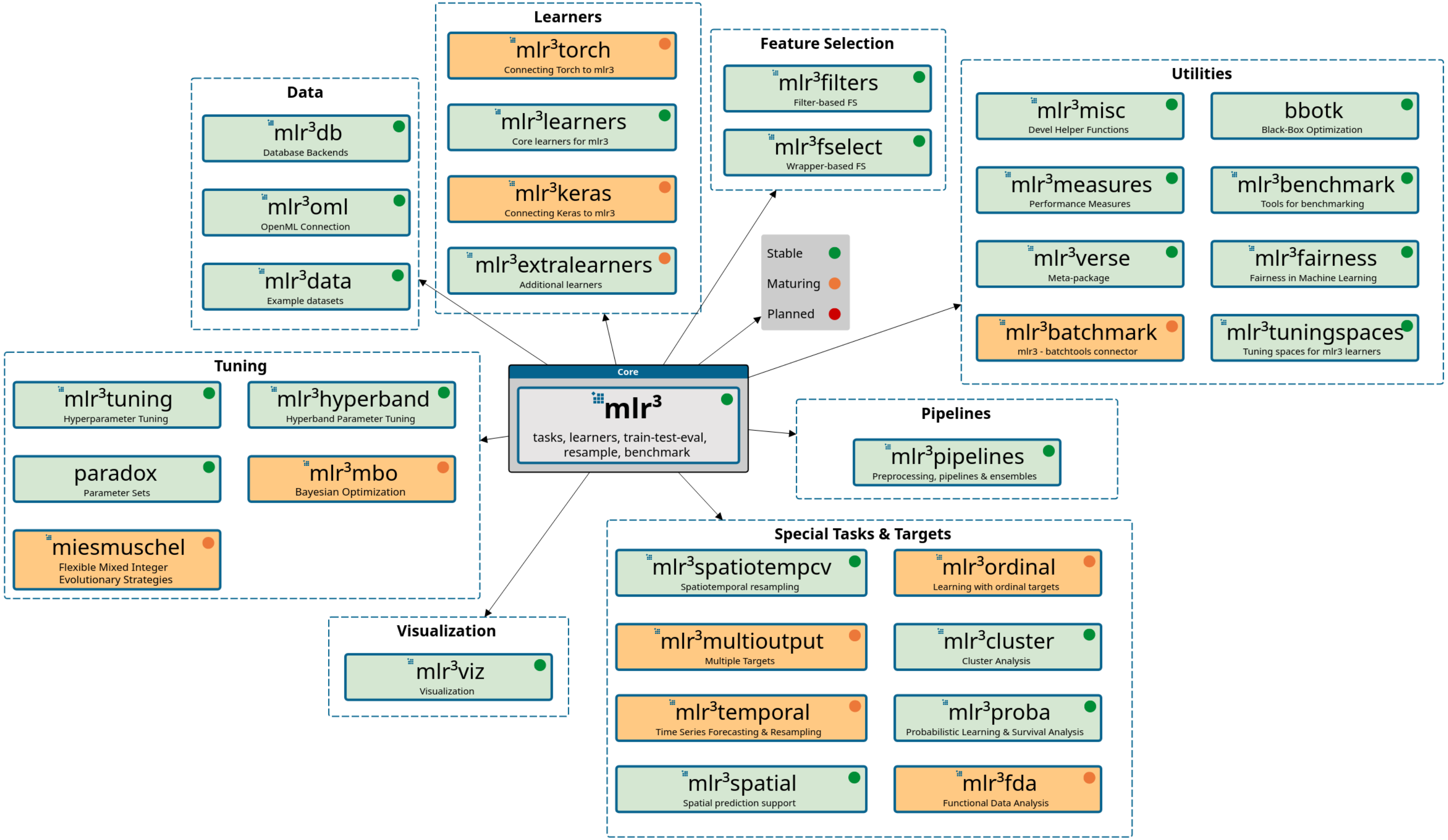
useR! Salzburg 2024

mlr3mbo: Modern and Flexible Bayesian Optimization

Lennart Schneider – LMU Munich & Munich Center for Machine Learning

mlr3 Ecosystem for Machine Learning



What is Black Box Optimization?

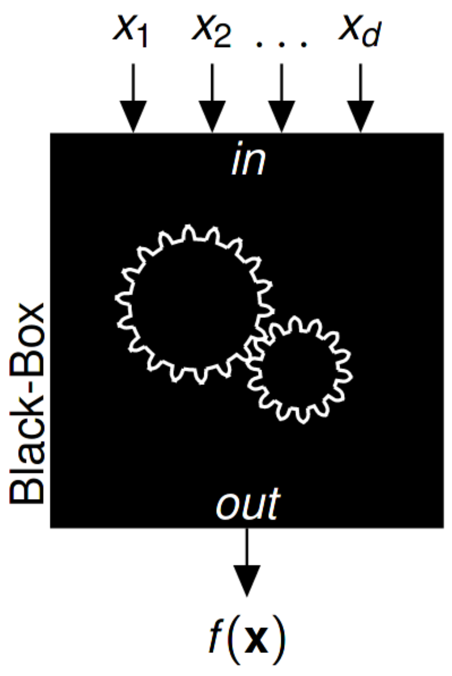
Optimization: Find

$$\min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x})$$

with objective function

$$f:\ \mathcal{S} \to \mathbb{R},$$

where $\mathcal{S}$ is usually box constrained.

