教師あり学習（分類）演習ガイド

タスク

Wisconsin Breast Cancer Diagnostic Data Set（https://www.kaggle.com/uciml/breast-cancer-wisconsin-data）は乳がん細胞の 30 の特徴量、良性・悪性いずれかの診断所見、および、事例 ID がセットになったデータセットで、569 個の事例からなる。569 事例の内訳は 357 事例が良性、212 事例が悪性である。ここでの目的は、乳がん細胞核の 30 の特徴量から良性・悪性の診断を行うことができる機械学習モデルを構築することであり、教師あり学習の分類問題となる。この演習では、アルゴリズムとして k 近傍法を利用したモデル構築および決定木を用いたモデル構築を行う。

演習環境

R および R Studio がインストールされていることを前提とする。新規プロジェクト（ディレクトリ名は wbcd とする）を立ち上げ、新規スクリプトを開いておく。R のコマンドは Source ペインに開いたスクリプトに入力して Ctrl + Enter で実行するか、Console ペインに直接入力して Enter で実行する。

この演習ガイドでは入力すべきコマンドを青文字の Lucida Console フォントで、実行結果を黒字の Lucida Console フォントで、以下の例のように示す。

```
> str(iris)
'data.frame':  150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

k 近傍法を用いたモデル構築

冒頭でも述べた通り、なすべきタスクは、Wisconsin Breast Cancer Diagnostic Data Set を利用して、乳がん細胞核の 30 の特徴量から良性・悪性の診断を行うことができる機械学習モデルを構築することであり、教師あり学習の分類問題となる。本章では、アルゴリズムとして k 近傍法（k-nearest neighbor method）を利用したモデル構築を行う。

データの準備

https://www.kaggle.com/uciml/breast-cancer-wisconsin-data から data.csv をダウンロードして現在の作業ディレクトリ（以下で確認）に配置する。現在の作業ディレクトリは getwd()で確認できる。

```
> getwd()
[1] "C:/Users/d_suz/OneDrive/ドキュメント/wbcd"
```

read.csv()を用いて data.csv ファイルを読み込む。stringsAsFactors = FALSE は文字列データを因子（factor）としてではなく文字列のまま読み込むという意味である。

```
> wbcd <- read.csv("data.csv", stringsAsFactors = FALSE)
```

さらに読み込んだデータのデータ構造を確認する。

```
> str(wbcd)
'data.frame':  569 obs. of  32 variables:
 $ id                    : int  842302 842517 84300903 84348301 84358402 843786 844359 ...
 $ diagnosis             : chr  "M" "M" "M" "M" ...
 $ radius_mean           : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean          : num  10.4 17.8 21.2 20.4 14.3 ...
 $ perimeter_mean        : num  122.8 132.9 130 77.6 135.1 ...
 $ area_mean             : num  1001 1326 1203 386 1297 ...
 $ smoothness_mean       : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
（中略）
 $ symmetry_worst        : num  0.46 0.275 0.361 0.664 0.236 ...
 $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

32 個の変数を持つ 569 事例からなるデータフレームである。もし、余分な変数 (X) が入っている場合は NULL を代入することで削除する。

```
> wbcd$X <- NULL
```

まず、id 変数は単なる事例の識別番号であり、機械学習モデルの特徴量としては不要なので削除する。

```
> wbcd$id <- NULL
> str(wbcd)
'data.frame':  569 obs. of  31 variables:
 $ diagnosis             : chr  "M" "M" "M" "M" ...
 $ radius_mean           : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean          : num  10.4 17.8 21.2 20.4 14.3 ...
（後略）
```

次に、diagnosis（診断結果）の集計を table()関数で確認する。

```
> table(wbcd$diagnosis)

  B   M
357 212
```

diagnosis（診断結果）は B = Benign（良性），M = Malignant（悪性）の二水準からなる。R の機械学習分類機は目的変数が factor（因子）指定されている必要があるため、diagnosis を factor に変更する。あわせて、わかり

やすさのため、水準（levels）のラベル（labels）の B を Benign に、M を Malignant にそれぞれ変更する。

```
> wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B","M"), labels = c("Benign", "Malignant"))
> str(wbcd)
'data.frame':  569 obs. of  31 variables:
 $ diagnosis            : Factor w/ 2 levels "Benign","Malignant": 2 2 2 2 2 2 2 2 2 ...
 $ radius_mean          : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean         : num  10.4 17.8 21.2 20.4 14.3 ...
```

（後略）

これにより確かに、diagnosis が Factor w/ 2 levels “Benign”, “Malignant”になっており、diagnosis が 2 水準からなる因子であり、水準のラベルが Benign と Malignant となっていることが確認できる。

データの範囲を把握するため、例えば 2 列目（raduis_mean）から 6 列目（smootheness_mean）の特徴量に関して基本統計量を summary()関数で求める。

```
> summary(wbcd[2:6])
  radius_mean      texture_mean   perimeter_mean     area_mean      smoothness_mean
 Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5   Min.   :0.05263
 1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637
 Median :13.370   Median :18.84   Median : 86.24   Median : 551.1   Median :0.09587
 Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9   Mean   :0.09636
 3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530
 Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0   Max.   :0.16340
```

取り得る値の範囲が特徴量によって大きく異なることがわかる。smoothness_mean は 0.05 から 0.16 の範囲の値を取るのに対し、area_mean は 143.5 から 2501.0 である。

k 近傍法のための前処理

k 近傍法では特徴量空間における距離に基づいて分類を行う。ここで、距離の計算に関して、特徴量の間で取り得る値の範囲が大きく異なると望ましくない。今の場合、距離の計算において area_mean は大きな影響を及ぼすが、smoothness_mean はその値の小ささのためほとんど考慮されないことになる。これを防ぐため、どの特徴量に関しても取り得る値の範囲が[0,1]になるように正規化 min-max normalization を行う。

まず、min-max normalization のための関数 normalize を以下のように定義する。なお、Source ペインでは Enter で、Console ペインでは Shift+Enter でコマンドの途中で改行が可能である。

```
> normalize <- function(x){
+     return (
+         (x-min(x))/(max(x)-min(x))
+     )
+ }
```

なお、左端の+記号は改行しているもののコードが継続していることを示す単なる表記である。コマンド入力中に、わざわざ+記号を入力する必要はない。

最後の行で Ctrl+Enter で実行する。

1 列目は diagnosis（診断結果）であるため、それ以外の 2 列目から 31 列目のデータに対して normalize を適用する。ここで lapply()関数を用いる。lapply(somelist, somefunc) 関数は第一引数に指定したリスト somelist の個々の要素に、第二引数で指定した関数 somefunc を適用する。lapplay(wbcd[2:31], normalize)とした場合、リスト wbcd[2:31]の要素である wbcd[2]から wbcd[31]のそれぞれ(これらはベクトルである)に対して、normalize 関数を適用する。lapply()関数の戻り値はリストとなるため、as.data.frame()関数によりリストからデータフレームへと変換する。なお、diagnosis は後ほど、正解ラベルとして別途保存する。

```
> wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
> summary(wbcd_n[2:6])
 texture_mean    perimeter_mean     area_mean      smoothness_mean  compactness_mean
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.2185   1st Qu.:0.2168   1st Qu.:0.1174   1st Qu.:0.3046   1st Qu.:0.1397
Median :0.3088   Median :0.2933   Median :0.1729   Median :0.3904   Median :0.2247
Mean   :0.3240   Mean   :0.3329   Mean   :0.2169   Mean   :0.3948   Mean   :0.2606
3rd Qu.:0.4089   3rd Qu.:0.4168   3rd Qu.:0.2711   3rd Qu.:0.4755   3rd Qu.:0.3405
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

確かに、どの特徴量も取り得る値の範囲が[0,1]になっていることが確認できる。

訓練データセットとテストデータセットの準備

データセットを訓練データとテストデータに分割する。元のデータセット 569 サンプルから無作為に 469 サンプルを選んで訓練データセット、残りの 100 サンプルをテストデータセットとしたい。

データセットの並びがランダムな場合

もし、データセットが元々ランダムに 569 サンプル並んでいる場合、単純に先頭から順番に、1 から 469 を訓練データセット、470 から 569 までをテストデータセットとすればよい。

