mlrHyperopt: Effortless and collaborative hyperparameter optimization experiments

Jakob Richter
July 5, 2017
Faculty of Statistics, TU Dortmund University
1. Motivation for caret\textsuperscript{1} Users
2. Motivation for mlr\textsuperscript{2} Users
3. Website and API
4. Parameter Tuning
5. Lessons learned

\textsuperscript{1}https://topepo.github.io/caret
\textsuperscript{2}https://mlr-org.github.io/mlr
Motivation
**caret** automatically performs a grid search for all learners.

```r
library(caret)

system.time({
  m.c = train(iris[,1:4], iris[,5], method = "rf")
})
## user  system elapsed
##   4.533   0.016   4.552

system.time({
  m.r = randomForest(iris[,1:4], iris$Species)
})
## user  system elapsed
##   0.025   0.000   0.026
```

How to find out what is going on?

```r
m.c$results
```

<table>
<thead>
<tr>
<th>mtry</th>
<th>Accuracy</th>
<th>Kappa</th>
<th>AccuracySD</th>
<th>KappaSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9485003</td>
<td>0.9218204</td>
<td>0.02473556</td>
<td>0.03739386</td>
</tr>
<tr>
<td>2</td>
<td>0.9490167</td>
<td>0.9226138</td>
<td>0.02537238</td>
<td>0.03837125</td>
</tr>
<tr>
<td>3</td>
<td>0.9499133</td>
<td>0.9239744</td>
<td>0.02897600</td>
<td>0.04377608</td>
</tr>
</tbody>
</table>
Can I find out in advance which parameters will be tuned?

\texttt{modelLookup("rf")} gives some information.

\begin{verbatim}
modelLookup("rf")
## model parameter label forReg forClass probModel
## 1 rf mtry #Randomly Selected Predictors TRUE TRUE TRUE
\end{verbatim}
Can I find out in advance which parameters will be tuned?

http://github.com/topepo/caret/blob/master/models/files
reveals all details.

Jakob Richter (TU Dortmund)
Extract from `models/files/gbm.R`:

```r
out <- expand.grid(
    interaction.depth = seq(1, len),  #<- parameter range depends on tuning budget
    n.trees = floor((1:len) * 50),  #<- ..
    shrinkage = .1,
    n.minobsinnode = 10)

# ...
# Random Search
out <- data.frame(
    n.trees = floor(runif(len, min = 1, max = 5000)),
    interaction.depth = sample(1:10, replace = TRUE, size = len),
    shrinkage = runif(len, min = .001, max = .6),
    n.minobsinnode = sample(5:25, replace = TRUE, size = len))
out <- out[!duplicated(out),]
```
**mlr** provides parameter definitions for all learners.

```r
library(mlr)
lrn = makeLearner("classif.randomForest")
filterParams(getParamSet(lrn), tunable = TRUE)
```

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
<th>len</th>
<th>Def</th>
<th>Constr</th>
<th>Req</th>
<th>Tunable</th>
<th>Trafo</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntree</td>
<td>integer</td>
<td>-</td>
<td>500</td>
<td>1 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>mtry</td>
<td>integer</td>
<td>-</td>
<td>-</td>
<td>1 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>replace</td>
<td>logical</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>classwt</td>
<td>numericvector</td>
<td>&lt;NA&gt;</td>
<td>-</td>
<td>0 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>cutoff</td>
<td>numericvector</td>
<td>&lt;NA&gt;</td>
<td>-</td>
<td>0 to 1</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>sampsize</td>
<td>integervector</td>
<td>&lt;NA&gt;</td>
<td>-</td>
<td>1 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>nodesize</td>
<td>integer</td>
<td>-</td>
<td>1</td>
<td>1 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>maxnodes</td>
<td>integer</td>
<td>-</td>
<td>-</td>
<td>1 to Inf</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>importance</td>
<td>logical</td>
<td>-</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>localImp</td>
<td>logical</td>
<td>-</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
<td>TRUE</td>
<td>-</td>
</tr>
</tbody>
</table>

But **ParamSets** are unconstrained and include possibly unimportant parameters.
Necessary to define own **ParamSets** for tuning:

```r
ps = makeParamSet(
    makeIntegerParam("mtry", lower = 1, upper = 4),
    makeIntegerParam("nodesize", lower = 1, upper = 10)
)
tuneParams(lrn, iris.task, cv10, measures = acc,
    par.set = ps, makeTuneControlGrid(resolution = 3L))
```

## Tune result:
## Op. pars: mtry=1; nodesize=6
## acc.test.mean=0.953
Deviate from the defaults in `caret`:

```r
grid = expand.grid(mtry = 2:4, nodesize = c(1,5,10))
```

```r
m = caret::train(iris[,1:4], iris[,5],
    method = "rf", tuneGrid = grid)
```

```r
## Error: The tuning parameter grid should have columns mtry
```

It seems you have to write you own custom method\(^3\).

