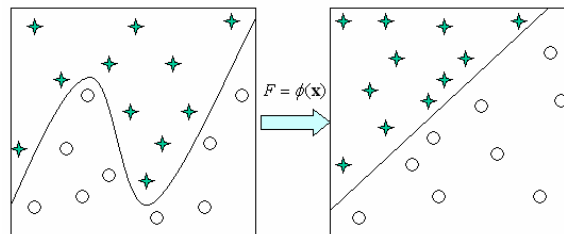


Rとカーネル法・サポートベクターマシン

1. カーネル法とは

図 1 に示すように、非線形データ構造を線形構造に変換することができれば、線形データ解析手法で非線形データを容易に扱うことができる。

図 1 変換による線形化のイメージ



データを変換することで、非線形構造を線形構造に変換することが可能である。例えば、図 2(a)に示す 2次元平面座標系 (x,y) 上の 4つの点 $A1(1, 1)$ 、 $A2(1, -1)$ 、 $A3(-1, -1)$ 、 $A4(-1, 1)$ を考えよう。仮に $A1$ と $A3$ がひとつのクラス、 $A2$ と $A4$ がひとつのクラスだとすると、平面上でクラスの境界線を一本の直線で引くことができない。しかし、新しい変数 $z=xy$ を導入し、2次元平面 (x,y) 上の 4つの点を 3次元空間 (x,y,z) に射影すると $A1(1, 1, 1)$ 、 $A2(1, -1, -1)$ 、 $A3(-1, -1, 1)$ 、 $A4(-1, 1, -1)$ になり、両クラスは平面で切り分けることが可能である。例えば、 $z=0$ の平面を境界面とすることができる。

図 2 データ写像の例

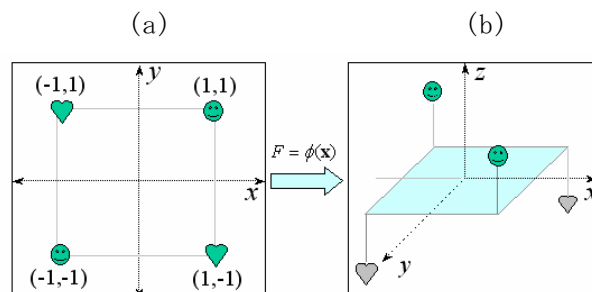


図 1 では、関数 $\phi(\mathbf{x})$ を用いて個体の特徴・属性ベクトルについて変換を施している。関数 $\phi(\mathbf{x})$ は、通常高次元への写像関数で、 \mathbf{x} を入力空間、変換された \mathbf{F} を特徴空間と呼ぶ。

従来のデータ解析方法では、高い次元のデータを低次元に縮約して分析を行う。その典型的

な方法としては、主成分分析、因子分析、対応分析、多次元尺度法などがある。

データを高い次元の特徴空間に射影すると非線形問題を線形問題に置き換えることが可能であるが、計算量が増える。カーネル(kernel)法は、データを高次元に射影し、線形問題に置き換えると同時に計算量の問題を解決する技法である。

カーネル法では射影された高次元のデータを直接計算するのではなく、任意の個体 \mathbf{x} , \mathbf{z} を変換した $\phi(\mathbf{x})$, $\phi(\mathbf{z})$ の内積 $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ のような処理を借りて間接的に高次元のデータについて計算処理を行う。このようなデータの変換と内積のような演算を組み合わせた関数をカーネル関数と呼び、 $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ のように表記する。カーネルに関する厳密な定義やカーネル関数の性質などについては[1]、[3]、[4]が詳しい。

カーネル法を取り入れた幾つかのデータ解析方法が提案されている。例えば、カーネル主成分分析、カーネル正準相関分析、カーネルクラスター分析、カーネルk平均ほう、カーネル回帰分析、カーネル判別分析などがある。本稿では、カーネル主成分分析とカーネル法による分類器サポートベクターマシンについて紹介する。

2 カーネル主成分分析

カーネル主成分分析(KPCA; kernel principal component analysis)は、非線形主成分分析とも呼ばれている。カーネル主成分分析には幾つかのアルゴリズムが提案されているが、その大まかな流れは次のステップを取る。

- (1) カーネル関数 $K(\mathbf{x}, \mathbf{z})$ を決める
- (2) データから写像行列 $K_{m \times m}$ を求める
- (3) $K_{m \times m}$ の固有値と固有ベクトルを求める
- (4) 固有値と固有ベクトルを正規化する

