orf: Ordered Random Forests

Michael Lechner & Gabriel Okasa

e-Rum 2020

SEW-HSG
Swiss Institute for Empirical Economic Research
University of St.Gallen, Switzerland

Introduction

Ordered Choice Models

- categorical dependent variable with inherent ordering
- example: product quality
  1. very good
  2. good
  3. neutral
  4. bad
  5. very bad
- many other examples such as education level, income level, opinion surveys, ratings, sport outcomes, ...
- ordered nature should be taken into account

Parametric Models

- ordered probit & ordered logit
- assumptions about the distribution of the error term
- estimation usually via maximum likelihood

Quantities of Interest:

- choice probabilities:

$$P[Y_i = m \mid X_i = x]$$

- marginal effects:

$$\frac{\partial P[Y_i = m \mid X_i = x]}{\partial x^k}$$

Ordered Forest

- estimation of ordered choice model based on the random forest (Breiman 2001)
- improves on ***parametric*** models by allowing for flexible functional form
- improves on ***nonparametric*** models by allowing for larger covariate space
- alternative to standard ordered probit and ordered logit with:
  - conditional choice probabilities
  - marginal effects
  - approximate inference
- theoretical foundations developed in Lechner and Okasa (2019)

R-package

R-package

- implementation of the Ordered Forest estimator
- available from the CRAN repository (version 0.1.3)
- source code available on GitHub
- heavy-lifting done with C++ via `Rcpp` package (Eddelbuettel and François 2011)
- underlying forests are based on the `ranger` package (Wright and Ziegler 2017)

```r
# install orf package
install.packages("orf", dependencies = c("Imports", "Suggests"))
```

Estimation

Ordered Forest

Algorithm 1: ORDERED FOREST

Input: Data (X_i, Y_i); $Y_i \in \{1, ..., M\}$

Output: Probabilities $\hat{P}_{m,i} = \hat{P}[Y_i = m \mid X_i = x]$

begin

  CUMULATIVE PROBABILITIES;

  for $m = 1$ **to** $M - 1$ **do**

    create binary indicator variables: $Y_{m,i} = \mathbf{1}(Y_i \leq m)$;

    estimate regression random forest: $P[Y_{m,i} = 1 \mid X_i = x]$;

    predict conditional probabilities: $\hat{Y}_{m,i} = \hat{P}[Y_{m,i} = 1 \mid X_i = x]$;

  ORDERED CLASS PROBABILITIES;

  for $m = 2$ **to** M **do**

    compute class probabilities: $\hat{P}_{m,i} = \hat{Y}_{m,i} - \hat{Y}_{m-1,i}$;

    if $\hat{P}_{m,i} < 0$ **then**

      set $\hat{P}_{m,i} = 0$;

    set $\hat{P}_{m,i} = \frac{\hat{P}_{m,i}}{\sum_{m=1}^{M} \hat{P}_{m,i}}$;

orf: data()

- load the orf package and an example dataset

```
# load the orf package
library("orf")
# load example data
data(odata)
```

- define the inputs as a vector of outcomes Y and a matrix of features X

```
# specify response and covariates
Y <- as.numeric(odata[, 1])
X <- as.matrix(odata[, -1])
```

orf: `orf()`

- conditional choice probabilities $P[Y_i = m \mid X_i = x]$ as a target of interest
- estimate the probabilities by the Ordered Forest using the main function `orf()`
- arguments include the data and the forest-specific tuning parameters

```
# estimate ordered forest with user specified settings
orf_model <- orf(X, Y, num.trees = 1000, mtry = 2, min.node.size = 5,
                       replace = FALSE, sample.fraction = 0.5,
                       honesty = TRUE, honesty.fraction = 0.5,
                       inference = FALSE, importance = FALSE)
```

orf: orf()

- fitted probabilities $\hat{P}[Y_i = m \mid X_i = x]$ as the main output

```r
# predicted probabilities for each outcome category
head(orf_model$predictions)
```

```
#>      Category 1 Category 2 Category 3
#> [1,] 0.80427874  0.1272509 0.06847033
#> [2,] 0.52357922  0.2905586 0.18586215
#> [3,] 0.30901512  0.2997291 0.39125575
#> [4,] 0.16406209  0.5175266 0.31841135
#> [5,] 0.38910222  0.4460181 0.16487966
#> [6,] 0.07452973  0.1023059 0.82316437
```

- access to underlying forests through `orf_model$forests`
- access to accuracy measures through `orf_model$accuracy`
- and many more. . .

orf: print.orf()

```
# print the output of the orf estimation
print(orf_model)
```

```
#> Ordered Forest object of class orf
#>
#> Number of Categories:                    3
#> Sample Size:                             1000
#> Number of Trees:                         1000
#> Build:                                   Subsampling
#> Mtry:                                    2
#> Minimum Node Size:                       5
#> Honest Forest:                           TRUE
#> Weight-Based Inference:                  FALSE
```

