

# AutoML: Bayesian Optimization for HPO

Extensions of Bayesian Optimization

Bernd Bischl   Frank Hutter   Lars Kotthoff  
Marius Lindauer   Joaquin Vanschoren

# Beyond the Standard Bayesian Optimization Setting

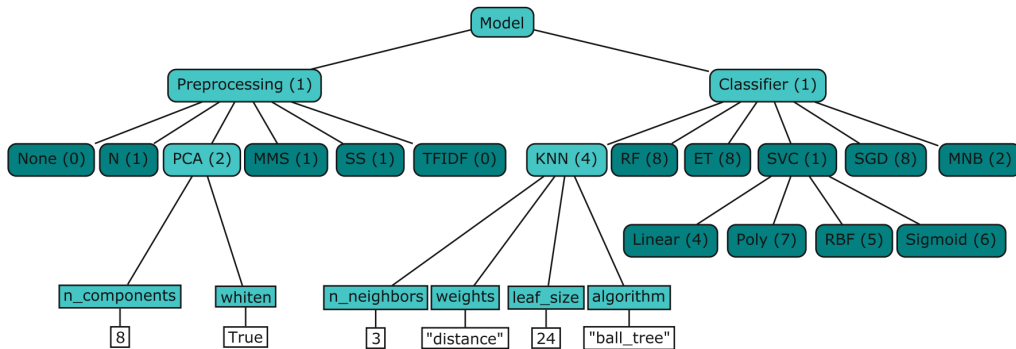
## Standard Bayesian optimization problems

- Low-dimensional functions
- Continuous, smooth functions
- Sequential optimization

## Extensions

- Structured search spaces: categorical & conditional hyperparameters
- High dimensions
- Parallel evaluations
- Optimization with constraints

# Structured Search Spaces: Categorical & Conditional Hyperparameters

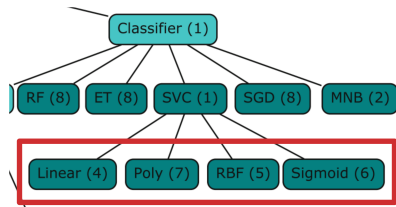


Example of a structured search space [Hutter et al. 2019]

# Structured Search Spaces: Categorical Hyperparameters

Properties of categorical hyperparameters:

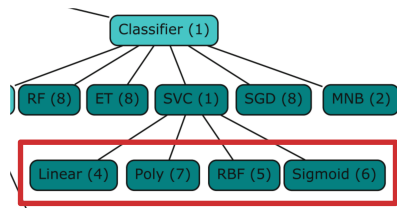
- Finite, discrete set of values
- No natural order between values
- Potentially different distances between values



# Structured Search Spaces: Categorical Hyperparameters

Properties of categorical hyperparameters:

- Finite, discrete set of values
- No natural order between values
- Potentially different distances between values



This has to be taken into account by the surrogate model:

- Random Forests *natively* handle categorical inputs [Hutter et al, 2011]
- *One-hot* encoding provides a simple general solution
- Gaussian Processes can use a (weighted) **Hamming Distance Kernel** [Hutter. 2009]:

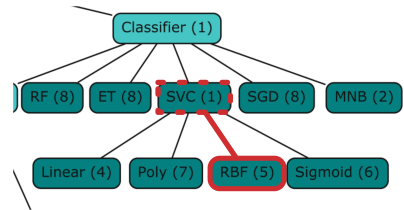
$$\kappa_{\theta}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j) = \exp \sum_{l=1}^d (-\theta \cdot \delta(\lambda_{i,l} \neq \lambda_{j,l}))$$

- Neural networks can learn **entity embeddings** for categorical inputs [Guc and Berkhahn. 2016]

# Structured Search Spaces: Conditional Hyperparameters

Conditional hyperparameters:

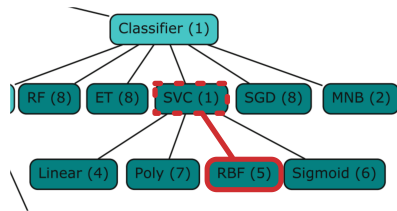
- Are **only relevant** if certain other hyperparameters take on certain values
- **Should be ignored** by the model **if not active**



# Structured Search Spaces: Conditional Hyperparameters

Conditional hyperparameters:

- Are **only relevant** if certain other hyperparameters take on certain values
- **Should be ignored** by the model if not active



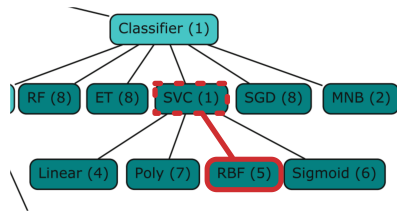
Modelling conditional hyperparameters:

- Setting the values for inactive hyperparameter to a specific value (e.g. 0)
- Random Forests [Hutter et al. 2011] and Tree Parzen Estimators [Bergstra et al. 2011] can **natively** handle conditional inputs
- There exist **several kernels for Gaussian Processes** to handle conditional inputs [Hutter and Osborne. 2013; Lévesque et al. 2017; Jenatton et al. 2017]