2.1 パッケージと関数

パッケージ kernlab には、カーネル主成分分析の関数 **kpca** がある。パッケージ kernlab は CRAN ミラーサイトからダウンロードできる。次に関数 kpca の書き式を示す。

```
kpca(x, kernel = "rbfdot", features=0, kpar= list(sigma = 0.1), ...)
```

引数 x はマトリックスとデータフレーム形式のデータである。引数 kernel では、用いるカーネル関数を指定する。デフォルトには"rbfdot"(ガウシアン)が指定されているが、これ以外に、カーネル関数"polydot"(多項式)、"vanilladot"(線形)、"tanhdot"(タンジェント)、"laplacedot"(ラプラシアン)、"besseldot"(ベッセル)、"anovadot"(ANOVA RBF)、"splinedot"(スプライン)が用意されている。これらの関数は [2]に定義されている。

引数 features では、求める主成分の数を指定する。デフォルトはゼロになっている。引数

kpar はカーネル関数に用いるパラメータを指定する。

結果としては、固有値 `eig()`、主成分ベクトル `kpc()`、用いたデータの主成分得点 `pcv()`、回転・射影後の主成分得点 `rotated()` が返される。

2.2 カーネル主成分分析の例

次にデータ `iris` を用いた主成分得点の散布図を作成するコマンドとその結果を図 3(a) に示す。

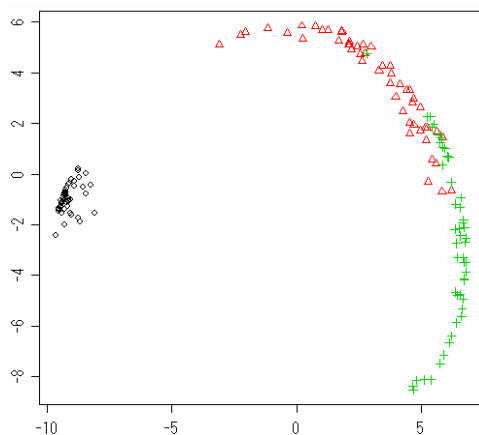
```
> library(kernlab)
> x<-as.matrix(iris[,1:4])
> iris.kpc1<-kpca(x, kernel="rbfdot", features=2, kpar=list(sigma=0.1))
> plot(pcv(iris.kpc1), col=as.integer(iris[,5]))
```

カーネル主成分分析法は、古典的主成分分析方法と異なり、用いるカーネル関数および `kpar` のパラメータによって返される結果が異なる。カーネル関数を `kernel="polydot"`、`kpar` のパラメータを `list(degree =1)` にしたコマンドラインとその結果を図 3(b) に示す。

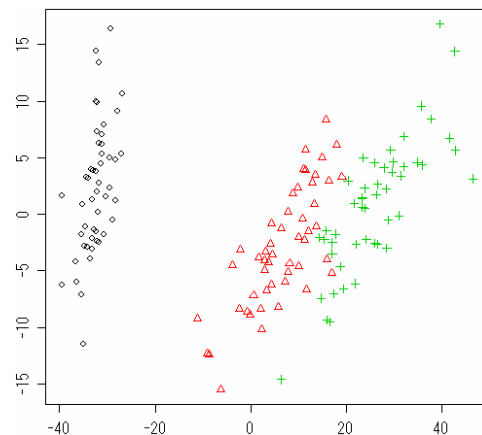
```
> iris.kpc2<-kpca(x, kernel="polydot", kpar= list(degree= 1), features=2)
> plot(pcv(iris.kpc2), col=as.integer(iris[,5]))
```

図 3 iris のカーネル主成分得点散布図

(a)



(b)



このようにカーネル関数、`kpar` のパラメータは主成分の結果に大きく影響する。どのようなカーネル関数を用い、`kpar` のパラメータをどのような値にするべきであるかに関しては、用いるデータに依存するので、経験に頼るのが現状である。

返された主成分ベクトル `kpc()` は、新しいデータを当てはめるときの学習モデルとして用いる。

次に `iris.kpc` の結果を用いて新しいデータ `new.data` を当てはめる書き式を示す。

```
predict(kpc(iris.kpc), new.data)
```

