～手を動かしながら学ぶ！～

エンジニアのための データサイエンス・ハンズオン

株式会社リクルートコミュニケーションズ
尾崎 隆 (Takashi J. OZAKI, Ph. D.)

2015/2/4

今更ですが、自己紹介を…

ブログやってます

ついでに、宣伝を…

夏に本出しました（もうそろそろ売り切った気がしないでもない）

序

今回のハンズオンの目的・目指すもの

今回のハンズオンが目指すもの

- データ分析の「雰囲気」を知る
- どんな「手法」があるかを眺める
- 実際にエンジニアとして実装する時に何をすれば良いのかイメージできるようになってもらう
  - このハンズオンではRを使いますが、実際には他の言語（Python, Javaなど）で実装するはず

0

今回のハンズオンで最低限必要なR上でのデータ操作

最低限これだけ覚えておきましょう(1)

データを外部から読み込む場合

```
# RStudioであれば右上ウィンドウの “Import Dataset” で対話的にやれる

# 外部ファイルを読み込む
data<-read.table(file,header=T,sep=',', ...)

# クリップボードから読み込む(Win)
data<-read.table("clipboard",header=T)
```

最低限これだけ覚えておきましょう(2)

データテーブルの中身を操作する

```
# as.factor関数でnumeric型からfactor型に変換する
d$cat<-as.factor(d$cat)

# 行・列を抽出or削除する
d<-d[,-5] # 5列目を削除する
d<-d[,1:15] # 1-15列目のみ抽出する
d<-d[,-c(2,4,7)] # 2,4,7列目を削除する
d<-d[2,] # 2行目だけを抽出する
d<-d[idx,] # which関数などで特定した行のみ抽出する
d<-d[d$cat=='hoge',] # cat列がhogeの行のみ抽出する
```

