Journal of Statistical Software
September 2010, Volume 36, Issue 11. http://www.jstatsoft.org/

Feature Selection with the Boruta Package

Miron B. Kursa
University of Warsaw

Witold R. Rudnicki
University of Warsaw

Abstract

This article describes a R package **Boruta**, implementing a novel feature selection algorithm for finding *all relevant variables*. The algorithm is designed as a wrapper around a Random Forest classification algorithm. It iteratively removes the features which are proved by a statistical test to be less relevant than random probes. The **Boruta** package provides a convenient interface to the algorithm. The short description of the algorithm and examples of its application are presented.

Keywords: feature selection, feature ranking, random forest.

1. Introduction

Feature selection is often an important step in applications of machine learning methods and there are good reasons for this. Modern data sets are often described with far too many variables for practical model building. Usually most of these variables are irrelevant to the classification, and obviously their relevance is not known in advance. There are several disadvantages of dealing with overlarge feature sets. One is purely technical — dealing with large feature sets slows down algorithms, takes too many resources and is simply inconvenient. Another is even more important — many machine learning algorithms exhibit a decrease of accuracy when the number of variables is significantly higher than optimal (Kohavi and John 1997). Therefore selection of the small (possibly minimal) feature set giving best possible classification results is desirable for practical reasons. This problem, known as *minimal-optimal* problem (Nilsson, Peña, Björkegren, and Tegnér 2007), has been intensively studied and there are plenty of algorithms which were developed to reduce feature set to a manageable size.

Nevertheless, this very practical goal shadows another very interesting problem — the identification of all attributes which are in some circumstances relevant for classification, the so-called *all-relevant* problem. Finding all relevant attributes, instead of only the non-redundant ones,

may be very useful in itself. In particular, this is necessary when one is interested in understanding mechanisms related to the subject of interest, instead of merely building a black box predictive model. For example, when dealing with results of gene expression measurements in context of cancer, identification of all genes which are related to cancer is necessary for complete understanding of the process, whereas a *minimal-optimal* set of genes might be more useful as genetic markers. A good discussion outlining why finding all relevant attributes is important is given by Nilsson *et al.* (2007).

The *all-relevant problem* of feature selection is more difficult than usual *minimal-optimal* one. One reason is that we cannot rely on the classification accuracy as the criterion for selecting the feature as important (or rejecting it as unimportant). The degradation of the classification accuracy, upon removal of the feature from the feature set, is sufficient to declare the feature important, but lack of this effect is not sufficient to declare it unimportant. One therefore needs another criterion for declaring variables important or unimportant. Moreover, one cannot use filtering methods, because the lack of direct correlation between a given feature and the decision is not a proof that this feature is not important in conjunction with the other features (Guyon and Elisseeff 2003). One is therefore restricted to wrapper algorithms, which are computationally more demanding than filters.

In a wrapper method the classifier is used as a black box returning a feature ranking, therefore one can use any classifier which can provide the ranking of features. For practical reasons, a classifier used in this problem should be both computationally efficient and simple, possibly without user defined parameters.

The current paper presents an implementation of the algorithm for finding all relevant features in the information system in a R (R Development Core Team 2010) package **Boruta** (available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=Boruta`). The algorithm uses a wrapper approach built around a random forest (Breiman 2001) classifier (Boruta is a god of the forest in the Slavic mythology). The algorithm is an extension of the idea introduced by Stoppiglia, Dreyfus, Dubois, and Oussar (2003) to determine relevance by comparing the relevance of the real features to that of the random probes. Originally this idea was proposed in the context of filtering, whereas here it is used in the wrapper algorithm. In the remaining sections of this article firstly a short description of the algorithm is given, followed by the examples of its application on a real-world and artificial data set.