```
> wbcd_train <- wbcd_n[1:469,]
> wbcd_test <- wbcd_n[470:569,]
```

ここで、wbcd_n[1:469,] は 1 行目から 469 行目までのすべての列（特徴量）を意味する。

目的変数となる diagnosis のデータは、訓練データセットの正解ラベル wbcd_train_labels、テストデータセットの正解ラベル wbcd_test_labels に分けて格納する。wbcd の 1 列目が diagnosis である。

```
> wbcd_train_labels <- wbcd[1:469,1]
> wbcd_test_labels <- wbcd[470:569,1]
```

データセットの並びがランダムではない場合

もし、データセットの並びがランダムではなく、例えば、1 行目から 357 行目が良性腫瘍（Benign）の事例、358 行目から 569 行目が悪性腫瘍（Malignant）の事例のように並んでいる場合、先の分割方法では不適当である。つまり、訓練データセットの多数が良性腫瘍の事例、テストデータセットがすべて悪性腫瘍の事例になってしまう。そこで、このような場合には、1 から 569 の整数の並びから無作為に 469 個の整数を選んで数字列を作成し、これを添え字として利用することで、データセットを分割する

無作為な数字列の生成には sample()関数を用いる。sample(569,469)とすると、1 から 569 の整数の並びから無作為に 469 個の整数を選んで数字列を作成することができる。なお、set.seed(101)は乱数の種を固定することで、生成する乱数列を後から再現できるようにしている。

```
> set.seed(101)
> train_sample <- sample(569,469)
> str(train_sample)
 int [1:469] 430 95 209 442 351 315 246 131 521 352 ...
```

train_sample は 430 95 209 442 … と、無作為に並んだ整数列であることが確認できる。これを添え字として wbcd_n[train_sample,]とすることで訓練データセットを作成する。

```
> wbcd_train <- wbcd_n[train_sample,]
```

これにより、wbcd_train は 430 行目、95 行目、209 行目、442 行目、…の事例を集めたデータセットとなる。

次に、これら以外のデータを集めてテストデータセットとしたい。これは wbcd_n[-train_sample,]のように添え字にマイナスをつけることで可能である。

```
> wbcd_test <- wbcd_n[-train_sample,]
```

目的変数となる diagnosis のデータは、訓練データセットの正解ラベル wbcd_train_labels、テストデータセットの正解ラベル wbcd_test_labels に分けて格納する。

```
> wbcd_train_labels <- wbcd[train_sample,1]
> wbcd_test_labels <- wbcd[-train_sample,1]
```

訓練データとテストデータの比率の確認

訓練データセットとテストデータセットに分割後、それぞれのデータセットにおける良性と悪性の比率が、互いに同じ程度になっていることを以下の通り確認する。prop.table()関数は実際の事例の数ではなく、比率を求めたいときに用いる。

```
> prop.table(table(wbcd_train_labels))
wbcd_train_labels
```

```
   Benign Malignant
0.6247335 0.3752665
```

```
> prop.table(table(wbcd_test_labels))
wbcd_test_labels
   Benign Malignant
     0.64      0.36
```

訓練データセットとテストデータセットで、比率がおおむね同じになっていることが確認できた。

モデルの訓練

k 近傍法（k nearest neighbor method）の実装 knn()を用いるため、class パッケージをインストールし、ロードする。

```
> install.packages("class")
> library(class)
```

class パッケージの knn()関数は 4 つの引数を取る。構文は以下のとおりである。

p <- knn(train, test, class, k)

- ✓ train 訓練データを格納するデータフレーム
- ✓ test テストデータを格納するデータフレーム
- ✓ class 訓練データの正解ラベルを格納する因子ベクトル
- ✓ k いくつの近傍の投票でクラスを決定するか

なお戻り値 p はテストデータに対する予測ラベルを格納する因子ベクトルとなる。

ところで、k 近傍法では訓練といっても、訓練データを格納するだけであり、テストデータを与えれば直ちに、予測を行うことができる。ここでは訓練データ 469 事例の平方根に近い k=21 とする。テストデータがどのクラスに分類されるかを予測する際に、特徴量空間におけるテストデータの 21 個の近傍点を見つけ、多数派となった方のクラスに分類する。なお、必ず多数派が決定できるようにするため k には奇数を指定する。

```
> wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels, k =
 21)
```

これで wbcd_test_pred は、wbcd_test の 100 事例について、それぞれ特徴量から予測した診断結果（Benigin か Malignant のいずれか）100 個からなるベクトルとなっている。

モデルの性能評価

テストデータの正解ラベルは wbcd_test_labels に、予測ラベルは wbcd_test_pred に格納されている。これらをクロス集計して、モデルの性能評価を行う。パッケージ gmodels をインストール、ロードし、クロス集計する。

```
> install.packages("gmodels")
```

```
> library("gmodels")
> CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)

   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  100

               | wbcd_test_pred
wbcd_test_labels |    Benign | Malignant | Row Total |
-----------------|-----------|-----------|-----------|
         Benign |        63 |         1 |        64 |
                |     0.984 |     0.016 |     0.640 |
                |     0.940 |     0.030 |           |
                |     0.630 |     0.010 |           |
-----------------|-----------|-----------|-----------|
      Malignant |         4 |        32 |        36 |
                |     0.111 |     0.889 |     0.360 |
                |     0.060 |     0.970 |           |
                |     0.040 |     0.320 |           |
-----------------|-----------|-----------|-----------|
   Column Total |        67 |        33 |       100 |
                |     0.670 |     0.330 |           |
-----------------|-----------|-----------|-----------|
```

Malignant（悪性）と予測して事実悪性であった例（真陽性）が 32 例、悪性と予測して実は良性であった例（偽陽性）が 1 例、Benign（良性）と予測して事実良性であった例（真陰性）が 63 例、良性と予測して実は悪性であった例（偽陰性）が 4 例となっている。100 例中 95 例が正しく分類できており、accuracy は 95%となる。

決定木を用いたモデル構築

冒頭でも述べた通り、なすべきタスクは、Wisconsin Breast Cancer Diagnostic Data Set を利用して、乳がん細胞核の 30 の特徴量から乳がんの良性・悪性を診断する機械学習モデルを構築することであり、教師あり学習の分類問題となる。ここでは決定木を用いたモデル構築を行う。

データの準備

先の k 近傍法を利用したモデル構築の際に準備した wbcd を用いる。

```
> str(wbcd)
'data.frame':  569 obs. of  31 variables:
 $ diagnosis            : Factor w/ 2 levels "Benign","Malignant": 2 2 2 2 2 2 2 2 2 ...
 $ radius_mean          : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean         : num  10.4 17.8 21.2 20.4 14.3 ...
（後略）
```

訓練データセットとテストデータセットの準備

決定木の場合、特徴量を標準化する必要がない。このため、標準化などの前処理を行わずに直接 wbcd から訓練データ 469 例とテストデータ 100 例に分割する。1 列目は diagnosis（診断結果）であり目的変数であるから、正解ラベルとして分離する。特徴量のデータフレームを作成する際には、wbcd[train_sample, -1] などとすることで 1 列目の diagnosis を含まないようにすることができる。

