AutoML: Interpretability

Overview: Automated Empirical Analysis

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Idea

- Big challenge of ML: Interpretability
  - In some applications, it is required to "understand" a prediction
  - Users have less trust in systems, they can't understand

Idea

- Big challenge of ML: Interpretability
  - In some applications, it is required to "understand" a prediction
  - Users have less trust in systems, they can't understand
- AutoML is even worse?
  - AutoML is a black-box that automates the design of another blackbox (ML)
  - Also ML-developers have an basic understanding of the design of their ML pipelines
- Automated empirical interpretability helps to
  - understand the finally returned ML system
  - understand the AutoML process

Approach

- Insights:
  - AutoML is yet another optimization problem
  - (Most) AutoML approach are iterative in nature

$\rightsquigarrow$ AutoML generates a lot of empirical data

Cost c

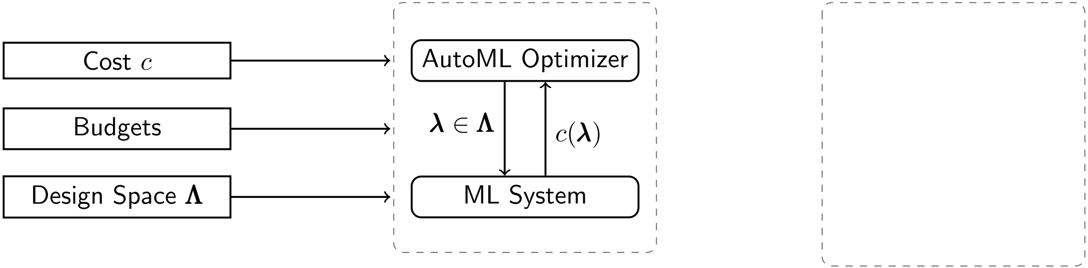
Budgets

Design Space $\boldsymbol{\Lambda}$

Approach

- Insights:
  - AutoML is yet another optimization problem
  - (Most) AutoML approach are iterative in nature

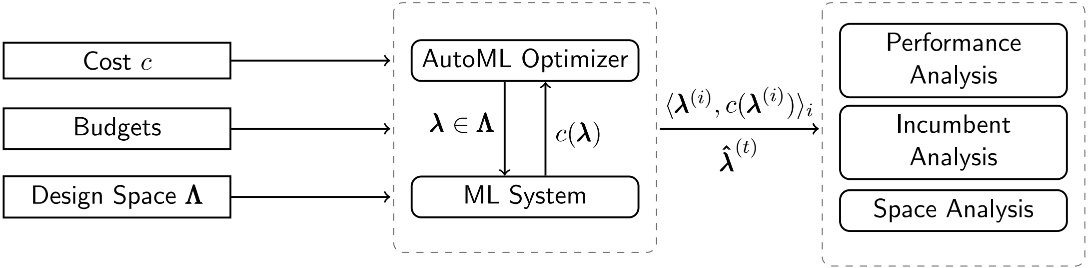
$\rightsquigarrow$ AutoML generates a lot of empirical data



Approach

- Insights:
  - AutoML is yet another optimization problem
  - (Most) AutoML approach are iterative in nature

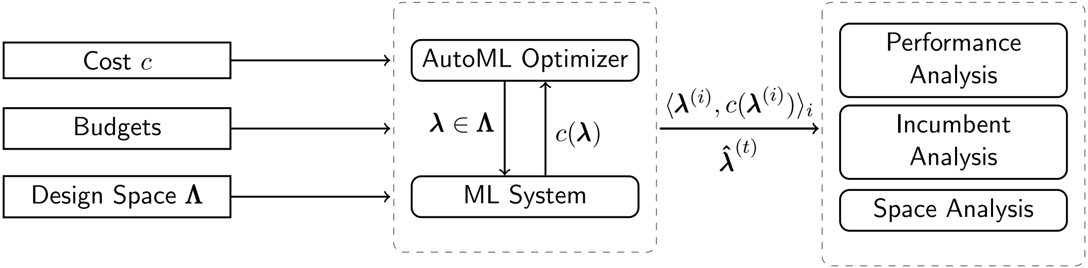
$\rightsquigarrow$ AutoML generates a lot of empirical data



Approach

- Insights:
  - AutoML is yet another optimization problem
  - (Most) AutoML approach are iterative in nature

$\rightsquigarrow$ AutoML generates a lot of empirical data



$\rightsquigarrow$ Let's use this data to learn something about our AutoML problem

Basic Examples

- Visualize final incumbent $\hat{\lambda}$
  - ML pipeline with its components
  - Neural architecture

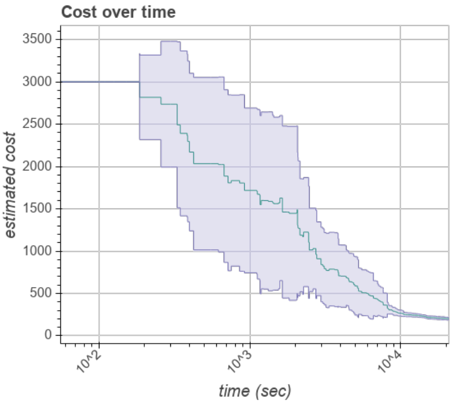
Basic Examples

- Visualize final incumbent $\hat{\lambda}$
  - ML pipeline with its components
  - Neural architecture
- Compare what changed between λ_{def} and $\hat{\lambda}$

Basic Examples

- Visualize final incumbent $\hat{\boldsymbol{\lambda}}$
  - ML pipeline with its components
  - Neural architecture
- Compare what changed between $\boldsymbol{\lambda}_{\text{def}}$ and $\hat{\boldsymbol{\lambda}}$
- Show $\hat{\boldsymbol{\lambda}}$ on different budgets (if you used a multi-fideltiy approach)

