Model-Based Optimization for Expensive Black-Box Problems and Hyperparameter Optimization

Bernd Bischl

Computational Statistics, LMU Munich

Dec 1st, 2017

Sequential model-based optimization

Parallel batch proposals

Multicriteria SMBO

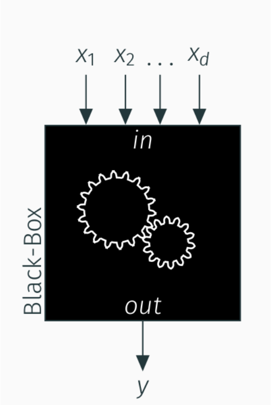
Interesting Challenges

ML Model Selection and Hyperparameter Optimization

Section 1

Sequential model-based optimization

Expensive Black-Box Optimization



$$y = f(\boldsymbol{x}) \;, \quad f : \mathbb{X} \to \mathbb{R} \tag{1}$$

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{X}} f(\boldsymbol{x}) \tag{2}$$

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

Sequential model-based optimization

- Setting: Expensive black-box problem $f : x \to \mathbb{R} = min$!
- Classical problem: Computer simulation with a bunch of control parameters and performance output; or algorithmic performance on 1 or more problem instances; we often optimize ML pipelines
- Idea: Let's approximate f via regression!

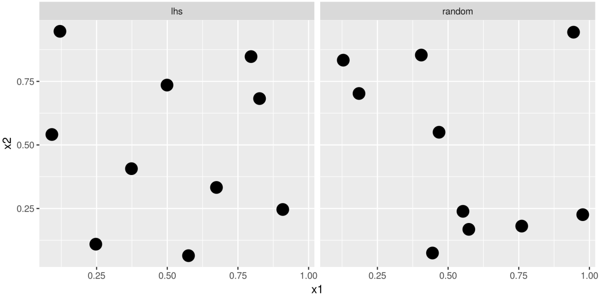
Generic MBO Pseudo Code

- Create initial space filling design and evaluate with f
- In each iteration:
  - Fit regression model on all evaluated points to predict $\hat{f}(\mathbf{x})$ and uncertainty $\hat{s}(\mathbf{x})$
  - Propose point via infill criterion

$$\mathrm{EI}(x) \uparrow \iff \hat{f}(\mathbf{x}) \downarrow \wedge \hat{s}(\mathbf{x}) \uparrow$$

  - Evaluate proposed point and add to design
  - EGO proposes kriging (aka Gaussian Process) and EI
    Jones 1998, Efficient Global Opt. of Exp. Black-Box Functions

Latin Hypercube Designs

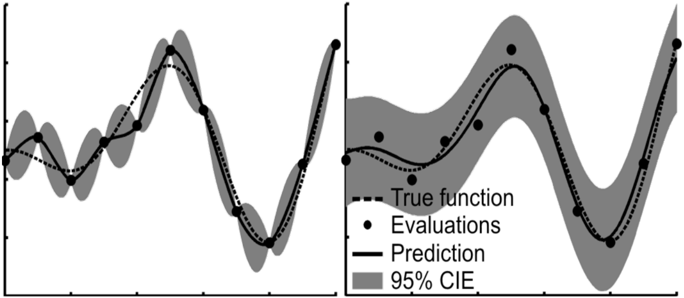


- Initial design to train first regression model
- Not too small, not too large
- LHS / maximin designs: Min dist between points is maximized
- But: Type of design usually has not the largest effect on MBO, and unequal distances between points could even be beneficial

Kriging and local uncertainty prediction

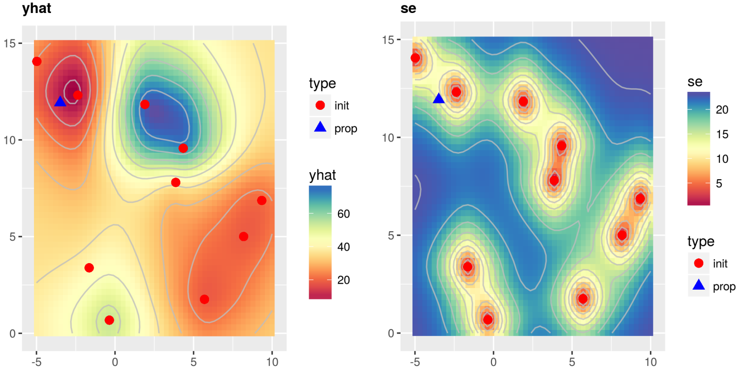
Model: Zero-mean GP $Y(x)$ with const. trend and cov. kernel $k_\theta(x_1, x_2)$.

- $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, $\boldsymbol{K} = (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1,\ldots,n}$
- $\boldsymbol{k}_*(\boldsymbol{x}) = (k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}))^T$
- $\hat{\mu} = \mathbf{1}^T \boldsymbol{K}^{-1} \boldsymbol{y} / \mathbf{1}^T \boldsymbol{K}^{-1} \mathbf{1}$ (BLUE)
- Prediction: $\hat{f}(\boldsymbol{x}) = E[Y(x)|Y(x_i) = y_i, i = 1, \ldots, n] =$
  $\hat{\mu} + \mathbf{k}_n(x)^T K^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1})$
- Uncertainty: $\hat{s}^2(\boldsymbol{x}) = Var[Y(x)|Y(x_i) = y_i, i = 1, \ldots, n] =$
  $\sigma^2 - \mathbf{k}_n^T(x) K^{-1} \mathbf{k}_n(x) + \frac{(1 - \mathbf{1}^T K^{-1} \mathbf{k}_n^T(x))^2}{\mathbf{1}^T K^{-1} \mathbf{1}}$



Kriging / GP is a spatial model

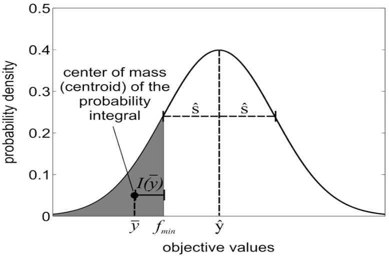
- Correlation between outcomes (y_1, y_2) depends on dist of x_1, x_2
  E.g. Gaussian covar kernel $k(x_1, x_2) = exp(\frac{-||x_1 - x_2||}{2\sigma})$
- Useful smoothness assumption for optimization
- Posterior uncertainty at new x increases with dist to design points
- Allows to enforce exploration

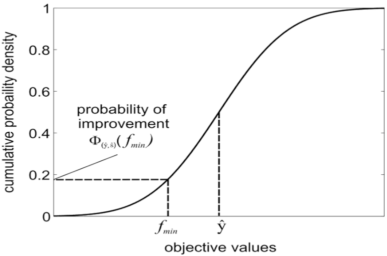


Infill Criteria: Expected Improvement

- Define improvement at x over best visited point with $y = f_{min}$ as random variable $I(x) = |f_{min} - Y(x)|^+$
- For kriging $Y(x) \sim N(\hat{f}(\boldsymbol{x}), \hat{s}^2(\boldsymbol{x}))$ (given $x = x$)
- Now define $EI(x) = E[I(x)|x = x]$
- Expectation is integral over normal density starting at f_{min}
- Alternative: Lower confidence bound (LCB) $\hat{f}(\boldsymbol{x}) - \lambda \hat{s}(\boldsymbol{x})$

Result: $EI(x) = \left(f_{min} - \hat{f}(\boldsymbol{x})\right) \Phi\left(\frac{f_{min} - \hat{f}(\boldsymbol{x}))}{\hat{s}(\boldsymbol{x})}\right) + \hat{s}(\boldsymbol{x}) \phi\left(\frac{f_{min} - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right)$

