

AutoML: Dynamic Configuration & Learning

Learning to Adjust Learning Rates

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

- Optimization of a function:

$$\theta \in \arg \min F(\mathbf{X}; \theta)$$

where \mathbf{X} is an input matrix and f is parameterized by θ .

- Optimization of a function:

$$\theta \in \arg \min F(\mathbf{X}; \theta)$$

where \mathbf{X} is an input matrix and f is parameterized by θ .

$$F(\mathbf{X}; \theta) = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}^{(i)}; \theta)$$

Learning Step Size Policies [Daniel et al. 2016]

- **Idea:** Learn the hyperparameters of the weight update (short notation)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla F(\theta^{(t)})$$

$$\nabla F(\theta^{(t)}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\theta^{(t)})$$

Learning Step Size Policies [Daniel et al. 2016]

- **Idea:** Learn the hyperparameters of the weight update (short notation)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla F(\theta^{(t)})$$

$$\nabla F(\theta^{(t)}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\theta^{(t)})$$

- For SGD, this would be for example the learning rate α

Learning Step Size Policies [Daniel et al. 2016]

- **Idea:** Learn the hyperparameters of the weight update (short notation)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla F(\theta^{(t)})$$

$$\nabla F(\theta^{(t)}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\theta^{(t)})$$

- For SGD, this would be for example the learning rate α
- **Note (i):** α have to be adapted in the course of the training
 - ▶ similar to learning rate schedules (e.g., cosine annealing)

Learning Step Size Policies [Daniel et al. 2016]

- **Idea:** Learn the hyperparameters of the weight update (short notation)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla F(\theta^{(t)})$$

$$\nabla F(\theta^{(t)}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\theta^{(t)})$$

- For SGD, this would be for example the learning rate α
- **Note (i):** α have to be adapted in the course of the training
 - ▶ similar to learning rate schedules (e.g., cosine annealing)
- **Note(ii):** later we denote the learnt hyperparameters as λ

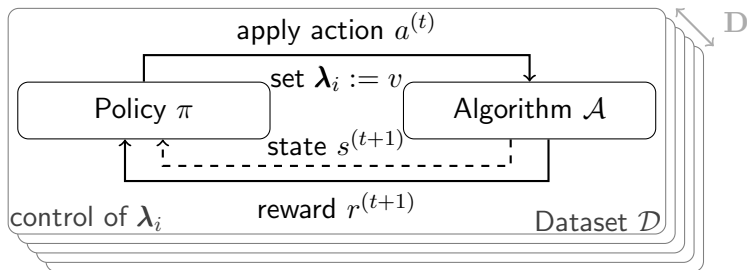
Learning Step Size Policies [Daniel et al. 2016]

- **Idea:** Learn the hyperparameters of the weight update (short notation)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla F(\theta^{(t)})$$
$$\nabla F(\theta^{(t)}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\theta^{(t)})$$

- For SGD, this would be for example the learning rate α
- **Note (i):** α have to be adapted in the course of the training
 - ▶ similar to learning rate schedules (e.g., cosine annealing)
- **Note(ii):** later we denote the learnt hyperparameters as λ
- **Idea:** Use reinforcement learning to learn a policy $\pi : s \mapsto a$ to control the learning rate (or other adaptive hyperparameters)

Recap: Reinforcement Learning for Dynamic Algorithm Configuration



To apply that, we need to define:

- 1 State description
- 2 Action space
- 3 Reward function

Predictive change in function value:

$$s_1 = \log \left(\text{Var}(\Delta \tilde{f}_i) \right)$$

$$\Delta \tilde{f}_i = \tilde{f}(\mathbf{x}^{(i)}; \theta + \delta\theta) - f(\mathbf{x}^{(i)}; \theta)$$

where $\tilde{f}(\mathbf{x}^{(i)}; \theta + \delta\theta)$ is done by a first order Taylor expansion

Predictive change in function value:

$$s_1 = \log \left(\text{Var}(\Delta \tilde{f}_i) \right)$$

$$\Delta \tilde{f}_i = \tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta) - f(\mathbf{x}^{(i)}; \theta)$$

where $\tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta)$ is done by a first order Taylor expansion