# Structured Search Spaces: Conditional Hyperparameters

Conditional hyperparameters:

- Are **only relevant** if certain other hyperparameters take on certain values
- **Should be ignored** by the model **if not active**



Modelling conditional hyperparameters:

- Setting the values for inactive hyperparameter to a specific value (e.g. 0)
- Random Forests [Hutter et al. 2011] and Tree Parzen Estimators [Bergstra et al. 2011] can **natively** handle conditional inputs
- There exist **several kernels for Gaussian Processes** to handle conditional inputs [Hutter and Osborne. 2013; Lévesque et al. 2017; Jenatton et al. 2017]

Overall, structured search spaces are **still an active research topic** and far from solved



# High Dimensions

- Issues

- ▶ Standard Gaussian processes  
do not tend to fit well in high dimensions
- ▶ Maximizing the acquisition function  
is computationally challenging

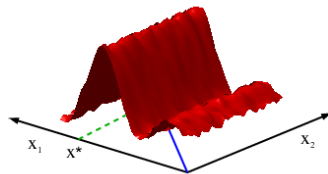
# High Dimensions

- Issues

- ▶ Standard Gaussian processes do not tend to fit well in high dimensions
- ▶ Maximizing the acquisition function is computationally challenging

- There is still hope

- ▶ Many optimization problems have *low effective dimensionality*
- ▶ Not all dimensions interact with each other



# High Dimensions

- Issues

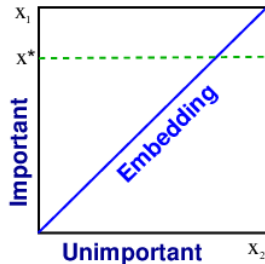
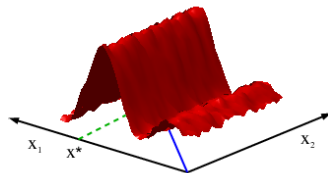
- ▶ Standard Gaussian processes do not tend to fit well in high dimensions
- ▶ Maximizing the acquisition function is computationally challenging

- There is still hope

- ▶ Many optimization problems have *low effective dimensionality*
- ▶ Not all dimensions interact with each other

- Possible solutions

- ▶ Optimize in a lower-dimensional embedding [Wang et al. 2016]
- ▶ Fit additive models on subsets of dimensions [Kandasamy et al. 2015]
- ▶ Use other models; e.g., random forests



# Parallel Bayesian Optimization: Multi-point Acquisition Functions

- Often, we have many parallel compute units
- How should these be exploited in (the typically inherently sequential) Bayesian optimization?

# Parallel Bayesian Optimization: Multi-point Acquisition Functions

- Often, we have many parallel compute units
- How should these be exploited in (the typically inherently sequential) Bayesian optimization?
- To select a batch of  $q$  points in parallel, we need to compute the multi-point acquisition function. E.g., for expected improvement:

$$q\text{-EI}(\boldsymbol{\lambda}_{1,\dots,q}) = \mathbb{E} \left[ c(\hat{\boldsymbol{\lambda}}^{(t)}) - \min_{i=1,\dots,q} \hat{c}(\boldsymbol{\lambda}_i) \right]$$

# Parallel Bayesian Optimization: Multi-point Acquisition Functions

- Often, we have many parallel compute units
- How should these be exploited in (the typically inherently sequential) Bayesian optimization?
- To select a batch of  $q$  points in parallel, we need to compute the multi-point acquisition function. E.g., for expected improvement:

$$q\text{-EI}(\boldsymbol{\lambda}_{1,\dots,q}) = \mathbb{E} \left[ c(\hat{\boldsymbol{\lambda}}^{(t)}) - \min_{i=1,\dots,q} \hat{c}(\boldsymbol{\lambda}_i) \right]$$

- For EI and KG, this requires *expensive-to-compute*  $q$ -dimensional Gaussian cumulative distributions [Ginsbourger et al. 2008; Wu and Frazier. 2018; Wang et al. 2019]

# Parallel Bayesian Optimization: Multi-point Acquisition Functions

- Often, we have many parallel compute units
- How should these be exploited in (the typically inherently sequential) Bayesian optimization?
- To select a batch of  $q$  points in parallel, we need to compute the multi-point acquisition function. E.g., for expected improvement:

$$q\text{-EI}(\boldsymbol{\lambda}_{1,\dots,q}) = \mathbb{E} \left[ c(\hat{\boldsymbol{\lambda}}^{(t)}) - \min_{i=1,\dots,q} \hat{c}(\boldsymbol{\lambda}_i) \right]$$

- For EI and KG, this requires *expensive-to-compute*  $q$ -dimensional Gaussian cumulative distributions [Ginsbourger et al. 2008; Wu and Frazier. 2018; Wang et al. 2019]
- Nevertheless, multi-point acquisition functions can be optimized efficiently with gradient descent via the reparameterization trick [Wilson et al. 2018]

