Fast-and-frugal classification in R (ffcr)

Marcus Buckmann and Özgür Şimşek

2021-04-28

The *ffcr* R package is used to construct two families of transparent classification models: *fast-and-frugal trees* and *tallying* models. The book *Classification in the Wild: The Science and Art of Transparent Decision Making.* (Katikopoulos et al. 2020) describes these models, their applications and the algorithms to construct them in detail.

A fast-and-frugal tree is a decision tree with a simple structure: one branch of each node exits the tree, the other continues to the next node until the final node is reached. A tallying model gives pieces of evidence the same weight. The package contains two main functions: `fftree` to train fast-and-frugal trees and `tally` to train tallying models.

To illustrate the functionality of the package, we use the *Liver* data set (Ramana, Babu, and Venkateswarlu 2011) that we obtained from the UCI machine learning repository (Dua and Graff 2017). It contains 579 patients of which 414 have a liver condition and the other 165 do not. We predict which patient has a liver condition using medical measures and the age and gender of the people.

Training fast-and-frugal trees

The `fftree` function encompasses three different methods to train fast-and-frugal trees. These are named *basic*, *greedy*, and *cross-entropy.*[1]

We train a fast-and-frugal tree on the Liver data set. When the first column of a data set contains the class labels that we want to predict, we can simply pass the data set as the first argument. We limit the size of the tree to at most four nodes. The *greedy* method is the default algorithm. It is fast and usually produces accurate trees.

```r
library(ffcr)
model <- fftree(liver, method = "greedy", max_depth = 4)
```

Alternatively, we can call the *fftree* function using the formula syntax. Here we train the fast-and-frugal tree using only a few selected features.

```r
fftree(diagnosis ~ age + albumin + proteins + aspartate, data = liver, max_depth = 4)
```

Printing the model shows the structure of the tree and its fitting performance in the data set. Additionally to standard performance measures such as accuracy and the F1 score, the output also states the *depth* of the tree, the number of unique *features* that are split in the tree, and the *frugality*, which is the average number of nodes visited until a prediction is made.

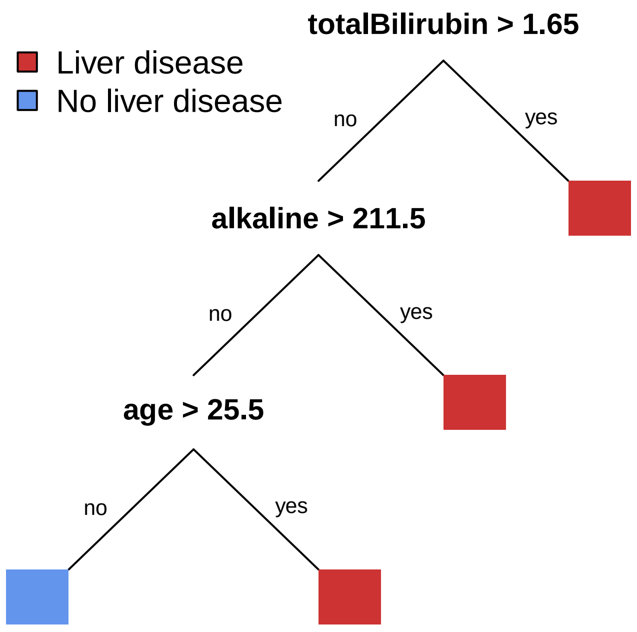
```r
print(model)
#> Fast-and-frugal Tree object
#> Trained with : "recursive" method.
#>
```

[1] In the book (Katikopoulos et al. 2020), we refer to the cross-entropy optimization (Rubinstein 1999) as the *best-fit* method. It does not guarantee to find the best possible tree but often produces more accurate trees than the other methods.

```
#> Call:
#> fftree(data = data, method = "greedy", max_depth = 4, formula = formula)
#>
#> Formula:
#> diagnosis ~ age + sex + totalBilirubin + directBilirubin + alkaline +
#>     alamine + aspartate + proteins + albumin + albuminGlobulin
#>
#> Tree:
#> Reason / Prediction / (Proportion of class 'Liver disease') / (Number of objects classified)
#>
#>   totalBilirubin > 1.65: Liver disease (0.91) (213)
#>     alkaline > 211.5: Liver disease (0.76) (132)
#>       age <= 25.5: No liver disease (0.30) (27)
#>         age > 25.5: Liver disease (0.55) (207)
#>
#>
#> Fitted values:
#>     Observed         Predicted        N
#>     Liver disease    Liver disease    406
#>     No liver disease Liver disease    146
#>     Liver disease    No liver disease   8
#>     No liver disease No liver disease  19
#>
#> Fitting performance:
#>     Accuracy            0.73
#>     Sensitivity         0.98
#>     Specificity         0.12
#>     Balanced accuracy   0.55
#>     F1 score            0.84
#>
#>     Depth     3.00
#>     Features  3.00
#>     Frugality 2.04
```