その他については…

適宜ハンズオンの途中で説明していきます ~~（手抜き）~~

サンプルデータセットについて

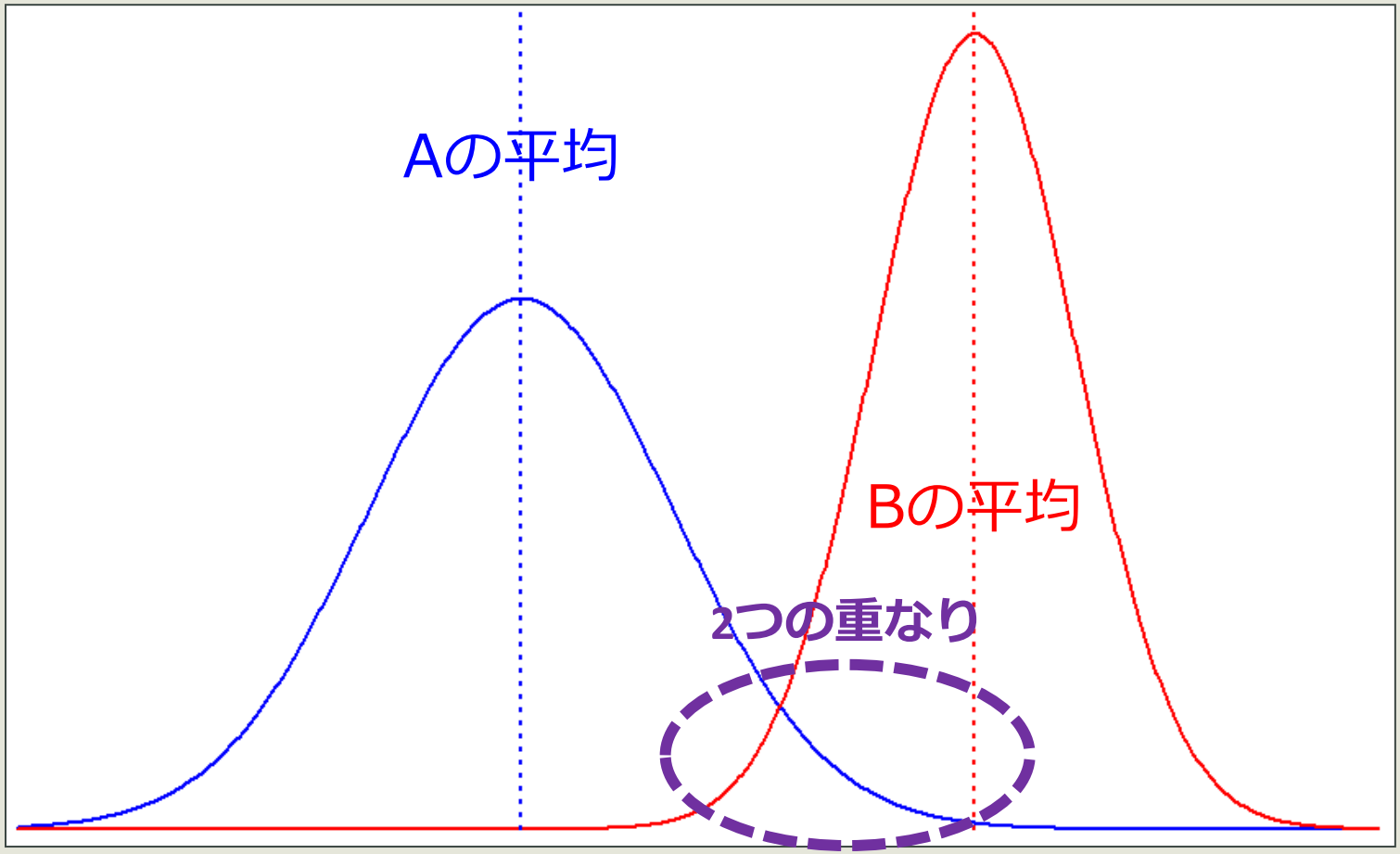
以下のGitHubリポジトリをクローンしてください

https://github.com/ozt-ca/tjo.hatenablog.samples/tree/master/r_samples/public_lib/hikalabo20150205

1

統計学的検定をやってみよう

「検定」とは何か



ラフに言えば「2つの分布の重なりに基づいて、2つの変数が異なるかどうか」を統計学的に判断するということ

ここでは以下の統計学的検定を扱います

- t検定（Welchの検定）
- カイ二乗検定
- Wilcoxonの順位和検定
- おまけ：いつまでサンプルを集めれば良い？

t検定（Welchの検定）

```
# ワーキングフォルダを移動する
setwd(“./hptest")
# データを読み込む
d1<-read.table("hyptest1.csv",header=T,sep=',')
d2<-read.table("hyptest2.csv",header=T,sep=',')
# データの先頭部を眺める
head(d1)
head(d2)
# 箱ひげ図で全体像を見る
boxplot(d1)
boxplot(val~id,d2)
# t検定を行ってみる
t.test(d1$x1,d1$x2)
t.test(val~id,d2)
```

カイ二乗検定（独立性の検定）

```
# データを読み込む
mx1<-read.table("chisq1.csv",sep=',')
mx1<-as.matrix(mx1)
mx2<-read.table("chisq2.csv",sep=',')
mx2<-as.matrix(mx2)
mx3<-read.table("chisq3.csv",sep=',')
mx3<-as.matrix(mx3)
mx4<-read.table("chisq4.csv",sep=',')
mx4<-as.matrix(mx4)
# カイ二乗検定orフィッシャーの正確確率検定
chisq.test(mx1)
fisher.test(mx1)
chisq.test(mx2)
fisher.test(mx2)
chisq.test(mx3)
fisher.test(mx3)
chisq.test(mx4)
fisher.test(mx4)
```

順位和検定（Mann-Whiteny / Wilcoxon検定）

```
# データを読み込む
d<-read.table("ranksum.csv",header=T,sep=',')
# 箱ひげ図で全体像を見る
boxplot(d)
# t検定を行ってみる
t.test(d$x1,d$x2)
# Wilcoxon検定を行ってみる
wilcoxon.test(d$x1,d$x2)
```

