Adabag 2.0: una librería de R para Adaboost.M1 y Bagging

III Jornadas de Usuarios de R
17 y 18 de Noviembre de 2011
Escuela de Organización Industrial
Madrid

Esteban Alfaro Cortés
Matías Gámez Martínez
Noelia García Rubio
Univ. de Castilla-La Mancha
F. C.C. Económicas y Empresariales
Albacete

Índice

Facultad de Ciencias Económicas y Empresariales de Albacete

1. Introducción

- En las últimas décadas se han desarrollado muchos métodos de clasificación combinados basándose en árboles de clasificación. En la librería que presentamos se implementan dos de los más populares, Adaboost.M1 (versión multiclase de Adaboost) y Bagging.
- La principal diferencia es que mientras Adaboost [Freund y Shapire, 1996] construye sus clasificadores base secuencialmente, actualizando la distribución sobre el conjunto de entrenamiento para crear cada uno de ellos, Bagging [Breiman, 1996] combina los clasificadores individuales construidos sobre réplicas boostrap del conjunto de entrenamiento.

Facultad de Ciencias Económicas y Empresariales de Albacete

1. Introducción

- Adabag 2.1 es una actualización de Adabag que añade una nueva función "margins" para calcular los márgenes de los clasificadores. La primera versión fue publicada en junio de 2006 y se ha utilizado ampliamente en tareas de clasificación en campos científicos muy diversos:
  - Kriegler, B. & Berk, R. (2007). Estimación de las personas sin hogar en Los Ángeles (estimación espacial en pequeñas áreas).
  - Stewart, B. & Zhukov, Y. (2009). Análisis del contenido para clasificación de documentos militares.
  - De Bock, K.W. , Coussement, K. & Van den Poel, D. (2010). Clasificación conjunta basada en Modelos Aditivos Generalizados.
  - De Bock, K.W. , & Van den Poel, D. (2011). Predicción de la fuga de clientes.
  - Y en paquetes de R como "digeR", GUI para analizar datos DIGE (Difference In-Gel Electrophoresis) bidimensionales, realizado por: Fan, Y., Murphy, T.B. & Watson, W.G. (2009): digeR Manual.

2. Algoritmos: AdaBoost

AdaBoost (Adaptative Boosting), Freund y Schapire (1996)

- **Utiliza reponderación en lugar de remuestreo**
  - **El clasificador débil anterior tiene sólo un 50% de precisión sobre el conjunto de entrenamiento con los nuevos pesos**
- **Es especialmente adecuado para clasificadores débiles**
- **La clasificación final se basa en el voto ponderado de los clasificadores individuales**