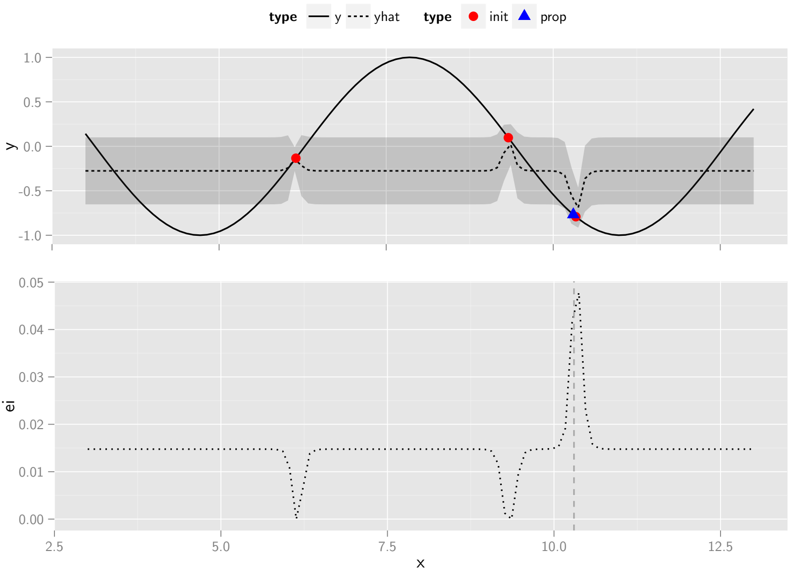


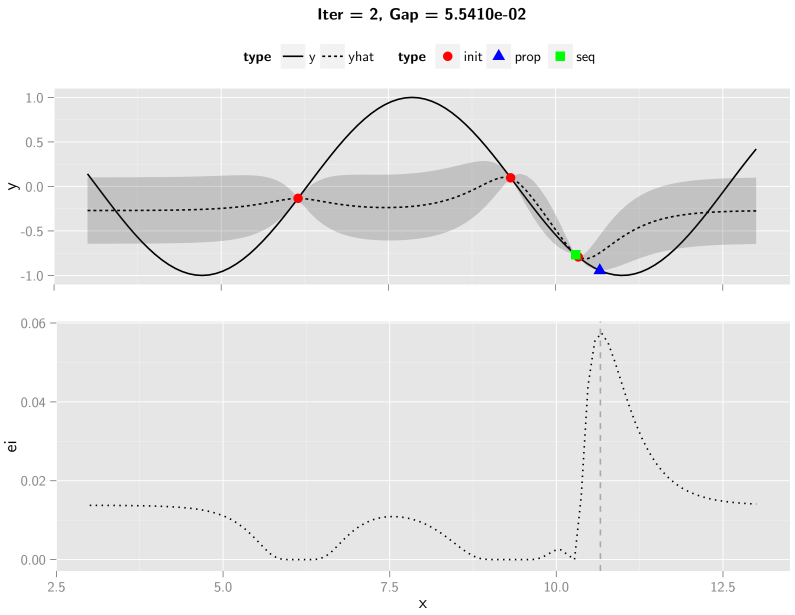


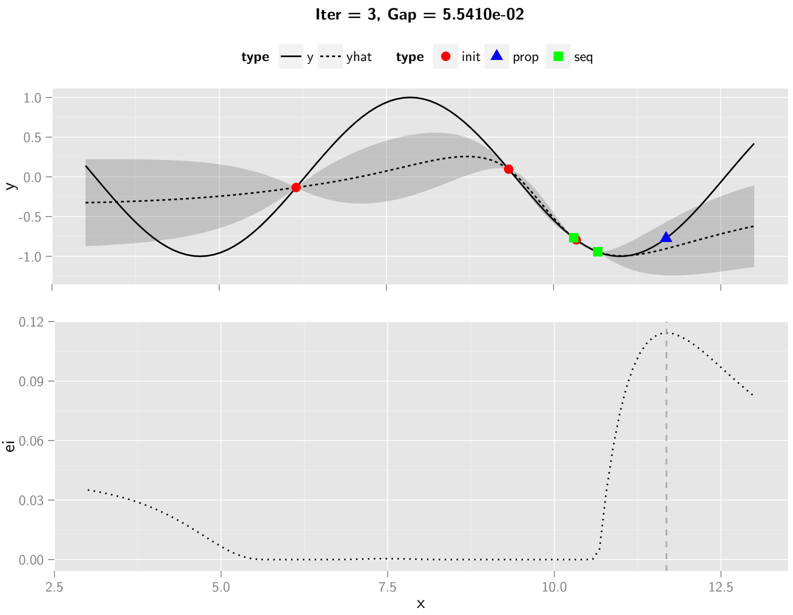
FOCUSSEARCH

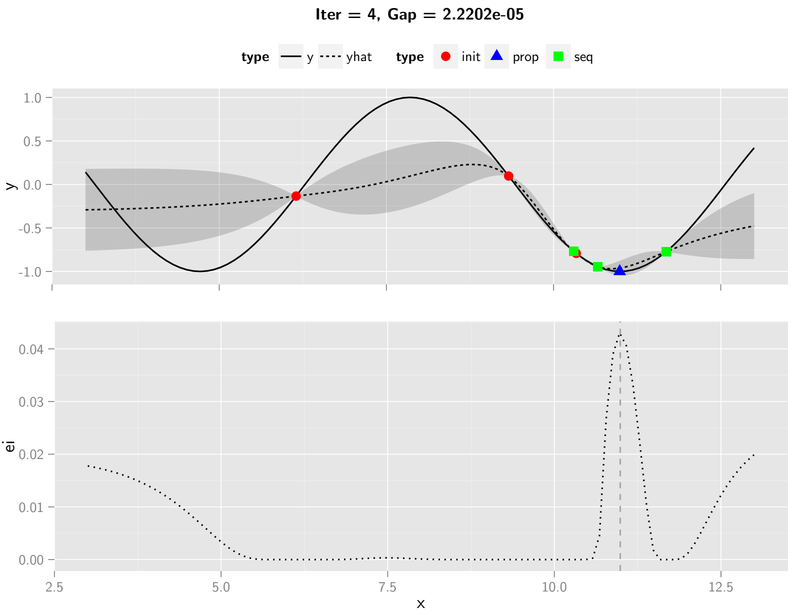
- EI optimization is multimodal and not that simple
- But objective is now cheap to evaluate
- Many different algorithms exist, from gradient-based methods with restarts to evolutionary algorithms
- We use an iterated, focusing random search coined "focus search"
- In each iteration a random search is performed
- We then shrink the constraints of the feasible region towards the best point in the current iteration (focusing) and iterate, to enforce local convergence
- Whole process is restarted a few times
- Works also for categorical and hierarchical params