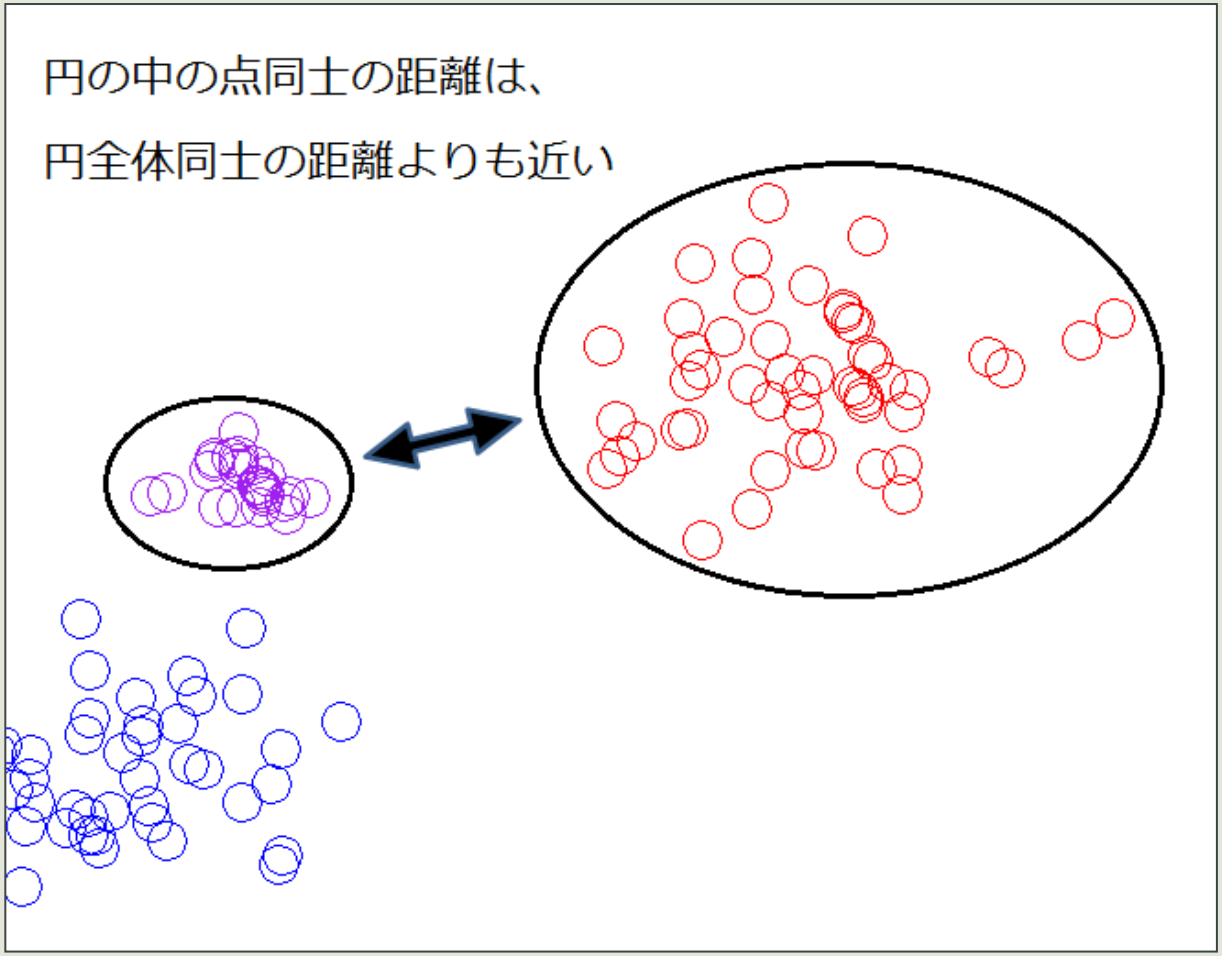
サンプルサイズと効果量の関係を知る

```
# データを読み込む
d<-read.table("effectsize_samplesize.csv",header=T,sep=',')
# 箱ひげ図で全体像を見る
boxplot(d)
# 30サンプルでt検定を行ってみる
t.test(d$x1[1:30],d$x2[1:30])
# 全サンプルで検定を行ってみる
t.test(d$x1,d$x2)
# 効果量を計算してみる
e_size<-(mean(d$x2)-mean(d$x1))/3000
```

2

クラスタリングをやってみよう

クラスタリングとは



サンプル同士の「距離」に基づいて似たものをまとめていくこと

サンプルデータについて

“The data was obtained from 20 different perfumes by using a handheld odor meter(OMX-GR sensor).

Names of these perfumes are: ajayeb, ajmal, amreaj, aood, asgar_ali, bukhoor, burberry, dehenalaod, junaid, kausar, rose, solidmusk, TeaTreeOil, raspberry, RoseMusk, strawberry, constrected2, carolina_herrera, oudh_ma'alattar, constrected1.

Each column represent a measurement and there were 28 takes (one each second) .“

(https://archive.ics.uci.edu/ml/datasets/Perfume+Data)