Facultad de Ciencias Económicas y Empresariales de Albacete

AdaBoost.M1

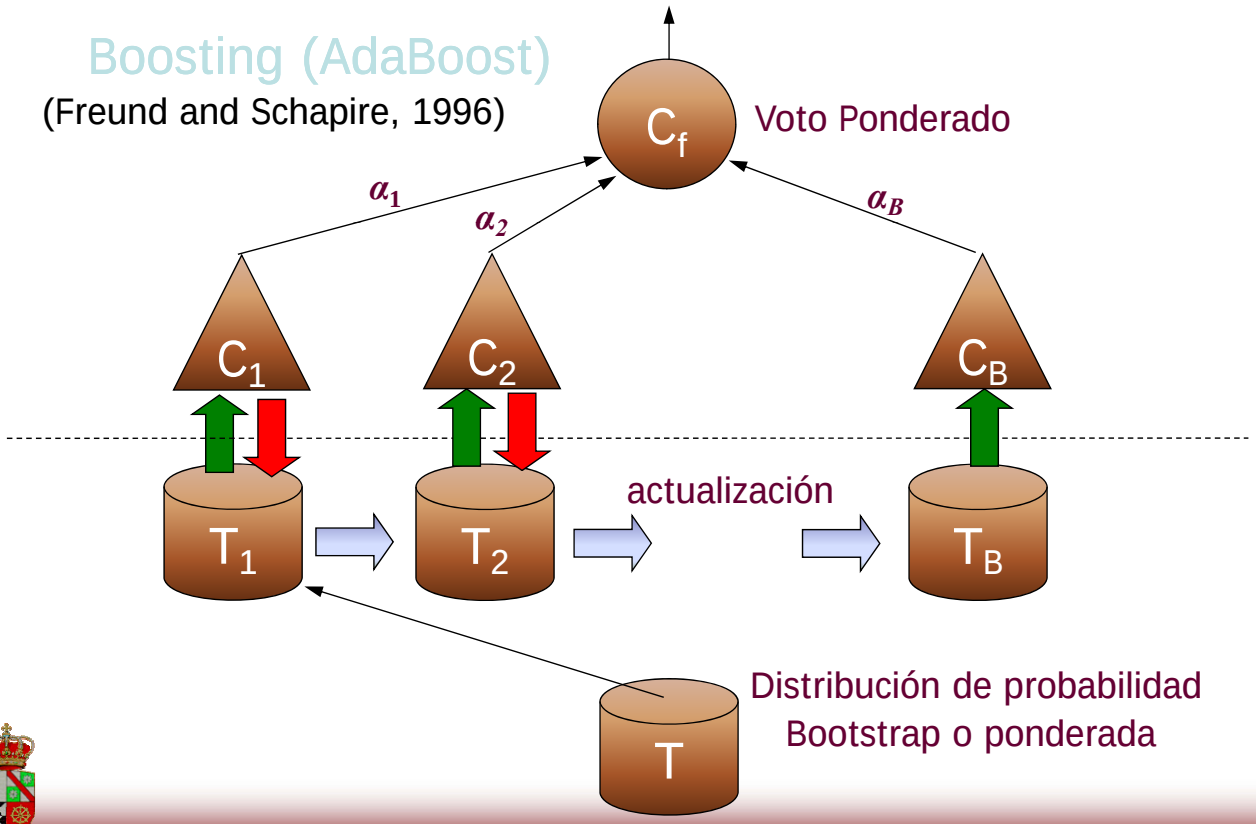
AdaBoost.M1 se aplica de la siguiente manera:

1. **Hacer una versión ponderada del conjunto de entrenamiento (T_b), inicialmente todos los pesos son $1/n$**
2. **Repita el procedimiento para b=1, 2, ...,B**
   - **a) Construir sobre T_b un clasif. individual $C_b(x_i)$ ={1,…, k}**
   - **b) Calcular:** $$e_b = \sum_{1}^{n} w_i^b \, \mathrm{I}(\mathrm{C_b}(\mathbf{x}_i) \neq y_i) \quad \text{y} \quad \alpha_b = 1/2 \ln\left(\frac{1-e_b}{e_b}\right)$$
   - **c) Actualizar y normalizar los pesos:** $$w_i^{b+1} = w_i^b \exp(\alpha_b \, \mathrm{I}(\mathrm{C_b}(\mathbf{x}_i) \neq y_i))$$
3. **Construir el clasificador final (suponiendo Y={1,…, k}):**

$$\mathrm{C_f}(\mathbf{x}_i) = \arg\max_{j \in Y} \sum_{b=1}^{B} \alpha_b \, \mathrm{I}(\mathrm{C_b}(\mathbf{x}_i) = j)$$

Boosting (AdaBoost)

(Freund and Schapire, 1996)

