Using Meta Learning to Initialize Bayesian Optimization

Matthias Feurer\textsuperscript{1}  Jost Tobias Springenberg\textsuperscript{2}  Frank Hutter\textsuperscript{1}

\textsuperscript{1}Research Group on Learning, Optimization, and Automated Algorithm Design
\textsuperscript{2}Machine Learning Lab
Department of Computer Science, University of Freiburg, Germany
ECAI-2014 Workshop on Meta-learning & Algorithm Selection, 19 August 2014
Your task: Build an Iris classification system

Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM.

Your task: Build an Iris classification system

Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM.
Your task: Build an Iris classification system

- Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM
- Manual tuning -> fiddling with hyperparameters.

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0.
Your task: Build an Iris classification system

- Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM
- Manual tuning -> fiddling with hyperparameters.
- Better: Use automated methods like PSO, GA or SMBO

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0.
Your task: Build an Iris classification system

- Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM
- Manual tuning -> fiddling with hyperparameters.
- Better: Use automated methods like PSO, GA or SMBO
- Best: AutoWeka

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0.
Adding the Iris Japonica to the dataset

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0; Bottom right: Iris Japonica is licensed by KENPEI under CC BY-SA 3.0
Adding the Iris Japonica to the dataset

- Manual tuning:
  Use experience and start from the parameters found on the Iris dataset

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0; Bottom right: Iris Japonica is licensed by KENPEI under CC BY-SA 3.0
Adding the Iris Japonica to the dataset

- **Manual tuning:**
  Use experience and start from the parameters found on the Iris dataset

- **Automated methods**
  -> start from scratch

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0; Bottom right: Iris Japonica is licensed by KENPEI under CC BY-SA 3.0
Adding the Iris Japonica to the dataset

- Manual tuning:
  Use experience and start from the parameters found on the Iris dataset

- Automated methods
  -> start from scratch

- Cast use experience into an algorithm.

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0; Bottom right: Iris Japonica is licensed by KENPEI under CC BY-SA 3.0
Sequential Model-based Bayesian Optimization (SMBO)

ML Algorithm $A$

Configuration Space $\Lambda$ of $A$

Dataset $D$
Sequential Model-based Bayesian Optimization (SMBO)

ML Algorithm $A$

Configuration Space $\Lambda$ of $A$

Dataset $D$

Configuration Task

Configuration $\lambda^*$
Sequential Model-based Bayesian Optimization (SMBO)

1. **ML Algorithm** $A$
2. **Configuration Space $\Lambda$ of $A$**
3. **Dataset $D$**

**Configuration Task**

- **Fit regression model on** $(\lambda, A_{\lambda}(D))$ **pairs**
- **Evaluate** $A_{\lambda}(D)$
- **Select promising configuration** $\lambda \in \Lambda$

**Configuration $\lambda^*$**
Metalearning-Initialized SMBO (MI-SMBO)

- ML Algorithm $A$
- Configuration Space $\Lambda$ of $A$
- Dataset $D_{new}$

1. **Fit regression model on pairs of** $(\lambda, A_\lambda(D_{new}))$
2. **Select promising configuration** $\lambda \in \Lambda$
3. **Evaluate** $A_\lambda(D_{new})$

Configuration Task

**Configuration** $\lambda^*$
Metalearning-Initialized SMBO (MI-SMBO)

Find Datasets $D_i$ similar to $D_{\text{new}}$ → Initialize Search with $\lambda^*_i$ → Fit regression model on pairs of $(\lambda, A_{\lambda}(D_{\text{new}}))$ → Select promising configuration $\lambda \in \Lambda$ → Evaluate $A_{\lambda}(D_{\text{new}})$ → Configuration Task → Configuration $\lambda^*$
Metafeatures

- # training examples: 150
- # classes: 3
- # features: 4
- # numerical features: 4
- # categorical features: 0
- missing values? No

