mlrMBO: A Toolbox for Model-Based Optimization of Expensive Black-Box Functions

Bernd Bischl\textsuperscript{1} \quad Jakob Richter\textsuperscript{2} \quad Jakob Bossek\textsuperscript{3} \quad Daniel Horn\textsuperscript{2} \quad Michel Lang\textsuperscript{2}

June 28, 2016

\textsuperscript{1}Department of Statistics, LMU Munich

\textsuperscript{2}Faculty of Statistics, TU Dortmund University

\textsuperscript{3}Westfälische-Wilhelms-Universität Münster, Germany
When to use mlrMBO?
When to use mlrMBO?

Answer: When `optim(par, f(x))` is not enough!

- `f(x)` is expensive. (That's why GAs won't work!)
- `f(x)` is not convex.
- `f(x)` is noisy.
- `par` is not only numeric.
When to use mlrMBO?

Answer: When `optim(par, f(x))` is not enough!

- `f(x)` is expensive.
  (→ that’s why GAs won’t work!)
- `f(x)` is not convex.
- `f(x)` is noisy.
- `par` is not only numeric.

![Graphs showing different optimization landscapes](image_url)
Model-Based Optimization
Expensive Black-Box Optimization

\[ y = f(x), \quad f : \mathbb{X} \rightarrow \mathbb{R} \]
\[ x^* = \arg\min_{x \in \mathbb{X}} f(x) \]

- \( y \), target value
- \( x \in \mathbb{X} \subseteq \mathbb{R}^d \), domain
- \( f(x) \) function with considerably long runtime
- Goal: Find optimum \( x^* \)
Basic Idea

Function evaluations are expensive, so keep number of black-box evaluations low

- Try to predict function values by regression model.
  → surrogate model
- Search for points leading to finding the optimum on the surrogate model.
- Update surrogate model with evaluated points.

Search mechanism balances exploitation and exploration.

- Just evaluate $\mathbf{x}$ where
  - Predicted function value is low: $\Downarrow \hat{y}(\mathbf{x})$
  - Uncertainty is high: $\Uparrow s^2(\mathbf{x})$
  ⇒ infill criterion: $\text{Inf}(\mathbf{x})$
- Popular choice proposed by Jones et al. (1998):
  Improvement: $I(\mathbf{x}) = \max(0, |f(\mathbf{x}^*) - f(\mathbf{x})|)$
  Expected Improvement $EI(\mathbf{x}) = E(I(\mathbf{x}))$
Evaluate initial design ♦ to generate surrogate model  \( \hat{y} \) ——, propose new point ▲ based on maximum of  \( EI \) ——.
Update surrogate model $\hat{y}$ with evaluated point ■, propose new point ▲ based on maximum of $EI$. 

Iter = 2, Gap = 1.3598e-01
Iter = 3, Gap = 3.9405e−02
MBO Visualization

...until budget is exhausted.
Application
Expensive Black-Box Optimization

mlrMBO can be used for:

- Expensive Black-Box Optimization
- Hyperparameter Tuning for Machine Learning Methods
- Machine Learning Pipeline Configuration
- Algorithm Configuration
- ...

Jakob Richter (TU Dortmund, Germany)
Hyperparameter Tuning

• Still common practice: grid search
  For a SVM it might look like:
  • $C \in (2^{-12}, 2^{-10}, 2^{-8}, \ldots, 2^8, 2^{10}, 2^{12})$
  • $\gamma \in (2^{-12}, 2^{-10}, 2^{-8}, \ldots, 2^8, 2^{10}, 2^{12})$
  • Evaluate all $13^2 = 169$ combinations $C \times \gamma$

• Bad because:
  • optimum might be ”off the grid”
  • lots of evaluations in bad areas
  • lots of costly evaluations

• How bad? ➔
Hyperparameter Tuning

- Because of budget restrictions grid might even be smaller!
- Unpromising area quite big!
- Lots of costy evaluations!

