AutoML: Introduction

Risks of AutoML

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Risks of AutoML

1. Users apply AutoML without understanding anything.
   - Users might wonder why (Auto-)ML does not perform well after they passed in poor data.

Risks of AutoML

2. Users trust AutoML too much.
   - humans might not use human reasoning skills and do not second guess machine decisions

Risks of AutoML

3. We enable non-ML experts to use ML
   without knowing the risks and consequences of ML.
   - E.g., bias in data and trained models

Risks of AutoML

4. Could result in deployment of ...

Risks of AutoML

4. Could result in deployment of ...
   - inaccurate models due to lack of understanding of statistical concepts, e.g., sampling bias, overfitting, concept drift, ...

Risks of AutoML

4. Could result in deployment of ...
   - inaccurate models due to lack of understanding of statistical concepts, e.g., sampling bias, overfitting, concept drift, ...
   - biased and unfair models due to lack of understanding ethical practices and use of features such as gender and race for predicting outcomes

Mitigating the Risks

1. Raise awareness of the risks
   - Before using AutoML, users should still have a basic understanding of how to apply ML properly

Mitigating the Risks

1. Raise awareness of the risks
   - Before using AutoML, users should still have a basic understanding of how to apply ML properly

2. Build systems that automatically warn of these risks
   - AutoML should go hand in hand in bias analysis
   - ⇝ Future research topics (as of March 2020)

Mitigating the Risks

1. Raise awareness of the risks
   - Before using AutoML, users should still have a basic understanding of how to apply ML properly
2. Build systems that automatically warn of these risks
   - AutoML should go hand in hand in bias analysis
   - ⇝ Future research topics (as of March 2020)
3. Can we automatically find ML systems that are less prone to such risks?
   - ⇝ Future research topics (as of March 2020)

AutoML: Introduction

How is Everything connected?

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

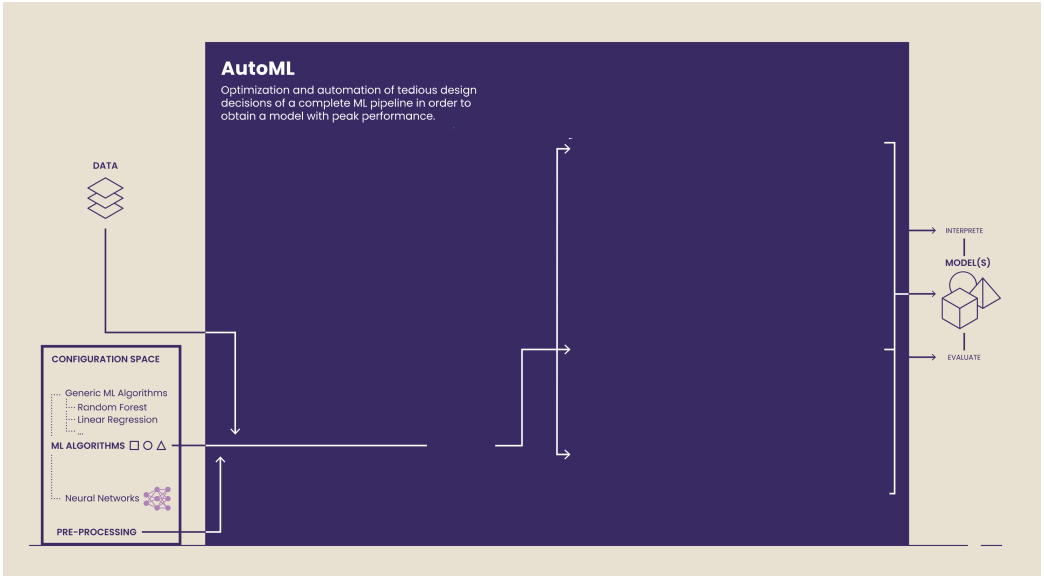
Building a Successful AutoML Tool

$\rightsquigarrow$ There is not a single way for building successful AutoML tools, but there are well established ideas.

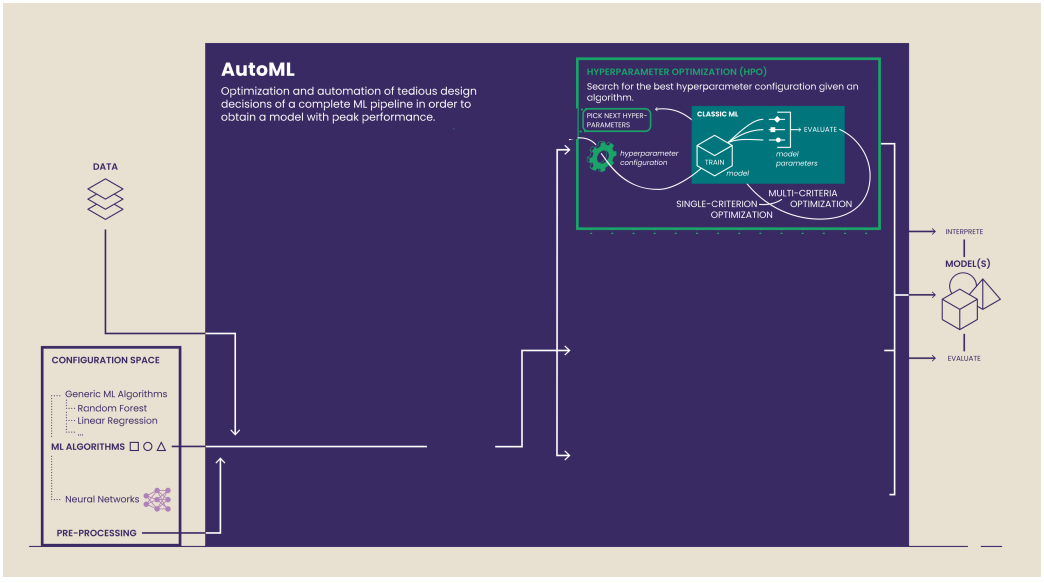
- Hyperparameter optimization
- Neural architecture search
- Model and algorithm selection
- Dynamic approaches

$\rightsquigarrow$ But how can we combine all of these?