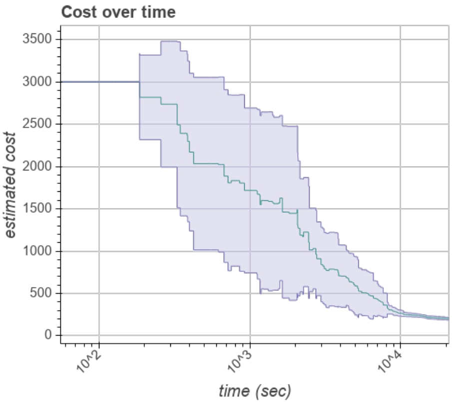
Cost Over Time



Source: [Lindauer et al. 2019]

- Study how your AutoML tool improves cost (or loss) over time

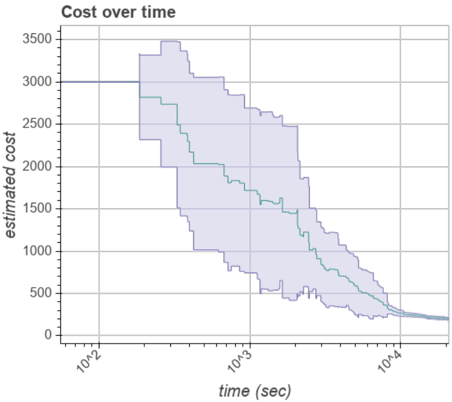
Cost Over Time



Source: [Lindauer et al. 2019]

- Study how your AutoML tool improves cost (or loss) over time
- Allows to identify whether
  - you need less time next time or

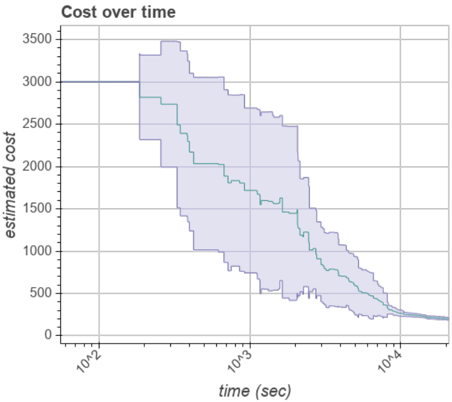
Cost Over Time



Source: [Lindauer et al. 2019]

- Study how your AutoML tool improves cost (or loss) over time
- Allows to identify whether
  - you need less time next time or
  - the AutoML system is still improving; so you should give it more time

Cost Over Time



Source: [Lindauer et al. 2019]

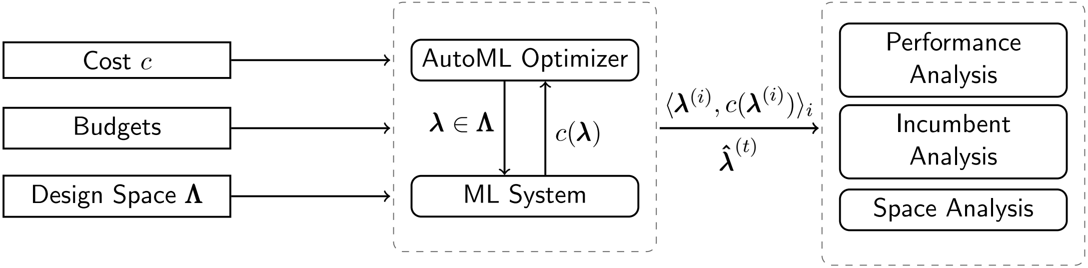
- Study how your AutoML tool improves cost (or loss) over time
- Allows to identify whether
  - you need less time next time or
  - the AutoML system is still improving; so you should give it more time
- Notes:
  - Plot on log-scale to see details in the beginning
  - If you done several runs, plot distribution (e.g., median and $25/75\%$-quartiles)

AutoML: Interpretability

Studying the AutoML Optimization Process

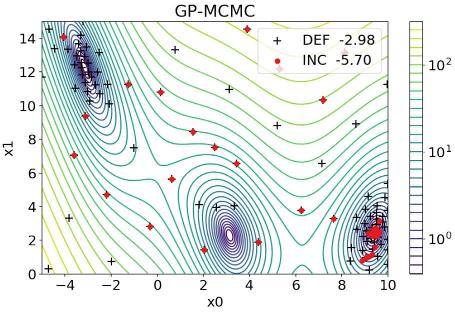
Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Idea



$\rightsquigarrow$ focus on how the AutoML optimizer samples from the design space $\boldsymbol{\Lambda}$

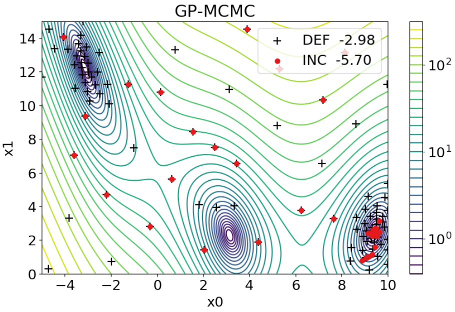
Analyzing the Sampling Behavior



Source: [Lindauer et al. 2019]

- Plot of a 1D or 2D function

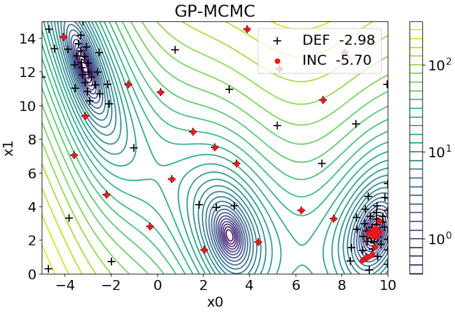
Analyzing the Sampling Behavior



Source: [Lindauer et al. 2019]

- Plot of a 1D or 2D function
- Background shows the ground truth (real function values)