2. Boruta algorithm

Boruta algorithm is a wrapper built around the random forest classification algorithm implemented in the R package **randomForest** (Liaw and Wiener 2002). The random forest classification algorithm is relatively quick, can usually be run without tuning of parameters and it gives a numerical estimate of the feature importance. It is an ensemble method in which classification is performed by voting of multiple unbiased weak classifiers — decision trees. These trees are independently developed on different bagging samples of the training set. The importance measure of an attribute is obtained as the loss of accuracy of classification caused by the random permutation of attribute values between objects. It is computed separately for all trees in the forest which use a given attribute for classification. Then the average and standard deviation of the accuracy loss are computed. Alternatively, the Z score

computed by dividing the average loss by its standard deviation can be used as the importance measure. Unfortunately the Z score is not directly related to the statistical significance of the feature importance returned by the random forest algorithm, since its distribution is not $N(0, 1)$ (Rudnicki, Kierczak, Koronacki, and Komorowski 2006). Nevertheless, in Boruta we use Z score as the importance measure since it takes into account the fluctuations of the mean accuracy loss among trees in the forest.

Since we cannot use Z score directly to measure importance, we need some external reference to decide whether the importance of any given attribute is significant, that is, whether it is discernible from importance which may arise from random fluctuations. To this end we have extended the information system with attributes that are random by design. For each attribute we create a corresponding 'shadow' attribute, whose values are obtained by shuffling values of the original attribute across objects. We then perform a classification using all attributes of this extended system and compute the importance of all attributes.

The importance of a shadow attribute can be nonzero only due to random fluctuations. Thus the set of importances of shadow attributes is used as a reference for deciding which attributes are truly important.

The importance measure itself varies due to stochasticity of the random forest classifier. Additionally it is sensitive to the presence of non important attributes in the information system (also the shadow ones). Moreover it is dependent on the particular realization of shadow attributes. Therefore we need to repeat the re-shuffling procedure to obtain statistically valid results.

In short, Boruta is based on the same idea which forms the foundation of the random forest classifier, namely, that by adding randomness to the system and collecting results from the ensemble of randomized samples one can reduce the misleading impact of random fluctuations and correlations. Here, this extra randomness shall provide us with a clearer view of which attributes are really important.

The Boruta algorithm consists of following steps:

1. Extend the information system by adding copies of all variables (the information system is always extended by at least 5 shadow attributes, even if the number of attributes in the original set is lower than 5).

2. Shuffle the added attributes to remove their correlations with the response.

3. Run a random forest classifier on the extended information system and gather the Z scores computed.

4. Find the maximum Z score among shadow attributes (MZSA), and then assign a hit to every attribute that scored better than MZSA.

5. For each attribute with undetermined importance perform a two-sided test of equality with the MZSA.

6. Deem the attributes which have importance significantly lower than MZSA as 'unimportant' and permanently remove them from the information system.

7. Deem the attributes which have importance significantly higher than MZSA as 'important'.

8. Remove all shadow attributes.

9. Repeat the procedure until the importance is assigned for all the attributes, or the algorithm has reached the previously set limit of the random forest runs.

In practice this algorithm is preceded with three start-up rounds, with less restrictive importance criteria. The startup rounds are introduced to cope with high fluctuations of Z scores when the number of attributes is large at the beginning of the procedure. During these initial rounds, attributes are compared respectively to the fifth, third and second best shadow attribute; the test for rejection is performed only at the end of each initial round, while the test for confirmation is not performed at all.

The time complexity of the procedure described above in realistic cases is approximately $O(P \cdot N)$, where P and N are respectively the numbers of attributes and objects. That may be time consuming for large data sets; still, this effort is essential to produce a statistically significant selection of relevant features.

To illustrate the scaling properties of Boruta algorithm we performed following experiment using Madalon data set. It is an artificial data set, which was one of the NIPS2003 problems. (Guyon, Gunn, Ben-Hur, and Dror 2005) The data set contains 2000 objects described with 500 attributes. We generated subsamples of Madelon set containing 250, 500, 750, ..., 2000 objects. Then for each subsample we created seven extended sets containing respectively 500, 1000, ..., 3500 superficial attributes obtained as a uniform random noise. Then we performed standard feature selection with Boruta on each of 64 test sets and measured the execution time. The results of the experiment are displayed in Figure 1. One may see almost perfect linear scaling for the increasing number of attributes. On the other hand execution times grow faster than the number of objects, but the difference is not very big and it seems to converge to linear scaling for large number of objects.