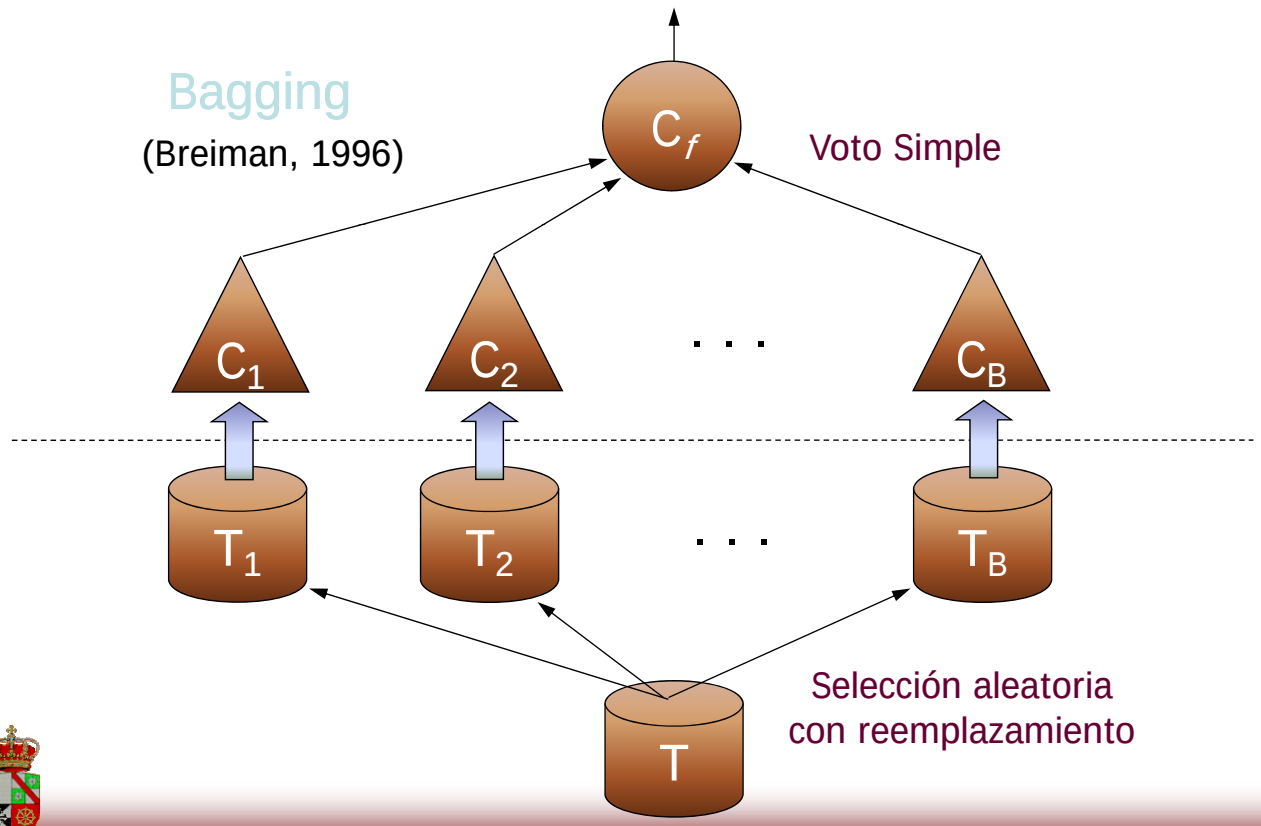


Facultad de Ciencias Económicas y Empresariales de Albacete

2. Algoritmos: Bagging

Bootstrapping and aggregating (Breiman, 1996)

- **Para b = 1, 2, .., B**
  - **Realizar con reemplazamiento n^*=n muestras de T**
  - **Entrenar el clasificador C_b**
- **El clasificador final se obtiene por mayoría simple de C_1 .. C_B**
- **Aumenta la estabilidad del clasificador/disminuye su varianza**



Facultad de Ciencias Económicas y Empresariales de Albacete

3. Funciones: Boosting

`adaboost.M1` Applies the Adaboost.M1 algorithm to a data set

Description

Fits the Adaboost.M1 algorithm proposed by Freund and Schapire in 1996 using classification trees as single classifiers.

Usage

```
adaboost.M1(formula, data, boos = TRUE, mfinal = 100,
coeflearn = 'Breiman', control)
```

Arguments

`formula` a formula, as in the `lm` function.

`data` a data frame in which to interpret the variables named in `formula`.

`boos` if TRUE (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If FALSE, every observation is used with its weights.

`mfinal` an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.

`coeflearn` if `'Breiman'`(by default), `alpha=1/2ln((1-err)/err)` is used. If `'Freund' alpha=ln((1-err)/err)` is used. Where `alpha` is the weight updating coefficient.

`control` options that control details of the rpart algorithm. See `rpart.control` for more details.