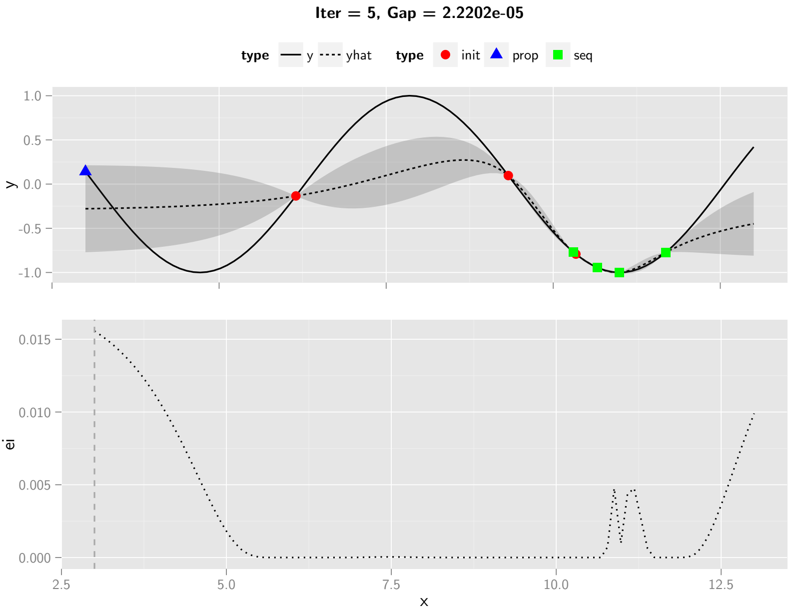
Iter = 1, Gap = 2.0795e-01



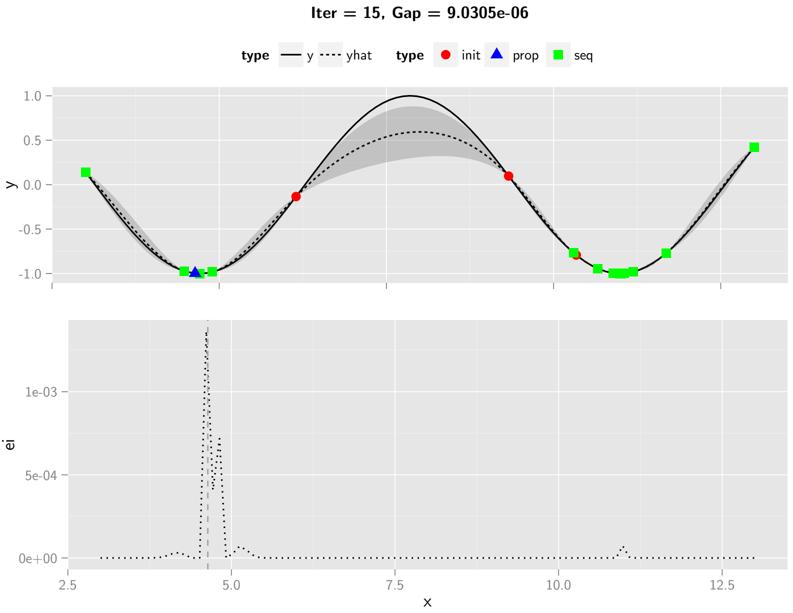

Iter = 2, Gap = 5.5410e-02
type
y
yhat
type
init
prop
seq
y
1.0
0.5
0.0
-0.5
-1.0
ei
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
x


Iter = 3, Gap = 5.5410e-02
type
y
yhat
type
init
prop
seq
1.0
0.5
0.0
-0.5
-1.0
y
0.12
0.09
0.06
0.03
0.00
ei
2.5
5.0
7.5
10.0
12.5
x


Iter = 4, Gap = 2.2202e-05
type
y
yhat
type
init
prop
seq
1.0
0.5
0.0
-0.5
-1.0
y
0.04
0.03
0.02
0.01
0.00
ei
2.5
5.0
7.5
10.0
12.5
x

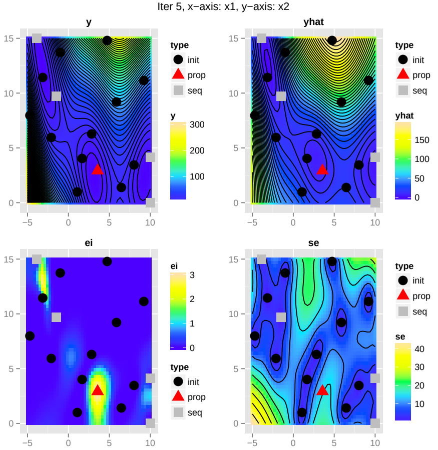


Iter = 5, Gap = 2.2202e-05
type
y
yhat
type
init
prop
seq
1.0
0.5
0.0
-0.5
-1.0
y
0.015
0.010
0.005
0.000
ei
2.5
5.0
7.5
10.0
12.5
x


Iter = 15, Gap = 9.0305e-06
type
y
yhat
type
init
prop
seq
1.0
0.5
0.0
-0.5
-1.0
y
1e-03
5e-04
0e+00
ei
2.5
5.0
7.5
10.0
12.5
x

mlrMBO: Model-Based Optimization Toolbox

- Any regression from mlr
- Arbtritrary infill
- Single - or multi-crit
- Multi-point proposal
- Via parallelMap and batchtools runs on many parallel backends and clusters
- Algorithm configuration
- Active research

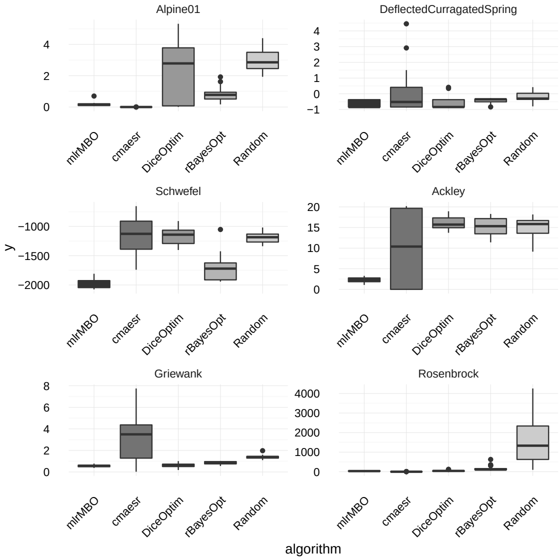


- mlr: `https://github.com/mlr-org/mlr`
- mlrMBO: `https://github.com/mlr-org/mlrMBO`
- mlrMBO Paper on arXiv (under review) `https://arxiv.org/abs/1703.03373`

Benchmark MBO on artificial test functions

- Comparison of mlrMBO on multiple different test functions
  - Multimodal
  - Smooth
  - Fully numeric
  - *Well known*
- We use GPs with
  - LCB with $\lambda = 1$
  - Focussearch
  - 200 iterations
  - 25 point initial design, created by LHS sampling
- Comparison with
  - Random search
  - CMAES
  - other MBO implementations in R

MBO GP vs. competitors in 5D



Section 2

Parallel batch proposals

Motivation for batch proposal

- Function evaluations expensive
- Often many cores available on a cluster
- Underlying f can in many cases not be easily parallelized
- Natural to consider batch proposal
- Parallel MBO: suggest q promising points to evaluate: $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_q^*$
- We need to balance exploration and exploitation
- Non-trivial to construct infill criterion for this

Review of parallel MBO strategies

- **Constant liar**: (Ginsbourger et al., 2010)
  - Fit kriging model based on real data and find $\boldsymbol{x}_1^*$ according to EI-criterion.
  - "Guess" $f(\boldsymbol{x}_{i-1}^*)$, update the model and find $\boldsymbol{x}_i^*$, $i = 2, ..., q$
  - Use f_{min} for "guessing"

- q-**LCB**: (Hutter et al., 2012)
  - q times: sample λ from $Exp(1)$ and optimize single LCB criterion
  - $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathrm{LCB}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \; \hat{f}(\boldsymbol{x}) - \lambda\hat{s}(\boldsymbol{x})\,.$

Multiobjectivization and proposed idea

- **Multiobjectivization**
  - Originates from multi-modal optimization
  - Add distance to neighbors for current set as artificial objective
  - Use multiobjective optimization
  - Select by hypervolume or first objective or ...

- **Our approach**
  - Decouple $\hat{f}(\boldsymbol{x})$ and $\hat{s}(\boldsymbol{x})$ as objectives – instead of EI – to have different exploration / exploitation trade-offs
  - Consider distance measure as potential extra objective
  - Run multiobjective EA to select q well-performing, diverse points
  - Distance is possible alternative if no or bad $\hat{s}(\boldsymbol{x})$ estimator
  - Decoupling $y(\boldsymbol{x})$, $\hat{s}(\boldsymbol{x})$ potential alternative when *EI* derivation does not hold for other model classes

Bischl, Wessing et al:*MOI-MBO: Multiobjective infill for parallel model-based optimization*, LION 2014

Experimental setup

Problem Instances

- All 24 test functions of the black-box optimization benchmark (BBOB) noise-free test suite
- Dimensions $d \in \{5, 10\}$

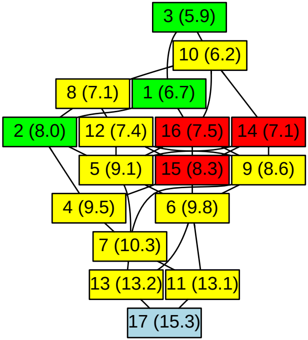
Budget

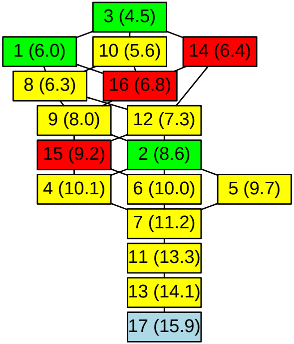
- For every function 10 initial designs of size $5 \cdot d$
  $\Rightarrow$ 10 statistical replications for each problem instance
- $40 \cdot d$ function evaluations on top of the initial design
- Parallel optimization: batches of size $q = 5$