```
> set.seed(101)
> train_sample <- sample(569,469)
> str(train_sample)
 int [1:469] 430 95 209 442 351 315 246 131 521 352 ...
> wbcd_train2 <- wbcd[train_sample,-1]
> wbcd_test2 <- wbcd[-train_sample,-1]
```

逆に 1 列目の diagnosis のみからなるベクトルを以下のようにして準備する。

```
> wbcd_train2_labels <- wbcd[train_sample,1]
> wbcd_test2_labels <- wbcd[-train_sample,1]
```

また、一応ここでも、各データセットにおける良性と悪性の比率が、互いに同じ程度になっていることを確認しておく。

```
> prop.table(table(wbcd_train2_labels))
wbcd_train2_labels
   Benign malignant
```

```
0.6247335 0.3752665
```

```
> prop.table(table(wbcd_test2_labels))
wbcd_test2_labels
   Benign malignant
     0.64      0.36
```

モデルの訓練

決定木を用いたモデル構築に C5.0 アルゴリズムを用いるため、C50 パッケージをインストール・ロードする。

```
> install.packages("C50")
> library(C50)
```

C5.0()の構文は以下のとおりである。

m <- C5.0(train, class, trials =1, costs = NULL)

- ✓ train 訓練データを格納するデータフレーム
- ✓ class 訓練データの正解クラスを格納する因子ベクトル
- ✓ trials ブースティングの回数（オプション．デフォルトは 1）
- ✓ costs コスト（オプション）

C5.0 関数の戻り値は、予測に用いることのできる訓練済みモデルである。

訓練済みモデルを用いた予測は predict()関数で行う。predcit()関数の構文は以下のとおりである。

p <- predict(m, test, type = “class”)

- ✓ m C5.0()関数で訓練されたモデル
- ✓ test テストデータを格納したデータフレーム
- ✓ type 戻り値のタイプ：予測されるクラス class か、確率 prob か（オプション）

C5.0()関数の第一引数に訓練データセット wbcd_train2, 第二引数に対応する正解クラスを格納するベクトルを指定して呼び出す。

```
> wbcd_model <- C5.0(wbcd_train2, wbcd_train2_labels)
```

訓練済みモデルを格納したオブジェクト名を入力すると決定木に関する基本データが得られる。

```
> wbcd_model

Call:
C5.0.default(x = wbcd_train2, y = wbcd_train2_labels)
```

```
Classification Tree
Number of samples: 469
Number of predictors: 30

Tree size: 12

Non-standard options: attempt to group attributes
```

サンプルサイズ 469、予測子（特徴量）30 を用いて訓練した結果、木の深さが 12 となったことがわかる。決定木を用いて実際にどのようなルールで分類しているかは summary()で知ることができる。

```
> summary(wbcd_model)
Call:
C5.0.default(x = wbcd_train2, y = wbcd_train2_labels)

C5.0 [Release 2.07 GPL Edition]        Thu Mar 26 01:57:06 2020
-------------------------------
Class specified by attribute `outcome'
Read 469 cases (31 attributes) from undefined.data
Decision tree:

area_worst > 880.8:
:...concavity_mean > 0.06335: malignant (138)
:   concavity_mean <= 0.06335:
:   :...texture_worst <= 29.72: Benign (6/1)
:       texture_worst > 29.72: malignant (6)
area_worst <= 880.8:
:...concave.points_worst <= 0.1318:
    :...area_se <= 36.46: Benign (258/1)
    :   area_se > 36.46:
    :   :...symmetry_worst <= 0.206: malignant (2)
    :       symmetry_worst > 0.206:
    :       :...concave.points_worst <= 0.1108: Benign (12)
    :           concave.points_worst > 0.1108: malignant (2)
    concave.points_worst > 0.1318:
    :...texture_mean > 20.28: malignant (19)
        texture_mean <= 20.28:
        :...symmetry_worst > 0.3549: malignant (4)
            symmetry_worst <= 0.3549:
```

```
:...area_worst <= 809.8: Benign (17)
    area_worst > 809.8:
    :...area_worst <= 832.7: malignant (3)
        area_worst > 832.7: Benign (2)
```

これによると、面積の最悪値が 880.8 より大きく、凹部平均が 0.06335 より大きい場合、悪性と判定される。この規則には 138 例があてはまり、それらは実際に悪性である。面積の最悪値が 880.8 より大きく、凹部平均が 0.0635 以下、テクスチャの最悪値が 29.72 以下の場合、良性と判定される。この規則には 6 例があてはまり、うち 5 例は良性、1 例は悪性である（6/1）ことがわかる。

```
Evaluation on training data (469 cases):

          Decision Tree
        ----------------
        Size      Errors

          12    2( 0.4%)   <<


         (a)   (b)    <-classified as
        ----  ----
         293          (a): class Benign
           2   174    (b): class malignant
```

上記混同行列からこの決定木モデルは、訓練データ 469 例に関して、誤分類は 2 例、誤り率 0.4%であることがわかる。悪性 176 例に関して、正しく陽性と判定（真陽性）したのが 174 例、誤って陰性と判定（偽陰性）のが 2 例、良性 293 例に関して、正しく陰性と判定（真陰性）したのが 293 例、誤って陽性と判定（偽陽性）は 0 例となっている。

モデルの性能評価

テストデータ wbcd_test2 に対して訓練済み決定木モデル wbcd_model を利用して予測を行う。予測結果は wbcd_pred に格納する。対応する正解ラベルは wbcd_test_labels に格納されている。これらをクロス集計して、モデルの性能評価を行う。以下では、CrossTable 関数でクロス集計表を作成する。第 1 引数と第 2 引数にそれぞれ正解ラベルと予測ラベルを指定する。prop.chisq（カイ二乗統計量）、prop.c（列比率）、prop.r（行比率）は不要であれば FALSE 指定する。クロス集計表の行が実際と列が予測であることを見やすくするため dnn を用いて行見出しと列見出しを別途している。

```
> wbcd_pred <- predict(wbcd_model, wbcd_test2)
> CrossTable(wbcd_test2_labels, wbcd_pred, prop.chisq = FALSE, prop.c = FALSE, prop.r =
FALSE, dnn = c('actual','predicted'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  100

            | predicted
     actual |    Benign | malignant | Row Total |
------------|-----------|-----------|-----------|
     Benign |        61 |         3 |        64 |
            |     0.610 |     0.030 |           |
------------|-----------|-----------|-----------|
   malignant |         3 |        33 |        36 |
            |     0.030 |     0.330 |           |
------------|-----------|-----------|-----------|
Column Total |        64 |        36 |       100 |
------------|-----------|-----------|-----------|
```

100 例の中で、94 例について正しく予測しており、accuracy は 94%となる。

なお、決定木を複数組み合わせることも可能である。10 の決定木を組み合わせる場合は以下のとおりである。