3. Funciones: Boosting

`adaboost.M1` Applies the Adaboost.M1 algorithm to a data set

Details

Adaboost.M1 is a simple generalization of Adaboost for more than two classes.

Value

An object of class `adaboost.M1`, which is a list with the following components:

`formula` the formula used.
`trees` the trees grown along the iterations.
`weights` a vector with the weighting of the trees of all iterations.
`votes` a matrix describing, for each observation, the number of trees that assigned it to each class, weighting each tree by its `alpha` coefficient.
`class` the class predicted by the ensemble classifier.
`importance` returns the relative importance of each variable in the classification task. This measure is the number of times each variable is selected to split.

3. Funciones: Boosting

Facultad de Ciencias Económicas y Empresariales de Albacete

`predict.boosting` Predicts from a fitted adaboost.M1 object

Description

Classifies a data frame using a fitted adaboost.M1 object.

Usage

```
## S3 method for class 'boosting'
predict(object, newdata, ...)
```

Arguments

`object` fitted model object of class `adaboost.M1`. This is assumed to be the result of some function that produces an object with the same named components as that returned by the `adaboost.M1` function.
`newdata` data frame containing the values at which predictions are required. The predictors referred to in the right side of `formula(object)` must be present by name in newdata.
`...` further arguments passed to or from other methods.

3. Funciones: Boosting

predict.boosting Predicts from a fitted adaboost.M1 object

Value

An object of class `predict.boosting`, which is a list with the following components:

`formula` the formula used.
`votes` a matrix describing, for each observation, the number of trees that assigned it to each class, weighting each tree by its `alpha` coefficient.
`class` the class predicted by the ensemble classifier.
`confusion` the confusion matrix which compares the real class with the predicted one.
`error` returns the average error.

Facultad de Ciencias Económicas y Empresariales de Albacete

3. Funciones: Boosting

boosting.cv Runs v-fold cross validation with adaboost.M1

Description

The data are divided into v non-overlapping subsets of roughly equal size. Then, `adaboost.M1` is applied on `(v-1)` of the subsets. Finally, predictions are made for the left out subsets, and the process is repeated for each of the `v` subsets.

Usage

```
boosting.cv(formula, data, v = 10, boos = TRUE, mfinal = 100,
coeflearn = "Breiman", control)
```

Arguments

`formula` a formula, as in the `lm` function.
`data` a data frame in which to interpret the variables named in `formula`.
`boos` if `TRUE` (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If `FALSE`, every observation is used with its weights.
`v` an integer, specifying the type of v-fold cross validation. Defaults to 10. If `v` is set as the number of observations, leave-one-out cross validation is carried out. Besides this, every value between two and the number of observations is valid and means that roughly every v-th observation is left out.
`mfinal` an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.
`coeflearn` if `'Breiman'`(by default), `alpha=1/2ln((1-err)/err)` is used. If `'Freund'` `alpha=ln((1-err)/err)` is used. Where `alpha` is the weight updating coefficient.
`control` options that control details of the rpart algorithm. See `rpart.control` for more details.

3. Funciones: Boosting

`boosting.cv` Runs v-fold cross validation with adaboost.M1

Value

An object of class `boosting.cv`, which is a list with the following components:

`class` the class predicted by the ensemble classifier.

`confusion` the confusion matrix which compares the real class with the predicted one.

`error` returns the average error.

3. Funciones: Bagging

`bagging` Applies the Bagging algorithm to a data set

Description

Fits the Bagging algorithm proposed by Breiman in 1996 using classification trees as single classifiers.

Usage

```
bagging(formula, data, mfinal = 100, control)
```

Arguments

`formula` a formula, as in the `lm` function.

`data` a data frame in which to interpret the variables named in `formula`.

`mfinal` an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.

`control` options that control details of the rpart algorithm. See `rpart.control` for more details.

3. Funciones: Bagging

`bagging` Applies the Bagging algorithm to a data set

Details

Unlike boosting, individual classifiers are independent among them in bagging.

