BOOSTING

(ENSEMBLE LEARNING II)

PRÁCTICA

Paquetes y funciones

Los paquetes

`ada, adabag,`

son los específicos de este tema.

Además, usaremos de temas anteriores los:

`rpart, e1071, class, MASS.`

Para conjuntos de datos, además de los anteriores, usaremos:

`ElemStatLearn,mlbench`

Documentación

Además de los help de las funciones, hay las vignettes de los paquetes, y los artículos:

- ada: Culp, M., Johnson, K., Michailidis, G. (2006). ada: an R Package for Stochastic Boosting, Journal of Statistical Software, 16.

Datos

Los mismos que en la práctica anterior, Ensemble Learning I.

AdaBoost demos

Ver el applet: An applet demonstrating adaboost, en la página de Yoav Freund, uno de los proponentes originales del método.

Otra demo, en MATLAB, se puede encontrar en Boosting demo.

AdaBoost en R

También hay varias implementaciones en R de versiones de Boosting. Emplearemos aquí la función `ada`, del paquete de igual nombre.

Cargamos los paquetes `ada` y `rpart`. Preparamos tres juegos de parámetros de control de `rpart`, para emplear como posibles *weak learners* en boosting:

En primer lugar, guardamos los que vienen por defecto en `default`, luego los parámetros para el árbol más simple, que tiene con tres nodos (árbol de profundidad 1), en `stump`, y, finalmente, para un árbol de profundidad 2 en `four`.

```
library(ada)
require(rpart)
default<-rpart.control()
stump<-rpart.control(cp=-1,maxdepth=1,minsplit=0)
four<-rpart.control(cp= -1,maxdepth=2,minsplit=0)
```

Seguid los ejemplos del help de la función `ada`, con los datos `iris`, para ver la sintaxis.

A continuación ved las ilustraciones de la sección 4 del artículo mencionado más arriba, Culp, M., Johnson, K., Michailidis, G. (2006). ada: an R Package for Stochastic Boosting, Journal of Statistical Software, 16. En el enlace hay también un fichero con el código R empleado en el artículo.

Se genera un conjunto de datos de prueba con dos clases `y=1, y=2` a separar a partir de dos variables predictoras, `x1` y `x2`:

```
n<-500
p<-10
f<-function(x, a, b, d) a * (x - b)^2 + d
```

```
set.seed(100)
x1<-runif(n/2, 0, 4)
y1<-f(x1, -1, 2, 1.7)+runif(n/2, -1, 1)
x2<-runif(n/2, 2, 6)
y2<-f(x2, 1, 4, -1.7)+runif(n/2, -1, 1)
y<-factor(c(rep(1, n/2), rep(2, n/2)))
mat<-matrix(rnorm(n * 8), ncol = 8)
dat<-data.frame(y = y, x1=c(x1,x2),x2=c(y1,y2),mat)
names(dat) <- c("y", paste("x", 1:10, sep = ""))
```

Representad el conjunto, constatando que las dos clases son difíciles de separar:

```
plot(dat$x1, dat$x2, pch = c(1:2)[y], col = c(2,4)[y],
              xlab=names(dat)[2], ylab=names(dat)[3])
```

La clase y=1 se representa con círculos rojos y la clase y=2 con triángulos azules. Separad un conjunto de entrenamiento:

```
indtrain <- sample(1:n, 100, FALSE)
train <- dat[indtrain,]
test <- dat[-indtrain,]
```

Ajustad un modelo adaBoost discreto, primero con un árbol `rpart` completo como weak learner, `control=default`, luego con `control=stump` y con `control=four`.

```
gdis <- ada(y~.,data=train, iter=50, loss="e",
type="discrete", control=default)
gdis
summary(gdis)
plot(gdis)
```

Añadid el conjunto de test al modelo:

```
gdis1 <- addtest(gdis, test[,-1], test[,1])
gdis1
```

Haced las predicciones para el conjunto de test y calculad la matriz de confusión:

```
yhat<-predict(gdis,newdata=test)
T<-table(True=test$y,Pred=yhat)
```

Siguiendo el código del artículo, tenéis más ejemplos de sintaxis de la función `ada`.

Tarea para entregar

Con los mismos datos `wine` para clasificación y `concrete` para regresión que usamos en la tarea anterior, más los datos `vowel`, que también hemos empleado en alguna ocasión, preparar métodos de predicción mediante:

1. Un árbol
2. Bagging
3. Random Forest
4. AdaBoost

Evaluar la calidad de las predicciones, ajustando parámetros del modelo cuando convenga, mediante los métodos de evaluación conocidos.