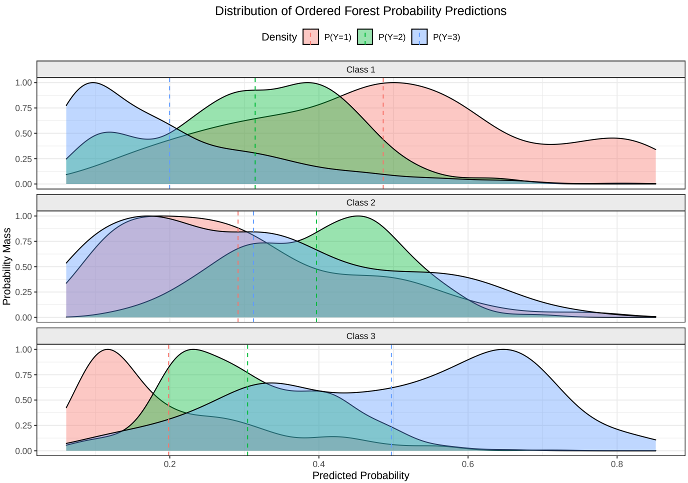
orf: summary.orf()

```r
# summarize the output of the orf estimation
summary(orf_model, latex = FALSE)
```

```
#> Summary of the Ordered Forest Estimation
#>
#> type               Ordered Forest
#> categories         3
#> build              Subsampling
#> num.trees          1000
#> mtry               2
#> min.node.size      5
#> replace            FALSE
#> sample.fraction    0.5
#> honesty            TRUE
#> honesty.fraction   0.5
#> inference          FALSE
#> importance         FALSE
#> trainsize          500
#> honestsize         500
#> features           4
#> mse                0.50974
#> rps                0.1559
```

orf: `plot.orf()`

```r
# plot the estimated probability distributions
plot(orf_model)
```



Prediction

Prediction

- split your data randomly into train and test sample

```r
# specify response and covariates for train and test
idx <- sample(seq(1, nrow(odata), 1), 0.8*nrow(odata))
# train set
Y_train <- odata[idx, 1]
X_train <- odata[idx, -1]
# test set
Y_test <- odata[-idx, 1]
X_test <- odata[-idx, -1]
```

- estimate the Ordered Forest using the training set

```r
# estimate Ordered Forest with default settings
orf_train <- orf(X_train, Y_train)
```

orf: `predict.orf()`

- predict the ***probabilities*** $\hat{P}[Y_i = m \mid X_i = x]$ for the test set

```
# predict the probabilities with the estimated orf
orf_test <- predict(orf_train, newdata = X_test, type = "probs", inference = FALSE)
```

- predict the ***classes*** $\hat{Y} = m$ for which $\max_{m=1,\dots,M} \hat{P}[Y_i = m \mid X_i = x]$ for the test set

```
# predict the probabilities with the estimated orf
orf_test <- predict(orf_train, newdata = X_test, type = "class", inference = FALSE)
```

- visualize the output using `print()` and `summary()` commands

Effects

Effects

- estimate the marginal effect for ***categorical*** x^k as discrete change

$$\hat{ME}_i^{k,m}(x) = \left\{ \hat{P}\left[Y_i = m \mid X_i^k = \left\lceil x^k \right\rceil, X_i^{-k} = x^{-k}\right] - \hat{P}\left[Y_i = m \mid X_i^k = \left\lfloor x^k \right\rfloor, X_i^{-k} = x^{-k}\right] \right\}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote rounding up and down to the nearest integer

- estimate the marginal effect for ***continuous*** x^k as numeric approximation

$$\hat{ME}_i^{k,m}(x) = \frac{1}{2h} \left\{ \hat{P}\left[Y_i = m \mid X_i^k = x^k + h, X_i^{-k} = x^{-k}\right] - \hat{P}\left[Y_i = m \mid X_i^k = x^k - h, X_i^{-k} = x^{-k}\right] \right\}$$

where h is the evaluation window for the effect

Inference

- Wager and Athey (2018) prove consistency and normality of the RF predictions
  - subsampling & honesty
- weighting representation of ordered forest predictions

$$\hat{P}_{m,i} = \sum_{i=1}^{N} \hat{w}_{m,i}(x) Y_{m,i} - \sum_{i=1}^{N} \hat{w}_{m-1,i}(x) Y_{m-1,i}$$

- use forest weights for deriving the variance of the estimator
- adaptation of the weight-based inference as proposed in Lechner (2019)
- crucial condition:
  - weights and outcomes must be independent $\rightarrow$ sample splitting
  - requiring honest forest instead of honest trees only

orf: `margins.orf()`

- marginal effect at the mean: $\hat{ME}_i^{k,m}(\bar{x})$

```
# evaluate marginal effects of the ordered forest at the mean
orf_margins <- margins(orf_model, eval = "atmean", window = 0.1,
                                  inference = TRUE, newdata = NULL)
```