\(^3\)https://stackoverflow.com/questions/38625493/tuning-two-parameters-for-random-forest-in-caret-package
### mlr vs. caret

**In caret...**

+ Tuning is the default.
+ Tuning with defaults is easy.
- Deviating from defaults is a hassle and needs expert knowledge.

**In mlr...**

+ Train works like the default of the package.
- Tuning needs expert knowledge.
+ Deviating from defaults is easy.

To solve this problem in mlr we want to share the expert knowledge with...
mlrHyperopt
mldrHyperopt enables access to a web database of Parameter Configurations for many machine learning methods in R.

Why an online database?

- Defaults in packages will always be controversial.
- Knowledge changes over time but R packages have to maintain reproducibility.
- Defaults differ for different scenarios. (data set size etc.)
mlrHyperopt stores tuning parameters in ParConfig:

- Parameter Set of tunable parameters
- fixed Parameter Values to overwrite defaults
- associated learner and note

Features of the Parameter Set:\footnote{https://github.com/berndbischl/ParamHelpers}:

- Parameter values can be: real-valued, integer, discrete, logical, ...
- Parameters can have:
  - transformations (to account non-uniform distribution of interesting regions)
  - requirements on other parameters (to represent hierarchical structures)
- Bounds and defaults can depend on the task size, number of features, etc.
Web Interface

Overview of all ParConfigs uploaded to
http://mlrhyperopt.jakob-r.de/parconfigs

Jakob Richter (TU Dortmund)
Tune the parameters for the **ranger** Random Forest with **mlr**\(^5\).

```r
library(mlrHyperopt)

lrn = makeLearner("classif.ranger")
(pc = downloadParConfigs(learner.class = getLearnerClass(lrn)))

## Parameter Configuration
## Parameter Values: num.threads=1, verbose=FALSE, respect.unordered.factors=TRUE
## Associated Learner: classif.ranger
## Parameter Set:

# Type len  Def Constr Req Tunable Trafo
min.node.size integer - 1 1 to 10 - TRUE -
mtry integer - floor(sqrt(p)) 1 to p - TRUE -

ps = getParConfigParSet(pc[[1]])
ps = evaluateParamExpressions(ps, dict = getTaskDictionary(iris.task))

lrn = setHyperPars(lrn, par.vals = getParConfigParVals(pc[[1]]))


```
```r
tuneParams(lr, iris.task, resampling = cv10, par.set = ps,
          measures = acc, control = makeTuneControlRandom(maxit = 10))
```

## Tune result:
## Op. pars: min.node.size=3; mtry=1
## acc.test.mean=0.96

Dependent search space for the tuning of a support vector machine.

```r
def ps = makeParamSet(
def makeDiscreteParam("kernel", c("rbfdot", "polydot")),
def makeNumericParam("C", -5, 5, trafo = function(x) 2^x),
def makeNumericParam("sigma", lower = -10, upper = 10,
def trafo = function(x) 2^x, requires = quote(kernel == "rbfdot")),
def makeNumericParam("degree", lower = 1, upper = 5,
def requires = quote(kernel == "polydot"))

pc = makeParConfig(ps, learner.name = "ksvm")
uploadParConfig(pc)
## [1] "23"
```

Jakob Richter (TU Dortmund)
With the following **ParamHelpers** functions we can generate grids for **caret**

- `generateGridDesign`
- `generateRandomDesign`
- `generateDesign` (Latin Hypercube Sample)
- `generateDesignOfDefaults` (to be used in combination)

```r
pc = downloadParConfigs(learner.name = "nnet")
grid = generateRandomDesign(n = 10L, par.set = pc[[1]]$par.set, 
    trafo = TRUE)
tr = caret::train(iris[,1:4], iris[,5], method = "nnet", 
    tuneGrid = grid, trace = FALSE)
tr$bestTune
## size   decay
## 8   14 0.4467496
```
Tuning with mlrHyperopt
A heuristic decides for tuning method:

**Tuning Methods:**

- **grid search:** 1 parameter, 2 mixed parameters
- **random search:** > 2 mixed parameters
- **Bayesian optimization with mlrMBO**: all parameters numeric

Default parameter sets from `mlrHyperopt` are used:

```r
(h.res = hyperopt(task = iris.task, learner = "classif.ksvm"))
```