# Asynchronous Parallel Bayesian Optimization with Pending Evaluations

- In practice, typically, not all function evaluations take the same amount of time
  - ▶ Thus, we need to select **some new points** while we're still waiting for **pending evaluations** at other points



# Asynchronous Parallel Bayesian Optimization with Pending Evaluations

- In practice, typically, not all function evaluations take the same amount of time
  - ▶ Thus, we need to select **some new points** while we're still waiting for **pending evaluations** at other points
- Simple solution: **hallucinate observations for pending evaluations**, and use otherwise standard methods:

# Asynchronous Parallel Bayesian Optimization with Pending Evaluations

- In practice, typically, not all function evaluations take the same amount of time
  - ▶ Thus, we need to select **some new points** while we're still waiting for **pending evaluations** at other points
- Simple solution: **hallucinate observations for pending evaluations**, and use otherwise standard methods:
  - ▶ **Constant Liar**: Choose a fixed value (constant) [Ginsbourger et al. 2010]
  - ▶ **Kriging Believer**: Use the current mean prediction (belief) [Ginsbourger et al. 2010]
  - ▶ **Monte Carlo Fantasies**

# Asynchronous Parallel Bayesian Optimization with Pending Evaluations

- In practice, typically, not all function evaluations take the same amount of time
  - ▶ Thus, we need to select **some new points** while we're still waiting for **pending evaluations** at other points
- Simple solution: **hallucinate observations for pending evaluations**, and use otherwise standard methods:
  - ▶ **Constant Liar**: Choose a fixed value (constant) [Ginsbourger et al. 2010]
  - ▶ **Kriging Believer**: Use the current mean prediction (belief) [Ginsbourger et al. 2010]
  - ▶ **Monte Carlo Fantasies**
    - ★ Sample pending evaluations from the model
    - ★ Update copy of the model with these samples
    - ★ Compute acquisition function under each updated copy
    - ★ Define acquisition function as an average over these sampled acquisition functions

# Bayesian Optimization with Constraints

## Several types of constraints

- 1 **Known constraints:**  
can be accounted for when optimizing  $u$
- 2 **Hidden constraints:** no function value is observed due to a failed function evaluation [Lee et al. 2010]
- 3 **Unknown constraints:** there's an additional, but unknown constraint function (e.g., memory used), which can be observed and modeled

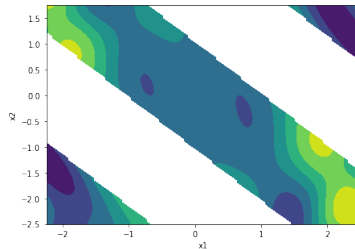


Image source: [GPFlowOpt Tutorial,  
Apache 2 License]

# Bayesian Optimization with Constraints

Several types of constraints

- 1 **Known constraints:**  
can be accounted for when optimizing  $u$
- 2 **Hidden constraints:** no function value is observed due to a failed function evaluation [Lee et al. 2010]
- 3 **Unknown constraints:** there's an additional, but unknown constraint function (e.g., memory used), which can be observed and modeled

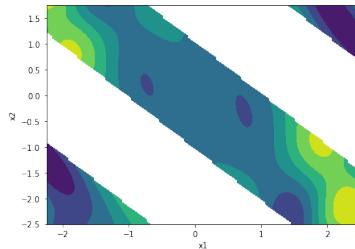


Image source: [GPFlowOpt Tutorial, Apache 2 License]

Most general solution: **Expected Constrained Improvement** [Lee et al. 2010]:

$$ECI(\lambda) = EI(\lambda)h(\lambda),$$

where  $h(\lambda)$  is the probability that  $\lambda$  is a valid configuration.

Further literature in [Frazier. 2018] and [Feurer and Hutter 2019].

## Even more extensions

Bayesian optimization has been extended to numerous scenarios:

- Multi-task, Multi-fidelity and Meta-learning → separate lecture
- Multi-objective Bayesian optimization → separate lecture
- Bayesian optimization with safety guarantees [Sui et al. 2015]
- Directly optimizing for ensemble performance [Lévesque et al. 2016]
- Combination with local search methods [Taddy et al. 2009; Eriksson et al. 2019]
- Optimization of arbitrary spaces that can be described by a kernel (e.g., neural network architectures [Kandasamy et al. 2018] or molecules [Griffiths and Hernández-Lobato. 2017])
- Many more (too many to mention)

## Questions to Answer for Yourself / Discuss with Friends

- **Discussion.** What would happen if you treat a categorical hyperparameter as continuous (e.g.,  $\{A, B, C\}$  as  $\{0, 0.5, 1\}$ ), in Bayesian optimization using a Gaussian Process?
- **Repetition.** Which methods can you use to impute values for outstanding evaluations? What are advantages and disadvantages of each method?
- **Discussion.** What are worst case scenarios that could happen if you ignore the noise during Bayesian optimization?