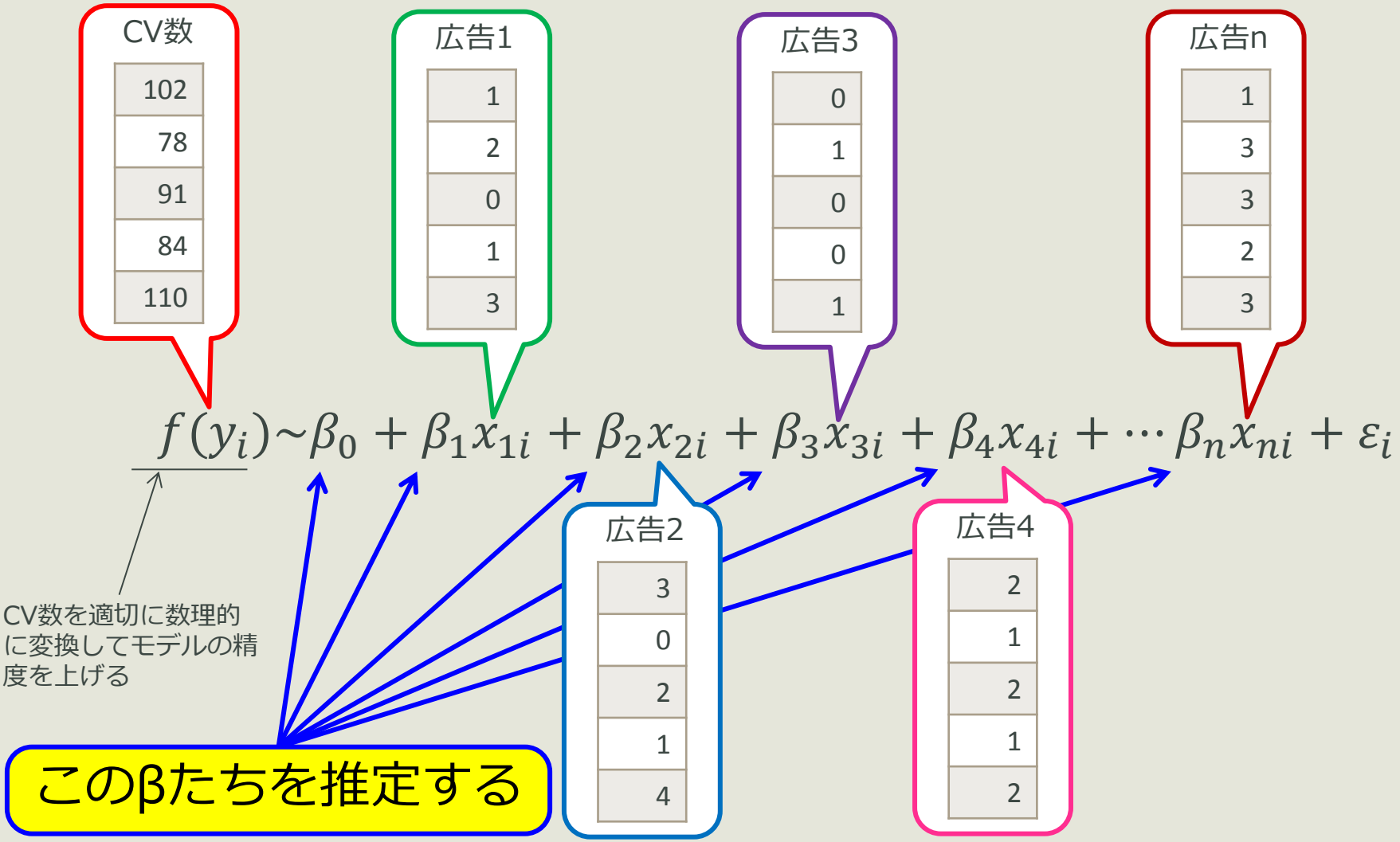
Ward法による階層的クラスタリングを実践してみる

```
# データを読み込む
d<-read.table("perfume.csv",header=T,sep=',')
mx<-as.matrix(d[,-1]) # マトリクス形式に直す
row.names(mx)<-as.vector(d[,1]) # 行名を与える
# 距離行列を算出する
mx.dist<-dist(mx)
# Ward法で階層的クラスタリングを演算する
# R 3.1.0以上では “ward.D2” を選択すること
mx.hcl<-hclust(mx.dist,method = "ward")
# デンドログラム（樹状図）をプロットしてみる
plot(mx.hcl)
```

3

統計モデリングをやってみよう

正規線形回帰モデル（いわゆる重回帰分析）



サンプルデータについて

For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

(https://archive.ics.uci.edu/ml/datasets/Wine+Quality)

重回帰分析による統計モデリングを行ってみる(1)

```
# データを読み込む
dr_train<-read.table("wine_red_train.csv",header=T,sep=',')
dr_test<-read.table("wine_red_test.csv",header=T,sep=',')
dw_train<-read.table("wine_white_train.csv",header=T,sep=',')
dw_test<-read.table("wine_white_test.csv",header=T,sep=',')
# データを眺める
head(dr_train)
head(dw_train)
# 目的変数の分布を確認する
# ここで正規分布に従っていない場合は別の方法が必要
# （例えば一般化線形モデルなど）
hist(dr_train$quality)
hist(dw_train$quality)
```

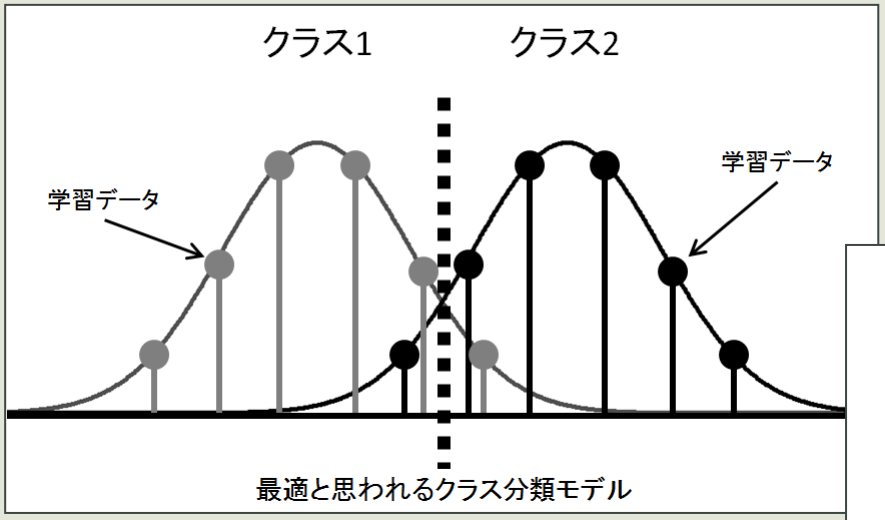
重回帰分析による統計モデリングを行ってみる(2)