Value

An object of class `bagging`, which is a list with the following components:

`formula` the formula used.
`trees` the trees grown along the iterations.
`votes` a matrix describing, for each observation, the number of trees that assigned it to each class.
`class` the class predicted by the ensemble classifier.
`samples` the bootstrap samples used along the iterations.
`importance` returns the relative importance of each variable in the classification task. This measure is the number of times each variable is selected to split.

Facultad de Ciencias Económicas y Empresariales de Albacete

3. Funciones: Bagging

`predict.bagging` Predicts from a fitted bagging object

Description

Classifies a dataframe using a fitted bagging object.

Usage

```
## S3 method for class 'bagging'
predict(object, newdata, ...)
```

Arguments

`object` fitted model object of class `bagging`. This is assumed to be the result of some function that produces an object with the same named components as that returned by the `bagging` function.
`newdata` data frame containing the values at which predictions are required. The predictors referred to in the right side of `formula(object)` must be present by name in newdata.
`...` further arguments passed to or from other methods.

3. Funciones: Bagging

predict.bagging Predicts from a fitted bagging object

Value

An object of class `predict.bagging`, which is a list with the following components:

`formula` the formula used.
`votes` a matrix describing, for each observation, the number of trees that assigned it to each class.
`class` the class predicted by the ensemble classifier.
`confusion` the confusion matrix which compares the real class with the predicted one.
`error` returns the average error.

Facultad de Ciencias Económicas y Empresariales de Albacete

3. Funciones: Bagging

bagging.cv Runs v-fold cross validation with Bagging

Description

The data are divided into v non-overlapping subsets of roughly equal size. Then, `bagging` is applied on `(v-1)` of the subsets. Finally, predictions are made for the left out subsets, and the process is repeated for each of the `v` subsets.

Usage

```
bagging.cv(formula, data, v = 10, mfinal = 100, control)
```

Arguments

`formula` a formula, as in the `lm` function.
`data` a data frame in which to interpret the variables named in `formula`.
`v` an integer, specifying the type of v-fold cross validation. Defaults to 10. If `v` is set as the number of observations, leave-one-out cross validation is carried out. Besides this, every value between two and the number of observations is valid and means that roughly every v-th observation is left out.
`mfinal` an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.
`control` options that control details of the rpart algorithm. See `rpart.control` for more details.

3. Funciones: Bagging

`bagging.cv` Runs v-fold cross validation with Bagging

Value

An object of class `bagging.cv`, which is a list with the following components:

`class` the class predicted by the ensemble classifier.

`confusion` the confusion matrix which compares the real class with the predicted one.

`error` returns the average error.

3. Funciones: Márgenes

`margins` Calculates the margins

Description

Calculates the margins of an adaboost.M1 or bagging classifier for a data frame.

Usage

```
margins(object, newdata, ...)
```

Arguments

`object` This object must be the output of one of the functions `bagging`, `adaboost.M1`, `predict.bagging` or `predict.boosting`. This is assumed to be the result of some function that produces an object with two components named `formula` and `class`, as those returned for instance by the `bagging` function.

`newdata` The same data frame used for building the `object`.

3. Funciones: Márgenes

`margins` Calculates the margins

Details

Intuitively, the margin for an observation is related to the certainty of its classification. It is calculated as the difference between the support of the correct class and the maximum support of a wrong class.

Value

An object of class `margins`, which is a list with only a component:

`margins` A vector with the margins.

4. Ejemplo: Vehicle (4 clases)


```
library(adabag)
data(Vehicle)
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
Vehicle.rpart <- rpart(Class~.,data=Vehicle[sub,],maxdepth=5)
Vehicle.rpart.pred <- predict(Vehicle.rpart,newdata=Vehicle[-sub,], type="class")
tb <- table(Vehicle.rpart.pred,Vehicle$Class[-sub])
error.rpart <- 1-(sum(diag(tb))/sum(tb))
tb
Vehicle.rpart.pred bus opel saab van
              bus   64    7    9   0
              opel   3   26   12   0
              saab   0   37   46   9
              van    2    2    5  60
> error.rpart
[1] 0.3049645
```

