その一方で，大規模データに対する計算コストの高さやハイパーパラメータの設定の困難さといった問題点も認識され [6]，これを解決するための研究も盛んに行われている [7]–[9].

本稿では，SVM の基本原理とその効率的学習アルゴリズムについて解説する．前半では，SVM を特徴づけるマージン最大化，ソフトマージン，カーネルトリック等の手法を説明し，SVM が広く普及した理由と解決すべき問題点を整理する．後半では，SVM の効率的学習アルゴリズムとして広く用いられている分割法について，その基本アイデア，代表的な分割法の例，大域収束性に関する最近の結果を示す．

2 SVM の基本原理

2.1 パターン識別

パターン識別の例として，与えられた文字がアルファベットの「A」であるか否かを判定する問題を考えよう．文字はすべて $M \times N$ 画素の画像として表現されるとする．また我々の手元には，人間の目によって「A」であるか否かが判定された文字が L 個あるとする．このとき，判定済みの L 個のサンプルを基に，未知の文字が「A」であるか否かを判定する識別器を構成するにはどうすればよいか．ここで重要なのは，識別器が L 個のサンプルだけでなく未知の文字に対しても正しく判定できるようにすることである．各文字は $M \times N$ 画素の画像として与えられるので，$n = MN$ 次元の実ベクトル $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ で表現される（肩符 T は転置を表す）．判定済みの L 個のサンプルを $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_L$ とし，それらに対応するクラスラベルを $d_1, d_2, \cdots, d_L$ とする．ただし，各 d_l は $\boldsymbol{x}_l$ が文字「A」であれば +1 の値をとり，そうでなければ −1 の値をとるものとする．このとき上

[*1] サポートベクターマシンやサポートベクタマシンと表記されることもあるが，vector の訳語として「ベクトル」が定着しているので本稿ではサポートベクトルマシンと表記する．

述の問題は，$\{(\boldsymbol{x}_l, d_l)\}_{l=1}^{L}$ を基に，任意の n 次元実ベクトル $\boldsymbol{x}$ に対して $+1$ か -1 のいずれかを出力する 2 値関数（識別関数という）をいかに構成するかということになる．以下では $\{(\boldsymbol{x}_l, d_l)\}_{l=1}^{L}$ を訓練サンプルという（$\{\boldsymbol{x}_l\}_{l=1}^{L}$ を訓練サンプルということもある）．

2.2 最適分離超平面とサポートベクトル

はじめに最も単純な識別器として，識別関数が次式で表される線形識別器を考えよう．

$$y(\boldsymbol{x}) = \mathrm{sgn}\left(\boldsymbol{w}^T\boldsymbol{x} + b\right) \tag{1}$$

ここで $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]^T$ であり，$\mathrm{sgn}(u)$ は，$u \geq 0$ のときに 1 を，$u < 0$ のときに 0 をとる符号関数である．線形識別器 (1) の出力は超平面

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0 \tag{2}$$

を境に変わるので，これを識別器 (1) の識別面という．さて，訓練サンプル $\{(\boldsymbol{x}_l, d_l)\}_{l=1}^{L}$ が与えられたとき，パラメータ $\boldsymbol{w}, b$ をどのように定めればよいだろうか．以下では当分の間，議論を簡単にするため，2 つの集合 $X^+ = \{\boldsymbol{x}_l \,|\, d_l = 1\}$, $X^- = \{\boldsymbol{x}_l \,|\, d_l = -1\}$ は線形分離可能であると仮定する（図 1 参照）．このとき，$l = 1, 2, \ldots, L$ に対して $y(\boldsymbol{x}_l) = d_l$ を満たす，すなわち，訓練サンプルを誤りなく識別する線形識別器 (1) が無数に存在する（図 1 において白丸と黒丸を分ける直線は無数に存在する）．パターン識別の目標は訓練サンプルだけでなく未知のパターンを正しく識別することであるから，無数の線形識別器の中から未知のパターンに対する識別率が最も高くなりそうなものを選ばねばならない．一つの候補は識別面が X^+ と X^- のちょうど真ん中を通るものである．これを最適分離超平面という．未知のパターンが訓練サンプルと同じような現れ方をすると仮定すれば，識別面が最適分離超平面と一致するような識別器を選ぶのは自然な考え方であろう．

