サポートベクターマシンとカーネル回帰

石川 徹也　<tiskw111@gmail.com>

2020 年 3 月 20 日

はじめに

本文書の目的は，カーネルサポートベクターマシン（以下，K-SVM と略す）を理解することにあります．職場の同僚や友人から良く受ける質問のひとつに「K-SVM のハイパーパラメータはどのように調整すれば良いか？」というものがあります．彼らはグリッドサーチや直感（？！）などでハイパーパラメータを探索・調整しているのですが，「中身をきちんと理解していれば，手動でちょろっと調整方法することもできるのになぁ」と思うことがあります．もちろんグリッドサーチが悪いと言っているのでは決してありません．最終的に性能を出そうとすれば，ある程度の領域に目星を付けた上でグリッドサーチなどのパラメータ探索手法を使用するのが一般的です．しかしその「目星」を付けられるかどうかは非常に重要です．さもなければグリッドサーチの探索範囲が爆発し，実時間内に良いハイパーパラメータが探索できなくなってしまいます．また，SVM はその汎化性能を利用して「ちょっとお試しで」使うことも多いのですが，そういうときにもハイパーパラメータの目星が付けられないと，そういった使い方もしにくくなります．ならば同僚に解説文書を読ませてみようかな，と思ったのがこの文書を書くきっかけでした．

本節では，サポートベクターマシン（以下，SVM と略します）について，数式を出さずにラフにお話ししてみましょう．そもそも SVM とは，与えられた教師付き学習データから 2 クラスのパターン識別器を構成するための手法です．ここで強調したいのは，SVM の本質は回帰にある，という点です．つまり，SVM なんて小難しい名称をしていますが，結局は学習データを曲面で回帰し，得られた回帰曲線を判別器として利用する，という処理をしているだけに過ぎません．ちょっと乱暴かもしれませんが，言ってしまえば最小二乗法による点群の回帰とほとんど同じなのです．ただし，「回帰」ではなく「分類」として高い性能が出るよう，通常とはやや異なる特殊なやり方で回帰を行う点に，SVM の工夫があるだけです．

具体例でご説明しましょう．図 1 左に示すように，クラス A あるいは B に属する 2 次元の点群があったとします．ここでの目的は，図 1 右のようにクラス A と B をうまく識別できるような判別曲線を作成することとします．このような判別曲線を作っておけば，どちらのクラスに分類されるか分からない点が与えられたとき，この判別曲線を用いて自動的にクラス A か B かを判別できるためです．これを行うために，まずデータの

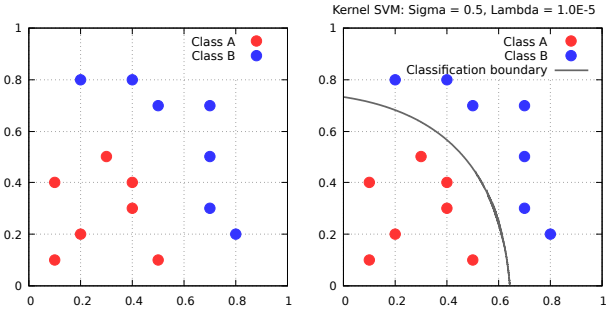

図 1　2 クラス分類の例と K-SVM による判別曲線

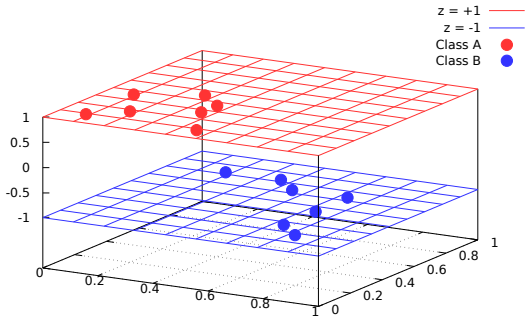

図 2　2 クラス分類のデータに z 軸を追加したところ

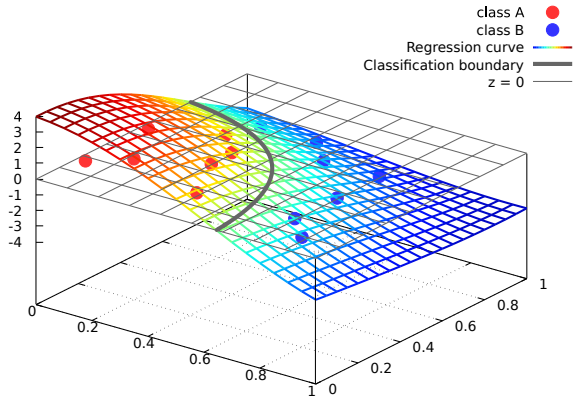

図 3　2 クラス分類のデータを曲面で「回帰」したところ

次元を 1 つ拡大し，3 次元にします．この拡大された 3 つ目の次元を，ここでは z 軸と呼ぶことにしましょう．この z 軸を何に使用するかというと，クラスを表現するために使用します．つまり，クラス A に属する点群は $z = +1$ に，クラス B に属する点群は $z = -1$ に設定します．このときデータ点群の全体は図 2 右のようになります．この拡大されたデータ点群を，曲面で回帰してみましょう．例えば図 3 のような曲面が得られたとします．先程，クラス A に属する点群は $z = +1$ に，クラ

ス B に属する点群は $z=-1$ に設定したことを思い出して下さい．したがって，クラス A の点が多い領域では回帰した曲面も $z=+1$ に近くなり，逆にクラス B が多い領域では回帰曲面は $z=-1$ に近づきます．そこで，この回帰曲面が正の値をとっている領域はクラス A に，ゼロおよび負の値をとっている領域はクラス B と予測してしまえば良いではないか，と考えるのは自然なことでしょう．SVM はまさにこれと同じことをしているのです．図 2 右の判別曲線は，図 3 の回帰曲面が $z=0$ を交差する場所をプロットしているだけに過ぎません．これが SVM の基本的なアイディアです．SVM の本質は回帰にある，と言ったのはこのためです．

さらに，回帰曲面としてどの関数を採用するかによって SVM の柔軟性は大きく変化します．SVM の最も基本的な形態は，回帰曲線に通常の平面（高次元なので正確には超平面）を用いた分類器であり，これがいわゆる通常の SVM です．本文書ではこれを特に線形 SVM と呼ぶことにします．そして回帰曲面として無限の自由度を持つ局曲面を採用したのがカーネルサポートベクターマシン（以下，K-SVM と略します）です．図 3 は実際に K-SVM で生成した曲面です．

無限の自由度，と聞くと戸惑う読者がおられるかもしれませんが，これは関数をパラメータに持つ回帰関数，という意味です．例えば，回帰関数のパラメータが N 次元の実ベクトルであれば，その回帰関数は実 N 次元の自由度を持ち得ますが，回帰関数のパラメータが関数であれば（すなわち汎関数），その回帰関数は実数では測り切れないほどの自由度を持ち得る，すなわち無限次元の自由度を持つと言えます．そんな汎関数を回帰関数に定めたのが，K-SVM なのです．