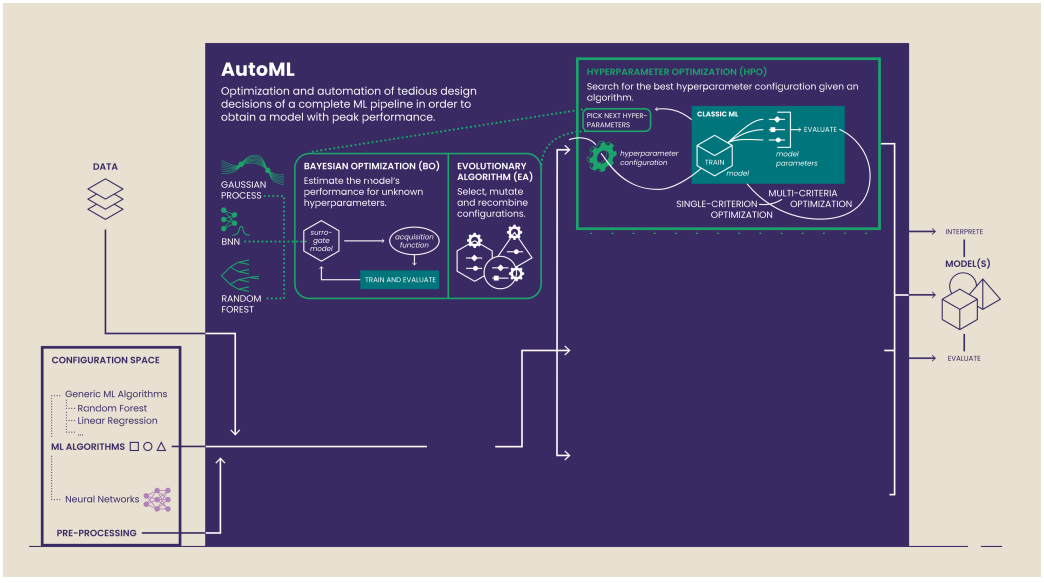
AutoML


AutoML
Optimization and automation of tedious design decisions of a complete ML pipeline in order to obtain a model with peak performance.
DATA
CONFIGURATION SPACE
Generic ML Algorithms
Random Forest
Linear Regression
...
ML ALGORITHMS
Neural Networks
PRE-PROCESSING
INTERPRETE
MODEL(S)
EVALUATE

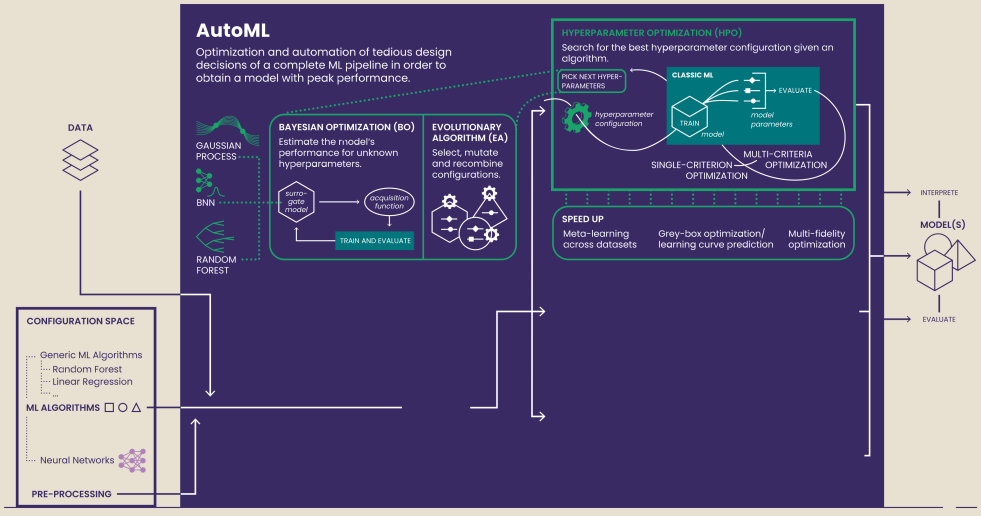
AutoML + HPO


AutoML
Optimization and automation of tedious design decisions of a complete ML pipeline in order to obtain a model with peak performance.
HYPERPARAMETER OPTIMIZATION (HPO)
Search for the best hyperparameter configuration given an algorithm.
PICK NEXT HYPER-PARAMETERS
hyperparameter configuration
CLASSIC ML
TRAIN
model
EVALUATE
model parameters
SINGLE-CRITERION OPTIMIZATION
MULTI-CRITERIA OPTIMIZATION
DATA
CONFIGURATION SPACE
Generic ML Algorithms
Random Forest
Linear Regression
...
ML ALGORITHMS
Neural Networks
PRE-PROCESSING
INTERPRETE
MODEL(S)
EVALUATE