The timings are reported in CPU hours. Using the values from the largest data set, one can estimate the time required to complete Boruta run on a single core of modern CPU to be one hour per one million (attribute $\times$ objects).

One should notice that in many cases, in particular for a biomedical problems, the computation time is a small fraction of the time required to collect the data. One should also note, that the prime reason for running the 'all-relevant' feature selection algorithm is not the reduction of computation time (altough it can be achieved if the data set pruned from non-informative attributes will be subsequently analysed numerous times). The main reason is to find all attributes for which their correlation with decision is higher than that of the random attributes. Moreover, while Boruta is generally a sequential algorithm, the underlying random forest classifier is a trivially parallel task and thus Boruta can be distributed even over a hundreds of cores, provided that a parallel version of the random forest algorithm is used.

3. Using the Boruta package

The Boruta algorithm is implemented in **Boruta** package.

```
R> library("Boruta")
```

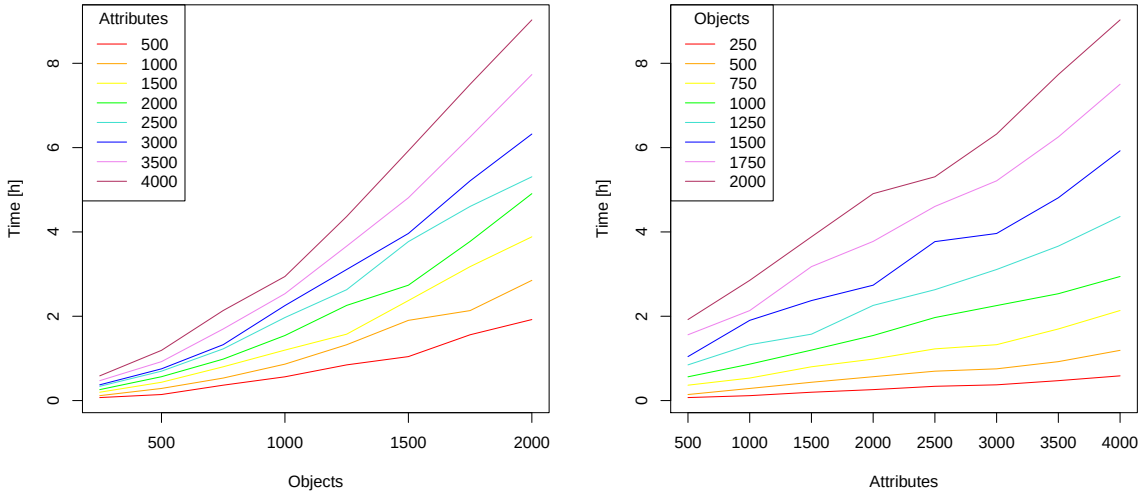

Figure 1: The scaling properties of Boruta with respect to the number of attributes (left) and number of objects (right). Each line on the left panel corresponds to the set with identical number of objects and on the right panel it corresponds to the set with identical number of attributes. One may notice that scaling is linear with respect to number of attributes and not far from linear with respect to the number of objects.

The ozone data from UCI Machine Learning Repository (Asuncion and Newman 2007) and available in **mlbench** package (Leisch and Dimitriadou 2010) is used as the first example:

```
R> library("mlbench")
R> data("Ozone")
R> Ozone <- na.omit(Ozone)
```

The algorithm is performed by the `Boruta` function. For its arguments, one should specify the model, either using a formula or predictor data frame with a response vector; the confidence level (which is recommended to be left default) and the maximal number of random forest runs.

One can also provide values of `mtry` and `ntree` parameters, which will be passed to `randomForest` function. Normally default `randomForest` parameters are used, they will be sufficient in most cases since random forest performance has rather a weak dependence on its parameters. If it is not the case, one should try to find `mtry` and `ntree` for which random forest classifier achieves convergence at minimal value of the OOB error.

