AutoML: Beyond AutoML

Structured Procastination

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Structured Procrastination [Kleinberg et al. 2017]

Idea

- incumbent driven methods (such as aggressive racing with adaptive capping) provide no theoretical guarantees about runtime

Structured Procrastination [Kleinberg et al. 2017]

Idea

- incumbent driven methods (such as aggressive racing with adaptive capping) provide no theoretical guarantees about runtime
- task: for a fix set of configuration, identify the one with the best average runtime
- instead of top-down capping, use bottom up capping

Structured Procrastination [Kleinberg et al. 2017]

Idea

- incumbent driven methods (such as aggressive racing with adaptive capping) provide no theoretical guarantees about runtime
- task: for a fix set of configuration, identify the one with the best average runtime
- instead of top-down capping, use bottom up capping
- start with a minimal cap-time and increase it step by step

Structured Procrastination [Kleinberg et al. 2017]

Idea

- incumbent driven methods (such as aggressive racing with adaptive capping) provide no theoretical guarantees about runtime
- task: for a fix set of configuration, identify the one with the best average runtime
- instead of top-down capping, use bottom up capping
- start with a minimal cap-time and increase it step by step
- unsuccessful runs (with too small cap-time) are procrastinated to later

$\leadsto$ worst-case runtime guarantees

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 1 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 2 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while b *remains* **do**

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 3 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$

Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`

initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`

while b *remains* **do**

  determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 4 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while b *remains* **do**
    determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;
    get first element $(i^{(k)}, \kappa)$ from $Q_{\hat{\boldsymbol{\lambda}}}$;

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 5 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while b *remains* **do**
  determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;
  get first element $(i^{(k)}, \kappa)$ from $Q_{\hat{\boldsymbol{\lambda}}}$;
  Run $\hat{\boldsymbol{\lambda}}$ on $i^{(k)}$ capped at κ;
  if *terminates* **then**
    $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) :=$ t;

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 6 Structured Procrastination

Input : finite (small) set of configurations $\boldsymbol{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while *b remains* **do**
- determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;
- get first element $(i^{(k)}, \kappa)$ from $Q_{\hat{\boldsymbol{\lambda}}}$;
- Run $\hat{\boldsymbol{\lambda}}$ on $i^{(k)}$ capped at κ;
- **if** *terminates* **then**
  - $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) :=$ t;
- **else**
  - $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) := \kappa$;
  - Insert $(i^{(k)}, 2 \cdot \kappa)$ at the end of $Q_{\hat{\boldsymbol{\lambda}}}$;

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 7 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \dots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while *b remains* **do**
    determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;
    get first element $(i^{(k)}, \kappa)$ from $Q_{\hat{\boldsymbol{\lambda}}}$;
    Run $\hat{\boldsymbol{\lambda}}$ on $i^{(k)}$ capped at κ;
    if *terminates* **then**
        $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) :=$ t;
    else
        $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) := \kappa$;
        Insert $(i^{(k)}, 2 \cdot \kappa)$ at the end of $Q_{\hat{\boldsymbol{\lambda}}}$;
    Replenish queue $Q_{\hat{\boldsymbol{\lambda}}}$ if too small;

Structured Procrastination: Outline [Kleinberg et al. 2017]

Algorithm 8 Structured Procrastination

Input : finite (small) set of configurations $\mathbf{\Lambda}$, minimal cap-time κ_0, sequence of instances $i^{(1)}, \ldots, i^{(N)}$
Output : best incumbent configuration $\hat{\boldsymbol{\lambda}}$

for each $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ initialize a queue $Q_{\boldsymbol{\lambda}}$ with entries $(i^{(k)}, \kappa_0)$; `// small queue in the beginning`
initialize a look-up table $R(\boldsymbol{\lambda}, i) = 0$; `// optimistic runtime estimate`
while b *remains* **do**
  determine the best $\hat{\boldsymbol{\lambda}}$ according to $R(\boldsymbol{\lambda}, \cdot)$;
  get first element $(i^{(k)}, \kappa)$ from $Q_{\hat{\boldsymbol{\lambda}}}$;
  Run $\hat{\boldsymbol{\lambda}}$ on $i^{(k)}$ capped at κ;
  if *terminates* **then**
    $R(\hat{\boldsymbol{\lambda}}, i^{(k)})$:= t;
  else
    $R(\hat{\boldsymbol{\lambda}}, i^{(k)}) := \kappa$;
    Insert $(i^{(k)}, 2 \cdot \kappa)$ at the end of $Q_{\hat{\boldsymbol{\lambda}}}$;
  Replenish queue $Q_{\hat{\boldsymbol{\lambda}}}$ if too small;
return $\hat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} \sum_{k=1}^{N} R(\boldsymbol{\lambda}, i^{(k)})$

Extensions

- We can derive theoretical optimality guarantees with structured procrastination (SP)

Extensions

- We can derive theoretical optimality guarantees with structured procrastination (SP)
- In practice, plain SP is rather slow and requires the setting of some hyperparameters

Extensions

- We can derive theoretical optimality guarantees with structured procrastination (SP)
- In practice, plain SP is rather slow and requires the setting of some hyperparameters
- Several extensions and similar ideas:
  - [Kleinberg et al. 2019]
  - [Weisz et al. 2018]
  - [Weisz et al. 2019]