```
> wbcd_model2 <- C5.0(wbcd_train2, wbcd_train2_labels, trials =10)
> wbcd_pred2 <- predict(wbcd_model2, wbcd_test2)
> CrossTable(wbcd_test2_labels, wbcd_pred2, prop.chisq = FALSE, prop.c = FALSE, prop.r =
 FALSE, dnn = c('actual','predicted'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  100
```

	predicted		
actual	Benign	malignant	Row Total
Benign	62	2	64
	0.620	0.020	
malignant	2	34	36
	0.020	0.340	
Column Total	64	36	100

100 例中 96 例を正しく分類でき、accuracy は 96%となる。

教師あり学習（回帰）演習ガイド

タスク

カリフォルニア大学アーバイン校 Machine Learning Repository で公開されている Wine Quality Data Set（https://archive.ics.uci.edu/ml/datasets/Wine+Quality）は、ポルトガル北部 "Vinho Verde"（ヴィーニョ・ヴェルデ）ワインの赤ワイン 1,599 品種、白ワイン 4,898 品種のそれぞれについて、11 種類の科学的特性および品質スコア（鑑定士によるブラインドのテイスティングで付与された 0 から 10 の 11 段階の評価値）がセットになったデータセットである。ここでの目的は、白ワインのデータを用いて、11 種類の科学的特性に基づいて品質スコアを推定する機械学習モデルを構築することである。品質スコアは 11 のクラスと見ることもできるが、評価数値の並びと間隔に意味のある間隔尺度と解釈することもできる。これを踏まえ、このタスクを教師あり学習の回帰のタスクととらえる。本演習では、アルゴリズムとして、多重線形回帰を利用したモデル構築およびと SVM を利用したモデル構築を行う。

線形回帰を用いたモデル構築

冒頭でも述べた通り、なすべきタスクは、Wine Quality Data Set のうち白ワインデータセットを利用して、11 種類の科学的特性に基づいて品質スコアを推定する機械学習モデルを構築することであり、教師あり学習の回帰問題である。本章では、アルゴリズムとして多重線形回帰を利用したモデル構築を行う。

データの準備

新規プロジェクト（ディレクトリ名は wine とする）を立ち上げ、新規スクリプトを開いておく。

https://archive.ics.uci.edu/ml/datasets/Wine+Quality の Download: Data Folder から winequality-white.csv をダウンロードして現在の作業ディレクトリに保存する。

read.csv()を用いて winequality-white.csv ファイルを読み込む。ただし、このデータはデータ区切りがカンマ（ , ）ではなくセミコロン（ ; ）なので、引数に sep = ";" を指定する。

```
> white <- read.csv("winequality-white.csv", sep=";")
```

さらに読み込んだデータのデータ構造を確認する。

```
> str(white)
'data.frame':  4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
```

```
$ pH                 : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
$ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
$ alcohol            : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
$ quality            : int  6 6 6 6 6 6 6 6 6 6 ...
```

12 個の変数からなる 4,898 事例からなるデータフレームであることが確認できる。

http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality.names ファイルに詳しい説明がある。入力には客観検査による科学的特徴量が 11 含まれており、出力としてワイン鑑定士の感覚に基づく品質評価値が与えられている。品質評価値は 0（very bad）から 10（very excellent）の 11 段階からなる。科学的な特徴量の説明は以下のとおりである。

fixed acidity: 酒石酸濃度
volatile acidity: 酢酸濃度
citric acid: クエン酸濃度
residual sugar: 残糖濃度
chlorides: 塩化ナトリウム濃度
free sulfur dioxide: 遊離 SO2 濃度
total sulfur dioxide: 総 SO2 濃度
density: 密度
pH: pH
sulphates: 硫化カリウム濃度
alcohol: アルコール度数

データの把握・可視化

まず、データの特徴を把握・可視化するため、いくつかの特徴量の分布や散布図を描きたい。このため、複数の特徴量の分布や特徴量間の散布図、相関係数を並べた散布図行列を作ることのできる pairs.panesl() 関数を用いる。pairs.panesl()関数は psych パッケージに含まれているため、これをインストールし、ロードする。

```
> install.packages("psych")
> library(psych)
```

pairs.panels() 関数は引数にデータフレームを指定するだけで、データフレームに含まれるすべての特徴量の分布と任意の二つの特徴量を組み合わせた散布図および相関係数を並べた散布図行列を描くことができる。しかし、今取り扱っているデータフレーム white は品質評価値を含む 12 の変数からなるため、そのまま散布図行列を描くと 12×12 の散布図や分布関数グラフが並んだ散布図行列を描くことになり、視認しにくい。これをふまえ、ここでは試みにデータフレームのうち 4 つを指定して、描くこととする。

```
> pairs.panels(white[c("volatile.acidity","pH","alcohol","quality")])
```

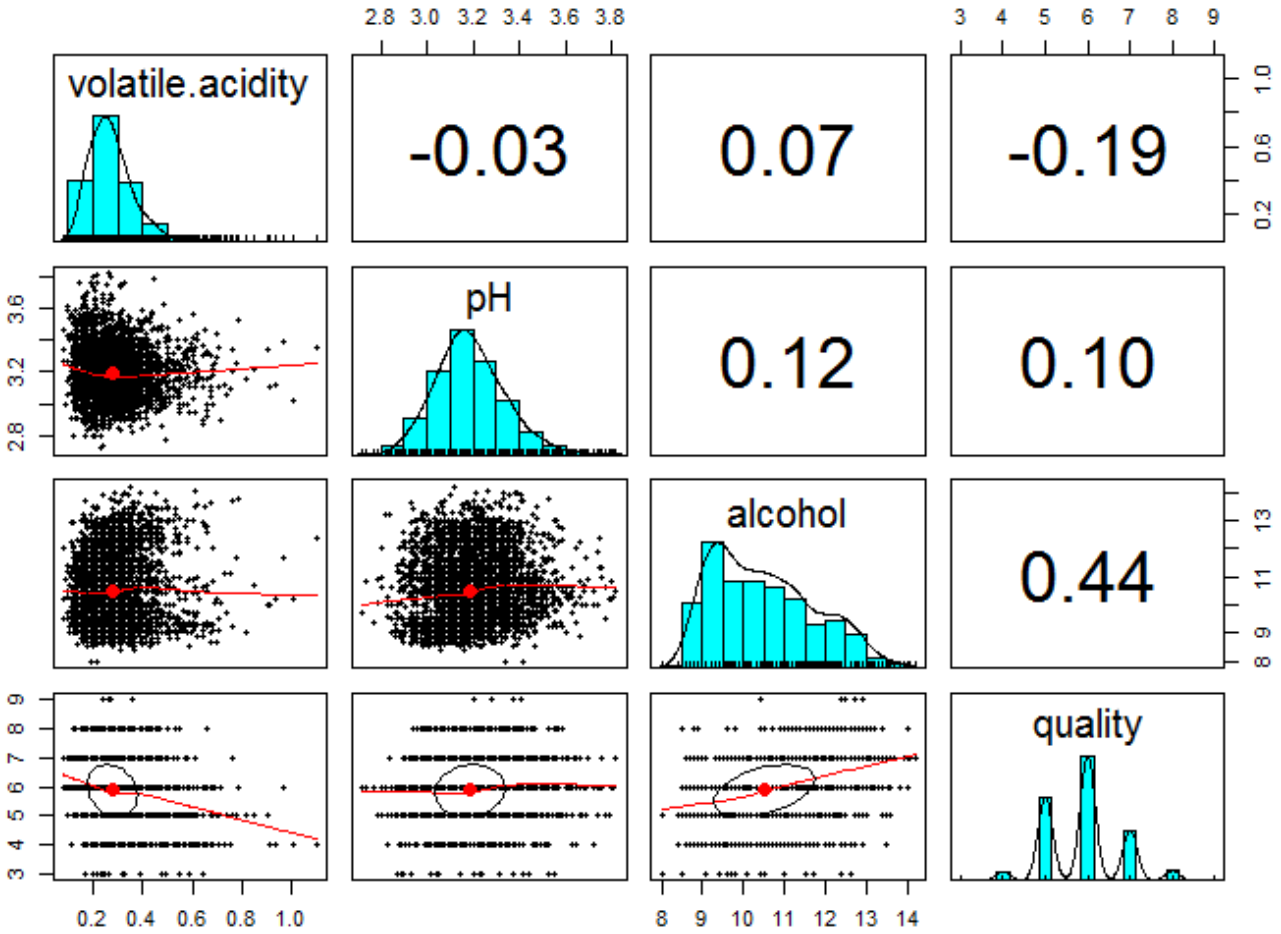


図 1 psych パッケージの pairs.panels()関数で描いた散布図行列の例

対角線に並んでいるのが各特徴量の分布、左下は二つの特徴量の散布図、右上は二つの特徴量の相関係数となっている。散布図に描かれている楕円は相関楕円であり、相関の強さを可視化している。楕円中心の点は平均値を示している。楕円が長く伸びているほど相関が強い。散布図に描かれている曲線は loess 平滑化曲線で x 軸と y 軸の変数の一般的な関係を示している。平滑化曲線が直線に近い場合、変数間の関係は線型に近く、その相関の程度は相関係数に反映される。

図 1 の場合、quality（ワイン鑑定士による品質評価値）の分布を見ると、6 にピークがあり、極端に高い評価や低い評価は少ない分布であること、alcohol と quality の相関が 0.44 と高いこと、遊離酸と品質は弱い負の相関であるなどが見える。

計量経済分析のパッケージである PerformanceAnalytics を用いても同様の散布図行列を描くことができる。