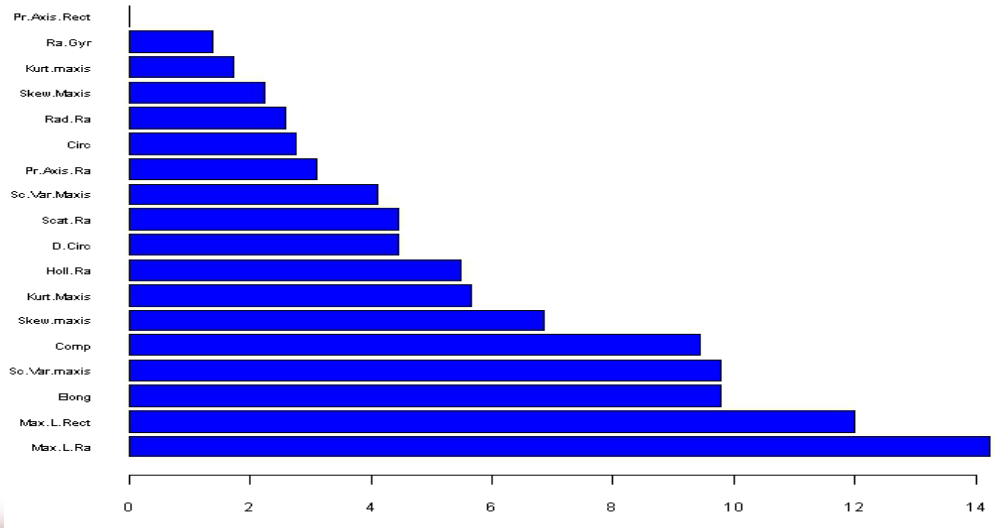
4. Ejemplo: Vehicle (4 clases)


```
> Vehicle.bagging <- bagging(Class ~.,data=Vehicle[sub, ],mfinal=50,
control=rpart.control(maxdepth=3))
> Vehicle.bagging.pred <- predict.bagging(Vehicle.bagging,newdata=Vehicle[-
sub, ])
> Vehicle.bagging.pred$confusion
 Observed Class
Predicted Class bus opel saab van
           bus   67   10    9   1
           opel   1   32   11   0
           saab   1   25   40   0
           van    0    5   12  68
> Vehicle.bagging.pred$error
[1] 0.2659574
```

4. Ejemplo: Vehicle (4 clases)

Facultad de Ciencias Económicas y Empresariales de Albacete

```
> Vehicle.bagging$importance
      Comp       Circ     D.Circ     Rad.Ra  Pr.Axis.Ra  Max.L.Ra   Scat.Ra      Elong
  9.433962   2.744425   4.459691   2.572899   3.087479  14.236707   4.459691   9.777015
Pr.Axis.Rect  Max.L.Rect Sc.Var.Maxis Sc.Var.maxis     Ra.Gyr  Skew.Maxis  Skew.maxis  Kurt.maxis
  0.000000  12.006861   4.116638   9.777015   1.372213   2.229846   6.861063   1.715266
 Kurt.Maxis    Holl.Ra
  5.660377   5.488851
> barplot(sort(Vehicle.bagging$importance,dec=T),col="blue",horiz=T,las=1)
```