本文書では線形 SVM と K-SVM について解説します．特に K-SVM の解説では関数空間を全面に押し出した議論を試みています．読了後に「なーんだ，SVM は（K-SVM も含めて）ただの線形回帰だったのか」と思って頂けたのであれば，この文書は成功かなと思います．

SVM は Vapnik *et al.*[1] によって，K-SVM は Boser *et al.*[2] によってそれぞれ発表されました．1900 年代の前半には関数解析学がほぼ完成されていたことを考えると，カーネル法による SVM の拡張に約 30 年間もかかったという事実は，個人的にはやや驚きを感じます．おそらく実ベクトル空間（有限次元）から関数空間（無限次元）という思考の跳躍に時間がかかったのでしょう．そんな思考の跳躍を体感しながら本文書をお読み頂ければ，著者としてはこの上ない喜びです．

1 線形サポートベクターマシン

1.1 サポートベクターマシンのアイディア

いま，2 クラスのパターン識別器を構成するための教師付き学習データを

$$\mathcal{D} \triangleq \big\{(\boldsymbol{x}_i, y_i) \,\big|\, \boldsymbol{x}_i \in \mathbb{R}^m,\, y_i = \pm 1,\, i \in \mathcal{I}\big\}, \tag{1}$$

とおく．ただし $\mathcal{I}$ は要素数が n の添字集合，$\boldsymbol{x}_i$ は i 番目のデータの m 次元特徴ベクトル，y_i は i 番目のデータが属するクラスを表す．また，y_i が $+1$ であるデータの集合をクラス A，y_i が -1 であるデータの集合をクラス B と呼ぶことにしよう．この学習データを，回帰パラメータを $\boldsymbol{w}$ とする超平面 $y=\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$ で回帰し，その $\boldsymbol{w}$ を用いて分類器 $y=\mathrm{Sgn}(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x})$ を構成するのが SVM の基本方針である．ただし Sgn は

$$\mathrm{Sgn}\, x \triangleq \begin{cases} +1, & x \geq 0, \\ -1, & \text{else}, \end{cases} \tag{2}$$

によって定義される，符号関数に良く似た関数である．

> NOTE: 一般によく用いられる符号関数 $y=\mathrm{sgn}\, x$ を使用してしまうと，$x=0$ のときに $y=0$ となり，定義されない第 3 のクラスが生じてしまう．新たに符号関数 Sgn を定義したのは，そのような状況を避けるためだけであり，それ以上の意図は特にない．

1.2 最小 2 乗法による回帰は分類器には向いていない

問題は，学習データ $\mathcal{D}$ をどのように超平面 $y=\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$ で回帰するかである．悪例として，最小 2 乗法による回帰を考えてみよう．最小 2 乗法による回帰は

$$\min_{\boldsymbol{w}} \sum_{i\in\mathcal{I}} \big(y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i\big)^2, \tag{3}$$

と定式化される．しかし最小 2 乗法による回帰が，分類のための回帰として相応しくないことは，直感的にも明らかである．なぜならば，本来は分類にあまり影響を与えるべきでないはずの，識別境界から遠い点群の影響が強く出すぎてしまい，良い識別境界を得ることができない．

このことを定量的に議論してみよう．最小 2 乗法に限らず，どの回帰方法も回帰曲面と真値の差 $y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$ を何らかの関数で数値化し，これを小さくすることで回帰パラメータ $\boldsymbol{w}$ を決定するのが基本である．つまり，超平面による一般の回帰を数式で表現すると，V を適当なペナルティ関数として

$$\min_{\boldsymbol{w}} \sum_{i\in\mathcal{I}} V(y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i), \tag{4}$$

となる．最小 2 乗法は $V(x)=x^2$ の場合に相当する．さて，これから分類器にとって理想的な V の関数形を議論したいのだが，このままではいささか都合が悪い．というのも，$y_i=+1$ の場合と $y_i=-1$ の場合とで場合分けが避けられないためである．なぜならば，分類器を作成するという観点で見れば，$y_i=+1$ の場合は，$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i$ が正の大きな値，すなわち $y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i$ が負の大きな値である分には何の問題もない．分類は $y=\mathrm{Sgn}(\boldsymbol{c}^{\mathsf{T}}\boldsymbol{x})$ という式にしたがって行われるためである．むしろマージンが大きく取れているため，好ましいと言ってよい．しかし，$y_i=-1$ の場合は，$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i$ が負の大きな値，すなわち $y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i$ が正の大きな値である方が好ましい．このように，$y_i=+1$ の場合と $y_i=-1$ の場合とで，好ましい状況が反転しているため，このままだと 2 つのクラスを同時に議論するのが難しい．そこで，場合分けを避けるために $y_i - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i$ に y_i

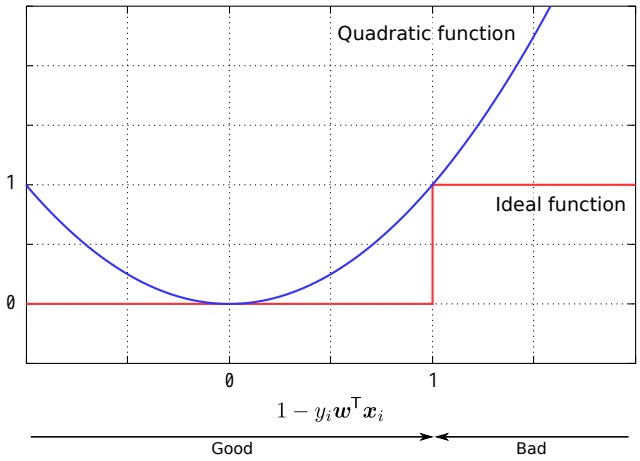

図 4　理想的なペナルティ関数と最小 2 乗法のペナルティ関数

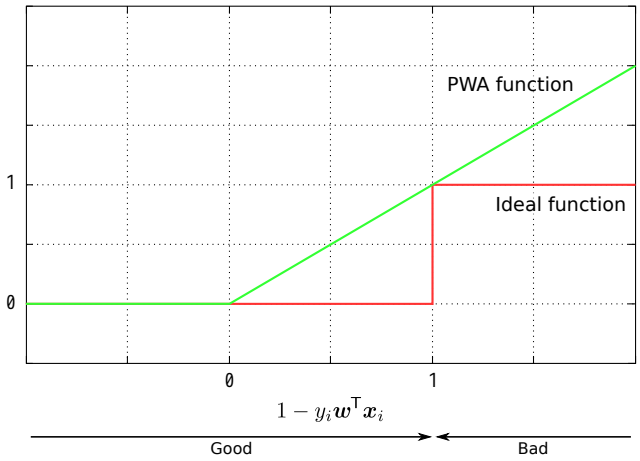

図 5　理想的なペナルティ関数と SVM で使われるペナルティ関数

を掛けあわせた式

$$y_i\big(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\big) = 1 - y_i \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i \tag{5}$$

