知的情報処理
7. 過学習: すべてを鵜呑みにしてはいけない

櫻井彰人
慶應義塾大学理工学部

本項の予定

- モデル選択
  - 仮説の評価
- 過学習という問題
  - 学習データの偏りとノイズ
  - 学習（訓練）誤差と予測（汎化）誤差
  - RとWekaで実感する
- 過学習対策
  - 決定木作成時の例

目次

- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

モデル選択

- データ分析の第一の目的は、データを生成した仕組みを推測すること
  - 第二の目的は、その結果を行動に役立てること
- 「仕組み」は「モデル」
  - 統計的には、「仮説」
    - 従って、モデル選択＝仮説選択
    - なお、選択範囲は、「仮説空間」

目次

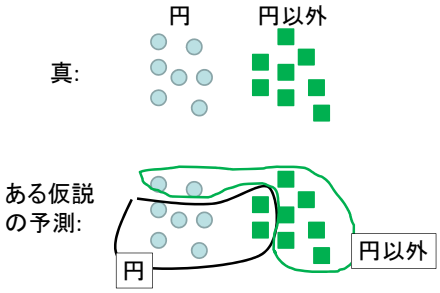
- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

モデルの評価

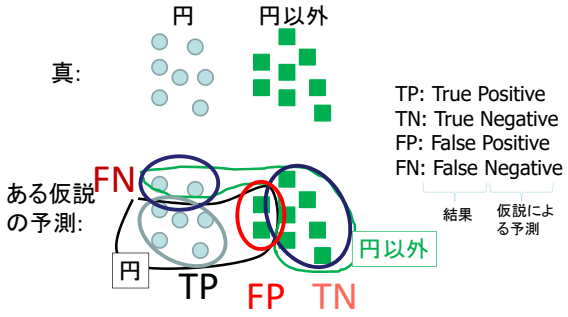
- 選択するには、評価する必要がある
- モデル（仮説）
  - 決定木学習においては、一つの決定木
    naïve Bayes学習なら、一つの「計算式」
    （k-近傍法では、学習データ＋計算方法）
- 評価方法の要件
  - 何らかの意味で「精度」や「信頼性」の高い仮説を用いたい。
  - 将来現れるデータに対しての値が欲しい

目立つモデルがあればよいが、世の中甘くない
http://www.selectioncriteria-examples.com/

Precision と Recall の前に



TP, TN, FP, FN



Confusion matrix

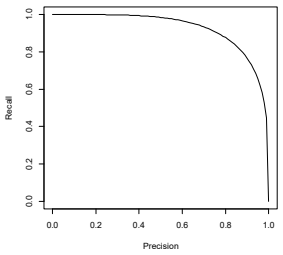
		真 P	真 N
仮説の予測	P	**TP** (True Positive)	**FP** (False Positive)
	N	**FN** (False Negative)	**TN** (True Negative)

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

両者のTradeoff と F-measure



$$F = \frac{1}{\frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right)}$$

Confusion matrix

		真値 P	真値 N
仮説の予測値	P	**TP** (True Positive)	**FP** (False Positive)
	N	**FN** (False Negative)	**TN** (True Negative)

1-α=specificity

$\alpha = \text{FPR} = \frac{FP}{FP+TN}$ 第一種の過誤

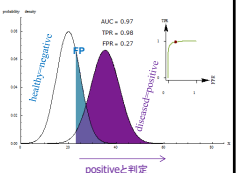
$\text{Precision} = \frac{TP}{TP+FP}$

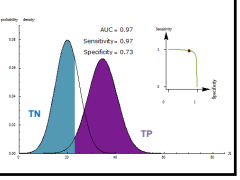
$\text{Recall} = \frac{TP}{TP+FN}$ 第二種の過誤

$\beta = \text{FNR} = \frac{FN}{FN+TP}$

1-β=sensitivity=recall=TPR=$\frac{TP}{FN+TP}$

$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$





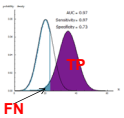
帰無仮説は、「陽性でない」
（陽性であることを示したいから）

第一種の過誤＝棄却した(陽性だと言った)が、それは誤り
第二種の過誤＝受理した(陰性だと言った)が、それは誤り

ROC curve

- Receiver operating characteristics
  - "ROC"という用語はレーダが開発された当初、操作盤上にあったノブの名
    - http://www.math-koubou.jp/stata/files/r12/est006.pdf

ROC curve

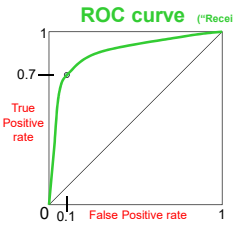


$$\frac{\text{\# true positives}}{\text{\# true positives + \# false negatives}}$$

