Applied Machine Learning

MLR3 Tuning
Parameter Transformations & AutoTuner

Learning goals

- Parameter transformation techniques in mlr3tuning
- AutoTuner for automated hyperparameter optimization
- Nested resampling implementation in mlr3

Parameter Transformation

PARAMETER TRANSFORMATION

- Sometimes we do not want to optimize over an evenly spaced range
- $k = 1$ vs. $k = 2$ probably more interesting than $k = 101$ vs. $k = 102$

$\Rightarrow$ Use transformations (part of `ParamSet`)

PARAMETER TRANSFORMATION

- Sometimes we do not want to optimize over an evenly spaced range
- $k = 1$ vs. $k = 2$ probably more interesting than $k = 101$ vs. $k = 102$

$\Rightarrow$ Use transformations (part of `ParamSet`)

Example:

1. optimize from $\log(1) \ldots \log(100)$
2. transform by $\exp()$ in `trafo` function
3. don't forget to `round` (k must be integer)

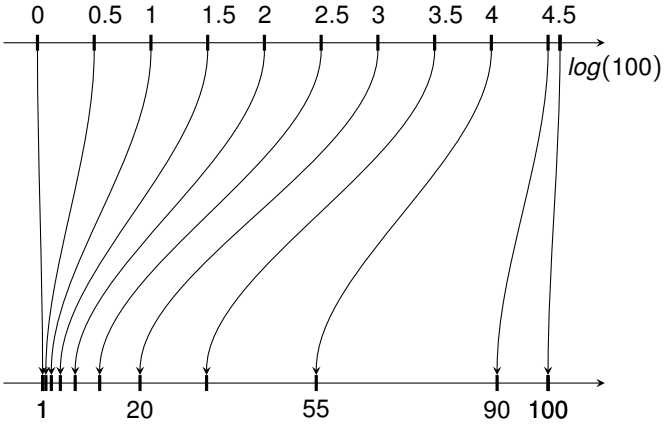
```
searchspace_knn_trafo = ps(
  "k" = p_dbl(log(1), log(35), trafo = function(x) round(exp(x)))
)
```

PARAMETER TRANSFORMATION

What is our transformation doing?

PARAMETER TRANSFORMATION

What is our transformation doing?



PARAMETER TRANSFORMATION

Tune again with `searchspace_knn_trafo` ...

```
inst_trafo = ti(task, learner, resampling, measure,
    terminator, searchspace_knn_trafo)
gsearch$optimize(inst_trafo)

#>        k learner_param_vals  x_domain classif.ce
#>    <num>             <list>    <list>      <num>
#> 1:   3.6          <list[1]> <list[1]>       0.21
```

PARAMETER TRANSFORMATION

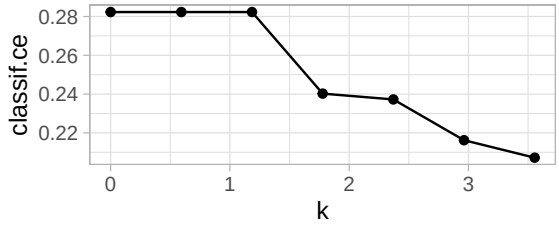
Tune again with `searchspace_knn_trafo` ...

```
inst_trafo = ti(task, learner, resampling, measure,
    terminator, searchspace_knn_trafo)
gsearch$optimize(inst_trafo)

#>        k learner_param_vals  x_domain classif.ce
#>    <num>             <list>    <list>      <num>
#> 1:   3.6          <list[1]> <list[1]>       0.21
```

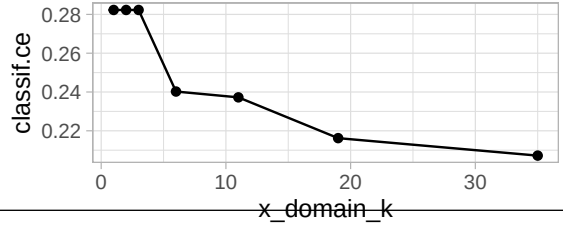
```
arv_trafo = as.data.table(inst_trafo$archive)
arv_trafo[, 1:4]

#>        k classif.ce x_domain_k runtime_learners
#>    <num>      <num>      <num>             <num>
#> 1:  2.96       0.22         19              0.13
#> 2:  3.56       0.21         35              0.19
#> 3:  1.78       0.24          6              0.11
#> 4:  1.19       0.28          3              0.17
#> 5:  0.59       0.28          2              0.11
#> 6:  0.00       0.28          1              0.12
#> 7:  2.37       0.24         11              0.12
```