を使うことにしよう．値 y_i を掛けることで，$y_i = -1$ の場合にのみ符号を反転させる効果が生じるため，前述の場合分けが見事に回避される．したがって，今後は超平面による一般の回帰を

$$\min_{\boldsymbol{w}} \sum_{i \in \mathcal{I}} V(1 - y_i \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i), \tag{6}$$

として話を進める．相変らず最小 2 乗法は $V(x) = x^2$ の場合に相当することに注意せよ．

> NOTE: そもそもクラスを表す値 y_i を ± 1 と置いたためにこの仕組みが上手く動作していることに注意されたい．つまり，クラスのラベル y_i を設定した段階からすでにこのカラクリが仕込まれていたのである．高々符号を合わせるための小細工と言ってしまえばそれまでだが，これを意図的に仕組んだ先人の閃きには流石と舌を巻かざるを得ない．

さて，回帰問題 (6) におけるペナルティ関数 V のうち，分類器を構成する上で最も適切なペナルティ関数は何か，という問題について考察しよう．分類器を構成する上で望ましいと考えられるやり方のひとつは，正しく分類されている学習データに対してはペナルティを与えず，正しく分類されていない学習データに対しては一律にペナルティ $+1$ を与えることである．このやり方に対応するペナルティ関数 $V = V_{\text{ideal}}$ を構成してみよう．分類は式 $y = \mathrm{Sgn}(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x})$ にしたがって行なわれることから，$\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i$ と y_i が同符号，すなわち $1 - y_i\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i$ が 1 よりも小さければ，そのデータは正しく分類されているので，ペナルティを加える必要はない．逆に，$\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i$ と y_i が異符号，すなわち $1 - y_i\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i$ が 1 よりも大きければ，ペナルティを加える必要がある．以上より，

$$V_{\text{ideal}}(x) \triangleq \begin{cases} 1, & x \geq 1, \\ 0, & \text{else}, \end{cases} \tag{7}$$

という関数が分類器を作成する上では理想的と言える．この理想的なペナルティ関数 V_{ideal} と，最小 2 乗法におけるペナルティ関数 $V_{\text{quadratic}}$ を図示したものが図 4 である．

最小 2 乗法におけるペナルティ関数 $V_{\text{quadratic}}$ は，理想的なペナルティ関数 V_{ideal} と比較して

- 十分に Good な領域にある点（図 4 における $1 - y_i\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i < 0$ の領域）に対してもペナルティを与えてしまう
- ペナルティの与え方が均等でなく，2 次関数的に急増する

という点で決定的に見劣りしており，分類に適した回帰とは言えないことが分かる．

1.3　理想と現実の妥協案

分類器を構成する上で，回帰問題 (6) のペナルティ関数 $V(x)$ を 2 次関数にするのは問題が多いことはすでに見た．しかし，ペナルティ関数 V を V_{ideal} にするのは，最適化の技術的な問題から現実的でない．なぜならば，回帰問題 (6) を $\boldsymbol{w}$ の最適化問題として捉えた場合，安定的に大域最適解を求めるためには最適化問題が凸最適化問題である必要があり，そのためにはペナルティ関数 V もまた凸関数である必要があるのだが，図 4 からすぐに分かるように，$V = V_{\text{ideal}}$ の場合はそれが成り立たない．そこで，理想的なペナルティ関数 V_{ideal} に近く，かつ凸関数となるような関数 V として，図 5 に示す区分的アファイン関数 (*piecewise affine function, PWA function*) を使うことにしよう．この区分的アファイン関数を用いて回帰を行うのが他ならぬ SVM なのである．

1.4　サポートベクターマシンの定式化

図 5 における区分的アファイン関数を V_{svm} とおくとしよう．このとき SVM は

$$y = \mathrm{Sgn}\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}\right), \tag{8}$$

$$\boldsymbol{w} = \operatorname*{argmin}_{\boldsymbol{w}} \sum_{i \in \mathcal{I}} V_{\text{svm}}\left(1 - y_i \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right), \tag{9}$$

と定式化される．以下，最適化問題 (9) が線形計画問題

$$\begin{aligned} &\min_{\boldsymbol{p}} \ \ \boldsymbol{c}^\mathsf{T}\boldsymbol{p}, \\ &\text{s.t.} \ \ \boldsymbol{A}\boldsymbol{p} \geq \boldsymbol{b}, \end{aligned} \tag{10}$$

に帰着されることを示す．ペナルティ関数 $V_{\mathrm{svm}}\left(1-y_i\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i\right)$ の値を ξ_i とおくと，$\xi_i = V_{\mathrm{svm}}\left(1-y_i\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i\right)$ は

$$\begin{aligned}\min_{\xi_i}\ & \xi_i, \\ \text{s.t.}\ & \xi_i \geq 1-y_i\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i, \\ & \xi_i \geq 0,\end{aligned} \tag{11}$$

と等価である．これは SVM に限らず区分的アファイン関数の最適化で頻繁に利用されるテクニックである．したがって最適化問題 (9) は

$$\begin{aligned}\min_{\xi_i, i\in\mathcal{I}}\ & \sum_{i\in\mathcal{I}}\xi_i, \\ \text{s.t.}\ & \xi_i \geq 1-y_i\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w}, \quad \text{for all } i\in\mathcal{I}, \\ & \xi_i \geq 0, \qquad\qquad\ \ \text{for all } i\in\mathcal{I},\end{aligned} \tag{12}$$

と変形できる．さらにここで

$$\boldsymbol{\xi} \triangleq \begin{pmatrix}\xi_1 \\ \vdots \\ \xi_n\end{pmatrix},\ \boldsymbol{p} \triangleq \begin{pmatrix}\boldsymbol{w} \\ \boldsymbol{\xi}\end{pmatrix},\ \boldsymbol{Q} \triangleq \begin{pmatrix}y_1\boldsymbol{x}_1^{\mathsf{T}} \\ \vdots \\ y_n\boldsymbol{x}_n^{\mathsf{T}}\end{pmatrix}, \tag{13}$$

$$\boldsymbol{A} \triangleq \begin{pmatrix}\boldsymbol{Q} & \boldsymbol{I}_n \\ \boldsymbol{O}_{n,m} & \boldsymbol{I}_n\end{pmatrix},\ \boldsymbol{b} \triangleq \begin{pmatrix}\boldsymbol{1}_n \\ \boldsymbol{0}_n\end{pmatrix},\ \boldsymbol{c} \triangleq \begin{pmatrix}\boldsymbol{0}_m \\ \boldsymbol{1}_n\end{pmatrix}, \tag{14}$$