The iris pictures on this slide are from wikimedia commons and used under the following licenses: Top left: Iris Versicolor is public domain; Bottom left: Iris setosa is licensed by Radomil under CC BY-SA 3.0; Top right: Iris Virginica is licensed by C T Johansson under CC BY 3.0.
For a new dataset $D_{\text{new}}$:

- Sort known datasets $D_{1:N}$ by distance to $D_{\text{new}}$.
- For each of these datasets, extract the best known hyperparameter configuration $\lambda_{D_i}^*$.  
- Initialize SMBO with the first $k$ hyperparameter configurations from the sorted list.
Similarity of Datasets
Finding the nearest datasets (1)
Finding the nearest datasets (2)
Finding the nearest datasets (3)
Finding the nearest datasets (3)
Finding the nearest datasets (4)
Commonly used in literature, the $L_1$ norm:

$$d(D_{\text{new}}, D_j) = \sum_i |m_i^{\text{new}} - m_i^j|$$  \hspace{1cm} (1)
Experimental Setup

- 57 datasets from the OpenML repository
Experimental Setup

- 57 datasets from the OpenML repository
- 46 metafeatures from the literature:
  - Split into five different subsets, including landmarking
    [Pfahringer et al. 2000]
Experimental Setup

- 57 datasets from the OpenML repository
- 46 metafeatures from the literature:
  - Split into five different subsets, including landmarking [Pfahringer et al. 2000]
- Two case studies
  - Support Vector Machine with MI-Spearmint [Snoek et al. 2012]
  - AutoSklearn with MI-SMAC [Hutter et al. 2011]

Tried 5, 10, 20 and 25 initial configurations ran each instantiation 10 times on each dataset → 26220 optimization runs therefore, precomputed a dense grid for every dataset
Experimental Setup

- 57 datasets from the OpenML repository
- 46 metafeatures from the literature:
  - Split into five different subsets, including landmarking [Pfahringer et al. 2000]
- Two case studies
  - Support Vector Machine with MI-Spearmint [Snoek et al. 2012]
  - AutoSklearn with MI-SMAC [Hutter et al. 2011]
- Tried 5, 10, 20 and 25 initial configurations
Experimental Setup

- 57 datasets from the OpenML repository
- 46 metafeatures from the literature:
  - Split into five different subsets, including landmarking [Pfahringer et al. 2000]
- Two case studies
  - Support Vector Machine with MI-Spearmint [Snoek et al. 2012]
  - AutoSklearn with MI-SMAC [Hutter et al. 2011]
- Tried 5, 10, 20 and 25 initial configurations
- ran each instantiation 10 times on each dataset
  \( \rightarrow 26220 \) optimization runs
Experimental Setup

- 57 datasets from the OpenML repository
- 46 metafeatures from the literature:
  - Split into five different subsets, including landmarking
    - [Pfahringer et al. 2000]
- Two case studies
  - Support Vector Machine with MI-Spearmint [Snoek et al. 2012]
  - AutoSklearn with MI-SMAC [Hutter et al. 2011]
- Tried 5, 10, 20 and 25 initial configurations
- ran each instantiation 10 times on each dataset
  - → 26220 optimization runs
- therefore, precomputed a dense grid for every dataset
Combined Algorithm Selection and Hyperparameter Optimization problem (CASH)
Combined Algorithm Selection and Hyperparameter Optimization problem (CASH)

Classifier

SVM

Random Forest

Max Features

Criterion

Min Samples Split

LinearSVM

C(LinearSVM)

loss

C(SVM)

Max Features

gamma

C(SVM)