Visualization: Preference relation graph

- Each node represents an approach (mean rank in braces)
- Two nodes are connected with an edge if one approach (the upper) is significantly better than the other (the lower) according to the sign test

Result Graphs





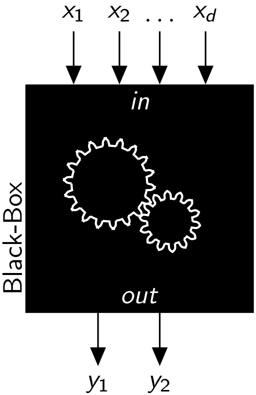
- 01: ego
- 02: ego_ea_ei
- 03: ego_ea_mean
- 04: moi_ei.dist_nb_first
- 05: moi_ei.dist_nb_hv
- 06: moi_ei.dist_nn_first
- 07: moi_ei.dist_nn_hv
- 08: moi_mean.se.dist_nb_first
- 09: moi_mean.se.dist_nb_hv

- 10: moi_mean.se.dist_nn_first
- 11: moi_mean.se.dist_nn_hv
- 12: moi_mean.se_nn_first
- 13: moi_mean.se_nn_hv
- 14: par_cl
- 15: par_cl_ea
- 16: par_lcb
- 17: random_search

Section 3

Multicriteria SMBO

Model-based multi-objective optimization



$$\min_{\boldsymbol{x} \in \mathbb{X}} \mathbf{f}(\boldsymbol{x}) = \boldsymbol{y} = (y_1, ..., y_m) \text{ with } \mathbf{f} : \mathbb{R}^d \to \mathbb{R}^m$$

- $\boldsymbol{y}$ *dominates* $\tilde{\boldsymbol{y}}$ if (3)

$$\forall i \in \{1, ..., m\} : y_i \leq \tilde{y}_i \quad (4)$$

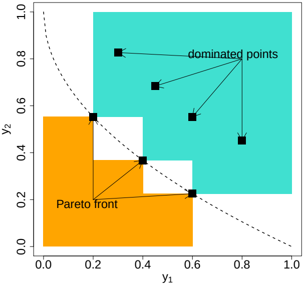
$$\text{and } \exists i \in \{1, ..., m\} : y_i < \tilde{y}_i \quad (5)$$

- Set of non-dominated solutions:

  $\mathcal{X}^* := \{\mathbf{x} \in \mathcal{X} | \nexists \tilde{\mathbf{x}} \in \mathcal{X} : \mathbf{f}(\tilde{\mathbf{x}}) \text{ dominates } \mathbf{f}(\mathbf{x})\}$

- Pareto set $\mathcal{X}^*$, Pareto front $\mathbf{f}(\mathcal{X}^*)$
- Goal: Find $\hat{\mathcal{X}}^*$ of non-dominated points that estimates the true set $\mathcal{X}^*$

Model-based multi-objective optimization



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

- $\boldsymbol{y}$ *dominates* $\tilde{\boldsymbol{y}}$ if (6)

$$\forall i \in \{1, \ldots, m\} : y_i \leq \tilde{y}_i \tag{7}$$

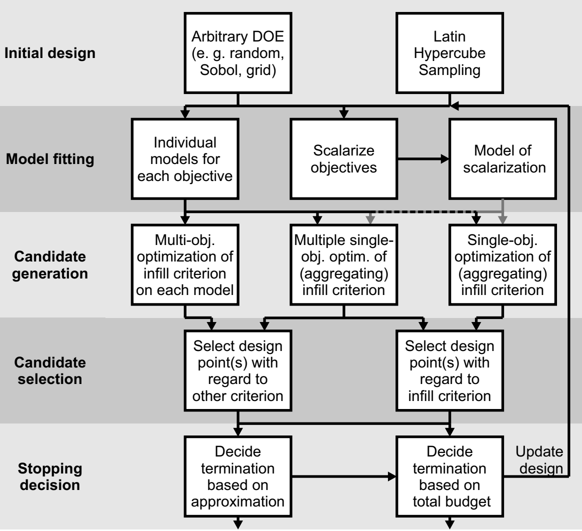
$$\text{and } \exists i \in \{1, \ldots, m\} : y_i < \tilde{y}_i \tag{8}$$

- Set of non-dominated solutions:

$$\mathcal{X}^* := \{\mathbf{x} \in \mathcal{X} | \nexists \tilde{\mathbf{x}} \in \mathcal{X} : \mathbf{f}(\tilde{\mathbf{x}}) \text{ dominates } \mathbf{f}(\mathbf{x})\}$$

- Pareto set $\mathcal{X}^*$, Pareto front $\mathbf{f}(\mathcal{X}^*)$
- Goal: Find $\hat{\mathcal{X}}^*$ of non-dominated points that estimates the true set $\mathcal{X}^*$

Taxonomy



Horn, Wagner, Bischl et al:*Model-based multi-objective optimization: Taxonomy, multi-point proposal, toolbox and benchmark*, EMO 2014

Batch Proposal

- Most MBMO lack way to propose $N > 1$ points (batch evaluation)
- Batch evaluations are essential for distributed computing
- We integrated such mechanism(s) for arbitrary MBMO
- Replaced single phases of the taxonomy

ParEGO

1. Scalarize objectives using the augmented Tchebycheff norm

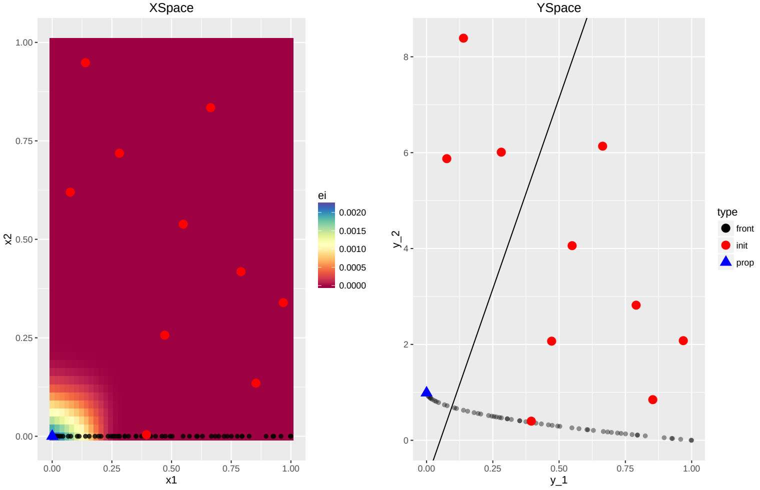
$$\max_{i=1,\ldots,d} [w_i f_i(\mathbf{x})] + \rho \sum_{i=1}^{d} w_i f_i(\mathbf{x})$$

   with uniformly distributed weight vector $\mathbf{w}$ ($\sum w_i = 1$) and fit surrogate model to the respective scalarization.
2. Single-objective optimization of EI (or LCB?)