とおけば，式 (10) に帰着される．ただし $\boldsymbol{0}_\alpha$ は α 次元の零ベクトル，$\boldsymbol{1}_\alpha$ はすべての要素が 1 の α 次元ベクトル，$\boldsymbol{I}_\alpha$ は α 次元の単位行列，$\boldsymbol{O}_{\alpha,\beta}$ は $\alpha\times\beta$ 次元の零行列である．

1.5 線形サポートベクターマシンの Python 実装

線形 SVM の実装としては C/C++ で実装された LIBSVM [*1] や LIBLINEAR [*2] が有名である．またこれらを Python から統一的なインタフェースで呼び出すことの出来るライブラリとして scikit-learn [*3] がある．応用上はこれらのパッケージを使うのが現実的であるが，試みに前節で紹介した定式化を Python で実装したものを本文書と同じレポジトリに同梱してある．必要に応じて参考にして頂きたい．ただし線形計画問題を解くパッケージとして Pulp を，グラフの描画に Matplotlib を使用した．これらのパッケージは以下のコマンドでインストールできる．

```
$ pip3 install pulp matplotlib
```

2 カーネル回帰と関数解析

2.1 カーネルサポートベクターマシンとカーネル回帰

線形 SVM では，回帰曲面を超平面 $y = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$ としたが，回帰曲線を一般の $f_w(\boldsymbol{x})$ とした場合，SVM は次のように一般化される．

$$y = \mathrm{Sgn}\left(f_w(\boldsymbol{x})\right), \tag{15}$$

$$w = \operatorname*{argmin}_w \sum_{i\in\mathcal{I}} V_{\mathrm{svm}}\left(y_i - f_w(\boldsymbol{x}_i)\right), \tag{16}$$

ただし w は回帰曲線のパラメータである．先に見た通り，式 (16) の最適化問題は

$$\begin{aligned}\min_{\xi_i, i\in\mathcal{I}}\ & \sum_{i\in\mathcal{I}}\xi_i, \\ \text{s.t.}\ & \xi_i \geq 1-y_i f_w(\boldsymbol{x}_i), \ \text{for all } i\in\mathcal{I}, \\ & \xi_i \geq 0, \qquad\qquad\ \ \text{for all } i\in\mathcal{I},\end{aligned} \tag{17}$$

と等価である．回帰曲線 f_w としてどのような曲線を用いるかによって SVM の柔軟性が大きく左右されるのだが，回帰曲線として無限大の自由度を持つカーネル回帰を採用したのが K-SVM である．カーネル回帰 (*kernel regression*) とは，適切に選ばれた関数の無数の足し合わせで回帰曲線を構成する手法であり，先に述べたとおり，無限の自由度を有する回帰手法である．本小節では，まずはこのカーネル回帰について見てみよう．

> NOTE: 「適切」とは，カーネル関数が対称かつ正定値になるように，という意味である．しかし本節ではそこまでは踏み込まないため，「厳密に任意ではないが，とりあえずガウス関数ならオッケー」程度に考えて頂ければ十分である．

簡単のため，x も $f_w(x)$ も 1 次元の実数変数であるとしよう．このとき，カーネル回帰は回帰曲線を

$$f_w(x) \triangleq \int_{-\infty}^{\infty} w(p)\phi(x,p)\,\mathrm{d}p, \tag{18}$$

とおく手法である．ただし f_w は回帰曲線，ϕ は任意に定められた関数，w はパラメータ関数であり，ϕ は事前に与えておく関数，パラメータ関数 w は与えられたデータ点群に最もよくフィットするものを最適化によって求めるものとする．関数 ϕ はどのように与えても良いが，頻繁に用いられる ϕ の例として，標準偏差 σ の正規分布

$$G_\sigma(x) \triangleq \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{19}$$

を x 方向に p だけ平行移動した関数が挙げられる．すなわち

$$\phi(x,p) = G_\sigma(x-p) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-p)^2}{2\sigma^2}\right), \tag{20}$$

である．このとき f_w は

$$f_w(x) = \int_{-\infty}^{\infty} w(p)G_\sigma(x-p)\,\mathrm{d}p, \tag{21}$$

となる．上式を見れば分かるように，正規分布を平行移動しつつ，重み w をかけながら足し込むというやり方で回帰曲線を構成している．この回帰は，後に説明するように，RBF カーネル (*radial basis function*) と呼ばれるものに相当する．以下では一般の ϕ に対して議論を進めていくが，具体性の欲しい読者は $\phi(x,p) = G_\sigma(x-p)$ として読み進めて構わない．

さて，回帰曲線の関数系が定まったところで，次は回帰曲線のパラメータ関数 w の決定の仕方について説明しよう．パラメータ関数 w の決定の仕方は様々な方法があるが，ここでは最も簡単な最小 2 乗法を紹介する．回帰のデータ点群を

$$\mathcal{D} \triangleq \left\{(x_i, y_i)\ \middle|\ x_i\in\mathbb{R},\, y_i\in\mathbb{R},\, i\in\mathcal{I}\right\}, \tag{22}$$

[*1] https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[*2] https://www.csie.ntu.edu.tw/~cjlin/liblinear/
[*3] https://scikit-learn.org/stable/

とすると，パラメータ関数 w は最適化問題

$$\min_{w} \sum_{i \in \mathcal{I}} \big(y_i - f_w(x_i)\big)^2 \tag{23}$$

を解くことで求めることが出来そうな気がするが，実は式 (23) はあまり好ましくない．注意すべき点として，式 (18) で定められる回帰曲線 f_w のパラメータは関数 w であるから，曲線 f_w は「関数 1 つ分」の自由度を持つ．これは実数に換算すると「無限個分」の自由度である．回帰に与えられるデータ点群は高々有限個しかないため，曲線 f_w はそれらのデータ点をすべて通ることができてしまい，さらにそのようなパラメータ関数 w は無数に存在する．これは次の 2 つの理由で好ましくない．1 つ目の理由は，単純に式 (23) が唯一解を持たない点である．唯一解を持たない最適化問題は，数学的に解きにくいだけでなく，応用上は最適解として得られた関数のうちどれが良いかをさらに選択しなければいけないという意味で，あまり便利とは言えない．そして 2 つ目の理由は，こちらの方がより重要なのであるが，汎化性能に欠ける曲線が出来上がってしまう点である．一般に回帰問題でよく言われるように，与えられたデータ点をすべて通る回帰曲線は好ましくない．回帰の目的の 1 つに予測がある．すなわち回帰で得られた曲線を予測に使用するケースが機械学習ではよくあるが，与えられたデータ点をすべて通ってしまうような曲線は，凹凸が激しく荒れた曲線になりやすいため，予測という観点からは役に立たないことが多い．この現象は過学習 (*over fitting*) と呼ばれており，与えられたデータ点に対して回帰曲線の自由度が高すぎる場合に発生することが知られている．通常の回帰問題，例えば線形回帰であれば，データ量が少ないからデータ点を増やす，あるいは回帰曲線自由度が高すぎるからモデル式の次数を下げるなどの対策が取られる．しかしカーネル回帰の場合，そもそもの自由度が無限大であるから，いくらデータ点を増やしても足りることはないし，モデル式の自由度を下げることも出来ない．そこでカーネル回帰では，式 (23) に正則化項，すなわちパラメータ関数 w が無駄に大きくならないように抑える効果を持つ項を付け加えた形

