AutoML: Dynamic Configuration & Learning

Dynamic Configuration

Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Iterative Optimization Heuristics

- Many iterative heuristics in algorithms are dynamic and adaptive
  1. the algorithm's behavior changes over time
  2. the algorithm's behavior changes based on internal statistics

Iterative Optimization Heuristics

- Many iterative heuristics in algorithms are dynamic and adaptive
  1. the algorithm's behavior changes over time
  2. the algorithm's behavior changes based on internal statistics

- These heuristics might control other hyperparameters of the algorithms

Iterative Optimization Heuristics

- Many iterative heuristics in algorithms are dynamic and adaptive
  1. the algorithm's behavior changes over time
  2. the algorithm's behavior changes based on internal statistics

- These heuristics might control other hyperparameters of the algorithms
- Example: learning rate schedules for training DNNs
  1. exponential decaying learning rate: based on number of iterations, learning rate decreases

Iterative Optimization Heuristics

- Many iterative heuristics in algorithms are dynamic and adaptive
  1. the algorithm's behavior changes over time
  2. the algorithm's behavior changes based on internal statistics

- These heuristics might control other hyperparameters of the algorithms
- Example: learning rate schedules for training DNNs
  1. exponential decaying learning rate: based on number of iterations, learning rate decreases
  2. Reduce learning rate on plateaus: if the learning stagnates for some time, the learning rate is decreased by a factor

Iterative Optimization Heuristics

- Many iterative heuristics in algorithms are dynamic and adaptive
  1. the algorithm's behavior changes over time
  2. the algorithm's behavior changes based on internal statistics
- These heuristics might control other hyperparameters of the algorithms
- Example: learning rate schedules for training DNNs
  1. exponential decaying learning rate: based on number of iterations, learning rate decreases
  2. Reduce learning rate on plateaus: if the learning stagnates for some time, the learning rate is decreased by a factor
- other examples: restart probability of search, mutation rate of evolutionary algorithms, ...

Parametrization of Learning Rate Schedules

- How can we parameterize learning rate schedules?
  1. exponential decaying learning rate:
     - initial learning rate
     - minimal learning rate
     - multiplicative factor

Parametrization of Learning Rate Schedules

- How can we parameterize learning rate schedules?
  1. exponential decaying learning rate:
     - initial learning rate
     - minimal learning rate
     - multiplicative factor
  2. Reduce learning rate on plateaus:
     - patience (in number of epochs)
     - patience threshold
     - decreasing factor
     - cool-down break (in number of epochs)

Parametrization of Learning Rate Schedules

- How can we parameterize learning rate schedules?
  1. exponential decaying learning rate:
     - initial learning rate
     - minimal learning rate
     - multiplicative factor
  2. Reduce learning rate on plateaus:
     - patience (in number of epochs)
     - patience threshold
     - decreasing factor
     - cool-down break (in number of epochs)

$\rightsquigarrow$ Many hyperparameters only to control a single hyperparameter

Parametrization of Learning Rate Schedules

- How can we parameterize learning rate schedules?
  1. exponential decaying learning rate:
     - initial learning rate
     - minimal learning rate
     - multiplicative factor
  2. Reduce learning rate on plateaus:
     - patience (in number of epochs)
     - patience threshold
     - decreasing factor
     - cool-down break (in number of epochs)

$\leadsto$ Many hyperparameters only to control a single hyperparameter

- Still not guaranteed that optimal setting of e.g. learning rate schedules will lead to optimal learning behavior
  - Learning rate schedules are only heuristics

Dynamic Algorithm Configuration

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

Definition

Let

- $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,

Dynamic Algorithm Configuration

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

Definition

Let

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

Dynamic Algorithm Configuration

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

Definition

Let

- $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over meta 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 assumed that an algorithm runs with static settings
- However, settings, such as learning rate, have to be adapted over time

Definition

Let

- $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over meta 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 control policy $\pi \in \Pi$ on $\mathcal{D} \in \mathbf{D}$

Dynamic Algorithm Configuration

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

Definition

Let

- $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ be a hyperparameter configuration of an algorithm $\mathcal{A}$,
- $p(\mathcal{D})$ be a probability distribution over meta 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 control policy $\pi \in \Pi$ on $\mathcal{D} \in \mathbf{D}$

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

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

Dynamic Algorithm Configuration as Contextual MDP [Biedenkapp et al. 2020]

State $s^{(t)}$ are described by statistics gathered in the algorithm run

Dynamic Algorithm Configuration as Contextual MDP [Biedenkapp et al. 2020]

State $s^{(t)}$ are described by statistics gathered in the algorithm run

Action $a^{(t)}$ change hyperparameters according to some control policy π

Dynamic Algorithm Configuration as Contextual MDP [Biedenkapp et al. 2020]

State $s^{(t)}$ are described by statistics gathered in the algorithm run

Action $a^{(t)}$ change hyperparameters according to some control policy π

Transition run the algorithm from state $s^{(t)}$ to $s^{(t+1)}$ for a "short" moment by using the hyperparameters defined by $a^{(t)}$

Dynamic Algorithm Configuration as Contextual MDP [Biedenkapp et al. 2020]

State $s^{(t)}$ are described by statistics gathered in the algorithm run

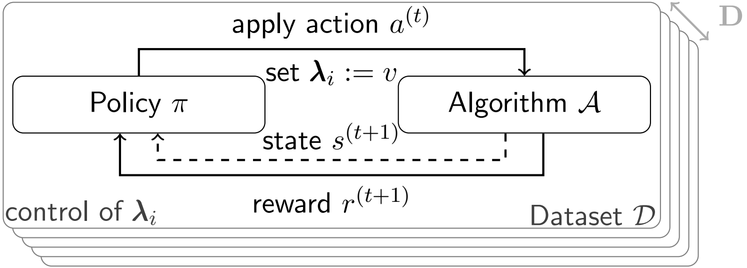
Action $a^{(t)}$ change hyperparameters according to some control policy π