```
> install.packages("PerformanceAnalytics")
> library("PerformanceAnalytics")
```

今度は多少見にくくはなるが、すべての変数に関する散布図行列を描いてみる。

```
> chart.Correlation(white)
```

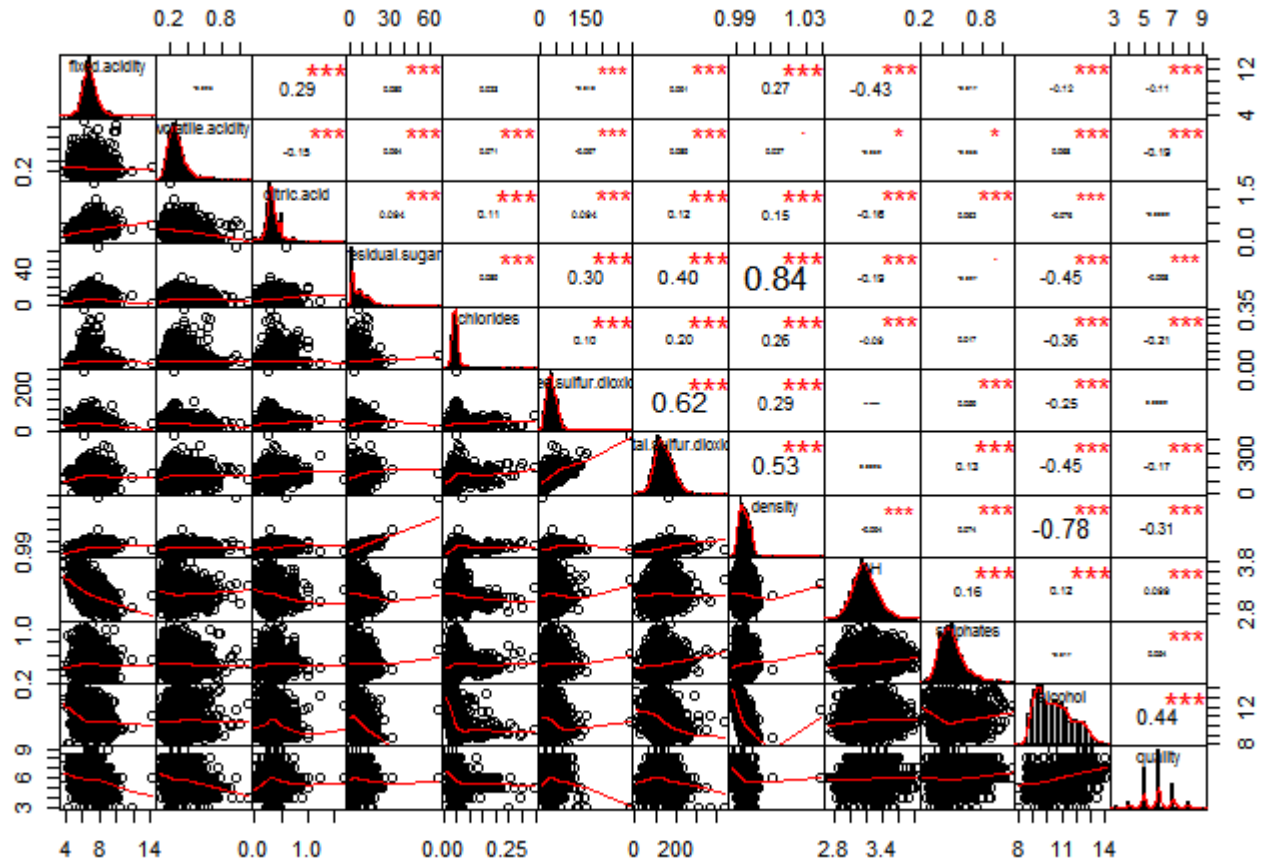


図 2　PerformanceAnalytics パッケージの chart.Correlation()関数で描いた散布図行列の例

psych パッケージの pairs.panels()関数と同じ情報が含まれているが、それ以外にも相関係数のフォントサイズがその数値の大小を反映しており、一見して相関が大きい関係を見つけやすい、*の数でその相関係数が有意か否かを知ることができる、などのメリットがある。*の数は、P 値が 0~0.001 の間であれば ***、0.001 から 0.01 の間であれば ** 、0.01 から 0.05 の間であれば * 、0.05 から 0.1 の間であれば ．、それ以上は無印となる。これをふまえ、目的変数である quality との相関が 5%有意水準で有意であるところの*, **, ***の印が付いた特徴量だけを説明変数として重回帰分析を行うという方針を考えることができる。

また、互いに相関が高い変数をともに独立変数として採用した場合に多重共線性の問題が生じることから、いずれか一方を不採用にするという方針を考えることもできる。多重共線性とは、変数間の高い相関のために回帰係数の分散が大きくなり、係数の有意性が失われる現象である。これを踏まえ、今の場合であれば、density と residual sugar の 2 変数についてはその相関が 0.84 と高いため、多重共線性を回避するため density は採用せずに residual sugar だけを採用する、free.sulfur.dioxide と total.sulfure.dioxide の 2 変数については、その相関が 0.62 と高いため、free.sulfur.dioxide を採用せずに total.sulfur.dioxide だけを採用する、などが考えられる。

しかし、今回はベースラインの実験としてすべての変数を独立変数として採用して回帰分析を行うことにする。

訓練データセットとテストデータセットの準備

データを訓練データとテストデータに分割する。分割の比率は訓練：テスト ＝8:2 や 7:3 などとすることが多い。ここでは、4,898 のうち 3,750 を訓練データ、残りをテストデータとする。

```
> set.seed(101)
> train_sample <- sample(4898,3750)
> white_train <- white[train_sample,]
> white_test <- white[-train_sample,]
```

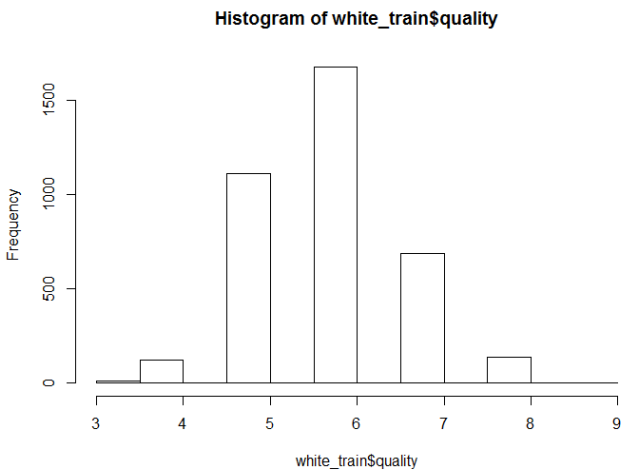
訓練データとテストデータの比率の確認

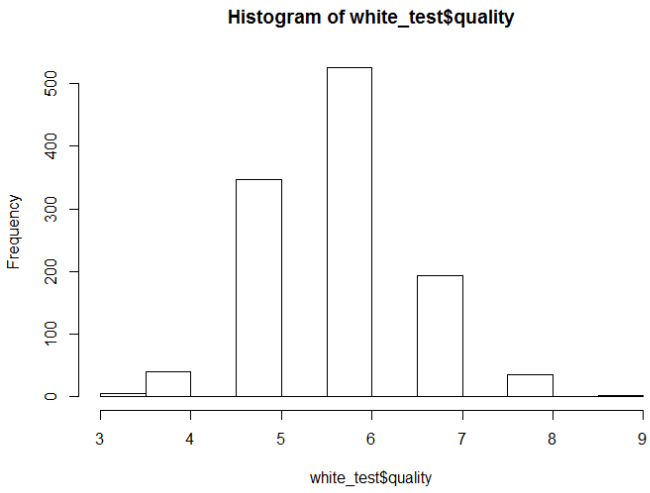
訓練データセットとテストデータセットに分割後、それぞれのデータセットにおける quality の記述統計量や分布を確認し、分割が適正であるか確認する。