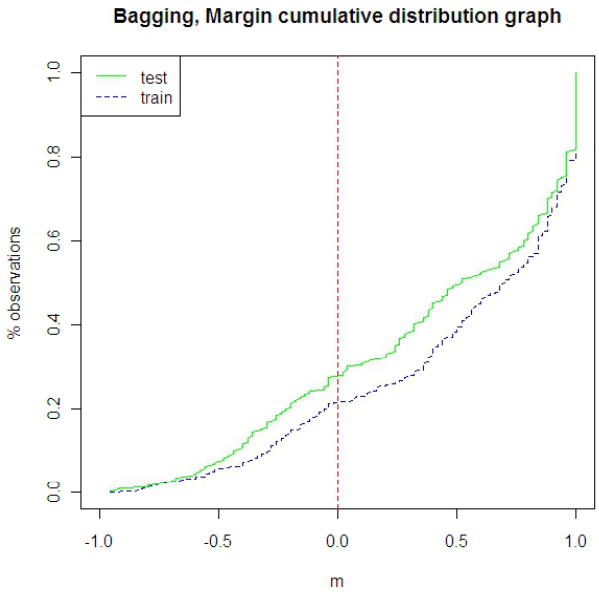


4. Ejemplo: Vehicle (4 clases)


```
> Vehicle.bagging.cv <- bagging.cv(Class~.,data=Vehicle)
> names(Vehicle.bagging.cv)
[1] "class"     "confusion" "error"
> Vehicle.bagging.cv$confusion
 Clase real
Clase estimada bus opel saab van
          bus  206   20   28   1
          opel   3  106   72   0
          saab   0   70   91   1
          van    9   16   26 197
> Vehicle.bagging.cv$error
[1] 0.2907801
```

Facultad de Ciencias Económicas y Empresariales de Albacete

```
> margins(Vehicle.bagging,Vehicle[sub,])->Vehicle.bagging.margins # training set
> margins(Vehicle.bagging.pred,Vehicle[-sub,])->Vehicle.bagging.predmargins # test set
> margins.test<-Vehicle.bagging.predmargins[[1]]
> margins.train<-Vehicle.bagging.margins[[1]]
> plot(sort(margins.train), (1:length(margins.train))/length(margins.train),type="l", xlim=c(-1,1),
+ main="Bagging, Margin cumulative distribution graph",xlab="m",ylab="% observations", col="blue3",lty=2)
> abline(v=0, col="red",lty=2)
> lines(sort(margins.test), (1:length(margins.test))/length(margins.test),type="l", cex=.5,col="green")
```

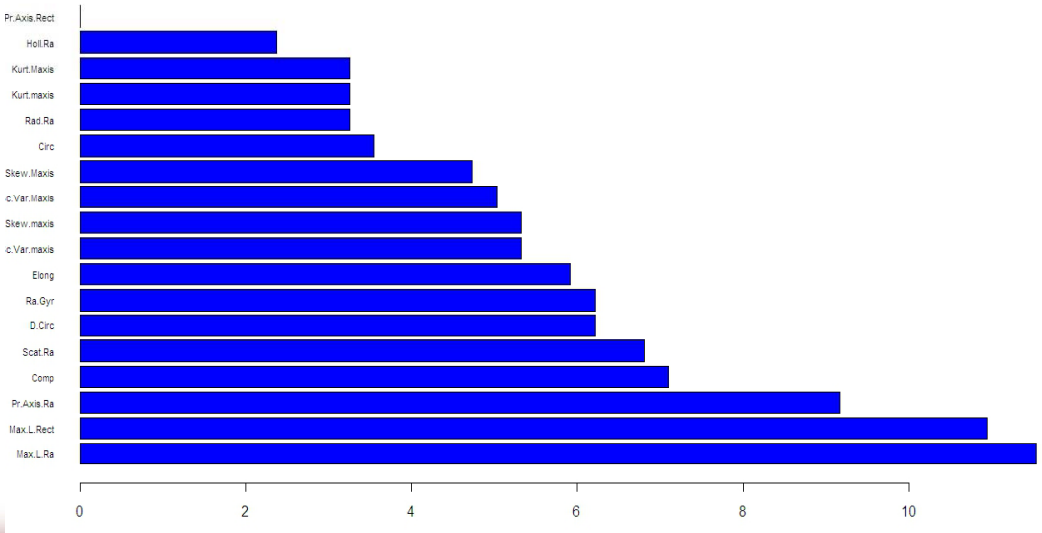


4. Ejemplo: Vehicle (4 clases)


```
> Vehicle.adaboost <- adaboost.M1(Class ~.,data=Vehicle[sub,],mfinal=50,
+ control=rpart.control(maxdepth=3))
> Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost,newdata=Vehicle[-
sub, ])
> Vehicle.adaboost.pred$confusion
 Observed Class
Predicted Class bus opel saab van
           bus   67    2    3   0
           opel   2   41   20   1
           saab   0   27   45   4
           van    0    2    4  64
> error.adaboost <- 1-sum(diag(Vehicle.adaboost.pred$confusion))/sum(
Vehicle.adaboost.pred$confusion)
> error.adaboost
[[1] 0.2304965
```