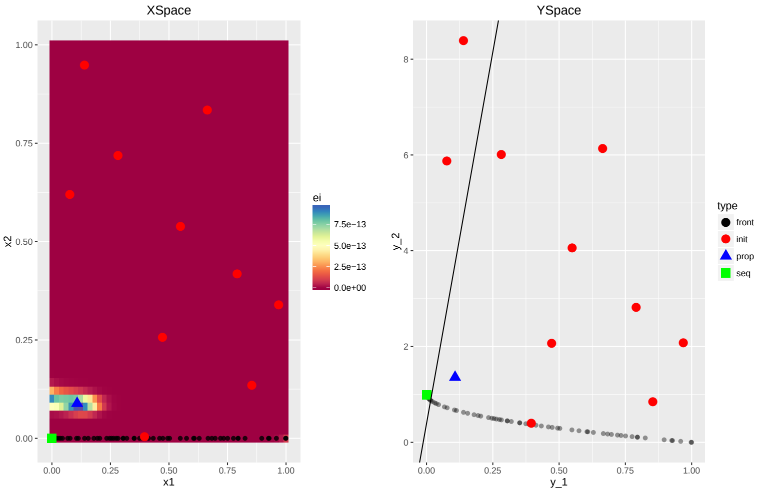
Batch proposal: **Increase the number and diversity of randomly drawn weight vectors**

- If N points are desired, cN ($c > 1$) weight vectors are considered
- Greedily reduce set of weight vectors by excluding one vector of the pair with minimum distance
- Scalarizations implied by each weight vector are computed
- Fit and optimize models for each scalarization
- Optima of each model build the batch to be evaluated