X^+ と X^- を分離する任意の超平面 $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ に対して，それから最も近い訓練サンプルまでの距離

$$\rho(\boldsymbol{w}, b) = \min_{1 \leq l \leq L} \frac{|\boldsymbol{w}^T\boldsymbol{x}_l + b|}{\|\boldsymbol{w}\|^2} \tag{3}$$

をその超平面のマージンと定義する．最適分離超平面はマージンを最大にする分離超平面であるから，X^+ と X^- を分離するという制約の下でマージンを最大化すれば最適分離超平面が求められる．しかしながら，これでは $\boldsymbol{w}$ と b が一意に決まらないので，最も近い訓練サンプルに対して $|\boldsymbol{w}^T\boldsymbol{x}_l + b| = 1$ が成り立つという制約を追加する．これにより，最適分離超平面を求める問題は次の形に定式化される．

問題 1 制約条件

$$d_l(\boldsymbol{w}^T\boldsymbol{x}_l + b) \geq 1, \; l = 1, 2, \ldots, L$$

の下で目的関数 $\|\boldsymbol{w}\|^2/2$ を最小にする $\boldsymbol{w}, b$ を求めよ．

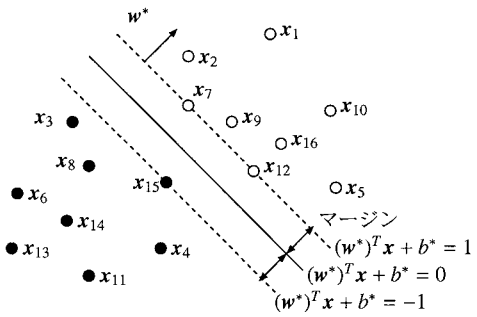

図 1　最適分離超平面とサポートベクトル．○と●はそれぞれ $d_l = 1$, $d_l = -1$ であることを表す．この図では $\boldsymbol{x}_7, \boldsymbol{x}_{12}, \boldsymbol{x}_{15}$ がサポートベクトルである．

問題 1 の最適解を $(\boldsymbol{w}^*, b^*)$ とすると，制約条件を等式で満足するパターンが X^+, X^- のそれぞれに少なくとも一つずつ存在する（図 1 の例では $\boldsymbol{x}_7, \boldsymbol{x}_{12}, \boldsymbol{x}_{15}$）．これらは最適分離超平面に最も近いパターンであり，最適分離超平面をサポート（支持）していることからサポートベクトルとよばれる．これがサポートベクトルマシンという名称の由来である．

2.3 ソフトマージン

マージン最大化という考え方は X^+ と X^- が線形分離可能でない訓練サンプルにも容易に拡張できる．この場合，問題 1 には許容解が存在しないので，目的関数と制約条件を次のように修正する．

問題 2 制約条件

$$d_l(\boldsymbol{w}^T\boldsymbol{x}_l + b) \geq 1 - \xi_l, \; l = 1, 2, \ldots, L$$
$$\xi_l \geq 0, \; l = 1, 2, \ldots, L$$

の下で目的関数

$$\|\boldsymbol{w}\|^2/2 + C\sum_{l=1}^{L}\xi_l$$

を最小にする $\boldsymbol{w}, b, \boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_L]^T$ を求めよ．

ここで ξ_l は訓練サンプル $\boldsymbol{x}_l$ に対する識別誤りの度合いを表す変数であり（図 2 参照），C は正の定数である．変数 ξ_l の導入により問題 2 では許容解の存在が保証される．その上で目的関数に $C\sum_{l=1}^{L}\xi_l$ という項が加わっているため，識別誤りの度合いは必然的に小さくなる．この手法はソフトマージンとよばれている．尚，X^+ と X^- が線形分離可能な場合，問題 2 の最適解は $\boldsymbol{\xi}^* = \boldsymbol{0}$ を満たすので，問題 1 の最適解と一致する．