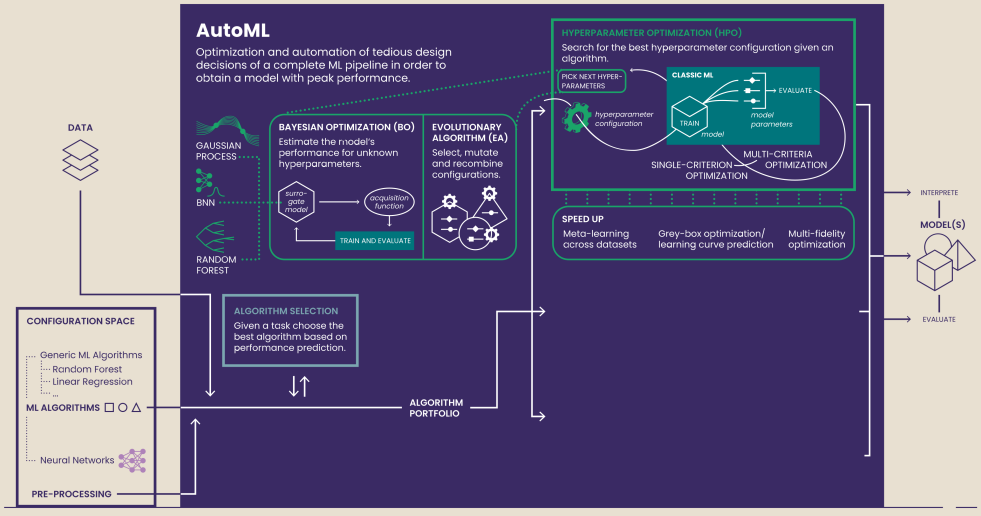
AutoML + HPO + BO



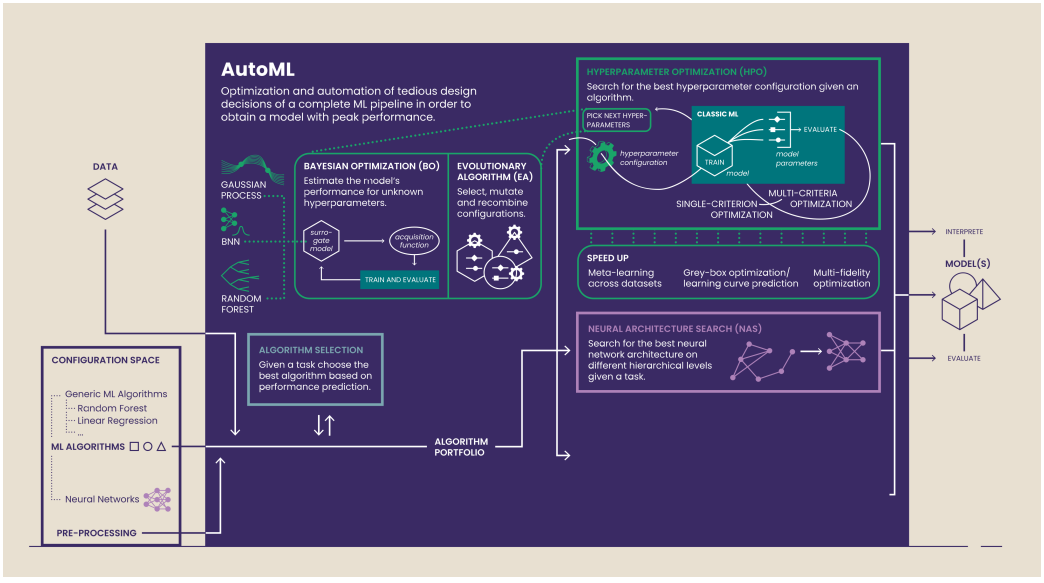
AutoML + HPO + BO + Speed up



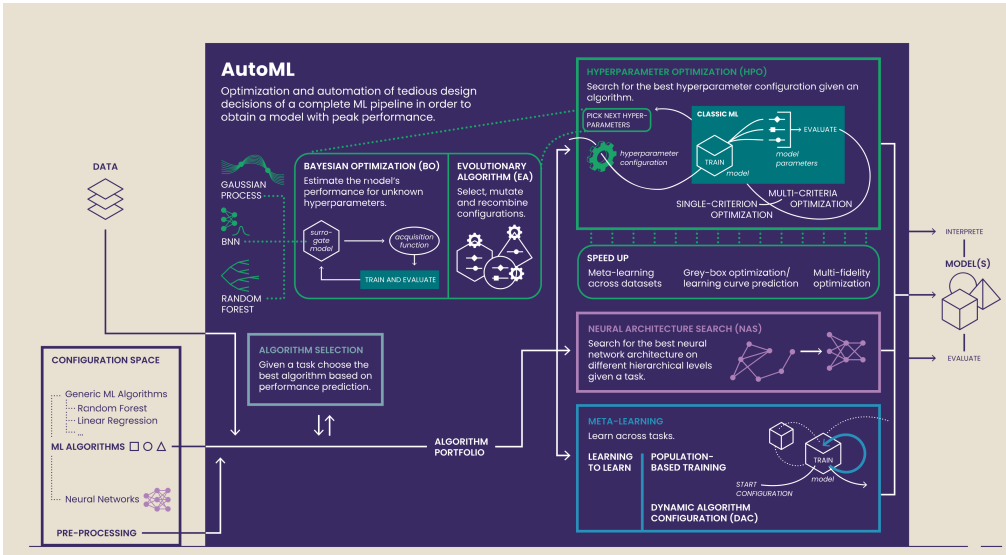
AutoML + HPO + BO + Speed up + AS



AutoML + HPO + BO + Speed up + AS + NAS



AutoML + HPO + BO + Speed up + AS + NAS + Dynamic



AutoML: Introduction

The Big Picture

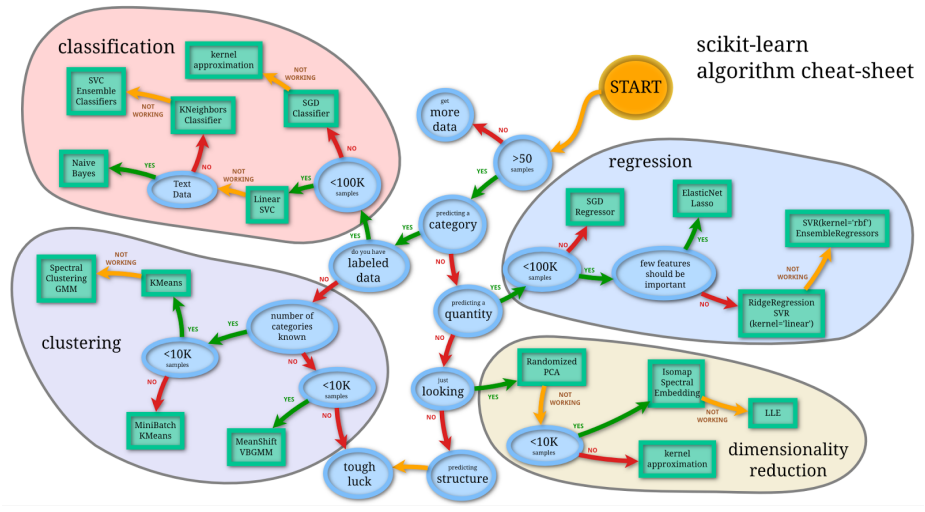
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Machine Learning