Facultad de Ciencias Económicas y Empresariales de Albacete

```
> Vehicle.adaboost$importance
       Comp         Circ       D.Circ       Rad.Ra    Pr.Axis.Ra     Max.L.Ra      Scat.Ra        Elong
   7.100592     3.550296     6.213018     3.254438     9.171598    11.538462     6.804734     5.917160
Pr.Axis.Rect    Max.L.Rect Sc.Var.Maxis Sc.Var.maxis       Ra.Gyr    Skew.Maxis    Skew.maxis    Kurt.maxis
   0.000000    10.946746     5.029586     5.325444     6.213018     4.733728     5.325444     3.254438
 Kurt.Maxis      Holl.Ra
   3.254438     2.366864
> barplot(sort(Vehicle.adaboost$importance,dec=T),col="blue",horiz=T,las=1)
```

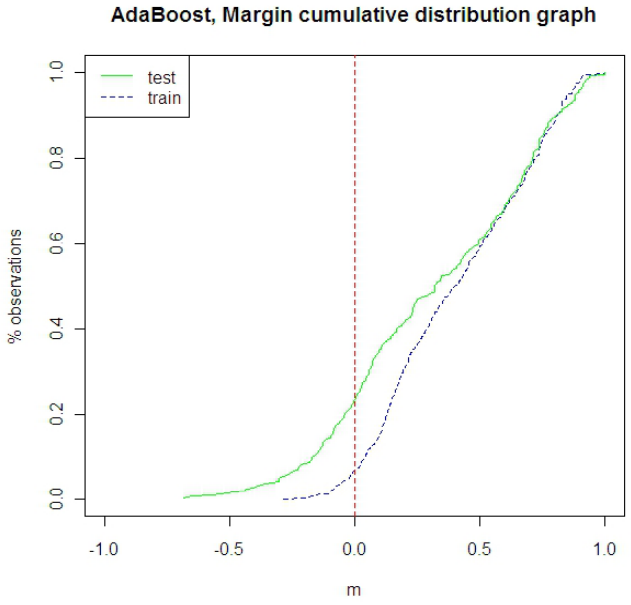


4. Ejemplo: Vehicle (4 clases)


```
> Vehicle.adaboost.cv <- boosting.cv(Class~.,data=Vehicle)
> names(Vehicle.adaboost.cv)
[1] "class"     "confusion" "error"
> Vehicle.adaboost.cv$confusion
                Observed Class
Predicted Class bus opel saab van
           bus  213    2    3   1
           opel   3  116   74   2
           saab   1   88  134   4
           van    1    6    6 192
> Vehicle.adaboost.cv$error
[1] 0.2257683
```

Facultad de Ciencias Económicas y Empresariales de Albacete

```
> margins(Vehicle.adaboost,Vehicle[sub,])->Vehicle.adaboost.margins # training set
> margins(Vehicle.adaboost.pred,Vehicle[-sub,])->Vehicle.adaboost.predmargins # test set
> margins.test<-Vehicle.adaboost.predmargins[[1]]
> margins.train<-Vehicle.adaboost.margins[[1]]
> plot(sort(margins.train), (1:length(margins.train))/length(margins.train),type="l", xlim=c(-1,1),
+main="AdaBoost, Margin cumulative distribution graph",xlab="m",ylab="% observations", col="blue3",lty=2)
> abline(v=0, col="red",lty=2)
> lines(sort(margins.test), (1:length(margins.test))/length(margins.test),type="l", cex=.5,col="green")
```



5. Conclusiones

Facultad de Ciencias Económicas y Empresariales de Albacete

Las funciones incluidas en esta librería permiten:

1. **Resolver problemas de clasificación con más de dos clases.**
2. **Entrenar y predecir para nuevas observaciones.**
3. **Medir la importacia relativa de las variables.**
4. **Calcular los márgenes de los clasificadores combinados.**
5. **Estimar el error mediante validación cruzada.**

¡Gracias por su atención!

e-mail: Esteban.Alfaro@uclm.es
Matias.Gamez@uclm.es
Noelia.Garcia@uclm.es