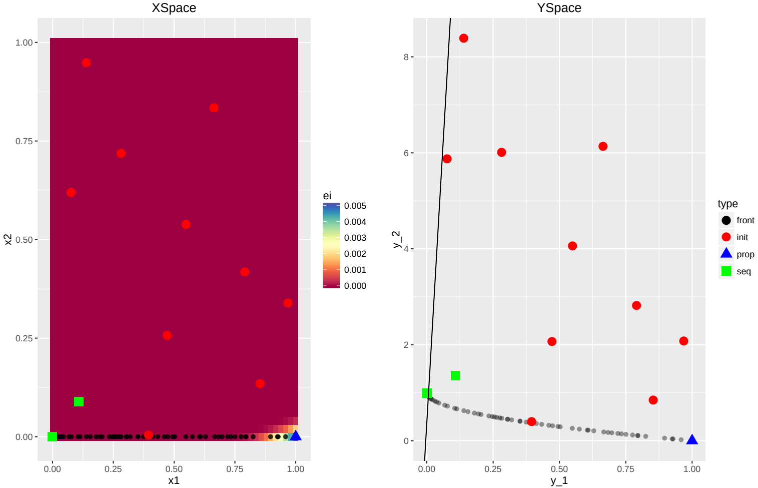
Animation of ParEGO



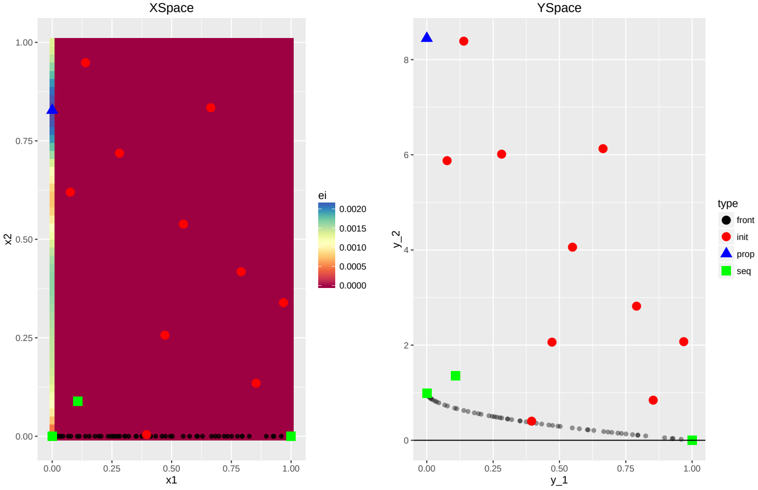
Animation of ParEGO



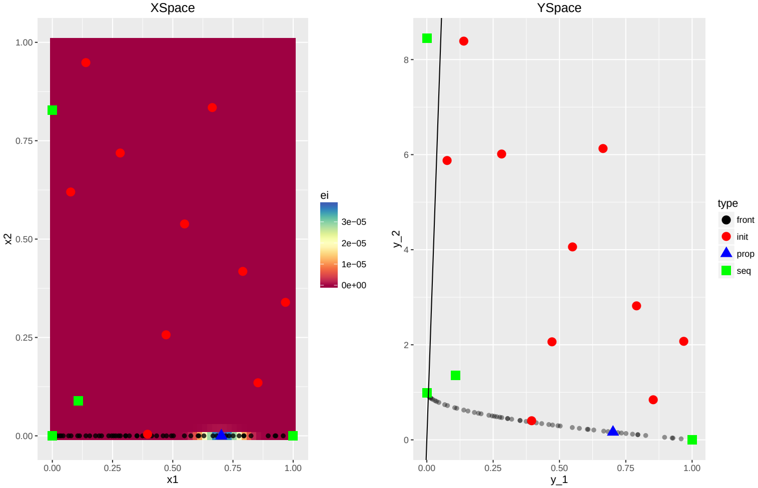
Animation of ParEGO



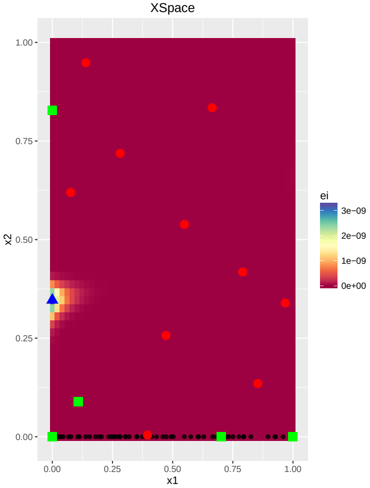
Animation of ParEGO

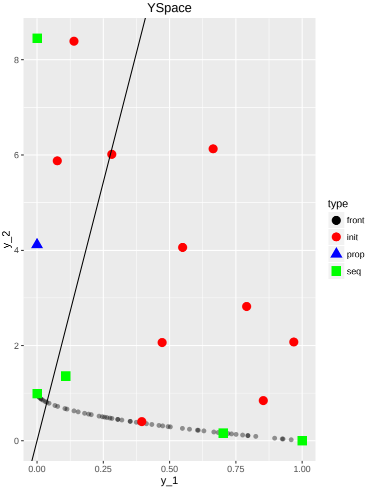


Animation of ParEGO

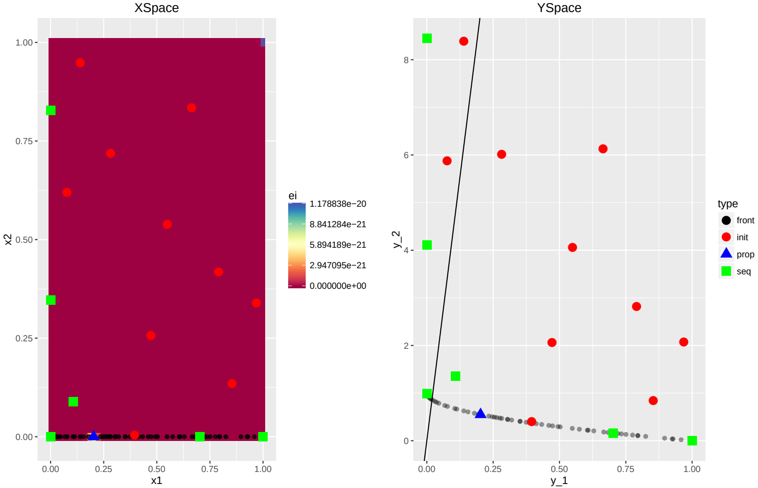


Animation of ParEGO

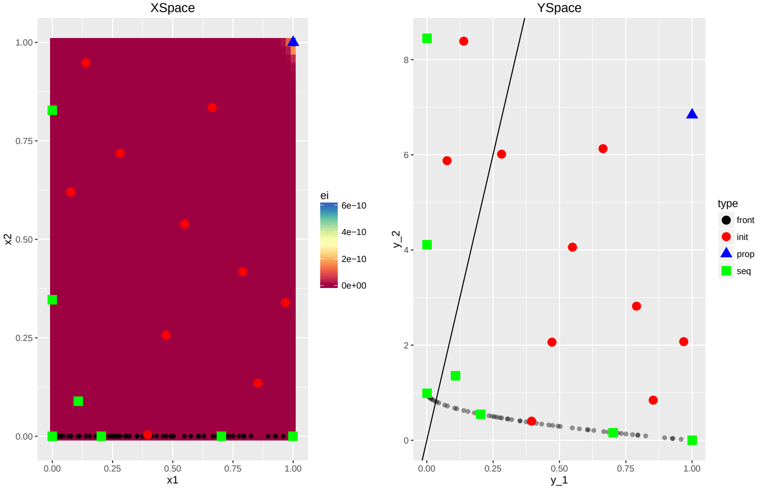




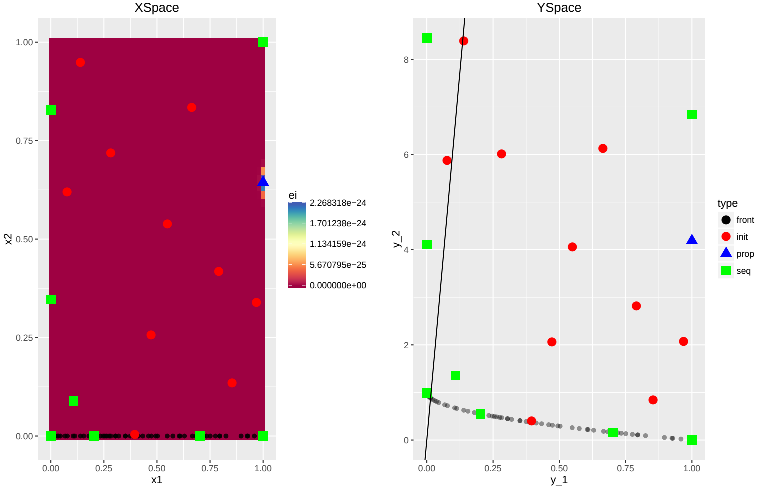
Animation of ParEGO



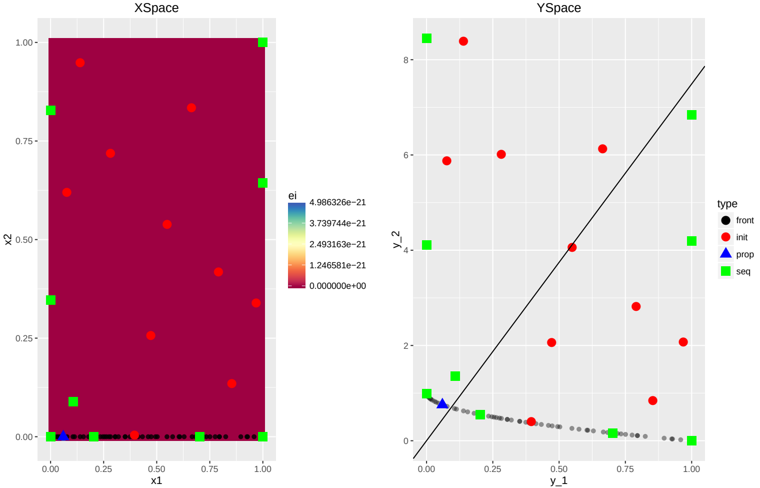
Animation of ParEGO



Animation of ParEGO



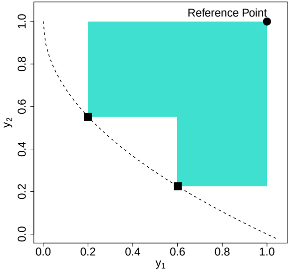
Animation of ParEGO



SMS-EGO

- Individual models for each objective
- Single-objective optimization of aggregating infill criterion: Calculate contribution of the confidence bound of representative solution to the current front approximation

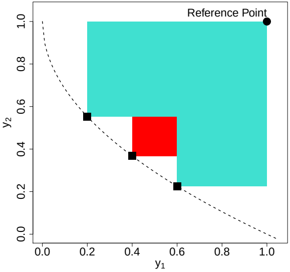
- Calculate LCB for each objective
- Measure contribution with regard to the hypervolume indicator
- For ε-dominated ($\preceq_\varepsilon$) solutions, a penalty $\Psi(\mathbf{x}) = -1 + \prod_{j=1}^{m} \left(1 + (l(\mathbf{x}) - y_j^{(i)})\right)$ is added (Actually not needed for Focussearch.)



SMS-EGO

- Individual models for each objective
- Single-objective optimization of aggregating infill criterion: Calculate contribution of the confidence bound of representative solution to the current front approximation

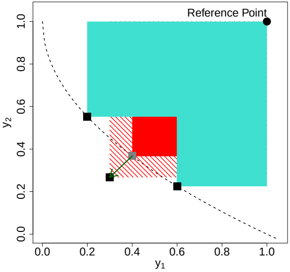
- Calculate LCB for each objective
- Measure contribution with regard to the hypervolume indicator
- For ε-dominated ($\preceq_\varepsilon$) solutions, a penalty $\Psi(\mathbf{x}) = -1 + \prod_{j=1}^{m} \left(1 + (l(\mathbf{x}) - y_j^{(i)})\right)$ is added (Actually not needed for Focussearch.)



SMS-EGO

- Individual models for each objective
- Single-objective optimization of aggregating infill criterion: Calculate contribution of the confidence bound of representative solution to the current front approximation

- Calculate LCB for each objective
- Measure contribution with regard to the hypervolume indicator
- For ε-dominated ($\preceq_\varepsilon$) solutions, a penalty $\Psi(\mathbf{x}) = -1 + \prod_{j=1}^{m} \left(1 + (l(\mathbf{x}) - y_j^{(i)})\right)$ is added (Actually not needed for Focussearch.)



SMS-EGO: Batch proposal

Modification of phase candidate generation: **Use simulated evaluations for candidate generation**

- The proposed point $\vec{x}^*$ is not directly evaluated, but the LCB $l(\vec{x}^*)$ is added to the current approximation without refitting the model
- Repeat until N points for a batch evaluation have been found

Approximative RBF-SVM Training Algorithms

SVM solver	Description
Pegasos	Stochastic Gradient Descent
BSGD	Budgeted Stochastic Gradient Descent
LLSVM	Low-rank kernel approximation + linear solver
LIBSVM	"Exact" SMO solver
LASVM	Online variant of SMO solver
LIBBVM/CVM	Minimum Enclosing Ball (only squared hinge loss)
SVMperf	Cutting Plane Algorithm

- What is the trade-off between training time and prediction error?
- Most solvers have 2 additional parameters on top of C and γ
- Optimizing 2 expensive objectives in a 4-dim parameter space.
- Replace grid search with more sophisticated PAREGO-algorithm.

Approximative SVM Training Algorithms

- **We expect:** Every solver has a trade-off between training time and prediction error: Given more time a solver (should) reach a better solution.
- **Our goal:** Analyze this trade-off! Solve the multi-criteria optimization problem with respect to the two objectives error and training time by varying the parameters.
- **The challenge:** Optimizing 2 expensive objectives in a 4-dimensional parameter space.
- **Our approach:** Replace standard grid search with more sophisticated PAREGO-algorithm.