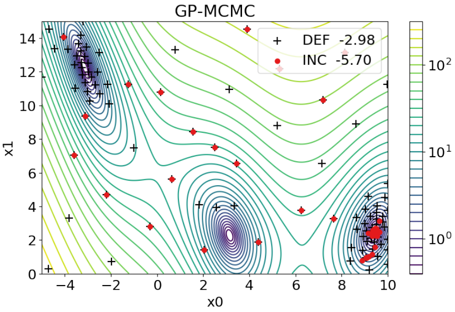
Analyzing the Sampling Behavior



Source: [Lindauer et al. 2019]

- Plot of a 1D or 2D function
- Background shows the ground truth (real function values)
- Dots are sampled points in the search space

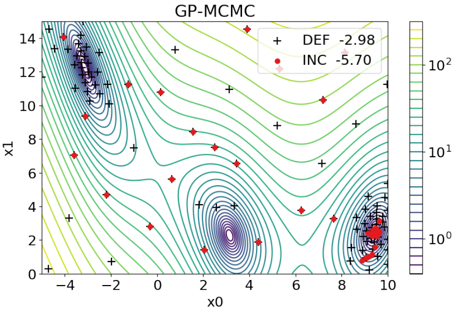
Analyzing the Sampling Behavior



Source: [Lindauer et al. 2019]

- Plot of a 1D or 2D function
- Background shows the ground truth (real function values)
- Dots are sampled points in the search space
- Typical approach in Bayesian Optimization community

Analyzing the Sampling Behavior

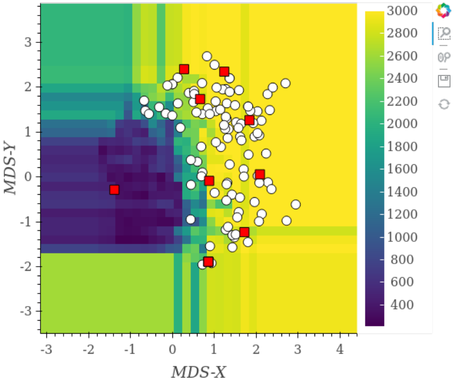


Source: [Lindauer et al. 2019]

- Plot of a 1D or 2D function
- Background shows the ground truth (real function values)
- Dots are sampled points in the search space
- Typical approach in Bayesian Optimization community

⇝ Impossible for higher dimensional problems?

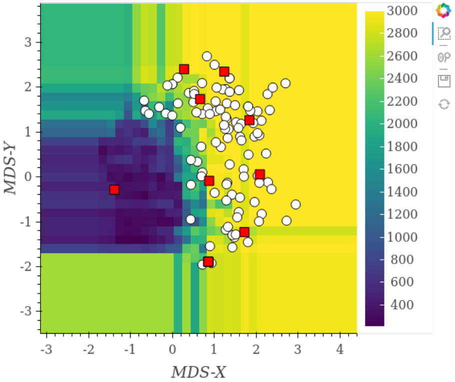
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D

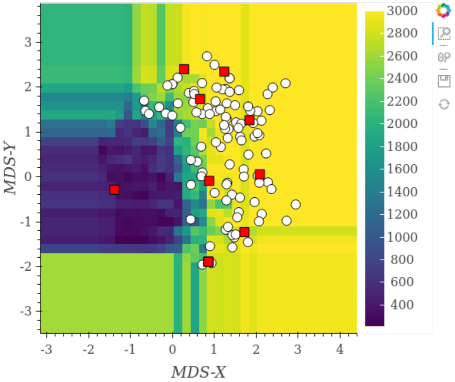
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D
  1. Use an MDS to project down to 2-D

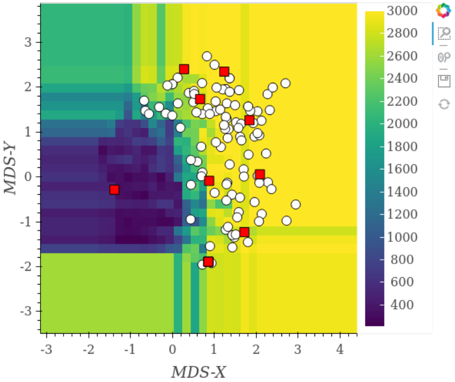
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D
  1. Use an MDS to project down to 2-D
  2. Each dot is single hyperparameter configuration

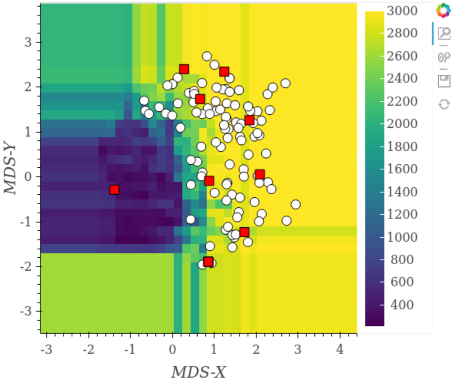
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D
  1. Use an MDS to project down to 2-D
  2. Each dot is single hyperparameter configuration
  3. Red squares are intermediate incumbents

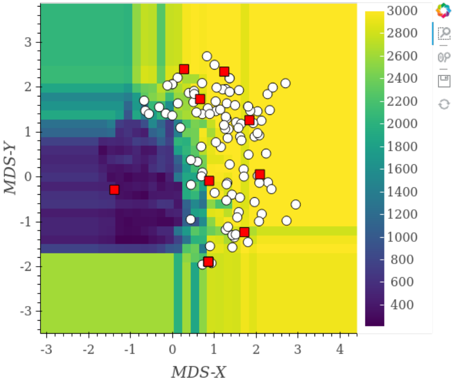
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D
  1. Use an MDS to project down to 2-D
  2. Each dot is single hyperparameter configuration
  3. Red squares are intermediate incumbents
  4. The background is colored wrt a performance-estimate
     (e.g., reusing model fitted during BO)