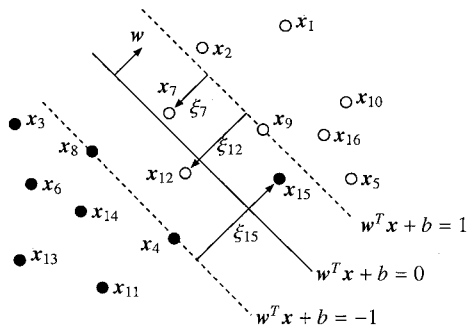

図 2　ソフトマージン．超平面 $w^T x + b = 0$ を定めると各 ξ_i の値が決まる（この図の例では $\xi_7, \xi_{12}, \xi_{15}$ 以外はすべて 0）．問題 2 では $\|w\|^2 + C\sum_{l=1}^{L}\xi_l$ を最小にする超平面を求めなければならない．

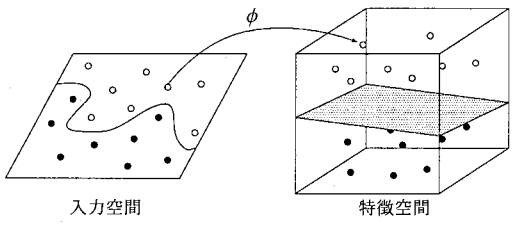

図 3　高次元空間への写像．写像 ϕ が非線形であれば特徴空間における分離超平面は入力空間では非線形の分離曲面に相当する．

2.4　高次元空間への写像とカーネル関数

マージン最大化やソフトマージンについて述べてきたが，これらを駆使したところで (1) は線形識別器であるから限界がある．そこで入力パターン $\boldsymbol{x}$ を関数 ϕ によって高次元空間（特徴空間という）に写し，その空間内で上述の手法を適用することが考えられる（図 3 参照）．このとき関数 ϕ は非線形でなければならない．また，パターンの個数が一定ならば空間の次元が高いほど線形分離できる確率が高くなるという Cover の定理 [10] によれば，ϕ の次元はできるだけ高い方が望ましい．

特徴空間で最適分離超平面を求めるには，問題 2 において $\boldsymbol{x}_l$ を $\phi(\boldsymbol{x}_l)$ で置き換えたものを解けばよい．しかしながら，この方法では変数の個数が ϕ の次元（特徴空間の次元）$+L+1$ となるため ϕ の次元を高くするにしたがって最適化問題の規模も大きくなってしまう．そこで双対問題とカーネル関数を利用してこの問題を回避する．双対問題とは元の最適化問題（主問題という）から得られる別の形の最適化問題のことであり，両者の間には密接な関係がある [11]．特に問題 1, 2 のような凸 2 次計画問題では，主問題の最適解に対する目的関数値と双対問題の最適解に対する目的関数値が一致し，双対問題の最適解から主問題の最適解を構成することが可能になる．問題 2 の双対問題を以下に示す．

問題 3 制約条件

$$\sum_{l=1}^{L} d_l\alpha_l = 0, \quad 0 \le \alpha_l \le C,\ l = 1,2,\ldots,L$$

の下で目的関数

$$\sum_{l=1}^{L}\alpha_l - \frac{1}{2}\sum_{k,l=1}^{L} d_k d_l \alpha_k \alpha_l \boldsymbol{x}_k^T \boldsymbol{x}_l$$

を最大にする $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_L]^T$ を求めよ．

問題 2 に対する Karush-Kuhn-Tucker (KKT) 条件を整理すると，問題 2 の最適解 (w^*, b^*, ξ^*) と問題 3 の最適解 α^* の間の関係として

$$w^* = \sum_{l=1}^{L}\alpha_l^* d_l \boldsymbol{x}_l \tag{4}$$

$$d_l(w^{*T}\boldsymbol{x}_l + b^*)\begin{cases} \ge 1, & \alpha_l^* = 0 \text{ のとき} \\ = 1, & 0 < \alpha_l^* < C \text{ のとき} \\ \le 1, & \alpha_l^* = C \text{ のとき} \end{cases} \tag{5}$$