$$\min_{w} \sum_{i \in \mathcal{I}} \big(y_i - f_w(x_i)\big)^2 + \lambda \int_{-\infty}^{\infty} |w(p)|^2 \, \mathrm{d}p, \tag{24}$$

に修正することで，過学習の問題を解決している．ただし λ は正の実数パラメータである．これでめでたく無限の自由度を持つ回帰を実現することが出来るのだが，読者はまだ上式の第 2 項が正則化項に見えていないであろう．これを説明するためには関数解析に関する多少の知識が必要である．そこで，少しだけ関数解析の世界に寄り道をして，カーネル回帰に必要な知識をご説明しよう．

2.2　関数空間の導入

座標平面 $\mathbb{R}^2$ や座標空間 $\mathbb{R}^3$，一般に実 n 次元空間 $\mathbb{R}^n$ は，実に便利な数学的解析ツールである．機械設計をするときも，データ解析をするときも，経済学をするときでさえ，我々は実

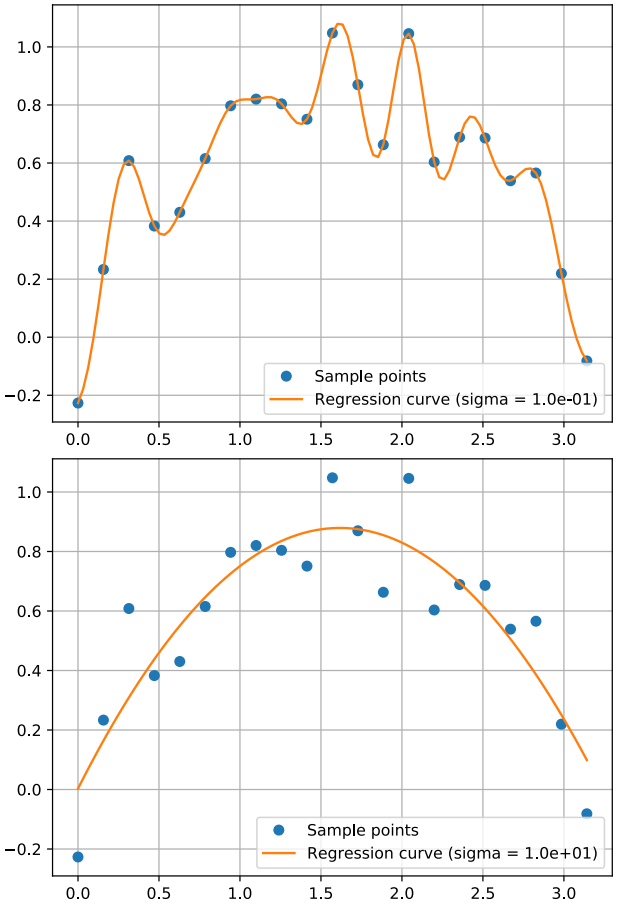

図 6　回帰問題において過学習した例（上）と過学習を避けることのできた例（下）

n 次元空間上で数理的な解析をすることが多い．なぜ実 n 次元空間は便利なのだろうか．理由は色々と考えられるであろうが，成分表示が出来るからでは決してあるまい．ベクトルを $\boldsymbol{x}$ などの一つの記号で置くことで実数の組であることを忘れ，和や定数倍，ノルム（絶対値），内積，外積などの各種ツールや概念を駆使することで解析をすることの価値を，我々はすでに知っている．先人らはその価値を，実数の組であるベクトル以外にも適用しようとした．すなわち，実数の組でない要素，例えば関数に対し，和や定数倍，ノルム，内積などを定義することで，関数をより扱いやすいものにしようと試みた．その成果が現在の Hilbert 空間である．本文書では，Hilbert 空間のひとつであり，関数解析の入り口でもある L^2 空間について紹介する．

関数 $f : \mathbb{R} \to \mathbb{R}$ に関する次の積分

$$\int_{-\infty}^{\infty} |f(x)|^2 \, \mathrm{d}x, \tag{25}$$

が有限確定の値を持つとき，関数 f は 2 乗可積分であると言われる．2 乗可積分な関数の集合を L^2 空間と定義しよう．すなわち

$$L^2 \triangleq \left\{ f : \mathbb{R} \to \mathbb{R} \;\middle|\; \int_{-\infty}^{\infty} |f(x)|^2 \, \mathrm{d}x < \infty \right\}, \tag{26}$$

である．積分区間はどのようにとっても構わないが，一度定めたらそれを一貫して使い続けなければならないことに注意せよ．

ここでは K-SVM への応用を見越して積分区間は $(-\infty, \infty)$ としている．さて，これから L^2 空間の要素である関数に対し，和，定数倍，ノルム，内積を定義するのだが，和と定数倍は通常の関数の和と定数倍でよいので割愛する．関数 $f, g \in L^2$ に対し，f と g の内積 $\langle f, g \rangle$ を

$$\langle f, g \rangle \triangleq \int_{-\infty}^{\infty} f(x)g(x)\,\mathrm{d}x, \tag{27}$$

と定義する．実ベクトル空間の場合と同じく，内積 $\langle f, g \rangle$ は実数値である．内積 $\langle f, g \rangle$ が 0 のとき，関数 f, g は直交すると言われる．関数 $f \in L^2$ のノルム $\|f\|$ を，f 同士の内積の平方根として定義する．すなわち

$$\|f\| \triangleq \sqrt{\langle f, f \rangle} = \sqrt{\int_{-\infty}^{\infty} |f(x)|^2\,\mathrm{d}x}, \tag{28}$$

である．L^2 空間の定義より，任意の関数 $f, g \in L^2$ に対して，$\langle f, g \rangle$ および $\|f\|$ が有限確定の値を持つことが直ちに証明できる．つまり $f, g \in L^2$ であれば $\langle f, g \rangle$ や $\|f\|$ を安心して使うことができる，ということだ．

さて，内積やノルムの性質をいくつか確認しておこう．まずは内積の性質を，定理という形でまとめておく．

定理 2.1 (関数空間 L^2 上の内積の性質)