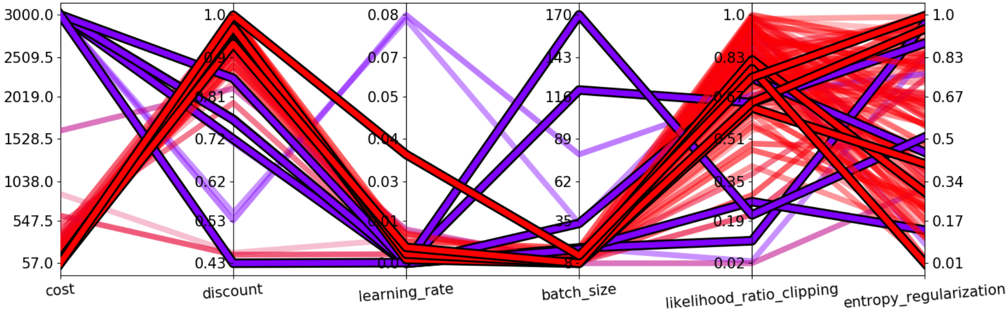
Analyzing the Sampling Behavior in N-D



Source: [Lindauer et al. 2019]

- Same idea as before
  but we have to project N-D into 2-D
  1. Use an MDS to project down to 2-D
  2. Each dot is single hyperparameter configuration
  3. Red squares are intermediate incumbents
  4. The background is colored wrt a performance-estimate (e.g., reusing model fitted during BO)
  5. Extension: Animation by showing how points get added over time

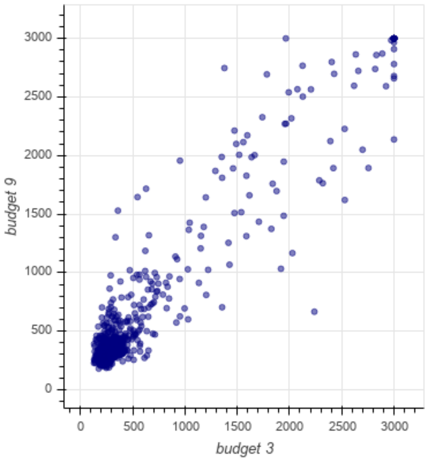
Parallel Coordinate Plot [Golovin et al. 2017]



Source: [Lindauer et al. 2019]

- Each coordinate is one hyperparameter;
- Except the most left one: cost or loss

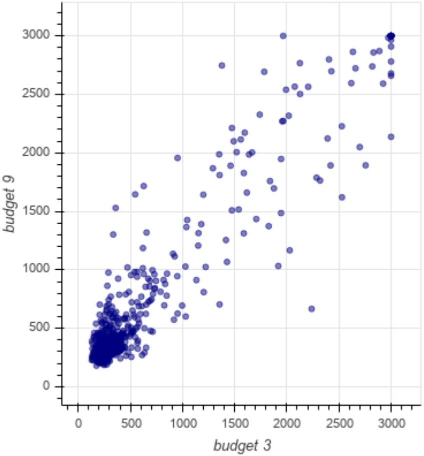
Multi-Fidelity Checks



Source: [Lindauer et al. 2019]

- Challenge of multi-fidelity approaches:
  - How to choose the fidelities (a.k.a. budgets)

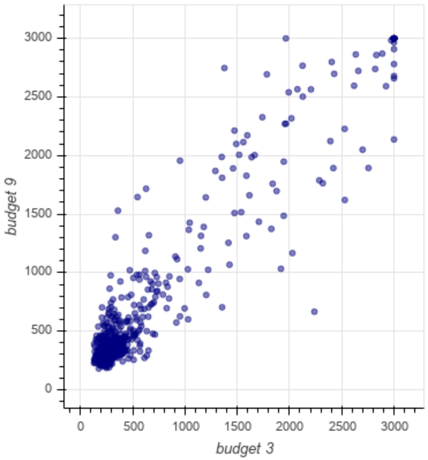
Multi-Fidelity Checks



Source: [Lindauer et al. 2019]

- Challenge of multi-fidelity approaches:
  - How to choose the fidelities (a.k.a. budgets)
- Important Property:
  - Decisions on small budgets should be reasonable for higher budgets

Multi-Fidelity Checks



Source: [Lindauer et al. 2019]

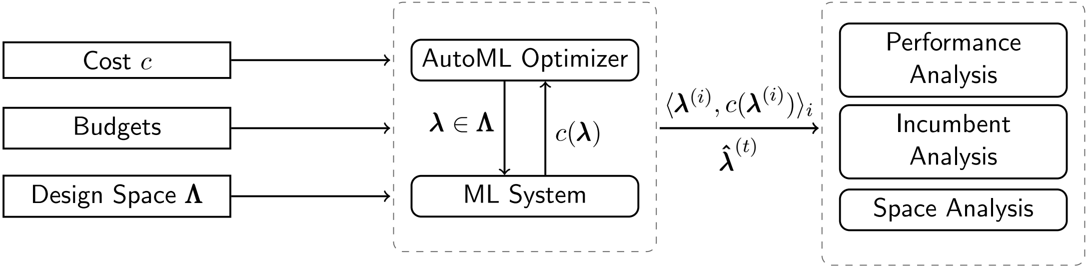
- Challenge of multi-fidelity approaches:
  - How to choose the fidelities (a.k.a. budgets)
- Important Property:
  - Decisions on small budgets should be reasonable for higher budgets
- Analysis:
  1. Scatter plot of performance on Budget X vs. Budget Y
  2. Each dot is sampled hyperparameter configuration
  3. Compute rank correlation (here: 0.69)

AutoML: Interpretability

Incumbent Analysis and Local Hyperparameter Importance

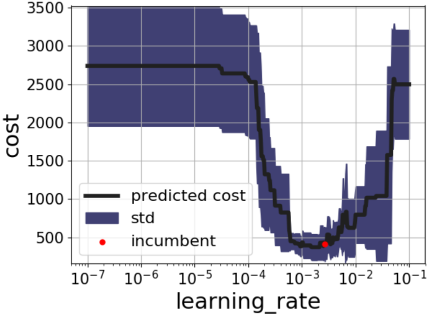
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Idea



$\rightsquigarrow$ focus on why is the eventually returned configuration a good choice

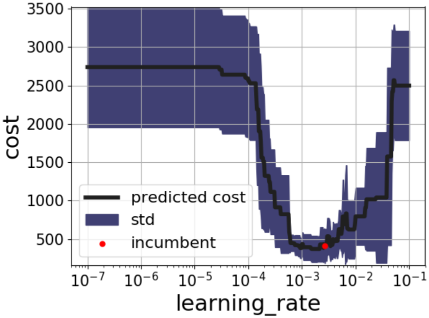
Local Importance [Biedenkapp et al. 2018]



Source: [Lindauer et al. 2019]

- Typical question of users:
  - How would the performance change if we change hyperparameter $\boldsymbol{\lambda}_i$?