To visualize the tree we use

```
plot(model)
```



The sensitivity of this tree is very high, while the specificity is low—the tree nearly always predicts *Liver disease*. This produces an accurate tree when considering the number of total misclassifications as the relevant performance metric. To increase the specificity of the tree, we can weigh the observations such that both classes of patients get the same share. Let p be the proportion of patients that have liver disease. We weigh the patients with liver disease by 1-p, and the patients without disease by p. Doing that the specificity increases substantially:

```r
p <- sum(liver$diagnosis == "Liver disease")/nrow(liver)
weights <- c("No liver disease" = p, "Liver disease" = 1 - p)
model <- fftree(liver, weights = weights, cv = TRUE, max_depth = 4)
model
#> Fast-and-frugal Tree object
#> Trained with : "recursive" method.
#>
#> Call:
#> fftree(data = data, max_depth = 4, weights = weights, cv = TRUE,
#>     formula = formula)
#>
#> Formula:
#> diagnosis ~ age + sex + totalBilirubin + directBilirubin + alkaline +
#>     alamine + aspartate + proteins + albumin + albuminGlobulin
#>
#> Tree:
#> Reason / Prediction / (Proportion of class 'Liver disease') / (Number of objects classified)
```

```
#>
#>   directBilirubin > 1.05: Liver disease (0.94) (165)
#>     alamine > 66.5: Liver disease (0.89) (61)
#>       alkaline <= 211.5: No liver disease (0.52) (242)
#>         age <= 27.5: No liver disease (0.50) (26)
#>           age > 27.5: Liver disease (0.78) (85)
#>
#>
#> Fitted values:
#>     Observed         Predicted        N
#>     Liver disease    Liver disease    275
#>     No liver disease Liver disease     36
#>     Liver disease    No liver disease 139
#>     No liver disease No liver disease 129
#>
#> Fitting performance:
#>     Accuracy            0.70
#>     Sensitivity         0.66
#>     Specificity         0.78
#>     Balanced accuracy   0.72
#>     F1 score            0.76
#>
#>     Depth     4.00
#>     Features  4.00
#>     Frugality 2.52
#>
#> Cross-validation performance:
#>     Accuracy            0.64
#>     Sensitivity         0.60
#>     Specificity         0.75
#>     Balanced accuracy   0.67
#>     F1 score            0.70
#>
#>     Depth     3.80
#>     Features  3.80
#>     Frugality 2.47
```

By default, the *fftree* method fits a fast-and-frugal tree to the complete data set. Here we have set 'cv = TRUE' to additionally estimate the predictive performance of the tree using 10-fold cross-validation.

To make predictions according to a fast-and-frugal tree, we can use the `predict` function. It either returns the class label (*response*), or the performance across the observations (*metric*). Note that for the latter, the class labels need to be included in the data that is passed to the predict function.

```
model <- fftree(diagnosis ~ ., data = liver[1:300,], weights = c(1-p,p), max_depth = 4)

predict(model, newdata = liver[301:310,], type = "response")
#>  [1] "Liver disease" "Liver disease" "Liver disease" "Liver disease"
#>  [5] "Liver disease" "Liver disease" "Liver disease" "Liver disease"
#>  [9] "Liver disease" "Liver disease"
predict(model, newdata = liver[301:nrow(liver),], type = "metric")
#>          Accuracy       Sensitivity       Specificity Balanced accuracy
#>        0.70250896        0.97409326        0.09302326        0.53355826
#>          F1 score    True positives   False positives    True negatives
#>        0.81917211      188.00000000       78.00000000        8.00000000
#>   False negatives 
#>        5.00000000 
```

Learning tallying models

The package implements two different methods to train tallying models, *basic* and *cross-entropy*. As for fast-and-frugal trees, we set a maximum size (*max_size*) of four to obtain a simple model and weigh the observations to make sure that the tallying model strikes a good balance between high sensitivity and specificity. The *predict* function is used in the same way as it is for fast-and-frugal trees.

```
p <- sum(liver$diagnosis == "Liver disease")/nrow(liver)
weights <- c("No liver disease" = p, "Liver disease" = 1 - p)
model <- tally(diagnosis ~ ., data = liver[1:300,], weights = weights, max_size = 4)
model
#> Tallying object
#> Trained with : "basic" method.
#> 
#> Call:
#> tally(data = data, formula = formula, max_size = 4, weights = weights)
#> 
#> Formula:
#> diagnosis ~ age + sex + totalBilirubin + directBilirubin + alkaline + 
#>     alamine + aspartate + proteins + albumin + albuminGlobulin
#> 
#> Reasons:
#>    + age              >   27.50
#>    + directBilirubin  >    0.65
#>    + alkaline         >  209.50
#>    ------------------------------------
#> Predict Liver disease if at least 2 reasons hold.
#> 
#> 
#> Fitted values:
#>     Observed         Predicted        N
#>     Liver disease    Liver disease    153
#>     No liver disease Liver disease     20
#>     Liver disease    No liver disease  68
#>     No liver disease No liver disease  59
#> 
#> Fitting performance:
#>     Accuracy            0.71
#>     Sensitivity         0.69
#>     Specificity         0.75
#>     Balanced accuracy   0.72
#>     F1 score            0.78
predict(model, newdata = liver[301:nrow(liver),], type = "metric")
#>          Accuracy       Sensitivity       Specificity Balanced accuracy 
#>         0.6559140         0.6010363         0.7790698         0.6900530 
#>          F1 score    True positives   False positives    True negatives 
#>         0.7073171       116.0000000        19.0000000        67.0000000 
#>   False negatives 
#>        77.0000000 
```

References

Dua, Dheeru, and Casey Graff. 2017. "UCI Machine Learning Repository." University of California, Irvine, School of Information; Computer Sciences. http://archive.ics.uci.edu/ml.

Katikopoulos, Konstantinos V., Özgür Şimşek, Marcus Buckmann, and Gerd Gigerenzer. 2020. *Classification in the Wild: The Science and Art of Transparent Decision Making.* MIT Press.

Ramana, Bendi Venkata, M. Surendra Prasad Babu, and N. B. Venkateswarlu. 2011. "A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis." *International Journal of Database Management Systems* 3 (2): 101–14.

Rubinstein, Reuven. 1999. "The Cross-Entropy Method for Combinatorial and Continuous Optimization." *Methodology and Computing in Applied Probability* 1 (2): 127–90.