以下，$f, g, h \in L^2$, $a, b \in \mathbb{R}$ とする．関数空間 L^2 上の内積 $\langle \cdot, \cdot \rangle$ について次が成り立つ：

- 可換性：$\langle f, g \rangle = \langle g, f \rangle$
- 双線形性：$\langle af + bg, h \rangle = a\langle f, h \rangle + b\langle g, h \rangle$
- シュワルツの不等式：$\langle f, g \rangle = \|f\|\,\|g\|$

証明： 可換性，双線形性はいずれも内積の定義より自明であるため割愛する．シュワルツの不等式（*Schwarz inequality*）は本文書には直接は関係がないため，証明は末尾の付録で行う．■

次はノルムの性質を，こちらも定理という形でまとめておく．

定理 2.2 (関数空間 L^2 上のノルムの性質)

以下，$f, g, h \in L^2$ とする．関数空間 L^2 上のノルム $\|\cdot\|$ について次が成り立つ：

- 正値性：$\|f\| \geq 0$
- 正定性：$\|f\| = 0 \iff f = \underline{0}$
- 三角不等式：$\|f + g\| \leq \|f\| + \|h\|$

証明： 正値性はノルムの定義から自明，正定性も定義よりほぼ自明であろう．三角不等式は本文書には直接関係しないため，証明は末尾の付録で行う．■

NOTE: 正定性って本当に自明か？と疑問に思った読者は良い直感をお持ちである．お気付きの通り，零関数とは異なるが積分すれば 0 になる関数などいくらでも存在する．例えばディリクレ関数（*Dirichlet function*）が良い例である．これは，著者が関数空間 L^2 における等号 $=$ を真面目に定義せずに先に進んでしまったことが原因なのだが，これは本文書の範囲内を越える．何でもかんでもひとつの文書に詰め込むのは著者の好みではない．したがって，この点は興味ある読者への愛ある課題として残しておく．詳細は成書を参照されたい．

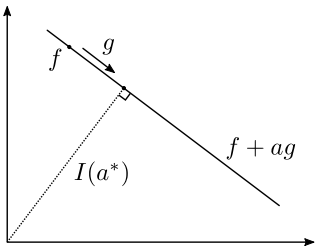

図 7　関数空間をベクトル空間のように表現したイメージ図

関数空間が実ベクトル空間と感覚的にはほぼ同じであるということを体感してもらうために，例題を 1 つ紹介する．

例題

実数 a をパラメータとする積分値

$$I(a) = \int_{-\infty}^{\infty} \left|f(x) + ag(x)\right|^2 \mathrm{d}x, \tag{29}$$

を最小化するパラメータ $a = a^*$ を求めよ．ただし関数 f, $g \in L^2$ とし，g は零関数ではないとする．

解 1：直接計算による．

$$I(a) = \|f + ag\|^2 = \|f\|^2 + 2a\langle f, g \rangle + a^2\|g\|^2,$$
$$\therefore\ a^* = -\frac{\langle f, g \rangle}{\|g\|^2}$$

■

解 2：定義より $I(a) = \|f + ag\|^2$ であるから，$I(a)$ は関数空間上の直線 $f + ag$ と原点との距離の 2 乗である．よって，図 7 より，この距離が最小になるのは，$f + ag$ と g が直交するときである．したがって

$$\langle f + a^*g, g \rangle = 0,$$
$$\langle f, g \rangle + a\|g\|^2 = 0,$$
$$\therefore\ a^* = -\frac{\langle f, g \rangle}{\|g\|^2}$$

■

これ以外にも，線形代数で学んだ定理や性質が，関数空間 L^2 についてもほとんど成り立つ．大学の線形代数の授業では，実ベクトルであるということを使わずにあえて抽象的な議論で定理や性質の証明を進めていたであろう．そのため，実ベクトルであれば成立するのはあたりまえジャン！とか思うような定理に対しても，逐一証明を与えていたはずである．これは，実ベクトル以外の対象を想定してのことであり，まさに今，その状況に出会ったという訳である．

2.3　カーネル回帰の定式化

さて，話をカーネル回帰に戻そう．カーネル回帰の回帰曲線 f_w は式 (18) で定義されるが，関数 $\phi(x, p)$ を分かり易さのために $\phi_x(p)$ と書き表すことにすれば，

$$f_w(x) = \langle w, \phi_x \rangle, \tag{30}$$

と表すことができる．同様に，パラメータ w は最適化問題 (24) によって得られるのであったが，これは

$$\min_{w\in L^2}\sum_{i\in\mathcal{I}}\big(y_i-\langle w,\phi_{x_i}\rangle\big)^2+\lambda\|w\|^2, \tag{31}$$

と表すことができる．上式は，w がベクトルではなく関数であるという点を除き，表面的には一般的な線形回帰と全く同じであることに注意されたい．これが関数解析を導入した恩恵のひとつである．このように書けただけでも解けそうな気がするではないか！

以下，最適化問題 (31) の具体的な解法を示す．ポイントは，求めるべき関数 w を一般性を失うことなく

$$w(p)=\sum_{j\in\mathcal{I}}a_j\phi_{x_j}(p), \tag{32}$$

と置くことができる点にある．ただし $a_j\in\mathbb{R}$ である．言い換えれば，関数 w は関数 $\phi_{x_j},\ j\in\mathcal{I}$ の 1 次結合で表すことができる．これにより，無限次元の最適化が有限次元の最適化に落ち，計算機で計算が実行できるようになるのである．これはレプレゼンター定理 (*Representer theorem*) として一般に良く知られている事実である．定理としてまとめておこう．

定理 2.3 (リプレゼンター定理)

最適化問題

$$\min_{w\in L^2}\sum_{i\in\mathcal{I}}V\big(y_i-\langle w,\phi_{x_i}\rangle\big)+\lambda\|w\|^2,$$

の解 w は関数 $\phi_{x_j},\ j\in\mathcal{I}$ の *1* 次結合で表すことができる．ただし $V:\mathbb{R}\to\mathbb{R}$ は任意の関数とする．

証明： 証明は帰謬法による．すなわち，仮に式 (32) が成立せず，余分な項 $\widehat{w}(p)$ が存在したとする．すなわち

$$w(p)=\sum_{j\in\mathcal{I}}a_j\phi_{x_j}(p)+\widehat{w}(p), \tag{33}$$

と表せたとする．このとき $\widehat{w}=\underline{0}$ が導かれることを示す．

まず，$\widehat{w}$ は $\sum_{j\in\mathcal{I}}a_j\phi_{x_j}$ では表現できない成分でなければならないから，ベクトルの一次独立性より

$$\langle\widehat{w},\phi_{x_1}\rangle=\langle\widehat{w},\phi_{x_2}\rangle=\cdots=\langle\widehat{w},\phi_{x_n}\rangle=0, \tag{34}$$

