Consensus models weighted by AUC for multiple class responses

Andreas Dominik Cullmann

February 1, 2013

1 Introduction

This vignette shows how to build a consensus model for the `fgl` data (see [Venables and Ripley, 2002] or `?MASS::fgl`). We will follow the modelling process shown in [Marmion et~al., 2009, Figure 1] restricting ourselves to only two 'Single-models': a classification tree and a multinomial log-linear model using package `rpart` and package `nnet`, respectively.

2 Example

After loading the data, we create random indices for training and evaluation sets (since the training and evaluation sets are complements with respect to the data set, two indices are redundant, but `fgl[ind.eval, ]` might be easier to read than `fgl[!ind.train, ]`).

```
> library(MASS)
> data(fgl)
> set.seed(100)
> ind.train <- sample(nrow(fgl)
+                     , size = floor(nrow(fgl)*.7)
+                     , replace = FALSE
+                     )
> ind.eval <- setdiff(seq(1:nrow(fgl)), ind.train)
```

Choosing `fgl$type` as response and all other variables as predictors, we calibrate a classification tree using `rpart::rpart` (cf. [Venables and Ripley, 2002, p. 264]):

```
> library(rpart)
> set.seed(123)
> fgl.rpart <- rpart(type ~ .
+                   , data = fgl[ind.train, ]
+                   , parms = list(split = "information")
+                   )
> newcp <- max(fgl.rpart$cptable[,"CP"] *
+              as.numeric(fgl.rpart$cptable[,"xerror"] <
+                         sum(fgl.rpart$cptable[dim(fgl.rpart$cptable)[1]
```

```
+                                                 , c("xerror","xstd")]))
+             ) + 1e-13
> fgl.rpart.pruned <- prune(fgl.rpart, cp = newcp)
```

and a multinomial log-linear model using `nnet::multinom`:

```
> library(nnet)
> fgl.multinom <- multinom(type ~ .
+                           , data = fgl[ind.train, ]
+                           , trace = FALSE
+                           )
```

We can now assess the model accuracy using either a confusion matrix of the classified predictions

```
> library(mda)
> confusion(predict(fgl.rpart.pruned
+                   , newdata = fgl[ind.eval, ]
+                   , type = "class")
+          ,fgl[ind.eval, ]$type)

         true
predicted WinF WinNF Veh Con Tabl Head
    WinF    25     6   3   0    0    1
    WinNF    3    10   0   5    2    0
    Veh      0     0   0   0    0    0
    Con      0     0   0   0    0    0
    Tabl     0     0   0   0    0    0
    Head     0     0   0   0    3    7
attr(,"error")
[1] 0.3538462

> confusion(predict(fgl.multinom
+                   , newdata = fgl[ind.eval, ]
+                   , type = "class")
+          , fgl[ind.eval, ]$type)

         true
predicted WinF WinNF Veh Con Tabl Head
    WinF    20     3   1   0    0    0
    WinNF    7    11   0   4    0    0
    Veh      1     0   2   0    0    0
    Con      0     0   0   1    0    0
    Tabl     0     1   0   0    2    1
    Head     0     1   0   0    3    7
attr(,"error")
[1] 0.3384615
```

or a multiple class version of AUC using the raw predictions:

```
> library(HandTill2001)
> auc(multcap(response = fgl[ind.eval, ]$type
+             , predicted = predict(fgl.rpart.pruned
+                                   , newdata = fgl[ind.eval, ])
+             ))
```

```
[1] 0.8401488
```

```
> auc(multcap(response = fgl[ind.eval, ]$type
+             , predicted = predict(fgl.multinom
+                                   , newdata = fgl[ind.eval, ]
+                                   , type = "probs"
+                                   )
+             ))
```

```
[1] 0.8299983
```

To enhance predictive performance, we decide to use both models to build a consenus model. Furthermore, we want to use the 'weighted average' consensus method given by [Marmion et~al., 2009, Eqn 1], which uses the pre-evaluated AUC of the models as weights. So we split the training set into two complementary subsets: 'inner training' and 'inner evaluation'.

```
> set.seed(100)
> ind.inner.train <- sample(ind.train
+                           , size = floor(length(ind.train)*0.7)
+                           , replace = FALSE
+                           )
> ind.inner.eval <- setdiff(ind.train, ind.inner.train)
```

We then refit our two models to the 'inner training' data:

```
> wa.fgl.multinom <- multinom(fgl.multinom
+                             , data = fgl[ind.inner.train, ]
+                             , trace = FALSE
+                             )
> wa.fgl.rpart <- rpart(type ~ .
+                       , data = fgl[ind.inner.train, ]
+                       , parms = list(split = "information")
+                       )
> newcp <- max(wa.fgl.rpart$cptable[,"CP"] *
+             as.numeric(wa.fgl.rpart$cptable[,"xerror"] <
+                        sum(wa.fgl.rpart$cptable[dim(wa.fgl.rpart$cptable)[1]
+                                                 , c("xerror","xstd")]))
+             ) + 1e-13
> wa.fgl.rpart.pruned <- prune(wa.fgl.rpart, cp = newcp)
```

and calculate pre-evaluation AUC (which we save in a `list`) using the 'inner evaluation' data and the refitted models:

```
> li <- list()
> li$rpart$auc <- auc(multcap(response = fgl[ind.inner.eval, ]$type
+                             , predicted = predict(wa.fgl.rpart.pruned
+                                 , newdata = fgl[ind.inner.eval, ])))
> li$mllm$auc <- auc(multcap(response = fgl[ind.inner.eval, ]$type
+                            , predicted = predict(wa.fgl.multinom
+                                , newdata = fgl[ind.inner.eval, ]
+                                , type = "probs")))
```

We add the predictions using the models (the 'original' or 'Single' ones, not the refitted) for the evaluation set

```
> li$rpart$predictions <- predict(fgl.rpart.pruned
+                                  , newdata = fgl[ind.eval, ])
> li$mllm$predictions <- predict(fgl.multinom
+                                 , newdata = fgl[ind.eval, ]
+                                 , type = "probs")
```

and obtain the consensus predictions as ([Marmion et~al., 2009, Eqn 1])

```
> predicted <- Reduce('+', lapply(li, function(x)
+                                  x$auc * x$predictions)
+ ) / Reduce('+', sapply(li,"[", "auc"))
```

which perform (slightly) better than the predictions using the 'single models':

```
> auc(multcap(response= fgl[ind.eval, ]$type
+              , predicted = predicted)
+     )

[1] 0.8773105
```

To classify the predicted probabilities, we choose the class with highest predicted probability (which.max gives the _first_ maximum):

```
> classes.predicted <-
+   factor(x =
+          apply(predicted
+                , 1
+                , function(x)
+                dimnames(predicted)[[2]][which.max(x)])
+          , dimnames(predicted)[[2]]
+          )
```

References

[Marmion et~al., 2009] Marmion, M., Parviainen, M., Luoto, M., Heikkinen, R.~K., and Thuiller, W. (2009). Evaluation of consensus methods in predictive species distribution modelling. *Diversity and Distributions*, 15(1):59–69.

[Venables and Ripley, 2002] Venables, W.~N. and Ripley, B.~D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.