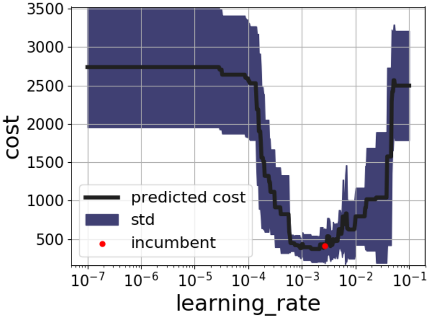
Local Importance [Biedenkapp et al. 2018]



Source: [Lindauer et al. 2019]

- Typical question of users:
  - How would the performance change if we change hyperparameter $\boldsymbol{\lambda}_i$?
- Problem: Running full study is often too expensive
  - Each run of an ML-system is potential expensive

Local Importance [Biedenkapp et al. 2018]



Source: [Lindauer et al. 2019]

- Typical question of users:
  - How would the performance change if we change hyperparameter $\boldsymbol{\lambda}_i$?
- Problem: Running full study is often too expensive
  - Each run of an ML-system is potential expensive
- Key Ideas:
  - Re-use probabilistic models as trained in BO
  - Plot performance change around $\hat{\boldsymbol{\lambda}}^{(t)}$ along each dimension

Quantifying Local Importance [Biedenkapp et al. 2018]

$$\mathsf{VAR}_{\boldsymbol{\lambda}}(i) \quad = \quad \sum_{v \in \boldsymbol{\Lambda}_i} \left(\mathbb{E}_{v \sim \boldsymbol{\Lambda}_i}[L(\boldsymbol{\lambda})] - L(\boldsymbol{\lambda}[\boldsymbol{\lambda}_i := v])\right)^2 \tag{1}$$

Quantifying Local Importance [Biedenkapp et al. 2018]

$$\mathsf{VAR}_{\boldsymbol{\lambda}}(i) = \sum_{v \in \boldsymbol{\Lambda}_i} \left(\mathbb{E}_{v \sim \boldsymbol{\Lambda}_i}[L(\boldsymbol{\lambda})] - L(\boldsymbol{\lambda}[\boldsymbol{\lambda}_i := v])\right)^2 \tag{1}$$

$$\mathsf{LPI}(i \mid \boldsymbol{\lambda}) = \frac{VAR_{\boldsymbol{\lambda}}(i)}{\sum_j VAR_{\boldsymbol{\lambda}}(j)} \tag{2}$$

Quantifying Local Importance [Biedenkapp et al. 2018]

$$\mathrm{VAR}_{\boldsymbol{\lambda}}(i) = \sum_{v \in \boldsymbol{\Lambda}_i} \left(\mathbb{E}_{v \sim \boldsymbol{\Lambda}_i}[L(\boldsymbol{\lambda})] - L(\boldsymbol{\lambda}[\boldsymbol{\lambda}_i := v])\right)^2 \quad (1)$$

$$\mathrm{LPI}(i \mid \boldsymbol{\lambda}) = \frac{VAR_{\boldsymbol{\lambda}}(i)}{\sum_j VAR_{\boldsymbol{\lambda}}(j)} \quad (2)$$

⇝ While fixing all other hyperparameters to the incumbent value, the hyperparameter with the highest variance is the most important one

Ablation Study for Importance

- Users often start from some kind of default configuration
  1. As given in the documentation
  2. Or as always used in the last time

Ablation Study for Importance

- Users often start from some kind of default configuration
  1. As given in the documentation
  2. Or as always used in the last time
- Key Idea: Going from the default to the automatically optimized configuration, which choices where important?

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42]
\end{aligned}
$$

Ablation Study for Importance

- Users often start from some kind of default configuration
  1. As given in the documentation
  2. Or as always used in the last time
- Key Idea: Going from the default to the automatically optimized configuration, which choices where important?

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\mathrm{start})} &= [1, 1, 0, 100] \\
\boldsymbol{\lambda}^{(\mathrm{end})} &= [0.98, 2.42, 1, 42]
\end{aligned}
$$

- Cheap approach: Assess $\boldsymbol{\lambda}^{(\mathrm{end})}$ with each hyperparameter value from $\boldsymbol{\lambda}^{(\mathrm{start})}$

Ablation Study for Importance

- Users often start from some kind of default configuration
  1. As given in the documentation
  2. Or as always used in the last time
- Key Idea: Going from the default to the automatically optimized configuration, which choices where important?

$$\begin{aligned} \boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] \\ \boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] \end{aligned}$$

- Cheap approach: Assess $\boldsymbol{\lambda}^{(\mathsf{end})}$ with each hyperparameter value from $\boldsymbol{\lambda}^{(\mathsf{start})}$
- Expensive approach: Try all mixtures of $\boldsymbol{\lambda}^{(\mathsf{end})}$ and $\boldsymbol{\lambda}^{(\mathsf{start})}$
  - Only feasible for small spaces and fairly cheap ML systems

Ablation Study for Importance

- Users often start from some kind of default configuration
  1. As given in the documentation
  2. Or as always used in the last time
- Key Idea: Going from the default to the automatically optimized configuration, which choices where important?

$$\boldsymbol{\lambda}^{(\text{start})} = [1, 1, 0, 100]$$
$$\boldsymbol{\lambda}^{(\text{end})} = [0.98, 2.42, 1, 42]$$

