Variable importances for model-based trees

Marjolein Fokkema

9-9-2022

In general, I think three approaches are possible to compute importances:

1) Importances based on improvement in likelihood (i.e., an impurity-based measure). This approach is implemented for semforests in the semtree package, and easy to do for all types of model-based trees, see below.

2) Importances based on predictive accuracy (i.e., permutation-based importances). This is relatively easy to compute for GLM trees / supervised trees, because can simply compute the loss before and after permutation of the predictor variable of interest, and can also use OOB observations for this. This approach is implemented in the mobForest package.

For non-supervised trees, like raschtree and also semtrees, I believe this is more tricky, but not impossible. We simply have to choose what we take as the loss, and determinse how to compute it for test/OOB observations. For example, for raschtrees, we could compute a squared error loss: Take the observed pattern of item response for person i, and use those to compute the model-based probabilities of having each of the items correct for person i. Take the squared difference, and sum over persons and items. Next, permute the values of the predictor variable of interest, recompute model-based probabilities of having each of the items correct, take squared difference and sum over persons and items. The difference between the two sums thus obtained quantifies the importance of the predictor variable of interest; it quantifies how much the predictor variable of interest contributes to explaining the item responses.

Package psychotree has a dependence on package mirt. Using function `mirt::fscores`, theta's can be computed from a fitted mirt model by specifying the `response.pattern` argument. Next, using function `expected.item` (and extracting item objects from the fitted model using function `extract.item`), expected item responses can be computed for every item, given the computed theta values. For the rasch model, `fscores` returns probabilities of answering correctly.

3) Importances based on estimated coefficients or effect-sizes in the terminal nodess (i.e., most similar to PRE importances).

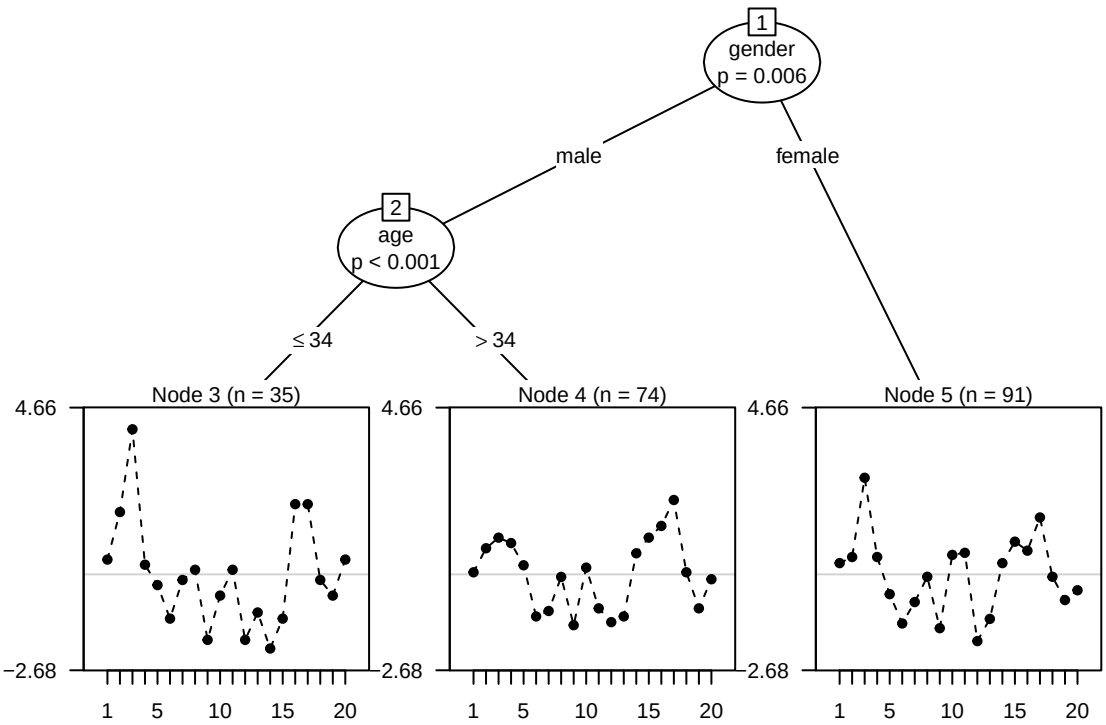
Importances based on improvement in likelihood for model-based trees

```
##
## Function that takes any model-based tree as input, and returns the variable
## importances for all partitioning variables specified.
##
## The function checks for every split (inner) node that is not the root:
## How much the split improves the logLik
```

```
## Then adds this reduction to the varimp for that variable
##
varimp_mobtree <- function(object) {
  imps <- numeric(ncol(object[[1]]$node$info$test))
  names(imps) <- colnames(object[[1]]$node$info$test)
  for (i in 1:length(object)) {
    if (!is.null(object[[i]]$node$split)) {
      ## Sum logLik of the two daughter nodes, substract logLik of the mother node
      LL_redux <-  (logLik(object[[i]]$node$kids[[1]]$info$object) +
                      logLik(object[[i]]$node$kids[[2]]$info$object)) -
        logLik(object[[i]]$node$info$object)
      ## Add logLik reduction to current varimps
      which_var <-
        rownames(attr(terms(object), "factors"))[object[[i]]$node$split$varid]
      imps[which_var] <- imps[which_var] + LL_redux
    }
  }
  return(imps)
}
```