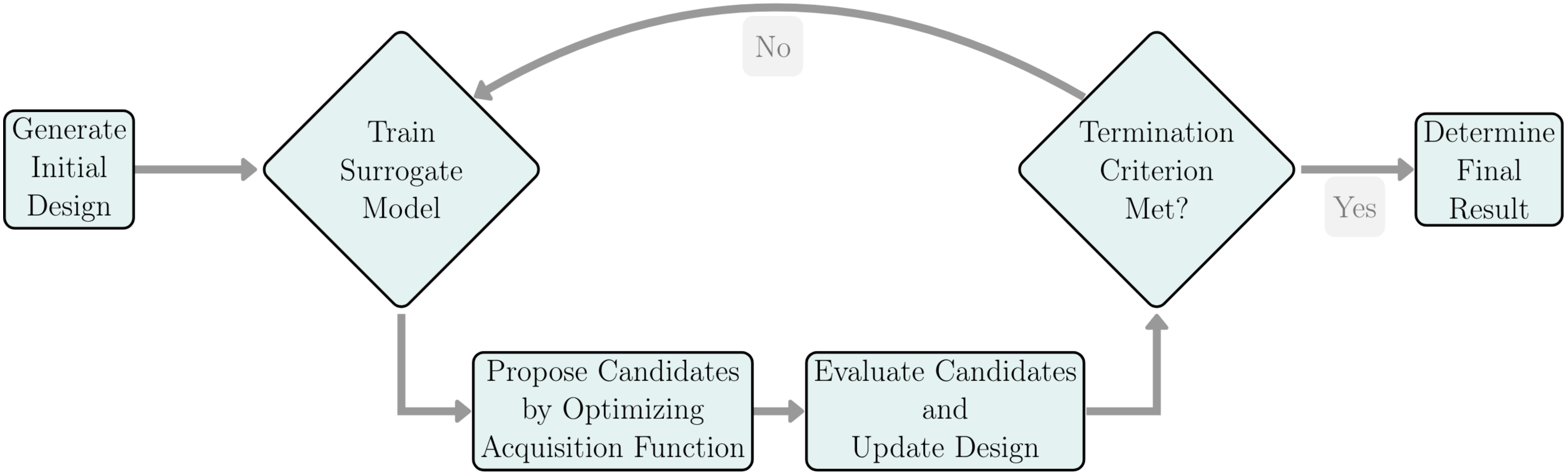


Optimization gets **really hard** if ...

- ... there is no analytic description of $f:\ \mathcal{S} \to \mathbb{R}$ (**black box**)
- ... evaluations of f for given values of $\mathbf{x}$ are **time consuming**

What is Bayesian Optimization?

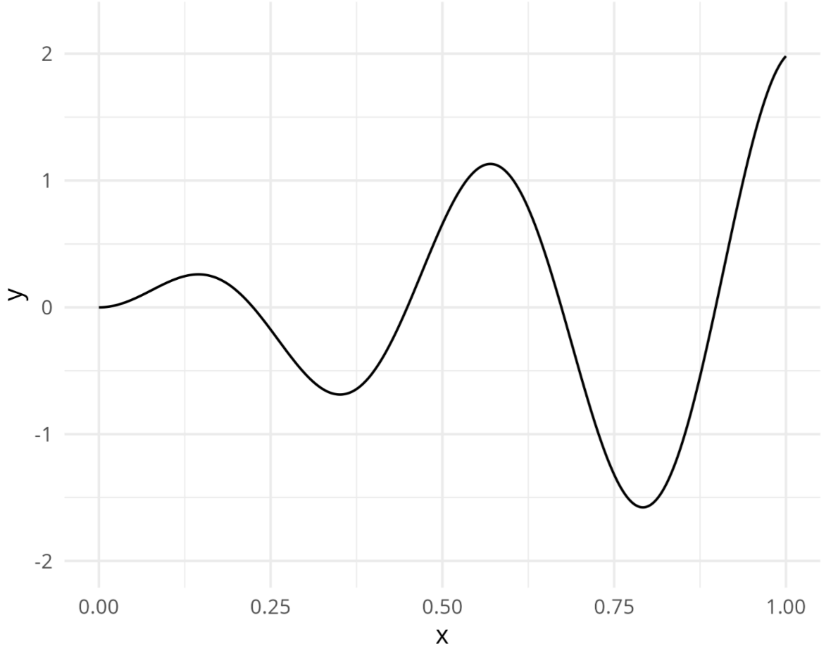
In a nutshell: smart + sample efficient way to (sequentially) optimize a black box



What is Bayesian Optimization?