が成立する．もし仮に式 (34) が成立せず，例えば $\langle\widehat{w},\phi_{x_j}\rangle\neq 0$ であったとする．この場合，a_j を

$$a_j+\frac{\langle\widehat{w},\phi_{x_j}\rangle}{\|\phi_{x_j}\|}, \tag{35}$$

と置き換えることで，式 (34) を成立させることができる．これはグラム・シュミットの正規直交化法 (*Gram-Schmidt orthonormalization*) と全く同じ操作である．したがって，一般に式 (34) が成立するとして差し支えない．

最適化の目的関数の第 1 項目に式 (33) を代入し，さらに式 (34) を適用すると，

$$\begin{aligned}\sum_{i\in\mathcal{I}}V\big(y_i-\langle w,\phi_{x_i}\rangle\big)&=\sum_{i\in\mathcal{I}}V\Big(y_i-\Big\langle\sum_{j\in\mathcal{I}}a_j\phi_{x_j}+\widehat{w},\phi_{x_i}\Big\rangle\Big)\\&=\sum_{i\in\mathcal{I}}V\Big(y_i-\sum_{j\in\mathcal{I}}a_j\langle\phi_{x_j},\phi_{x_i}\rangle\Big),\end{aligned}$$

となる．上式に $\widehat{w}$ は一切登場しないこと，すなわち目的関数の第 1 項目は $\widehat{w}$ に無関係であることに注意せよ．その一方で正則化項 $\|w\|^2$ を計算してみると，式 (34) より，

$$\|w\|^2=\Big\|\sum_{j\in\mathcal{I}}a_j\phi_{x_j}\Big\|^2+\|\widehat{w}\|^2, \tag{36}$$

となる．したがって，最小化すべき目的関数に表れる $\widehat{w}$ は，$\|\widehat{w}\|^2$ だけであり，目的関数を大きくする効果しかない．よって $\widehat{w}=\underline{0}$ とするのが最適である．よって題意は成り立つ． ■

さて，式 (32) を最適化問題 (31) に代入してみよう．ここで

$$k_{ij}\triangleq\langle\phi_{x_i},\phi_{x_j}\rangle, \tag{37}$$

とおくと，最適化問題 (31) は

$$\min_{a_i\in\mathbb{R}}\sum_{i\in\mathcal{I}}\Big(y_i-\sum_{j\in\mathcal{I}}a_jk_{ij}\Big)^2+\lambda\sum_{j\in\mathcal{I}}\sum_{j\in\mathcal{I}}a_ik_{ij}a_j \tag{38}$$

となる．これを行列形式にまとめるために

$$\boldsymbol{y}=\begin{pmatrix}y_1\\\vdots\\y_N\end{pmatrix},\boldsymbol{a}=\begin{pmatrix}a_1\\\vdots\\a_N\end{pmatrix},\boldsymbol{K}=\begin{pmatrix}k_{11}&\cdots&k_{1N}\\\vdots&\ddots&\vdots\\k_{N1}&\cdots&k_{NN}\end{pmatrix}, \tag{39}$$

とおくと，式 (38) は

$$\min_{\boldsymbol{a}\in\mathbb{R}^N}\|\boldsymbol{y}-\boldsymbol{K}\boldsymbol{a}\|^2+\lambda\boldsymbol{a}^{\mathsf{T}}\boldsymbol{K}\boldsymbol{a}, \tag{40}$$

となる．ただし $N=\#\mathcal{I}$ である．式 (40) は $\boldsymbol{a}$ で微分することで直ちに解くことができて

$$\boldsymbol{a}=(\boldsymbol{K}+\lambda\boldsymbol{I})^{-1}\boldsymbol{y}, \tag{41}$$

である．このときの回帰曲線は

$$f_w(x)=\sum_{j\in\mathcal{I}}a_j\langle\phi_{x_j},\phi_x\rangle, \tag{42}$$

であり，特に $x=x_i$ のときは

$$f_w(x_i)=\sum_{j\in\mathcal{I}}k_{ij}a_j, \tag{43}$$

であることに注意せよ．

最後に，特徴関数 $\phi(x,p)$ が平行移動した正規分布 $G_\sigma(x-p)$ であるとき，カーネル関数 $\langle\phi_{x_j},\phi_x\rangle$ がどうなるかを見てみよう．とは言え，ただの単純計算なので詳細は省略する．結果的に

$$\langle\phi_{x_i},\phi_{x_j}\rangle\propto\exp\left(-\frac{|x_i-x_j|^2}{4\sigma^2}\right), \tag{44}$$

を得る．式 (44) の左辺と右辺の比例係数は無視してしまう場合が多い．なぜならば，最終的に最適化問題や回帰曲線の導出に特徴関数は登場せず，カーネル関数しか登場しない．よって特徴関数を適当にスカラー倍してやることで，カーネル関数の形をシンプルに保つことができるのである．式 (44) は一般に RBF カーネルと呼ばれ，カーネル回帰や K-SVM では最も頻繁に用いられるカーネル関数である．図 6 の回帰曲線は，実は RBF カーネルによるカーネル回帰の例である．図 6 の上図と下図の違いは，正規分布の標準偏差 σ の違いであり，上図は $\sigma = 0.1$，下図は $\sigma = 10$ である．カーネル回帰は特徴関数，すなわちここでは正規分布 $G_\sigma(x-p)$ を平行移動しながら足し込むという形で表現されていることを思い出せば，上図は「柔らかい」曲線，下図は「硬い」曲線で回帰していることが分かるであろう．

3　カーネルサポートベクターマシン

3.1　カーネル回帰の定式化

カーネル回帰の定式化が理解出来ていれば，K-SVM の定式化は容易である．K-SVM は，一般化された SVM(15)-(17) における $f_w(\boldsymbol{x})$ をカーネル回帰に置きかえたものであり，結局のところ，最適化問題 (17) を解くことに帰着される．まずは $f_w(\boldsymbol{x})$ をカーネル回帰に置きかえた最適化問題 (17) を明らかにしよう．式 (43) を最適化問題 (17) に代入し，さらに正則化項を付加すると

$$\begin{aligned} \min_{\xi_i, a_i \in \mathbb{R}} \ & \sum_{i\in\mathcal{I}} \xi_i + \lambda \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} a_i k_{ij} a_j, \\ \text{s.t.} \ & \xi_i \geq 1 - y_i \sum_{j\in\mathcal{I}} k_{ij} a_j, \ \text{ for all } i \in \mathcal{I}, \\ & \xi_i \geq 0, \qquad\qquad\qquad \text{for all } i \in \mathcal{I}, \end{aligned} \tag{45}$$

