Towards Explainable AutoML: xAutoML

Marius Lindauer

@LindauerMarius
@AutoML_org

* Slides available at automl.org/talks
AutoML Process

Evaluate Performance

“Guess” new ML Settings

Train ML Modell

AutoML Loop

Optimized ML-Pipeline

Pre-Processing
Normalization, Feature Selection, Feature Reduction, ...

Predictive Model
Linear model, SVM, RF, XGB, DNN, ...

Post-Processing
Who is using AutoML?

Users of ML without any deep expertise in ML

ML experts / Data scientists … (AutoML researchers)
Are ML-Experts using AutoML?

- Bouthillier and Varoquaux [2020] showed that authors of NeurIPS and ICLR papers:
  - a) they often **optimize their hyperparameters** (>75%)
  - b) they often **do it manually** and don’t use AutoML tools

- Crisan and Fiore-Gartland [2021] interviewed data scientists and concluded:
  - a) experts **don’t necessarily trust AutoML**
  - b) **visualization of results and processes** can help to increase the acceptance of AutoML results
Previous Approach-Agnostic xAutoML Methods

Note: Of course, approach-specific methods are also possible, e.g. [Ru et al. 2021].
General Setup

\[ c(t) = \mathcal{L}(\theta^*, D_{\text{val}}; \lambda^{(t)})^* \]

\[ \theta^* \in \min_{\theta} \mathcal{L}(\theta; D_{\text{train}}, \lambda^{(t)}) \]

* can in principle be any kind of cost function.
Visualization of Pipelines

- Visualization of sampled pipelines (incl. algorithms and hyperparameters) and their performance
- More descriptive analysis

Source: [Ono et al. 2020](Ono et al. 2020)
Parallel Coordinate Plots

- Visualization of sampling in high-dimensional hyperparameter spaces [Golovin et al. 2017]
- Allows to identify:
  - optimization focus of AutoML optimizers
  - well-performing combination of settings
  - Interaction effects between settings (to some degree)
- Rather qualitative, less quantitative analysis
- Follow up: Conditional parallel coordinate plots [Weidele et al. 2019]
Ablation Studies

- AutoML can start from some default settings
  a. Defined by the algorithm developer
  b. User expertise

- Question: **Which settings ($\lambda$) changes had the bigger impact on the performance?**

- Characteristic:
  a. Quantitative analysis
  b. Hard to visualize in high-dimensional spaces
  c. Rather local: subspace between default and incumbent setting

- Efficiency in high-dimensional spaces:
  a. Greedy approach [Fawcett and Hoos 2016]
  b. Use of surrogate models instead of expensive function evaluations (e.g., from Bayesian Optimization) [Biedenkapp et al. 2017]
ICE Curves and LPI

- **ICE Curve** [Goldstein et al. 2017]: individual effect of one feature for an individual observation
  - Slice through the space in one dimension at a given observation
- **LPI: Local (Hyper-)Parameter Importance** [Biedenkapp et al. 2018]
  - Same idea as ICE curves, but single ICE curve centered at the incumbent setting returned by an AutoML tool
  - Quantitative importance of hyperparameters:
    \[
    LPI(h \mid \lambda) = \frac{\text{Var}_{v \in \Lambda_h} \hat{c}(\lambda[v = v])}{\sum_{h' \in \mathcal{H}} \text{Var}_{w \in \Lambda_{h'}} \hat{c}(\lambda[w = w])}
    \]
## fANOVA for Hyperparameter Importance

- Fraction of explained variance by main and interaction effects of hyperparameters can be quantified by

\[
\nabla_{\mathcal{H}' \subset \mathcal{H}} = \frac{\frac{1}{||A_{\mathcal{H}'}||} \int \hat{c}(\lambda_{\mathcal{H}'})^2 d\lambda_{\mathcal{H}'}}{\frac{1}{||A||} \int (\hat{y}(\lambda) - \hat{c}_\mathcal{D})^2 d\lambda}
\]