Illustrate use with a `raschtree`:

```
library("psychotree")
data("DIFSim", package = "psychotree")
rt <- raschtree(resp ~ age + gender + motivation, data = DIFSim)
plot(rt, gp = gpar(cex = .7))
```



```
## print logLik values of each node
for (i in 1:length(rt)) {
  print(paste("node", i, logLik(rt[[i]]$node$info$object)))
}
```

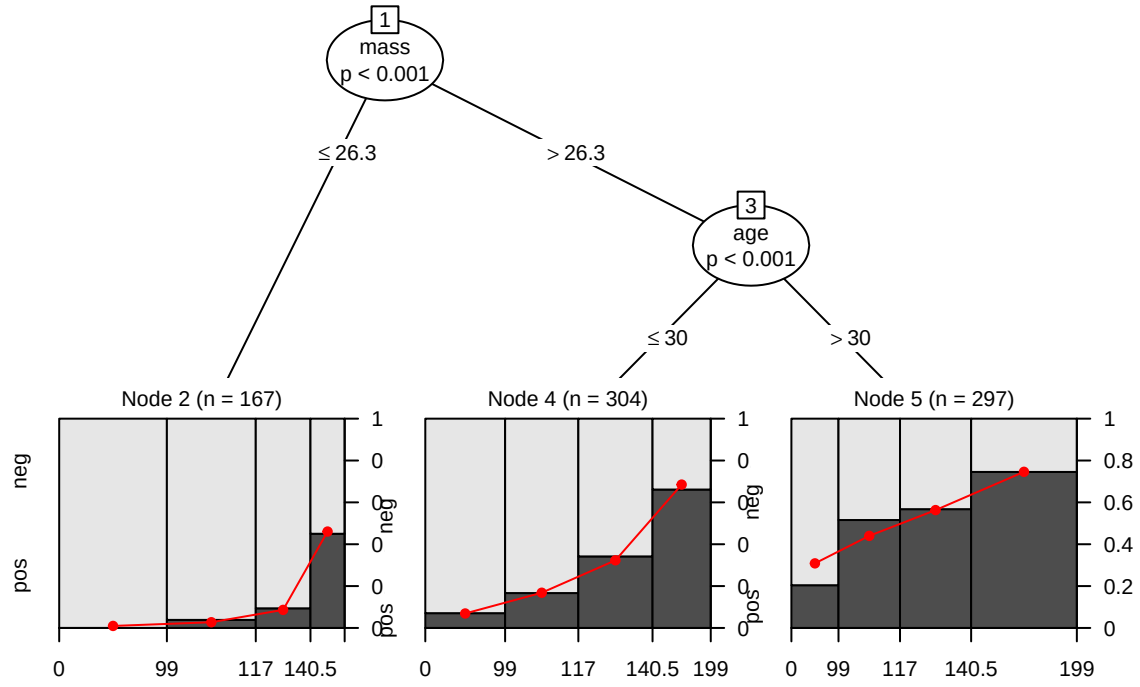
```
## [1] "node 1 -1764.32316282135"
## [1] "node 2 -967.225924957507"
## [1] "node 3 -273.247841215574"
## [1] "node 4 -649.935172289657"
## [1] "node 5 -773.730416739513"
```

```
varimp_mobtree(rt)
```

```
##        age     gender motivation
##   44.04291   23.36682    0.00000
```

Illustrate use with a `glmtree`:

```
data("PimaIndiansDiabetes", package = "mlbench")
gt <- glmtree(diabetes ~ glucose | pregnant + pressure + triceps + insulin +
                mass + pedigree + age, data = PimaIndiansDiabetes,
              family = binomial)
plot(gt, gp = gpar(cex = .7))
```




```
## print logLik values of each node
for (i in 1:length(gt)) {
  print(paste("node", i, logLik(gt[[i]]$node$info$object)))
}
```

```
## [1] "node 1 -404.359818675696"
## [1] "node 2 -30.2511982895619"
## [1] "node 3 -344.224956244593"
## [1] "node 4 -140.490542092715"
## [1] "node 5 -184.716103886093"
```

```
varimp_mobtree(gt)
```

```
## pregnant pressure  triceps  insulin     mass pedigree      age
##  0.00000  0.00000  0.00000  0.00000 29.88366  0.00000 19.01831
```

How does semforest do this?