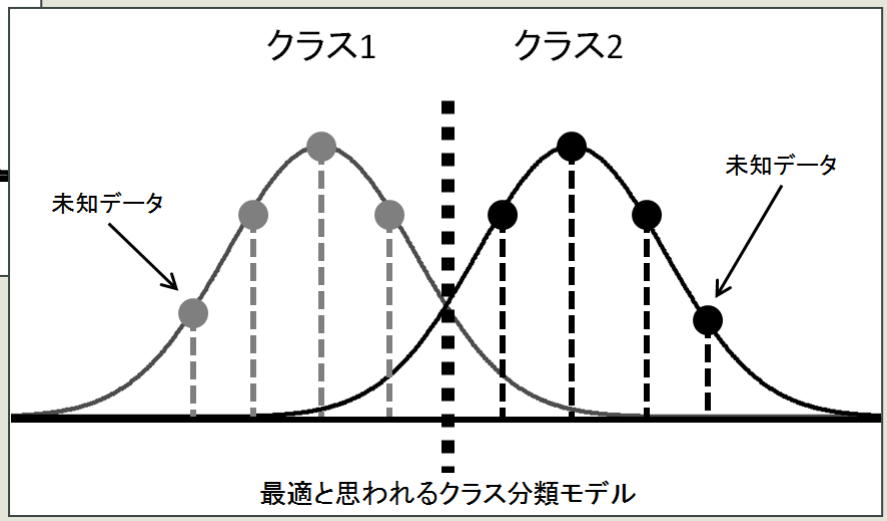
```
# lm関数で線形回帰モデルを推定する
dr.lm<-lm(quality~.,dr_train) # 赤ワイン
dw.lm<-lm(quality~.,dw_train) # 白ワイン
# 推定した線形回帰モデルの内容を確かめる
summary(dr.lm)
summary(dw.lm)
# 推定結果の当てはまりをチェックする
cor(dr_test$quality,predict(dr.lm,newdata=dr_test[,-12]))^2
plot(dr_test$quality,predict(dr.lm,newdata=dr_test[,-12]))
cor(dw_test$quality,predict(dw.lm,newdata=dw_test[,-12]))^2
plot(dw_test$quality,predict(dw.lm,newdata=dw_test[,-12]))
```