となり，さらにこれを行列形式にまとめると

$$\begin{aligned} \min_{\boldsymbol{\xi}, \boldsymbol{a} \in \mathbb{R}^N} \ & \mathbf{1}^\mathsf{T}\boldsymbol{\xi} + \lambda \boldsymbol{a}^\mathsf{T} \boldsymbol{K} \boldsymbol{a}, \\ \text{s.t.} \ & \boldsymbol{\xi} \geq \mathbf{1} - \boldsymbol{L}\boldsymbol{a}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{46}$$

となる．ただし

$$\boldsymbol{\xi} \triangleq \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix}, \quad \boldsymbol{L} \triangleq \begin{pmatrix} y_1 k_{11} & \cdots & y_1 k_{1n} \\ \vdots & \ddots & \vdots \\ y_n k_{n1} & \cdots & y_n k_{nn} \end{pmatrix} \tag{47}$$

である．式 (46) は 2 次計画問題に帰着することができ，かつ行列 $\boldsymbol{K}$ が正定値行列であることから（これはカーネル関数が正定値関数であることから直ちにしたがう），最適解は唯一存在し，また数値的にも安定して解くことができる．

3.2　カーネルサポートベクターマシンの Python 実装

K-SVM は先にも紹介した LIBSVM や scikit-learn から利用可能である．特に scikit-learn は統一的なインタフェースで様々な機械学習アルゴリズムが利用可能となっており，実応用上はこちらのパッケージを利用するのが良い．

付録：本文で省略した証明

シュワルツの不等式

関数空間 L^2 上の内積の性質としてシュワルツの不等式（*Schwarz inequality*）を紹介したが，これ再掲し証明しよう．

定理 3.1 (シュワルツの不等式)

関数空間 L^2 に対し，$f, g \in L^2$, $a, b \in \mathbb{R}$ とする．このとき

$$\langle f, g \rangle = \|f\| \, \|g\|, \tag{48}$$

が成り立つ．

証明： 任意の $\alpha, \beta \in \mathbb{R}$ に対して $2\alpha\beta \leq \alpha^2 + \beta^2$ が成り立つことは明らかであるから，$\alpha = f(x)/\|f\|$, $\beta = g(x)/\|g\|$ とおけば

$$\frac{2f(x)g(x)}{\|f\| \, \|g\|} \leq \frac{f(x)^2}{\|f\|^2} + \frac{g(x)^2}{\|g\|^2}, \tag{49}$$

が成り立つ．上式の両辺を区間 $(-\infty, \infty)$ で積分すれば，

$$\frac{2\langle f, g \rangle}{\|f\| \, \|g\|} \leq \frac{\|f\|^2}{\|f\|^2} + \frac{\|g\|^2}{\|g\|^2} = 2, \tag{50}$$

を得る．上式より示すべき式は直ちに導かれる． ■

三角不等式

関数空間 L^2 上のノルムの性質として三角不等式を紹介したが，これ再掲し証明しよう．

定理 3.2 (三角不等式)

関数空間 L^2 に対し，$f, g \in L^2$ とする．このとき

$$\|f + g\| \leq \|f\| + \|g\|, \tag{51}$$

が成り立つ．

証明： シュワルツの不等式によれば

$$\|f+g\|^2 \leq \|f\|^2 + \langle f, g \rangle + \|g\|^2 = \left(\|f\|^2 + \|g\|^2\right)^2, \tag{52}$$

が成り立つ．よって証明すべきは明らか． ■

おわりに

本文書は，SVM という実用的な機械学習のツールを詳細に解説しつつ，同時に関数解析の効用を読者に理解して頂こうと，著者なりに試みた結果なのですが，いかがでしたでしょうか．

本文書における SVM の説明の大部分は赤穂 [3] にしたがっています．カーネル回帰の説明は私のオリジナルによる部分が多いですが，関数空間の導入部分は手抜きが甚だしいと言わざるを得ません．関数解析に興味ある読者は関数解析学の成書を参照して下さい．私の知る範囲での関数解析学の良書として，Jost [4] や堀内ら [5] を挙げておきます．

また，本文書では K-SVM に主眼をおいたため，実は一般的でないアプローチで SVM を導入しています．一般的なアプローチ（マージン最適化）は様々な文献で紹介されていますの

で，興味のある読者はそちらをご参照下さい．代表的な参考書としては Bishop[6] を挙げておきます．

再生核 Hilbert 空間についても，本文書では触れませんでした．カーネル関数の設計に興味のある読者は，ぜひ触れてみると良いと思います．

本文書で紹介した内容の発展として，RFF (*Random Fourier Features*) あるいは ORF (*Orthogonal Random Features*) によるカーネル関数の有限次元化が挙げられます．カーネル法の限界のひとつに，学習に必要な計算量が学習データ数に応じて多項式的に増加してしまうという点があります．これは，カーネル法はビッグデータを扱えないことを意味しており，無限の自由度を手にした代償とも言えるでしょう．この問題を解消するために，RFF や ORF ではカーネル関数を適切な次数で有限近似し，カーネルの自由度を活かしつつ計算量を抑えることで高速化を実現しています．興味ある読者は是非 Rahimi *et al.*[7]，Yu *et al.*[8] をご参照下さい．本文書の知識で十分に読み進められると思います．

最後に，Toyota Research Institute Advanced Development, Inc. の乙部成史くんには本文書の誤植をいくつも指摘して頂きました．この場を借りて厚く御礼申し上げます．また，私の数学的活動は，2017 年に逝去された恩師，山下弘一郎先生や，大学および大学院で私の指導教官を担当して下さった早川朋久准教授をはじめ，数学で私と関わりを持ったすべての方々のおかげで成り立っています．そして，数学的活動の以前に，そもそも私の生は両親によって与えられ，妻によって支えられています．

参考文献

[1] V. Vapnik and A. Lerner, “Pattern recognition using generalized portrait method”, Automation and Remote Control, vol. 24, 1963.

[2] B. Boser, I. Guyon and V. Vapnik, “A training algorithm for optimal margin classifiers”, Proc. of the fifth annual workshop on Computational learning theory, pp. 144-152, 1992

[3] 赤穂昭太郎，「カーネル多変量解析」，岩波書店，2008.

[4] J. Jost, “Postmodern Analysis”, Springer, 2005.

[5] 堀内利郎，下村勝孝，「関数解析の基礎 – ∞ 次元の微積分」，内田老鶴圃，2005.

[6] C. Bishop, “Pattern Recognition and Machine Learning”, Springer, 2010.

[7] A. Rahimi and B. Recht, “Random Features for Large-Scale Kernel Machines”, Neural Information Processing Systems, 2007.

[8] F. X. Yu, A. T. Suresh, K. Choromanski, D. H. Rice and S. Kumar, “Orthogonal Random Features”, Neural Information Processing Systems, 2016.