が得られる．式 (5) から b^* を求めるには，l_0 を $0 < \alpha_l^* < C$ を満たす任意の l として，

$$b^* = d_{l_0} - w^{*T}\boldsymbol{x}_{l_0} = d_{l_0} - \sum_{l=1}^{L}\alpha_l^* d_l \boldsymbol{x}_l^T \boldsymbol{x}_{l_0} \tag{6}$$

とすればよい．以上のことから，最適分離超平面を問題 3 の最適解 α^* を用いて表すと次式のようになる．

$$w^{*T}\boldsymbol{x} + b^* = \sum_{l=1}^{L}\alpha_l^* d_l \boldsymbol{x}_l^T \boldsymbol{x} + d_{l_0} - \sum_{l=1}^{L}\alpha_l^* d_l \boldsymbol{x}_l^T \boldsymbol{x}_{l_0} = 0 \tag{7}$$

ここで問題 3 の $\boldsymbol{x}_l$ をすべて $\phi(\boldsymbol{x}_l)$ に置き換えることを考えよう．問題 3 の変数の個数は訓練サンプル数 L に等しく $\boldsymbol{x}_l$ の次元には依存しないので，$\boldsymbol{x}_l$ を $\phi(\boldsymbol{x}_l)$ に置き換えても問題の規模は変わらない．ただ，特徴空間の次元が高くなるにつれて $\phi(\boldsymbol{x}_l)$ の計算時間が長くなってしまうという問題がある．ところが，問題 3 においても最適分離超平面の式 (7) においても $\boldsymbol{x}$ は内積の形でしか表れないことに注意すると，非線形写像 ϕ を陽に扱う必要はなく，その内積値 $\phi^T(\boldsymbol{x}_k)\phi(\boldsymbol{x}_l)$ だけを計算できればよいことがわかる．いよいよカーネル関数の登場である．

カーネル関数 $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ とは二つの n 次元ベクトルを引数にもち，次式のように表される関数である．

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \phi^T(\boldsymbol{x})\phi(\tilde{\boldsymbol{x}}) \tag{8}$$

カーネル関数であるための必要十分条件は Mercer の定理 [1] に与えられている．また，その必要十分条件を満

足する関数として次の2つがよく知られている．

$$K_{\mathrm{G}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \exp\left(-\frac{\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|^2}{2\sigma^2}\right) \tag{9}$$

$$K_{\mathrm{P}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = (1+\boldsymbol{x}^T\tilde{\boldsymbol{x}})^p \tag{10}$$

式 (9), (10) はそれぞれガウシアンカーネル，多項式カーネルとよばれている．ただし σ は任意の正数，p は任意の自然数である．特に多項式カーネルについては簡単な計算によって関数 ϕ を求めることができる．例えば $p=2$ の場合 $\phi(\boldsymbol{x})$ =[1, $\sqrt{2}x_1,\ldots,\sqrt{2}x_n$, $x_1^2,\ldots,x_n^2$, $\sqrt{2}x_1x_2,\ldots,\sqrt{2}x_{n-1}x_n]^T$ となり，$\phi(\boldsymbol{x})$ は n 次元空間から $(n^2+n+2)/2$ 次元空間への写像であることがわかる．一方，ガウシアンカーネルの場合，ϕ を陽な形で表せないことや ϕ は無限次元であることが知られている．カーネル関数を利用すれば写像 ϕ を陽に扱うことなく特徴空間における内積 $\phi^T(\boldsymbol{x})\phi(\tilde{\boldsymbol{x}})$ を計算することができる．この手法をカーネルトリックという．ガウシアンカーネルの場合，特徴空間は無限次元であるので，無限次元ベクトルの内積を計算していることになる．