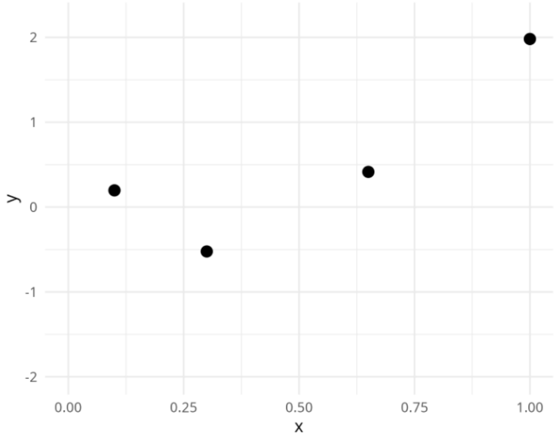
SURROGATE MODELING

Running example = minimize this “black-box”:



Starting point:

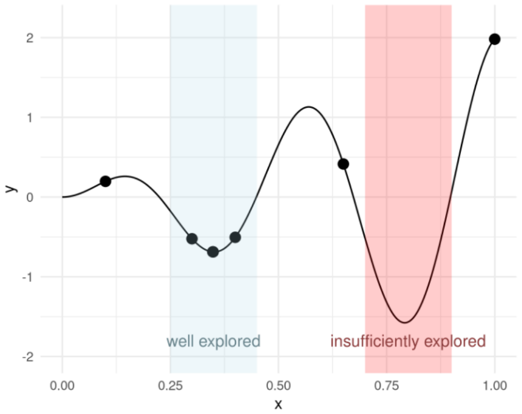
- We do not know the objective function $f : \mathcal{S} \to \mathbb{R}$
- But we can evaluate f for a few different inputs $\mathbf{x} \in \mathcal{S}$
- For now we assume that those evaluations are noise-free
- **Idea:** Use the data $\mathcal{D}^{[t]} = \{(\mathbf{x}^{[i]}, y^{[i]})\}_{i=1,\ldots,t}$, $y^{[i]} := f(\mathbf{x}^{[i]})$, to derive properties about the unknown function f



What is Bayesian Optimization?

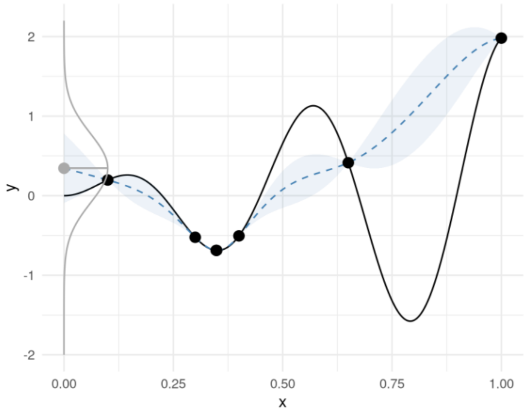
BAYESIAN SURROGATE MODELING

Goal:
Find trade-off between **exploration** (areas we have not visited yet) and **exploitation** (search around good design points)



BAYESIAN SURROGATE MODELING

- Denote by $Y \mid \mathbf{x}, \mathcal{D}^{[t]}$ the (conditional) RV associated with the posterior predictive distribution of a new point $\mathbf{x}$ under a SM; will abbreviate it as $Y(\mathbf{x})$
- Most prominent choice for a SM is a **Gaussian process**, here $Y(\mathbf{x}) \sim \mathcal{N}\left(\hat{f}(\mathbf{x}), \hat{s}^2(\mathbf{x})\right)$



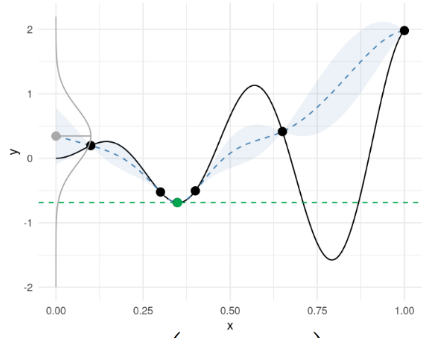
For now we assume an interpolating SM; $\hat{f}(\mathbf{x}) = f(\mathbf{x})$ and $\hat{s}(\mathbf{x}) = 0$ for training points

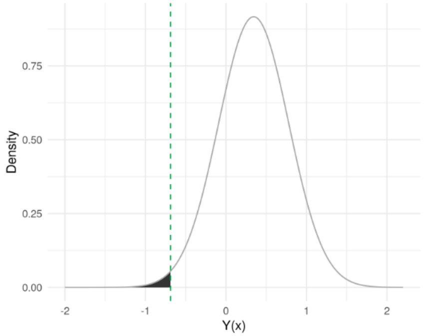
What is Bayesian Optimization?