3 サポートベクターマシン

サポートベクターマシン (SVM; support vector machine) は、分類と回帰問題を主としたデータ解析方法で、広く知られるようになったのは 1990 年代の中頃であり、Vapnik, V の貢献が高く評価されている。support vector machine をサポートベクターマシンと訳するかそれともサポートベクトルマシンと訳するかについては議論があるが、本稿ではサポートベクターマシンと呼ぶことにする。SVM は高次元の分類問題が得意であると言われている。

SVM は、カーネル関数の力を借りて、線形分離可能な高次元の空間で線形的なアプローチで学習を行うアルゴリズムである。

学習データ集合 (\mathbf{x}_1, y_1) 、 (\mathbf{x}_2, y_2) 、 \dots 、 (\mathbf{x}_m, y_m) があるとする。この $\mathbf{x} = (x_1, x_2, \dots, x_n)$ は個体の特徴ベクトル、 y は目的変数である。 y は回帰問題では数値、分類問題ではクラスのラベルである。

通常、線形回帰と線形判別の問題では次に示す線形モデルを用いる。図 4 における点線上の個体をサポートベクターと呼ぶ。

$$y = f(\mathbf{x}) = \sum_{i=1}^p w_i x_i + b$$

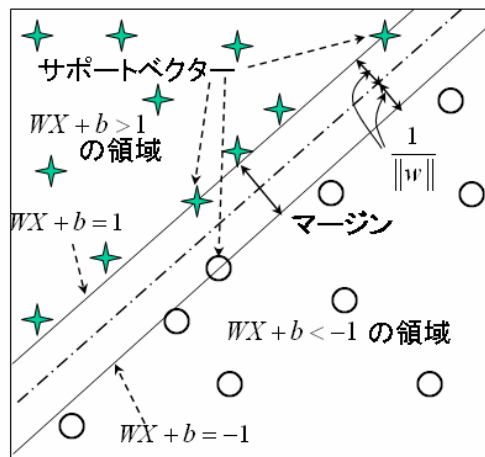


図 4 SVM のイメージ

初期の SVM は 2 群線形分類器として提案されたが、多くの改良が施されている。その 1 つが、カーネル法を用いた SVM である。カーネル法による SVM はカーネル関数を用いて次に示す線形関数で表されるが、非線形分類器である。

$$f(\mathbf{x}) = \sum_{i=1}^n \beta_i K(\mathbf{x}_i, \mathbf{x}) + b$$

式の最適化は特徴空間でクラス間のマージンを最大にするアプローチで行う [3][4].

3.1 パッケージと関数

ここではパッケージ `kernlab` の中の SVM 関数 `ksvm` を紹介する。関数 `ksvm` の書き式を次に示す。

```
ksvm(formula, data, kernel = "rbfdot", kpar=list(sigma = 0.1), type=NULL, cross = 0, ...)
```

引数 `formula` はモデルに用いるデータの書き式、`data` は用いるデータ、引数 `kernel` と `kpar` は前節で説明したカーネル関数と関数に用いるパラメータである。

引数 `type` では、分類と回帰のタイプを指定する。デフォルトは、目的変数が質的データの場合は `C-svc` 分類法、量的データの場合は `eps-svr` 回帰を行うように設定されている。分類方法としては `"nu-svc"`、`"C-bsvc"`、回帰方法としては `"nu-svr"`、`"eps-svr"` が用意されている。引数 `cross` では `n` 重交差確認法の `n` を指定する。デフォルトはゼロになっている。

テストデータを訓練結果へ当てはめる関数は `predict` である。

3.2 データと関数の使用例

パッケージ `kernlab` には、分類問題として面白いデータセット `spam` が用意されている。データセットは、4601 の電子メールを 58 項目に分けて記録したものである。第 58 列がクラス情報 `spam`, `nonspam` で、残りの 57 項目はメールの特徴を記録したものである。この `spam` とは受け取りたくないのに届いた迷惑メールを指す。

```
> library(kernlab)
> data(spam);dim(spam)
[1] 4601 58
> table(spam[,58])
nonspam  spam
 2788    1813
```

返された結果から分かるようにデータは 1813 の `spam` メールと 2788 の `nonspam` メールにより構成されている。データセットの第 1 列から 48 列までは、データ `spam` の変数の名前に用いた文字列がメールに使用された頻度である。ただし、`num857` のように `num***` になっているのは、その数値***が現れた頻度である。49 列から 54 列までは記号 ;、(、[、!、\$、# の使用頻度、55 列から 57 列は、メールに用いられた大文字の平均値、大文字が連続使用された