- Cheap approach: Assess $\boldsymbol{\lambda}^{(\text{end})}$ with each hyperparameter value from $\boldsymbol{\lambda}^{(\text{start})}$
- Expensive approach: Try all mixtures of $\boldsymbol{\lambda}^{(\text{end})}$ and $\boldsymbol{\lambda}^{(\text{start})}$
  - Only feasible for small spaces and fairly cheap ML systems
- Trade-off: Find a way from $\boldsymbol{\lambda}^{(\text{start})}$ to $\boldsymbol{\lambda}^{(\text{end})}$ in a greedy fashion [Fawcett and Hoos. 2016]

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\% \end{aligned}$$

Greedy Ablation Study

Given:

$$\boldsymbol{\lambda}^{(\mathsf{start})} = [1, 1, 0, 100] \qquad L_{\mathsf{start}} = 20\%$$
$$\boldsymbol{\lambda}^{(\mathsf{end})} = [0.98, 2.42, 1, 42] \qquad L_{\mathsf{end}} = 4\%$$

1st Iteration:

$$\boldsymbol{\lambda}^{(1)} = [0.98, 1, 0, 100] \qquad L_1 = 19\%$$

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\% \end{aligned}$$

1st Iteration:

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= [0.98, 1, 0, 100] & L_1 &= 19\% \\ \boldsymbol{\lambda}^{(2)} &= [1, 2.42, 0, 100] & L_2 &= 20\% \end{aligned}$$

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\% \end{aligned}$$

1st Iteration:

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= [0.98, 1, 0, 100] & L_1 &= 19\% \\ \boldsymbol{\lambda}^{(2)} &= [1, 2.42, 0, 100] & L_2 &= 20\% \\ \boldsymbol{\lambda}^{(3)} &= [1, 1, 1, 100] & L_3 &= 7\% \end{aligned}$$

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\% \end{aligned}$$

1st Iteration:

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= [0.98, 1, 0, 100] & L_1 &= 19\% \\ \boldsymbol{\lambda}^{(2)} &= [1, 2.42, 0, 100] & L_2 &= 20\% \\ \boldsymbol{\lambda}^{(3)} &= [1, 1, 1, 100] & L_3 &= 7\% \\ \boldsymbol{\lambda}^{(4)} &= [1, 1, 0, 42] & L_4 &= 16\% \end{aligned}$$

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\% \end{aligned}$$

1st Iteration:

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= [0.98, 1, 0, 100] & L_1 &= 19\% \\ \boldsymbol{\lambda}^{(2)} &= [1, 2.42, 0, 100] & L_2 &= 20\% \\ \boldsymbol{\lambda}^{(3)} &= [1, 1, 1, 100] & L_3 &= 7\% \\ \boldsymbol{\lambda}^{(4)} &= [1, 1, 0, 42] & L_4 &= 16\% \end{aligned}$$

$\rightsquigarrow$ 1st step: $\boldsymbol{\lambda}_2$ – flipping hyperparameter 3

Greedy Ablation Study

Given:

$$\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\%
\end{aligned}$$

2nd Iteration:

$$\boldsymbol{\lambda}^{(1)} = [0.98, 1, 1, 100] \quad L_1 = 6\%$$

Greedy Ablation Study

Given:

$$\begin{aligned} \boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\ \boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\ \boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\% \end{aligned}$$

2nd Iteration:

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= [0.98, 1, 1, 100] & L_1 &= 6\% \\ \boldsymbol{\lambda}^{(2)} &= [1, 2.42, 1, 100] & L_2 &= 7\% \end{aligned}$$

Greedy Ablation Study

Given:

$$\begin{aligned}
\boldsymbol{\lambda}^{(\mathsf{start})} \quad &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} \quad &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(\mathsf{end})} \quad &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\%
\end{aligned}$$

2nd Iteration:

$$\begin{aligned}
\boldsymbol{\lambda}^{(1)} &= [0.98, 1, 1, 100] & L_1 &= 6\% \\
\boldsymbol{\lambda}^{(2)} &= [1, 2.42, 1, 100] & L_2 &= 7\% \\
\boldsymbol{\lambda}^{(3)} &= [1, 1, 1, 42] & L_3 &= 5\%
\end{aligned}$$

$\rightsquigarrow$ 2nd step: $\boldsymbol{\lambda}_3$ – flipping hyperparameter 4

Greedy Ablation Study

Given:

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(s2)} &= [1, 1, 1, 42] & L &= 5\% \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\%
\end{aligned}
$$

3rd Iteration:

$$
\begin{aligned}
\boldsymbol{\lambda}^{(1)} &= [0.98, 1, 1, 100] & L_1 &= 4\% \\
\boldsymbol{\lambda}^{(2)} &= [1, 2.42, 1, 100] & L_2 &= 5\%
\end{aligned}
$$

$\rightsquigarrow$ 2nd step: $\boldsymbol{\lambda}_3$ – flipping hyperparameter 1

Greedy Ablation Study

Ablation Path:

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\mathsf{start})} &= [1, 1, 0, 100] & L_{\mathsf{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 42] & L &= 5\% \\
\boldsymbol{\lambda}^{(s3)} &= [0.98, 1, 1, 42] & L &= 4\% \\
\boldsymbol{\lambda}^{(s4)} &= [0.98, 2.42, 1, 42] & L &= 4\% \\
\boldsymbol{\lambda}^{(\mathsf{end})} &= [0.98, 2.42, 1, 42] & L_{\mathsf{end}} &= 4\%
\end{aligned}
$$

Greedy Ablation Pseudo Code

Algorithm 1 Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\mathbf{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;

$P \leftarrow [\,]$;

Greedy Ablation Pseudo Code

Algorithm 2 Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\mathbf{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;

$P \leftarrow []$;

foreach $t \in \{1 \ldots |\mathbf{\Lambda}|\}$ **do**

Greedy Ablation Pseudo Code

Algorithm 3 Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\boldsymbol{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;