EXPECTED IMPROVEMENT

Goal: Propose $\mathbf{x}^{[t+1]}$ that maximizes the **Expected Improvement** (EI):

$$a_{\text{EI}}(\mathbf{x}) \quad = \quad \mathbb{E}(\max\{f_{\min} - Y(\mathbf{x}), 0\})$$



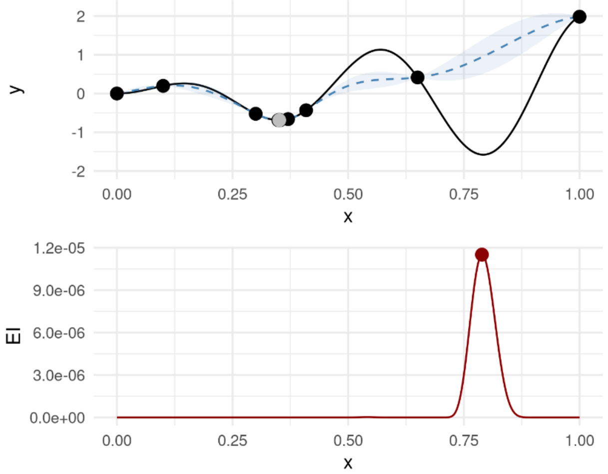


If $Y(\mathbf{x}) \sim \mathcal{N}\left(\hat{f}(\mathbf{x}), \hat{s}^2(\mathbf{x})\right)$, we can express the EI in closed-form as:

$$a_{\text{EI}}(\mathbf{x}) = (f_{\min} - \hat{f}(\mathbf{x}))\Phi\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\phi\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right),$$

EXPECTED IMPROVEMENT

The EI is capable of exploration and quickly proposes promising points in areas we have not visited yet



Here, also a result of well-calibrated uncertainty $\hat{s}(\mathbf{x})$ of our GP.

Building Blocks of Bayesian Optimization

Surrogate Model

Acquisition Function

Acquisition Function Optimizer

Optimizer

Loop Function

Surrogate Model

Acquisition Function

Acquisition Function Optimizer

mlr3mbo Building Blocks

Surrogate Model	`SurrogateLearner`	*Surrogate Model Containing a Single Learner*
Acquisition Function	`AcqFunction`	*Acquisition Function Base Class*
Acquisition Function Optimizer	`AcqOptimizer`	*Acquisition Function Optimizer*
Loop Function	`loop_function`	*Loop Functions for Bayesian Optimization*
Optimizer	`mlr_optimizers_mbo` `OptimizerMbo`	*Model Based Optimization*

