Practical Machine Learning in R

Input Processing

Lars Kotthoff\textsuperscript{1,2}
larasko@uwyo.edu

\textsuperscript{1}with slides from Bernd Bischl and Michel Lang
\textsuperscript{2}slides available at http://www.cs.uwyo.edu/~larasko/ml-fac
Why do we care?

▷ real-world data often messy
▷ missing values
▷ uninformative features
▷ outliers
▷ a lot of time is spent cleaning up data
Replacing Missing Values

▷ many learners cannot handle missing values at all ("missings" property)
▷ numeric features: impute with constant, mean, median, mode, min, max, random value of estimated normal/uniform/ECDF
▷ factor features: impute with constant, mode, random value of ECDF

data = data.frame(x = c(1, NA, 3, 4), y = c("a", "b", NA, "b"))
imputed = impute(data, cols = list(x = imputeMean(), y = imputeMode()))
imputed$data

#> x | y
#> ---|---
#> 1 | a
#> 2 | b
#> 3 | b
#> 4 | b
Replacing Missing Values

▷ can specify imputation depending on data type
▷ can add new columns to indicate what was imputed

```r
imputed <- impute(data,
                  classes = list(numeric = imputeMean(), factor = imputeMode()),
                  dummy.classes = c("numeric", "factor"))
imputed$data

## x y x.dummy y.dummy
## 1 1.000000 a FALSE FALSE
## 2 2.666667 b TRUE FALSE
## 3 3.000000 b FALSE TRUE
## 4 4.000000 b FALSE FALSE
```
Wrapped Learners

- wrap learner in imputation method
- imputation happens transparently – wrapped learner can be used just like original learner
- test data is imputed like training data (e.g. with same mode value)

```r
learner = makeImputeWrapper("classif.rpart",
    classes = list(numeric = imputeMedian()))
```

```r
definition
## Learner classif.rpart.imputed from package rpart
## Type: classif
## Name: ; Short name: 
## Class: ImputeWrapper
## Properties: twoclass,multiclass,missings,numerics,factors,ordered,prob,weights,featimp
## Predict-Type: response
## Hyperparameters: xval=0
```
Constant Features

- most models technically break in presence of constant features
- constant features hold no information

```r
data = data.frame(target = c("a", "a", "b"), x = 1:3, y = c(1, 1, 1))
task = makeClassifTask(data = data, target = "target")

# remove all features where fraction of values differing
# from mode value is <= 5%
getTaskData(removeConstantFeatures(task, perc = 0.05))

## Removing 1 columns: y

## target  x
## 1 a 1
## 2 a 2
## 3 b 3
```
Merging of Factor Levels

▷ For the target variable:

```r
task = joinClassLevels(iris.task, new.levels = list(new = c("setosa", "virginica")))
table(getTaskTargets(task))
```

```
## new versicolor
##     100     50
```

▷ For the features:

```r
mergeSmallFactorLevels(task, min.perc = 0.01)
```
Unbalanced Classes

- one class much more/less frequent than others
- always predicting largest class gives good accuracy
- never predicting smallest class loses little accuracy
- solution: under- and oversample (for binary tasks)
task = `makeClassifTask`(id = "test",
   data = `data.frame`(a = `sample`(1:100, 100),
                      class = `factor`(c(`rep"a", 98), ”b”, ”b”)),
   target = "class")

```
task.under = `undersample`(task, rate = 0.5)
table(`getTaskTargets`(task.under))
```

```r
##
## a  b
## 49 2
```

```
task.over = `oversample`(task, rate = 100, cl = "b")
table(`getTaskTargets`(task.over))
```

```r
##
## a  b
## 98 200
```
Unbalanced Classes

- can under-/oversample in wrappers
- can wrap wrappers

```r
learner = makeUndersampleWrapper("classif.rpart", usw.rate = 0.5)
learner = makeImputeWrapper(learner,
    classes = list(numeric = imputeMedian()))
learner

## Learner classif.rpart.undersampled.imputed from package mlr,rpart
## Type: classif
## Name: ; Short name:
## Class: ImputeWrapper
## Properties: numerics,factors,ordered,missings,weights,prob,twoclass,multiclass,featimp
## Predict-Type: response
## Hyperparameters: xval=0,usw.rate=0.5
```
Exercises

http://www.cs.uwyo.edu/~larsko/ml-fac/05-input-exercises.Rmd