## Tune result:

```
## Op. pars: C=101; sigma=0.0432
## mmce.test.mean=0.0267
```

```r
m = mlr::train(h.res$learner, iris.task)
```

---

6https://mlr-org.github.io/mlrMBO/
### Benchmark

#### OpenML Data Sets

<table>
<thead>
<tr>
<th>OpenML_ID</th>
<th>Name</th>
<th>p</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>mfeat-morphological</td>
<td>6</td>
<td>2000</td>
</tr>
<tr>
<td>3493</td>
<td>monks-problems-2</td>
<td>6</td>
<td>601</td>
</tr>
<tr>
<td>3510</td>
<td>JapaneseVowels</td>
<td>14</td>
<td>9961</td>
</tr>
<tr>
<td>3883</td>
<td>mfeat-karhunen</td>
<td>64</td>
<td>2000</td>
</tr>
<tr>
<td>3896</td>
<td>ada_agnostic</td>
<td>48</td>
<td>4562</td>
</tr>
<tr>
<td>3903</td>
<td>pc3</td>
<td>37</td>
<td>1563</td>
</tr>
<tr>
<td>9914</td>
<td>wilt</td>
<td>5</td>
<td>4839</td>
</tr>
<tr>
<td>9970</td>
<td>hill-valley</td>
<td>100</td>
<td>1212</td>
</tr>
<tr>
<td>34536</td>
<td>Internet-Advertisements</td>
<td>1558</td>
<td>3279</td>
</tr>
</tbody>
</table>

Algorithms: caret with grid and random search and mlrHyperopt. Each with a budget of 10 and 50 CV10-evaluations.

---

7https://www.openml.org/
All Results

Jakob Richter (TU Dortmund)
Performance with a **budget of 10 10CV-Evaluations**.

<table>
<thead>
<tr>
<th></th>
<th>caret grid</th>
<th>caret random</th>
<th>mlrHyperopt</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>caret grid</td>
<td>0.00</td>
<td>0.09</td>
<td>0.11</td>
<td>0.40</td>
</tr>
<tr>
<td>caret random</td>
<td>0.09</td>
<td>0.00</td>
<td>0.09</td>
<td>0.49</td>
</tr>
<tr>
<td>mlrHyperopt</td>
<td>0.14</td>
<td>0.12</td>
<td>0.00</td>
<td>0.47</td>
</tr>
<tr>
<td>default</td>
<td>0.09</td>
<td>0.02</td>
<td>0.04</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The table gives the fractions of instances where \( H_0 : \text{acc}_A \leq \text{acc}_B \) is rejected by the paired *Wilcoxon*-test to level \( \alpha = 0.05 \). A column, \( B \) rows.

*i.e.: mlrHyperopt* is significantly better than the default settings in 47% of the cases.
Performance: Dominance

Performance with a budget of 50 10CV-Evaluations.

<table>
<thead>
<tr>
<th></th>
<th>caret grid</th>
<th>caret random</th>
<th>mlrHyperopt</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>caret grid</td>
<td>0.00</td>
<td>0.09</td>
<td>0.21</td>
<td>0.30</td>
</tr>
<tr>
<td>caret random</td>
<td>0.40</td>
<td>0.00</td>
<td>0.12</td>
<td>0.54</td>
</tr>
<tr>
<td>mlrHyperopt</td>
<td>0.40</td>
<td>0.16</td>
<td>0.00</td>
<td>0.53</td>
</tr>
<tr>
<td>default</td>
<td>0.33</td>
<td>0.02</td>
<td>0.05</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The table gives the fractions of instances where $H_0 : acc_A \leq acc_B$ is rejected by the paired Wilcoxon-test to level $\alpha = 0.05$. A column, $B$ rows.

i.e.: mlrHyperopt is significantly better than the default settings in 53% of the cases.
Which Learner Tuner Combination is a Good Choice?

Rankings of averaged performances of each combination on each dataset.

Jakob Richter (TU Dortmund)
Lessons Learned
Lessons Learned

- Parameter Tuning is only beneficial on some data and for some methods.
- `caret`’s grid search has performance problems on big data sets (ksvm, nnet).
- `caret`’s grid search sub model trick is beneficial (glmnet).
- The benchmark indicates that *random search* is better than the grid search.
## Benefits

- Transparent and reproducible benchmarks in combination with OpenML:
  - *e.g.* Tune ml method A on parameter space with id 123 on OpenML Task 456.

## Outlook

- Implement voting system / advanced statistics

### Find us on GitHub

- [github.com/jakob-r/mlrHyperopt](https://github.com/jakob-r/mlrHyperopt)
- [github.com/mlr-org/mlr](https://github.com/mlr-org/mlr)