ランダムフォレストと.632 推定値

2015.08.26　　株式会社　応用数理研究所　　佐々木俊久

１．ランダムフォレスト　R 関数

Rの randomForest ライブラリー内　randomForest 関数

```
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
        mtry=if (!is.null(y) && !is.factor(y))
        max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
        replace=TRUE, classwt=NULL, cutoff, strata,
        sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
        nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
        maxnodes = NULL,
        importance=FALSE, localImp=FALSE, nPerm=1,
        proximity, oob.prox=proximity,
        norm.votes=TRUE, do.trace=FALSE,
        keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
        keep.inbag=FALSE, ...)
```

２．ランダムフォレストの手順

(1) 学習用データセット（データ数ｎ）から sampsize 個のサンプルを作成する。

サンプリング抽出方法(replace)

a)　replace=TRUE

サンプリング抽出データセットと同じｎ回の復元(with replacement)抽出

データの重複あり

b)　replace = FALSE

ｎより少ない数の非復元(without replacement)抽出

データの重複なし

指定なし時　.632* n 個

サンプリングされたデータで学習し，残りのデータは(3)での検証用になる。

(2) データセットから mtry 個の変数をサンプリングする。

(3) 決定木(tree)を作る

(4) (1)～(3)を繰り返す(ntree 回)

(5) (4)で量産された決定木のすべてに対して，予測対象データを入れる

(6) (5)の結果から多数決(識別)，あるいは平均をとり(回帰)，これを予測結果とする。

３．.632 推定量[1]

.632 推定量では，まず元のサンプルサイズと同じ回数の復元抽出を行い，1 回目の Bootstrap サンプルとし，このデータを用いて予測モデルを構築する。構築されたモデルを用いて，Bootstrap サンプルとして抽出されなかった対象者の予測を行い，予測精度(エラー率)を評価する。この操作を繰り返し，平均を取ることによって，Bootstrap に基づくエラー率の推定値とする。

４．.632 の由来

復元抽出を行った場合，ある対象者のデータが少なくとも 1 回以上抽出される確率は，

$$1-(1-\frac{1}{n})^n$$

で，元のサンプルサイズ n が大きいとき

$$\lim_{n\to\infty}(1-(1-\frac{1}{n})^n)=1-e^{-1}=1-0.368=0.632$$

になります。

参考　$\lim_{n\to\infty}((1+\frac{1}{n})^n)=e$

５．R(S-PLUS)でのシミュレーション

(1) 方法 1

sample 関数(replace=T)を使用して，1:n を復元ありで抽出した時の，ある値が選ばれる回数。　５００回行っての結果（確率）

```
n <- 100
res<-matrix(0,500,n)
for(i in 1:500) {
 ii <- unique(sample(1:n, n, replace=T))
 res[i,ii] <- 1
}
 apply(res, 2, sum)　# 各数値 1:n の抽出回数
```

[1] http://www.sascom.jp/download/pdf/usergroups11_A-10.pdf
マイクロアレイデータによる予後予測モデル構築における .632+推定量によるエラー率の補正と ROC 曲線解析

```
 [1] 313 315 309 303 291 336 314 318 338 303 317 332 338 319 316 298 307 324
[19] 303 317 311 310 329 315 308 317 323 327 312 315 305 311 318 331 293 330
[37] 305 344 332 341 318 321 325 317 329 309 327 314 311 323 301 314 315 318
[55] 329 307 313 312 318 320 299 322 311 325 321 345 320 321 311 304 316 341
[73] 309 301 314 326 311 326 329 327 329 302 302 332 311 326 321 330 324 335
[91] 314 291 321 311 323 322 292 316 313 302

apply(res, 2, sum)/500   # 抽出確率
 [1] 0.626 0.630 0.618 0.606 0.582 0.672 0.628 0.636 0.676 0.606 0.634 0.664
[13] 0.676 0.638 0.632 0.596 0.614 0.648 0.606 0.634 0.622 0.620 0.658 0.630
[25] 0.616 0.634 0.646 0.654 0.624 0.630 0.610 0.622 0.636 0.662 0.586 0.660
[37] 0.610 0.688 0.664 0.682 0.636 0.642 0.650 0.634 0.658 0.618 0.654 0.628
[49] 0.622 0.646 0.602 0.628 0.630 0.636 0.658 0.614 0.626 0.624 0.636 0.640
[61] 0.598 0.644 0.622 0.650 0.642 0.690 0.640 0.642 0.622 0.608 0.632 0.682
[73] 0.618 0.602 0.628 0.652 0.622 0.652 0.658 0.654 0.658 0.604 0.604 0.664
[85] 0.622 0.652 0.642 0.660 0.648 0.670 0.628 0.582 0.642 0.622 0.646 0.644
[97] 0.584 0.632 0.626 0.604

mean(apply(res, 2, sum)/500)   # 平均確率
[1] 0.6345

  1:1000 での結果
n <- 1000
res<-matrix(0,500,n)
for(i in 1:500) {
 ii <- unique(sample(1:n, n, replace=T))
 res[i,ii] <- 1
}

mean(apply(res, 2, sum)/500)   # 平均確率
[1] 0.632914
```

(2) 方法２

sample 関数(replace=T)を使用して，1:n を復元ありで抽出した時の
選ばれた値の個数の比率が求める確率になる。

```
n<-100
res<- rep(0,500)
for(i in 1:500)
   res[i] <- length(unique(sample(1:n, n, replace = T)))/n
mean(res)
[1] 0.63554

n <- 1000
res<- rep(0,500)
for(i in 1:500)
  res[i] <- length(unique(sample(1:n, n, replace = T)))/n
mean(res)
[1] 0.632462
```

n が大きくなるほど，0.632 に近づきます。

以上