Setting `doTrace` argument to 1 or 2 makes `Boruta` report the progress of the process; version 2 is a little more verbose, namely it shows attribute decisions as soon as they are cleared.

```
R> set.seed(1)
R> Boruta.Ozone <- Boruta(V4 ~ ., data = Ozone, doTrace = 2, ntree = 500)
```

```
Initial round 1: ..........
 1  attributes rejected after this test:  V2

Initial round 2: ..........
 1  attributes rejected after this test:  V3

Initial round 3: ..........
Final round: ..........
 8  attributes confirmed after this test:  V1 V5 V7 V8 V9 V10 V11 V12
....
 1  attributes confirmed after this test:  V13
....
 1  attributes rejected after this test:  V6
```

```
R> Boruta.Ozone

Boruta performed 48 randomForest runs in 2.540633 mins.
         9  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12 V13
         3  attributes confirmed unimportant: V2 V3 V6
```

The Ozone set consists of 12 attributes; three of them are rejected, two after the initial round 2, and one during the final round. The remaining attributes are indicated as confirmed. Figure 2 shows the Z scores variability among attributes during the Boruta run. It can be easily generated using the plot method of `Boruta` object:

```
R> plot(Boruta.Ozone)
```

One can see that Z score of the most important shadow attribute clearly separates important and non important attributes.

Moreover, it is clearly evident that attributes which consistently receive high importance scores in the individual random forest runs are selected as important. On the other hand, one can observe quite sizeable variability of individual scores. The highest score of a random attribute in a single run is higher than the highest importance score of two important attributes, and the lowest importance score of five important attributes. It clearly shows that the results of Boruta are generally more stable than those produced by feature selection methods based on a single random forest run, and this is why several iterations are required.

Due to the fact that the number of random forest runs during Boruta is limited by the `maxRuns` argument, the calculation can be forced to stop prematurely, when there are still attributes which are judged neither to be confirmed nor rejected — and thus finally marked tentative. For instance[1]:

```
R> set.seed(1)
R> Boruta.Short <- Boruta(V4 ~ ., data = Ozone, maxRuns = 12)
```

[1]The number of steps and the seed were intentionally selected to show this effect in the familiar data set. Due to slight differences between Windows and Linux versions of **randomForest** package, which probably arise due to compilation, the actual results of the procedure described above might differ slightly from the results shown here (these were obtained in R version 2.10.0 and **randomForest** version 4.5-33 on x86-64 Linux workstation).