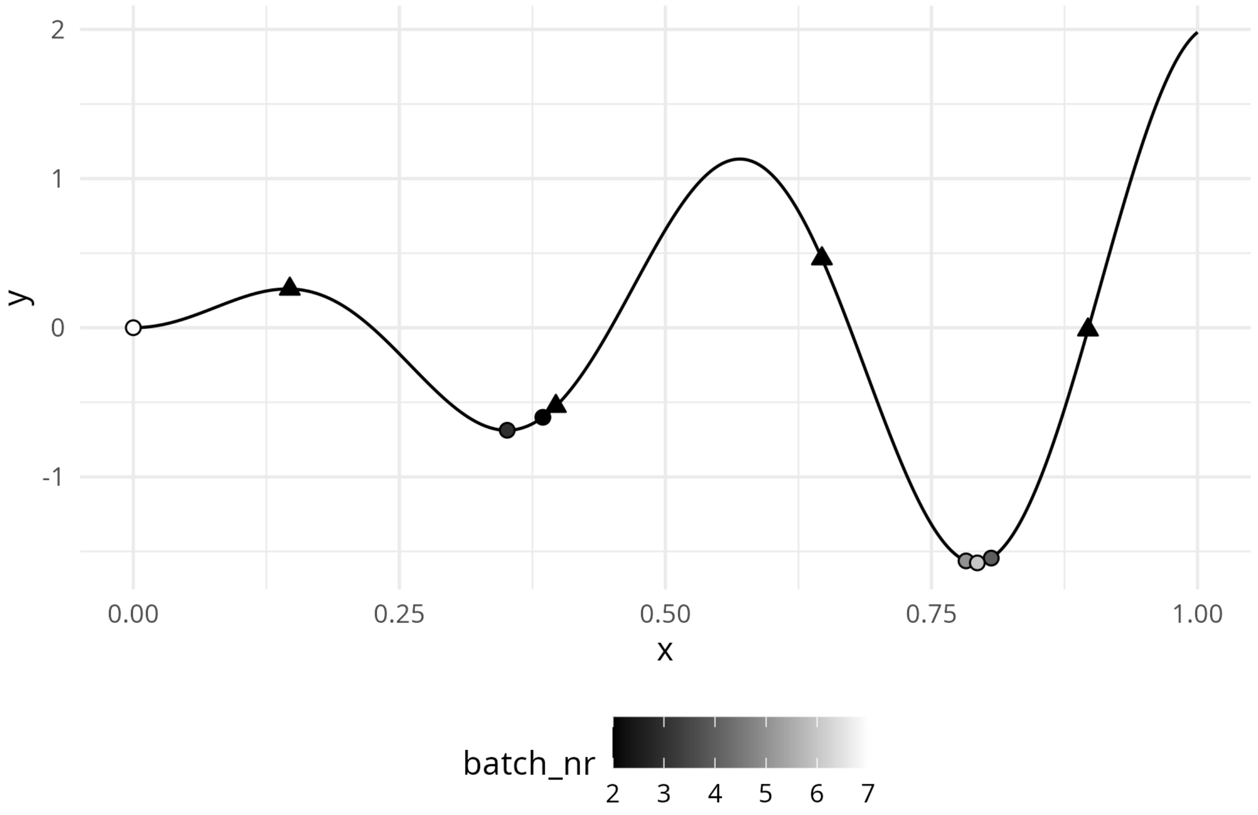
mlr3mbo Example

```
library(mlr3mbo)

sinus_1D = function (xs) 2 * xs$x * sin(14 * xs$x)
domain = ps(x = p_dbl(lower = 0, upper = 1))
codomain = ps(y = p_dbl(tags = "minimize"))

objective = ObjectiveRFun$new(sinus_1D,
  domain = domain, codomain = codomain)

instance = OptimInstanceBatchSingleCrit$new(objective,
  search_space = domain,
  terminator = trm("evals", n_evals = 10))

gp = srlrn(default_gp())
ei = acqf("ei")
direct = acqo(
  optimizer = opt("nloptr", algorithm =
"NLOPT_GN_ORIG_DIRECT"),
  terminator = trm("stagnation", iters = 100, threshold = 1e-
5))

optimizer = opt("mbo",
  loop_function = bayesopt_ego,
  surrogate = gp,
  acq_function = ei,
  acq_optimizer = direct)

optimizer$optimize(instance)
```

```
optimizer$optimize(instance)
INFO  [01:15:59.422] [bbotk] Starting to optimize 1 parameter(s) with '<OptimizerMbo>' and '<TerminatorEvals>
INFO  [01:15:59.465] [bbotk] Evaluating 4 configuration(s)
INFO  [01:15:59.510] [bbotk] Result of batch 1:
INFO  [01:15:59.513] [bbotk]          x          y
INFO  [01:15:59.513] [bbotk]  0.8970541 -0.0136594
INFO  [01:15:59.513] [bbotk]  0.3970540 -0.5262615
INFO  [01:15:59.513] [bbotk]  0.6470541  0.4631649
INFO  [01:15:59.513] [bbotk]  0.1470540  0.2597830
INFO  [01:16:01.058] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:01.082] [bbotk] Result of batch 2:
INFO  [01:16:01.087] [bbotk]          x x_domain    acq_ei .already_evaluated          y
INFO  [01:16:01.087] [bbotk]  0.3848245 <list[1]> 0.0430948              FALSE -0.6007964
INFO  [01:16:03.739] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:03.759] [bbotk] Result of batch 3:
INFO  [01:16:03.763] [bbotk]          x x_domain     acq_ei .already_evaluated          y
INFO  [01:16:03.763] [bbotk]  0.3513438 <list[1]> 0.05868126              FALSE -0.6877696
INFO  [01:16:07.093] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:07.110] [bbotk] Result of batch 4:
INFO  [01:16:07.114] [bbotk]          x x_domain     acq_ei .already_evaluated         y
INFO  [01:16:07.114] [bbotk]  0.8061425 <list[1]> 0.02873241              FALSE -1.544768
INFO  [01:16:08.629] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:08.652] [bbotk] Result of batch 5:
INFO  [01:16:08.660] [bbotk]          x x_domain     acq_ei .already_evaluated         y
INFO  [01:16:08.660] [bbotk]  0.7824773 <list[1]> 0.06894026              FALSE -1.563646
INFO  [01:16:11.105] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:11.122] [bbotk] Result of batch 6:
INFO  [01:16:11.125] [bbotk]          x x_domain     acq_ei .already_evaluated        y
INFO  [01:16:11.125] [bbotk]  0.7930956 <list[1]> 0.03032725              FALSE -1.57699
INFO  [01:16:12.568] [bbotk] Evaluating 1 configuration(s)
INFO  [01:16:12.586] [bbotk] Result of batch 7:
INFO  [01:16:12.597] [bbotk]            x x_domain      acq_ei .already_evaluated            y
INFO  [01:16:12.597] [bbotk]  2.540263e-05 <list[1]> 0.005038133              FALSE 1.806822e-08
INFO  [01:16:12.654] [bbotk] Finished optimizing after 10 evaluation(s)
INFO  [01:16:12.656] [bbotk] Result:
INFO  [01:16:12.659] [bbotk]          x x_domain        y
INFO  [01:16:12.659] [bbotk]      <num>    <list>    <num>
INFO  [01:16:12.659] [bbotk]  0.7930956 <list[1]> -1.57699
```