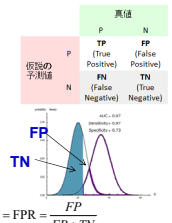
$$\text{Recall} = \frac{TP}{TP+FN} = \text{TPR}$$

$$\beta = \text{FNR} = \frac{FN}{FN+TP}$$

1-β=sensitivity



$$\frac{\text{\# false positives}}{\text{\# false positives + \# true negatives}}$$



$$\alpha = \text{FPR} = \frac{FP}{FP+TN}$$

1-α=specificity

ROC Curves

- 閾値(後述)を変えながら、サンプルの正誤を数える
- area under the curve (AUC) が大きい方がよい
- 異なった学習手法の性能を比較するのに適している

どれを使おうか

- 以降では、正解率(precision)を使おう
  - これは、分類問題のとき
  - 0/1問題(0か1かに分類)であれば、

$$\frac{1}{N}\sum_{i=1}^{N}|t(x_i) - f(x_i)|$$

- 回帰(近似)問題では、誤差の二乗和を。

$$\frac{1}{N}\sum_{i=1}^{N}(t(x_i) - f(x_i))^2$$

目次

- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

何が問題か

- 目的は、予測誤差(汎化誤差)の減少

$$E\left((t(x) - f(x))^2\right)$$

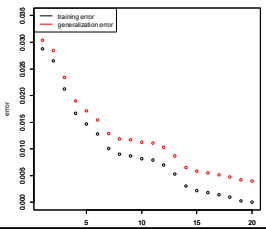
- ところが、これは測定できない
- 簡単に測れる数値は、学習誤差

$$\frac{1}{N}\sum_{i=1}^{N}(t(x_i) - f(x_i))^2$$

- もし、「学習誤差減少」=「予測誤差減少」であれば、問題ない
- しかし、そうはならない。これが問題

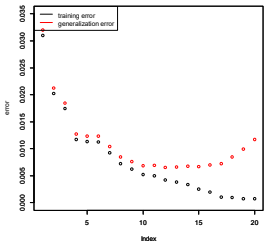
どういうことか

- いくつかの学習器を作って、学習誤差を測定し、それが減少する順に並べたとしよう
- 仮に図のようになったとしよう。すなわち、学習誤差が小さければ、予測誤差が小さいとしよう。
- 学習誤差が一番小さいものを選べばよい。



ところが

- ところがそうはいかないのである。
- 図のようなことがよくあるのである



では、どうすればよいか

- 予測誤差が測定できないのでは、どうしようもない。
- そこで、近似する方法を考えよう。
  - 一つは、validation set を用いる方法
  - 一つは、cross validationを用いる方法
  - 一つは、情報量基準を用いる方法

予測誤差とテスト誤差と訓練誤差

- （訓練データと同じ母集団から、同じ方法で抽出した）データに対する、仮説出力値の、真の出力値に対する誤差・誤り率の期待値が予測（汎化）誤差。

$$E\left(\left(t(x)-f(x)\right)^2\right)$$

  - 測定できないので、訓練データとは異なる（独立な）「テストデータ」を用いて近似する

$$\frac{1}{M}\sum_{x_l \in Test}\left(t(x_l)-f(x_l)\right)^2$$

- なお、訓練誤差は、訓練データ（学習データ）に対する、仮説出力値の、観測された出力値に対する誤差であった。式の形は、テスト誤差と同じだが、使うサンプルが違う

$$\frac{1}{N}\sum_{l=1}^{N}\left(t(x_l)-f(x_l)\right)^2$$

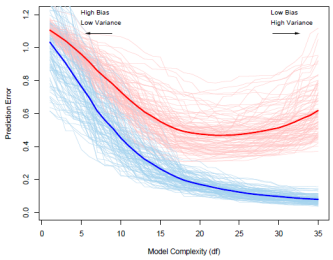


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Elements of Statistical Learning

テスト誤差最小でよいか？

- テスト誤差最小の学習器を選んでよいか？
  Yesである
- では、その分類器の予測誤差の推定として、そのテスト誤差を使ってよいのか？
  No である。
- なぜなら、その「テスト誤差」は学習に使ってしまっていたからである！！

どうするか？

- もう一組の（独立な）サンプルを用意して、それで誤差を測定すればよい。これこそが、「テスト誤差」である。
- （誤差最小の）モデル・学習器を選択するのに用いた誤差は、validation error と呼ばれる。
- 整理すると

Training set	training error	学習に用いるデータセット・誤差
Validation set	validation error	モデル選択に用いるデータセット・誤差
Test set	test error	性能表示に用いるデータセット・誤差