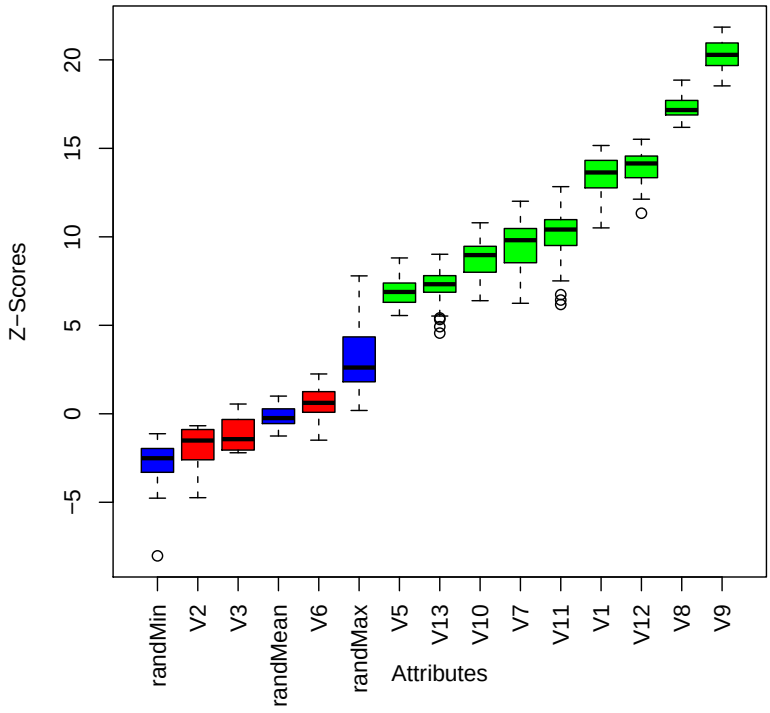

Figure 2: Boruta result plot for ozone data. Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute. Red and green boxplots represent Z scores of respectively rejected and confirmed attributes.

```
R> Boruta.Short

Boruta performed 42 randomForest runs in 2.3612 mins.
        8  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12
        2  attributes confirmed unimportant: V2 V3
        2  tentative attributes left: V6 V13
```

One should consider increasing the `maxRuns` parameter if tentative attributes are left. Nevertheless, there may be attributes with importance so close to MZSA that Boruta won't be able to make a decision with the desired confidence in realistic number of random forest runs. Therefore **Boruta** package contains a `TentativeRoughFix` function which can be used to fill missing decisions by simple comparison of the median attribute Z score with the median Z score of the most important shadow attribute:

```
R> TentativeRoughFix(Boruta.Short)

Boruta performed 42 randomForest runs in 2.3612 mins.
Tentatives roughfixed over 12 last randomForest runs.
        9  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12 V13
        3  attributes confirmed unimportant: V2 V3 V6
```

One can obviously treat such attributes manually.

For easy transfer of Boruta results to other classifiers and tools, the **Boruta** package contains functions that extract the results and convert them into a convenient form. The `getConfirmedFormula` and `getNonRejectedFormula` create a formula object that defines a model based respectively only on confirmed or on confirmed and tentative attributes:

```
R> getConfirmedFormula(Boruta.Ozone)
```

```
V4 ~ V1 + V5 + V7 + V8 + V9 + V10 + V11 + V12 + V13
```

The `attStats` function creates a data frame containing each attribute's Z score statistics and the fraction of random forest runs in which this attribute was more important than the most important shadow one:

```
R> attStats(Boruta.Ozone)
```

```
         meanZ    medianZ      minZ       maxZ  normHits  decision
V1  13.3911279 13.6373356 10.505555 15.1610346 1.0000000 Confirmed
V2  -2.0475252 -1.5112547 -4.741706 -0.6750894 0.0000000  Rejected
V3  -1.2097874 -1.4335204 -2.202290  0.5520193 0.0000000  Rejected
V5   6.9889240  6.8839769  5.552918  8.8074357 0.9166667 Confirmed
V6   0.5866514  0.6179196 -1.491181  2.2507610 0.1250000  Rejected
V7   9.4355872  9.8092537  6.244625 12.0112148 0.9791667 Confirmed
V8  17.3302697 17.1651707 16.186920 18.8550455 1.0000000 Confirmed
V9  20.3332547 20.2826539 18.530345 21.8499295 1.0000000 Confirmed
V10  8.7124127  8.9674981  6.391154 10.7939586 0.9791667 Confirmed
V11 10.0848916 10.4122110  6.179540 12.8348468 0.9583333 Confirmed
V12 13.9761395 14.1462836 11.335510 15.5130497 1.0000000 Confirmed
V13  7.1691008  7.3218887  4.561458  9.0149381 0.9166667 Confirmed
```

4. Example: Madelon data

Madelon is an artificial data set, which was one of the NIPS2003 problems. (Guyon *et al.* 2005) The data set contains 2000 objects corresponding to points located in 32 vertices of a 5-dimensional hypercube. Each vertex is randomly assigned one of two classes: -1 or $+1$, and the decision of each object is a class of its vertex. 500 attributes are constructed in the following way: 5 of them are randomly jittered coordinates of points; 15 others are random linear combinations of the first 5; finally the rest of the system is a uniform random noise. The task is to extract 20 important attributes from the system.