以上より特徴空間における最適分離超平面を求める問題は次のように定式化される．

問題 4 制約条件

$$\sum_{l=1}^{L} d_l\alpha_l = 0, \quad 0 \le \alpha_l \le C,\ l=1,2,\ldots,L$$

の下で目的関数

$$\sum_{l=1}^{L}\alpha_l - \frac{1}{2}\sum_{k,l=1}^{L} d_kd_l\alpha_k\alpha_lK(\boldsymbol{x}_k,\boldsymbol{x}_l)$$

を最大にする $\alpha_1,\alpha_2,\ldots,\alpha_L$ を求めよ．

また，問題 4 の最適解を $\boldsymbol{\alpha}^*$ とすると，式 (7) より最適分離超平面は次式で与えられる．

$$\sum_{l=1}^{L}\alpha_l^*d_lK(\boldsymbol{x}_l,\boldsymbol{x}) + d_{l_0} - \sum_{l=1}^{L}\alpha_l^*d_lK(\boldsymbol{x}_l,\boldsymbol{x}_{l_0}) = 0 \tag{11}$$

この式には形式的にすべての訓練サンプルが含まれているが，実際には $\boldsymbol{\alpha}^*$ の要素の多くが 0 となる（図 2 と式 (5) を参照）ので，最適分離超平面は訓練サンプルのごく一部によって表現されることに注意する．

SVM は問題 4 を解いて得られる最適分離超平面 (11) を識別面とするパターン識別器である．線形識別器を出発点として，それにマージン最大化，ソフトマージン，カーネルトリックといった巧妙な手法が組み合わされて完成された識別器であることがお分かり頂けたかと思う．SVM の学習は凸 2 次計画問題に帰着されるので，多層パーセプトロンの学習に見られるような局所最適解の問題が存在しない．また，問題 4 を解くことによって簡潔な識別関数が自動的に得られる．このことは，多層パーセプトロンや RBF ネットワークにおける中間層の素子数が自動的に決まることに相当する．パターンを高次元の特徴空間に写像することにより入力空間での識別面の表現力を高める一方，特徴空間ではマージン最大化によってできるだけ簡潔な超平面を求めようとする．SVM の高い汎化能力（未知パターンに対する識別能力）はこの両者のバランスによって実現されているといえる．一方，問題点としては，訓練サンプルが多くなると凸 2 次計画問題の規模が大きくなり学習に長い時間を要することや，カーネル関数 K および定数 C の設定法が難しいことなどが挙げられる．

3 SVM の効率的学習アルゴリズム

3.1 分割法

SVM を計算機上に実装するためには，2 次計画問題を解くプログラムが必要である．2 次計画問題は最も基本的な最適化問題の一つであり，これまでに数多くの数値解法が提案されている．また，それらの一部は各種ソフトウェアにも組みこまれている[*2]．SVM を実装する一つの方法はこれらの関数を利用することである．しかしながら，大量の訓練サンプルを扱う場合には解くべき 2 次計画問題の規模が大きくなり，従来の方法では学習に膨大な時間を要してしまう．それどころか最悪の場合，目的関数の係数（2 次の項で対称性を考慮しても $L(L+1)/2$ 個ある）を計算機のメモリに蓄えることができず，学習不可能という状況に陥ってしまう．これらの問題を解決するために，最近，分割法（Decomposition Method）とよばれる SVM の効率的な学習アルゴリズムがいくつか提案されている [7]–[9].

分割法は，問題 4 の任意の許容解（たとえば $\boldsymbol{\alpha}=\boldsymbol{0}$）を初期値とし，1) L 個の変数の中から更新対象となる q 個の変数（Working Set とよばれる）を適当な方法で選択する，2) 選択された変数だけに関する 2 次計画問題（部分最適化問題という）を解いてそれらの値を更新する，という操作を反復的に実行して元の問題の最適解を求める方法の総称である．変数 $\boldsymbol{\alpha}$ の現在の値を $\bar{\boldsymbol{\alpha}}$ で表し，簡単のため $\alpha_1,\alpha_2,\ldots,\alpha_q$ を更新対象変数とすると，部分最適化問題は次のようになる．

