Hyperparameter Tuning with mlr3tuning::CHEAT SHEET

### Class Overview

The package provides a set of R6 classes which allow to (a) define general hyperparameter (HP) tuning instances, i.e., the black-box objective that maps HP configurations (HPCs) to resampled performance values; (b) run black-box optimizers; (c) combine learners with tuners (for nested resampling).

### TuningInstance* - Search Scenario

Evaluator and container for resampled performances of HPCs. The (internal) eval_batch(...) calls benchmark() to eval a table of HPCs. Stores archive of all evaluated experiments and final result. Tune instance = TuningInstanceSingleCrit(task, learner, resampling, measure, terminators, ss)

store Benchmark result = TRUE to store resampled evals and store_models = TRUE for fitted models.

### Example

```r
# Define task, learner
task = make_classif_task(id = "classif", n_samples = 1000)
learner = make_learner("classif.svm", id = "svm")

# Define search space
param_set = make_param_set("
  cost = p_dbl(lower = 1e-4, upper = 1e4),
  gamma = p_dbl(lower = 1e-4, upper = 1e4),
"

# Set up tuning instance
instance = make_tuning_instance(task, learner, param_set)

# Run tuning instance
result = instance$optimize()

# Access result
learner_param = instance$result$learner_param_vals
x_domain = instance$result$x_domain

```

### Tuner - Search Strategy

Generates HPCs and passes to tuning instance for evaluation until termination. Create: tuner = lrn("grid_search", batch_size = 5)

#### Example

```r
# Define learner
learner = make_learner("classif.svm", id = "svm")

# Define search space
param_set = make_param_set("gamma = p_dbl(lower = 1e-4, upper = 1e4),"

# Set up tuner
tuner = make_tuner(learner, param_set)

# Run tuner
result = tuner$optimize()

# Access result
learner_param = result$learner_param_vals
x_domain = result$x_domain
```

### Execute Tuning and Access Results

Get evaluated HPCs and performances; and result. x_domain_* cols contain HP values after trafo (if any).

```r
# Optimize HPCs in Learner
learner$tunable = TRUE

# Run tuner
result = learner$tune()

# Access result
learner_param = result$learner_param_vals
x_domain = result$x_domain
```

### Nested Resampling

Just resample AutoTuner; now has inner and outer loop.

#### Example

```r
# Define learner
learner = make_learner("grid_search", task = task, inner, outer, measure = "classif.ce")

# Run nested resampling
result = learner$tune_nested()

# Access result
x_domain = result$x_domain
```

### Terminators - When to stop

Construction: terminators = c("local", "global")

- **evals**: (n_evals)
- **run_time**: (run_time)
- **clock_time**: (clock_time)
- **perf_reached**: (perf_reached)
- **stagnation**: (stagnation)
- **run_time**: (run_time)
- **terminators**: (terminators)

### Loggning and Parallelization

```r
# Change log-level only for mlr3tuning.
future$plan("strategy")

# Sets the parallelization backend. Speeds up tuning by running iterations in parallel.
```