- Efficient computation on a RF as surrogate model [Hutter et al. 2014]
- Allows to study importance across datasets [van Rijn & Hutter 2018, Sharma et al. 2019]

<table>
<thead>
<tr>
<th></th>
<th>fANOVA</th>
<th>LPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>discount</td>
<td>19.32</td>
<td>38.88</td>
</tr>
<tr>
<td>learning rate</td>
<td>3.70</td>
<td>35.4</td>
</tr>
<tr>
<td>batch size</td>
<td>15.77</td>
<td>21.5</td>
</tr>
<tr>
<td># units 1</td>
<td>1.86</td>
<td>0.07</td>
</tr>
<tr>
<td># units 2</td>
<td>0.39</td>
<td>0.01</td>
</tr>
</tbody>
</table>

PPO on cartpole

Source: [Lindauer et al. 2019]
Explaining HPO
via Partial Dependence Plots

Julia Moosbauer¹, Julia Herbinger¹, Giuseppe Casalicchio¹, Marius Lindauer², Bernd Bischl¹
For, a subset $S$ of the hyperparameters, the partial dependence function is:

$$c_S(\lambda_S) := \mathbb{E}_{\lambda_C} [c(\lambda)] = \int_{\Lambda_C} c(\lambda_S, \lambda_C) d\mathbb{P}(\lambda_C)$$

and can be approximated by Monte-Carlo integration:

$$\hat{c}_S(\lambda_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{m}\left(\lambda_S, \lambda^{(i)}_C\right)$$

where

$$\left(\lambda^{(i)}_C\right)_{i=1,\ldots,n} \sim \mathbb{P}(\lambda_C)$$

and $\lambda_S$ for a set of grid points.

→ Average of ICE curves.

[Hutter et al. 2014] showed how to do this efficiently for RFs as surrogate models.
Quantifying Uncertainties in PDPs

\[
\hat{s}_S^2 (\lambda_S) = \nabla_{\hat{c}} \left[ \hat{c}_S (\lambda_S) \right] \\
= \nabla_{\hat{c}} \left[ \frac{1}{n} \sum_{i=1}^{n} \hat{c} (\lambda_S, \lambda_C^{(i)}) \right] \\
= \frac{1}{n^2} 1^\top \hat{K} (\lambda_S) 1.
\]

→ requires a kernel correctly specifying the covariance structure (e.g., GPs).

Approximation:

\[
\hat{s}_S^2 (\lambda_S) \approx \frac{1}{n} \sum_{i=1}^{n} \hat{K} (\lambda_S)_{i,i}
\]

→ Model-agnostic (local) approximation
Problem: Biased Sampling

- PDPs assume that the data is i.i.d.
- Obviously not the case for efficient AutoML tools with a focus on high-performance regions

- Example:
  - BO with GPs and LCB
  - Different exploration rate for LCB to show different sampling bias

\[ \text{LCB}(\lambda) = \mu(\lambda) + \beta \cdot \sigma(\lambda) \]
Impact of the Sampling Bias

- Simply using all observations from AutoML tools might lead to misleading PDPs
- Uncertainty estimates help to quantify the poor fits

→ of course, sampling bias is wanted and the solution cannot be to change the sampling behavior
Partitioning of Space

Partition space to obtain interpretable subspaces $\mathcal{N}'$.