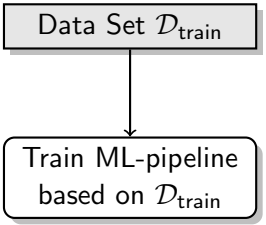
"Machine learning is the science of getting computers to act without being explicitly programmed."

by Andrew Ng

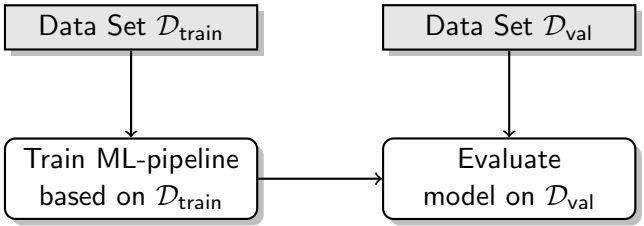
Machine Learning requires many design decisions



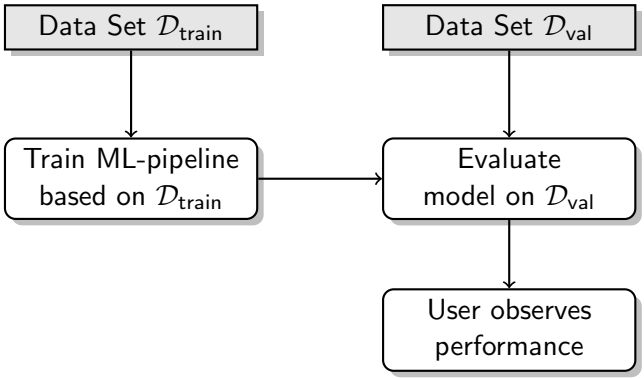
Machine Learning Workflow



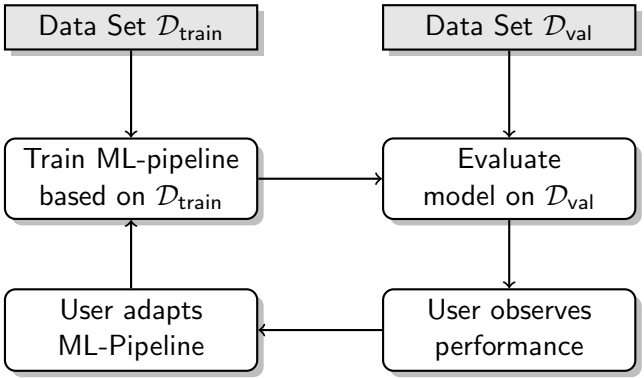
Machine Learning Workflow



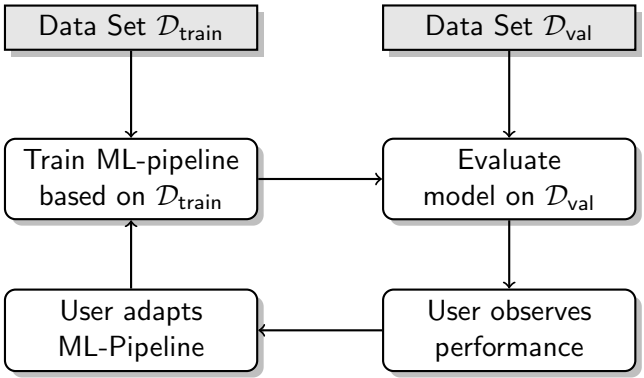
Machine Learning Workflow



Machine Learning Workflow



Machine Learning Workflow



$\rightsquigarrow$ Users indirectly teach machines how to learn.

Machine Learning does not scale up

- Basics in machine learning are not hard to grasp

Machine Learning does not scale up

- Basics in machine learning are not hard to grasp
- Achieving state-of-the-art performance is quite hard

Machine Learning does not scale up

- Basics in machine learning are not hard to grasp
- Achieving state-of-the-art performance is quite hard
- Design decisions are often not intuitive and require a lot of expertise
  - making these design decisions is a tedious and error-prone task

Machine Learning does not scale up

- Basics in machine learning are not hard to grasp
- Achieving state-of-the-art performance is quite hard
- Design decisions are often not intuitive and require a lot of expertise
  - making these design decisions is a tedious and error-prone task
- The job market for ML-experts is nearly empty

Machine Learning does not scale up

- Basics in machine learning are not hard to grasp
- Achieving state-of-the-art performance is quite hard
- Design decisions are often not intuitive and require a lot of expertise
  - making these design decisions is a tedious and error-prone task
- The job market for ML-experts is nearly empty
- Even with experts, developing new ML-applications takes time