最も長い文字列の文字数、用いられた大文字の総数である。

まずデータセット spam から、訓練用データとテスト用データを作成する。ここではサンプリング方法を用いることにする。同じサンプリング結果を再現するため、乱数のシード(種)を関数 **set.seed** で指定する。用いるシードの番号は自由であるが、ここでは番号 50 を用いる。このシードを用いることにより、読者のマシン上でも同じ乱数が得られる。ここでは訓練用データの個体数を 2500 にし、その残りをテスト用とする。

```
> set.seed(50)
> tr.num<-sample(4601,2500)
> spam.train<-spam[tr.num,]
> spam.test<-spam[-tr.num,]
```

訓練データを用いて学習を行い、その結果に基づいて関数 predict を用いてテストを行うことにする。

```
> spam.svm <- ksvm(type~.,data=spam.train, kernel="rbfdot",kpar=list(sigma=0.01))
> spam.pre <- predict(spam.svm, spam.test[, -58])
> (spam.tab<-table(spam.test[,58], spam.pre))
      spam.pre
      nonspam spam
nonspam  1226   56
spam      96   723
> 1-sum(diag(spam.tab))/sum(spam.tab)
[1] 0.0723465
```

ランダムサンプリングした 2500 のメールを用いて学習を行い、残り 2101 のメールについてテストを行った結果、誤判別(識別)率は約 0.0724 である。

学習データを用いて交差確認法で、誤判別率などについて考察を行うこともできる。交差確認法の n を 10 にしたコマンドラインとその結果を次に示す。

```
> (train.cro <- ksvm(type~.,data=spam.train, kernel="rbfdot",kpar=list(sigma=0.05),
  C=5, cross=10))
```

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)
parameter : cost C = 5

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.05

Number of Support Vectors : 967
Training error : 0.0164
Cross validation error : 0.082

結果として、学習のエラーと交差確認のエラーが返される。返された交差確認のエラーは 0.082 である。この値は、テストデータを用いて行ったテストの結果 0.072 と大きい差がない。このように、交差確認法を用いて、作成したモデルの精度を把握することができる。

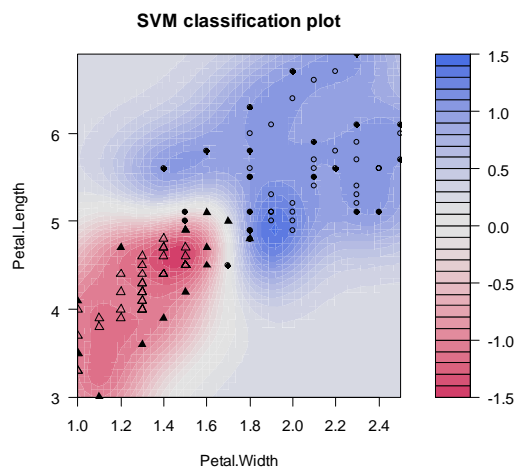
用いるデータが 2 変数で、2 クラスに分類する問題の場合は、関数 `plot` を用いてカラフルな散布図を作成することができる。

データ `iris` の一部分を用いた例のコマンド次に、返される散布図を図 5 に示す。

```
> set.seed(10)
> y<-as.matrix(iris[51:150,5])
> iris1<-data.frame(iris[51:150,3:4],y)
> ir.ksvm<- ksvm(y~.,data=iris1)
> plot(ir.ksvm,data=iris1[,1:2])
> table(iris1$y,predict(ir.ksvm,iris1[,1:2]))
```

	versicolor	virginica
versicolor	48	2
virginica	3	47

図 5 iris の SVM 分類図



3.3 回帰分析のケーススタディ

関数 `ksvm` による回帰分析の書き方は、判別の問題と基本的には同じである。関数 `ksvm` のパフォーマンスを示すため、多項式回帰を説明する際に作成した多項式曲線の人工データを用いることにする。

```
> x1=seq(-10, 10, 0.1);set.seed(10)
> y1=50*sin(x1)+x1^2+10*rnorm(length(x1), 0, 1)
```