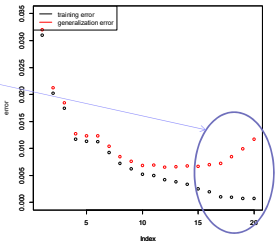
目次

- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

過学習

- over-learning とか over-training と呼ばれる
  - overfitting とも

このあたりのことを言う



http://www.staleytraining.com/articles/other/2010/avoid-overtraining.htm

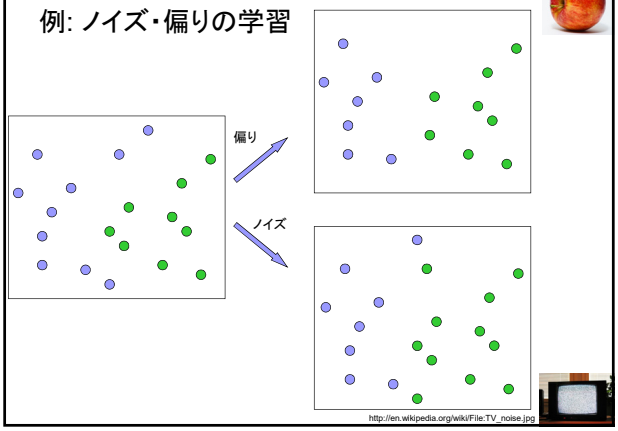
過学習 – なぜ起こるか

- 学習すべきでないものまで、学習してしまう
- 学習すべきでないもの
  - 学習データに含まれる偏り
    - 無限集合（真の概念が含む事例は無限個ある）の有限部分集合であるため、かならず、偏りがある。
  - 学習データに含まれる誤り
    - 現実データにはノイズがある。分類クラスにも属性値にもノイズは存在する。
- 学習してしまう
  - 学習能力が高いから
    - 調節可能なパラメータ数が多い

http://www.staleytraining.com/articles/other/2010/avoid-overtraining.htm

例: ノイズ・偏りの学習

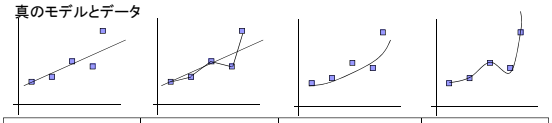
http://news.softpedia.com/news/Human-Perception-of-Gravity-Is-Biased-197339.shtml



http://en.wikipedia.org/wiki/File:TV_noise.jpg

ノイズ・偏りの学習: 関数近似

真のモデルとデータ

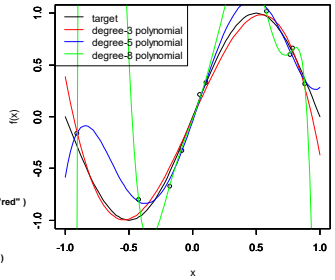


	1次式	2次多項式	全点を通る4次多項式
パラメータ数	2	3	5
残差（学習誤差）	大	中	0
予測誤差	小	中	大

本格的な(?) 関数近似のデモ:
http://www.mste.uiuc.edu/users/exner/java.f/leastsquares/

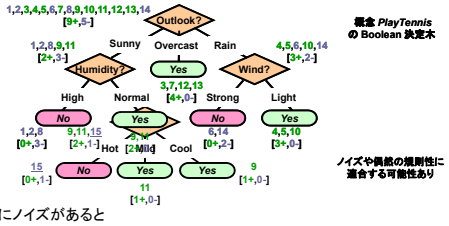
プログラム例

```
set.seed(123)
nData <- 10    # try 20 or 30
x <- 2 * (runif( nData ) - 0.5)
noiseSD <- 0.1;
y <- sin(pi*x) + rnorm(length(x), sd=noiseSD)
f <- function(x) sin(pi*x)

plot( f, xlim=c(-1,1), ylim=c(-1,1) )
points( x, y )

fit3a <- lm( y ~ poly(x, 3, raw=TRUE) )
fit3g <- function(x) predict( fit3a, data.frame( x=x ) )
par(new=T)
plot( fit3g, xlim=c(-1,1), ylim=c(-1,1), ylab="", xlab="", col="red" )

fit5a <- lm( y ~ poly(x, 5, raw=TRUE) )
par(new=T)
plot( function(u) predict( fit5a, data.frame( x=u ) ),
      xlim=c(-1,1), ylim=c(-1,1), ylab="", xlab="", col="blue" )

par(new=T)
plot( function(u) predict( lm( y ~ poly(x, 8, raw=TRUE) ), data.frame( x=u ) ),
      xlim=c(-1,1), ylim=c(-1,1), ylab="", xlab="", col="green" )

legend(par("usr")[1], par("usr")[4],
       c("target", "degree-3 polynomial", "degree-5 polynomial", "degree-8 polynomial"),
       lwd=1,
       col=c("black", "red", "blue", "green"),
       )
```