問題 5 制約条件

$$\sum_{l=1}^{q} d_l\alpha_l = -\sum_{l=q+1}^{L} d_l\bar{\alpha}_l$$
$$0 \le \alpha_l \le C, \quad l=1,2,\ldots,q$$

[*2] 例えば Matlab には quadprog という関数が用意されており，目的関数や制約条件の係数を関数の引数に与えるだけで簡単に最適解を得ることができる．

の下で目的関数

$$\sum_{l=1}^{q}(1-d_l\sum_{k=q+1}^{L}d_k\bar{\alpha}_k K(\boldsymbol{x}_k,\boldsymbol{x}_l))\alpha_l - \frac{1}{2}\sum_{k,l=1}^{q}d_k d_l\alpha_k\alpha_l K(\boldsymbol{x}_k,\boldsymbol{x}_l) \quad (12)$$

を最大にする $\alpha_1,\alpha_2,\ldots,\alpha_q$ を求めよ.

一般に q の値は L に比べて十分小さい（2 から数十の範囲）ので，部分最適化問題はごく短時間で解くことができる．また，目的関数の係数が 1 次と 2 次を合わせて $q+q(q+1)/2$ 個しかないので，計算機のメモリ消費量も少ない（その都度カーネルの計算が必要であるが）．ただし，分割法の計算時間や反復回数は Working Set の選択法に大きく依存するので注意が必要である．極端な例として，毎回同じ変数を選択すれば，2 回目の反復以降目的関数の値は変化せず分割法はいつまで経っても終了しない．変数の更新によって目的関数が最も増加するような Working Set を選択するのが望ましいが，そのために大量の計算が必要になっては意味がない．したがって，目的関数を大きく増加させるような Working Set をなるべく少ない手間で選択することが重要となる．

代表的な分割法として SMO (Sequential Minimal Optimiation) アルゴリズム [7]，LIBSVM [8], SVM$^{\text{light}}$ [9] があり，これらは Working Set の選択法や部分最適化問題の解法によって特徴づけられる．終了条件については，いずれのアルゴリズムも問題 4 に対する KKT 条件を少しだけ緩めたものを採用している．問題 4 に対する KKT 条件は次式のように簡潔な形で表現される [12].

$$\min_{i\in I_1(\alpha)} F_i(\alpha) \geq \max_{i\in I_2(\alpha)} F_i(\alpha) \quad (13)$$

ただし，$F_i(\alpha), I_1(\alpha), I_2(\alpha)$ の定義は以下の通りである.

$$F_i(\alpha)=\sum_{l=1}^{L}\alpha_l d_l K(\boldsymbol{x}_l,\boldsymbol{x}_i)-d_l$$
$$I_1(\alpha)=\{i\,|\,\alpha_i<C,d_i=1\}\cup\{i\,|\,\alpha_i>0,d_i=-1\}$$
$$I_2(\alpha)=\{i\,|\,\alpha_i<C,d_i=-1\}\cup\{i\,|\,\alpha_i>0,d_i=1\}$$

SMO アルゴリズムはその名の通り Working Set のサイズが最小（$q=2$）の分割法である．ヒューリスティックな方法で 2 個の変数を選択し，それらに関する部分最適化問題を解いて変数の値を更新していく．部分最適化問題は最適解が解析的に求まるので，2 次計画問題の数値解法が必要なく実装が容易であるという利点がある．終了条件には (13) を緩めた次の不等式が実質的に用いられている.

$$\min_{i\in I_1(\alpha)} F_i(\alpha) \geq \max_{i\in I_2(\alpha)} F_i(\alpha)-\tau \quad (14)$$

ここで τ は十分小さい正の定数である.