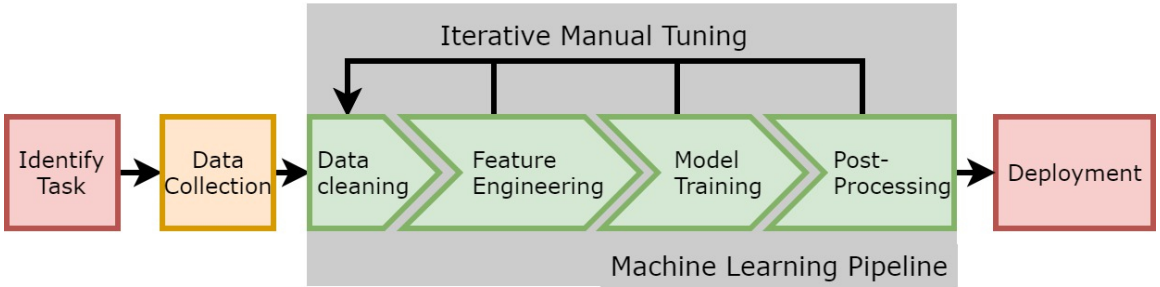
Machine Learning does not scale up

- Basics in machine learning are not hard to grasp
- Achieving state-of-the-art performance is quite hard
- Design decisions are often not intuitive and require a lot of expertise
  - making these design decisions is a tedious and error-prone task
- The job market for ML-experts is nearly empty
- Even with experts, developing new ML-applications takes time

Zoubin Ghahramani said that he often heard that:

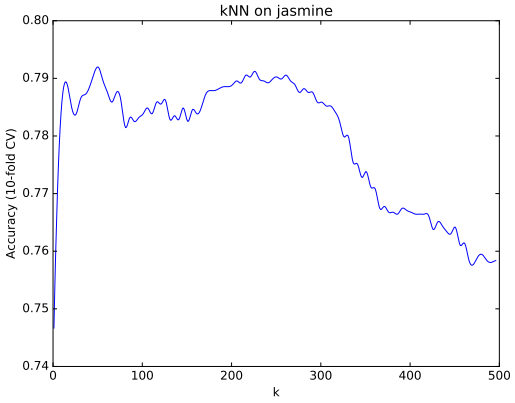
"I'd like to use machine learning, but I can't invest much time."

Why does ML development take a lot of time?



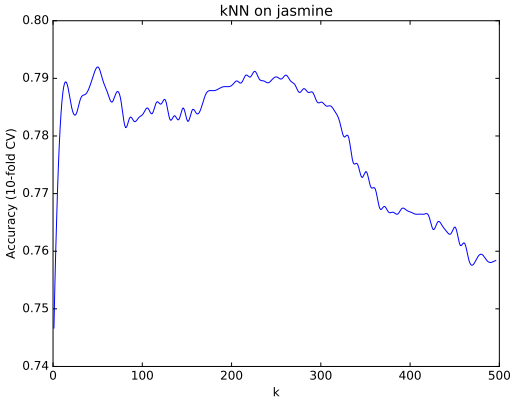
$\rightsquigarrow$ To achieve state-of-the-art performance,
this manual tuning has to be done for each new dataset again.

A Simple Example with k-NN



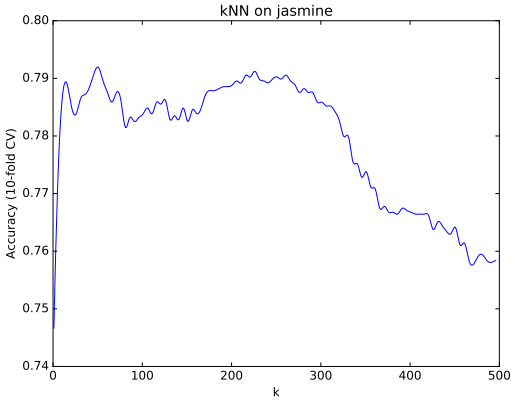
- k-nearest neighbors is one of the simplest ML algorithms

A Simple Example with k-NN



- k-nearest neighbors is one of the simplest ML algorithms
- Size of neighbourhood (k) is very important for its performance

A Simple Example with k-NN



- k-nearest neighbors is one of the simplest ML algorithms
- Size of neighbourhood (k) is very important for its performance
- The performance function depending on k is quite complex (not at all convex)

Goal of AutoML

AutoML

The goal of AutoML is to automate all parts of machine learning
(as needed) to *support* users efficiently building their ML-applications.

Goal of AutoML

AutoML

The goal of AutoML is to automate all parts of machine learning
(as needed) to *support* users efficiently building their ML-applications.

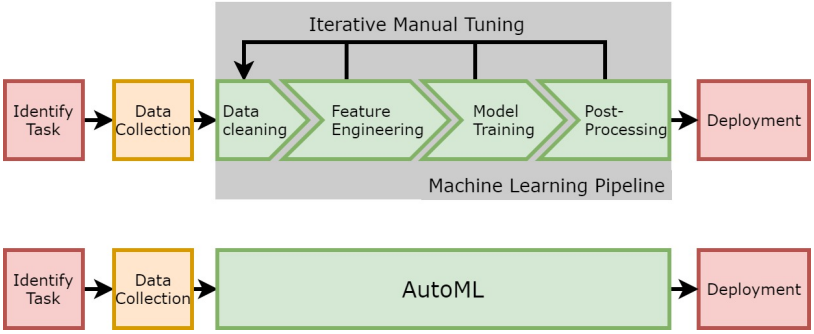
Informal Definition: AutoML System

Given

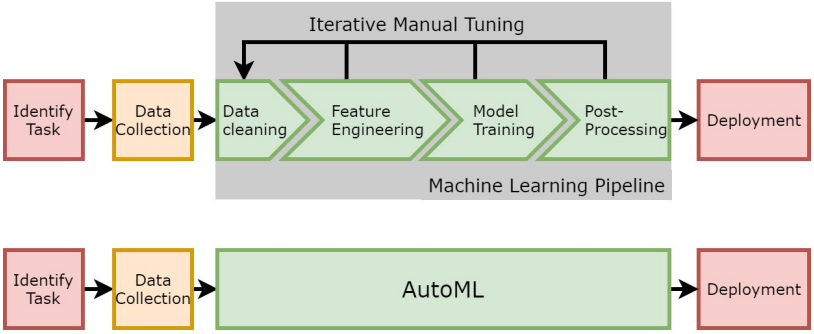
- a dataset
- a task (e.g., regression or classification)
- a cost metric (e.g., accuracy or RMSE)

an AutoML system automatically determines the approach that performs best for this particular application.