ノイズ・偏りの学習: 決定木の例

- 既出例: 帰納した木



- 訓練事例にノイズがあると
  - 事例 15: <*Sunny, Hot, Normal, Strong, ->
    - この例は実は noisy である. すなわち、正しいラベルは +
    - 以前に作成した木は、これを、誤分類する
  - 決定木はどのように更新されるべきか (incremental learning を考える)?
  - 新しい仮説 h' = T' の性能は h = T より*悪く* なると予想される（ノイズに騙されているから！）

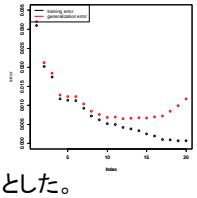
改めて：学習誤差と予測（汎化）誤差

- 学習（訓練）誤差
  - 学習器は、学習データを完全に表現すべく努力したはずだが、表現しきれずに残ってしまった誤差
    - 目標値（出力値）が離散値であれば、誤り数
    - 「完全に表現すべく」は、本当ではない。予測誤差（汎化）誤差を減らす努力を、学習アルゴリズムに組み込むことがある。
  - 学習終了時に求まる。データー個あたりの平均値
- 予測（汎化）誤差
  - 学習器が作った器械（予測器）で予測する時の誤差
  - データの分布が分れば、理論的に計算可能。しかし、実際に求めることはできない。データー個あたりの期待値

目次

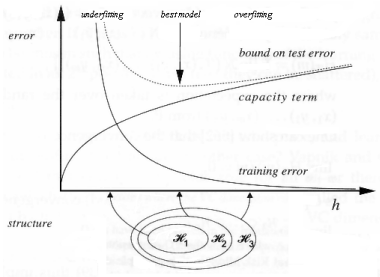
- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

横軸は何にするか？



変な問いに聞こえますが

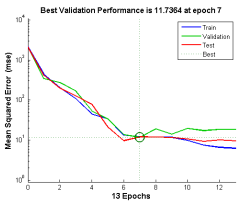
- 右図の場合、学習器をその学習誤差の降順に並べた。
- 多項式近似の時は、多項式の次数とした。
  - {一次多項式} ⊂ {二次多項式} ⊂ …であることに注意。
  - 機械学習では、仮説（モデル）の空間を、複雑さ（パラメータの数他）の順に並べたとき、… ⊂ {複雑さが低い仮説} ⊂ {複雑さが高い仮説} ⊂ … とする
  - このとき、仮説の複雑さを横軸にとれば、学習誤差はこの軸にそって減少することになる



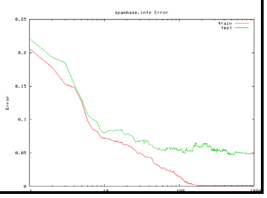
http://www.svms.org/srm/

もう一つの軸

- 学習に複数ステップを要する場合がある
- ニューラルネットワークのように、少しずつ学習を進めて行く場合



- Boostingのように、ステップごとに複雑度を上げていく（「複雑度」の軸と同じです）



http://www.mathworks.co.jp/products/neural-network/examples.html?file=/products/demos/shipping/nnet/fit_house_demo.html

http://jboost.sourceforge.net/doc.html

問題を言い換えると

- 問題なのは、
  - ある低複雑度の解（訓練誤差は大きい）と
  - ある高複雑度の解（訓練誤差は小さい）と
    が得られているとき、
  - 低複雑度の解の予測誤差が小さく
  - 高複雑度の解の予測誤差が大きく
    なること
- なお、複雑な解自体は問題ではない（問題かどうかは分らない）

http://thefuturebuzz.com/2008/09/29/simplicity-vs-complexity/

目次

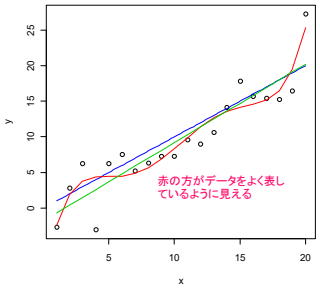
- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

非線形回帰時の過学習（Rによる）

```
library(nls2)
set.seed(1234)
x <- 1:20
y <- x+rnorm(20,sd=3)
plot(x,y)
xy <- data.frame(x=x,y=y)
res5 <- nls(y ~ a + b *x + c*x^2 + d *x^3 + e *x^4 + f*x^5, data=xy,
  start=list(a=1,b=1,c=0.5,d=0.1,e=0.05,f=0.001))
curve((x),col=4,add=T)              # 青
lines(x,predict(res5),col=2)        # 赤

res1 <- nls(y ~ a + b *x , data=xy, start=list(a=1,b=1))
lines(x,predict(res1),col=3)        # 緑
```