4

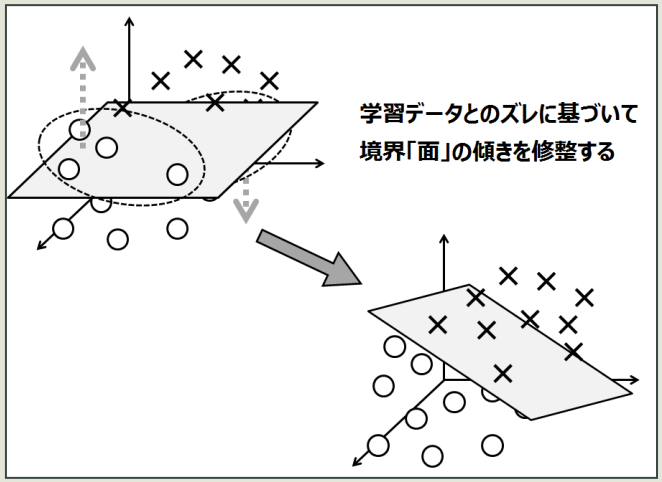
機械学習をやってみよう

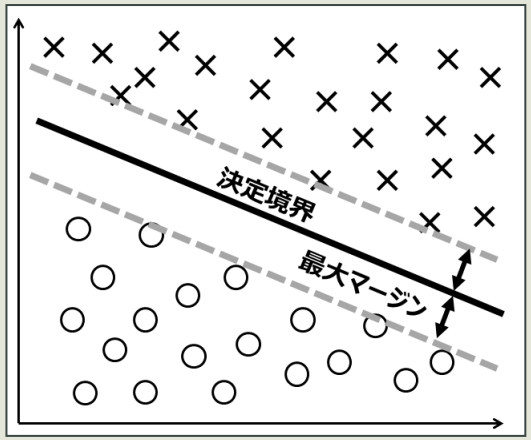
機械学習とは？

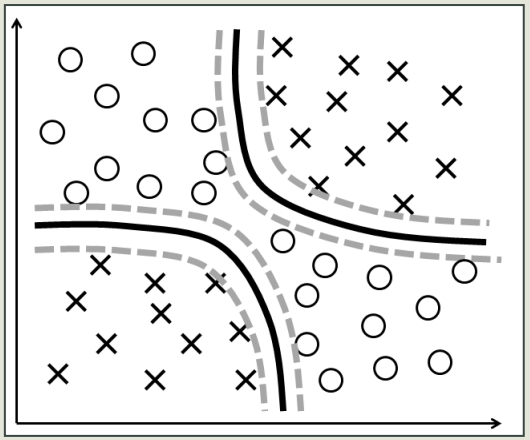

クラス1
クラス2
学習データ
学習データ
最適と思われるクラス分類モデル

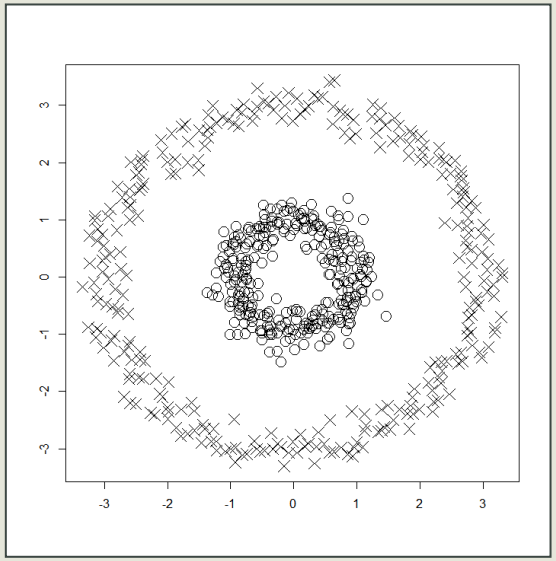

クラス1
クラス2
未知データ
未知データ
最適と思われるクラス分類モデル

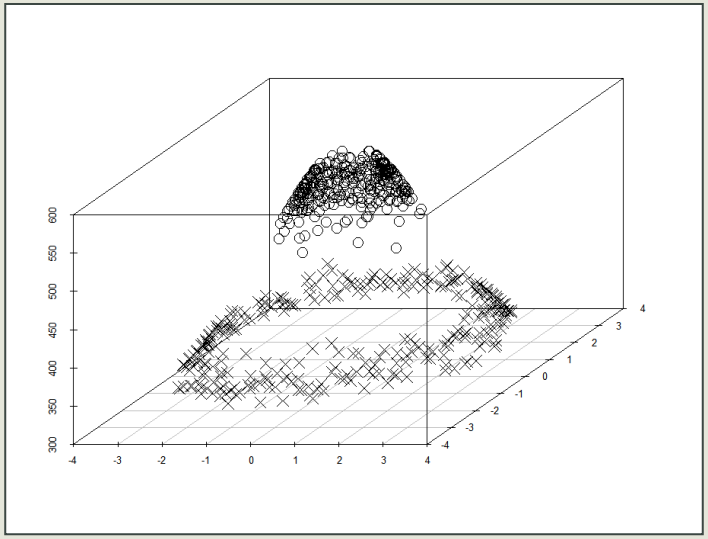
SVMとは

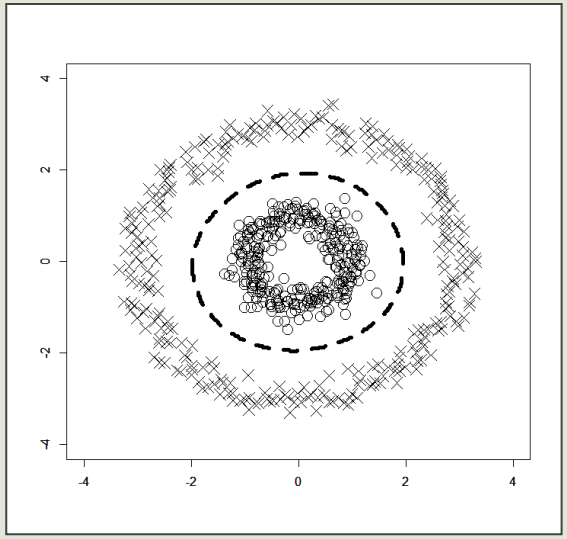




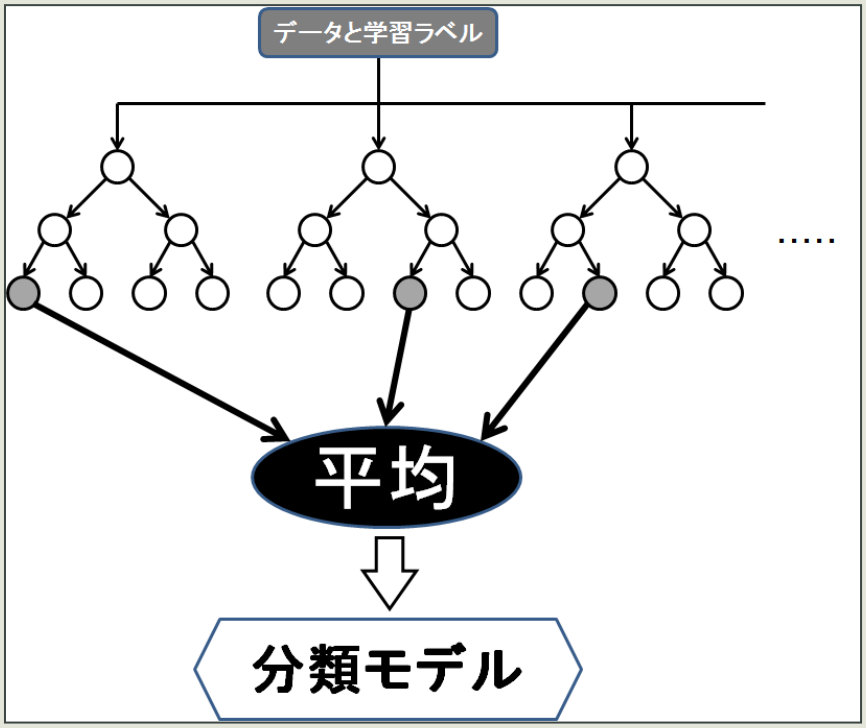


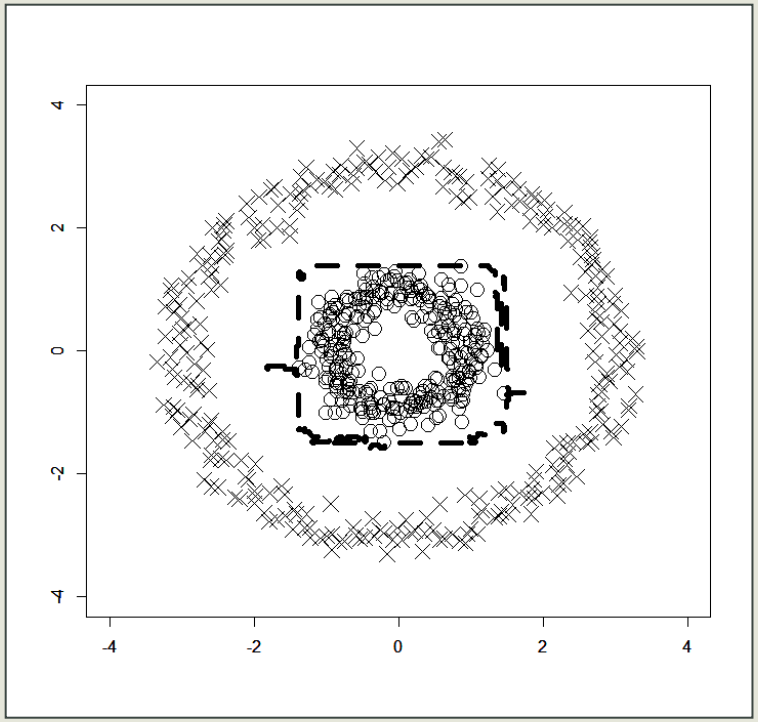






ランダムフォレストとは





サンプルデータ “chess” について

“The format for instances in this database is a sequence of 37 attribute values. Each instance is a board-descriptions for this chess endgame. The first 36 attributes describe the board. The last (37th) attribute is the classification: "win" or "nowin". There are 0 missing values. A typical board-description is

f,f,f,f,f,f,f,f,f,f,f,f,l,f,n,f,f,t,f,f,f,f,f,f,f,t,f,f,f,f,f,f,f,t,t,n,won

The names of the features do not appear in the board-descriptions. Instead, each feature correponds to a particular position in the feature-value list. For example, the head of this list is the value for the feature "bkblk".

The following is the list of features, in the order in which their values appear in the feature-value list:

[bkblk,bknwy,bkon8,bkona,bkspr,bkxbq,bkxcr,bkxwp,blxwp,bxqsq,cntxt,dsopp,dwipd,hdchk,katri,mulch,qxmsq,r2ar8,reskd,reskr,rimmx,rkxwp,rxmsq,simpl,skach,skewr,krxp,spcop,stlmt,thrsk,wkcti,wkna8,wknck,wkovl,wkpos,wtoeg]

In the file, there is one instance (board position) per line.”

(**https://archive.ics.uci.edu/ml/machine-learning-databases/chess/king-rook-vs-king-pawn/kr-vs-kp.names**)

“chess”データに対して機械学習を行ってみる(1)

```
# データを読み込む
# 学習データ
d_train<-read.table("chess_train.csv",header=T,sep=',')
# テストデータ
d_test<-read.table("chess_test.csv",header=T,sep=',')
# パッケージを読み込む
library(randomForest)
library(e1071)
# データを眺める
summary(d_train)
summary(d_test)
head(dr_train)
head(dw_train)
```

“chess”データに対して機械学習を行ってみる(2)

```
# パラメータチューニングをしながら機械学習を行う
# SVM
d.tune<-tune.svm(label~.,data=d_train)
d.tune$best.model
d.svm<-
svm(label~.,d_train,cost=d.tune$best.model$cost,gamma=d.tune$best.model$gamma)
# ランダムフォレスト
tuneRF(d_train[,-36],d_train[,36],doBest=T)
d.rf<-randomForest(label~.,d_train,mtry=20)
```

“chess”データに対して機械学習を行ってみる(3)

```
# 推定してみたモデルで予測を行ってみる
# テストデータの1サンプル目を眺めてみる
d_test$label[1]
# SVMで予測してみる
predict(d.svm,newdata=d_test[1,-36])
# ランダムフォレストで予測してみる
predict(d.rf,newdata=d_test[1,-36])
# それぞれテストデータ全体に対して予測してみる
# Confusion matrix（予測結果の比較行列）を出してみる
table(d_test$label,predict(d.svm,newdata=d_test[,-36]))
table(d_test$label,predict(d.rf,newdata=d_test[,-36]))
```