PARAMETER TRANSFORMATION

```
ggplot(data = arv_trafo, aes(x = k, y = classif.ce)) +
  geom_line() + geom_point()
```

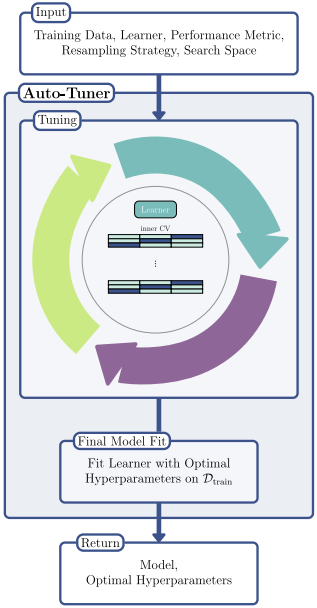



```
ggplot(data = arv_trafo, aes(x = x_domain_k, y = classif.ce)) +
  geom_line() + geom_point()
```



AutoTuner and Nested Resampling in mlr3

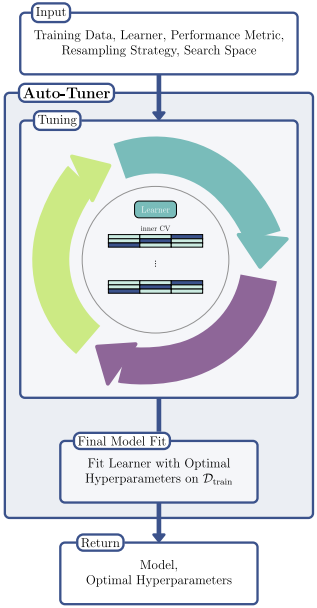
AUTOTUNER



Treat tuning as part of the learning process!

- Training:
  1. Tune model using (inner) resampling
  2. Train final model with optimal parameters on all data
- Predicting: Use final model

AUTOTUNER



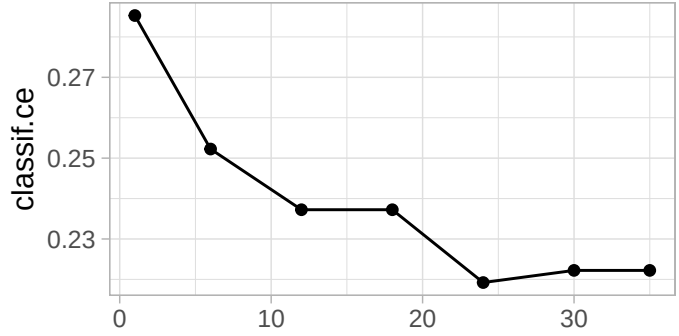
Treat tuning as part of the learning process!

- Training:
  1. Tune model using (inner) resampling
  2. Train final model with optimal parameters on all data
- Predicting: Use final model

```
optlrn = auto_tuner(
  tuner = tnr("grid_search", resolution = 7),
  learner = lrn("classif.kknn", k = to_tune(1, 35)),
  resampling = rsmp("holdout"),
  measure = msr("classif.ce"),
  terminator = trm("none"))
optlrn$train(tsk("german_credit"))
optlrn$learner # learner with best found HP

#> <LearnerClassifKKNN:classif.kknn>: k-Nearest-Neighbor
#> * Model: list
#> * Parameters: k=24
#> * Packages: mlr3, mlr3learners, kknn
#> * Predict Types:  [response], prob
#> * Feature Types: logical, integer, numeric,
#>   factor, ordered
#> * Properties: multiclass, twoclass
```

AUTOTUNER - ANALYZE TUNING RESULTS

```
arv = as.data.table(optlrn$tuning_instance$archive)
ggplot(arv, aes(x = k, y = classif.ce)) +
  geom_line() + geom_point() + xlab("")
```



AUTOTUNER - NESTED RESAMPLING

To estimate **tuned learner** performance, an outer resampling is required (performs nested resampling):

```
rr = resample(task = tsk("german_credit"), learner = optlrn,
  resampling = rsmp("cv", folds = 2), store_models = TRUE)
arv1 = as.data.table(rr$learners[[1]]$tuning_instance$archive)
arv2 = as.data.table(rr$learners[[2]]$tuning_instance$archive)
ggplot() +
  geom_line(data = arv1, aes(x = k, y = classif.ce)) +
  geom_line(data = arv2, aes(x = k, y = classif.ce), linetype = 2)
```