LIBSVM は，SMO と同じく $q=2$ の分割法であり，部分最適化問題の解法や終了条件は同じであるが，Working Sct の選択法が異なっている．バージョン 2.71 以前では，終了条件 (14) に最も大きく違反している 2 変数，すなわち，$I_1(\alpha)$ に属する i の中で $F_i(\alpha)$ が最大のものと $I_2(\alpha)$ に属する i の中で $F_i(\alpha)$ が最小のものが Working Set に選ばれる．実はこれは問題 4 の目的関数が最も急に増加するような 2 変数の選び方になっている．一方，バージョン 2.8 以降では，目的関数の 2 階微分までを考慮した Working Set の選択法が採用されており，更なる高速化が実現されている.

SVM$^{\text{light}}$ は，q の値を L 以下の任意の偶数に設定できる分割法であり，部分最適化問題は既存の数値解法を用いて解かれる．また，終了条件には SMO, LIBSVM と同じく (14) が用いられている[*3]．Working Set の選択は LIBSVM の拡張となっており，$I_1(\alpha)$ に属する i の中で $F_i(\alpha)$ の大きい方から $q/2$ 個，$I_2(\alpha)$ に属する i の中で $F_i(\alpha)$ の小さい方から $q/2$ 個が選ばれる．これは問題 4 の目的関数が最も急に増加するような q 変数の選び方になっている.

ここで述べた方法をはじめとする数多くのプログラムがインターネット上に公開されている[*4]ので，SVM をとにかく試してみたいという人は，これらの一つをダウンロードして利用されたい.

3.2 分割法の大域収束性

分割法の定義から明らかなように問題 4 の目的関数の値は反復回数とともに増加（厳密には非減少）し，また許容領域が有界であるから上に有界である．したがって目的関数の値は必ずある値に収束する．しかしながら，その収束先が最大値である保証はない．例えば前述のように Working Set に毎回同じ変数の組を選んでいては最大値に到達しない．分割法によって得られる解の列が最適解に収束するための条件は何か．これを明らかにすることは理論と実用の両面において基本的かつ重要な課題であり，これまでに多くの研究成果が発表されている [12]–[14]．以下ではその中からいくつかを紹介する.

Keerthi ら [12] は，$q=2$ の分割法の一般形である一般化 SMO アルゴリズムの挙動を詳細に解析し，終了条件に (14) を採用した場合の収束性に関して以下の定理を与えた.

定理 1 ([12]) 各反復において，終了条件 (14) に違反する 2 変数，すなわち，$i\in I_1(\alpha), j\in I_2(\alpha), F_i(\alpha)<F_j(\alpha)-\tau$ を満たす α_i と α_j を Working Set に選ぶならば，任意の $\tau>0$ に対して分割法は有限回の反復で終了する.

[*3] SVM$^{\text{light}}$ のソースコードでは $0(C)$ に十分近い変数を $0(C)$ に置き換える操作が入っているため厳密に言えば (14) と異なる.

[*4] http://www.kernel-machines.org/

Keerthi らは，まず，一般化 SMO アルゴリズムによって得られる解の列が収束することを示し，次に，背理法によってこのアルゴリズムが有限回の反復で終了することを証明した．ただし，Keerthi らのオリジナルの証明は最後のステップが不完全であるので注意されたい．完全な証明は文献 [13] に与えられている．

Takahashi ら [14] は，最近，一般的な分割法の収束性について考察し，終了条件として (14) の代わりに

$$\min_{i\in I_1^\delta(\alpha)} F_i(\alpha) > \max_{i\in I_2^\delta(\alpha)} F_i(\alpha) - \tau \tag{15}$$

を用いれば収束性が保証されることを証明した．ただし，

$$I_1^\delta(\alpha) = \{i \mid \alpha_i \le C-\delta, d_i = 1\} \cup \{i \mid \alpha_i \ge \delta, d_i = -1\}$$
$$I_2^\delta(\alpha) = \{i \mid \alpha_i \le C-\delta, d_i = -1\} \cup \{i \mid \alpha_i \ge \delta, d_i = 1\}$$