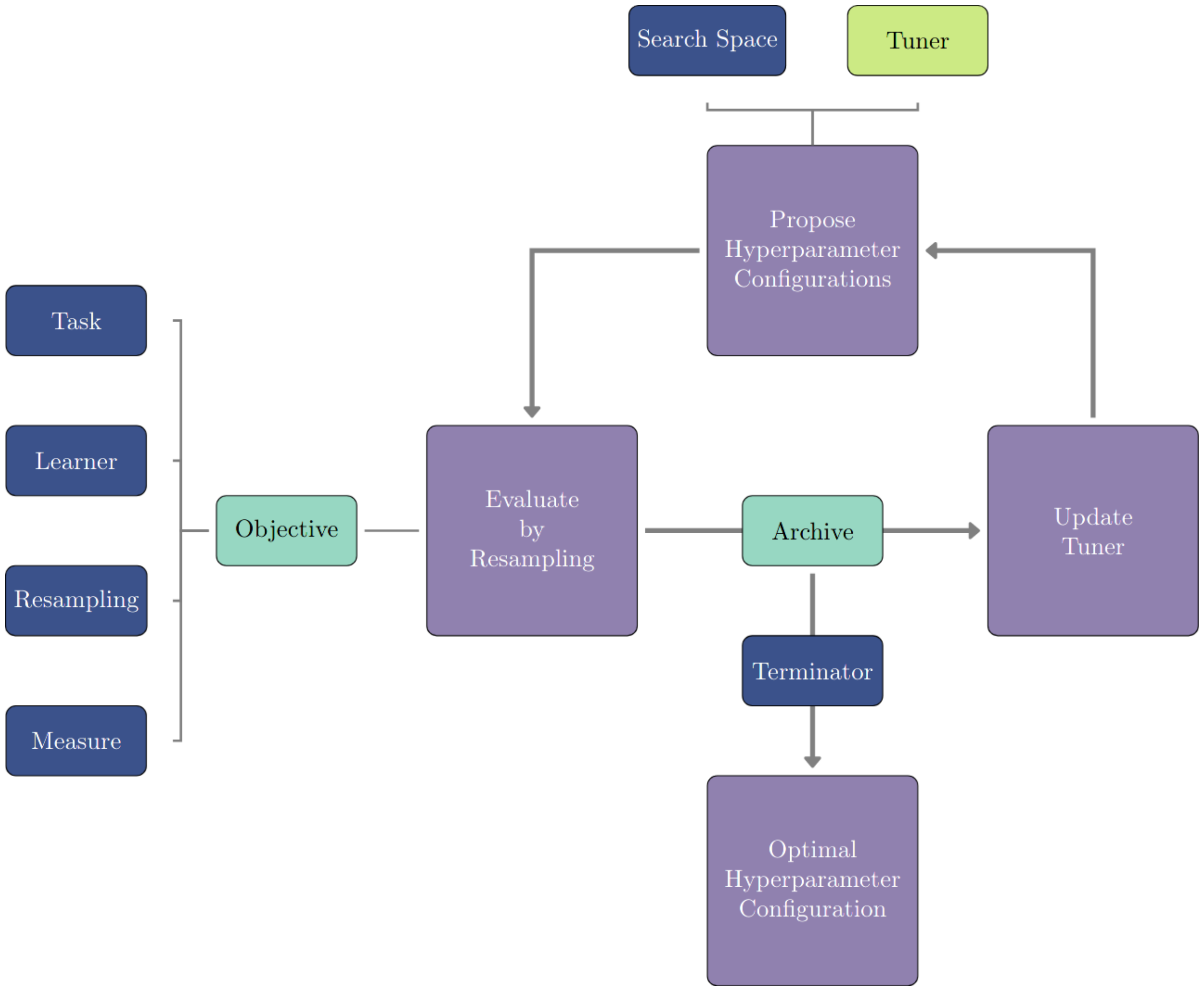
mlr3mbo Example

```
library(mlr3mbo)

sinus_1D = function (xs) 2 * xs$x * sin(14 * xs$x)
domain = ps(x = p_dbl(lower = 0, upper = 1))
codomain = ps(y = p_dbl(tags = "minimize"))

objective = ObjectiveRFun$new(sinus_1D,
  domain = domain, codomain = codomain)

instance = OptimInstanceBatchSingleCrit$new(objective,
  search_space = domain,
  terminator = trm("evals", n_evals = 10))

gp = srlrn(default_gp())
ei = acqf("ei")
direct = acqo(
  optimizer = opt("nloptr", algorithm =
"NLOPT_GN_ORIG_DIRECT"),
  terminator = trm("stagnation", iters = 100, threshold = 1e-
5))

optimizer = opt("mbo",
  loop_function = bayesopt_ego,
  surrogate = gp,
  acq_function = ei,
  acq_optimizer = direct)

optimizer$optimize(instance)
```



Hyperparameter Optimization (HPO)

- HPO of ML models == Black Box Optimization
- Estimate performance of a **learner** configured by a hyperparameter configuration via a **resampling** method on a **task** based on a performance **measure**
- HPO is costly and calls for **sample efficient** black box optimization
- This naturally calls for Bayesian Optimization