Madelon data is available from UCI Machine Learning Repository (Asuncion and Newman 2007) (loading of this data set may take several minutes):

```
R> root <-
+  "http://archive.ics.uci.edu/ml/machine-learning-databases/madelon/MADELON/"
R> predictors <- read.table(paste(root, "madelon_train.data", sep = ""))
R> decision <- read.table(paste(root, "madelon_train.labels", sep = ""))
R> Madelon <- data.frame(predictors, decision = factor(decision[, 1]))
```

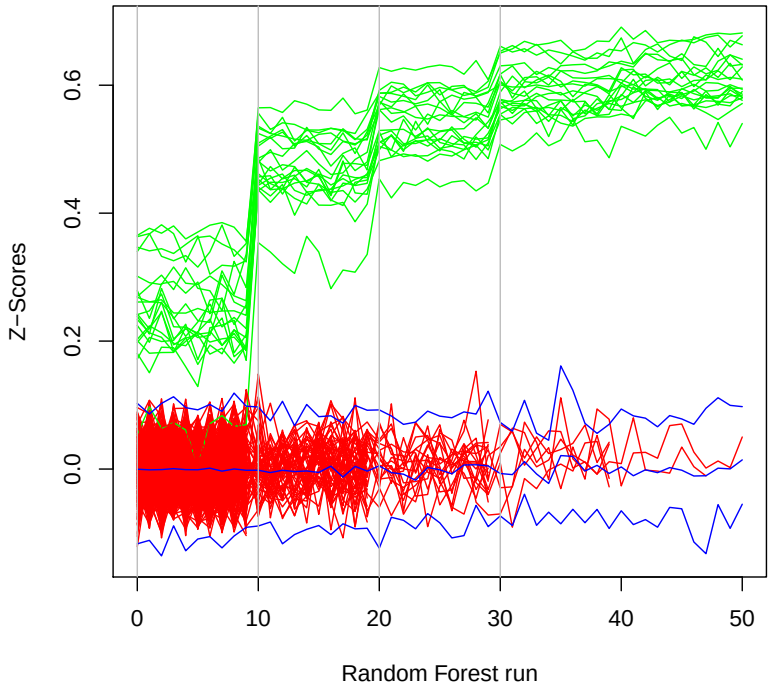

Figure 3: Z score evolution during Boruta run. Green lines correspond to confirmed attributes, red to rejected ones and blue to respectively minimal, average and maximal shadow attribute importance. Gray lines separate rounds.

Running Boruta (execution may take a few hours):

```
R> set.seed(7777)
R> Boruta.Madelon <- Boruta(decision ~ ., data = Madelon)

R> Boruta.Madelon

Boruta performed 51 randomForest runs in 1.861855 hours.
         20  attributes confirmed important: V29 V49 V65 V106 V129 V154 V242
V282 V319 V337 V339 V379 V434 V443 V452 V454 V456 V473 V476 V494
         480  attributes confirmed unimportant: V1 V2 V3 V4 V5 V6 V7 V8 V9
V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28
    (the rest of the output was omitted)
```

One can see that we have obtained 20 confirmed attributes. The `plotZScore` function visualizes the evolution of attributes' Z scores during a Boruta run:

```
R> plotZHistory(Boruta.Madelon)
```

The result can be seen on Figure 3. One may notice that consecutive removal of random noise increases the Z score of important attributes and improves their separation from the unimportant ones; one of them is even 'pulled' out of the group of unimportant attributes just after the first initial round. Also, on certain occasions, unimportant attributes may achieve a higher Z score than the most important shadow attribute, and this is the reason why we need multiple random forest runs to arrive at a statistically significant decision.

The reduction of attribute number is considerable (96%). One can expect that the increase of accuracy of a random forest classifier can be obtained on the reduced data set due to the elimination of noise.

It is known that feature selection procedure can introduce significant bias in resulting models. For example Ambroise and McLachlan (2002) have shown that, with the help of feature selection procedure, one can obtain a classifier, which is using only non-informative attributes and is 100% accurate on the training set. Obviously such classifier is useless and is returning random answers on the test set.

Therefore it is necessary to check whether Boruta is resistant to this type of error. It is achieved with the help of cross-validation procedure. The part of the data is set aside as a test set. Then the complete feature selection procedure is performed on the remaining data set – a training set. Finally the classifier obtained on the training set is used to classify objects from the test set to obtain classification error. The procedure is repeated several times, to obtain estimate of the variability of results.