[Auto-WEKA, Thornton et al. 2013]
## AutoSklearn: Hyperparameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Hyperparameter</th>
<th># Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>$\lambda_{\text{classifier}}$</td>
<td>3</td>
</tr>
<tr>
<td>Main</td>
<td>preprocessing</td>
<td>2</td>
</tr>
<tr>
<td>SVM</td>
<td>$\log_2(C)$</td>
<td>21</td>
</tr>
<tr>
<td>SVM</td>
<td>$\log_2(\gamma)$</td>
<td>19</td>
</tr>
<tr>
<td>LinearSVM</td>
<td>$\log_2(C)$</td>
<td>21</td>
</tr>
<tr>
<td>LinearSVM</td>
<td>penalty</td>
<td>2</td>
</tr>
<tr>
<td>RF</td>
<td>min splits</td>
<td>5</td>
</tr>
<tr>
<td>RF</td>
<td>max features</td>
<td>10</td>
</tr>
<tr>
<td>RF</td>
<td>criterion</td>
<td>2</td>
</tr>
<tr>
<td>PCA</td>
<td>variance to keep</td>
<td>2</td>
</tr>
</tbody>
</table>
## AutoSklearn: Hyperparameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Hyperparameter</th>
<th># Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>$\lambda_{\text{classifier}}$</td>
<td>3</td>
</tr>
<tr>
<td>Main</td>
<td>preprocessing</td>
<td>2</td>
</tr>
<tr>
<td>SVM</td>
<td>$\log_2(C)$</td>
<td>21</td>
</tr>
<tr>
<td>SVM</td>
<td>$\log_2(\gamma)$</td>
<td>19</td>
</tr>
<tr>
<td>LinearSVM</td>
<td>$\log_2(C)$</td>
<td>21</td>
</tr>
<tr>
<td>LinearSVM</td>
<td>penalty</td>
<td>2</td>
</tr>
<tr>
<td>RF</td>
<td>min splits</td>
<td>5</td>
</tr>
<tr>
<td>RF</td>
<td>max features</td>
<td>10</td>
</tr>
<tr>
<td>RF</td>
<td>criterion</td>
<td>2</td>
</tr>
<tr>
<td>PCA</td>
<td>variance to keep</td>
<td>2</td>
</tr>
</tbody>
</table>

1623 hyperparameter configurations
AutoSklearn: Results (1)

![Graph showing the difference to min function value over #Function evaluations for SMAC.](image)
AutoSklearn: Results (1)
AutoSklearn: Results (2)
AutoSklearn: Results (3)

MI-SMAC(10, $L_1$, landmarking) vs SMAC

#Function evaluations

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

MI-SMAC(10, $L_1$, landmarking) vs SMAC
AutoSklearn: Results (3)

MI-SMAC($10, L_1, \text{landmarking}$) vs SMAC
AutoSklearn: Results (3)
AutoSklearn: Results (3)
Open questions

- Does MI-SMBO scale to larger configuration spaces?
- What if gridsearch is too expensive?
- Can the metalearning component be added directly into the SMBO procedure?
Take home messages

- SMBO can be substantially improved by providing good initial configurations.
- Metalearning provides a sound framework to find these configurations.
- MI-SMAC improves on state-of-the-art methods on a large configuration space, namely AutoSklearn.
The end

Thank you for your attention.

Further questions: feurerm@cs.uni-freiburg.de

This presentation was partially supported by an ECCAI Travel Award and the ECCAI sponsors.
AutoSklearn: Results (5)

![Graph showing comparison of different optimization algorithms: SMAC, random, TPE, and MI-SMAC(10, L_1, landmarking). The graph plots the minimum function value against the number of function evaluations.]

- **SMAC**
- **random**
- **TPE**
- **MI-SMAC(10, L_1, landmarking)**
AutoSklearn: Results (7)

MI-SMAC(10, L1, all) vs MI-SMAC(10, L1, landmarking)
MI-SMAC(10, L1, all) vs SMAC
MI-SMAC(10, L1, all) vs TPE
MI-SMAC(10, L1, all) vs random
AutoSklearn: Results (8)

- SMAC vs MI-SMAC(10, L1, landmarking)
- SMAC vs TPE
- SMAC vs random

Graph showing the performance of different algorithms over function evaluations.
AutoSklearn: Results (9)