ML vs AutoML


Iterative Manual Tuning
Identify Task
Data Collection
Data cleaning
Feature Engineering
Model Training
Post-Processing
Deployment
Machine Learning Pipeline
Identify Task
Data Collection
AutoML
Deployment

ML vs AutoML



With AutoML, we ...

- support ML users
- improve the efficiency of developping new ML applications
- reduce the required ML-expertise
- might achieve better performance than developers w/o AutoML

AutoML in Research

AutoML enables:

1. more efficient research
   - AutoML has shown on subproblems to outperform human experts

AutoML in Research

AutoML enables:

1. more efficient research
   - AutoML has shown on subproblems to outperform human experts
2. more systematic research
   - humans tend to be unsystematic which leads to errors

AutoML in Research

AutoML enables:

1. more efficient research
   - AutoML has shown on subproblems to outperform human experts
2. more systematic research
   - humans tend to be unsystematic which leads to errors
3. more reproducible research
   - human's unsystematic approaches cannot be reproduced, but AutoML is systematic

AutoML in Research

AutoML enables:

1. more efficient research
   - AutoML has shown on subproblems to outperform human experts
2. more systematic research
   - humans tend to be unsystematic which leads to errors
3. more reproducible research
   - human's unsystematic approaches cannot be reproduced, but AutoML is systematic
4. broader use of ML also in other disciplines
   - ML should not be limited to computer scientists;
   - the most amazing applications of ML are often done by either interdisciplinary teams or even non-computer scientists

Challenges in AutoML

1. Each dataset potentially require different optimal ML-designs
   - Design decisions have to be made for each dataset again

Challenges in AutoML

1. Each dataset potentially require different optimal ML-designs
   - Design decisions have to be made for each dataset again
2. Training of a single ML model can be quite expensive (e.g., hours, days or weeks)
   - often, we cannot try many design decisions

Challenges in AutoML

1. Each dataset potentially require different optimal ML-designs
   - Design decisions have to be made for each dataset again
2. Training of a single ML model can be quite expensive (e.g., hours, days or weeks)
   - often, we cannot try many design decisions
3. the mathematical relation between design decisions and performance is (often) unknown
   - gradient-based optimization is not directly possible

Challenges in AutoML

1. Each dataset potentially require different optimal ML-designs
   - Design decisions have to be made for each dataset again
2. Training of a single ML model can be quite expensive (e.g., hours, days or weeks)
   - often, we cannot try many design decisions
3. the mathematical relation between design decisions and performance is (often) unknown
   - gradient-based optimization is not directly possible
4. optimization in highly complex spaces
   - incl. categorical choices, continuous parameters, conditional dependencies

AutoML: Introduction

Overview of AutoML Problems

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Objectives of AutoML

Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset

Objectives of AutoML

Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset
- AUC-ROC

Objectives of AutoML

Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset
- AUC-ROC
- precision & recall
- ...

Objectives of AutoML

Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset
- AUC-ROC
- precision & recall
- ...
- Memory consumption
- Inference time
- ...

Objectives of AutoML

Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset
- AUC-ROC
- precision & recall
- ...
- Memory consumption
- Inference time
- ...

$\rightsquigarrow$ AutoML optimizes an arbitrary cost function, denoted as c.

Objectives of AutoML

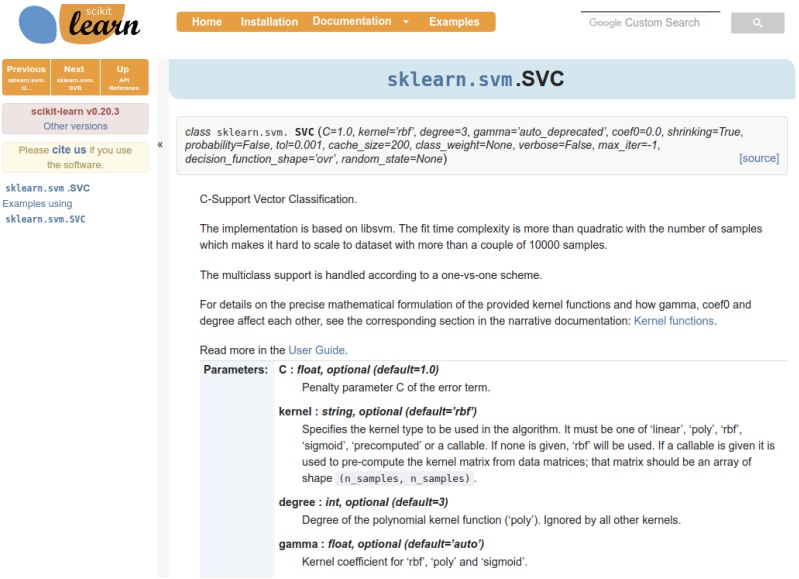
Different metrics to measure the success of ML and AutoML:

- Accuracy on a validation dataset
- AUC-ROC
- precision & recall
- ...
- Memory consumption
- Inference time
- ...

$\rightsquigarrow$ AutoML optimizes an arbitrary cost function, denoted as c.

$\rightsquigarrow$ $c(\cdot)$ can also return several cost metrics

Hyperparameters of an SVM



Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,

Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$

Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

- $\arg\min$ returns a set of optimal points of a given function. It suffices to find one element of this set and thus we use $\in$ instead of $=$.

Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

- $\arg\min$ returns a set of optimal points of a given function. It suffices to find one element of this set and thus we use $\in$ instead of $=$.
- Sometimes, we want to optimize for different metrics, instead of one
  - $\rightsquigarrow$ multi-objective optimization and Pareto fronts

Choosing an Algorithm

- Over the years, many ML-algorithms were proposed

Choosing an Algorithm

- Over the years, many ML-algorithms were proposed
- Most of these still have a reason for existence

Choosing an Algorithm

- Over the years, many ML-algorithms were proposed
- Most of these still have a reason for existence
- Examples for classification:
  - logistic regression
  - k-nearest neighbor
  - naïve Bayes
  - support vector machine
  - decision tree
  - random forest
  - gradient boosting
  - multi-layer perceptron
  - residual networks
  - ...

Choosing an Algorithm

- Over the years, many ML-algorithms were proposed
- Most of these still have a reason for existence
- Examples for classification:
  - logistic regression
  - k-nearest neighbor
  - naïve Bayes
  - support vector machine
  - decision tree
  - random forest
  - gradient boosting
  - multi-layer perceptron
  - residual networks
  - ...
- studied 179 classifiers on 121 datasets [Fernández-Delgado et al. 2015]

Choosing an Algorithm

- Over the years, many ML-algorithms were proposed
- Most of these still have a reason for existence
- Examples for classification:
  - logistic regression
  - k-nearest neighbor
  - naïve Bayes
  - support vector machine
  - decision tree
  - random forest
  - gradient boosting
  - multi-layer perceptron
  - residual networks
  - ...
- studied 179 classifiers on 121 datasets [Fernández-Delgado et al. 2015]
- In practice, we actually want to jointly choose the best ML-algorithm and its hyperparameters

CASH: Combined Algorithm Selection and Hyperparameter Optimization

Definition

Let

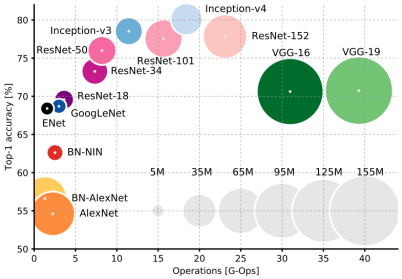
- $\mathbf{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k\}$ be a set of algorithms (a.k.a. portfolio)
- $\mathbf{\Lambda}$ be a set of hyperparameters of each machine learning algorithm $\mathcal{A}_i$
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

we want to find the best combination of algorithm $\mathcal{A} \in \mathbf{A}$ and its hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ minimizing:

$$(\mathcal{A}^*, \boldsymbol{\lambda}^*) \in \underset{\mathcal{A} \in \mathbf{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}}{\arg\min}\, c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Architectures of Neural Networks

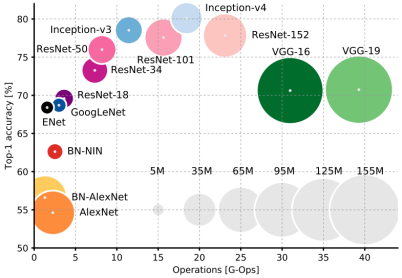
- Many architecture were proposed
- Differences in
  - Depth
  - Resolution
  - Width
  - Operators
  - Connections
  - ...



on Imagenet [Canzian et al. 2017]

Architectures of Neural Networks

- Many architecture were proposed
- Differences in
  - Depth
  - Resolution
  - Width
  - Operators
  - Connections
  - ...

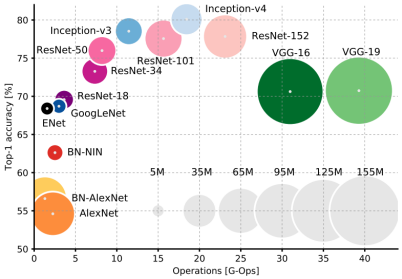


on Imagenet [Canzian et al. 2017]

- Already on a single dataset such as ImageNet, it is not obvious which architeture to choose

Architectures of Neural Networks

- Many architecture were proposed
- Differences in
  - Depth
  - Resolution
  - Width
  - Operators
  - Connections
  - ...

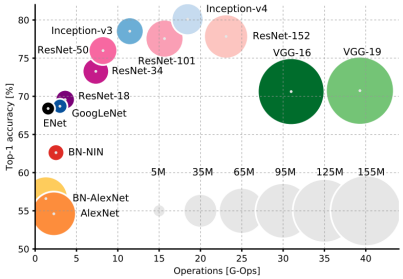


on Imagenet [Canzian et al. 2017]

- Already on a single dataset such as ImageNet, it is not obvious which architeture to choose
- For other datasets, you might need different architectures to achieve top-performance

Architectures of Neural Networks

- Many architecture were proposed
- Differences in
  - Depth
  - Resolution
  - Width
  - Operators
  - Connections
  - ...



on Imagenet [Canzian et al. 2017]

- Already on a single dataset such as ImageNet, it is not obvious which architeture to choose
- For other datasets, you might need different architectures to achieve top-performance
  - For similar datasets, you might use scaled versions of known architectures (e.g., CIFAR10 and Imagenet)

Neural Architecture Search (NAS)

Definition

Let

- $\boldsymbol{\lambda}$ be an architecture for a deep neural network N with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $N_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *neural architecture search (NAS)* problem is to find an architecture that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Neural Architecture Search (NAS)

Definition

Let

- $\boldsymbol{\lambda}$ be an architecture for a deep neural network N with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $N_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *neural architecture search (NAS)* problem is to find an architecture that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remark:

- very similar to the HPO definition

Neural Architecture Search (NAS)

Definition

Let

- $\boldsymbol{\lambda}$ be an architecture for a deep neural network N with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $N_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *neural architecture search (NAS)* problem is to find an architecture that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remark:

- very similar to the HPO definition
- In practice, you want jointly optimize HPO and NAS [Zela et al. 2018]

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution
- A different view: Can we learn from data and thus predict an algorithm/ hyperparameter configuration/ neural architecture?

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution
- A different view: Can we learn from data and thus predict an algorithm/ hyperparameter configuration/ neural architecture?

Definition

Let

- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution
- A different view: Can we learn from data and thus predict an algorithm/ hyperparameter configuration/ neural architecture?

