Speedup Techniques for Hyperparameter Optimization

Meta-Learning

Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Introduction

- Learning essentially never stops:
  - Many models are periodically re-fit to track changes in the data
  - Many models are re-fit to perform well on new tasks
- The best hyperparameter configuration tends to remain quite stable across tasks

Introduction

- Learning essentially never stops:
  - Many models are periodically re-fit to track changes in the data
  - Many models are re-fit to perform well on new tasks
- The best hyperparameter configuration tends to remain quite stable across tasks

For a good introduction to meta-learning in general, see [AutoML Book: Chapter 2]

Problem Statement

Given:

- a set of prior tasks: $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$,
- a set of new tasks: $t_{\text{new}} \in \mathcal{T}$,

Problem Statement

Given:

- a set of prior tasks: $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$,
- a set of new tasks: $t_{\text{new}} \in \mathcal{T}$,
- a set of learning algorithms, fully defined by $\theta_i \in \Theta$

Problem Statement

Given:

- a set of prior tasks: $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$,
- a set of new tasks: $t_{\text{new}} \in \mathcal{T}$,
- a set of learning algorithms, fully defined by $\theta_i \in \Theta$
- a set of prior evaluations $\mathcal{D}_{\text{meta}}$ on $t_j \in \mathcal{T}_{\text{meta}}$
- a set of evaluations $\mathcal{D}_{\text{new}}$ on new task t_{new}

Problem Statement

Given:

- a set of prior tasks: $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$,
- a set of new tasks: $t_{\text{new}} \in \mathcal{T}$,
- a set of learning algorithms, fully defined by $\theta_i \in \Theta$
- a set of prior evaluations $\mathcal{D}_{\text{meta}}$ on $t_j \in \mathcal{T}_{\text{meta}}$
- a set of evaluations $\mathcal{D}_{\text{new}}$ on new task t_{new}

Goal of meta-learning:

- use meta-data $\mathcal{D}_{\text{meta}}$ to choose $\theta_i \in \Theta$ for t_{new} better than only based on $\mathcal{D}_{\text{new}}$.

[adapted from AutoML Book: Chapter 2]

The Role of Meta-Features

- We can often extract additional characteristics for each task, called meta-features
- Each task t_j can be described by a vector of K meta-features:

$$m(t_j) = (m_{j,1}, \ldots, m_{j,K})$$

The Role of Meta-Features

- We can often extract additional characteristics for each task, called meta-features
- Each task t_j can be described by a vector of K meta-features:

$$m(t_j) = (m_{j,1}, \ldots, m_{j,K})$$

- This vector can be used to define a similarity measure between two tasks
  - e.g., calculating the Euclidean distance between $m(t_i)$ and $m(t_j)$
  - Based on similarity, we can transfer information from the most similar tasks to new task t_{new}

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes
- Statistical - characterize the data via descriptive statistics:
  - e.g., average or standard deviation of features, or their correlation with the labels

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes
- Statistical - characterize the data via descriptive statistics:
  - e.g., average or standard deviation of features, or their correlation with the labels
- Information-theoretic - measure the class entropy in the data
  - capture the amount of information in the data

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes
- Statistical - characterize the data via descriptive statistics:
  - e.g., average or standard deviation of features, or their correlation with the labels
- Information-theoretic - measure the class entropy in the data
  - capture the amount of information in the data
- Model-based - extracted from a model induced using the training data
  - these are often based on properties of decision tree models
  - e.g., number of leaves, number of nodes, shape of the tree

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes
- Statistical - characterize the data via descriptive statistics:
  - e.g., average or standard deviation of features, or their correlation with the labels
- Information-theoretic - measure the class entropy in the data
  - capture the amount of information in the data
- Model-based - extracted from a model induced using the training data
  - these are often based on properties of decision tree models
  - e.g., number of leaves, number of nodes, shape of the tree
- Landmarking - computed by running several fast ML algorithms on the dataset
  - e.g., is fast algorithm A better than fast algorithm B on this dataset?
  - this can capture different properties of the dataset, e.g., linear separability

Overview of Meta-Features in Machine Learning

- Simple - easily extracted from the data, describe the basic dataset structure
  - e.g., number of features, data points or classes
- Statistical - characterize the data via descriptive statistics:
  - e.g., average or standard deviation of features, or their correlation with the labels
- Information-theoretic - measure the class entropy in the data
  - capture the amount of information in the data
- Model-based - extracted from a model induced using the training data
  - these are often based on properties of decision tree models
  - e.g., number of leaves, number of nodes, shape of the tree
- Landmarking - computed by running several fast ML algorithms on the dataset
  - e.g., is fast algorithm A better than fast algorithm B on this dataset?
  - this can capture different properties of the dataset, e.g., linear separability
- Others - not included in the previous groups
  - e.g., time related measures, clustering and distance-based measures

Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)

Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)
- Can we learn from meta-data $\mathcal{D}_{\text{meta}}$ how to initialize HPO?

Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)
- Can we learn from meta-data $\mathcal{D}_{\text{meta}}$ how to initialize HPO?
- Note: just a single default configuration often does not perform great on a new dataset
  - Otherwise there would be no point in HPO

Meta-Learning for HPO Approach 2: Model-Warmstarting

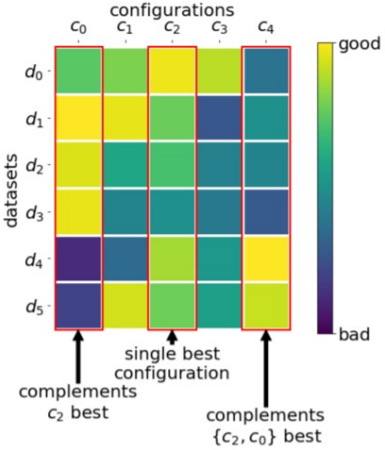
- Many HPO methods use a predictive model (e.g., Bayesian optimization)
- By running HPO on different datasets, we learn something about the search landscape
  - E.g., what are bad regions of the configuration space in general

Meta-Learning for HPO Approach 2: Model-Warmstarting

- Many HPO methods use a predictive model (e.g., Bayesian optimization)
- By running HPO on different datasets, we learn something about the search landscape
  - E.g., what are bad regions of the configuration space in general
- Given: n predictive models $\hat{c}_{\mathcal{D}_i} : \mathbf{\Lambda} \rightarrow \mathbb{R}$ from HPO on $\mathcal{T}_{\text{meta}}$
- How can we use these $\hat{c}_{\mathcal{D}_i}$ to speed up HPO?

Meta-Learning for HPO Approach 3: Task-independent Recommendations

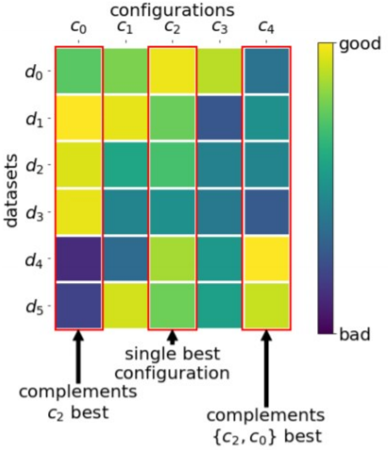
- *Idea:* learn a sorted list of defaults
- *Method:* mostly greedy on $\mathcal{T}_{\text{meta}}$
- *Results:* surprisingly strong, better than Bayesian Optimization



[Wistuba et al. 2015a], [Feurer et al. 2018], [Pfisterer et al. 2018]

Meta-Learning for HPO Approach 3: Task-independent Recommendations

- *Idea:* learn a sorted list of defaults
- *Method:* mostly greedy on $\mathcal{T}_{\text{meta}}$
- *Results:* surprisingly strong, better than Bayesian Optimization



Advantages

- Easy to share and use
- Strong anytime performance
- Embarrassingly parallel

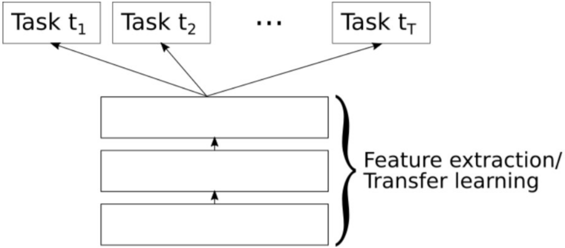
Disadvantages

- Not adaptive

[Wistuba et al. 2015a], [Feurer et al. 2018], [Pfisterer et al. 2018]

Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

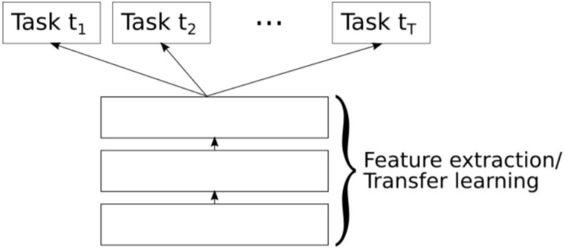
- Jointly train a “deep” neural network on all tasks



[Perrone et al. 2018]

Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

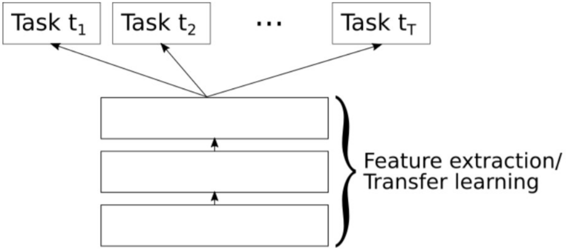
- Jointly train a "deep" neural network on all tasks
  - Have a separate output layer (head) for each task
  - Each head is a Bayesian linear regression (recall DNGO)



[Perrone et al. 2018]

Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

- Jointly train a "deep" neural network on all tasks
  - Have a separate output layer (head) for each task
  - Each head is a Bayesian linear regression (recall DNGO)
- This uses meta-learning for feature extraction on the hyperparameter configurations



[Perrone et al. 2018]

Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - Use $\mathcal{D}_{\text{meta}}$ to learn a mapping from $\mathcal{D}_{\text{new}}$ to the next configuration $\boldsymbol{\lambda}$ to evaluate
  - This mapping can be a (recurrent) neural net $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \boldsymbol{\lambda}$ parameterized by weights ϕ

Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - Use $\mathcal{D}_{\text{meta}}$ to learn a mapping from $\mathcal{D}_{\text{new}}$ to the next configuration $\boldsymbol{\lambda}$ to evaluate
  - This mapping can be a (recurrent) neural net $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \boldsymbol{\lambda}$ parameterized by weights ϕ
  - This mapping NN_ϕ constitutes a blackbox optimization algorithm

Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - Use $\mathcal{D}_{\text{meta}}$ to learn a mapping from $\mathcal{D}_{\text{new}}$ to the next configuration $\boldsymbol{\lambda}$ to evaluate
  - This mapping can be a (recurrent) neural net $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \boldsymbol{\lambda}$ parameterized by weights ϕ
  - This mapping NN_ϕ constitutes a blackbox optimization algorithm

- Existing approaches for learning a blackbox optimizer
  - Gradient descent on ϕ [Chen et al. 2017]
    - Simplest technique, but requires backpropagation through the optimization trace
    - This also requires the blackbox functions f used for training to be differentiable

Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - Use $\mathcal{D}_{\text{meta}}$ to learn a mapping from $\mathcal{D}_{\text{new}}$ to the next configuration $\boldsymbol{\lambda}$ to evaluate
  - This mapping can be a (recurrent) neural net $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \boldsymbol{\lambda}$ parameterized by weights ϕ
  - This mapping NN_ϕ constitutes a blackbox optimization algorithm
- Existing approaches for learning a blackbox optimizer
  - Gradient descent on ϕ [Chen et al. 2017]
    - Simplest technique, but requires backpropagation through the optimization trace
    - This also requires the blackbox functions f used for training to be differentiable
  - Reinforcement learning [Li & Malik, 2016]
    - Can be harder to get to work, but does not require differentiable f

Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - Depending on the problem at hand, you might need a different acquisition function

Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - Depending on the problem at hand, you might need a different acquisition function

- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2020]

Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - Depending on the problem at hand, you might need a different acquisition function

- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2020]

- Two options:
  - Only depend on predicted mean and variance: $u_\phi(\boldsymbol{\lambda}) = u_\phi(\mu_t(\boldsymbol{\lambda}), \sigma_t(\boldsymbol{\lambda}))$
    - This allows to learn a general acquisition function

Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - Depending on the problem at hand, you might need a different acquisition function

- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2020]

- Two options:
  - Only depend on predicted mean and variance: $u_\phi(\boldsymbol{\lambda}) = u_\phi(\mu_t(\boldsymbol{\lambda}), \sigma_t(\boldsymbol{\lambda}))$
    - This allows to learn a general acquisition function
  - Also depend on the $\boldsymbol{\lambda}$ value: $u_\phi(\boldsymbol{\lambda}) = u_\phi(\mu_t(\boldsymbol{\lambda}), \sigma_t(\boldsymbol{\lambda}), \boldsymbol{\lambda})$
    - This allows to fine-tune to the characteristics of $\mathcal{D}_{\text{meta}}$ (e.g., avoid poor parts of the space)

Questions to Answer for Yourself / Discuss with Friends

- Repetition. What are the different kinds of meta-features which can be used to describe machine learning datasets?
- Repetition. List all the different ways of using the meta data for HPO you recall
- Discussion. In the various meta-learning approaches, what will happen if all prior tasks are dissimilar to the target task?