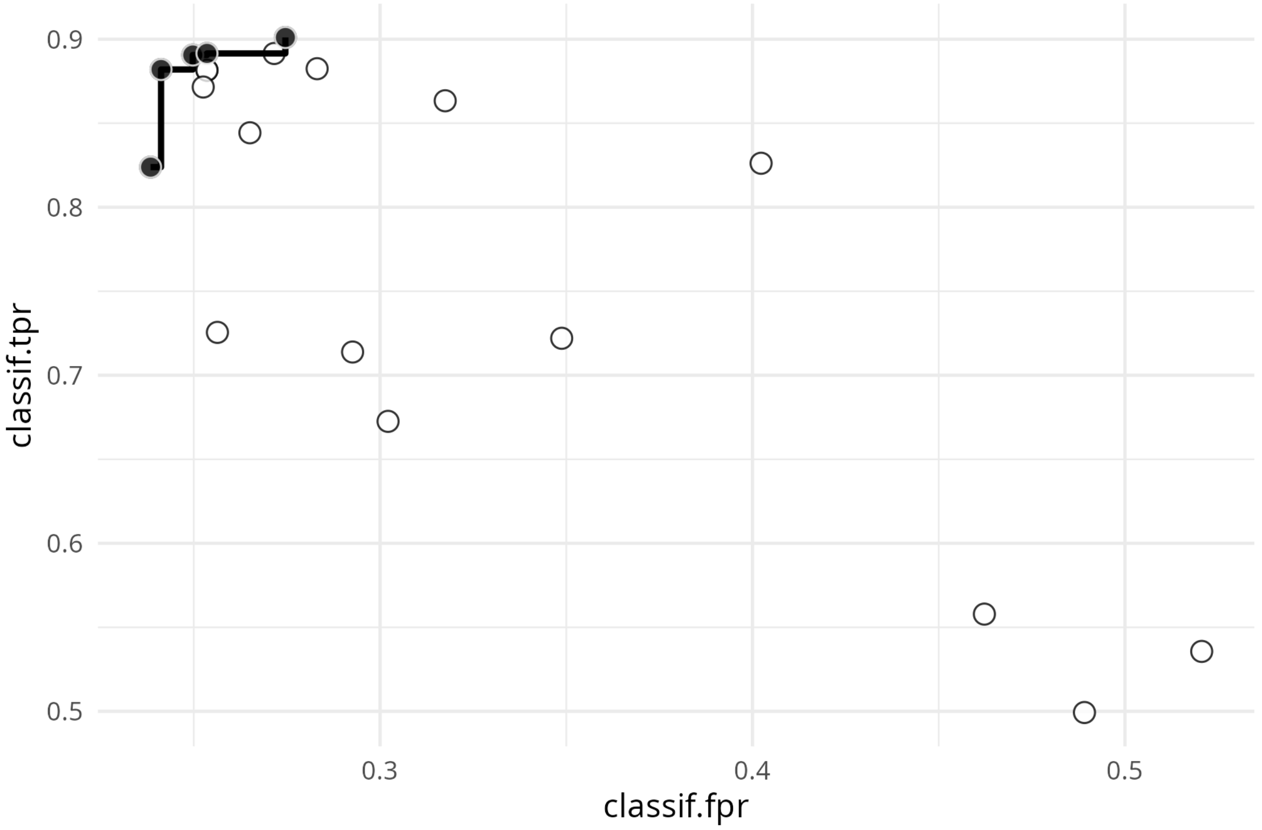
mlr3mbo HPO Example 1

```
library(mlr3mbo)
library(mlr3tuning)

tuner = tnr("mbo",
  loop_function = bayesopt_ego,
  surrogate = gp,
  acq_function = ei,
  acq_optimizer = direct)

xgboost = lrn("classif.xgboost",
  nrounds = to_tune(1, 5000, logscale = TRUE),
  eta = to_tune(0.001, 1, logscale = TRUE),
  alpha = to_tune(0.001, 1000, logscale = TRUE))

instance = tune(tuner,
  task = tsk("sonar"),
  learner = xgboost,
  resampling = rsmp("cv", folds = 5),
  Measures = msr("classif.ce"),
  term_evals = 20)
```

	alpha	eta	nrounds	classif.ce
1:	1.5557273	-5.3576940146	7.3248151	0.2166086
2:	-5.3520281	-1.9038161692	3.0661182	0.1729384
3:	5.0096050	-0.1768773495	0.9367700	0.4570267
4:	-1.8981507	-3.6307551949	5.1954668	0.1348432
5:	6.7365438	-4.4942246047	4.1307924	0.4711963
6:	-0.1712119	-1.0403467594	8.3894892	0.1591173
7:	-3.6250893	-6.2211633730	2.0014441	0.2644599
8:	3.2826662	-2.7672855791	6.2601409	0.4945412
9:	0.6922579	-1.4720814643	2.5337812	0.1923345
10:	-6.2154975	-4.9259593097	6.7924780	0.1444832
11:	-2.7616201	-3.1990202841	0.4044329	0.2981417
12:	4.1461356	-6.6528981037	4.6631297	0.4761905
13:	-0.7201501	-6.9061760025	6.5175389	0.2163763
14:	-1.9519857	-0.0047378294	8.5115513	0.1586527
15:	-5.1737097	-4.6004323566	8.5115513	0.1204413
16:	-5.3723476	-6.9075798038	8.5167441	0.1348432
17:	-5.3723476	-0.0005264255	7.5703671	0.1828107
18:	-4.6051702	-4.9036534388	7.5885417	0.1348432
19:	-1.9583028	-3.8360625452	8.5154459	0.**1204413**
20:	-1.5350567	-3.0653756279	8.5154459	0.1252033