Approximative SVM Training Algorithms

The parameters (C, γ) of the SVM itself were optimized over $2^{[-15,15]}$ respectively. Every solver has further approximation parameters:

SVM solver	Parameters	Optimization Space
Pegasos	#Epochs	$2^{[0,7]}$
BGSD	Budget size, #Epochs	$2^{[4,11]}$, $2^{[0,7]}$
LLSVM	Matrix rank	$2^{[4,11]}$
LIBSVM	ϵ (Accuracy)	$2^{[-13,-1]}$
LASVM	ϵ (Accuracy), #Epochs	$2^{[-13,-1]}$, $2^{[0,7]}$
LIBBVM/CVM	ϵ (Accuracy)	$2^{[-19,-1]}$
SVMperf	ϵ (Accuracy), #Cutting planes	$2^{[-13,-1]}$, $2^{[4,11]}$

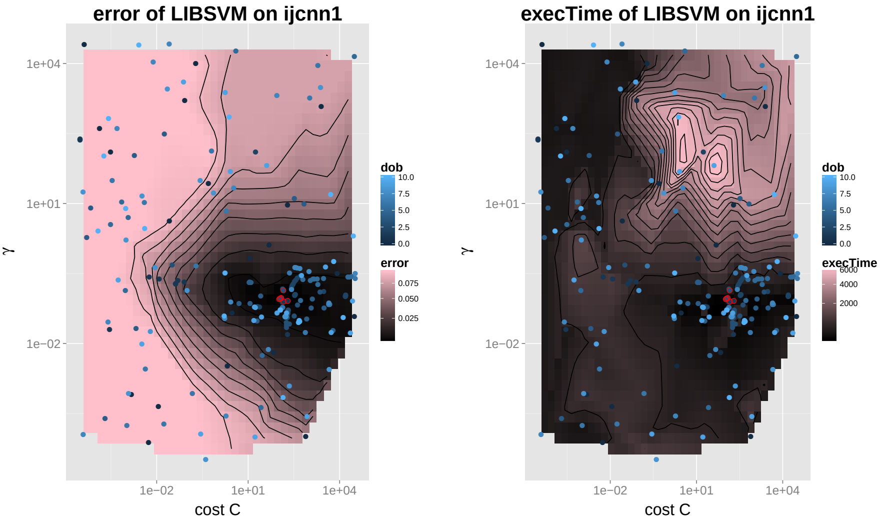
Additional parameters set to default values.

Datasets

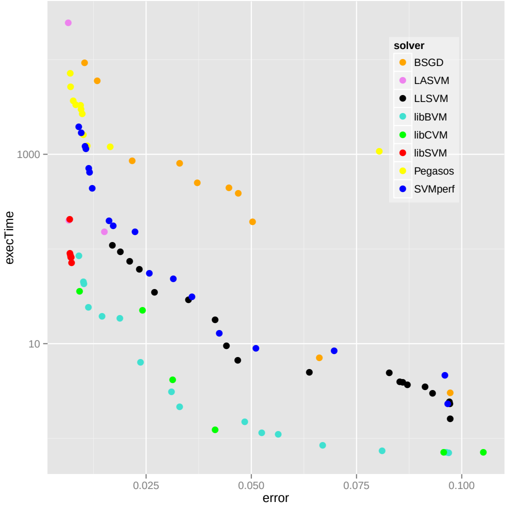
data set	# points	# features	class ratio	sparsity
wXa	34 780	300	34.45	95.19 %
aXa	36 974	123	3.17	88.72 %
protein	42 153	357	1.16	71.46 %
mnist	70 000	780	0.96	80.76 %
vehicle	98 528	100	1.00	0 %
shuttle	101 500	9	0.27	0.23 %
spektren	175 090	22	0.80	0 %
ijcnn1	176 691	22	9.41	40.91 %
arthrosis	262 142	178	1.19	0.01 %
cod-rna	488 565	8	2.00	0.02 %
covtype	581 012	54	1.05	78 %
poker	1 025 010	10	1.00	0 %

Table: Overview of the data sets.

Test error landscape (LIBSVM)



All Pareto fronts for ijcnn1 dataset



- 176 691 samples
  22 features
  9.41 class ratio
- Wee see:
  LIBSVM: exact but slow
  LIBBVM/CVM: good front - speed increase with small accuracy loss
  SVMperf / LLSVM / BSGD: can be really fast, but higher accuracy loss
  LASVM / Pegasos: less exact and even slower as LIBSVM

Section 4

Interesting Challenges

Challenge: The correct surrogate?

- GPs are very much tailored to what we want to do, due to their spatial structure in the kernel and the uncertainty estimator.
- But GPs are rather slow. And (fortunately) due to parallization (or speed-up tricks like subsampling) we have more design points to train on.
- Categorical features are also a problem in GPs (although methods exist, usually by changing the kernel)
- Random Forests handle categorical features nicely, are much faster. But they don't rely on a spatial kernel and the uncertainty estimation is much more heuristic / may not represent what we want.

Challenge: Time Heterogeneity

- Complex configuration spaces across many algorithms results in vastly different runtimes in design points.
- Actually just the RBF-SVM tuning can result in very different runtimes.
- We don't care how many points we evaluate, we care about total walltime of the configuration.
- The option to subsample further complicates things.
- Parallelization further complicates things.
- Option: Estimate runtime as well with a surrogate, integrate it into acquisition function.

Section 5

ML Model Selection and Hyperparameter Optimization

Automatic Model Selection

Prior approaches:

- Looking for the silver bullet model
  $\rightsquigarrow$ Failure
- Exhaustive benchmarking / search
  $\rightsquigarrow$ Very expensive, often contradicting results
- Meta-Learning:
  $\rightsquigarrow$ Good meta-features are hard to construct
  $\rightsquigarrow$ IMHO: Gets more interesting when combined with SMBO

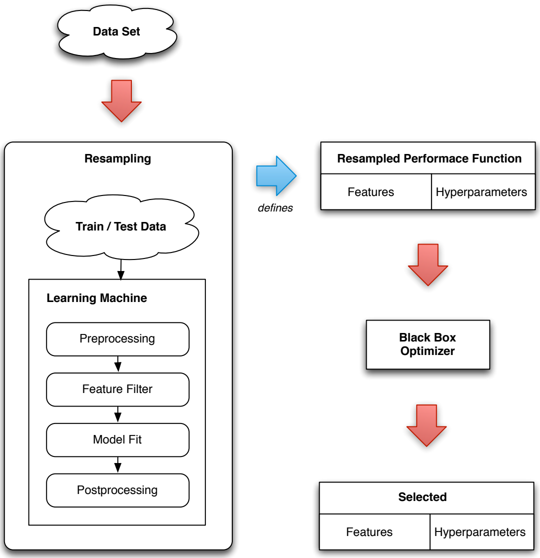
Goal for AutoML:

- Data dependent
- Automatic
- Include every relevant modeling decision
- Efficient
- Learn on the model-settings level!

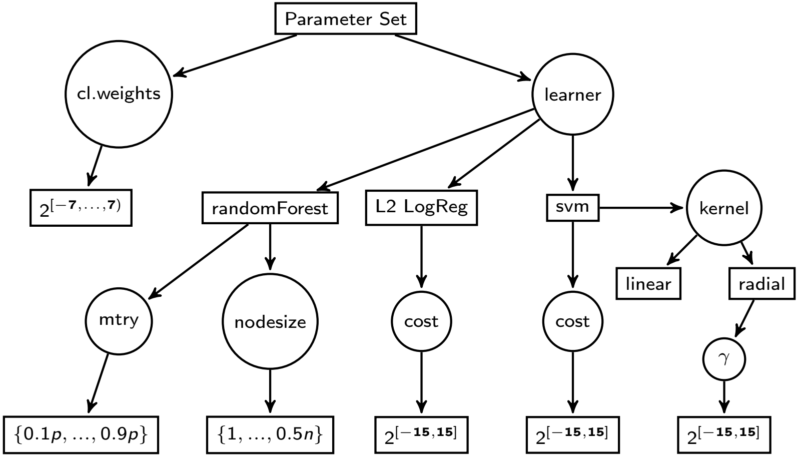
From Normal SMBO to Hyperarameter Tuning

- Objective function is resampled performance measure
- Parameter space $\theta \in \Theta$
  might be discrete and dependent / hierarchical
- No derivative for $f(\cdot, \theta)$, black-box
- Objective is stochastic / noisy
- Objective is expensive to evaluate
- In general we face a problem of algorithm configuration:
- $\rightsquigarrow$ Usual approaches: racing or model-based / bayesian optimization

From Normal SMBO to Hyperarameter Tuning



Complex Parameter Space



From Normal SMBO to Hyperarameter Tuning

- Initial design: LHS principle can be extended, or just use random
- Focus search: Can be (easily) extended, as it is based on random search. To zoom in for categorical parameters we randomly drop a category for each param which is not present in the currently best configuration.