Boruta performs several random forest runs to obtain statistically significant division between important and irrelevant attributes. One should expect that ranking obtained in the single RF run should be quite similar to that obtained from Boruta. We can check if this is the case, taking advantage of the cross-validation procedure described above.

Madelon data was split ten times into train and test sets containing respectively 90% and 10% of objects. Than, Boruta was run on each train set. Also, three random forest classifiers were generated on each train set: first using all attributes, the second one using only these attributes that were selected by Boruta, and the third one using the same number of attributes as found by Boruta, but selected as a top important by the first random forest trained on all attributes. Finally, the OOB error estimate on a train set and the error on a test set for all classifiers was collected.

The results are shown in the Table 1. One can see that both the OOB error as well as the error on the test set is consistently smaller for random forest runs performed on the reduced set of attributes. This observation is verified by a t test:

```
R> t.test(CV.Boruta$"Test conf.", CV.Boruta$"Test all", paired = TRUE)

        Paired t-test

data:  CV.Boruta$"Test conf." and CV.Boruta$"Test all"
t = -24.2727, df = 9, p-value = 1.636e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.198962 -0.165038
sample estimates:
```

	OOB all	OOB conf.	OOB RF	Test all	Test conf.	Test RF	Agreement
1	0.32	0.11	0.11	0.27	0.12	0.11	0.91
2	0.29	0.11	0.11	0.30	0.14	0.13	0.83
3	0.29	0.11	0.11	0.34	0.14	0.14	0.90
4	0.32	0.11	0.12	0.24	0.07	0.07	1.00
5	0.30	0.11	0.11	0.27	0.12	0.12	0.83
6	0.29	0.12	0.11	0.26	0.07	0.07	1.00
7	0.30	0.11	0.11	0.32	0.12	0.12	1.00
8	0.30	0.12	0.11	0.28	0.08	0.08	1.00
9	0.30	0.11	0.11	0.32	0.10	0.12	0.91
10	0.30	0.11	0.11	0.28	0.08	0.10	1.00

Table 1: Cross-validation of the error reduction due to limiting the information system to attributes claimed confirmed by Boruta.

```
mean of the differences
                -0.182
```

As one may expect, the feature ranking provided by plain random forest agrees fairly well with Boruta results. This explains why the simple heuristic feature selection procedure in random forest, namely selecting a dozen or so top scoring attributes, works well for obtaining good classification results. Nevertheless, this will not necessarily be a case when dealing with larger and more complex sets, where stochastic effects increase the variability of the random forest importance measure and thus destabilize the feature ranking.

One should note that the Boruta is a heuristic procedure designed to find all relevant attributes, including weakly relevant attributes. Following Nilsson *et al.* (2007), we say that attribute is weakly important when one can find a subset of attributes among which this attribute is not redundant. The heuristic used in Boruta implies that the attributes which are significantly correlated with the decision variables are relevant, and the significance here means that correlation is higher than that of the randomly generated attributes. Obviously the set of all relevant attributes may contain highly correlated but still redundant variables. Also, the correlation of the attribute with the decision does not imply causative relation; it may arise when both decision attribute and descriptive attribute are independently correlated with some other variable. An illustrative example of such situation was given by Strobl, Hothorn, and Zeileis (2009). Users interested in finding a set of highly relevant and uncorrelated attributes within the result returned by Boruta may use for example package **party** (Strobl *et al.* 2009), **caret** (Kuhn 2008; Kuhn, Wing, Weston, Williams, Keefer, and Engelhardt 2010), **varSelRF** (Diaz-Uriarte 2007, 2010) or **FSelector** (Romanski 2009) for further refinement.

5. Summary

We have developed Boruta, a novel random forest based feature selection method, which provides unbiased and stable selection of important and non-important attributes from an information system. Due to the iterative construction, our method can deal both with the

fluctuating nature of a random forest importance measure and the interactions between attributes. We have also demonstrated its usefulness on an artificial data set. The method is available as an R package.