mlr3mbo HPO Example 2 - Multiple Objectives

```
tuner = tnr("mbo",
  loop_function = bayesopt_parego,
  surrogate = gp,
  acq_function = ei,
  acq_optimizer = direct)

xgboost = lrn("classif.xgboost",
  nrounds = to_tune(1, 5000, logscale = TRUE),
  eta = to_tune(0.001, 1, logscale = TRUE),
  alpha = to_tune(0.001, 1000, logscale = TRUE))

instance = tune(tuner,
  task = tsk("sonar"),
  learner = xgboost,
  resampling = rsmp("cv", folds = 5),
  measures = msrs(c("classif.tpr", "classif.fpr")),
  term_evals = 20)
```



mlr3mbo Sugar

as.data.table(mlr_loop_functions) # list loop functions
1: bayesopt_ego Efficient Global Optimization single - crit
mlr3mbo::mlr_loop_functions_ego
2: bayesopt_emo Multi - Objective EGO multi - crit mlr3mbo::mlr_loop_functions_emo
3: bayesopt_mpcl Multipoint Constant Liar single - crit mlr3mbo::mlr_loop_functions_mpcl
4: bayesopt_parego ParEGO multi - crit mlr3mbo::mlr_loop_functions_parego
5: bayesopt_smsego SMS - EGO multi - crit mlr3mbo::mlr_loop_functions_smsego

as.data.table(mlr_learners) # list learners, get via lrn() and wrap as a surrogate via srlrn()
.
.
.

as.data.table(mlr_acqfunctions) # list acquisition functions, get via acqf()
  1: aei Augmented Expected Improvement mlr3mbo::mlr_acqfunctions_aei
  2: cb Lower / Upper Confidence Bound mlr3mbo::mlr_acqfunctions_cb
  3: ehvi Expected Hypervolume Improvement mlr3mbo::mlr_acqfunctions_ehvi
  4: ehvigh Expected Hypervolume Improvement via GH Quadrature
mlr3mbo::mlr_acqfunctions_ehvigh
  5: ei Expected Improvement mlr3mbo::mlr_acqfunctions_ei
  6: eips Expected Improvement Per Second mlr3mbo::mlr_acqfunctions_eips
  7: mean Posterior Mean mlr3mbo::mlr_acqfunctions_mean
  8: pi Probability Of Improvement mlr3mbo::mlr_acqfunctions_pi
  9: sd Posterior Standard Deviation mlr3mbo::mlr_acqfunctions_sd
10: smsego SMS - EGO mlr3mbo::mlr_acqfunctions_smsego

as.data.table(mlr_optimizers) # list optimizers, get via opt()
.
.
.

mlr3mbo Features and Roadmap

Features:

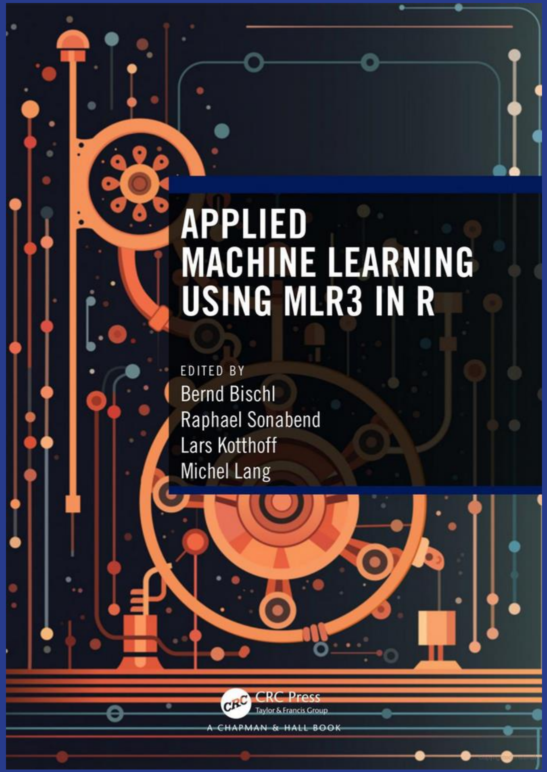
- Write custom loop functions and use them within any **OptimizerMbo** or **TunerMbo** for black box optimization and HPO
- Wrap any mlr3 regression learner as a surrogate
- Implement custom acquisition functions easily
- Wrap custom acquisition function optimizers
- Supports mixed-integer and hierarchical search spaces
- Can perform multi-objective optimization
- Uses "intelligent" defaults if loop function, surrogate, acquisition function or acquisition function optimizer are not provided, see **?mbo_defaults**

Roadmap:

- Better defaults
- Support non-myopic acquisition functions
- Make use of gradient information during acquisition function optimization
- Better support for batch parallel optimization
- Fully support asynchronous optimization

https://github.com/mlr-org/mlr3mbo
https://cran.r-project.org/package=mlr3mbo

THANK YOU!



SCAN ME