- mean marginal effect: $\frac{1}{N}\sum_{i=1}^{N} \hat{ME}_i^{k,m}(x)$

```
# evaluate mean marginal effects of the ordered forest
orf_margins <- margins(orf_model, eval = "mean", window = 0.1,
                                  inference = TRUE, newdata = NULL)
```

orf: margins.orf() I

```
summary(orf_margins, latex = FALSE) # summary of marginal effects
```

```
#> Summary of the Ordered Forest Margins
#>
#>
#> type              Ordered Forest Margins
#> evaluation.type   mean
#> evaluation.window 0.1
#> new.data          FALSE
#> categories        3
#> build             Subsampling
#> num.trees         1000
#> mtry              2
#> min.node.size     5
#> replace           FALSE
#> sample.fraction   0.5
#> honesty           TRUE
#> honesty.fraction  0.5
#> inference         TRUE
#>
#> ORF Marginal Effects:
#>
#> ---------------------------------------------------------------------------
```

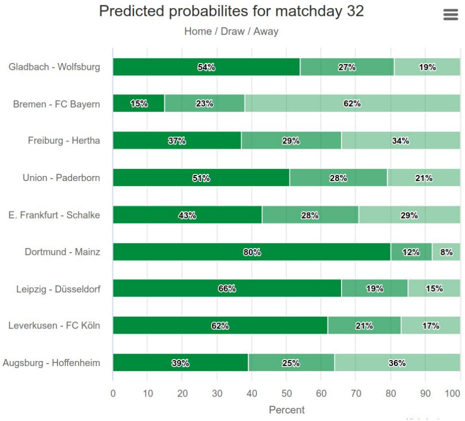
orf: `margins.orf()` II

```
#> X1
#>                    Class    Effect    StdErr    tValue    pValue
#>                      1     -0.1145    0.0234   -4.9019    0.0000    ***
#>                      2     -0.0163    0.0229   -0.7152    0.4745
#>                      3      0.1309    0.0304    4.2988    0.0000    ***
#> X2
#>                    Class    Effect    StdErr    tValue    pValue
#>                      1     -0.1098    0.0269   -4.0850    0.0000    ***
#>                      2     -0.0232    0.0371   -0.6238    0.5328
#>                      3      0.1329    0.0479    2.7741    0.0055    ***
#> X3
#>                    Class    Effect    StdErr    tValue    pValue
#>                      1     -0.1614    0.0416   -3.8816    0.0001    ***
#>                      2      0.0204    0.0445    0.4591    0.6461
#>                      3      0.1409    0.0623    2.2622    0.0237    **
#> X4
#>                    Class    Effect    StdErr    tValue    pValue
#>                      1      0.0020    0.0016    1.2403    0.2149
#>                      2      0.0000    0.0017    0.0120    0.9905
#>                      3     -0.0020    0.0021   -0.9441    0.3451
#> ----------------------------------------------------------------------
#> Significance levels correspond to: *** .< 0.01, ** .< 0.05, * .< 0.1
#> ----------------------------------------------------------------------
```

Apps

Apps

- SEW Soccer Analytics
- predicting soccer game outcomes
- probabilities of loss, draw, and win
- simulating the German Bundesliga
- weekly updates on Twitter
- more details in Goller et al. (2018)



Conclusion

Conclusion

- Ordered Forest as a new flexible ML estimator for ordered choice models
- as flexible as machine learning methods
- as interpretable as classical econometrics methods

- `orf` package implementing the estimator in R
- available on CRAN repository (version 0.1.3)
- supports S3 methods like `predict()`, `summary()`, `plot()`, ...

Thanks

Contact

gabriel.okasa@unisg.ch
okasag.github.io

References

References I

Breiman, L. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Eddelbuettel, Dirk, and Romain François. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. https://doi.org/10.18637/jss.v040.i08.

Goller, Daniel, Michael C. Knaus, Michael Lechner, and Gabriel Okasa. 2018. "Predicting Match Outcomes in Football by an Ordered Forest Estimator." *Economics Working Paper Series* No.1811. https://ideas.repec.org/p/usg/econwp/201811.html.

Lechner, Michael. 2019. "Modified Causal Forests for Estimating Heterogeneous Causal Effects." *CEPR Discussion Paper No. DP13430*. https://papers.ssrn.com/sol3/papers.cfm?abstract{_}id=3314050.

Lechner, Michael, and Gabriel Okasa. 2019. "Random Forest Estimation of the Ordered Choice Model."

References II

Wager, Stefan, and Susan Athey. 2018. "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests." *Journal of the American Statistical Association*, 1228–42. https://doi.org/10.1080/01621459.2017.1319839.

Wright, Marvin N., and Andreas Ziegler. 2017. "ranger : A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.