サンプルデータ “tennis” について

“Player 1 Name of Player 1
Player 2 Name of Player 2
Result Result of the match (0/1) - Referenced on Player 1 is Result = 1 if Player 1 wins (FNL.1>FNL.2)
FSP.1 First Serve Percentage for player 1 (Real Number)
FSW.1 First Serve Won by player 1 (Real Number)
SSP.1 Second Serve Percentage for player 1 (Real Number)
SSW.1 Second Serve Won by player 1 (Real Number)
ACE.1 Aces won by player 1 (Numeric-Integer)
DBF.1 Double Faults committed by player 1 (Numeric-Integer)
WNR.1 Winners earned by player 1 (Numeric)
UFE.1 Unforced Errors committed by player 1 (Numeric)
BPC.1 Break Points Created by player 1 (Numeric)
BPW.1 Break Points Won by player 1 (Numeric)
NPA.1 Net Points Attempted by player 1 (Numeric)
NPW.1 Net Points Won by player 1 (Numeric)
TPW.1 Total Points Won by player 1 (Numeric)
ST1.1 Set 1 result for Player 1 (Numeric-Integer)
ST2.1 Set 2 Result for Player 1 (Numeric-Integer)
ST3.1 Set 3 Result for Player 1 (Numeric-Integer)
ST4.1 Set 4 Result for Player 1 (Numeric-Integer)
ST5.1 Set 5 Result for Player 1 (Numeric-Integer)
FNL.1 Final Number of Games Won by Player 1 (Numeric-Integer)
FSP.2 First Serve Percentage for player 2 (Real Number)
FSW.2 First Serve Won by player 2 (Real Number)
SSP.2 Second Serve Percentage for player 2 (Real Number)
SSW.2 Second Serve Won by player 2 (Real Number)
ACE.2 Aces won by player 2 (Numeric-Integer)
DBF.2 Double Faults committed by player 2 (Numeric-Integer)
WNR.2 Winners earned by player 2 (Numeric)
UFE.2 Unforced Errors committed by player 2 (Numeric)
BPC.2 Break Points Created by player 2 (Numeric)
BPW.2 Break Points Won by player 2 (Numeric)
NPA.2 Net Points Attempted by player 2 (Numeric)
NPW.2 Net Points Won by player 2 (Numeric)
TPW.2 Total Points Won by player 2 (Numeric)
ST1.2 Set 1 result for Player 2 (Numeric-Integer)
ST2.2 Set 2 Result for Player 2 (Numeric-Integer)
ST3.2 Set 3 Result for Player 2 (Numeric-Integer)
ST4.2 Set 4 Result for Player 2 (Numeric-Integer)
ST5.2 Set 5 Result for Player 2 (Numeric-Integer)
FNL.2 Final Number of Games Won by Player 2 (Numeric-Integer)
Round Round of the tournament at which game is played (Numeric-Integer) ”

(**https://archive.ics.uci.edu/ml/datasets/Tennis+Major+Tournament+Match+Statistics**)

“tennis”データに対して機械学習を行ってみる(1)

```
# データを読み込む
# 学習データ（USオープン2013年男子）
dm<-read.table("tjo_usopen13men.csv",header=T,sep=',')
# テストデータ（USオープン2013年女子）
dw<-read.table("tjo_usopen13women.csv",header=T,sep=',')
names(dm)
summary(dm)
# Result列をfactor型に直す
dm$Result<-as.factor(dm$Result)
dw$Result<-as.factor(dw$Result)
```

“tennis”データに対して機械学習を行ってみる(2)

```
# SVMでチューニングしながら学習させる
dm.tune<-tune.svm(Result~.,data=dm)
dm.tune$best.model
dm.svm<-
svm(Result~.,dm,cost=dm.tune$best.model$cost,gamma=dm.tune$best.model$gamma)
# ランダムフォレストでチューニングしながら学習させる
tuneRF(dm[,-1],dm[,1],doBest=T)
dm.rf<-randomForest(Result~.,dm,mtry=3)
table(dw$Result,predict(dm.svm,newdata=dw[,-1]))
table(dw$Result,predict(dm.rf,newdata=dw[,-1]))
```

“tennis”データに対して機械学習を行ってみる(3)

```
# 不適切なパラメータを削除して別のデータセットを作る
dm2<-dm[,-c(2,3)]
dw2<-dw[,-c(2,3)]
# 以下繰り返し
dm2.tune<-tune.svm(Result~.,data=dm2)
dm2.tune$best.model
dm2.svm<-
svm(Result~.,dm2,cost=dm2.tune$best.model$cost,gamma=dm
2.tune$best.model$gamma)
tuneRF(dm2[,-1],dm2[,1],doBest=T)
dm2.rf<-randomForest(Result~.,dm2,mtry=5)
table(dw$Result,predict(dm2.svm,newdata=dw2[,-1]))
table(dw$Result,predict(dm2.rf,newdata=dw2[,-1]))
```

機械学習についてはPythonなどでも同じライブラリが使える

- SVM
  - R: {e1071}パッケージ
  - Python, Java, C++など：LIBSVM
    http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- ランダムフォレストなど
  - scikit-learnパッケージが著名

ということで…

さらに学びたい方は是非拙著をどうぞ！（相変わらず宣伝）