Acknowledgments

Computations were performed at ICM, grant G34-5. We would like to thank the reviewers and the technical editor for a helpful discussions which led to improvement of the paper.

References

Ambroise C, McLachlan GJ (2002). "Selection Bias in Gene Extraction on the Basis of Microarray Gene-Expression Data." *Proceedings of the National Academy of Sciences of the United States of America*, **99**(10), 6562–6.

Asuncion A, Newman DJ (2007). "UCI Repository of Machine Learning Databases." University of California, Irvine, Deptartment of Information and Computer Sciences, URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**, 5–32.

Diaz-Uriarte R (2007). "**GeneSrF** and **varSelRF**: A Web-Based Tool and R Package for Gene Selection and Classification Using Random Forest." *BMC Bioinformatics*, **8**(328).

Diaz-Uriarte R (2010). ***varSelRF**: Variable Selection Using Random Forests*. R package version 0.7-2, URL http://CRAN.R-project.org/package=varSelRF.

Guyon I, Elisseeff A (2003). "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research*, **3**, 1157–1182.

Guyon I, Gunn S, Ben-Hur A, Dror G (2005). "Result Analysis of the NIPS 2003 Feature Selection Challenge." *Advances in Neural Information Processing Systems*, **17**, 545–552.

Kohavi R, John GH (1997). "Wrappers for Feature Subset Selection." *Artificial Intelligence*, **97**, 273–324.

Kuhn M (2008). "Building Predictive Models in R Using the **caret** Package." *Journal of Statistical Software*, **28**(5), 1–26. URL http://www.jstatsoft.org/v28/i05/.

Kuhn M, Wing J, Weston S, Williams A, Keefer C, Engelhardt A (2010). ***caret**: Classification and Regression Training*. R package version 4.58, URL http://CRAN.R-project.org/package=caret.

Leisch F, Dimitriadou E (2010). ***mlbench**: Machine Learning Benchmark Problems*. R package version 2.0-0, URL http://CRAN.R-project.org/package=mlbench.

Liaw A, Wiener M (2002). "Classification and Regression by **randomForest**." *R News*, **2**(3), 18–22. URL http://CRAN.R-project.org/doc/Rnews/.

Nilsson R, Peña J, Björkegren J, Tegnér J (2007). "Consistent Feature Selection for Pattern Recognition in Polynomial Time." *The Journal of Machine Learning Research*, **8**, 612.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Romanski P (2009). ***FSelector**: Selecting Attributes*. R package version 0.18, URL http://CRAN.R-project.org/package=FSelector.

Rudnicki WR, Kierczak M, Koronacki J, Komorowski J (2006). "A Statistical Method for Determining Importance of Variables in an Information System." In S Greco, H Y, S Hirano, M Inuiguchi, S Miyamoto, HS Nguyen, R Slowinski (eds.), *Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006, Kobe, Japan, November 6–8, 2006, Proceedings*, volume 4259 of *Lecture Notes in Computer Science*, pp. 557–566. Springer-Verlag, New York.

Stoppiglia H, Dreyfus G, Dubois R, Oussar Y (2003). "Ranking a Random Feature for Variable and Feature Selection." *Journal of Machine Learning Research*, **3**, 1399–1414.

Strobl C, Hothorn T, Zeileis A (2009). "Party on! – A New, Conditional Variable Importance Measure for Random Forests Available in the **party** Package." *The R Journal*, **1**(2), 14–17. URL http://journal.R-project.org/archive/2009-2/RJournal_2009-2_Strobl~et~al.pdf.

Affiliation:

Miron B. Kursa, Witold R. Rudnicki
Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw
Pawinskiego 5A
02-105 Warsaw, Poland
E-mail: M.Kursa@icm.edu.pl, W.Rudnicki@icm.edu.pl
URL: http://www.icm.edu.pl/~rudnicki/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 36, Issue 11 *Submitted:* 2009-12-03
September 2010 *Accepted:* 2010-07-29