ipred: Improved Predictors

by Andrea Peters, Torsten Hothorn and Berthold Lausen

Introduction

In classification problems, there are several attempts to create rules which assign future observations to certain classes. Common methods are for example linear discriminant analysis or classification trees. Recent developments lead to substantial reduction of misclassification error in many applications. Bootstrap aggregation ("bagging", Breiman, 1996a) combines classifiers trained on bootstrap samples of the original data. Another approach is indirect classification, which incorporates a priori knowledge into a classification rule (Hand et al., 2001). Since the misclassification error is a criterion to assess the classification techniques, its estimation is of main importance. A nearly unbiased but highly variable estimator can be calculated by cross validation. Efron and Tibshirani (1997) discuss bootstrap estimates of misclassification error. As a by-product of bagging, Breiman (1996b) proposes the out-of-bag estimator.

However, the calculation of the desired classification models and their misclassification errors is often aggravated by different and specialized interfaces of the various procedures. We propose the **ipred** package as a first attempt to create a unified interface for improved predictors and various error rate estimators. In the following we demonstrate the functionality of the package in the example of glaucoma classification. We start with an overview about the disease and data and review the implemented classification and estimation methods in context with their application to glaucoma diagnosis.

Glaucoma

Glaucoma is a slowly processing and irreversible disease that affects the optic nerve head. It is the second most reason for blindness worldwide. Glaucoma is usually diagnosed based on a reduced visual field, assessed by a medical examination of perimetry and a smaller number of intact nerve fibers at the optic nerve head.

One opportunity to examine the amount of intact nerve fibers is using the Heidelberg Retina Tomograph (HRT), a confocal laser scanning tomograph, which does a three dimensional topographical analysis of the optic nerve head morphology. It produces a series of 32 images, each of 256×256 pixels, which are converted to a single topographic image, see Figure 1. A less complex, but although a less informative examination tool is the 2-dimensional fundus photography.

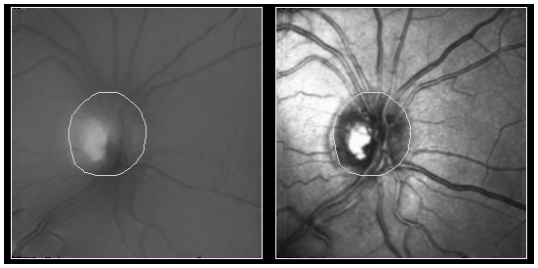

Figure 1: Topographic image of the optic nerve head, obtained by an examination with the Heidelberg Retina Tomograph.

However, in cooperation with clinicians and a priori analysis we derived a diagnosis of glaucoma based on three variables only: w_{lora} represents the loss of nerve fibers and is obtained by a 2-dimensional fundus photography, w_{cs} and w_{clv} describe the visual field defect (Peters et al., 2002).

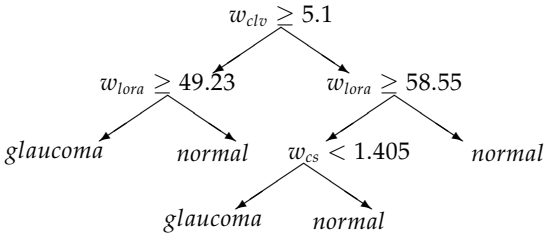

Figure 2: Glaucoma diagnosis.

Figure 2 represents the diagnosis of glaucoma in terms of a medical decision tree. A complication of the disease is that a damage in the optic nerve head morphology precedes a measurable visual field defect. Furthermore, an early detection is of main importance, since an adequate therapy can only slow down the progression of the disease. Hence, a classification rule for detecting early damages should include morphological informations, rather than visual field data only. Therfore, we construct classification rules based on 64 HRT variables and 7 anamnestic variables to predict the class membership of future observations. We use data accumulated at the Erlanger Eye Hospital (Hothorn et al., 2002) and match 170 observations of 85 normal and 85 glaucoma eyes by age and sex to prevent for possible confounding.

Bootstrap aggregation

Referring to the example of glaucoma diagnosis we first demonstrate the functionality of `bagging` and `predict.bagging`. We fit a bootstrap aggregated

classification tree with nbagg = 50 bootstrap replications by

```
R> fit <- bagging(diagnosis ~ ., nbagg = 50,
+   data = study.group, coob=TRUE)
```

where study.group contains explanatory HRT and anamnestic variables and the response of glaucoma diagnosis, a factor at two levels normal and glaucoma. print.bagging returns informations about the bagging object, i.e., the number of bootstrap replications used and, as requested by coob=TRUE, the out-of-bag estimate of misclassification error (Breiman, 1996b).

```
R> fit
Bagging classification trees
with 50 bootstrap replications

Out-of-bag misclassification error: 0.2242
```

The out-of-bag estimate uses the observations which are left out in a bootstrap sample to estimate the misclassification error at almost no additional computational costs. Hothorn and Lausen (2002) propose to use the out-of-bag samples for a combination of linear discriminant analysis and classification trees, called "Double-Bagging", which is available by choosing method="double".

predict.bagging predicts future observations according to the fitted model.

```
R> predict(fit,
+    newdata=study.group[c(1:3, 86:88), ])
[1] normal   normal   normal
                glaucoma glaucoma glaucoma
Levels:  glaucoma normal
```

Both bagging and predict.bagging rely on the rpart routines. The rpart routine for each bootstrap sample can be controlled in the usual way. By default rpart.control is used with minsplit=2 and cp=0. The function prune.bagging can be used to prune each of the trees in a bagging object to an appropriate size.