Transition run the algorithm from state $s^{(t)}$ to $s^{(t+1)}$ for a "short" moment by using the hyperparameters defined by $a^{(t)}$

Reward $r^{(t)}$ Return your current solution quality (or an approximation)

Dynamic Algorithm Configuration as Contextual MDP [Biedenkapp et al. 2020]

State $s^{(t)}$	are described by statistics gathered in the algorithm run
Action $a^{(t)}$	change hyperparameters according to some control policy π
Transition	run the algorithm from state $s^{(t)}$ to $s^{(t+1)}$ for a "short" moment by using the hyperparameters defined by $a^{(t)}$
Reward $r^{(t)}$	Return your current solution quality (or an approximation)
Context $\mathcal{D}$	A given dataset (or task)



Solving Dynamic Algorithm Configuration

Solve unknown MDP by using reinforcement learning (RL):

$$\mathcal{V}_{\mathcal{D}}^{\pi}(s^{(t)}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{\mathcal{D}}^{(t+k+1)} \middle| s^{(t)} = s\right]$$

Solving Dynamic Algorithm Configuration

Solve unknown MDP by using reinforcement learning (RL):

$$\mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t)}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{\mathcal{D}}^{(t+k+1)} | s^{(t)} = s\right]$$

$$= \mathbb{E}\left[r_{\mathcal{D}}^{(t+1)} + \gamma \mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t+1)}) | s^{(t+1)} \sim \mathcal{T}_{\mathcal{D}}(s^{(t)}, \pi(s^{(t)}))\right]$$

Solving Dynamic Algorithm Configuration

Solve unknown MDP by using reinforcement learning (RL):

$$\mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t)}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{\mathcal{D}}^{(t+k+1)} | s^{(t)} = s\right]$$

$$= \mathbb{E}\left[r_{\mathcal{D}}^{(t+1)} + \gamma \mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t+1)}) | s^{(t+1)} \sim \mathcal{T}_{\mathcal{D}}(s^{(t)}, \pi(s^{(t)}))\right]$$

$$\pi^* \in \arg\max_{\pi \in \mathbf{\Pi}} \int_{\mathbf{D}} p(\mathcal{D}) \int_{\mathcal{S}^{(0)}} \Pr(s^{(0)}) \cdot \mathcal{V}^{\pi}_{\mathcal{D}}(s^{(0)})\, \mathrm{d}s^{(0)}\, \mathrm{d}\mathcal{D}$$

Solving Dynamic Algorithm Configuration

Solve unknown MDP by using reinforcement learning (RL):

$$\mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t)}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{\mathcal{D}}^{(t+k+1)} | s^{(t)} = s\right]$$

$$= \mathbb{E}\left[r_{\mathcal{D}}^{(t+1)} + \gamma \mathcal{V}^{\pi}_{\mathcal{D}}(s^{(t+1)}) | s^{(t+1)} \sim \mathcal{T}_{\mathcal{D}}(s^{(t)}, \pi(s^{(t)}))\right]$$

$$\pi^* \in \operatorname*{arg\,max}_{\pi \in \mathbf{\Pi}} \int_{\mathbf{D}} p(\mathcal{D}) \int_{\mathcal{S}^{(0)}} \Pr(s^{(0)}) \cdot \mathcal{V}^{\pi}_{\mathcal{D}}(s^{(0)}) \, \mathrm{d}s^{(0)} \, \mathrm{d}\mathcal{D}$$

$\rightsquigarrow$ equivalent to Dynamic Algorithm Configuration definition

Dynamic Algorithm Configuration across Datasets [Biedenkapp et al. 2020]

- Challenge: Evaluating a policies on all datasets is often not feasible

Dynamic Algorithm Configuration across Datasets [Biedenkapp et al. 2020]

- Challenge: Evaluating a policies on all datasets is often not feasible
- Curriculum learning [Bengio et al. 2009] showed that we should have a curriculum of tasks we tackle

Dynamic Algorithm Configuration across Datasets [Biedenkapp et al. 2020]

- Challenge: Evaluating a policies on all datasets is often not feasible
- Curriculum learning [Bengio et al. 2009] showed that we should have a curriculum of tasks we tackle
- Self-paced learning [Kumar et al. 2010] tries to automatically find such as a curriculum
  - Focus on "easy" tasks where the agent can improve most:

Dynamic Algorithm Configuration across Datasets [Biedenkapp et al. 2020]

- Challenge: Evaluating a policies on all datasets is often not feasible
- Curriculum learning [Bengio et al. 2009] showed that we should have a curriculum of tasks we tackle
- Self-paced learning [Kumar et al. 2010] tries to automatically find such as a curriculum
  - Focus on "easy" tasks where the agent can improve most:

$$\max_{\pi,\mathbf{v}} \mathcal{C}(\pi, \mathbf{v}, K) = \sum_{i=1}^{|\mathbf{D}|} \mathbf{v}_i \mathcal{R}_i(\pi) - \frac{1}{K} \sum_{i=1}^{|\mathbf{D}|} \mathbf{v}_i$$

with θ being the agent's policy parameters and
$\mathbf{v}$ being a masking vector for choosing the tasks at hand.