```
> summary(white_train$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.887   6.000   9.000

> summary(white_test$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.847   6.000   9.000

> hist(white_train$quality)
> hist(white_test$quality)
```





品質評価値の分布が、訓練データセットとテストデータセットで似ていることが確認された。

モデルの訓練

11 の特徴量を独立変数、品質(0 から 10)を従属変数とする重回帰モデルを構築する。stats パッケージに含まれる lm() 関数を用いる（lm; linear model)。stats パッケージはデフォルトでインストールされているはずであるが、もし入っていない場合は以下のとおりインストール、ロードする。

```
> install.packages("stats")
```

```
> library(stats)
```

stats パッケージに含まれる lm()関数の構文は以下のとおりである。

m <- lm(dv ~ iv, data = mydata)

- ✓ dv 従属変数（推測したい目的変数）
- ✓ iv 独立変数（特徴量）、複数の独立変数は+でつないで並べる
- ✓ data: dv, iv 変数を含むデータフレーム

また予測の際の predict()関数の構文は以下のとおりである。

```
p <- predict(m, test)
```

- ✓ m: lm()関数で訓練されたモデル
- ✓ test テストデータを格納するデータフレーム

ここでは、white_train データフレームの quality を従属変数、それ以外のすべての変数を独立変数とする。独立変数は+で複数並べることができるが、従属変数以外のすべての変数を指定する場合はピリオド（ . ）で表現することができる。すなわち、下の 1)と 2)は同じ意味である。

1）独立変数以外のすべての変数を+でつないで並べる方法

```
> white_model <- lm(quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + density + pH + sulphates + alcohol, data=white_train)
```

2）独立変数以外のすべての変数をピリオド（ . ）で表現する方法

```
> white_model <- lm(quality ~ ., data=white_train)
```

訓練データによる学習の結果、どのような回帰モデルが生成されたかは summary() 関数で知ることができる。

```
> summary(white_model)
Call:
lm(formula = quality ~ ., data = white_train)
Residuals:
    Min      1Q  Median      3Q     Max
-3.6130 -0.4941 -0.0305  0.4586  3.0854

Coefficients:
                      Estimate   Std.Error  t value  Pr(>|t|)
(Intercept)          1.337e+02  2.055e+01   6.506 8.77e-11 ***
fixed.acidity        7.103e-02  2.344e-02   3.031  0.00245 **
volatile.acidity    -1.856e+00  1.305e-01 -14.222  < 2e-16 ***
citric.acid          3.057e-02  1.106e-01   0.277  0.78214
```

```
residual.sugar        7.590e-02  8.360e-03   9.079  < 2e-16 ***
chlorides            -9.178e-03  6.445e-01  -0.014  0.98864
free.sulfur.dioxide   5.388e-03  9.748e-04   5.527 3.47e-08 ***
total.sulfur.dioxide -2.934e-04  4.328e-04  -0.678  0.49788
density              -1.340e+02  2.086e+01  -6.424 1.49e-10 ***
pH                    6.729e-01  1.188e-01   5.662 1.61e-08 ***
sulphates             5.716e-01  1.139e-01   5.020 5.40e-07 ***
alcohol               2.202e-01  2.665e-02   8.262  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7527 on 3738 degrees of freedom
Multiple R-squared:  0.2831,    Adjusted R-squared:  0.281
F-statistic: 134.2 on 11 and 3738 DF,  p-value: < 2.2e-16
```

Intercept（切片）と 11 の特徴量について、偏回帰係数の予測値（Estimate）、標準誤差（Std. Error）、t 値、P 値が確認できる。また、*の数で、その特徴量の有意性を表現している。P 値が 0~0.001 の間であれば ***、0.001 から 0.01 の間であれば ** 、0.01 から 0.05 の間であれば * 、0.05 から 0.1 の間であれば . 、それ以上は無印となる。例えば 5%を有意水準と決めた場合、*,**,***の印が付いた特徴量が目的変数である quality に影響を与えていると解釈できる。そのほか、係数 volatile.acidity が 1 単位増えると quality が 1.86 下がる、alcohol 度数が 1 度上がると quality が 0.22 上がることなどが読み取れる。

なお、偏回帰係数の予測値の大小を比較して、どの特徴量が目的変数である品質スコアに大きな影響を与えているかを議論することはできない。それは、元のデータにおいて特徴量の間で取り得る値の範囲が大きく異なっていたり、そもそも単位が異なったりしているため、単純に数値の大小比較ができないことが理由である。そのようなことがしたい場合は、説明変数を標準化したうえで、重回帰分析を行う必要がある。標準化されたデータに基づいて重回帰分析を行った場合、得られる偏回帰係数は、標準偏回帰係数と呼ばれ、その大小関係によって、目的変数に大きな影響を与える特徴量を見出すことができる。

ただし、決定係数（Multiple R-squared）や調整済み決定係数（Adjusted R-squared）を見ると 0.2831 や 0.281 となっている。この係数は 0 から 1 の範囲の値をとり、モデルに組み入れた特徴量の分散によって目的変数の分散を説明できる割合を示している。0.2831 では訓練データによる学習（fitting）の時点で 28.3%程度しか説明できていないことになる。lm()は linear model 線形モデルであり、独立変数である特徴量と従属変数の間に線形関係を仮定しているため、実態が線形でない場合は説明力が下がる。説明力を上げるために、独立変数の分布を見て外れ値を除外したり、必要ならば対数変換して正規分布に近づけたり、などの前処理も考えられるが、ここでは行わない。

モデルの性能評価

テストデータに対して、訓練済みモデルを適用して品質を予測する。

```
> white_pred <- predict(white_model, white_test)
```

モデルの性能評価のため、モデルに基づいて予測した品質評価値と正解の品質評価値との相関を求める。

```
> cor(white_pred, white_test$quality)
[1] 0.5213139
```

相関は 0.521 であり、それほど悪くない結果といえる。また、相関による評価とは別に、予測値が正解値からどの程度離れているかという観点からモデルの性能を評価することもできる。このために平均絶対誤差関数 Mean Absolute Error function を定義し、絶対誤差を求める。