過学習が発生している

```
> # 学習誤差
> mean( (y-predict(res5) )^2)
[1] 5.808777
> mean( (y-predict(res1) )^2)
[1] 8.454419
> # 汎化誤差
> mean( (x-predict(res5) )^2)
[1] 3.544463
> mean( (x-predict(res1) )^2)
[1] 0.8988136
```



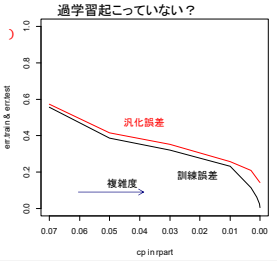
では、データ数を増やしたり、次数を上げたりしたらどうなるだろうか？

決定木での過学習（Rのrpart で）

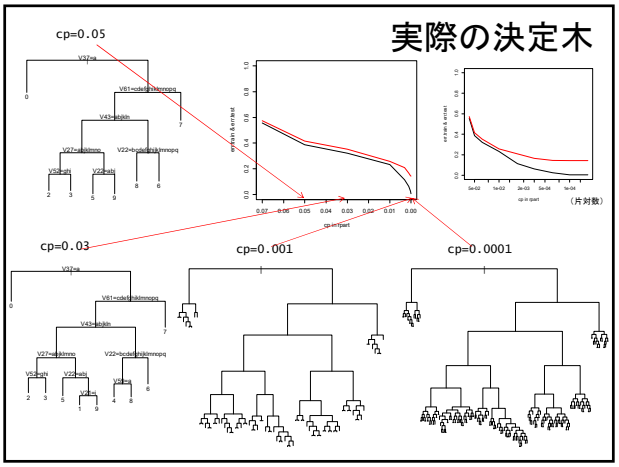
```
res <- c()
for (cp in c(0.07,0.05,0.03,0.01,0.003,0.001,0.0003,0.0001,0.00003)) {
  dig.tr <- rpart(V65~ ., dig,  control=rpart.control(minsplit=3, cp=cp))
  tbl <- table(dig[,65],predict(dig.tr, dig, type="class"))
  err.train <- 1 - sum(diag(tbl))/sum(tbl)

  tbl <- table(dig.test[,65],predict(dig.tr, dig.test, type="class"))
  err.test <- 1 - sum(diag(tbl))/sum(tbl)
  print(c(cp, err.train, err.test))
res <- rbind(res, c(cp, err.train, err.test) )
}
```

```
  [1] 0.0700000 0.5553230 0.5720646
  [1] 0.0500000 0.3876537 0.4162493
  [1] 0.0300000 0.3212137 0.3516973
  [1] 0.0100000 0.2312320 0.2576516
  [1] 0.0030000 0.1148313 0.2092376
  [1] 0.00100000 0.06251635 0.16583194
  [1] 0.00030000 0.02615747 0.14468559
  [1] 0.000100000 0.006016218 0.143016138
  [1] 0.000030000 0.005493068 0.142459655
```

過学習起こっていない？



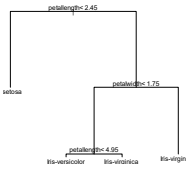
実際の決定木



補足：データの読込みと実験

あやめのデータ

```
library(rpart)
setwd("D:/R/Sample")
iris <- read.csv("07iris.csv", header=T)

(iris.tr <- rpart(class~ ., iris,
   control=rpart.control(minsplit=1)) )
plot(iris.tr); text(iris.tr)
```



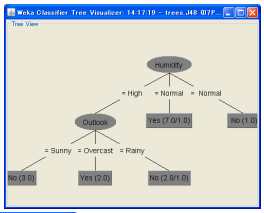
または

```
(iris.tr <- rpart(class~ ., iris,
   control=rpart.control(minsplit=1, cp=0.01)) )
plot(iris.tr); text(iris.tr)
```

http://www.hisuinosato.com/mt/archives/2008/06/post_656.html

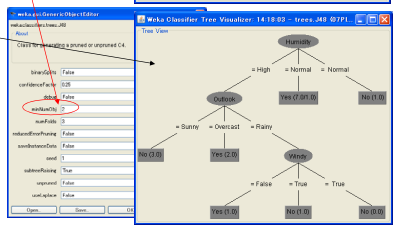
脱線: Weka では

07PlayTennis02.csv を読む
J48 で木を作成



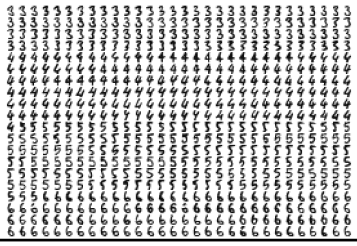
あれ？　分岐しない。
理由: 葉の最小データ(Obj)数が 2 以上
　　これを1以上にしてみよう

