`ipred` : Improved Predictors

This short manual is heavily based on Peters et al. (2002b) and needs some improvements.

1 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.

2 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. A less complex, but although a less informative examination tool is the 2-dimensional fundus photography. 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., 2002a).

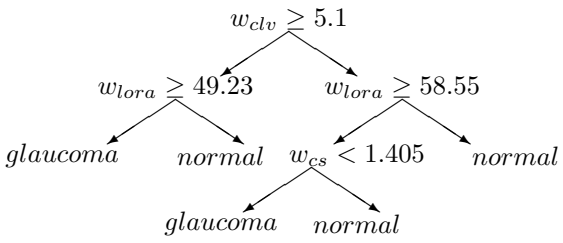

Figure 1: Glaucoma diagnosis.

Figure 1 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.

Two example datasets are included in the package. The first one contains measurements of the eye morphology only (`GlaucomaM`), including 62 variables for 196 observations. The second dataset (`GlaucomaMVF`) contains additional visual field measurements for a different set of patients. In both example datasets, the observations in the two groups are matched by age and sex to prevent any bias.

3 Bagging

Referring to the example of glaucoma diagnosis we first demonstrate the functionality of the `bagging` function. We fit `nbagg = 25` (default) classification trees for bagging by

```
>library(ipred)

Loading required package: rpart
Loading required package: MASS
Loading required package: mlbench
Loading required package: survival
Loading required package: splines
Loading required package: nnet
Loading required package: class

>data(GlaucomaM)
>gbag <- bagging(Class ~ ., data = GlaucomaM,
+     coob = TRUE)
```

where `GlaucomaM` contains explanatory HRT variables and the response of glaucoma diagnosis (`Class`), a factor at two levels `normal` and `glaucoma`. `print` returns informations about the returned 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).

```
>print(gbag)

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = Class ~ ., data = GlaucomaM, coob = TRUE)

Out-of-bag estimate of misclassification error:  0.1684
```

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 (2003) propose to use the out-of-bag samples for a combination of linear discriminant analysis and classification trees, called "Double-Bagging". For example, a combination of a stabilised linear disciminant analysis with classification trees can be computed along the following lines

```
>scomb <- list(list(model = slda, predict = function(object,
+     newdata) predict(object, newdata)$x))
>gbagc <- bagging(Class ~ ., data = GlaucomaM,
+     comb = scomb)
```

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

```
>predict(gbagc, newdata = GlaucomaM[c(1:3,
+     99:102), ])

[1] normal   normal   normal   glaucoma glaucoma
[6] glaucoma glaucoma
Levels: glaucoma normal
```

Both `bagging` and `predict` 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 `minsize=2` and `cp=0` and it is wise to turn cross-validation off (`xval=0`). The function `prune` can be used to prune each of the trees to an appropriate size.

4 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 `GlaucomaMVF`. We assign the given variables to explanatory and intermediate by specifying the input formula.

```
>data(GlaucomaMVF)
>GlaucomaMVF <- GlaucomaMVF[, -63]
>formula.indirect <- Class ~ 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 = list(list(model = lm))`.

```
>classify <- function(data) {
+     attach(data)
+     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)
+     detach(data)
+     factor(res, labels = c("glaucoma",
+         "normal"))
+ }
>fit <- inclass(formula.indirect, pFUN = list(list(model = lm)),
+     cFUN = classify, data = GlaucomaMVF)
```

`print` displays the subdivision of variables and the chosen modeling technique

```
>print(fit)
```

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

Predictive model per intermediate is lm
```

Furthermore, 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 1 and implemented in the function `classify`.
Prediction of future observations is now performed by

```
>predict(object = fit, newdata = GlaucomaMVF[c(1:3,
+     86:88), ])

        The following object(s) are masked from package:methods :

         as

[1] normal   normal   normal   glaucoma glaucoma
[6] glaucoma
Levels: glaucoma normal
```

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

```
>mypredict.rpart <- function(object, newdata) {
+     RES <- predict(object, newdata)
+     RET <- rep(NA, nrow(newdata))
+     NAMES <- rownames(newdata)
```

```
+     RET[NAMES %in% names(RES)] <- RES[NAMES[NAMES %in%
+         names(RES)]]
+     RET
+ }
>fit <- inbagg(formula.indirect, pFUN = list(list(model = rpart,
+     predict = mypredict.rpart)), cFUN = classify,
+     nbagg = 25, data = GlaucomaMVF)
```

The call for the prediction of values remains unchanged.