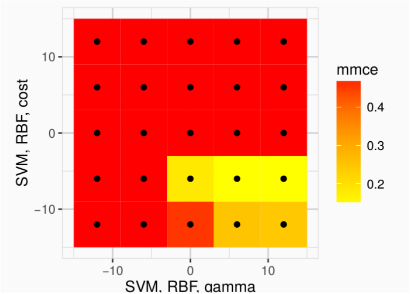
- Few approaches for GPs with categorical params exist (usually with new covar kernels), not very established
- Alternative: Random regression forest (mlrMBO, SMAC)
- Estimate uncertainty / confidence interval for mean response by efficient bootstrap technique[1], or jackknife, so we can define $EI(x)$ for the RF
- Dependent params in mlrMBO: Imputation:
- Many of the current techniques to handle these problems are (from a theoretical standpoint) somewhat crude

[1]Sexton et al, "Standard errors for bagged and random forest estimators, 2009."

Hyperparameter Tuning

- Still common practice: grid seach
  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 beacause:
  - optimum might be "off the grid"
  - lots of evaluations in bad areas
  - lots of costy evaluations
- How bad?

Hyperparameter Tuning

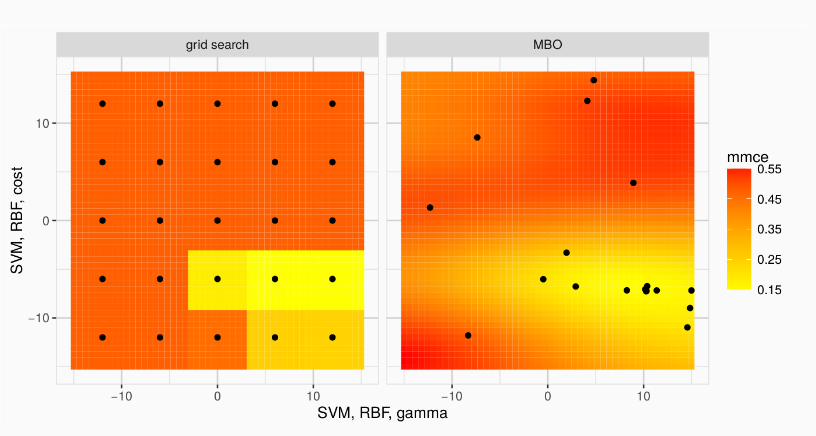


- Because of budget restrictions grid might even be smaller!
- Unpromising area quite big!
- Lots of costly evaluations!

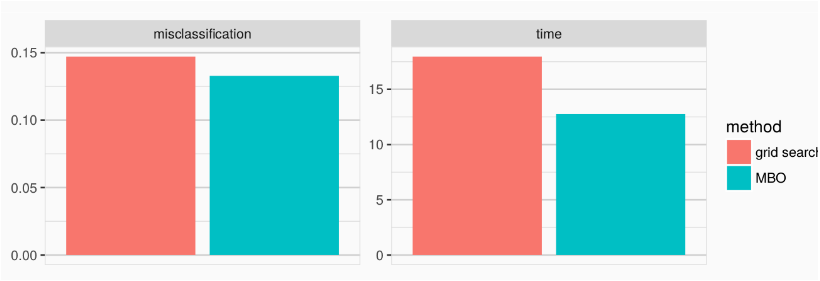
With **mlrMBO** it is not hard to do it better!
More interesting applications to time-series regression and cost-sensitive classification[2]

[2] Koch, Bischl et al: *Tuning and evolution of support vector kernels*, EI 2012

Hyperparameter Tuning



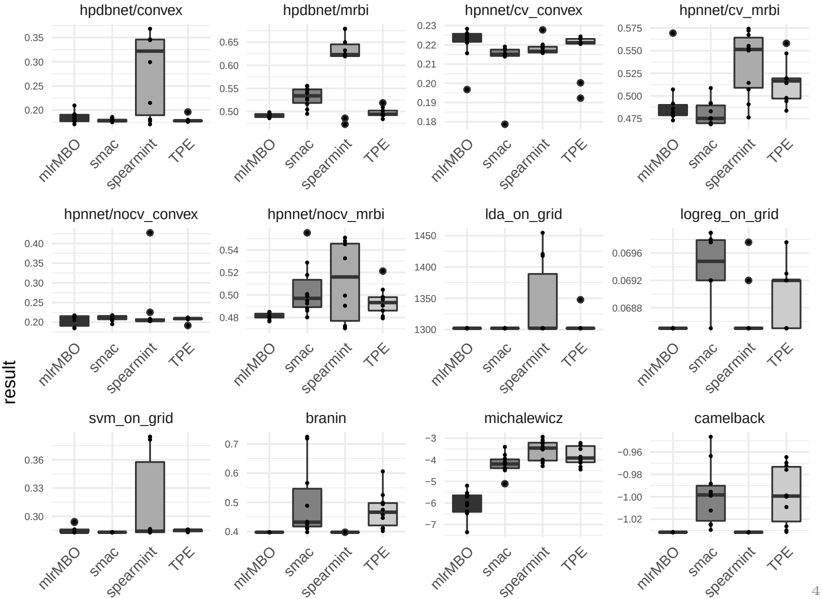
Hyperparameter Tuning



HPOlib

- HPOlib is a set of standard benchmarks for hyperparameter optimizer
- Allows comparison with
  - Spearmint
  - SMAC
  - Hyperopt (TPE)
- Benchmarks:
  - Numeric test functions (similar to the ones we've seen bevor)
  - Numeric machine learning problems (lda, SVM, logistic regression)
  - Deep neural networks and deep belief networks with 15 and 35 parameters.
- For benchmarks with discrete and dependent parameters (hpnnet, hpdbnet) a random forest with standard error estimation is used.

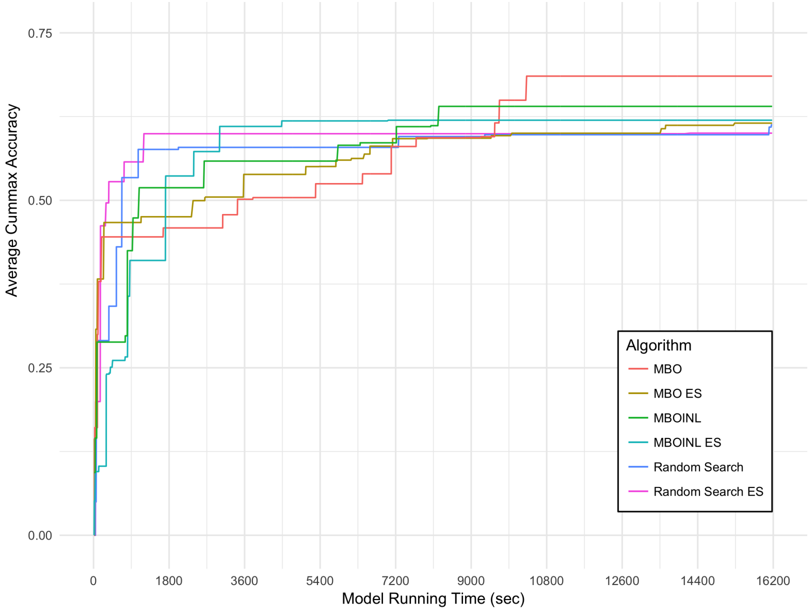
MBO: HPOLIB


result
hpdbnet/convex
hpdbnet/mrbi
hpnnet/cv_convex
hpnnet/cv_mrbi
hpnnet/nocv_convex
hpnnet/nocv_mrbi
lda_on_grid
logreg_on_grid
svm_on_grid
branin
michalewicz
camelback
mlrMBO
smac
spearmint
TPE

Deep Learning Configuration Example

- Dataset: CIFAR-10 (60000 32x32 images with 3 color channels; 10 classes)
- Configuration of a deep neural network (mxnet)
- Size of parameter set: 30, including number of hidden layers, activation functions, regularization, convolution layer setting, etc.
- Split: 2/3 training set, 1/6 test set, 1/6 validation set
- Time budget per tuning run: 4.5h (16200 sec)
- Surrogate: Random forest
- Acquisition: LCB with $\lambda = 2$

Deep Learning Configuration Example



Thanks! Any comments or questions?