J48 で改めて木を作成



参考: 数字認識のデータの読込み方

```
library(rpart)
setwd("D:/R/Sample")
dig <- read.csv("05optdigits.tra.csv", header=F,
   colClasses=c(rep("integer",64),"factor")
)
dig.test <- read.csv("05optdigits.tes.csv", header=F,
   colClasses=c(rep("integer",64),"factor")
)
```



脱線: NBでは

```
library(e1071)
setwd("D:/R/Sample")
xy<-read.csv("05optdigits.tra.csv",
  header=F, colClasses="factor")
xyt<-read.csv("05optdigits.tes.csv",
  header=F, colClasses="factor",
  as.is=TRUE)
tt<-as.data.frame(factor(xyt[,1],
  levels=levels(xy[,1])))
for (i in 2:65) {
  tt<-data.frame(tt,factor(xyt[,i],
  levels=levels(xy[,i])))
}
names(tt)<-names(xy)

m <- naiveBayes(xy[,-65], xy[,65])

# accuracy for learning data
predictedClass <-predict(m, xy)
(cm <- table(xy[,65], predictedClass))
sum(diag(cm))/sum(cm)
```

```
> # accuracy for learning data
> predictedTrain <-predict(m, xy)
> (cm <- table(xy[,65], predictedTrain))
   predictedTrain
      0   1   2   3   4   5   6   7   8   9
  0 372   0   0   0   3   0   1   0   0   0
  1   0 368   8   0   0   0   0   1   1  11
  2   0   1 358   0   0   0   0   2  12   7
  3   0   1   1 372   0   1   0   5   4   5
  4   0   4   0   0 357   0   3  15   3   5
  5   0   1   1   2   2 342   1   0   1  26
  6   0   3   0   0   1   0 373   0   0   0
  7   0   2   0   0   2   0   0 380   1   2
  8   1  11   0   0   2   0   1   0 363   2
  9   0   4   1  11   9   2   0  11   3 341
> sum(diag(cm))/sum(cm)
[1] 0.9484698
>
> # accuracy for test data
> predictedTest <-predict(m, tt)
> (cmt <- table(tt[,65], predictedTest))
   predictedTest
      0   1   2   3   4   5   6   7   8   9
  0 172   0   0   0   4   1   1   0   0   0
  1   0 152  15   0   0   1   1   0   1  12
  2   0   7 154   2   0   1   0   1   7   5
  3   0   1   1 158   0   2   0   8   5   8
  4   0   2   0   0 171   0   0   4   3   1
  5   0   0   0   1   2 168   1   0   0  10
  6   0   4   0   0   2   0 175   0   0   0
  7   0   0   0   0   6   0   0 169   0   4
  8   0  13   1   1   1   3   0   2 142  11
  9   0   2   1   4   6   4   0   2   5 156
> sum(diag(cmt))/sum(cmt)
[1] 0.899833
```

目次

- モデル選択
- モデルの評価
  - Precision, recall, confusion matrix etc.
- 何が問題か
  - 学習誤差と予測誤差の乖離
- 過学習
  - 何となぜ
  - 横軸は何にするか
  - 実例
- 過学習対策

過学習対策

- 一つの方法
  - 予測(汎化)誤差の推定値が最も小さいところ(複雑度、学習回数)の学習器を使う
    - Validation set を用いる。Cross validation を行う
- 他の方法
  - 情報量基準に基づいて最適な複雑度を推定する。

では、どうすればよいか

- 予測誤差が測定できないのでは、どうしようもない。
- そこで、近似する方法を考えよう。
  - 一つは、validation setを用いる方法
  - 一つは、cross validationを用いる方法
  - 一つは、情報量基準を用いる方法

何が問題か？ (続)

- 思い出すと、
  訓練誤差の大きさと予測誤差の大きさの逆転現象が起こりうることが問題
- 補足
  - 通常は、「訓練誤差の大・小≒予測誤差の大・小」と考える(考えたい)
  - 複雑度が大きいときにこの逆転現象が発生する可能性がある

補足: 実用上の問題点

- 過学習が起こっているかどうかを推定するには、テストデータ(正確には、validation dataset)が必要
  - (はっきりとは言わなかったが、これまで)テストデータを用いて、予測誤差の推定をしてきた
- では、テストデータ(validation data)がないときは、どうしたらよいだろうか？