5 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 = control.errorest(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 = control.errorest(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

```
>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

```
>errorest(Class ~ ., data = GlaucomaM,
+     model = lda, estimator = "cv", predict = mypredict.lda)

Call:
errorest.data.frame(formula = Class ~ ., data = GlaucomaM, model = lda,
    predict = mypredict.lda, estimator = "cv")

         10-fold cross-validation estimator of misclassification error

Misclassification error:  0.2041
```

For the indirect approach the specification of the call becomes slightly more complicated. The bias corrected estimator *.632+* is computed by

```
>errorest(formula.indirect, data = GlaucomaMVF,
+     model = inclass, estimator = "632plus",
+     pFUN = list(list(model = lm)), cFUN = classify)

        The following object(s) are masked from package:methods :

         as


        The following object(s) are masked from package:methods :

         as
```

```
The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as
```

```
The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as
```

```
The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as
```

```
The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as


The following object(s) are masked from package:methods :

 as
```

```
Call:
errorest.data.frame(formula = formula.indirect, data = GlaucomaMVF,
    model = inclass, estimator = "632plus", pFUN = list(list(model = lm)),
    cFUN = classify)

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

Misclassification error:  0.2417
```

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 `GlaucomaMVF` used in `errorest` must contain explanatory, intermediate and response variables.

Sometimes it may be necessary to reduce the number of predictors before training a classifier. Estimating the error rate after the variable selection leads to biased estimates of the misclassfication error and therefore one should estimate the error rate of the whole procedure. Within the `errorest` framework, this can be done as follows. First, we define a function which does both variable selection and training of the classifier. For illustration proposes, we select the predictors by comparing their univariate P-values of a two-sample t-test with a prespecified level and train a LDA using the selected variables only.

```
>mymod <- function(formula, data, level = 0.05) {
+     sel <- which(lapply(data, function(x) {
+         if (!is.numeric(x))
+             return(1)
+         else return(t.test(x ~ data$Class)$p.value)
+     }) < level)
+     sel <- c(which(colnames(data) %in%
```

```
+         "Class"), sel)
+     mod <- lda(formula, data = data[,
+         sel])
+     function(newdata) {
+         predict(mod, newdata = newdata[,
+             sel])$class
+     }
+ }
```

Note that `mymod` does not return an object of class `lda` but a function with argument `newdata` only. Thanks to lexical scoping, this function is used for computing predicted classes instead of a function `predict` passed to `errorest` as argument. Computing a 5-fold cross-validated error rate estimator now is approximately a one-liner.

```
>errorest(Class ~ ., data = GlaucomaM,
+     model = mymod, estimator = "cv", est.para = control.errorest(k = 5))

Call:
errorest.data.frame(formula = Class ~ ., data = GlaucomaM, model = mymod,
    estimator = "cv", est.para = control.errorest(k = 5))


        5-fold cross-validation estimator of misclassification error

Misclassification error:  0.2602
```

6 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 bagging for regression trees and `errorest` computes estimators of the mean squared error for regression models.

References

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a. 1

Leo Breiman. Out-of-bag estimation. Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 1, 4

B. Efron and R. Tibshirani. Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92 (438):548–560, 1997. 1, 8

D.J. Hand, H.G. Li, and N.M. Adams. Supervised classification with structured class definitions. *Computational Statistics & Data Analysis*, 36: 209–225, 2001. 1, 5

Torsten Hothorn and Berthold Lausen. Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, 36(6):1303–1309, 2003. 4

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, 2002a. 2, 7

Andrea Peters, Torsten Hothorn, and Berthold Lausen. ipred: Improved predictors. *R News*, 2(2):33–36, June 2002b. URL `http://CRAN.R-project.org/doc/Rnews/`. 1