$P \leftarrow []$;

foreach $t \in \{1 \dots |\boldsymbol{\Lambda}|\}$ **do**

  foreach $\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(end)})$ **do**

    $\boldsymbol{\lambda}'_\delta \leftarrow$ apply δ to $\boldsymbol{\lambda}$;

    evaluate $c(\boldsymbol{\lambda}'_\delta)$;

Greedy Ablation Pseudo Code

Algorithm 4 Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\mathbf{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;
$P \leftarrow []$;
foreach $t \in \{1 \ldots |\mathbf{\Lambda}|\}$ **do**
  foreach $\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(end)})$ **do**
    $\boldsymbol{\lambda}'_\delta \leftarrow$ apply δ to $\boldsymbol{\lambda}$;
    evaluate $c(\boldsymbol{\lambda}'_\delta)$;
  Determine most important change $\delta^* \in \arg\min_{\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(\text{end})})} c(\boldsymbol{\lambda}_\delta)$;
  $\boldsymbol{\lambda} \leftarrow$ apply δ^* to $\boldsymbol{\lambda}$;
  P.append(δ^*);

Greedy Ablation Pseudo Code

Algorithm 5 Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\mathbf{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;
$P \leftarrow [\,]$;
foreach $t \in \{1 \ldots |\mathbf{\Lambda}|\}$ **do**
  foreach $\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(end)})$ **do**
    $\boldsymbol{\lambda}'_\delta \leftarrow$ apply δ to $\boldsymbol{\lambda}$;
    evaluate $c(\boldsymbol{\lambda}'_\delta)$;
  Determine most important change $\delta^* \in \arg\min_{\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(\text{end})})} c(\boldsymbol{\lambda}_\delta)$;
  $\boldsymbol{\lambda} \leftarrow$ apply δ^* to $\boldsymbol{\lambda}$;
  P.append(δ^*);
return *Ablation path* P

Remarks on Ablation

- Even this greedy ablation requires $\mathcal{O}(n^2)$ steps

Remarks on Ablation

- Even this greedy ablation requires $\mathcal{O}(n^2)$ steps

$\rightsquigarrow$ We can also speedup that up by using surrogate models
[Biedenkapp et al. 2017]

Remarks on Ablation

- Even this greedy ablation requires $\mathcal{O}(n^2)$ steps

$\rightsquigarrow$ We can also speedup that up by using surrogate models
[Biedenkapp et al. 2017]

- Common observations:
  1. Some hyperparameters might not matter ($\boldsymbol{\lambda}_2$ in the example)

Remarks on Ablation

- Even this greedy ablation requires $\mathcal{O}(n^2)$ steps

$\rightsquigarrow$ We can also speedup that up by using surrogate models
[Biedenkapp et al. 2017]

- Common observations:
  1. Some hyperparameters might not matter ($\boldsymbol{\lambda}_2$ in the example)
  2. Often only a few of the hyperparameters have an big impact

Remarks on Ablation

- Even this greedy ablation requires $\mathcal{O}(n^2)$ steps

⇝ We can also speedup that up by using surrogate models
[Biedenkapp et al. 2017]

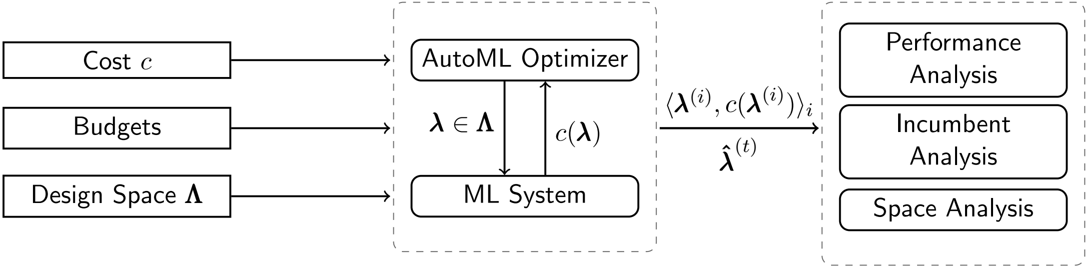
- Common observations:
  1. Some hyperparameters might not matter ($\boldsymbol{\lambda}_2$ in the example)
  2. Often only a few of the hyperparameters have an big impact
  3. You have plateaus in your ablation path because of interaction effects

AutoML: Interpretability

Global Hyperparameter Importance

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Idea



$\rightsquigarrow$ focus on which hyperparameters are important across the entire search space

Importance Analysis of Surrogate Model

- Key Idea: Surrogate models ($\mathbf{\Lambda} \to \mathbb{R}$) learn from observations how to predict the performance of a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$

Importance Analysis of Surrogate Model

- Key Idea: Surrogate models ($\mathbf{\Lambda} \to \mathbb{R}$) learn from observations how to predict the performance of a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$

$\leadsto$ These models can be used to figure out which hyperparameter was important

Importance Analysis of Surrogate Model

- Key Idea: Surrogate models ($\boldsymbol{\Lambda} \rightarrow \mathbb{R}$) learn from observations how to predict the performance of a hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$

$\leadsto$ These models can be used to figure out which hyperparameter was important

- For example:
  - Use forward selection [Hutter et al. 2013]
  - Use automatic feature relevance determination of the model (e.g., of a surrogate model based on random forest)

Importance Analysis of Surrogate Model

- Key Idea: Surrogate models ($\mathbf{\Lambda} \rightarrow \mathbb{R}$) learn from observations how to predict the performance of a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$

$\leadsto$ These models can be used to figure out which hyperparameter was important

- For example:
  - Use forward selection [Hutter et al. 2013]
  - Use automatic feature relevance determination of the model (e.g., of a surrogate model based on random forest)
- Advantages:
  - Very cheap to do, since we only have to query the surrogate model several times

Importance Analysis of Surrogate Model

- Key Idea: Surrogate models ($\mathbf{\Lambda} \to \mathbb{R}$) learn from observations how to predict the performance of a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$

$\rightsquigarrow$ These models can be used to figure out which hyperparameter was important

- For example:
  - Use forward selection [Hutter et al. 2013]
  - Use automatic feature relevance determination of the model (e.g., of a surrogate model based on random forest)
- Advantages:
  - Very cheap to do, since we only have to query the surrogate model several times
- Potential drawback:
  - The surrogate model might overfit to different subsets of the hyperparameters (if we don't provide sufficient data)

Global Importance Analysis

- Key idea: What is the importance of a hyperparameter by marginalizing over all other hyperparameter effects?