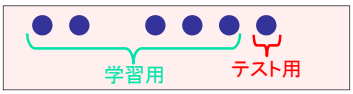
有力手法: cross validation

- 学習データを2つに分ける。一部を学習に、一部をテストに用いる
  - テストデータによる誤差を、予測誤差の推定値とする
- それを何回か繰り返し、「予測誤差の推定値」の平均値をとる
- 「繰り返す」時に、システマティックに行おう。
  - 学習データを、予め、k 等分し、その一個をテストに、残り k-1 個を学習に用いよう。それを k 回繰り返そう
  - 良い点：どのデータも一回だけテストデータになる。それを用いて、全体の正解率や、confusion matrix とすることができる

再掲

k 重クロスバリデーション
k-fold cross validation

訓練データを k 群に分け、$(k-1)$ 群で学習し、残りで予測誤差を計測する。これを全ての k 種類の組み合わせに対して行なう



万能ではないが、多くの場合に結構うまくいく

アルゴリズムや構造の適切さを測ることになる
構造や構造のパラメータ（複雑度）を決める目的で用いる

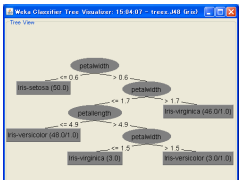
COM実験で行ったように、

Weka: デフォールトが 10-fold CV

あやめのデータで試してみよう

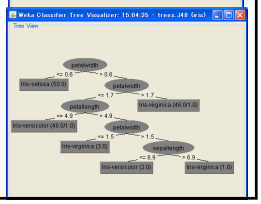
最小データ数（minNumObj）が2のとき:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```



最小データ数（minNumObj）が1のとき:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```



R で 10-fold cross validation

パッケージ bootstrap 中の crossval を用いる
使い方が少々面倒なので、プログラム全体を記す。
iris データに rpart を minsplit=30 で行った結果である。

```
setwd("D:/R/Sample")
iris <- read.csv("07iris.csv", header=T)

library(bootstrap)                # crossval will be used
theta.fit <- function (x,y) {
  tmp <- data.frame( sepallength=x[,1], sepalwidth=x[,2],
          petallength=x[,3], petalwidth=x[,4], class=y)
  return( rpart(class~., tmp, control=rpart.control(minsplit=30) ) )
}
theta.predict <- function( fit, x) {predict( fit, data.frame(x), type="class" ) }
results <- crossval(iris[,-5],iris[,5], theta.fit, theta.predict, ngroup=10)
(cm <- table( iris[,5], results$cv.fit ))
(accuracy <- sum(diag(cm))/sum(cm))
```

```
> (cm <- table( iris[,5], results$cv.fit ))

                 1  2  3
  Iris-setosa   50  0  0
  Iris-versicolor  0 47  3
  Iris-virginica   0  6 44
> (accuracy <- sum(diag(cm))/sum(cm))
[1] 0.94
```

分割はランダムに行われるので、実験ごとに結果は異なっても不思議ではない。

なお、全データで学習した結果の学習誤差は次のようにして求めることができる。

```
m <- rpart(class~., iris, control=rpart.control(minsplit=30) )
predicted <- predict( m, iris, type="class" )
correct <- iris[,5]
(cm <- table( correct, predicted ) )
(accuracyTraining <- sum(diag(cm))/sum(cm))
```

```
> m <- rpart(class~., iris, control=rpart.control(minsplit=30) )
> predicted <- predict( m, iris, type="class" )
> correct <- iris[,5]
> (cm <- table( correct, predicted ) )
             predicted
correct       Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa           50              0              0
  Iris-versicolor        0             49              1
  Iris-virginica         0              5             45
> (accuracyTraining <- sum(diag(cm))/sum(cm))
[1] 0.96
```

R で試す決定木のCV

下記の方法で CV ができる（ngroupがCV時の分割個数を表す）

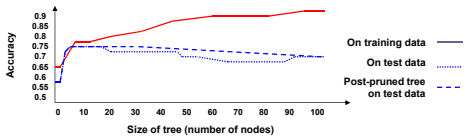
```
library(rpart)
setwd("D:/R/Sample")
xy <- read.csv("07PlayTennis02.csv", header=T)

library(bootstrap)                # crossval を使用する
theta.fit <- function (x,y) {
  tmp <- data.frame( Outlook=x[,1], Temperature=x[,2],
          Humidity=x[,3], Windy=x[,4], class=y)
  return( rpart(class~., tmp, control=rpart.control(minsplit=1) ) )
}
theta.predict <- function( fit, x) {predict( fit, data.frame(x), type="class" ) }
results <- crossval(xy[,-5], xy[,5], theta.fit, theta.predict, ngroup=7)
(cm <- table( xy[,5], results$cv.fit ))
(accuracy10CV <- sum(diag(cm))/sum(cm))
```