であり，δ は $C/2$ より小さい任意の正定数である．

定理 2 ([14]) 各反復において，終了条件 (15) に違反する 2 変数，すなわち，$i \in I_1^\delta(\alpha)$, $j \in I_2^\delta(\alpha)$, $F_i(\alpha) \le F_j(\alpha) - \tau$ を満たす α_i と α_j が Working Set に含まれるならば，任意の $\tau > 0, \delta \in (0, C/2)$ に対して分割法は有限回の反復で終了する．

定理 2 の証明は，最適化理論において古くから知られている大域収束定理 [15] に基づいている．詳細は省略するが，終了条件として (14) ではなく (15) が採用されているのは，この大域収束定理を適用するためである．式 (15) が等号なしの不等式であることや，$I_1^\delta(\alpha) \subseteq I_1(\alpha)$, $I_2^\delta(\alpha) \subseteq I_2(\alpha)$ が成り立つことに注意されたい．通常 τ や δ には十分小さい正定数が選ばれるので，(14) と (15) の違いは無視できるほど小さいもののように思われるかも知れないが，この小さな違いが定理 2 で本質的な役割を演じている．

4 おわりに

SVM の基本原理と効率的学習アルゴリズムとして広く用いられている分割法を紹介した．SVM の背景には，VC 次元や構造的損失最小化など，統計的学習理論における Vapnik の成果 [1] が数多くあるが，ここでは取り上げなかった．本稿を読んで学習やパターン認識に興味を持たれた方には，成書 [16]–[19] を読まれることをお薦めする．また，カーネル法はそれ自体で発展を続けており，最近では木構造やグラフ構造をもったデータに対してカーネル法を適用する試みもなされている [20]．カーネル法に関しては文献 [21] に詳しく述べられている．

参考文献

[1] V. Vapnik, *Statistical Learning Theory*, New York: Wiley, 1998.

[2] B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 1999.

[3] N. Cristianini and J. Shawa-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[4] 津田宏治, "サポートベクターマシンとは何か," 電子情報通信学会誌, vol.83, no.6, pp.460–466, June 2000.

[5] 赤穂昭太郎, 津田宏治, "サポートベクターマシン: 基本的仕組みと最近の発展, " 数理科学, no.444, pp.52–58, June 2000.

[6] 前田英作, "痛快!サポートベクトルマシン –古くて新しいパターン認識手法–," 情報処理学会誌, vol.42, no.7, pp.676–683, July. 2001.

[7] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Machines*, B. Schölkopf, C. Burges, and A. Smola, Eds., MIT Press, 1998.

[8] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[9] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods: Support Vector Machines*, B. Schölkopf, C. Burges, and A. Smola, Eds., MIT Press, 1998.

[10] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. Electronic Computers*, vol.14, pp.326–334, 1965.

[11] 福島雅夫, 非線形最適化の基礎, 朝倉書店, 2001.

[12] S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, pp.351–360, 2002.

[13] N. Takahashi and T. Nishi, "Rigorous proof of termination of SMO algorithm for support vector machines," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 774–776, May 2005.

[14] N. Takahashi and T. Nishi, "Global convergence of decomposition learning methods for support vector machines," *IEEE Trans. Neural Networks*, vol. 17, no. 6, pp.1362–1369, November 2006.

[15] W. I. Zangwill, *Nonlinear Programming: A Unified Approach*, Prentice-Hall, 1969.

[16] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern classification*, John Wiley & Sons, 2001.

[17] 渡辺澄夫, データ学習アルゴリズム, 共立出版, 2001.

[18] 麻生英樹, 津田宏治, 村田昇, パターン認識と学習の統計学, 岩波書店, 2003.

[19] 渡辺澄夫, 萩原克幸, 赤穂昭太郎, 本村陽一, 福水健次, 岡田真人, 青柳美輝, 学習システムの理論と実現, 森北出版, 2005.

[20] 鹿島久嗣, "カーネル法による構造データマイニング," 情報処理学会誌, vol.46, no.1, pp.27–33, Jan. 2005.

[21] B. Schölkopf and A. J. Smola, *Learning with kernels*, MIT Press, 2002.