Uncertainty variation across all ICE estimates:

$$L(\lambda_S, \mathcal{N}') = \sum_{i \in \mathcal{N}} \left( \hat{s}^2 \left( \lambda_S, \lambda_C^{(i)} \right) - \hat{s}^2_{\mathcal{N}'|\lambda_S} (\lambda_S) \right)^2$$

$$\hat{s}^2_{\mathcal{N}'|\lambda_S} (\lambda_S) := \frac{1}{|\mathcal{N}'|} \sum_{i \in \mathcal{N}'} \hat{s}^2 \left( \lambda_S, \lambda_C^{(i)} \right)$$

→ Uncertainty structure of ICE curves should maximally agree

Split Loss = Aggregation over all grid points:

$$R_{L2}(\mathcal{N}') = \sum_{g=1}^{G} L(\lambda_S^{(g)}; \mathcal{N}')$$

Note (i): Partition only along the marginalized dimensions

Ground truth

PDP incumbent

$\lambda_j < 6$

$\lambda_j \geq 6$
Empirical Results
Effect of Splitting on an Artificial Function

Main Insights:

- For higher-dimensional problems, PDPs are potentially more uncertain
- Mean Confidence (MC) increases with deeper trees
- More to gain for high-sampling bias cases
Explaining LCBench \cite{Zimmer et al. 2021}

Setting:

- Small configuration space of Auto-PyTorch Tabular
- Training of a RF as surrogate on LCBench with 2000 randomly sampled configurations
- Bayesian Optimization with 200 function evaluations

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>$\delta$ MC (%)</th>
<th>$\delta$ OC (%)</th>
<th>$\delta$ NLL (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>40.8 (14.9)</td>
<td>61.9 (13.5)</td>
<td>19.8 (19.5)</td>
</tr>
<tr>
<td>Learning rate</td>
<td>50.2 (13.7)</td>
<td>57.6 (14.4)</td>
<td>17.9 (20.5)</td>
</tr>
<tr>
<td>Max. dropout rate</td>
<td>49.7 (15.4)</td>
<td>62.4 (11.9)</td>
<td>17.4 (18.2)</td>
</tr>
<tr>
<td>Max. units</td>
<td>51.1 (15.2)</td>
<td>58.6 (12.7)</td>
<td>24.6 (22.0)</td>
</tr>
<tr>
<td>Momentum</td>
<td>51.7 (14.5)</td>
<td>58.3 (12.7)</td>
<td>19.7 (21.7)</td>
</tr>
<tr>
<td>Number of layers</td>
<td>30.6 (16.4)</td>
<td>50.9 (16.6)</td>
<td>13.8 (32.5)</td>
</tr>
<tr>
<td>Weight decay</td>
<td>36.3 (22.6)</td>
<td>61.0 (13.1)</td>
<td>11.9 (19.7)</td>
</tr>
</tbody>
</table>

*Improvement of mean confidence (MC), confidence close to incumbent (OC), and negative log-likelihood (NLL) after 6 splits*

Take-away:

$\rightarrow$ the confidence of PDPs improves across all hyperparameters and metrics
Explaining Auto-PyTorch (cont’d) [Zimmer et al. 2021]

Ground truth

PDP

incumbent

Subregion definition:
weight_decay <= 0.086

Subregion definition:
num_layers <= 4.5, weight_decay <= 0.0178, max_dropout <= 0.6966

Subregion definition:
batch_size <= 7.5329

Subregion definition:
max_dropout <= 0.7305, num_layers <= 4.5, batch_size <= 6.1739, weight_decay <= 0.0172
Future Work and Conclusion
Take-Home

Summary:

- Explaining AutoML is important to create trust in them
- Common iML methods such as PDPs can be used to explain AutoML
- However, i.i.d assumptions might be violated
- PDPs can be extended to uncertainty estimates
- Split into subspaces with better interpretability

Future Work:

- Additional samples to efficiently reduce sampling bias
- Extension to multi-fidelity setting AutoML
- Other iML methods
Goal: Human-Centered AutoML

AutoML Loop

- Evaluate Performance
- "Guess" new ML Settings
- Update of: Search space, Constraints, Preferences, Priors, ...
- Meta-Data: Setting → Performance
- Interpretation by User
- Explain Meta-Data
- Quantitative and Qualitative Explanations

Meta-Data:

- Setting → Performance
Thank you!