Definition

Let

- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- $\mathbf{P}$ a portfolio of algorithms $\mathcal{A} \in \mathbf{P}$, and

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution
- A different view: Can we learn from data and thus predict an algorithm/ hyperparameter configuration/ neural architecture?

Definition

Let

- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- $\mathbf{P}$ a portfolio of algorithms $\mathcal{A} \in \mathbf{P}$, and
- $c : \mathbf{P} \times \mathbf{D} \to \mathbb{R}$ be a cost metric

Per-Instance Algorithm Selection

- For solving the HPO, CASH and NAS problem, we use some kind of optimization approach to search for a solution
- A different view: Can we learn from data and thus predict an algorithm/ hyperparameter configuration/ neural architecture?

Definition

Let

- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- $\mathbf{P}$ a portfolio of algorithms $\mathcal{A} \in \mathbf{P}$, and
- $c : \mathbf{P} \times \mathbf{D} \to \mathbb{R}$ be a cost metric

the *per-instance algorithm selection problem* is to obtain a mapping $s : \mathcal{D} \mapsto \mathcal{A}$ such that

$$\arg\min_{s} \int_{\mathbf{D}} c(s(\mathcal{D}), \mathcal{D}) p(\mathcal{D}) \, \mathrm{d}\mathcal{D}$$

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Definition

Let

- λ be a hyperparameter configuration of an algorithm $\mathcal{A}$,

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Definition

Let

- $\boldsymbol{\lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Definition

Let

- $\boldsymbol{\lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- s_t be a state description of $\mathcal{A}$ solving $\mathcal{D}$ at time point t,

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Definition

Let

- $\boldsymbol{\lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- s_t be a state description of $\mathcal{A}$ solving $\mathcal{D}$ at time point t,
- $c : \boldsymbol{\Pi} \times \mathbf{D} \to \mathbb{R}$ be a cost metric assessing the cost of a conf. policy $\pi \in \Pi$ on $\mathcal{D} \in \mathbf{D}$

Dynamic Algorithm Configuration

- So far, we assume that an algorithm runs with some given settings
- However, some settings, such as learning rate, have to be adapted over time

Definition

Let

- $\boldsymbol{\lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over datasets $\mathcal{D} \in \mathbf{D}$,
- s_t be a state description of $\mathcal{A}$ solving $\mathcal{D}$ at time point t,
- $c : \mathbf{\Pi} \times \mathbf{D} \to \mathbb{R}$ be a cost metric assessing the cost of a conf. policy $\pi \in \Pi$ on $\mathcal{D} \in \mathbf{D}$

the *dynamic algorithm configuration problem (DAC)* is to obtain a configuration policy $\pi^* : s_t \times \mathcal{D} \mapsto \boldsymbol{\lambda}$ by optimizing its cost across a distribution of datasets:

$$\pi^* \in \arg\min_{\pi \in \mathbf{\Pi}} \int_{\mathbf{D}} p(\mathcal{D}) c(\pi, \mathcal{D}) \, \mathrm{d}\mathcal{D}$$

Summary

HPO	Search for the best hyperparameter configuration of a ML algorithm
CASH	Search for the best combination of algorithm and hyperparameter configuration
NAS	Search for the architecture of neural network
Selection	Predict the best algorithm (and its hyperparameter configuration)
DAC	Predict the best hyperparameter configuration for an algorithm state at a given time point

AutoML: Introduction

The Team

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Bernd Bischl

- Professor at
  the Ludwig-Maximilians-University of Munich (Germany)
- Head of the statistical learning and data science group
- Co-Director of Munich Center for Machine Learning
- Founder of MLR (Machine Learning in R)
- Co-Founder of `www.openml.org`
- Co-Founder and advisory board member of COSEAL
  (Configuration and Selection of Algorithms)

Frank Hutter

- Professor at
  the University of Freiburg (Germany)
- Head of the machine learning lab
- Won 1st and 2nd international AutoML challenge
- Co-author of popular AutoML tools Auto-WEKA,
  Auto-sklearn, Auto-PyTorch
- Co-founder and co-organizer of AutoML workshop series at
  ICML (since 2014)
- Co-Editor of book on *Automated Machine Learning:
  Methods, Systems, Challenges*
- Co-Head of `automl.org`

Lars Kotthoff

- Professor at
  the University of Wyoming (USA)
- Head of the Meta-Algorithmics, Learning and Large-Scale Empirical Testing lab (MALLET) and director of the Artificially Intelligent Manufacturing Center (AIM)
- Developer of MLR (Machine Learning in R)
- Organizer of competitions on algorithm selection
- Co-Editor of book on *Automated Machine Learning: Methods, Systems, Challenges*

Marius Lindauer

- Professor at
  the Leibniz University of Hannover (Germany)
- Head of the machine learning group
- Won 1st and 2nd international AutoML challenge
- Co-author of popular AutoML tools Auto-sklearn & Auto-PyTorch
- Co-Head of `automl.org`
- Co-Founder and advisory board member of COSEAL (Configuration and Selection of Algorithms)

Janek Thomas

- Group Lead for AutoML & XAI at Fraunhofer IIS
- Phd from the Ludwig-Maximilians-University Munich (Germany)
- Developer of MLR, the amlb AutoML benchmark and autoxgboost
- Co-Head of the Munich Datageeks e.V.

Joaquin Vanschoren

- Professor at
  Eindhoven University of Technology (Netherlands)
- Machine learning, Meta-Learning
- Co-Founder of `www.openml.org`
- Co-Editor of book on *Automated Machine Learning: Methods, Systems, Challenges*

Supported by

- Archit Bansal
- André Biedenkapp
- Omid Charrakh
- Difan Deng
- Katharina Eggensperger
- Matthias Feurer
- Maciej Janowski
- Julia Moosbauer
- Jakob Richter
- Julien Siems
- Guri Zabergja
- Arber Zela

... and many more