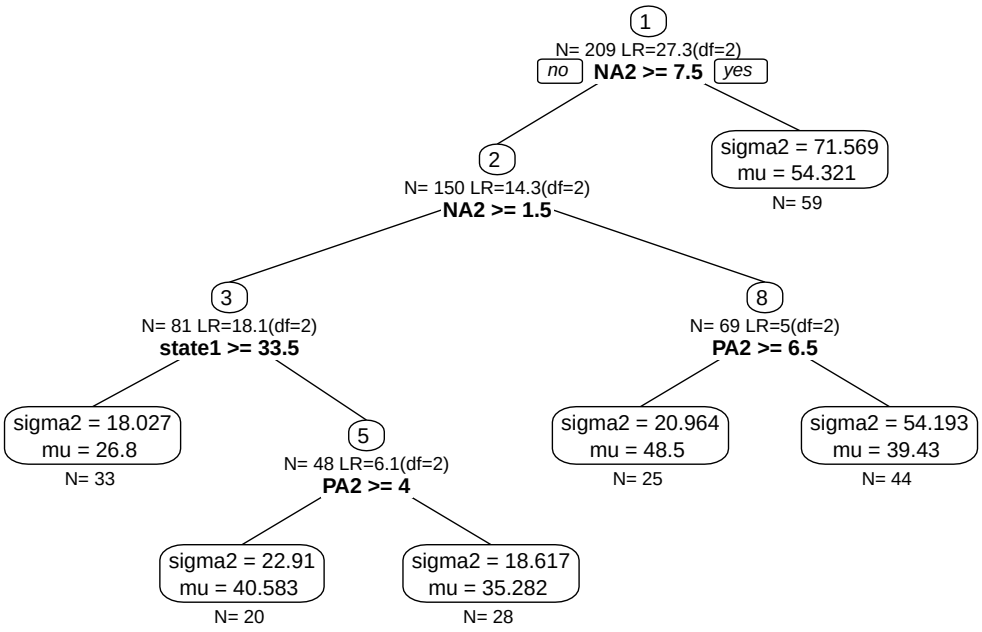
According to `?semtree::varimp`: The value of the -2LL of the leaf nodes is compared to baseline overall model. Not sure how one gets an importance for a variable in this way. Let's fit a single-tree semforest, plot the resulting tree and compute importances:

```
library("semtree")
library("psychTools")
data(affect)
affect$Film <- as.factor(affect$Film)
affect$lie <- as.ordered(affect$lie)
affect$imp <- as.ordered(affect$imp)

library("OpenMx")
manifests <- c("state2")
latents <- c()
model <- mxModel("Univariate Normal Model",
                 type="RAM",
                 manifestVars = manifests,
                 latentVars = latents,
                 mxPath(from="one",to=manifests, free=c(TRUE),
                        value=c(50.0) , arrows=1, label=c("mu") ),
                 mxPath(from=manifests,to=manifests, free=c(TRUE),
                        value=c(100.0) , arrows=2, label=c("sigma2") ),
                 mxData(affect, type = "raw")
)
control <- semforest.control(num.trees = 1)
forest <- semforest(model=model,
                    data = affect,
                    control = control,
                    covariates = c("Study","Film", "state1", "PA2","NA2","TA2"))
```

```
## > Model was not run. Estimating parameters now before running the forest.
## <U+2714> Tree construction finished [took 8s].
## <U+2714> Forest completed [took 9s]
```

```
vim <- semtree::varimp(forest)
plot(forest$forest[[1]], cex = .7)
```




```
print(vim, sort.values=TRUE)
```

```
## Variable Importance
##     Study      Film       TA2    state1       PA2       NA2
##   0.00000   0.00000   0.00000  15.82088  21.88670 140.51211
```

How does mobforest do this?

mobForest returns OOB permutation importances (which I guess assumes a supervised learning problem). It has a function `varimp.output`, which according to the documentation returns: "Variable importance matrix containing the decrease in predictive accuracy after permuting the variables across all trees."

"Values of variable 'm' in the oob cases are randomly permuted and R2 obtained through variable-m-permuted oob data is subtracted from R2 obtained on untouched oob data. The average of this number over all the trees in the forest is the raw importance score for variable m."