Indirect classification

Especially in a medical context it often occurs that a priori knowledge about a classifying structure is given. For example it might be known that a disease is assessed on a subgroup of the given variables or, moreover, that class memberships are assigned by a deterministically known classifying function. Hand et al. (2001) proposes the framework of indirect classification which incorporates this a priori knowledge into a classification rule. In this framework we subdivide a given data set into three groups of variables: those to be used predicting the class membership (explanatory), those to be used defining the class membership (intermediate) and the class membership variable itself (response). For future observations, an indirect classifier predicts values for the appointed intermediate variables based on explanatory variables only. The observation is classified based on their predicted intermediate variables and a fixed classifying function. This indirect way of classification using the predicted intermediate variables offers possibilities to incorporate a priori knowledge by the subdivision of variables and by the construction of a fixed classifying function.

We apply indirect classification by using the function inclass. Referring to the glaucoma example, explanatory variables are HRT and anamnestic variables only, intermediate variables are w_{lora}, w_{cs} and w_{clv}. The response is the diagnosis of glaucoma which is determined by a fixed classifying function and therefore not included in the learning sample study.groupI. We assign the given variables to explanatory and intermediate by specifying the input formula.

```
R> formula.indirect <- clv + lora + cs ~ .
```

The variables on the left-hand side represent the intermediate variables, modeled by the explanatory variables on the right-hand side. Almost each modeling technique can be used to predict the intermediate variables. We chose a linear model by pFUN = lm.

```
R> fit <- inclass(formula.indirect,
+    pFUN = lm, data = study.groupI)
```

print.inclass displays the subdivision of variables and the chosen modeling technique

```
R> fit
Indirect classification, with 3
             intermediate variables:
clv lora cs

Predictive model per intermediate is lm
```

Indirect classification predicts the intermediate variables based on the explanatory variables and classifies them according to a fixed classifying function in a second step, that means a deterministically known function for the class membership has to be specified. In our example this function is given in Figure 2 and implemented in the function classify.

```
R> classify <- function (data) {
+   clv <- data$clv
+   lora <- data$lora
+   cs <- data$cs
+   res <- ifelse(
+     (!is.na(clv) & !is.na(lora) & clv >= 5.1 &
+       lora >= 49.23372) |
+     (!is.na(clv) & !is.na(lora) & !is.na(cs) &
+       clv < 5.1 & lora >= 58.55409 &
+       cs < 1.405) |
+     (is.na(clv) & !is.na(lora) & !is.na(cs)
+       & lora >= 58.55409 & cs < 1.405) |
+     (!is.na(clv) & is.na(lora) & cs < 1.405),
```

```
+     0, 1)
+   factor(res, labels = c("normal", "glaucoma"))
+ }
```

Prediction of future observations is now performed by

```
R> predict(object = fit, cFUN = classify,
+   newdata = study.group[c(1:3, 86:88),])
[1] normal   normal   normal
               glaucoma glaucoma glaucoma
```

We execute a bagged indirect classification approach by choosing `pFUN = bagging` and specifying the number of bootstrap samples (Peters et al., 2002). Regression or classification trees are fitted for each bootstrap sample, with respect to the measurement scale of the specified intermediate variables

```
R> fit <- inclass(formula.indirect,
+   pFUN = bagging, nbagg = 50,
+   data = study.groupI)
R> fit

Indirect classification, with 3
           intermediate variables:
lora cs clv

Predictive model per intermediate
    is bagging with 50 bootstrap replications
```

The call for the prediction of values remains unchanged.

Error rate estimation

Classification rules are usually assessed by their misclassification rate. Hence, error rate estimation is of main importance. The function `errorest` implements a unified interface to several resampling based estimators. Referring to the example, we apply a linear discriminant analysis and specify the error rate estimator by `estimator = "cv"`, `"boot"` or `"632plus"`, respectively. A 10-fold cross validation is performed by choosing `estimator = "cv"` and `est.para = list(k = 10)`. The options `estimator = "boot"` or `estimator = "632plus"` deliver a bootstrap estimator and its bias corrected version *.632+* (see Efron and Tibshirani, 1997), we specify the number of bootstrap samples to be drawn by `est.para = list(nboot = 50)`. Further arguments are required to particularize the classification technique. The argument `predict` represents the chosen predictive function. For a unified interface `predict` has to be based on the arguments `object` and `newdata` only, therefore a wrapper function `mypredict` is necessary for classifiers which require more than those arguments or do not return the predicted classes by default. For a linear discriminant analysis with `lda`, we need to specify

```
R> mypredict.lda <- function(object, newdata){
+    predict(object, newdata = newdata)$class}
```

and calculate a 10-fold-cross-validated error rate estimator for a linear discriminant analysis by calling

```
R> errorest(diagnosis ~ ., data= study.group,
+   model=lda, estimator = "cv",
+   predict= mypredict.lda)

10-fold cross-validation estimator
    of misclassification error

Data:  diagnosis on .

Error 0.2675
```