Global Importance Analysis

- Key idea: What is the importance of a hyperparameter by marginalizing over all other hyperparameter effects?
- Key Insight: We can use a surrogate model to compute these effects

Global Importance Analysis

- Key idea: What is the importance of a hyperparameter by marginalizing over all other hyperparameter effects?
- Key Insight: We can use a surrogate model to compute these effects

fANOVA [Sobobl. 1993]

Write performance predictions as a sum of components:

$$\hat{y}(\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_n) = \hat{f}_0 + \textstyle\sum_{i=1}^{n} \hat{f}_i(\boldsymbol{\lambda}_i) + \sum_{i \neq j} \hat{f}_{ij}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j) + \dots$$

$$\hat{y}(\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_n) = \text{average response} + \text{main effects} + \text{2-D interaction effects} + \text{higher order effects}$$

Global Importance Analysis

- Key idea: What is the importance of a hyperparameter by marginalizing over all other hyperparameter effects?
- Key Insight: We can use a surrogate model to compute these effects

fANOVA [Sobobl. 1993]

Write performance predictions as a sum of components:

$$
\begin{aligned}
\hat{y}(\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_n) = & \quad \hat{f}_0 + \textstyle\sum_{i=1}^{n} \hat{f}_i(\boldsymbol{\lambda}_i) + \sum_{i \neq j} \hat{f}_{ij}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j) + \dots \\
\hat{y}(\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_n) = & \quad \text{average response} + \text{main effects} + \\
& \quad \text{2-D interaction effects} + \text{higher order effects}
\end{aligned}
$$

Variance Decomposition

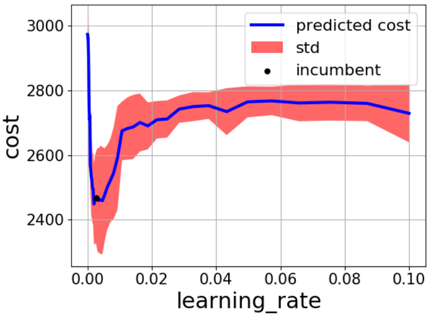
$$
\mathrm{V} = \frac{1}{||\boldsymbol{\Lambda}||} \int_{\boldsymbol{\lambda}_1} \dots \int_{\boldsymbol{\lambda}_n} [(\hat{y}(\boldsymbol{\lambda}) - \hat{f}_0)^2] d\boldsymbol{\lambda}_1 \dots d\boldsymbol{\lambda}_n
$$

fANOVA Analysis

- The fANOVA and variance decomposition can be done efficiently in linear time if the surrogate model is a random forest [Hutter et al. 2014]

fANOVA Analysis

- The fANOVA and variance decomposition can be done efficiently in linear time if the surrogate model is a random forest [Hutter et al. 2014]

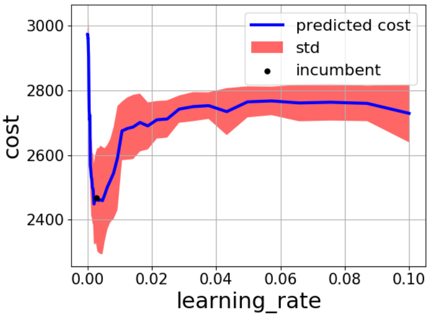


Source: [Lindauer et al. 2019]

- predicted cost is marginalized over all other hyperparameter effects

fANOVA Analysis

- The fANOVA and variance decomposition can be done efficiently in linear time if the surrogate model is a random forest [Hutter et al. 2014]



Source: [Lindauer et al. 2019]

- predicted cost is marginalized over all other hyperparameter effects
- Warning: The optimum on these curves does not have to be the global optimum across all hyperparameters

fANOVA Analysis

- How much of the variance can be explained by a hyperparameter (or combinations of hyperparamaters) marginalized over all other parameters?

Table: Exemplary analysis of PPO on cartpole

Hyperparameter	Explained Variance
Discount rate	19.3 %
Batch size	15.7 %
Learning rate	3.7 %
Likelihood ration clipping	3.4%
...	

fANOVA Analysis

- How much of the variance can be explained by a hyperparameter (or combinations of hyperparamaters) marginalized over all other parameters?

Table: Exemplary analysis of PPO on cartpole

Hyperparameter	Explained Variance
Discount rate	19.3 %
Batch size	15.7 %
Learning rate	3.7 %
Likelihood ration clipping	3.4%
...	
discount rate & batch size	10.4%
discount rate & likelihood ration clipping	4.4%
...	

Remarks on fANOVA

- Given compute higher-order interaction effects
  - Often too expensive for more than 2 or 3 dimensions

Remarks on fANOVA

- Given compute higher-order interaction effects
  - Often too expensive for more than 2 or 3 dimensions
- Implicit assumption: the surrogate model models the space fairly well

Remarks on fANOVA

- Given compute higher-order interaction effects
  - Often too expensive for more than 2 or 3 dimensions
- Implicit assumption: the surrogate model models the space fairly well
- Global analysis and local analysis of hyperparameter importance does not always agree [Biedenkapp et al. 2018]

Remarks on fANOVA

- Given compute higher-order interaction effects
  - Often too expensive for more than 2 or 3 dimensions
- Implicit assumption: the surrogate model models the space fairly well
- Global analysis and local analysis of hyperparameter importance does not always agree [Biedenkapp et al. 2018]

$\rightsquigarrow$ You should run both to get a good understanding of why an AutoML tool chose a configuration