With mlrMBO it’s not hard to do it better! →

Jakob Richter (TU Dortmund, Germany)
# Define classification learner and its Hyper Parameter search space

```r
lrn = makeLearner("classif.svm")
ps = makeParamSet(
    makeNumericParam("cost", -15, 15, trafo = function(x) 2^x),
    makeNumericParam("gamma", -15, 15, trafo = function(x) 2^x))
```

# Define Tuning Problem

```r
mbo.ctrl = makeMBOControl()
```

```r
mbo.ctrl = setMBOControlTermination(mbo.ctrl, iters = 10)
```

```r
surrogate.lrn = makeLearner("regr.km", predict.type = "se")
ctrl = mlr::makeTuneControlMBO(learner = surrogate.lrn,
    mbo.control = mbo.ctrl, same.resampling.instance = FALSE)
rdesc = makeResampleDesc("Subsample", iters = 10)
res.mbo = tuneParams(lrn, sonar.task, rd, par.set = ps,
    control = ctrl, show.info = FALSE)
```
Grid Search vs. MBO

grid search

MBO

SVM, RBF, gamma

SVM, RBF, cost

mmce

Jakob Richter (TU Dortmund, Germany)
Hyperparameter Tuning

Compare results:

# Grid Tuning Result:
res.grid

## Tune result:
## Op. pars: cost=64; gamma=0.0156
## mmce.test.mean=0.147

# Tuning Costs (Time):
sum(getOptPathExecTimes(res.grid$opt.path))

## [1] 17.967

# MBO Tuning Result:
res.mbo

## Tune result:
## Op. pars: cost=1.32e+03; gamma=0.00938
## mmce.test.mean=0.133

# Tuning Costs (Time):
sum(getOptPathExecTimes(res.mbo$opt.path))

## [1] 12.764

<table>
<thead>
<tr>
<th>misclassification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>grid search</td>
</tr>
</tbody>
</table>

Jakob Richter (TU Dortmund, Germany)
Compare to CMAES

It’s not hard to beat grid search! How about a state of the art optimizer?

MBO vs. cmaes on Rosenbrock 5D

Jakob Richter (TU Dortmund, Germany)
Extensions
Extensions

- Different surrogate models to support mixed valued domain \( X \)

\[
\begin{align*}
x_1 \text{ (kernel)} & \quad x_2 \text{ (cost)} \\
\text{linear} & \quad [0, \infty] \\
\text{radial} & \quad [0, \infty] \\
x_3 \text{ (gamma)} & \\
\end{align*}
\]

\[
\text{ps} = \text{makeParamSet(}
\begin{align*}
\text{makeDiscreteParam(id = "kernel",}
& \quad \text{values = c("linear", "radial")),}
\text{makeNumericParam(id = "cost"),}
\text{makeNumericParam(id = "gamma",}
& \quad \text{requires = quote(kernel!="linear"))}
\end{align*}
\)

- Different Infill-Criteria: \textit{mean}, \textit{EI}, \textit{CB}, ...

- Batch proposal for easy parallelization

```r
library(parallelMap)
ctrl = makeMBOControl(
  propose.points = 4L, ...)
# ...
parallelStartMulticore(4)
mbo(...)
parallelStop()
```
Advanced Extensions
Expensive Black-Box Optimization

\[
\min_{x \in \mathbb{X}} \mathbf{f}(x) = \mathbf{y} = (y_1, \ldots, y_m) \text{ with } \mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m
\]

- **\( y \) dominates \( \tilde{y} \) if**
  \[
  \forall i \in \{1, \ldots, m\} : y_i \leq \tilde{y}_i \\
  \text{and } \exists i \in \{1, \ldots, m\} : y_i < \tilde{y}_i
  \]

- Set of non-dominated solutions:
  \[
  \mathcal{X}^* := \{ x \in \mathcal{X} : \nexists \tilde{x} \in \mathcal{X} : \mathbf{f}(\tilde{x}) \text{ dominates } \mathbf{f}(x) \}
  \]

- \( \mathcal{X}^* \) is called Pareto set, \( \mathbf{f}(\mathcal{X}^*) \) Pareto front
- Goal: Find \( \hat{\mathcal{X}}^* \) of non-dominated points that estimates the true set \( \mathcal{X}^* \) Different methods for mlrMBO discussed in Horn et al. (2015).
Conclusion
### Conclusion

**Why use mlrMBO?**
- Efficient model based optimizer
- Powerfull toolbox for a wide variety of set-ups
- Different black box scenarios covered
- Improved exploration of search space within time budget
- *mlrMBO* is easy to use!

**Outlook**
- Improve user friendliness
- Improve parallel computation

We use R: Find us on GitHub
- [github.com/mlr-org/mlr](https://github.com/mlr-org/mlr)
- [github.com/mlr-org/mlrMBO](https://github.com/mlr-org/mlrMBO)
References

- Bischl, Bernd et al. (2014). “MOI-MBO: Multiobjective Infill for Parallel Model-Based Optimization”. In: Learning and Intelligent Optimization Conference. Florida. DOI: 10.1007/978-3-319-09584-4_17.