Disagreement of function values:

$$s_2 = \log \left(\text{Var}(f(\mathbf{x}^{(i)}; \theta)) \right)$$

Predictive change in function value:

$$s_1 = \log \left(\text{Var}(\Delta \tilde{f}_i) \right)$$

$$\Delta \tilde{f}_i = \tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta) - f(\mathbf{x}^{(i)}; \theta)$$

where $\tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta)$ is done by a first order Taylor expansion

Disagreement of function values:

$$s_2 = \log \left(\text{Var}(f(\mathbf{x}^{(i)}; \theta)) \right)$$

Discounted Average (smoothing noise from mini-batches):

$$\hat{s}_i \leftarrow \gamma \hat{s}_i + (1 - \gamma) s_i$$

Predictive change in function value:

$$s_1 = \log \left(\text{Var}(\Delta \tilde{f}_i) \right)$$

$$\Delta \tilde{f}_i = \tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta) - f(\mathbf{x}^{(i)}; \theta)$$

where $\tilde{f}(\mathbf{x}^{(i)}; \theta + \delta \theta)$ is done by a first order Taylor expansion

Disagreement of function values:

$$s_2 = \log \left(\text{Var}(f(\mathbf{x}^{(i)}; \theta)) \right)$$

Discounted Average (smoothing noise from mini-batches):

$$\hat{s}_i \leftarrow \gamma \hat{s}_i + (1 - \gamma) s_i$$

Uncertainty Estimate (noise level):

$$s_{K+i} \leftarrow \gamma s_{K+i} + (1 - \gamma) (s_i - \hat{s}_i)^2$$

Reward (average loss improvement over time):

$$r = \frac{1}{T-1} \sum_{t=2}^T \left(\log(L^{(t-1)}) - \log(L^{(t)}) \right)$$

RL for Step Size Policies: Learning [Daniel et al. 2016]

Reward (average loss improvement over time):

$$r = \frac{1}{T-1} \sum_{t=2}^T \left(\log(L^{(t-1)}) - \log(L^{(t)}) \right)$$

Optimal Policy:

$$\pi^*(\lambda \mid s) \in \arg \max_{\pi} \int \int p(s) \pi(\boldsymbol{\lambda} \mid s) r(\boldsymbol{\lambda}, s) \, ds \, d\boldsymbol{\lambda}$$

RL for Step Size Policies: Learning [Daniel et al. 2016]

Reward (average loss improvement over time):

$$r = \frac{1}{T-1} \sum_{t=2}^T \left(\log(L^{(t-1)}) - \log(L^{(t)}) \right)$$

Optimal Policy:

$$\pi^*(\lambda \mid s) \in \arg \max_{\pi} \int \int p(s) \pi(\boldsymbol{\lambda} \mid s) r(\boldsymbol{\lambda}, s) \mathrm{d} s \mathrm{d} \boldsymbol{\lambda}$$

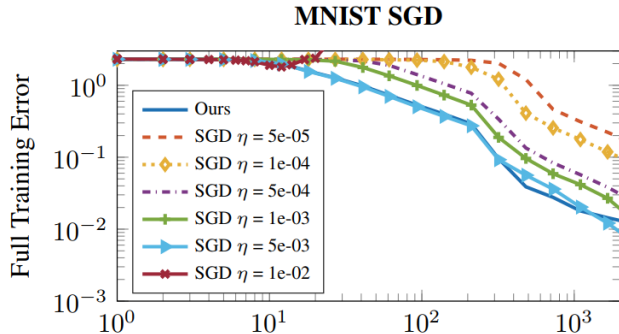
- can be learnt for example via Relative Entropy Policy Search (REPS) [Peter et al. 2010]

RL for Step Size Policies: Training [Daniel et al. 2016]

- Goal: obtain robust policies,
i.e., good performance for many different DNN architectures
 - ↪ Sample architectures e.g., with different numbers of filters and layers
 - ↪ (Sub-)Sample dataset
 - ↪ Sample number of optimization steps

RL for Step Size Policies: Training [Daniel et al. 2016]

- Goal: obtain robust policies,
i.e., good performance for many different DNN architectures
 - ↪ Sample architectures e.g., with different numbers of filters and layers
 - ↪ (Sub-)Sample dataset
 - ↪ Sample number of optimization steps



"Ours" refers to the approach by [Daniel et al. 2016] and η is the learning rate