決定木の複雑さを制御するパラメータは minisplit であるので、これを変えて、CV を試みる。なお、本例では、7-fold CV とした

決定木の場合の少々異なる方法

Reduced-Error Pruning

- Reduced-Error Pruning によるテスト誤差の減少



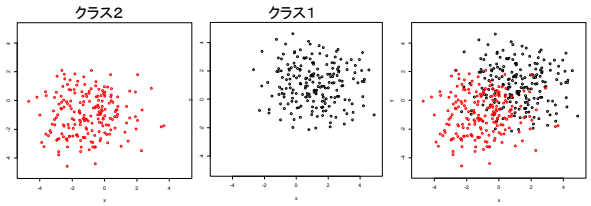
  - 節を刈ることによってテスト誤差が減少する
  - 注: $D_{validation}$ は D_{train} と D_{test} のどちらとも異なる
- 賛成論 と 批判論
  - 賛成: 最も正確な T' (T の部分木) のうちで最小のものが生成できる
  - 批判: T を作るのにわざわざデータ量を減らしている
    - $D_{validation}$ をとりおくだけの余裕があるか?
    - データ量が十分でなければ, 誤差をなおさら大きくする (D_{train} が不十分)

まとめ

- 過学習
  - 学習に使う構造(のパラメータ数)が大きすぎたり、構造が複雑すぎたりすると、学習データの偏りやノイズまで学習してしまう(ことがある)
  - データ数が与えられえいるなら、パラメータ数を変えて最適なものを選ぶ
  - その時、予測誤差(の推定値)が重要
  - 予測誤差の推定値は、cross validation で求める
- 学習ツール・アルゴリズムは、学習前・学習後に、様々な方法を用いて、過学習が起こりくい工夫はしている。

本日の課題

- 単純な例で、過学習が起こるかどうかを調べてみよう。
- 2個の異なる正規分布から生成されるサンプルに対し、生成元の正規分布が推定できるように学習する課題です。



データの生成

```
n.train <- 200      # 訓練データ数
n.test <- 200       # テストデータ数
sd <- 1.5           # 標準偏差は、共に、1.5

# クラス1は、中心(1,1), クラス2は、中心 (-1,-1) 、標準偏差 sd の正規分布
# 訓練データ
train1 <- data.frame(x=rnorm(n.train, mean=1,sd=sd),
                     y=rnorm(n.train, mean=1,sd=sd), c=rep(as.factor(1),n.train))
train2 <- data.frame(x=rnorm(n.train, mean=-1,sd=sd),
                     y=rnorm(n.train, mean=-1,sd=sd), c=rep(as.factor(2),n.train))
train <- rbind(train1, train2)

# テストデータ
test1 <- data.frame(x=rnorm(n.test, mean=1,sd=sd),
                    y=rnorm(n.test, mean=1,sd=sd), c=rep(as.factor(1),n.test))
test2 <- data.frame(x=rnorm(n.test, mean=-1,sd=sd),
                    y=rnorm(n.test, mean=-1,sd=sd), c=rep(as.factor(2),n.test))
test <- rbind(test1, test2)
```

訓練データのプロット

```
# クラス1の点のプロット
plot( subset(train, c==1)[,1:2], xlim=c(-5,5), ylim=c(-5,5))

# パネルを替えて、クラス2のプロット
dev.new()
plot( subset(train, c==2)[,1:2], col="red", xlim=c(-5,5), ylim=c(-5,5))

# パネルを替えて、クラス1とクラス2を重ねてプロット
dev.new()
plot( subset(train, c==1)[,1:2], xlim=c(-5,5), ylim=c(-5,5))
points( subset(train, c==2)[,1:2], col="red", xlab="", ylab="")
```

決定木(rpart)による学習

```
library(rpart)              # rpart を使うことにする
#
# スライド「決定木での過学習 (Rのrpart で) 」中のプログラムをここに持ってくる
# ただし、「被説明変数名 (クラス変数の名前) 」と「データの変数名」を変更のこと



# 訓練エラー、テストエラーのグラフのプロット
dev.new()
plot(res[,1:2],log="x", type="l",xlim=c(max(res[,1]),min(res[,1])),
  ylim=c(0,1.0), xlab="cp in rpart",ylab="err.train & err.test")
points(res[,c(1,3)],type="l",col="red",xlab="",ylab="")
```

決定木の複雑さを比較してみる

```
# cpの値を下記の2種類以外のものにして結果を比較して下さい

for (cp in c(0.05, 0.001)) {
  t <- rpart(c~ ., train, control=rpart.control(minsplit=3, cp=cp))
  dev.new()
  plot(t, main=paste("cp=",cp) )
#  text(t)       # ノード数が多い場合、重なりが多い
}
```