説明変数を `x1`、目的変数を `y1` にした関数 `ksvm` の使用例のコマンドを次に示す。

```
> xy.svm<-ksvm(x1, y1, epsilon=0.01, kpar=list(sigma=16))
> sy.pre<-predict(xy.svm, x1)
> plot(x1, y1, type="l")
> lines(x1, sy.pre, col="red", lty=2)
> legend(locator(1), c("実測値", "予測値"), lty=c(1, 2), col=c(1, 2))
```

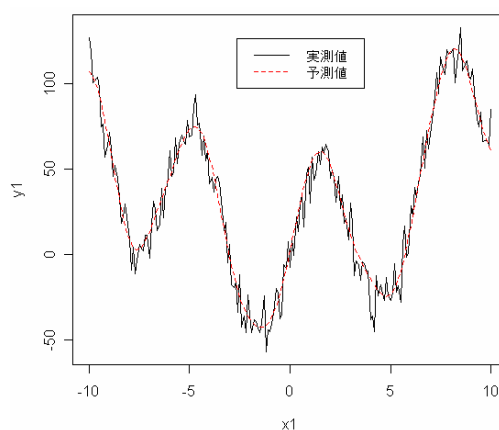


図 15.6 回帰問題における実測値と関数 `ksvm` の予測値

4 その他

パッケージ `kernelab` には、データの特徴を分析するアルゴリズム関数 `kfa`(Kernel Feature Analysis)、カーネルヘッビアン(Kernel Hebbian)アルゴリズムによる主成分分析関数 `khc`、カーネル準相関分析関数 `kcca`、適合ベクターマシン関数 `rvm`(Relevance Vector Machine)などがある。現段階の関数 `rvm` は回帰分析のみが機能している。

カーネル法は機械学習、パターン分析の方法として、研究・応用が広がりつつある。カーネル法によるパターン分析に関しては[1]が詳しい。

SVM は、狭義の分類と回帰問題だけではなく、自然言語処理などへも応用されている。カーネル法と SVM の基礎理論に関しては [3]、[4]が詳しい。また SVM に関しては [5]がある。

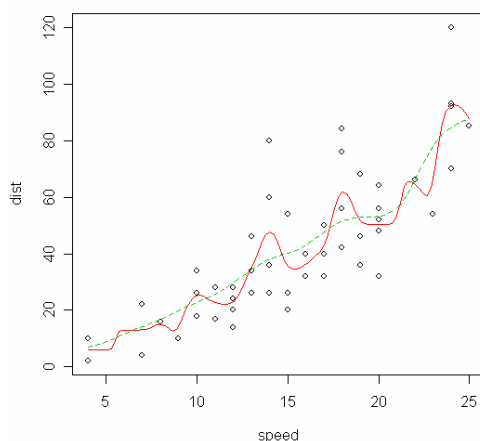
SVM は、パッケージ `klaR`、`e1071` にも実装されている。

カーネル法は回帰、平滑化や密度の推定などにも多く用いる。パッケージ `stats` には平滑化に関する関数 `kernel`、`kernapply`、`ksmooth`、密度を推定する関数 `density` がある。

カーネル法による関数 `ksmooth` を用いたカーネル回帰平滑化の例のコマンドとその結果を次に示す。

```
>attach(cars)
>plot(speed, dist)
>lines(ksmooth(speed, dist, "normal", bandwidth=1.3), col=2)
>lines(ksmooth(speed, dist, "normal", bandwidth=4), col=3,lty=2)
>detach("cars")
```

図 6 関数 `ksmooth` によるカーネル回帰平滑化



カーネル法による平滑化関数パッケージとしては `KernSmooth`、`ks` があり、パッケージ `ade4`、`assist`、`fields`、`lattice`、`splancs`、`sandwich` などにもカーネル法に関連する関数がある。

参考文献

- [1] J. Shawe-Taylor and N. Cristianini (2004): *Kernel Methods for Pattern Analysis*, Cambridge.
- [2] Karatzoglou, A., Smola, A., hornik, K., Zeileis, A. (2004): *kernlab—An S4 Package for Kernel Methods in R*, *Journal of statistical Software*, Vol. 11, Is. 9, p. 1-20. <http://www.jstasoft.org>
- [3] 麻生英樹, 津田宏治, 村田昇 (2003): *パターン認識と学習の統計学—新しい概念と手法*, 岩波書店
- [4] 大北剛訳 (2005): *サポートベクターマシン入門*, 共立出版
- [5] 前田英作 (2001): *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, p. 676-683