```
> MAE <- function(actual, predicted){
+     mean(abs(actual - predicted))
+ }
> MAE(white_pred, white_test$quality)
[1] 0.5853907
```

なお、左端の+記号は改行しているもののコードが継続していることを示す単なる表記である。コマンド入力中に、わざわざ+記号を入力する必要はない。

モデルの予測値と正解の品質評価値の誤差が約 0.59 である。品質の尺度が 0 から 10 であることをふまえると、このモデルはまずまずの性能を示していると言える。

SVM を用いたモデル構築

冒頭でも述べた通り、なすべきタスクは、Wine Quality Data Set のうち白ワインデータセットを利用して、11 種類の科学的特性に基づいて品質スコアを推定する機械学習モデルを構築することであり、教師あり学習の回帰問題である。本章では、アルゴリズムとして SVM を利用したモデル構築を行う。

データの準備

先の線形回帰モデルを利用したモデル構築の際に準備したデータフレーム white を用いる。

訓練データセットとテストデータセットの分割

先の線形回帰モデルを利用したモデル構築の際に、データセットの分割をすでに行っているため、これを利用する。訓練データセット white_train、テストデータセット white_test である。

モデルの訓練

11 の特徴量を独立変数、品質(0 から 10)を従属変数とする回帰モデルを SVM(Support Vector Machine)を利用して構築する。 kernlab パッケージの ksvm()関数を用いる。まず、kernlab パッケージをインストール、ロードする。

```
> install.packages("kernlab")
> library(kernlab)
```

kernlab パッケージに含まれる ksvm()関数の構文は以下のとおりである。

m <- ksvm(dv ~ iv, data = mydata, kernel = “rbfdot”, C = 1)

- ✓ dv 従属変数（推測したい目的変数）
- ✓ iv 独立変数（特徴量）、複数の独立変数は+でつないで並べる
- ✓ data: dv, iv 変数を含むデータフレーム
- ✓ kernel: “rbfdot” Gaussian RBF, “vanilladot” 線形など
- ✓ C: コスト：ソフトマージンのペナルティ。大きくするとマージンが狭くなる

また予測の際の predict()関数の構文は以下のとおりである。

```
p <- predict(m, test, type = “response”)
```

- ✓ m: ksvm()関数で訓練されたモデル
- ✓ test テストデータを格納するデータフレーム
- ✓ type 戻り値のタイプ。予測値（”response”）か予測の未調整の確率（”probabilities”）

ここでは、white_train データフレームの quality を従属変数、それ以外のすべての変数を独立変数として、SVM モデルの学習を行う。カーネルは Gaussian カーネル、コストは 1 とする。

```
> white_svm_rbf <- ksvm(quality ~ ., data = white_train, kernel = "rbfdot", C = 1)
```

訓練済みモデルのパラメータやモデルの適合状況に関する情報はモデルを格納した変数名を入力することで取得することができる。

```
> white_svm_rbf
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0813307429562604

Number of Support Vectors : 3239

Objective Function Value : -1727.689
Training error : 0.520094
```

モデルの性能評価

テストデータに対して、訓練済みモデルを適用して品質を予測する。

```
> white_pred_svm_rbf <- predict(white_svm_rbf, white_test)
> cor(white_pred_svm_rbf, white_test$quality)
          [,1]
[1,] 0.6177452
```

予測した品質と正解品質との相関は 0.617 であり、線形の重回帰モデルより優れている。

絶対誤差についても確認しておく。

```
> MAE(white_pred_svm_rbf, white_test$quality)
[1] 0.528741
```

重回帰モデルの絶対誤差 0.58 よりも改善されていることが確認された。

教師なし学習（次元削減）演習ガイド

タスク

Iris Species Data Set （http://archive.ics.uci.edu/ml/datasets/iris） は 3 種類の Iris（アヤメ）各 50 個体について、Sepal（萼、がく）の長さと幅、Petal（花弁）の長さと幅を計測した結果をまとめたデータセットである。Iris Species Data Set は R においては組み込みデータセットとして用意されており、str(iris)でそのデータ構造を確認することができる。1 行が 5 つの変数からなり、1 個体ごとに Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species が並んでいる。最後の Species は Iris の種類のことであり、Iris setosa, Iris versicolor, Iris virginica のいずれかである。

教師あり学習のタスクの場合、萼の長さと幅および花弁の長さと幅からアヤメの種類を識別するモデルを構築することが目的となり、この際に Species を教師データとして用いる。しかし、ここでは Species を用いずに、萼の長さと幅および花弁の長さと幅からアヤメの特徴を上手に表現できるような少数の合成変数を求めるタスクを考える。

このタスクは、本来の特徴空間が 4 次元であるのに対し、それよりも低い次元で、たとえば合成変数 2 つで張られる 2 次元の特徴空間を求めることから、次元削減と呼ばれる。次元削減は、Species のような正解データを用いないため、教師なし学習の一種である。本演習では、次元削減の手法として主成分分析を取り扱う。

データの準備

新規プロジェクト（ディレクトリ名は iris とする）を立ち上げ、新規スクリプトを開いておく。Iris Species Data Set は、R には組み込みデータセットとしてあらかじめ用意されている。

str(iris)でそのデータ構造を確認することができる。

```
> str(iris)
'data.frame':  150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

5 つの変数を持つ 150 個体のデータからなるデータフレームである。、1 個体ごとに Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species が並んでいる。Species は 3 水準の因子ベクトルであり、水準はそれぞれ setosa, virginica, versicolor である。1 種類につき 50 個体からなることが table(iris$Species)により確かめられる。

```
> table(iris$Species)
    setosa versicolor  virginica
        50         50         50
```

ただし、ここでは萼の長さと幅および花弁の長さと幅といった計測値だけから少数の合成変数を求めることが

目的である。このため、Sepcies（5 列目）は分離しておく。

```
> iris_data <- iris[,-5]
> iris_class <- iris[,5]
> rm(iris_ans)
> str(iris_class)
 Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> str(iris_data)
'data.frame':  150 obs. of  4 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
```

データの範囲を把握するため、基本統計量を summary()関数で求める。

```
> summary(iris_data)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

特徴量の単位はすべて cm である。取り得る値の範囲が特徴量によって多少異なることがわかる。なお、あとで紹介する主成分分析のための関数 prcomp()には引数 scale があり、データの標準化が必要な場合はその時に scale=TRUE を指定すればよく、あらかじめ標準化の処理を行う必要はない。

モデルの訓練

主成分分析の実装 prcomp()関数を利用する。prcomp()は stas パッケージに含まれており、stats パッケージ自体 R インストール時にデフォルトで含まれている。見当たらない場合は install.packages("stats")して、library(stats)すればよい。

stats パッケージの prcomp()関数は、引数に単に主成分分析の対象となるデータフレームを指定すれば十分であるが、scale 引数を True に指定すると、標準偏差を 1 とする標準化を行うことが可能である。scale 引数を指定しない場合、デフォルト値（False）となり、標準化は行われず中心化（平均を 0 にする操作）だけ行われる。

pca <- prcomp(mydata, scale. = False)

- ✓ mydata 主成分分析の対象となるデータフレーム
- ✓ scale オプション。True 指定で分析前にデータを標準化する。デフォルトは False

prcomp()の戻り値は、主成分分析の情報が格納されたオブジェクトである。次のように分析結果を確認することができる。

pca$sdev：主成分の標準偏差（共分散・相関行列の固有値の平方根）

pca$rotation：変数の負荷量行列

pca$center：特徴空間における元データの中心。分析前にこれを各データ点から引いてデータ中心を原点にシフトしている（中心化）。

pca$scale：標準化した場合は標準化に用いた標準偏差。標準化していない場合は FALSE

pca$x：主成分スコア。中心化（指定した場合は標準化）され、rotation 行列で回転されたデータ。cov(x)は対角行列となる。

prcomp()の第一引数に iris_data を与えて主成分分析を行う。中心化（平均を原点にシフト）だけを行う場合はオプションの scale は指定しなくてよい（False)。平均を 0 に、標準偏差を 1 にする標準化を行う場合は scale. = True を指定する。取り得る値の範囲が特徴量によって大きく異なる場合や、単位が異なる場合は標準化を行うべきである。ここではどちらでもよいが、標準化した場合の例を示す。