For the indirect approach the specification of the call becomes slightly more complicated. Again for a unified interface a wrapper function has to be used, which incorporates the fixed classification rule

```
R> mypredict.inclass <-
+   function(object, newdata){
+   predict.inclass(object = object,
+     cFUN = classify, newdata = newdata)
+  }
```

The bias corrected estimator *.632+* is computed by

```
R> errorest(formula.indirect,
+   data = study.groupI, model = inclass,
+   predict = mypredict.inclass,
+   estimator = "632plus",
+   iclass = "diagnosis", pFUN = lm)

.632+ Bootstrap estimator of misclassification
  error with 25 bootstrap replications

Data:  diagnosis

Error 0.2658
```

Because of the subdivision of variables and a formula describing the modeling between explanatory and intermediate variables only, we must call the class membership variable. Hence, in contrast to the function `inclass` the data set `study.groupI` used in `errorest` must contain explanatory, intermediate and response variables.

To summarize the performance of different classification techniques in the considered example of glaucoma diagnosis, the 10-fold cross-validated error estimator delivers the results given in the following table:

method	error estimate
lda	0.2237
rpart	0.2529
bagging	0.1882
double-bagging	0.1941
inclass-bagging	0.2059
inclass-lm	0.2294

lda denotes the linear discriminant analysis, *rpart* a classification tree, *bagging* bagging with 50 bootstrap samples, *double-bagging* bagging with 50 bootstrap samples, combined with LDA, *inclass-bagging* indirect classification using bagging and *inclass-lm* indirect classification using linear modeling.

Note that an estimator of the variance is available for the ordinary bootstrap estimator (`estimator="boot"`) only, see Efron and Tibshirani (1997).

Summary

ipred tries to implement a unified interface to some recent developments in classification and error rate estimation. It is by no means finished nor perfect and we very much appreciate comments, suggestions and criticism. Currently, the major drawback is speed. Calling `rpart` 50 times for each bootstrap sample is relatively inefficient but the design of interfaces was our main focus instead of optimization. Beside the examples shown, `bagging` can be used to compute bagged regression trees and `errorest` computes estimators of the mean squared error for regression models.

Bibliography

L. Breiman. Bagging predictors. *Machine Learning*, 24(2): 123–140, 1996a. 33

Leo Breiman. Out-of-bag estimation. Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34

B. Efron and R. Tibshirani. Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560, 1997. 33, 35, 36

D.J. Hand, H.G. Li, and N.M. Adams. Supervised classification with structured class definitions. *Computational Statistics & Data Analysis*, 36:209–225, 2001. 33, 34

T. Hothorn and B. Lausen. Double-bagging: Combining classifiers by bootstrap aggregation. submitted, 2002. preprint available under http://www.mathpreprints.com/. 34

T. Hothorn, I. Pal, O. Gefeller, B. Lausen, G. Michelson, and D. Paulus. Automated classification of optic nerve head topography images for glaucoma screening. In *Studies in Classification, Data Analysis, and Knowledge Organization (to appear)*. Proceedings of the 25th Annual Conference of the German Classification Society, 2002. 33

A. Peters, T. Hothorn, and B. Lausen. Glaucoma diagnosis by indirect classifiers. In *Studies in Classification, Data Analysis, and Knowledge Organization (to appear)*. Proceedings of the 8th Conference of the International Federation of Classification Societies, 2002. 33, 35

Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Medizininformatik, Biometrie und Epidemiologie, Waldstraße 6, D-91054 Erlangen
Andrea.Peters@imbe.imed.uni-erlangen.de
Torsten.Hothorn@rzmail.uni-erlangen.de
Berthold.Lausen@rzmail.uni-erlangen.de

Support from Deutsche Forschungsgemeinschaft SFB 539-C1/A4 is gratefully acknowledged.

Changes in R

by the R Core Team

User-visible changes

- XDR support is now guaranteed to be available, so the default save format will always be XDR binary files, and it is safe to distribute data in that format. (We are unaware of any platform that did not support XDR in recent versions of R.)

  `gzfile()` is guaranteed to be available, so the preferred method to distribute sizeable data objects is now via `save(compress = TRUE)`.

- `pie()` replaces `piechart()` and defaults to using pastel colours.

- `formatC()` has new arguments (see below) and `formatC(*, d = <dig>)` is no longer valid and must be written as `formatC(*, digits = <dig>)`.

- Missingness of character strings is treated much more consistently, and the character string `"NA"` can be used as a non-missing value.

- `summary.factor()` now uses a stable sort, so the output will change where there are ties in the frequencies.

New features

- Changes in handling missing character strings:

  – `"NA"` is no longer automatically coerced to a missing value for a character string. Use `as.character(NA)` where a missing value is required, and test via `is.na(x)`, not `x`