```
> irisPCA_st <- prcomp(iris_data, scale. = T)
```

モデルの性能評価

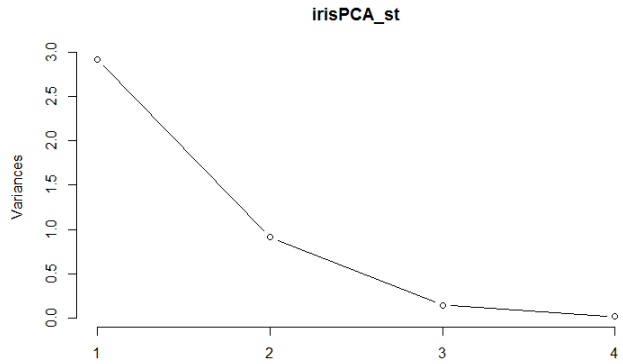
モデルの性能評価のため、主成分分析結果についての情報が格納されたオブジェクト irisPCA_st を調べる。結果の概要は戻り値に対する summary()で確認できる。

```
> summary(irisPCA_st)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

第 1 主成分(PC1)から順に並んでおり、各主成分に対する標準偏差、寄与率、累積寄与率が確認できる。今の場合第 1 主成分までデータの分散の 73.0%が、第 2 主成分までで 95.8%が説明できることが読み取れる。

また、screeplot()関数を使えば、主成分毎の分散をプロットして、どの主成分まで採用すればモデルが十分な説明力を持つか視覚的に検討することもできる。

```
> screeplot(irisPCA_st, type = "lines")
```



散布図描画

第 1 主成分（PC1）、第 2 主成分（PC2）の組み合わせが Iris データを上手に説明できるような変数の組になっているか、散布図を描いて確認したい。二つの変数で張られる二次元空間（平面）で描いた散布図で、アヤメの三種類がうまく分離できるようになっていることが望ましい。

主成分スコアは irisPCA_st$x に格納されている。

```
> str(irisPCA_st$x)
 num [1:150, 1:4] -2.26 -2.07 -2.36 -2.29 -2.38 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"

> head(irisPCA_st$x)
          PC1        PC2         PC3          PC4
[1,] -2.257141 -0.4784238  0.12727962  0.024087508
[2,] -2.074013  0.6718827  0.23382552  0.102662845
[3,] -2.356335  0.3407664 -0.04405390  0.028282305
[4,] -2.291707  0.5953999 -0.09098530 -0.065735340
[5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
[6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

150 行 4 列のデータであり、行は各データ点、列はそのデータ点の第 1 主成分から順に並べた主成分スコアである。

散布図描画のため、データフレームに変換したのち、Species を戻しておく。

```
> z <- as.data.frame(irisPCA_st$x)
> z$Species <- iris_class
> str(z)
'data.frame':  150 obs. of  5 variables:
```

```
 $ PC1    : num  -2.26 -2.07 -2.36 -2.29 -2.38 ...
 $ PC2    : num  -0.478 0.672 0.341 0.595 -0.645 ...
 $ PC3    : num  0.1273 0.2338 -0.0441 -0.091 -0.0157 ...
 $ PC4    : num  0.0241 0.1027 0.0283 -0.0657 -0.0358 ...
 $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

第 2 主成分までを考慮して散布図として plot する。ここでは、標準ライブラリの plot よりも綺麗な plot を描くことができる ggplot2 パッケージを利用する。

```
> install.packages("ggplot2")
> library(ggplot2)
```

ggplot()関数の第 1 引数にデータフレームを指定し、第 2 引数に aes()を用いて x 軸と y 軸に取りたい成分、ここでは PC1 と PC2 を指定する。これにより描画平面が規定される。

これに加え、+geom_point()を指定することで描画平面に点を散布することができる。この際、aes(colour = Species)を指定することで、アヤメの種類に応じて点の色を色分けすることができる。

```
> ggplot(z,aes(x=PC1,y=PC2))+geom_point(aes(colour = Species))
> ggsave(file="pca.png")
```

比較のため、元データ（iris）の 4 変数から任意の 2 変数（Sepal.Length, Sepal.Width）を選んだ場合の散布図を描く。

```
> ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width))+geom_point(aes(colour = Species))
> ggsave(file="original.png")
```

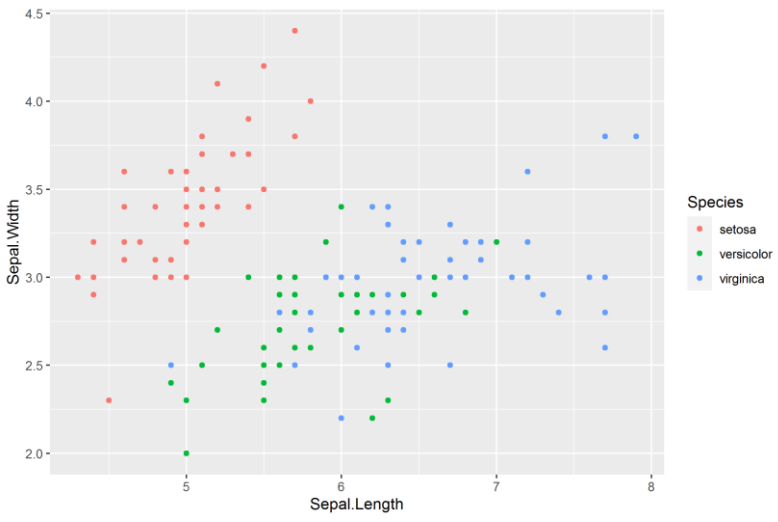


図 3　オリジナルデータから任意の 2 成分を選んで散布図

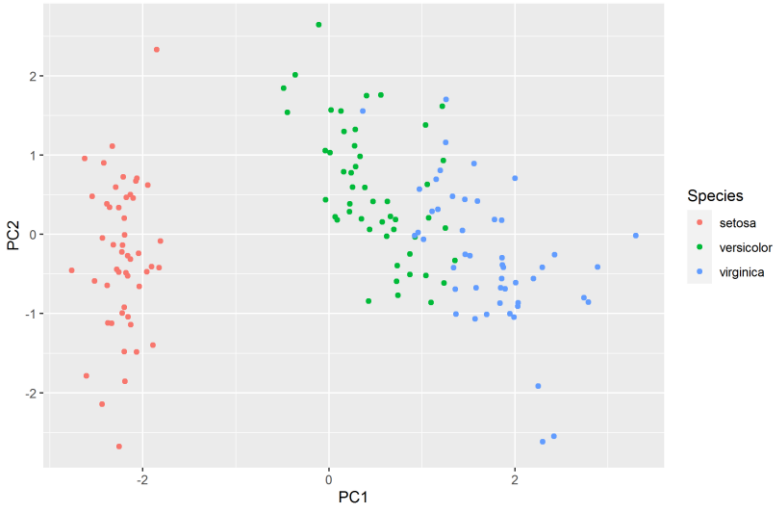


図 4　主成分分析後 PC1 と PC2 で描いた散布図

オリジナルデータを用いて適当に選んだ 2 成分で描いた散布図でも setosa, versicolor, virginica はそれぞれ固まって存在している。しかし、これら 3 品種を Sepal.Length の軸で見ても、Sepal.Width の軸で見ても綺麗に分離できそうにない。

これに対して、主成分分析を行って PC1 と PC2 で描いた散布図の場合、PC1 方向だけで見た場合でも setosa, versicolor, virginica が分離できそうである。つまり、第 1 主成分 PC1 が 3 つの品種を識別する変数となっていると言える。

なお、PC1 が元の変数をどのように合成した変数となっているかは irisPCA_st$rotation 主成分負荷量行列を見ればわかる。

```
> irisPCA_st$rotation
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

これによると、第 1 主成分 PC1 は 0.52 *Sepal.Length -0.269 * Sepal.Width +0.58* Petal.Length + 0.56* Petal.Width で構成されている。